دانشگاه شمال

مدارهای واسط

تهیه و تنظیم:
فانی

مهر ۱۳۹۱

مقدمه

قبل از بستن مدارها، موارد زیر را به دقت مطالعه کنید.

۱- قبل از بستن هر مدار، مبحث تئوری مربوط به آن مدار به طور کامل مطالعه شود. در این دستور کار سعی شده که مطالب تئوری که به آن نیاز دارید توضیح داده شود. برای اطلاعات بیشتر و تکمیلی می‌توانید به Datasheet ها و کتب مربوطه رجوع کنید.

۲- هرگاه جایی در نقشه‌های مدارها، به خروجی‌ای برخوردید که مشخص نشده که به جایی وصل شده باشد، مثل خروجی‌های مشخص شده با NC (No Connection) با آنها برخورد کنید؛ یعنی آن را آزاد رها کنید (به جایی وصل نکنید) ولی ورودی یک تراشه را نمی‌توان به جایی وصل نکرد چون این کار باعث اثرپذیری زیاد مدار نسبت به نویز می‌شود.

۳- هیچ گاه خروجی یک تراشه (پایه‌ای که صرفاً به عنوان خروجی مشخص شده) را به خروجی تراشه دیگر یا V_{CC} و GND یا هر منبع دیگر وصل نکنید. این کار به تراشه صدمه می‌رساند. (با این توضیحات متوجه می‌شویم که مهم است بدانیم کدام یک از خطوط مربوط به خروجی تراشه و کدام یک مربوط به ورودی آن می‌شود و این هم بر می‌گردد به این که چقدر روی عملکرد مدار تسلط داشته باشید. در نتیجه باز هم اهمیت مطالعه قسمت تئوری جلوه‌گر می‌شود)

۴- هیچ گاه LED ها (یا اصولاً هر قطعه‌ای که در آن LED به کار رفته مثل Seven Segment ها و Matrix LED ها) را به طور مستقیم به خروجی تراشه یا منبع تغذیه وصل نکنید، بلکه آن را با یک مقاومت بین 100 تا 200 اهمی (که باعث محدود کردن جریان می‌شود) سری نموده، وصل کنید.

۵- قسمت مهم کار در بستن مدارها عیب‌یابیشان است. وصل کردن مدار، به تنهایی چندان مهم نیست؛ بلکه مهم، این است که بتوانید در اسرع زمان آن را عیب‌یابی کنید. معمولاً مداری که وصل می‌کنید، دارای عیب‌هایی است و کم‌تر مداری بدون اشتباه وصل می‌شود. پس باید یاد بگیرید که بتوانید هر مداری را (حتی مداری که خودتان در بستن آن نقشی نداشته‌اید)، عیب‌یابی کنید. برای

این کار لازم است هر مداری را که می‌بندید و با هر تراشه‌ای که آشنا می‌شوید، طرز کار آن را به دقت یاد بگیرید تا بتوانید هنگامی که در مدارهای بزرگ‌تر، این تراشه‌ها را درکنار هم به کار می‌برید، با پی‌گیری ولتاژهای خروجی‌های قطعات مدار، به مشکل کار پی ببرید.

۶- دقت کنید باز کردن و دوباره بستن مداری که درست کار نمی‌کند، راه حل درستی نیست. بکوشید خودتان مشکل کار را پیدا کنید.

۷- به محدوده ولتاژهای '1' منطقی و '0' منطقی دقت کنید تا هنگامی که در عیب‌یابی مدارها به سطح ولتاژی غیر معمول برخوردید، بدانید که ممکن است در آن جا مشکلی وجود داشته باشد. وقتی می‌خواهید ولتاژ خروجی تراشه‌ای را بخوانید، اگر خروجی آن به LED متصل است، اول LED را جدا کنید و بعد ولتاژ را بخوانید.

۸- تا مطمئن نشده اید که یک تراشه یا یک قطعه خراب است، اقدام به تعویض آن نکنید. بهترین راه برای اطمینان از خراب بودن تراشه‌ها یا قطعات، تست کردن جداگانه آنها روی یک برد بورد سالم است.

۹- سعی کنید ولتاژهایی که با مالتی‌متر می‌خوانید، از روی پایه‌های فلزی تراشه‌ها باشد (یعنی ترمینال فلزی مالتی‌متر را (ممکن است که برای ظرافت بیشتر، به یک تکه سیم نیاز داشته باشید که محکم به آن وصل شود) به دقت به پایه‌ای که می‌خواهید مقدارش را بخوانید، تماس دهید نه به سوراخ‌های بورد)؛ چون احتمال خرابی بورد یا قطع بودن سیم‌ها وجود دارد. حتماً می‌دانید که برای خواندن مقدار ولتاژ جایی توسط مالتی‌متر، یک سر (ترمینال) مالتی‌متر را باید به زمین وصل کرد و ترمینال دیگر را به جایی که می‌خواهید ولتاژ آن را اندازه بگیرید. پس برای راحتی کار می‌توانید اول کار یک ترمینال مالتی‌متر را به طور ثابت به زمین (GND) مدار وصل کنید.

۱۰- همواره پیش از شروع کار، با مالتی‌متر چک کنید که اصولاً ولتاژ تغذیه به مدار و تراشه‌ها و قطعات می‌رسد یا خیر. با چک کردن پایه‌های مربوط به تغذیه تراشه‌ها می‌توانید قطعی‌های احتمالی سیم‌ها یا خرابی بورد (در راه رساندن ولتاژ تغذیه) را بررسی کنید. گاهی اوقات مداری که می‌بندید

دارای اشکالی است که روی ولتاژ تغذیه اثر می‌گذارد و حتی گاهی این ولتاژ را به حدود صفر می‌رساند. در این صورت باید در مدار دنبال جایی بگردید که خط V_{CC} ناخواسته به زمین (GND) وصل شده باشد. اگر مدار پیچیده است و پیدا کردن چنین موقعیتی مشکل است (در صورت استفاده از بردبورد) کافی است تک تک سیم‌های مدار (از سیم‌های تغذیه شروع کنید) را از جای خود جدا کنید و بعد به جای خود بر گردانید تا این که به سیمی برسید که با جدا کردن آن، ولتاژ تغذیه به مقدار اصلی خود برگردد. با این کار متوجه می‌شوید این سیم مشکل را به وجود آورده است و باید بررسی کنید که اشکال کار در وصل کردن این سیم چه بوده است.

۱۱- در صورتی که از بردبورد برای بستن مدارها استفاده می‌کنید، سیم‌ها را از روی تراشه‌ها (یا مقاومت‌ها، خازن‌ها یا مواردی مانند این‌ها) رد نکنید. چون در این صورت اگر آن تراشه خراب باشد جابه‌جایی آن مشکل خواهد بود و همچنین رد کردن قطعات با پایه‌های لخت از روی تراشه‌ها یا از روی هم احتمال اتصال ناخواسته آن‌ها را به هم و به تراشه افزایش می‌دهد.

۱۲- هرگز مطمئن نباشید مداری که بسته‌اید عاری از خطا است و دلیل کار نکردن آن خرابی اجزا است. اشتباه ممکن است همه جا و به دست هر کسی (حتی ماهرترین افراد) به وجود آید. باز هم تذکر داده می‌شود که بستن مدار به تنهایی کافی نیست، باید بتوانید مداری که بسته‌اید (خودتان یا دیگران) را به بهترین نحو و در کوتاه‌ترین زمان عیب‌یابی کنید و این کار جز با تمرین و ممارست زیاد ممکن نمی‌شود.

۱۳- FPGA ها، اصولاً تراشه‌های حساسی هستند. پس در حین کار کردن با آن کمال دقت را به عمل آورید تا به تراشه آسیبی نرسد. سعی کنید تا وقتی مطمئن نشدید که مدارتان درست است، منبع تغذیه را به آن وصل نکنید. به ویژه در جایی که علاوه بر ولتاژ کاری FPGA (ولتاژ 5^V یا 3.3^V)، با ولتاژهای بالاتری هم سر و کار دارید، بیش‌تر دقت کنید.

۱۴- برای اطمینان، وقتی می‌خواهید به FPGA یا تراشه‌های دیگر، مقدار بدهید، از یک مقاومت (حدود 1^K) در ورودی استفاده کنید (با ورودی سری کنید).

۱۵- شکل پایه‌های بعضی وسایل و قطعاتی که در این دستور کار شرح داده شـده، مربـوط بـه یـک سری وسایل و قطعات خاص می‌باشند و لزوماً جنبه عام ندارنـد. ممکـن اسـت اجزائـی کـه شـما در اختیار دارید کمی با موارد این دستور کار فرق کنند که باید به آن دقت کنید.

با آرزوی موفقیت شما

فانی، پاییز ۱۳۹۱

فهرست

۱- معرفی بورد آموزشی و اطلاعات پایه ۸

۲- اتصال موتور پله‌ای به FPGA ۶۰

۳- اتصال TSL230 (سنسور نور) به FPGA ۶۴

۴- دور شمار موتور ۶۷

۵- شمارشگر افراد با لیزر ۷۰

۶- آلارم ۷۴

۷- اتصال صفحه کلید ماتریسی به FPGA ۸۷

۸- تولید موسیقی ۹۲

۹- اتصال LCD به FPGA ۹۴

۱۰- اتصال LCD گرافیکی به FPGA ۲

۱۱- قفل رمزی ۹

۱۲- بازی حدس عدد ۱۸

۱۳- اتصال 8870 به FPGA برای دریافت اطلاعات تن تلفن ۳۱

۱۴- رقص نور ۳۹

۱۵- تابلو روان ۴۲

۱۶- اتصال 8255 (PPI) به FPGA ۴۷

۱۷- اتصال مبدل دیجیتال به آنالوگ DAC08 به FPGA ۵۲

۱۸- اتصال مبدل آنالوگ به دیجیتال ADC0804 به FPGA ۶۰

۱۹- اندازهگیری دما با مبدل آنالوگ به دیجیتال ۶۳

۲۰- ولت‌متر ۷۰

۲۱- مالتی متر ۷۸

۲۲- اتصال touch ۸۹

۲۳- اتصال 8254 (تایمر قابل برنامه‌ریزی) به FPGA ۱۰۳
۲۴- تولید موسیقی با استفاده از تراشه 8254 ..... ۱۱۳
۲۵- ارتباط سریال توسط FPGA ۱۲۲
۲۶- انتقال اطلاعات بصورت مادون قرمز با استفاده از FPGA ۱۲۷
۲۷- انتقال اطلاعات بصورت رادیویی با استفاده از FPGA ۱۳۳
۲۸- اتصال تراشه‌ی touch و adc آن ۱۳۸
۲۹- اتصال مبدل دیجیتال به آنالوگ AD7564 ..... ۱۵۰
۳۰- اتصال مبدل دیجیتال به آنالوگ AD667 ..... ۱۶۰
۳۱- اتصال مبدل آنالوگ به دیجیتال ad976 ..... ۱۶۴
۳۲- pwm ۱۷۲
۳۳- logic probe ۱۷۹
۳۴- کی پد و PPI و موتور پله ای ۱۸۵
۳۵- اتصال تلویزیون به FPGA و نمایش روی آن ۱۹۰
۳۶- اتصال تلویزیون به FPGA و فرستادن متن به آن ۱۹۸
۳۷- اتصال مانیتور به FPGA ۲۳۲
۳۸- اتصال کارت SD ۲۴۱
۳۹- ضرب بوث ۲۷۱
۴۰- دمو برای بورد آموزشی ۲۸۵
۴۱- دموی ورودی برای بورد آموزشی ۲۹۸
۴۲- مانیتور برای بورد آموزشی ۳۰۸
۴۳- کی بورد PS/2 برای بورد آموزشی ۳۱۹
۴۴- موس PS/2 برای بورد آموزشی ۳۲۳
۴۵- ارتباط سریال با کامپیوتر برای بورد آموزشی ۳۵۹
۴۶- اتصال تلویزیون برای بورد آموزشی ۳۶۴

۱- معرفی بورد آموزشی و اطلاعات پایه

امکانات تراشه‌ی Spartan II XC2S200 PQ208:

تعداد سلول‌های منطقی (Logic Cells): 5,292

تعداد گیت‌ها (System Gates): 200,000

تعداد CLB ها: 1,176

تعداد ورودی/خروجی‌های در اختیار کاربر: 140

امکانات بورد AMOL-2:

بوردی مبتنی بر آموزش دستگاه‌های جانبی پایه که مناسب یادگیری نحوه‌ی راه‌اندازی دستگاه‌های مختلف می‌باشد، چون برای آنها نیاز به بستن سخت‌افزار جداگانه نمی‌باشد و در نتیجه یاد گیرنده درگیر مشکلات احتمالی در اتصال سخت‌افزار نمی‌شود. همچنین پورت‌های آزاد این بورد، امکان اتصال دستگاه‌های متنوع دیگر را برای کاربر فراهم می‌سازد.

۳۷ پورت مجزای ورودی/خروجی برای کاربر که به طرق مختلف زیر و برای راحتی کار کاربر در نظر گرفته شده

- یک پورت باکسی ۲۰ پینی (۱۸ پین داده) و یک پورت باکسی ۱۰ پینی (۸ پین داده) مناسب کار با IDC با پایه‌های زمین و تغذیه (3.3v یا 5v قابل تنظیم با کلید)
- یک پین گرد ۲ پینی که یک پین را (به همراه زمین) برای کارهایی که با رابط پین گرد بهتر عمل می‌کنند (معمولاً برای ایجاد اتصال محکم‌تر) فراهم می‌کند.
- یک رابط پین هدری مادگی نظامی دو ردیفه‌ی ۱۰ پینی که مناسب ارتباط توسط سیم‌های معمولی یا سیم‌های مخصوص بردبورد می‌باشد.

منبع تغذیه‌ی داخلی (قابلیت اتصال بورد به برق شهر) با کلید روشن و خاموش به همراه ۳ رگولاتور جداگانه‌ی داخلی برای ولتاژهای 2.5v، 3.3v و 5v با آمپردهی 1.5A با LED نشان دهنده و Test Point برای هر ولتاژ

ولتاژهای 3.3v و 5v به طرق مختلفی در دسترس کاربر قرار می‌گیرند:

- داخل هر دو پورت باکسی یک پین زمین و یک پین قابل انتخاب تغذیه در نظر گرفته شده
- یک رابط پین هدری مادگی نظامی ۴ پینی مناسب کار با سیم‌های معمولی یا سیم‌های مخصوص بردبورد (یک پین 3.3v و یک پین 5v با ۲ پین زمین مشترک)
- دو پین گرد ۲ پینی برای ولتاژهای 3.3v و 5v (هر کدام به همراه زمین) مناسب برای اتصالات محکم‌تر

۳ اسیلاتور مجزای 1MHz، 10MHz و 50MHz که در بورد قرار داده شده و همچنین یک سوکت خالی که به اسیلاتور خارجی (پایه دار) مورد نیاز کاربر (هر دو نوع بزرگ «مستطیلی» و کوچک «مربعی») می‌تواند متصل شود.

ورودی داخل بوردی شامل ۸ کلید کوچک روشن/خاموش، ۸ کلید فشاری، یک DIP Switch هشت ردیفه و یک کی‌پد ۴×۴

خروجی داخل بوردی شامل ۱۶ عدد LED پر نور SMD که با روش PWM می‌توان به راحتی نورشان را کم یا زیاد کرد، یک 7-Segment چهارتایی ساعتی با نور قابل تنظیم به همان روش PWM و یک LCD کاراکتری کوچک و مناسب، با نور Back Light که آن هم قابل تنظیم است.

بازر اسیلاتور دار با کلید «خاموش/آماده به کار»

اتصال RCA به ورودی ویدیوی تلویزیون با ۸ سطح رنگ خاکستری

اتصال به مانیتور VGA با ۱۲ سیگنال رنگ (۴۰۹۶ رنگ مختلف)

اتصال به پورت COM (سریال) با سطح استاندارد RS232

اتصال موس و کی‌بورد PS/2

دو LED بـرای نشـان دادن INIT (آمـادگی کـار FPGA و نداشـتن مشـکل) و DONE (درسـت Program شدن FPGA)

جعبه‌ی مناسب و محکم با درب مناسب که از بورد در جابجایی‌ها و همچنین در حین کار محافظـت می‌کند.

ورودی پروگرمر JTAG که هم با پروگرمرهای قابل اتصال به پـورت پارالـل کـار مـی‌کنـد و هـم بـا پروگرمرهای USB

پروگرمر JTAG پورت پارالل با بیش از ۱ متر کابل برای راحتی اتصال و قابلیت تحرک بیشتر

حافظه‌ی فلش داخلی برای نگهداری اطلاعات پیکربندی و پروگرم کردن FPGA برای از بین نـرفتن اطلاعات با قطع برق

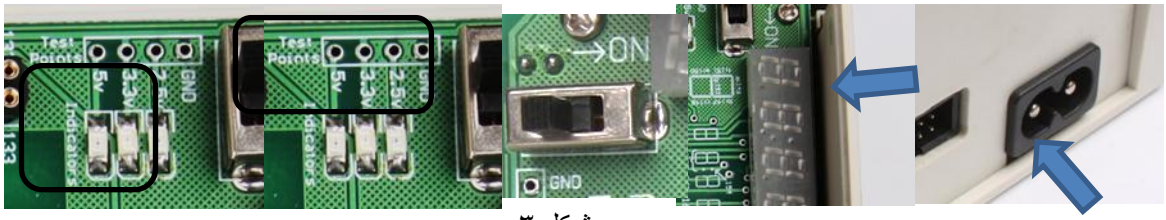
اطلاعات پایه برای راه‌اندازی بورد:

پروگرمر JTAG که به همراه بورد ارائه شده (شکل ۱) را به پورت پارالل کامپیوتر وصل کنید و سر انتهای کابل را به پورت نشان داده شده در شکل ۲ (باکس ۱۰ پینی) متصل کنید. دقت کنید که آن را نباید به پورت دیگری که مشابه این پورت است و از بالای بورد به آن دسترسی داریم (بر خلاف این پورت که تنها پورتیست که از زیر بورد قابل دسترس است) متصل کنید.

شکل ۱

شکل ۲

کابل برق را نیز به محل نشان داده شده در شکل ۳ متصل کنید. و کلید را (که در شکل نشان داده شده) در حالت ON قرار دهید. با این کار باید سه LED نشانگر ولتاژ که آنها هم در شکل نشان داده شده‌اند روشن شوند. البته روشن شدن این LED ها برای نشان دادن درست کار کردن منابع الزامیست (به شرطی که خود LED ها یا مسیر ارتباطیشان تحت شرایط خاصی معیوب نشده باشند) ولی کافی نیست چون عوامل متعددی در درست کار کردن مؤثرند. بهترین راه چک کردن، دیدن ولتاژهای بورد از طریق Test Point هر کدام با استفاده از مالتی‌متر می‌باشد. این Test Point ها را هم در شکل می‌توانید ببینید.

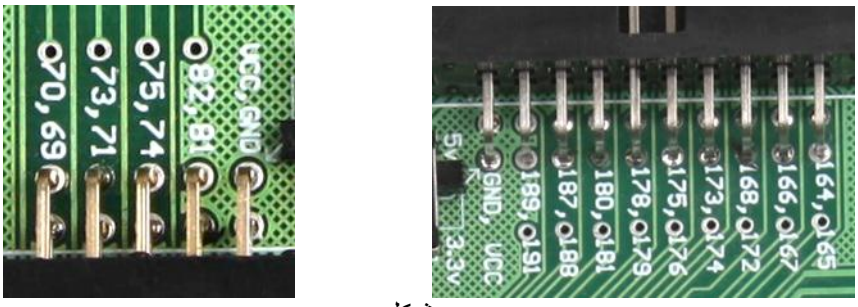


شکل ۳

برای دسترسی به پورت‌های باکسی، سوکت IDC را به باکس مورد نظر کـه در شـکل ۴ نشـان داده شده متصل کنید.

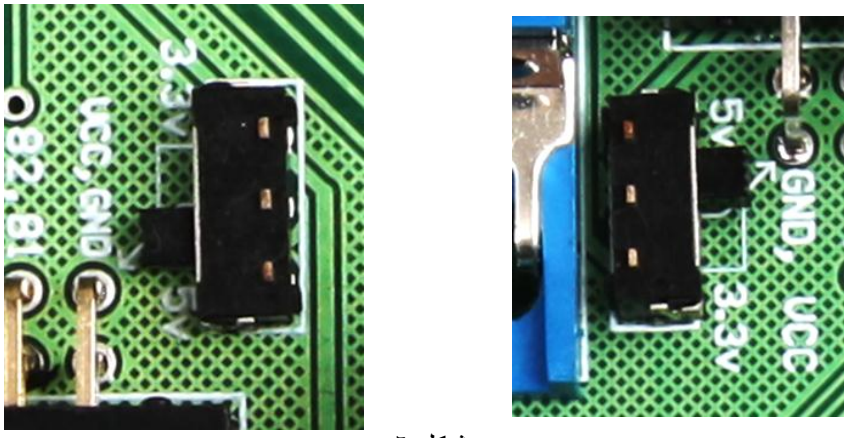
شکل ۴

برای دسترسی به هر پین این پورت‌ها باید به دو عددی که در کنار هر زوج پین نوشـته شـده توجـه کنید. عدد نزدیک پورت، به پین پایینی پورت (پین دورتر نسبت به اعداد) متعلق است و عدد دیگـر به پین بالایی پورت (پین نزدیکتر نسبت به اعداد) متعلق است. در مورد VCC و GND این پورت‌ها هم همین مسئله صادق است؛ بدین صورت که VCC به پین بالایی اشاره می‌کنـد و GND بـه پـین پایینی. شکل ۵ تصویری از این اعداد را نشان می‌دهد.



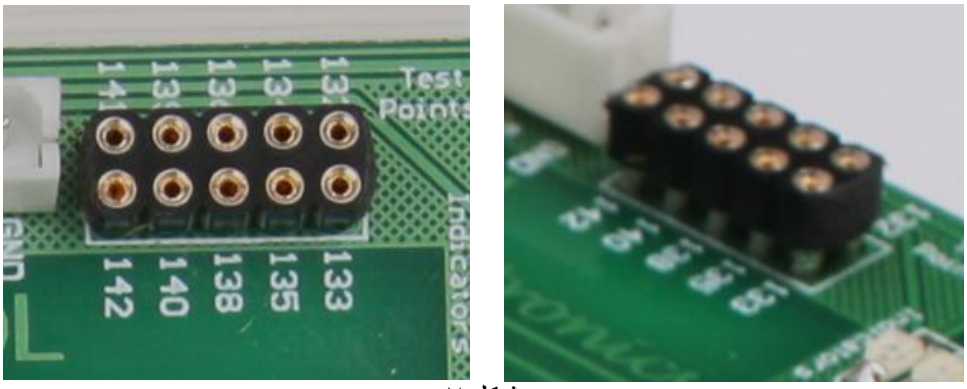
شکل ۵

ولتاژ VCC هر پورت باکسی را می‌توان توسط کلیدی که در کنار آن پورت قرار دارد، بـین 3.3v یـا 5v انتخاب کرد. به شکل ۶ رجوع کنید.



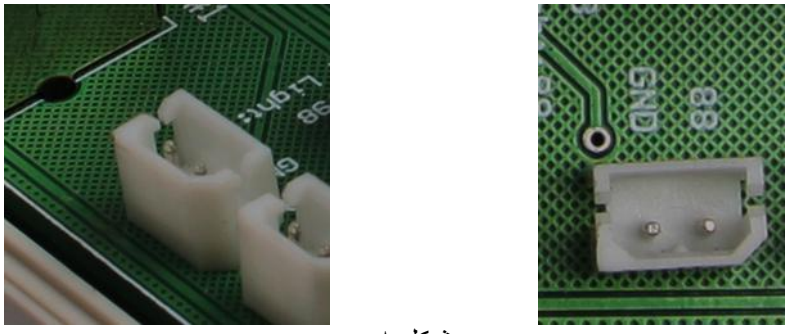
شکل ٦

روش دیگر، استفاده از پین هدر مادگی نظامی می‌باشد. در بورد یک رابط ۵×۲ قـرار داده شـده و در کنار هر پین آن، شماره‌ی پین مرتبط FPGA نوشته شده است. (شکل ۷)



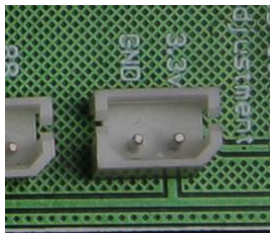
شکل ۷

همچنین در این بورد از یک رابط پین گرد دو پینی (شکل ۸) استفاده شده است که یکی از پین‌های FPGA را به همراه زمین بورد در بر می‌گیرد که در کنار آن مشخص شده است.



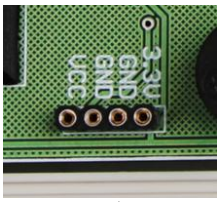
شکل ۸

برای دسترسی به ولتاژهای تولید شده در داخل بورد هم یک روش استفاده از پین گردها می‌باشد که دو عدد از آنها طبق شکل ۹ برای دسترسی به دو ولتاژ 3.3v و 5v (هر کدام به همـراه GND مـدار) در اختیار قرار دارد.

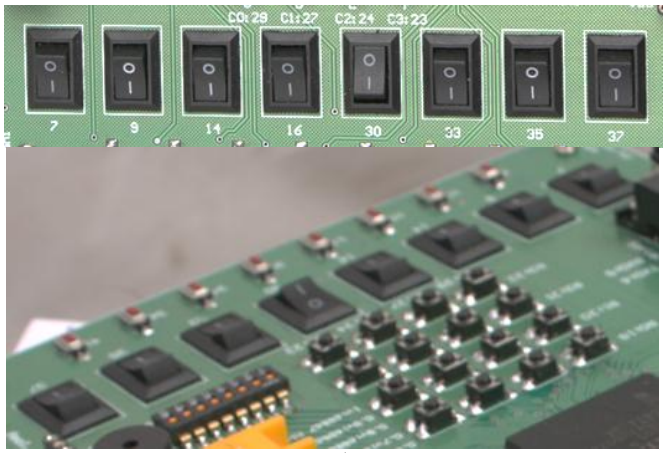
شکل ۹

همچنین از یک پین هدر مادگی نظامی هم برای در دسترس ساختن ولتاژهای 3.3v و 5v به همراه زمین مدار استفاده شده است. (شکل ۱۰) در کنار هر پین، ولتاژ مربوطه نوشته شده است.



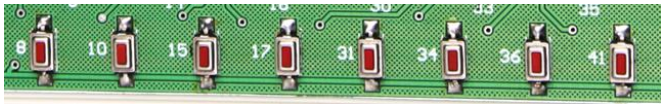
شکل ۱۰

برای ورودی بورد از ۸ کلید روشن/خاموش استفاده شده و شماره‌ی پین مرتبط در زیر هر کلید نوشته شده است. (شکل ۱۱)



شکل ۱۱

مورد دیگر ورودی‌ها، ۸ کلید فشاری (Push Button) می‌باشد و شماره‌ی پین مرتبط در سمت راستشان (برای راحتی کار، البته برای راست دست‌ها! به شرطی که از دست راستشان استفاده کنند!!) نوشته شده است. (شکل ۱۲)

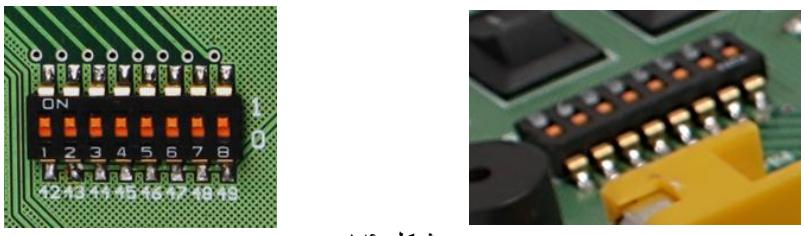


شکل ۱۲

همچنین یک کی‌پد ۴×۴ در نظر گرفته شده که در آن ارقام هگز 0 تا F برای کلیدها در نظر گرفته شده است (شکل ۱۳). R مخفف Row (سطر) و C مخفف Column (ستون) می‌باشد. R0 تا R3 به ترتیب به پین‌های ۲۹، ۲۷، ۲۴ و ۲۳ وصل شده و C0 تا C3 به ترتیب به پین‌های ۱۸، ۲۰، ۲۱ و ۲۲ وصل شده‌اند (در بعضی بوردها جای آنها اشتباه چاپ شده). ستون‌ها خروجی FPGA می‌باشند و سطرها ورودی آن که Pull up شده‌اند.

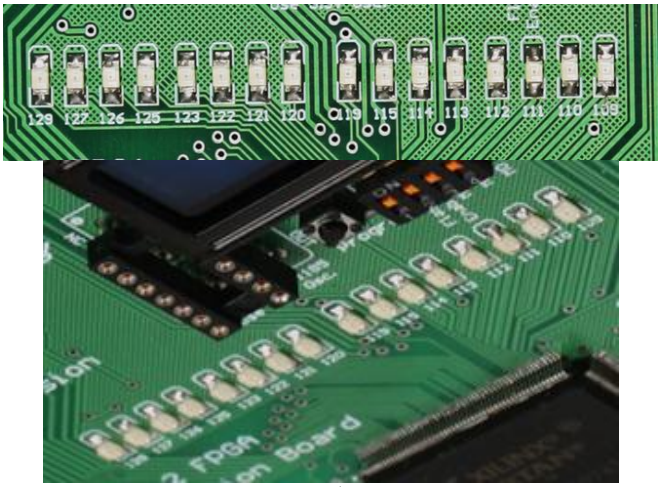
شکل ۱۳

همچنین مطابق شکل ۱۴، یک DIP-Switch هشت تایی هم برای ورودی در نظر گرفته شده که برای هر کدام از کلیدهایش عددی در زیر آن نوشته شده که مرتبط با پین مربوطه از FPGA می‌باشد.



شکل ۱۴

برای خروجی این بورد ۱۶ عدد LED در نظر گرفته شده (شکل ۱۵) که برای راحتی نمایش به دو دسته‌ی ۸ تایی و هر دسته هم به دو دسته‌ی ۴ تایی که با کمی فاصله از هم متمایز شده‌اند تقسیم شده است.

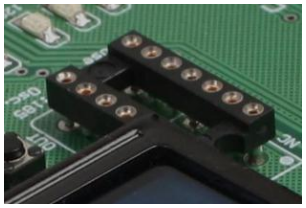
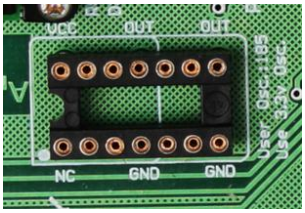


شکل ۱۵

دو LED مربوط به INIT و DONE هم برای نشان دادن درست انجام شدن عملیات اولیه‌ی FPGA و پروگرم شدن (Configure) درست FPGA در نظر گرفته شده است (شکل ۱۶). برای اینکه این LED ها که اکثر اوقات روشنند موجب اذیت شدن چشم نشوند، از رنگی ملایم (معمولاً سبز) استفاده شده است.

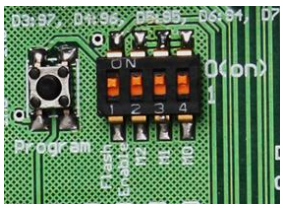
شکل ۱۶

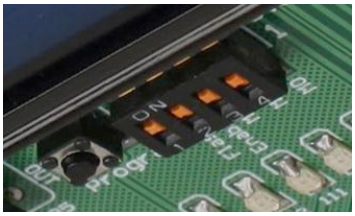
برای منابع پالس ساعت این بورد از سه اسیلاتور مجزای 1MHz، 10MHz و 50MHz در داخل بورد (زیر بورد) استفاده شده که شماره‌ی پین مربوط به هر کدام (که جزو GCLK ها می‌باشند) در کنار فرکانس‌ها در قسمتی از بورد نوشته شده است. (شکل ۱۷) همچنین GCLK چهارم هم به سوکت اسیلاتور (از نوع پایه‌دار) خارجی که در اختیار کاربر قرار دارد متصل شده است. این اسیلاتورها در دو نوع بزرگ (مستطیلی) و کوچک (مربعی) وجود دارند. محل پایه‌ی ۱ این اسیلاتورها که معمولاً با یک علامت دایره روی بدنه‌یشان مشخص می‌شود در روی بورد با همان علامت دایره مشخص شده است (پایه‌ی NC). شکل مربع یا مستطیل روی بورد (با یک گوشه‌ی تیز) هم می‌تواند به صحیح قرار دادن اسیلاتور کمک کند (شکل ۱۷).

شکل ۱۷

برای نگهداری اطلاعات پیکربندی FPGA از تراشه‌ی فلش XCF04SVO20C استفاده شده است. DIP Switch چهار تایی نشان داده شده در شکل ۱۸ و کلید فشاری کنار آن برای کار با فلش و همچنین تعیین نحوه‌ی پروگرم کردن FPGA در نظر گرفته شده است.

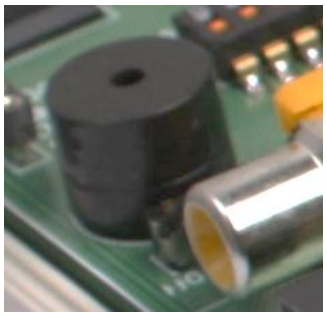




شکل ۱۸

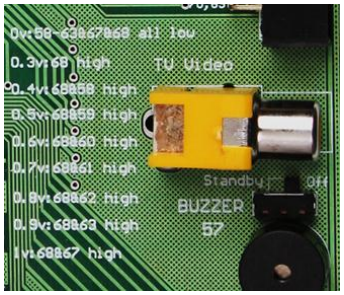
برای اینکه FPGA از طریق فلش پروگرم شود، M0، M1 و M2 که سه کلید از DIP Switch می‌باشند باید 0(ON) شوند (به این حالت مد Master Serial گفته می‌شود) و همچنین کلید Flash Enable یا Flash Boot باید فعال (0 یا ON) باشد. در این صورت هر وقت منبع تغذیه‌ی بورد وصل شود یا هر وقت کلید Program فشار داده شود، FPGA با محتویات فلش پروگرم خواهد شد. خود فلش را هم از طریق رابط JTAG می‌توان پروگرم کرد. در صورتی که بخواهید FPGA از طریق کامپیوتر و از رابط JTAG پروگرم شود، M1 را 0(ON) کنید و M0 و M2 را 1 کنید (به این حالت مد boundary Scan یا JTAG می‌گویند). مدهای دیگر FPGA در بورد آموزشی ما کاربردی ندارند.

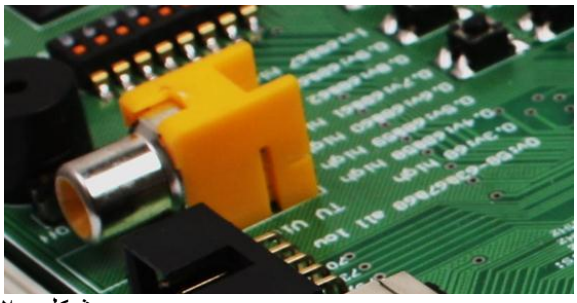
برای این بورد از یک بازر اسیلاتور دار استفاده شده که شماره‌ی پین مربوطه در کنار آن نوشته شده است. برای به صدا در آوردن بازر باید این پین را 0 کرد (Active Low است). همچنین کلیدی که در کنار بازر قرار دارد برای فعال سازی آن (standby یا حالت آماده به کار که منتظر فعال شدن پین می‌ماند) یا خاموش کردن آن (بدون توجه به مقدار پین؛ یعنی قطع صدا به صورت کامل) در نظر گرفته شده است. (شکل ۱۹)



شکل ۱۹

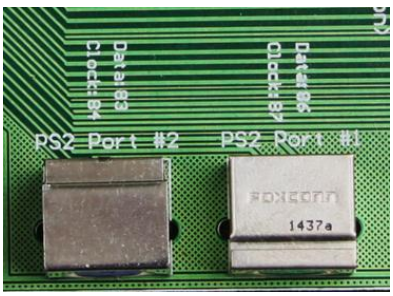
برای اتصال به تلویزیون از یک رابط RCA استفاده شده است (شکل ۲۰). این رابط به ورودی Video تلویزیون برای انتقال تصویر متصل می‌شود. تصاویر به صورت Gray Scale نمایش داده می‌شوند که ۸ سطح خاکستری (ولتاژهای 0.3v تا 1v با فاصله‌های 0.1v) را در بر می‌گیرد. در کنار رابط RCA نوشته شده که برای هر سطح ولتاژ پین‌های خروجی FPGA باید چه مقادیری داشته باشند. (پین‌هایی که برای هر سطح ولتاژ ذکر نشده باید 0(Low) شوند).





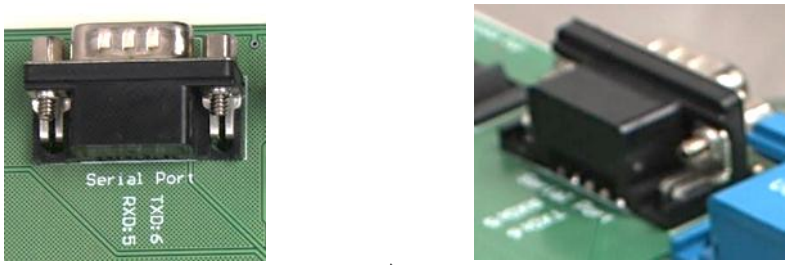
شکل ۲۰

با استفاده از دو رابط PS/2 روی بورد می‌توان به موس و کی‌بورد PS/2 متصل شد (فرقی با هم ندارند). برای تغذیه‌ی آنها از ولتاژ 3.3v استفاده شده است. پین‌های مرتبط با Data و Clock مربوط به هر کدام در کنارشان نوشته شده است. (شکل ۲۱)



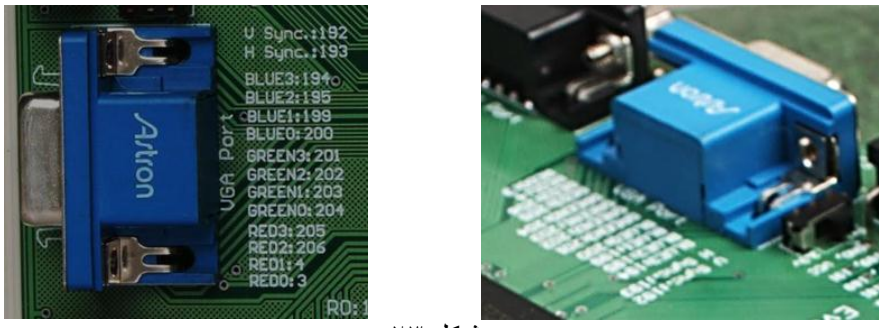
شکل ۲۱

برای اتصال به پورت COM (با سطح ولتاژ RS232) از رابط پورت سریال در بورد استفاده کنید (شکل ۲۲). دو پین مربوط به ارسال و دریافت در کنار آن نوشته شده است.



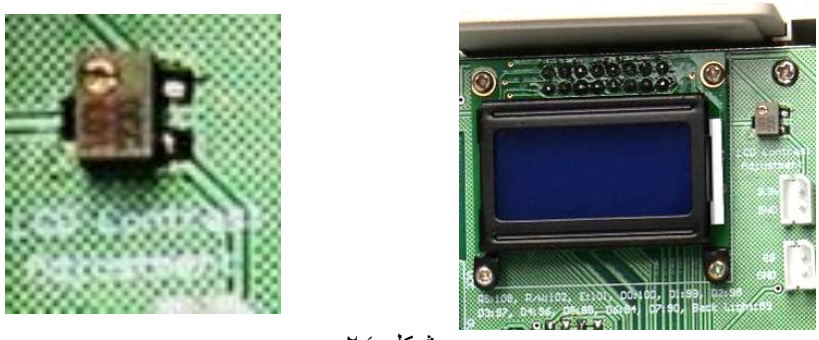
شکل ۲۲

با استفاده از رابط VGA می‌توان به مانیتور متصل شد. برای هر رنگ چهار بیت در نظر گرفته شـده که در کل می‌توان تا 4096 رنگ مختلف داشت. پین‌های مربوط به هر رنگ و دو سیگنال HSync و VSync در کنار پورت نوشته شده است. (شکل ۲۳)



شکل ۲۳

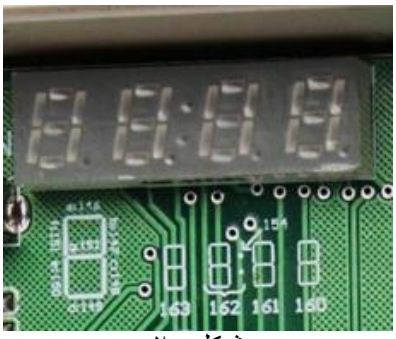
LCD در نظر گرفته شده برای این بورد، 8×2 می‌باشد. پین‌های مرتبط با آن در زیر آن نوشته شـده است. میزان Contrast را با مالتی‌ترنی که در کنار LCD قرار داده شده می‌توانید کم و زیاد کنید. نور Back Light این LCD را می‌توان با پین مشخص شده و به صورت PWM کم و زیـاد کـرد. (شـکل ۲۴)



شکل ۲٤

یک Seven Segment کوچک چهار رقمی (مالتی پلکس شده) ساعتی (با علامت دو نقطه) هم برای این بورد در نظر گرفته شده است (شکل ۲۵). همانطور که روی بورد به صورت گرافیکـی نشـان داده

شده، پین مربوط به دو نقطه با رقم سوم از راست با هم کار می‌کنند. در ضمن در این 7-Seg. علامت‌های ممیز کار نمی‌کنند.



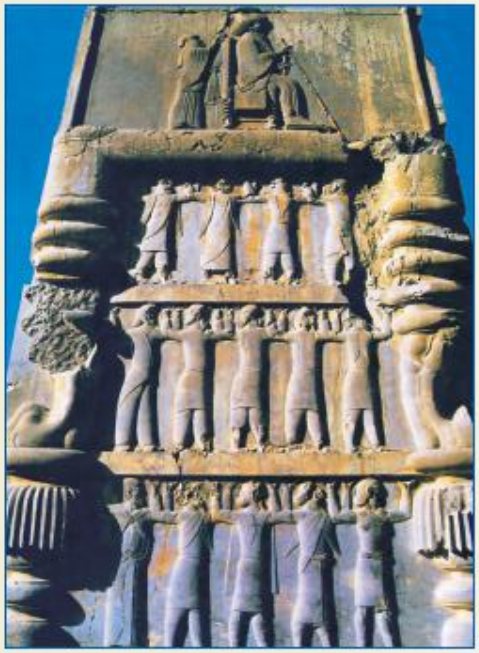
شکل ۲۵

در مورد زبان VHDL که یکی از زبان‌های اصلی برنامه‌نویسی برای ASIC یا FPGA می‌باشد هم با مطالعه‌ی کتاب‌های مربوطه می‌توانید اطلاعات مورد نیازتان را بدست آورید. یکی از این کتاب‌ها که ترجمه‌ی طراح همین بورد است، کتاب "VHDL" از آقای دکتر زین‌العابدین نوابی می‌باشد. انتشارات ترجمه‌ی آن هم «نوپردازان» می‌باشد و طرح روی جلد آن را می‌توانید در صفحه‌ی بعد ببینید.

زین‌العابدین نوابی

VHDL

تحلیل و مدل‌سازی سیستم‌های دیجیتال



رضا فانی
عضو هیأت علمی دانشگاه شمال آمل

زین‌العابدین نوابی

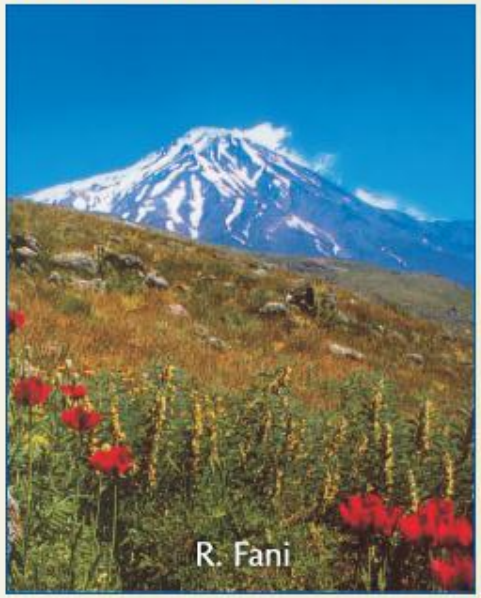
VHDL
تحلیل و مدل‌سازی سیستم‌های دیجیتال
رضا فانی

نوپردازان

Z. Navabi

VHDL

ANALYSIS AND MODELING OF DIGITAL SYSTEMS



ISBN 964-7006-64-0

نوپردازان

معرفی بوردی دیگر از AmOL electronics

بورد AMOL-1:

بوردی مبتنی بر ورودی/خروجی که مناسب پیش ساخت پروژه‌های متنوع و مختلف می‌باشد و به دلیل داشتن پورت‌های زیاد، آزادی عمل بیشتری در عین سادگی و کوچکی در اختیارتان قرار می-دهد.

۱۱۷ پورت مجزای ورودی/خروجی برای کاربر

آداپتور مجزا به همراه ۳ رگولاتور جداگانه‌ی داخلی برای ولتاژهای 2.5v، 3.3v و 5v با آمپردهی 1.5A به همراه LED نشان دهنده برای هر ولتاژ و همچنین Test Point برای هر ولتاژ

ولتاژهای 3.3v و 5v به طرق مختلفی در دسترس کاربر قرار می‌گیرند.

۳ اسیلاتور مجزای 1MHz، 10MHz و 50MHz که در بورد قرار داده شده و همچنین یک پایه‌ی ورودی (در رابط مادگی ۱۰ پینی) که به اسیلاتور خارجی مورد نیاز کاربر می‌تواند متصل شود.

ورودی داخل بوردی شامل ۲ کلید کوچک روشن/خاموش و ۶ کلید فشاری

خروجی داخل بوردی شامل ۸ عدد LED پر نور SMD که می‌توان با روش PWM به راحتی نورشان را کم یا زیاد کرد.

دو LED برای نشان دادن INIT (آمادگی کار FPGA و نداشتن مشکل) و DONE (درست پروگرم شدن FPGA)

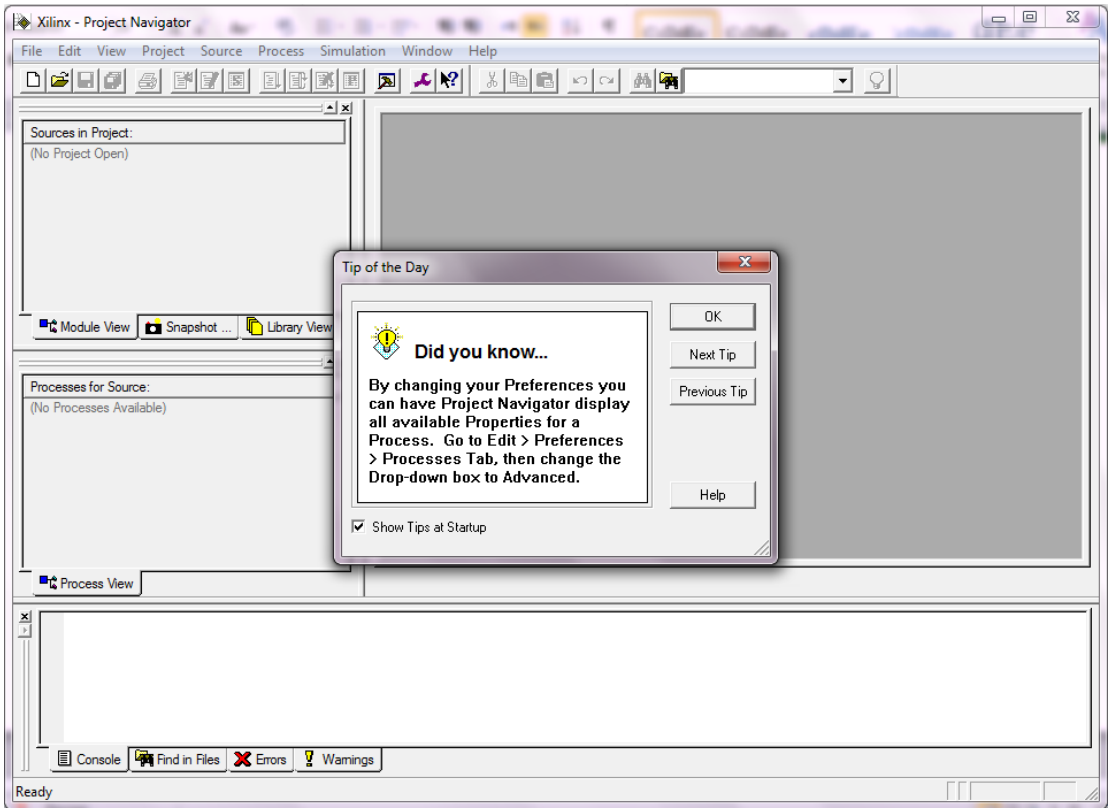
جعبه‌ی مناسب و محکم با درب مناسب که از بورد در جابجایی‌ها و همچنین در حین کار محافظت می‌کند.

ورودی پروگرمر JTAG که هم با پروگرمرهای قابل اتصال به پورت پارالل کار می‌کند و هم با پروگرمرهای USB

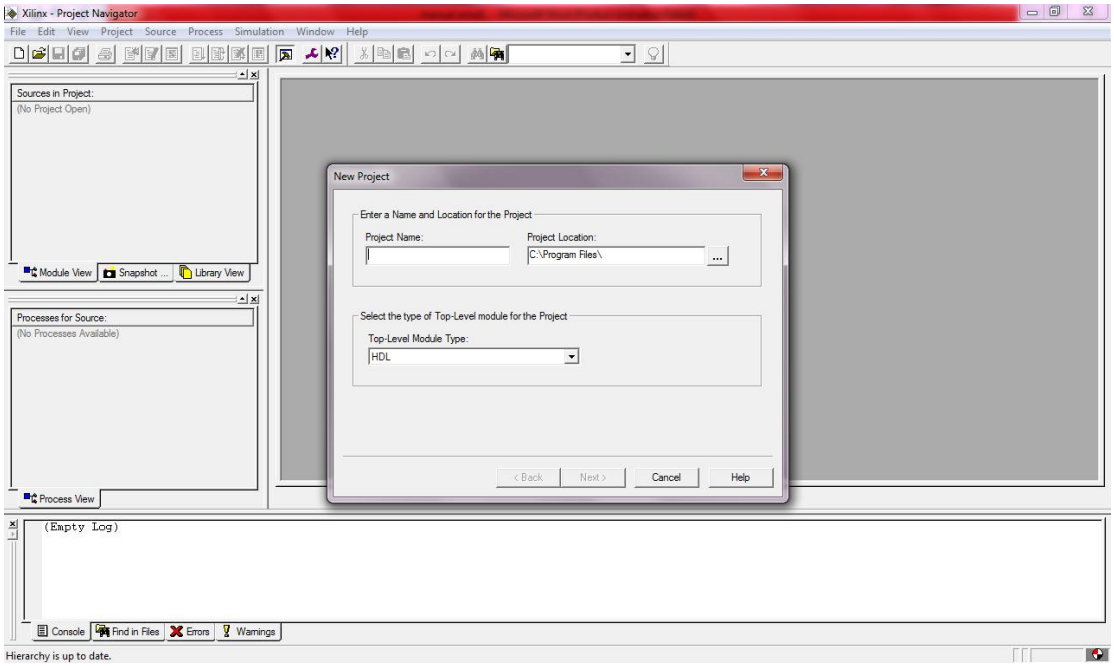
پروگرمر JTAG پورت پارالل با بیش از ۱ متر کابل برای راحتی اتصال و قابلیت تحرک بیشتر

راهنمای به کارگیری بورد AMOL-2 و نرم‌افزار ISE با مثال ساده:

بعد از نصب نرم‌افزار، از منـوی start ویندوز، در فولـدر Xilinx ISE 7.1i روی Project Navigator کلیک کنید؛ پنجره‌ی زیر باز می‌شود. برای اینکه پیغام (نکته‌ی وسط صفحه) برای بارهای بعدی کـه این برنامه را اجراء می‌کنید نشان داده نشود، تیک را بردارید و OK کنید.



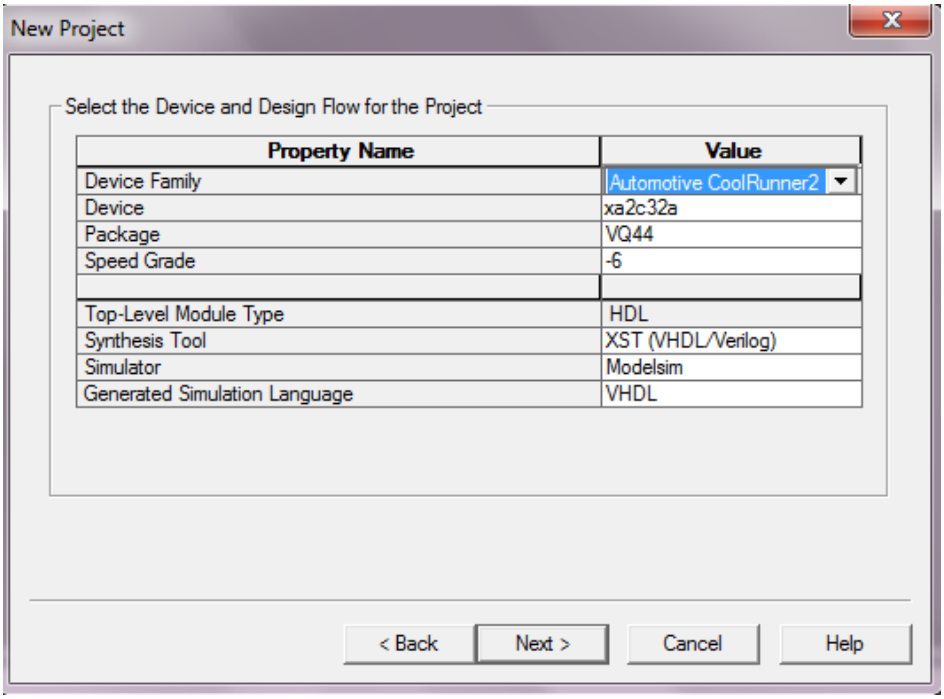
File -> New Project… را انتخـاب کنیـد (منظـور از منـوی File، گزینـه‌ی New Project… را انتخاب کنید). یک صفحه به صورت شکل زیر در صفحه‌ی اصلی باز می‌شود:



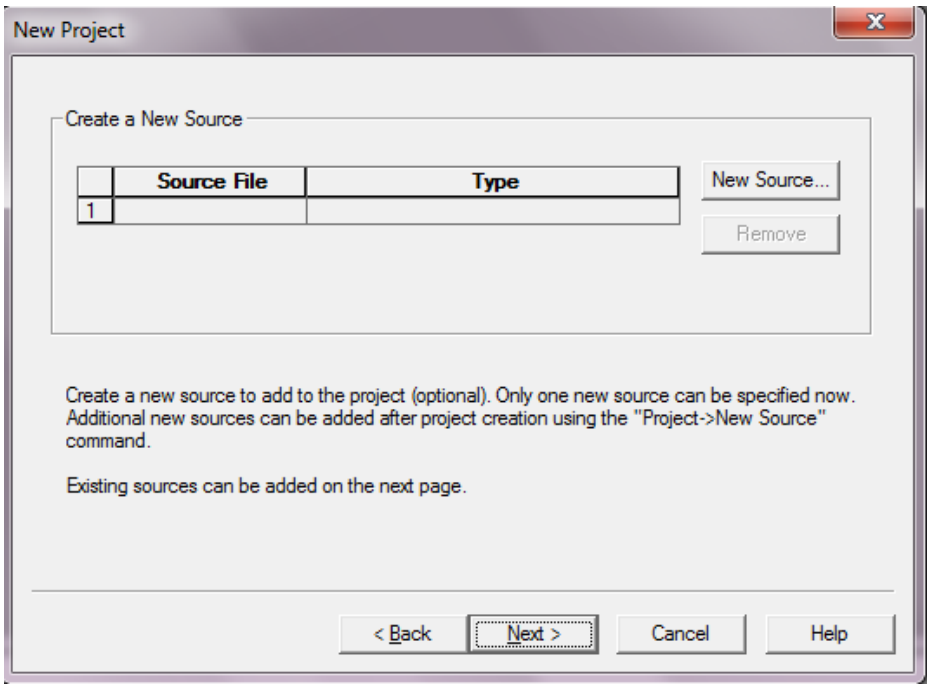
در کادر Project Name: اسم پروژه و در کادر Project Location: مسیر آن را وارد نمایید. توضیح اینکه همین که نامی برای پروژه وارد می‌کنید، فولدری به همان نام در مسیر فعلی تولید می‌شود و پروژه در آن قرار داده می‌شود. ما نام test را برای پروژه در نظر گرفته‌ایم و مسیر را درایو C: انتخاب کردیم که این پروژه در فولدر test (که همانطور که گفته‌ایم خود به خود ساخته می‌شود) در آن مسیر قرار می‌گیرد.

در کادر Top-Level Module Type: همان مورد پیش‌فرض که HDL می‌باشد را انتخاب کنید و کلید Next > را بزنید.

صفحه‌ی باز شده به صورت زیر در می‌آید:

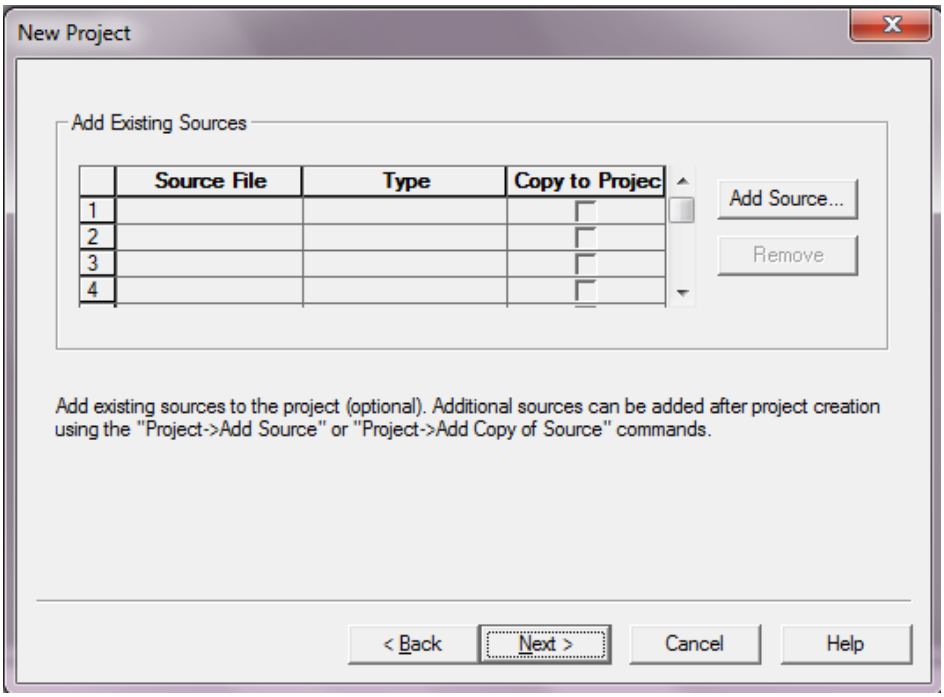


در کـادر Device Family، گزینـه‌ی Spartan2 را پیـدا کنیـد و انتخـاب کنیـد. در کـادر Device، xc2s200 را انتخاب کنید و در کادر Package، pq208 را انتخاب کنید و در قسمت Speed Grade، 5- را انتخاب کنید. چهار کادر بعدی را به همان صورت پیش‌فرضشان بپذیرید. یعنی همان‌هایی کـه در شکل بالا می‌بینید. سپس کلید Next > را بزنید. صفحه به صورت زیر در می‌آید:

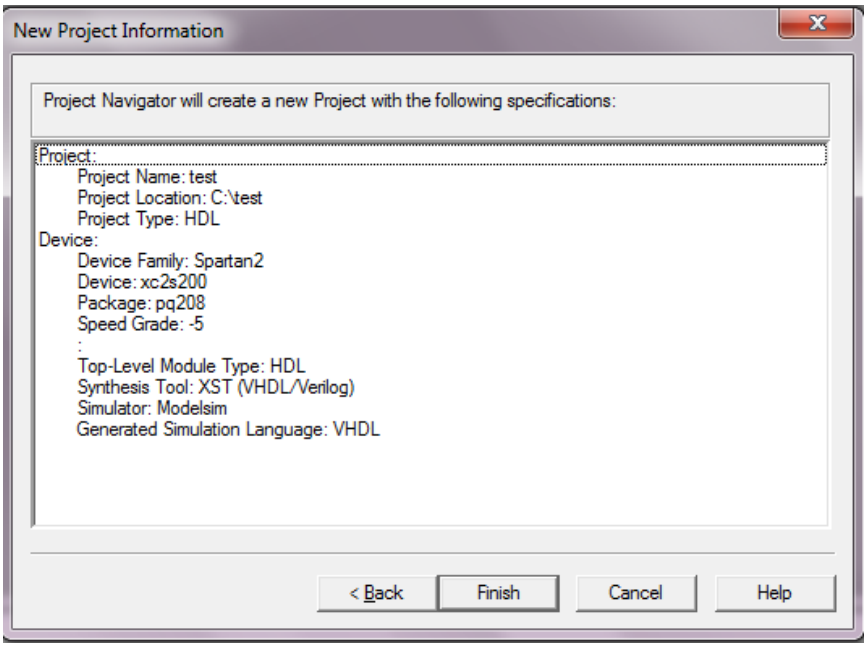


در این صفحه می‌توان فایل سورس تولید کرد و در صورتی که بخواهید می‌توانیـد بعـداً ایـن کـار را انجام دهید. ما هم این کار را به بعد موکول می‌کنیم. پس کلید Next > را بزنید.

صفحه به صورت زیر در می‌آید:



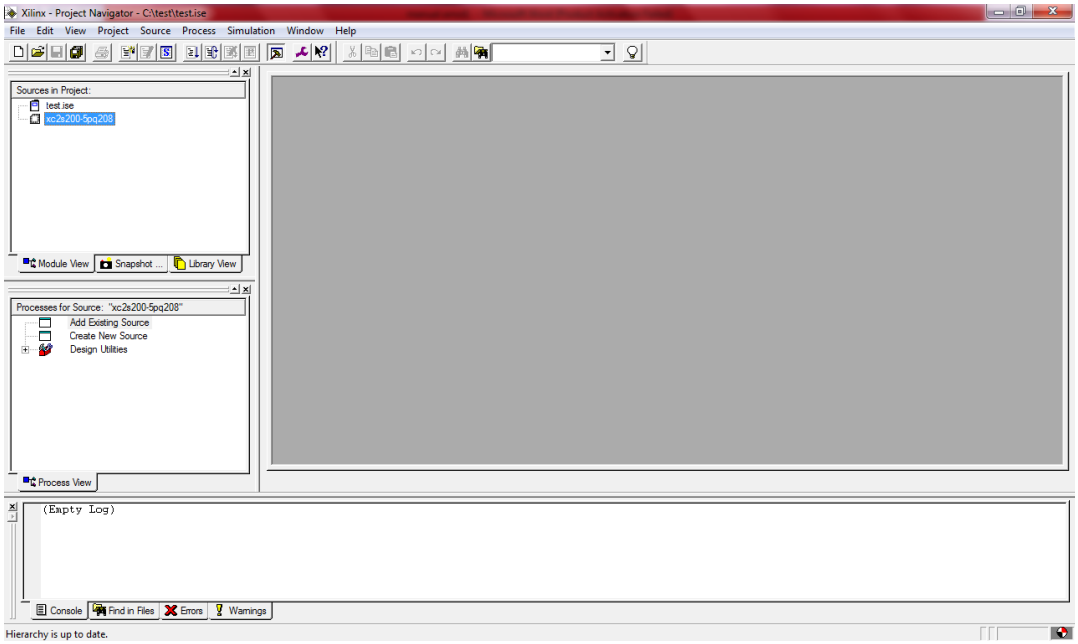
در این صفحه می‌توان فایل‌های از قبل تولید شده را به پروژه اضافه کرد که البته در صورتی که بخواهید می‌توانید بعداً این کار را انجام دهید. ما هم این کار را اینجا انجام نمی‌دهیم. پس کلید < Next را بزنید. صفحه به صورت زیر در می‌آید:



در این صفحه اطلاعات و تنظیمات پروژه که در صفحات قبلی ست کرده‌اید نشان داده می‌شود. برای تأیید کردن آن کلید Finish را بزنید و در صورتی که نیاز به اصلاح دارید می‌توانید با کلید < Back

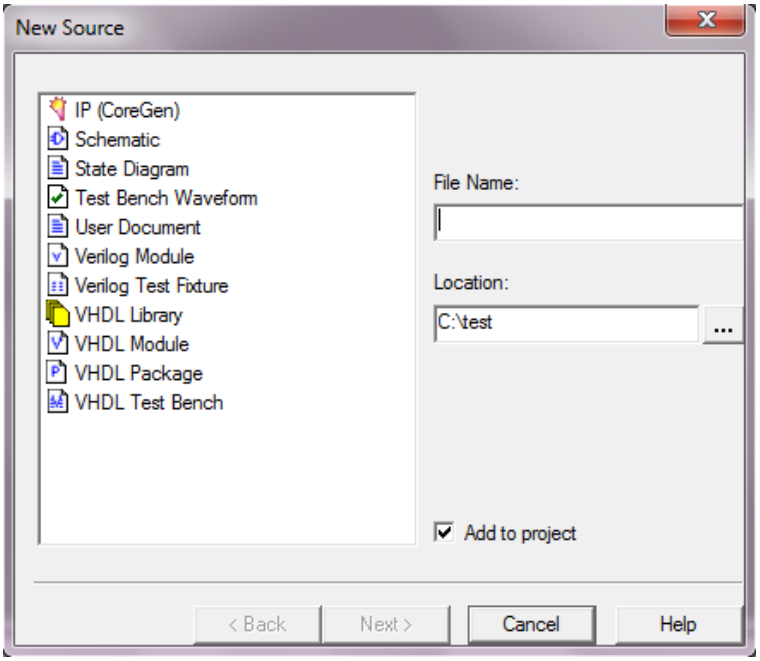
به صفحات قبل برگردید. با زدن کلید Finish صفحه‌ی فرعی بسته شده و به صفحه‌ی اصلی بر مـی-گردید.

صفحه‌ی اصلی بدینگونه خواهد شد:



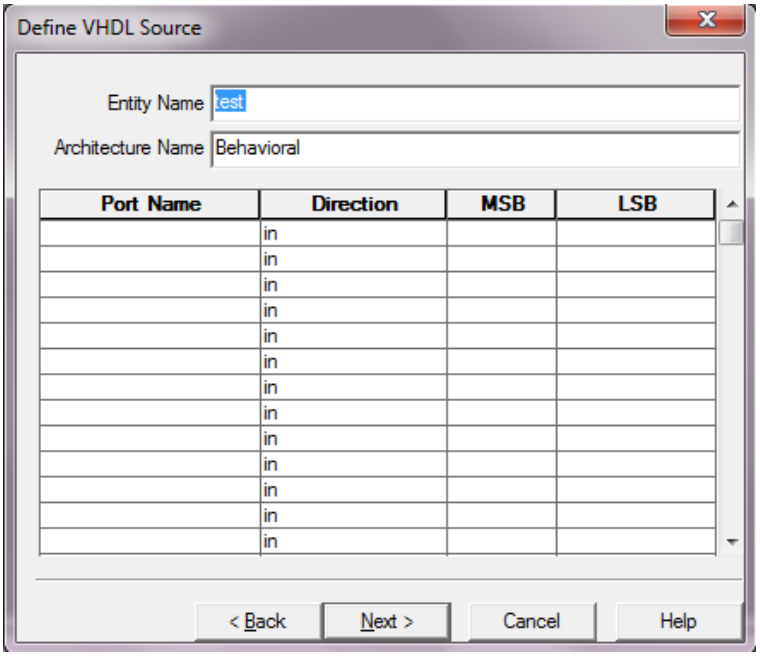
حال می‌خواهیم در پروژه یک فایل سورس بسازیم. روی Create New Source دابل کلیـک کنیـد.

صفحه‌ی زیر در صفحه‌ی اصلی باز می‌شود:



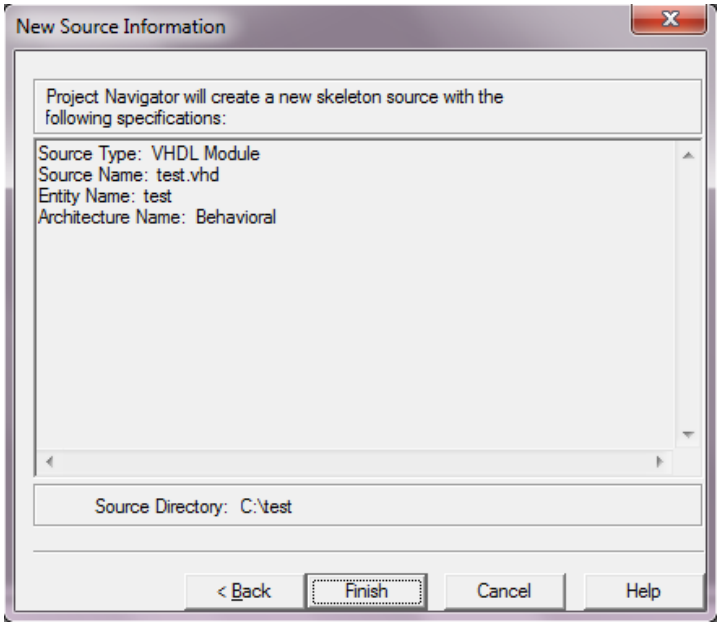
از سمت چپ این صفحه گزینه‌ی VHDL Module را انتخاب کرده و در کادر File Name: نام فایلی که قرار است تولید شود را بنویسید و کلید Next > را بزنید. ما نام test را برای فایل انتخاب کردیم.

صفحه به صورت زیر در می‌آید:



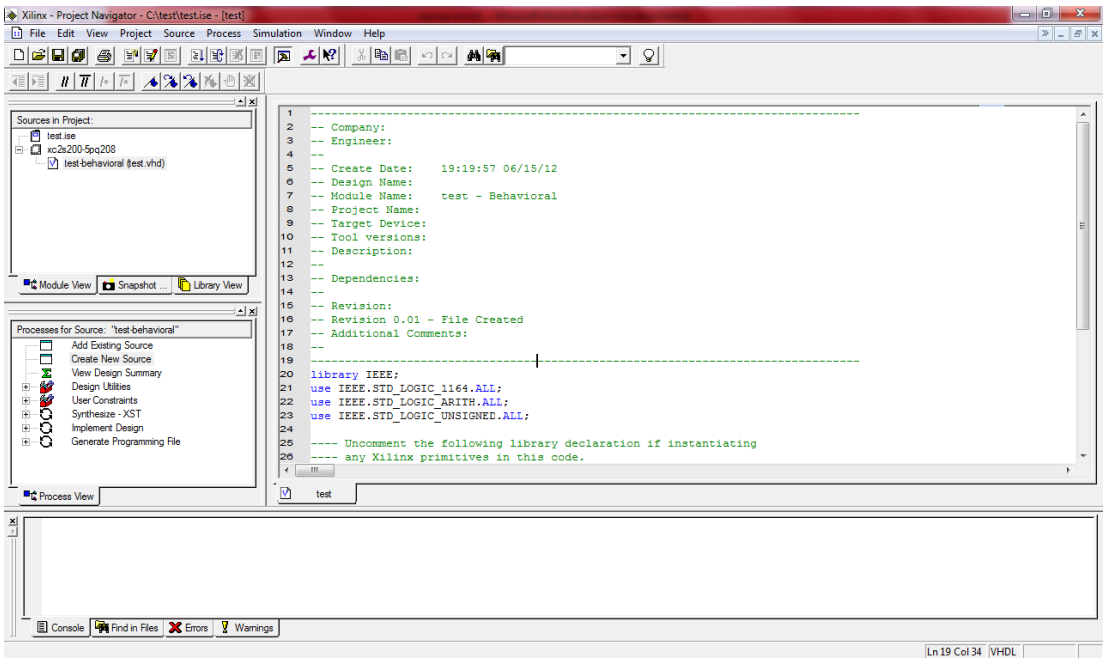
در این صفحه اسم Entity کامپوننت به صورت پیش‌فرض همان نام فایل در نظر گرفته شده و نام Architecture هم Behavioral در نظر گرفته شده که می‌توان آنها را مطابق دلخواه تغییر داد ولی ما آنها را به همان صورت می‌گذاریم. همچنین در این صفحه می‌توان ورودی/خروجی‌های کامپوننت را مشخص کرد که ما این کار را به بعد و در خود متن برنامه موکول می‌کنیم. کلید Next > را بزنید.

صفحه به صورت زیر تغییر می‌کند:



در این صفحه، اطلاعات فایل تولید شده که فایلیست با پسوند vhd نشان داده می‌شود. بـرای تأییـد کلید Finish را بزنید.

صفحه‌ی اصلی به صورت زیر در می‌آید:



در کادر سمت راست، متنی که خود نرم‌افزار ISE برای راحتی کار نوشـتن برنامـه تولیـد کـرده، بـه عنوان محتویات فایل تولید شده نشان داده می‌شود. از این به بعد باید تغییرات کد برنامه را در اینجا اعمال کرد. کد اولیه که توسط ابزار سنتز (ISE) تولید شده، به صورت زیر می‌باشد:

```
----------------------------------------------------------------------------------
-- Company:
```

```
-- Engineer:
--
-- Create Date:    19:19:57 06/15/12
-- Design Name:
-- Module Name:    test - Behavioral
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity test is
end test;

architecture Behavioral of test is

begin


end Behavioral;
```

می‌خواهیم برنامه‌ای بنویسیم که پس از سنتز شدن و پروگرم کـردن FPGA، هشـت عـدد از LED های روی بورد را روشن و خاموش (چشمک زن) کند. برای این کار در ابتـدا قسـمت‌هـای entity و architecture را به صورت زیر تغییر می‌دهیم:

```
entity test is
        PORT (clk, reset: IN STD_LOGIC; leds: OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
end test;

architecture Behavioral of test is
        SIGNAL leds_temp : STD_LOGIC_VECTOR (7 DOWNTO 0);
begin

        PROCESS (clk, reset)
                VARIABLE count : INTEGER;
        BEGIN
                IF reset = '1' then
                        count := 0;
                ELSIF rising_edge (clk) THEN
                        count := count + 1;
                        IF count = 50000000 THEN
```

```
                    count := 0;
                    leds_temp <= NOT leds_temp;
                END IF;
            END IF;
        END PROCESS;

leds <= leds_temp;

end Behavioral;
```

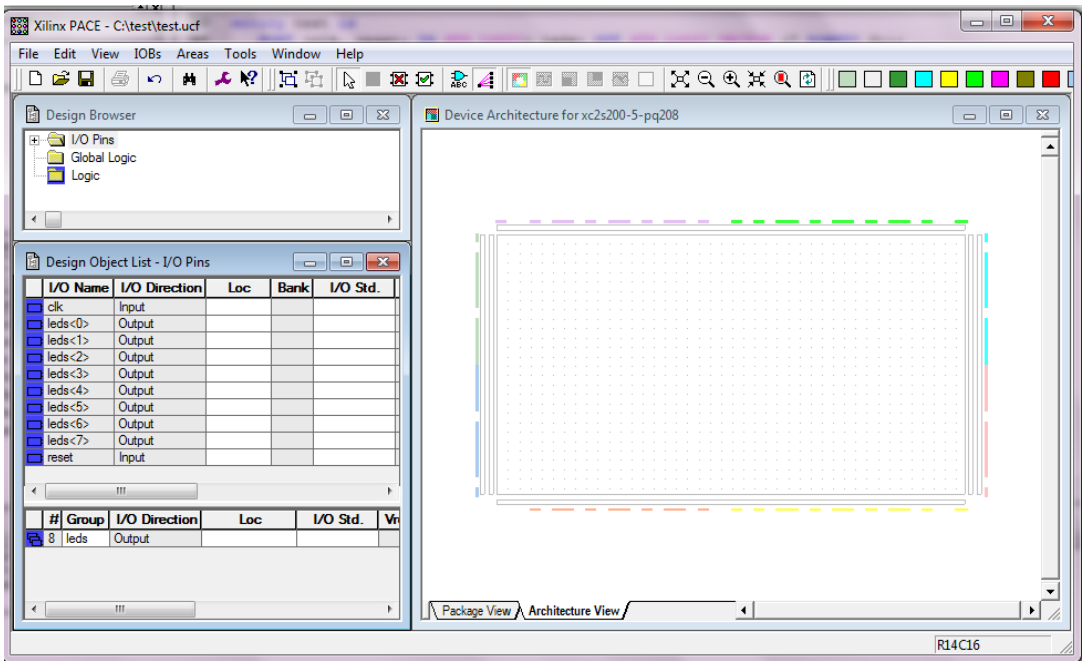
سپس تغییرات برنامه را ذخیره کنید و در کادر وسط سمت چپ صفحه روی Synthesize –XST یا علامت دایره مانند (فلش دار) کنار آن دابل کلیک کنید. با این کار کد نوشته شده، سنتز می‌شود و اطلاعات سنتز که مهمترینش حداکثر فرکانس مجاز clk است به صورت گذرا (و همچنین به صورت دائم با دابل کلیک کردن روی زیرگزینه‌ی View Synthesis Report داخل همان گزینه‌ی Synthesize –XST که با کلیک کردن روی علامت + کنارش باز می‌شود) نمایش داده می‌شود. برای کد و FPGA ما این مورد به صورت زیر گزارش داده شده است:

```
Minimum period: 17.440ns (Maximum Frequency: 57.339MHz)
```

که با این حساب می‌توان از هر سه اسیلاتور موجود در بورد (1MHz، 10MHz و 50MHz) استفاده کرد. adad koochektar az 50 dg madar dorost karesho anjam nemide

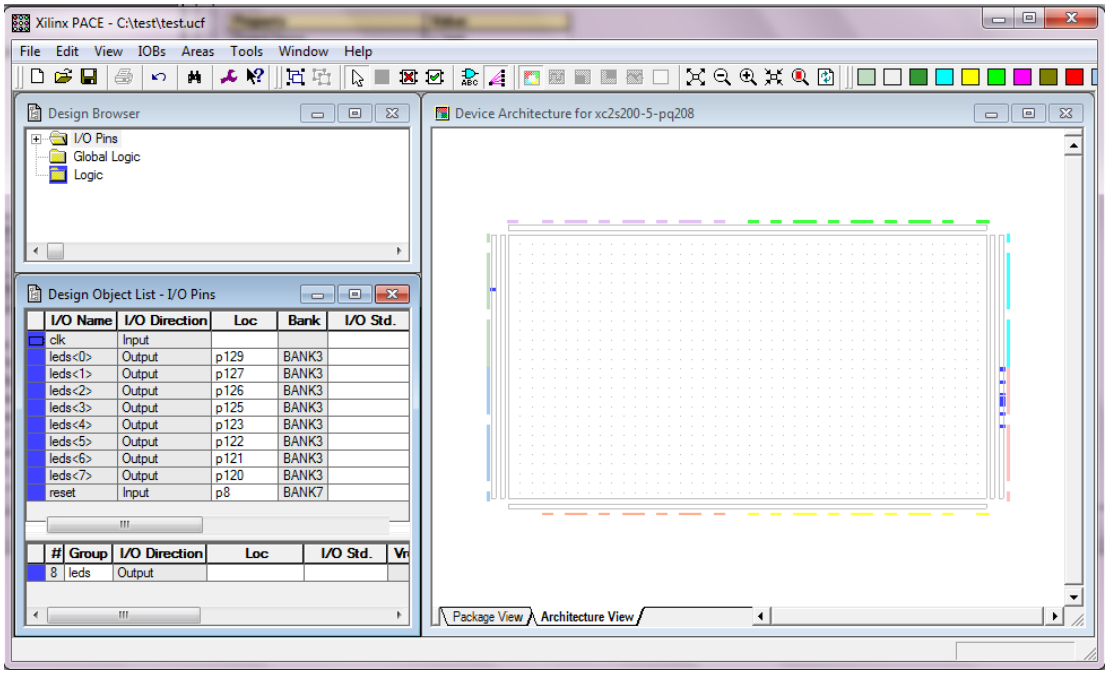
در این مرحله باید اطلاعات مربوط به پایه‌های مورد استفاده از FPGA را در فایلی با نام پروژه و پسوند UCF وارد کنیم. برای این کار روی علامت + کنار User Constraints در همان کادر وسطی سمت چپ صفحه کلیک کنید و سپس از منوی باز شده روی گزینه‌ی Assign Package Pins دابل کلیک کنید. کادری باز می‌شود که در آن در مورد تولید و اضافه کردن فایلی با پسوند UCF به پروژه از شما اجازه می‌خواهد که باید کلید Yes را کلیک کنید.

با این کار صفحه‌ی زیر (Xilinx PACE) باز می‌شود. در اینجا برای اینکه شما همه‌ی اطلاعات را بتوانید ببینید در شکل کمی جای کادر وسط سمت چپ را که شامل اسم ورودی/خروجی‌های کد می‌باشد، بزرگتر کرده‌ایم.



در ستون Loc در روبروی هر پورت باید شماره‌ی پینی از FPGA وارد شود کـه ورودی یـا خروجـی مورد نظر به آن وصل شده است. در مثال ما، پورت‌های leds به LED های روی بورد و پـورت reset به یکی از کلیدهای فشاری روی بورد متصل می‌شود. برای این کار در هر سطر در ستون Loc، حرف p را به همراه شماره‌ی پین مربوطه (مثلاً p120) می‌نویسیم.

کامل شده‌ی این موارد را (بجز سیگنال clk که در اینجا با شرایطی که طبق آن پیش رفته‌ایم نمی-توان به آن پین انتساب داد و به جای آن، به صورت متنی -چنانکه در ادامه ذکر مـی‌کنـیم- بـه آن سیگنال، پین انتساب می‌دهیم) در شکل زیر می‌بینید:



سپس دکمه‌ی ذخیره را بزنید و در کادری که باز می‌شود مورد پیش فرض آن را که XST {} Optional می‌باشد را قبول کرده و OK کنید و پس از آن، صفحه‌ی Xilinx PACE را می‌توانید ببندید.

حال برای تعیین پین سیگنال clk، در صفحه‌ی اصلی و همان کادر وسط سمت چپ که در آن گزینه‌ی User Constraints باز شده، روی زیرگزینه‌ی Edit Constraints (Text) دابل کلیک کنید. در سمت راست، محتویات فایل test.ucf که به پروژه اضافه شده نمایش داده می‌شود که برای مثال ما به صورت زیر می‌شود:

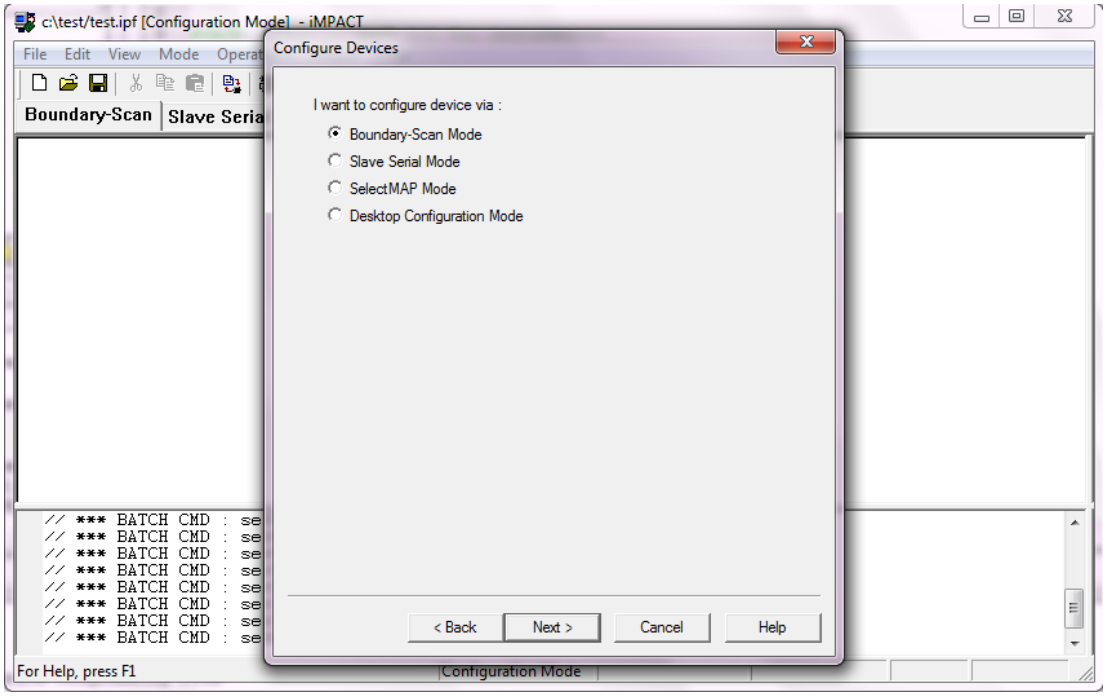
```
#PACE: Start of Constraints generated by PACE

#PACE: Start of PACE I/O Pin Assignments
NET "leds<0>"  LOC = "p129"  ;
NET "leds<1>"  LOC = "p127"  ;
NET "leds<2>"  LOC = "p126"  ;
NET "leds<3>"  LOC = "p125"  ;
NET "leds<4>"  LOC = "p123"  ;
NET "leds<5>"  LOC = "p122"  ;
NET "leds<6>"  LOC = "p121"  ;
NET "leds<7>"  LOC = "p120"  ;
NET "reset"  LOC = "p8"  ;

#PACE: Start of PACE Area Constraints

#PACE: Start of PACE Prohibit Constraints

#PACE: End of Constraints generated by PACE
```

برای سیگنال clk کافیست یکی از خطوط میانی (آنهایی که با NET شروع می‌شوند) را کپی کرده و جای قسمت سیگنال کـه در بـین دو گیومـه‌ی اول آمـده، سـیگنال clk را وارد کنیـد و در قسـمت شماره‌ی پین، "p80" را که مربوط به اسیلاتور داخلی 50MHz می‌باشد وارد کنید. (یعنی در حقیقت خط زیر را اضافه کنید:)

```
NET "clk"  LOC = "p80"  ;
```

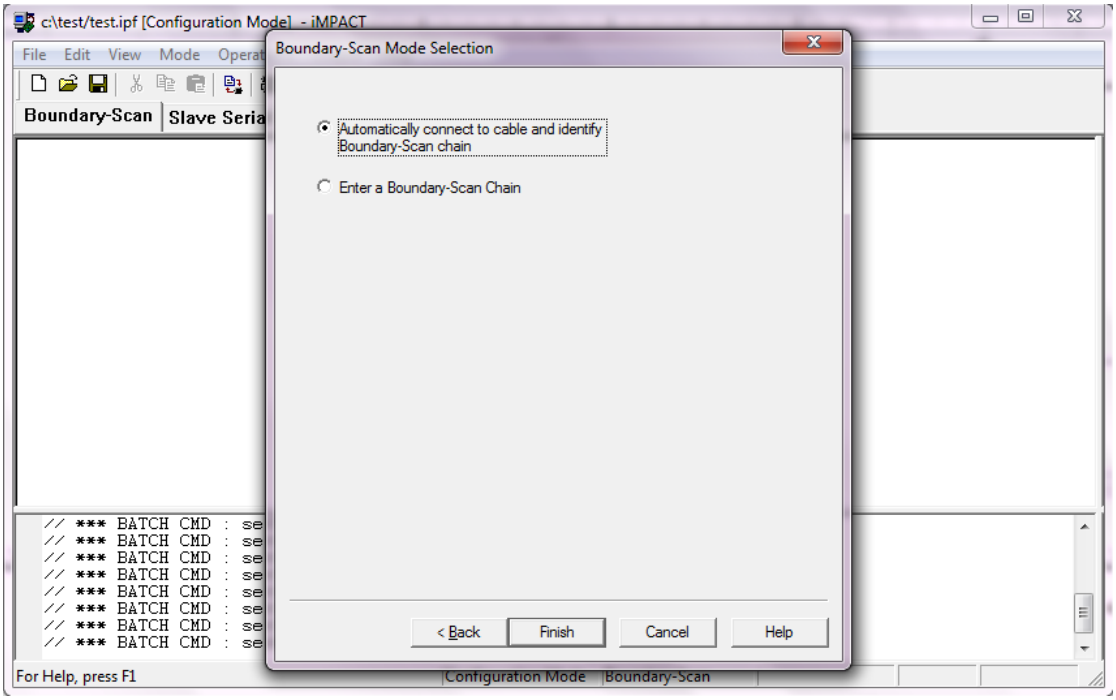
پـس از ایـن کـار، کلیـد ذخیـره را بزنیـد و سـپس در کـادر بـالایی سـمت چـپ روی خـود فایـل «test-behavioral (test.vhd)» کلیک کنید.

سپس در کادر وسط سمت چپ، روی گزینه‌ی Implement Design دابل کلیک کنید. سپس با بـاز کردن زیرگزینه‌ی آخر که Generate Programming File است (با کلیک روی علامـت + کنـار آن) روی گزینه‌ی Configure Device (iMPACT) دابل کلیک کنید. با این کار صفحه‌ی زیر بـاز مـی-شود:



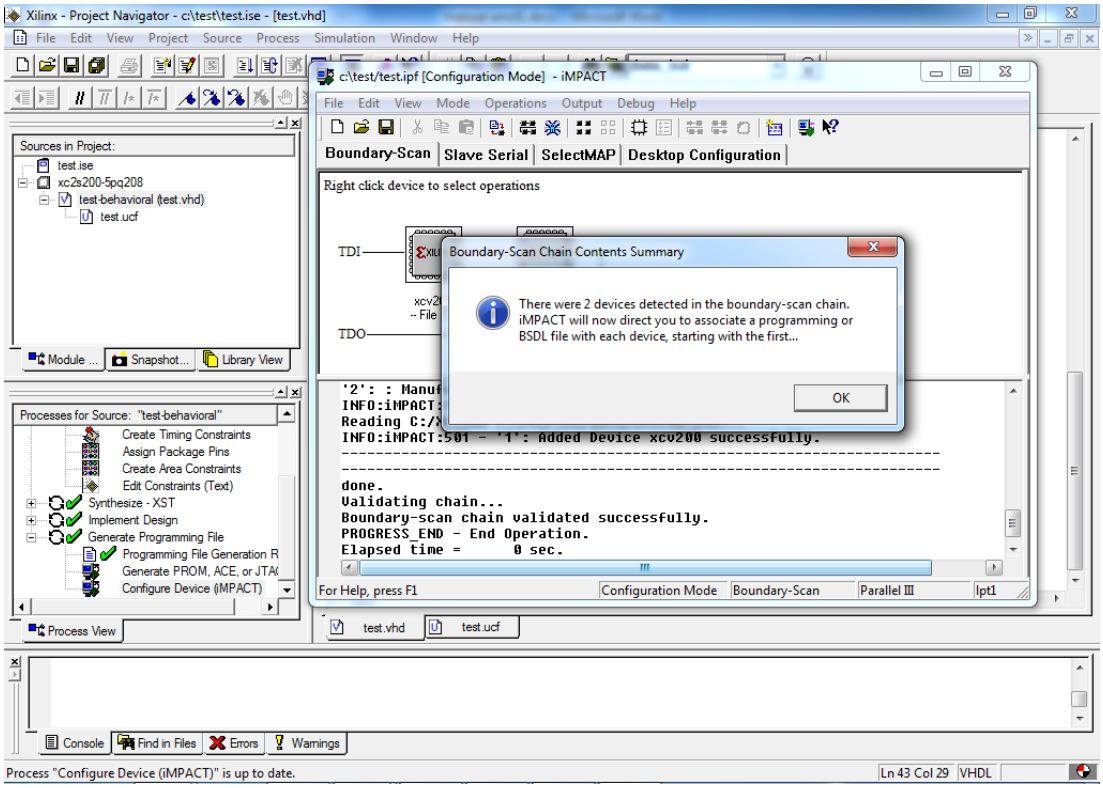
در این صفحه همان گزینه‌ی اول (Boundary-Scan Mode یا همان JTAG) را که پیش‌فـرض هـم می‌باشد قبول کرده و کلید Next > را بزنید.

با این کار صفحه به صورت زیر تغییر می‌کند:

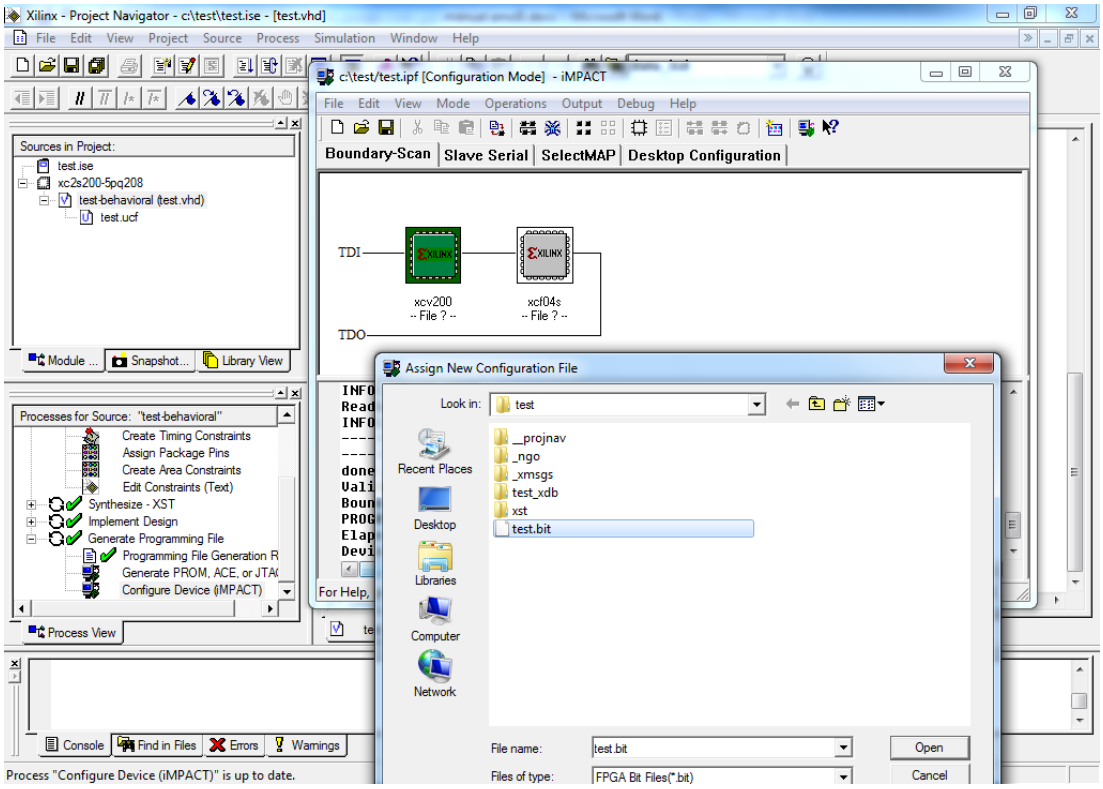


در این صفحه هم همان گزینه‌ی اول را که پیش‌فرض هم می‌باشد قبول کرده و کلید Finish را بزنید.

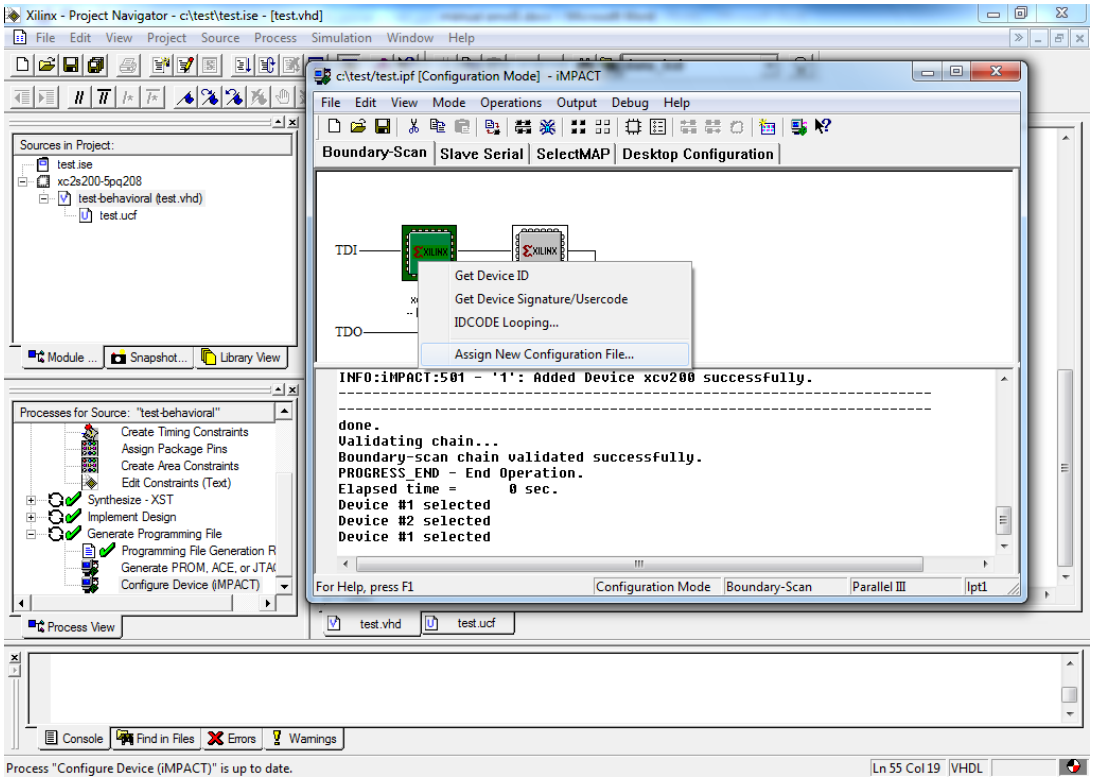
در این صورت دستگاه(های) (تراشه‌های) موجود در زنجیره‌ی Boundary-scan تشخیص داده شده و در یک صفحه‌ی کوچک تعداد آنها گزارش داده می‌شود (برای این بورد، یک تراشه‌ی FPGA و یک فلش) و پس از OK کردن این گزارش می‌توان نماد تراشه‌ها را به صورت گرافیکی مشاهده کرد.



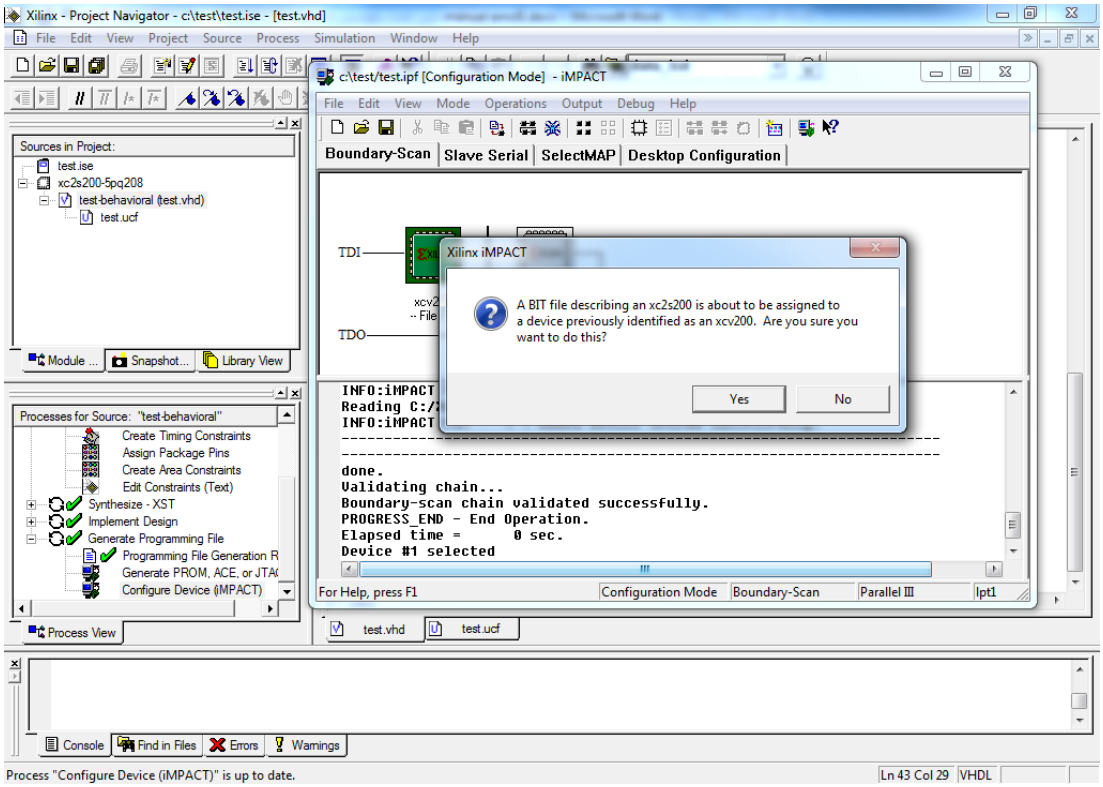
بعد از OK کردن، صفحه‌ی زیر باز می‌شود که برای انتخاب فایل برنامه‌ریزی یا همان پیکربندی (configuration) تراشه می‌باشد. در اینجا باید فایل Bitstream (فایلی با نام فایل VHDL و با پسوند bit) را که در مراحل قبلی تولید شده و در فولدر پروژه (فولدر پیش فرضی که با باز شدن این صفحه محتویات آن نشان داده می‌شود) قرار داده شده را باز (open) کنید. این همان فایلی است که اطلاعات پیکربندی در آن قرار دارد و باید با یکی از روش‌ها (در اینجا JTAG) به FPGA فرستاده شود (آن را پروگرم کند).



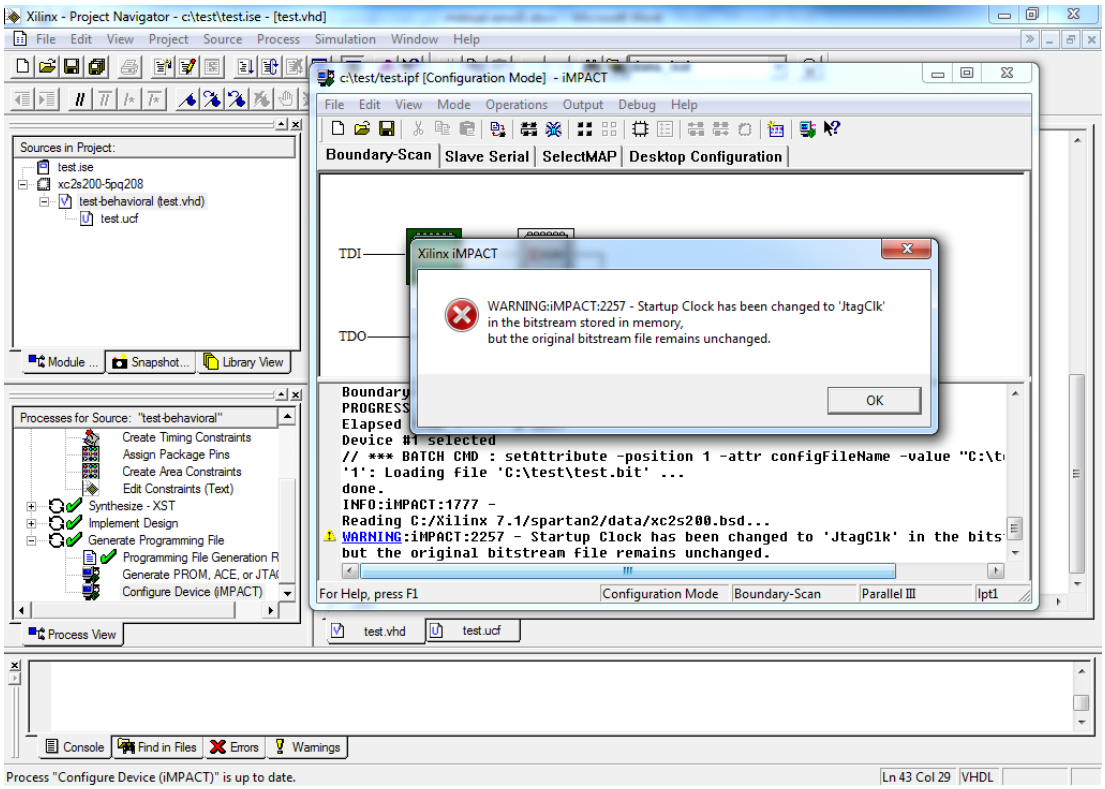
گاهی اوقات هم صفحه‌ی فوق‌الذکر به صورت اتوماتیک باز نمی‌شود که در این حالت باید روی نماد FPGA کلیک راست کنید و از گزینه‌ی آخر آن Assign New Configuration File... را انتخاب کنید. در این صورت همان صفحه‌ی شکل قبل باز می‌شود.



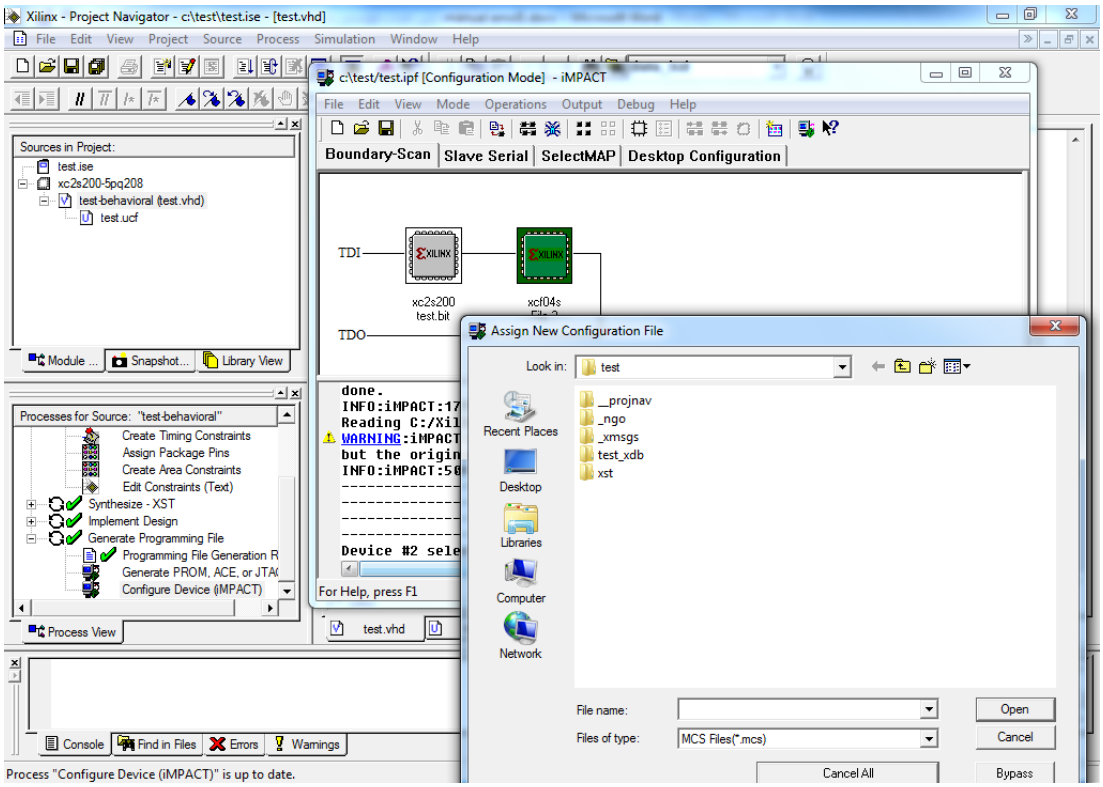
در بعضی از بوردها از تراشه‌های FPGA استفاده شده که اطلاعات نامشان را به کامپیوتر درست منتقل نمی‌کنند (اکثراً به جای XC2S200، XCV200 را به عنوان نام خود می‌دهند) در این صورت پیغام زیر نشان داده می‌شود که به راحتی می‌توان با زدن کلید Yes مشکل را حل کرد. (البته در انتهای عمل پروگرم کردن هم در صورتی که Verify را فعال کرده باشید در حالت فوق‌الذکر یک پیغام تفاوت جزئی که مربوط به همان نام تراشه است داده می‌شود که آن هم مهم نیست)



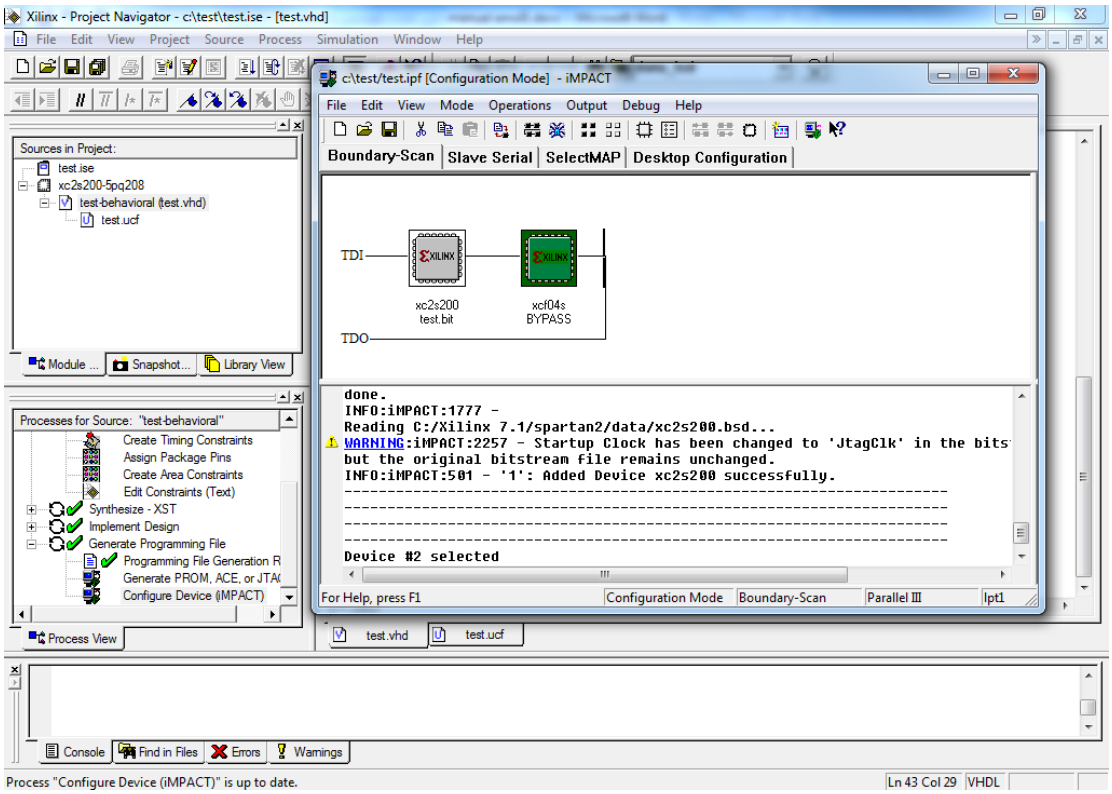
بعد از انتخاب فایل یک پیغام warning در مورد Clock مربوط بـه JTAG داده مـی‌شـود کـه مهـم نیست و می‌توانید آن را OK کنید.



بعد از انتخاب فایل پیکربندی FPGA، نوبت به فلش می‌رسد که صفحه‌ی زیر باز می‌شود و از شما فایل مربوط به فلش را می‌خواهد که فایلیست با پسوند mcs که باید به طریقی که بعداً گفته می‌شود ساخته شود. در نتیجه در حال حاضر چون هنوز این فایل تولید نشده و وجود ندارد از این مرحله (پروگرم کردن فلش) می‌گذریم. یعنی فلش را پروگرم نمی‌کنیم که در پروگرمینگ اصطلاحاً می‌گویند آن را Bypass می‌کنیم. پس واضح است که در اینجا باید کلید Bypass را بزنید.

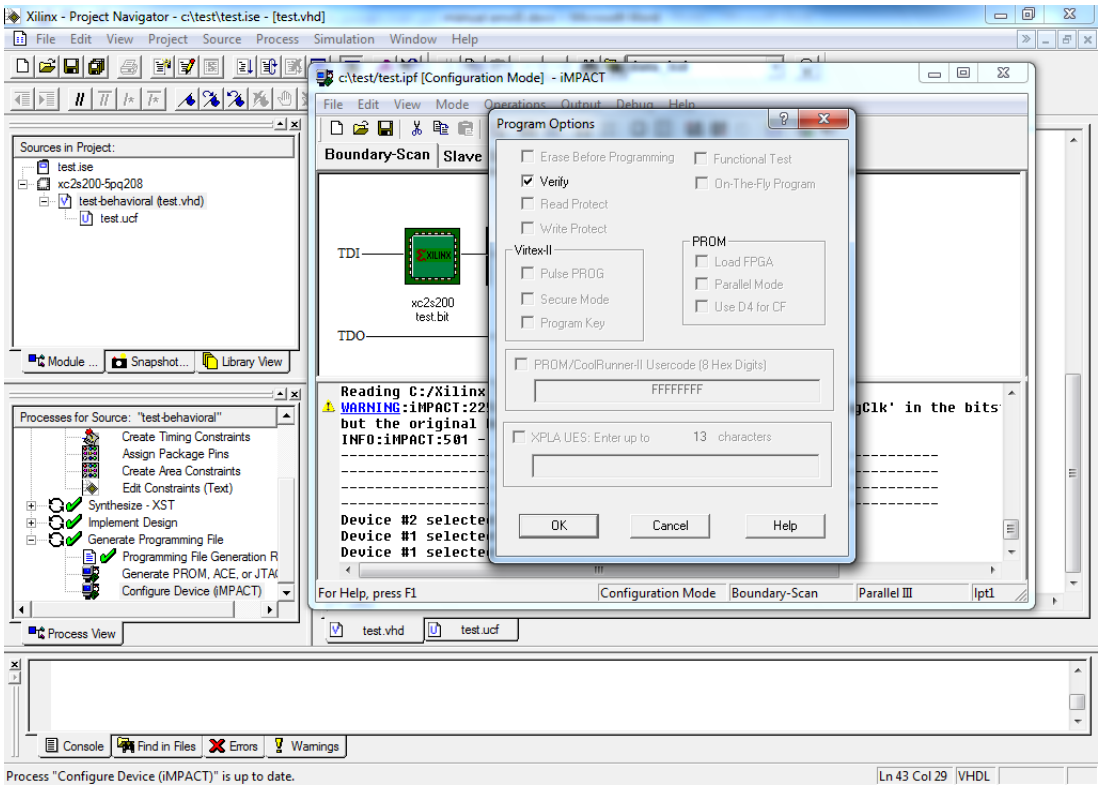


پس از آن فایل‌های پروگرمینگ هر دو تراشه در زیر نمادهای آنها نوشته می‌شود (در اینجا test.bit برای FPGA و BYPASS برای فلش (xcf04s)).



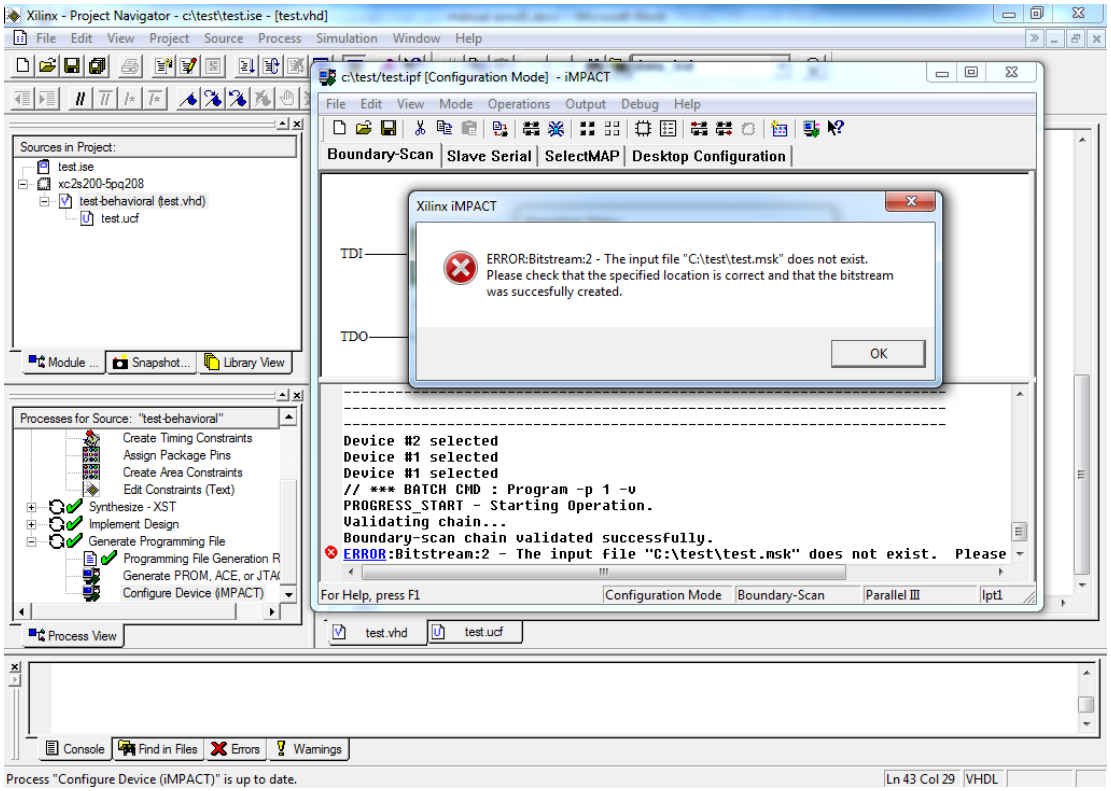
حال باید FPGA را پروگرم کنیم. برای این کار باید روی نماد FPGA راست کلیک کنید و گزینه‌ی Program را انتخاب کنید. صفحه‌ی زیر باز می‌شود که در آن گزینه‌ی Verify را که به طور پیش فرض فعال است را می‌توان فعال یا غیر فعال کرد. فعال کردن آن بدین معناست که بخواهیم بعد از عمل پروگرم کردن، با خواندن محتویات FPGA چک کند که این محتویات، با فایل پروگرم دقیقاً یکسان است یا خیر. با پروگرم کردن، اطلاعات Bitstream به FPGA فرستاده می‌شود و در صورت نداشتن مشکل در ارسال، پیغام Programming Succeeded داده می‌شود و در صورت به وجود آمدن هر گونه مشکلی، پیغام Programming Failed داده می‌شود. توجه کنید که به دلیل حساس بودن انتقال اطلاعات به نویز، کابل فلت پروگرمر را از دستگاه‌های قوی نویز ساز مثل کابل برق یا کیس کامپیوتر دور کنید، در غیر اینصورت ممکن است به صورت تصادفی بعضی از پروگرم کردن‌ها

درست انجام نشود (پیغام Programming Failed داده شود). در اینصورت باید عمل پروگرم را تکرار کنید تا به صورت درست انجام شود.

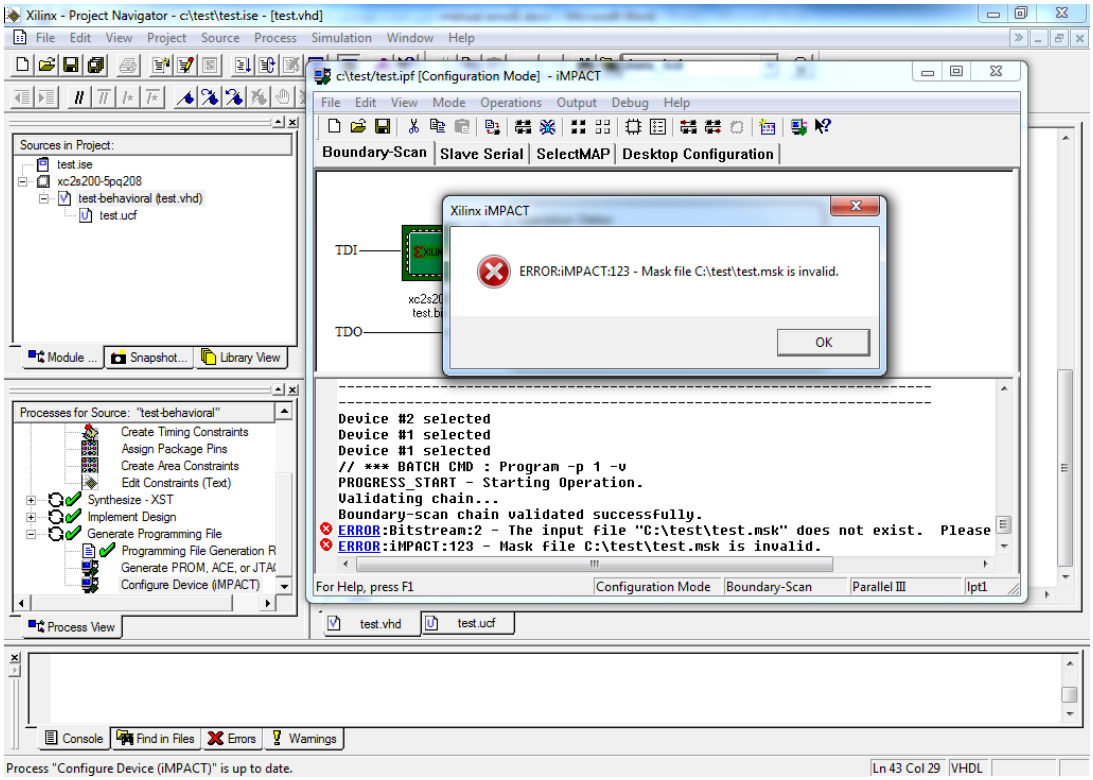


بعد از پروگرم کردن FPGA، بلافاصله کارش را شروع می‌کند.

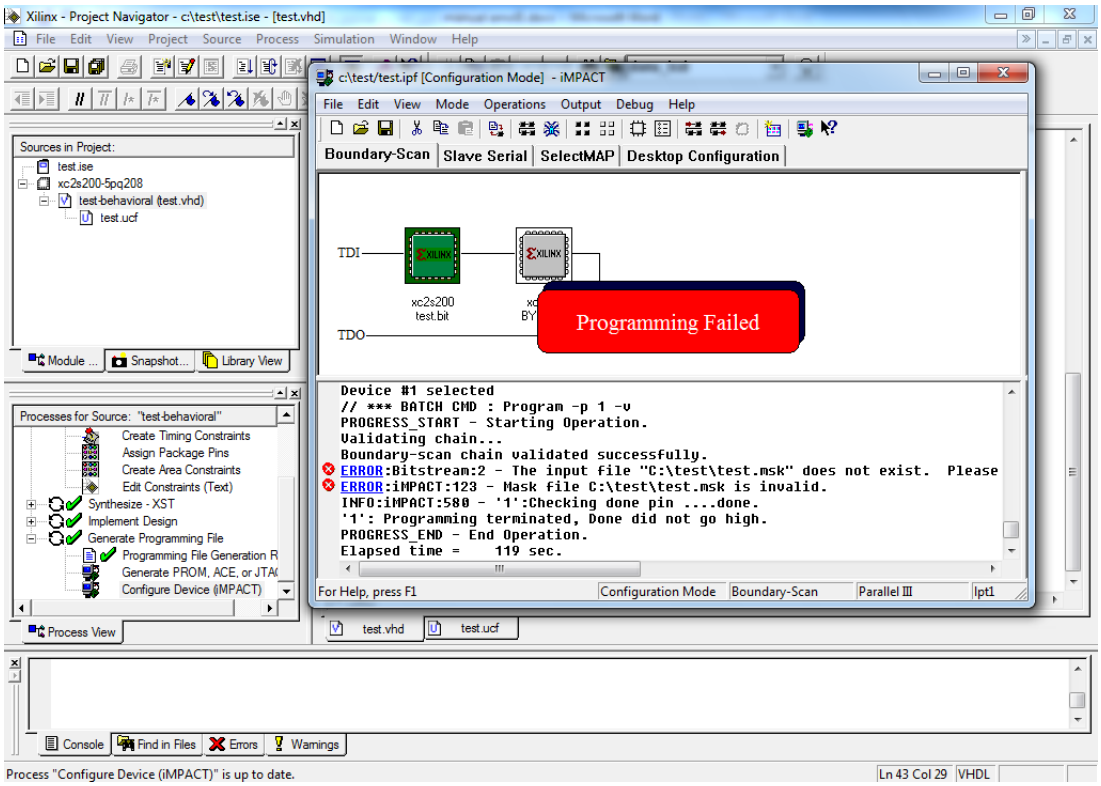
حال توجه شما را به یک نکته‌ی مهم جلب می‌کنیم. در صورتی که فقط یک بار بعد از نصب نرم‌افزار ISE، مراحل پروگرم کردن فلش را طی کرده باشید، بعد از زدن OK پنجره‌ی پروگرم، پیغام خطای زیر مبنی بر اینکه فایلی به نام mask برای پروژه وجود ندارد به شما داده می‌شود و اصلاً کار پروگرمینگ شروع نمی‌شود.



بعد از OK کردن صفحه‌ی فوق، پنجره‌ی زیر نشان داده می‌شود.

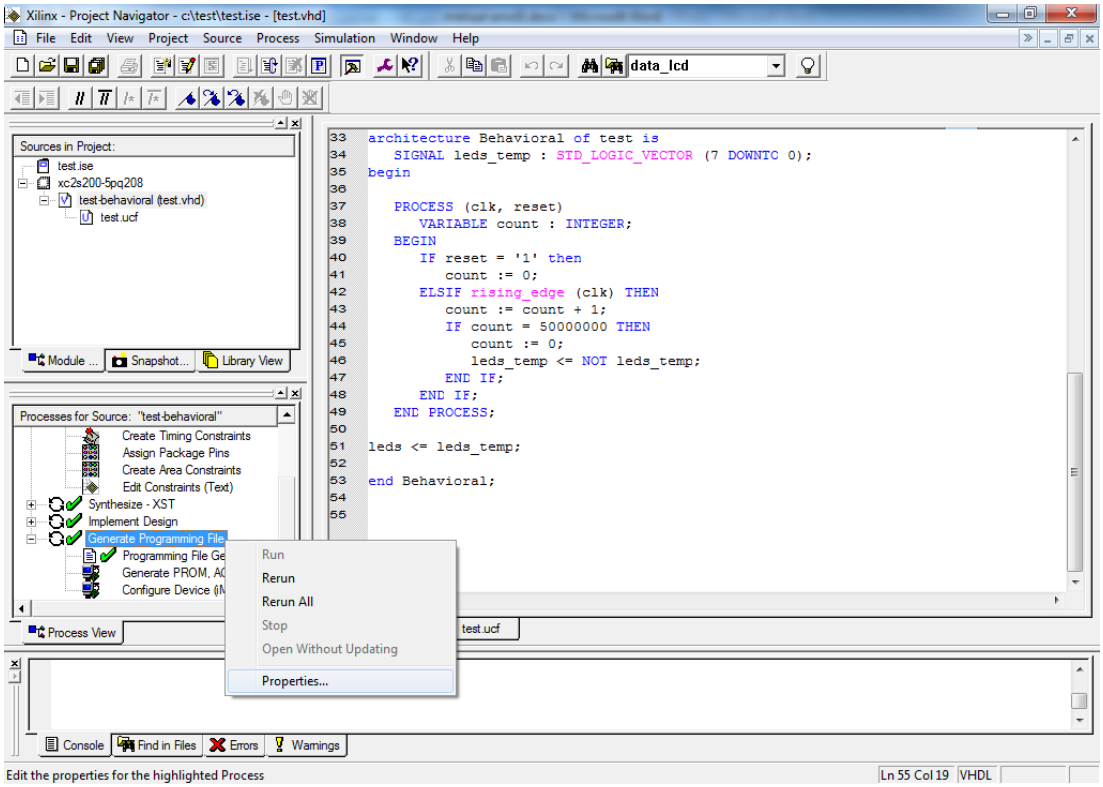


و پس از OK کردن آن، پیغام Programming Failed نشان داده می‌شود.

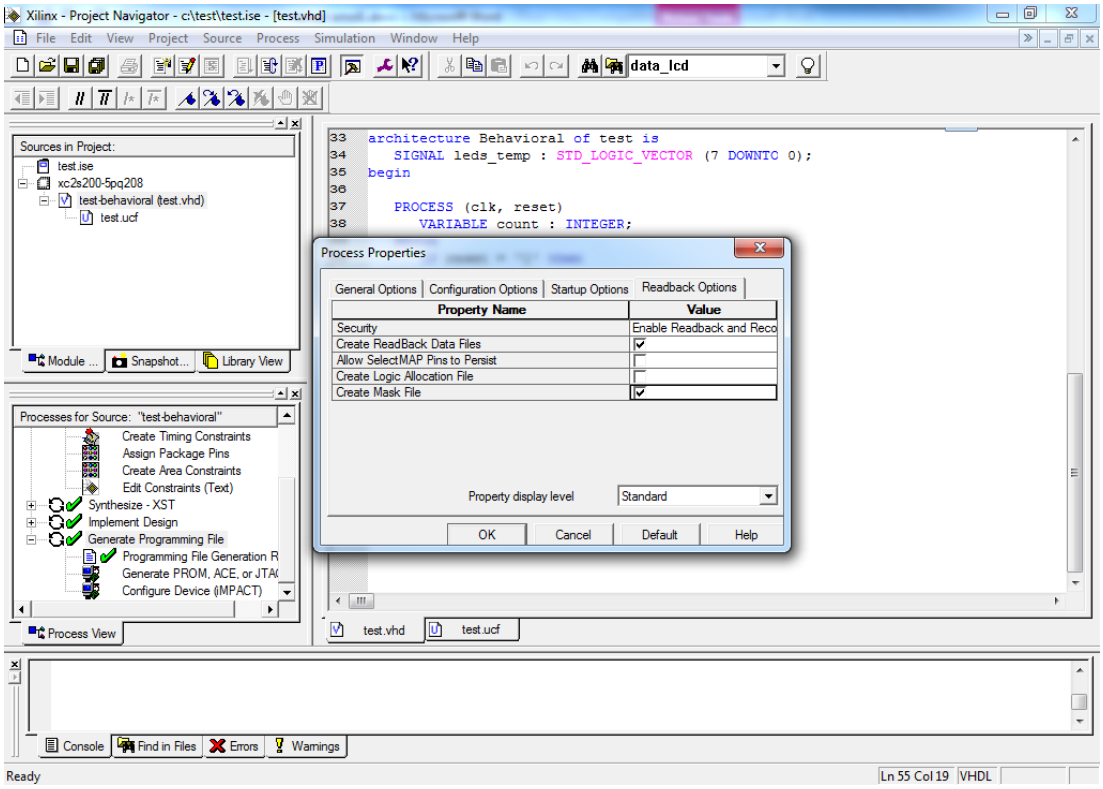


البته اگر مطابق با روال توضیح داده شده تا اینجا پیش آمده باشید (چون هنوز فلش را پروگرم نکرده‌اید) در این مرحله با این خطا مواجه نمی‌شوید ولی کافیست که فلش را یک بار پروگرم کنید (طبق توضیحات مربوطه که کمی بعد خواهیم داد)، پس از آن برای هر پروژه که برای بار اولش می‌خواهد روی FPGA پروگرم شود، این پیغام خطا را می‌بینید که برای رفع آن باید کارهایی که می‌گوییم را انجام دهید تا برای بارهای بعدی دیگر خطایی (فقط برای همان پروژه) داده نشود.

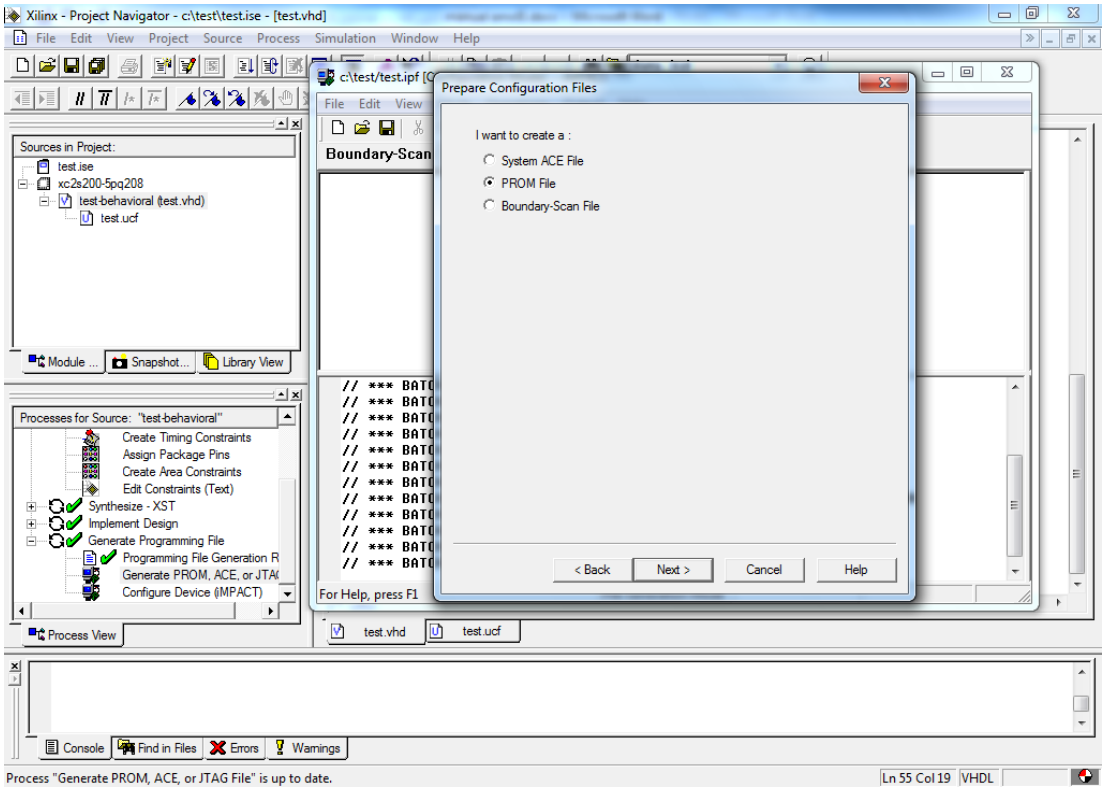
در قسمت پایین سمت چپ صفحه‌ی اصلی برنامه، روی گزینه‌ی Generate Programming File، راست کلیک کنید و در لیست باز شده، گزینه‌ی Properties... را انتخاب کنید.



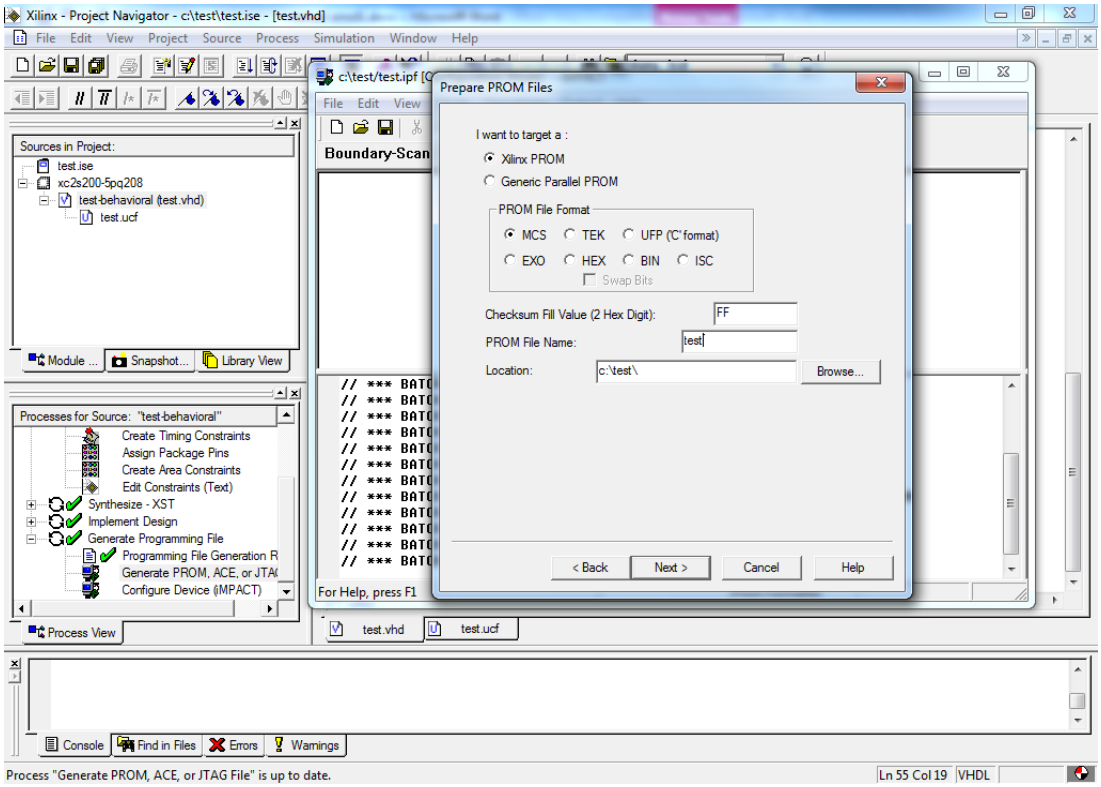
در زبانه‌ی آخر پنجره‌ی باز شده (Readback Options)، جلوی گزینه‌ی Create ReadBack Data Files و پس از آن گزینه‌ی Create Mask File تیک بزنید و OK کنید. سپس باید دوباره زیرگزینه‌ی Configure Device (iMPACT) را از گزینه‌ی Generate Programming File اجرا کنید و مطابق روالی که در صفحات پیش توضیح داده شده تراشه را پروگرم کنید. از این پس هم دیگر با خطای فوق‌الذکر برای پروژه‌ای که این کار را برایش انجام دادید مواجه نمی‌شوید. واضح است که برای پروژه‌های دیگر باید جداگانه این کار را انجام دهید.



برای کار با فلش باید در ابتدا فایل مربوط به پروگرم کردن فلش را بسازید. برای این کار زیرگزینه‌ی Generate PROM, ACE, or JTAG File را از گزینـه‌ی Generate Programming File اجـرا کنید. در صفحه‌ای که باز می‌شود، گزینه‌ی وسط یعنی PROM File را انتخاب کرده و کلید Next > را بزنید.

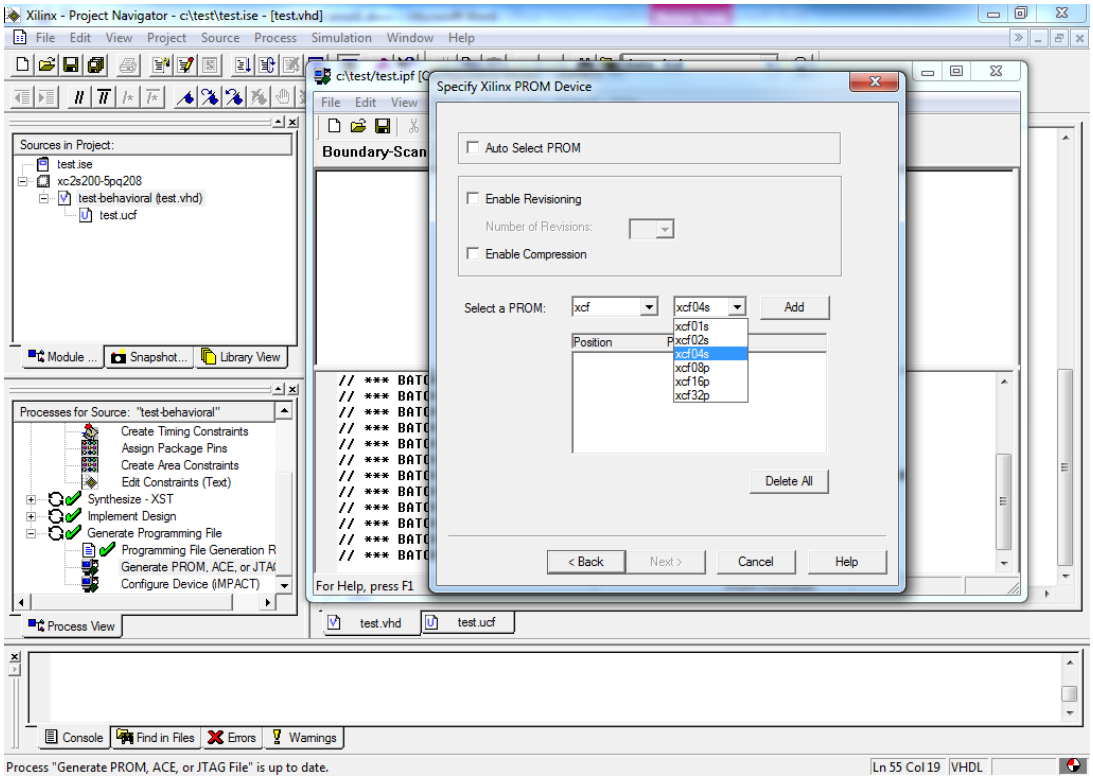


در صفحه‌ی بعدی در بالای صفحه، گزینه‌هـای Xilinx PROM و MCS (از PROM File Format) را که پیش‌فرض هم هستند را انتخاب کنید و در قسمت PROM File Name، جـای Untitled، نـام فایلی که قرار است ساخته شود تا با آن فلش را پروگرم کنیم را بنویسید در غیر اینصورت به صورت پیش فرض نام فایل همان Untitled در نظر گرفته می‌شود. ما در اینجا همان test را که نـام پـروژه هم هست در نظر گرفته‌ایم. پسوند فایل هم در هر صورت MCS است. اگر خواستید حتی مـی‌توانیـد مسیر فایل پروگرمینگ فلش را (در قسمت :Location) تغییر دهید.

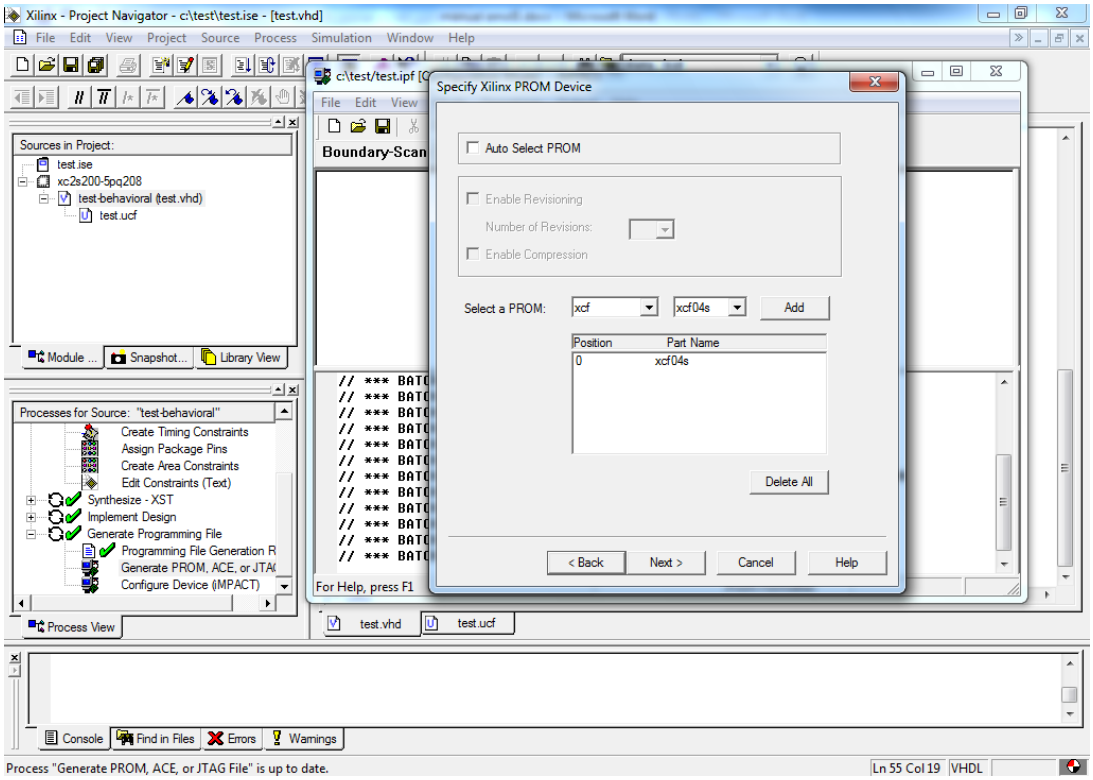


کلید Next > را بزنید.

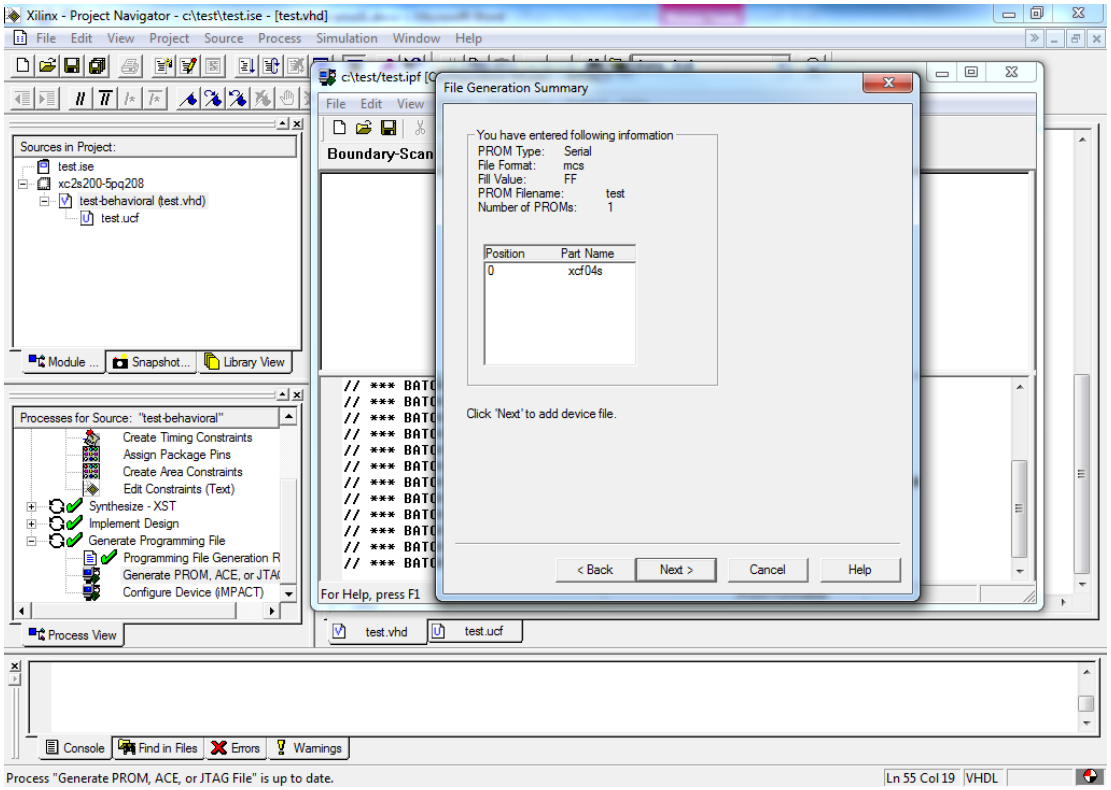
در صفحه‌ی بعدی در قسمت Select a PROM: از لیست حافظه‌های موجود، xcf04s را کـه همـان فلش این بورد می‌باشد انتخاب کنید و کلید Add را بزنید (اینجا با بقیه‌ی گزینه‌ها کاری نداریم).



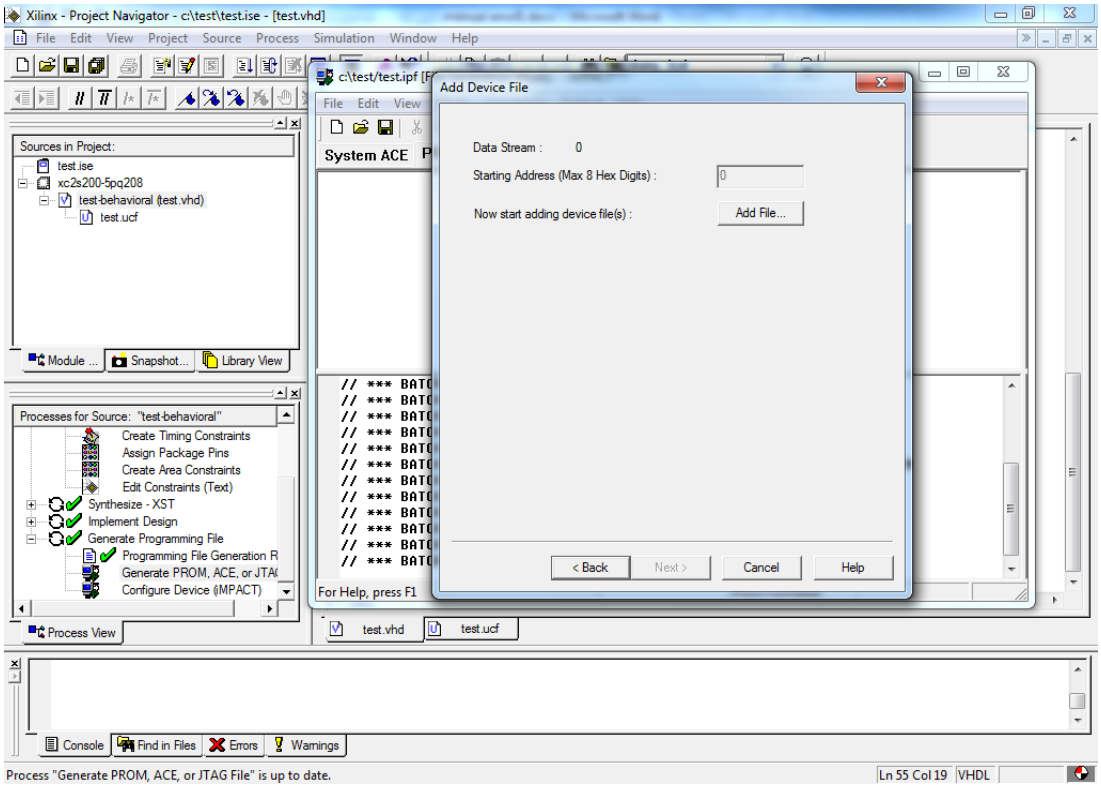
صفحه به صورت زیر در می‌آید. کلید < Next را بزنید.



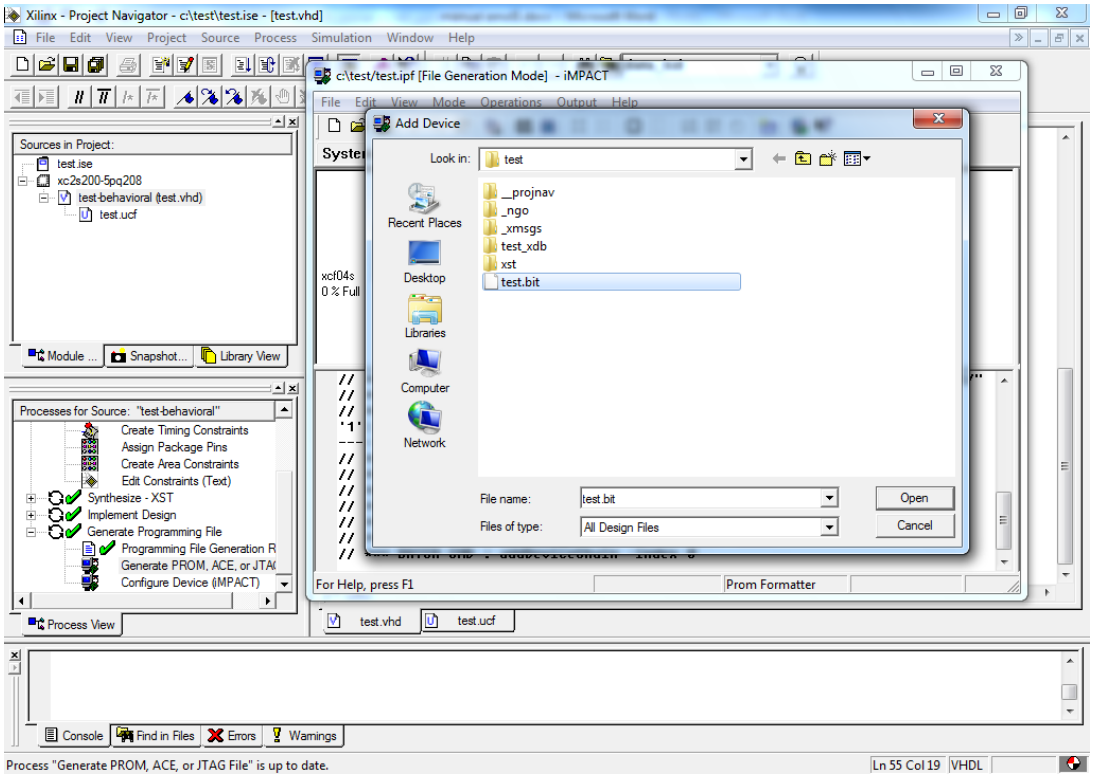
در اینجا اطلاعاتی که شما وارد کرده‌اید به شما گزارش داده می‌شود که در صورت تأیید گزینه‌ی Next > را می‌زنید و در صورتی که اشتباهی انجام داده‌اید می‌توانید با زدن کلید Back < به صفحات قبل برگردید و اشتباه را اصلاح کنید.



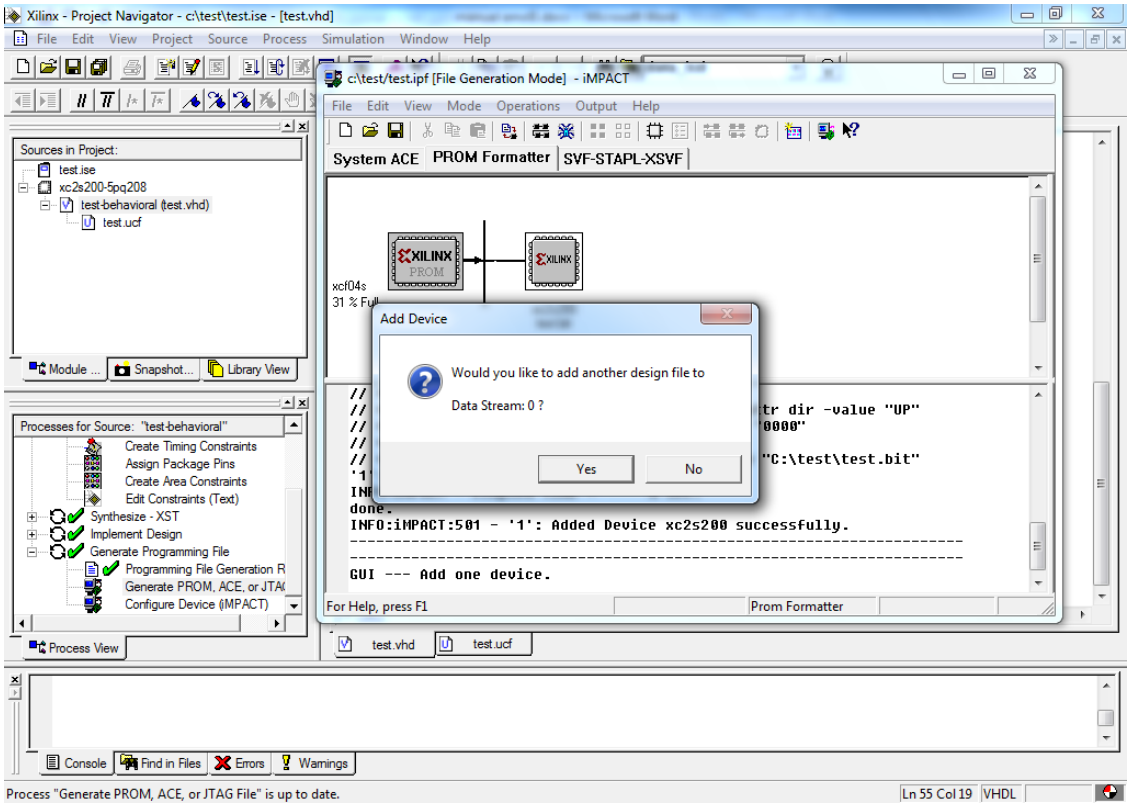
در اینجا باید فایل Bitstream اصلی را که قبلاً ساخته‌اید به نرم‌افزار بدهید تا از روی آن برایتان فایل پروگرمینگ فلش را بسازد. برای این کار در این صفحه کلید Add File… را بزنید.



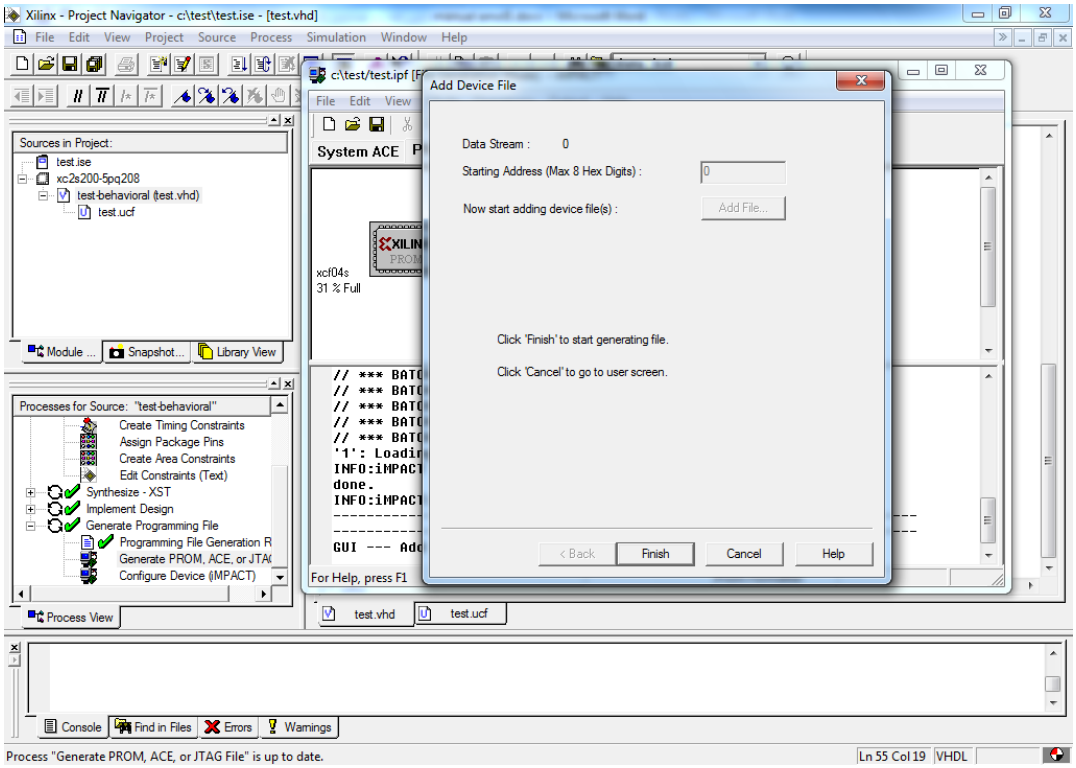
در صفحه‌ی باز شده، فایل فوق‌الذکر (در اینجا test.bit) را انتخاب کرده و Open کنید.



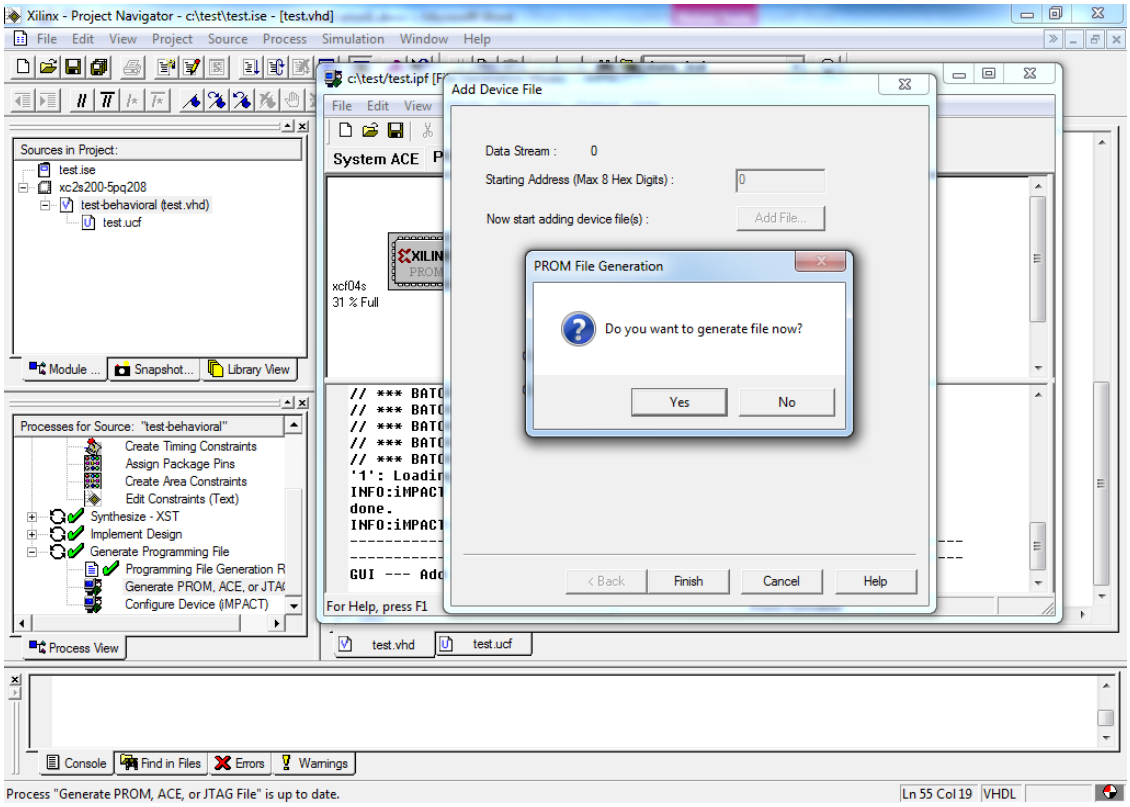
در صفحه‌ای که باز می‌شود از شما خواسته می‌شود که اگر باز هم فایل Bitstream دیگری می‌خواهید اضافه کنید این کار را انجام دهید که در اینجا چون فایل دیگری نداریم کلید No را می‌زنیم.



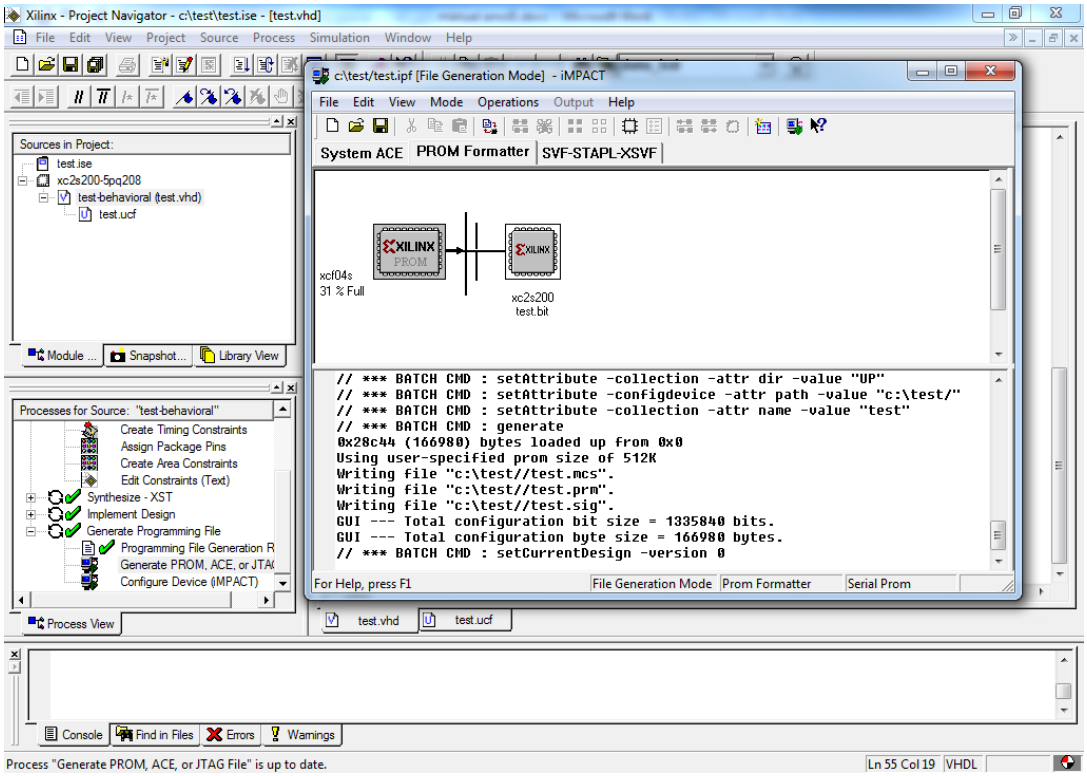
در صفحه‌ی باز شده، کلید Finish را بزنید.



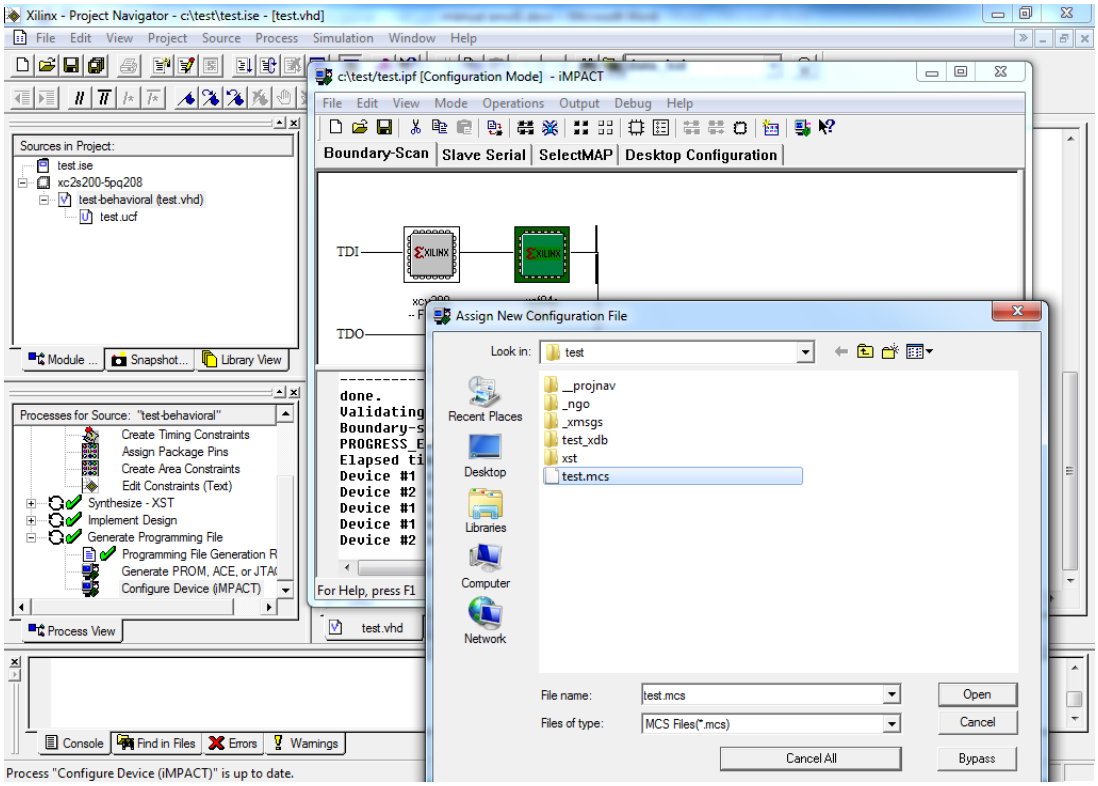
و در اینجا از شما خواسته می‌شود که آیا می‌خواهید هم اکنون فایل تولید شود که باید کلید Yes را بزنید.



در صورت عدم وجود مشکل، فایل ساخته می‌شود و در صفحه‌ی زیر به صورت گرافیکی نشان داده می‌شود که در فلش چه مقدار اطلاعات قرار است ذخیره شود. می‌توانید این صفحه را ببندید.



حال در صفحه‌ی اصلی می‌توانید فلش را پروگرم کنید. برای این کار روی نماد فلش کلیک راست کرده و گزینه‌ی Assign New Configuration File... را اجرا کنید. سپس فایل تولید شده برای پروگرمینگ فلش (در اینجا test.mcs) را انتخاب کنید و آن را Open کنید.

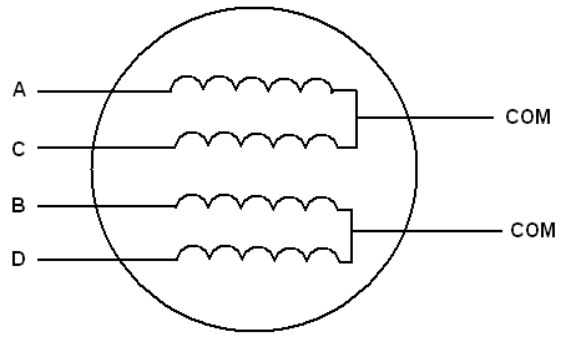


سپس روی نماد فلش دوباره کلیک راست کنید و Program را اجرا کنید. با کمی تـأخیر در صـورت نداشتن هیچ مشکلی فلش پروگرم شده و پیغام موفقیت داده می‌شود. در صـورت داده شـدن پیغـام خطا به دلیل ذکر شده در قبل مبنی بر اینکه عمل پروگرمینگ به واسطه‌ی داشتن کابـل بلنـد و در معرض نویز قرار داشتن، حساس می‌باشد دوباره عمل پروگرمینگ را تکرار کنید.

۲- اتصال موتور پله‌ای به FPGA

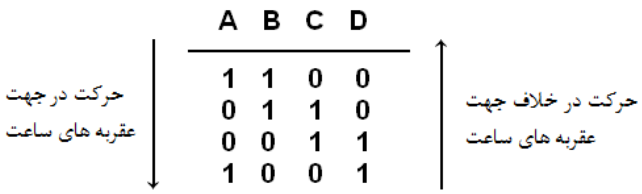
موتورهای پله‌ای معمولاً دارای دو سیم پیچ هستند. از سر وسط هر یک از این دو سیم پیچ هم اتصالی خارج شده است. پس این موتورها در مجموع شش سر دارند. دو سر وسط به نام COM نامیده می‌شوند و سرهای کناری A، B، C و D نام دارند.

همان طور که از شکل ۱ پیدا است به سادگی می‌توان سرهای موتور پله‌ای را با یک اهم متر تشخیص داد. مقدار مقاومت بین دو سر A و C دو برابر هر یک از مقاومت‌های بین سرهای A و COM یا C و COM است. همین امر برای سرهای B و D برقرار است. همچنین سیم پیچ‌های AC وBD به هم هیچ گونه اتصال مقاومتی ندارند. با این کار جفت سیم پیچ‌های AC از BD تشخیص داده می‌شوند. در ضمن سرهای A و C و همچنین سرهای B و D مثل هم عمل می‌کنند. یعنی می‌توان آنها را جای هم به کار برد.



شکل ۱ سیم‌پیچ‌های داخل موتو پله‌ای

برای این که موتور بتواند حرکت کند (به اندازه یک پله) باید تغییرات مقادیر سیم پیچ‌ها به صورت شکل ۲ به پایه‌های A، B، C و D داده شود. همچنین سرهای COM به هم و به منبع تغذیه موتور وصل می‌شوند. ولتاژ تغذیه موتور هم روی آن نوشته شده است.

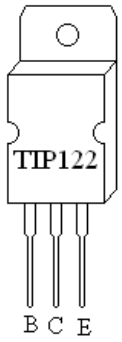


شکل ۲ نحوه اعمال اطلاعات به سیم‌پیچ‌های موتور

زاویه هر پله با تقسیم عدد ۳۶۰ بر تعداد پله در دور به دست می‌آید. برای نمونه، برای تعداد پله در دور ۴۸، زاویه هر پله 7.5^{o} می‌شود. معمولاً یکی از مقادیر تعداد پله در دور یا زاویه هر پله روی بدنه موتور نوشته می‌شود.

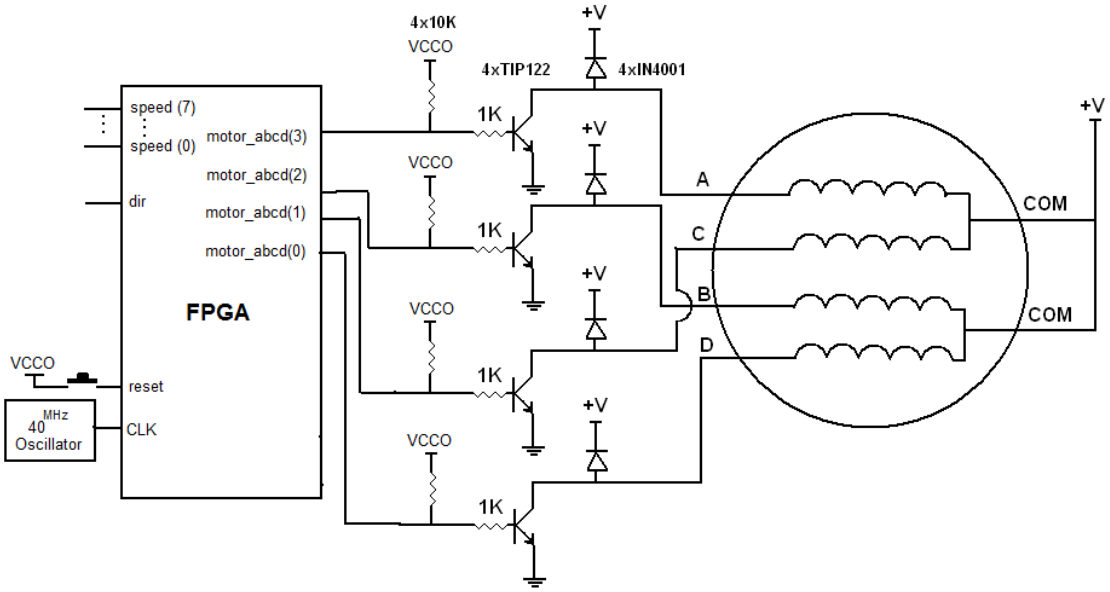
سرهای سیم پیچ‌های موتورهای پله‌ای معمولاً با رنگ‌های مختلف مشخص می‌شوند. نکته دیگری که باید در موتورهای پله ای به آن توجه کرد، مقدار جریانی است که موتور (یا هر سیم پیچ آن) می‌کشد. این مقدار یا مقدار مقاومت سیم پیچ‌ها، روی بدنه موتور نوشته می‌شود. اگر جریانی که موتور می‌کشد زیاد باشد، برای راه اندازی سیم پیچ‌ها باید از ترانزیستورهای نیمه قدرت (مثلاً ترانزیستور TIP122) استفاده کرد.

شکل ۳ هم پایه‌های ترانزیستور TIP122 را نشان می‌دهد. به جای ترانزیستور TIP122، می‌توانید از ترانزیستور 2N2219 هم استفاده نمایید. فقط دقت کنید که این ترانزیستورها زود داغ می‌شوند.



شکل ۳ پایه‌های ترانزیستور TIP122

مدار شکل ۴ اتصالات را نشان می‌دهد. در شکل ۴، V+ به منبع تغذیه موتور (نه لزوماً منبع تغذیه FPGA) وصل می‌شود. در صورتی که ترانزیستورها زیاد داغ می‌کنند، به جای دیودها از مقاومت‌های 2.2K استفاده کنید. اکنون مطابق شکل ۴ موتور پله‌ای را به FPGA وصل کنید.



شکل ۴ مدار اتصال موتور پله‌ای به FPGA

یک برنامه ساده برای چرخاندن موتور در دو جهت (تعیین توسط ورودی dir از FPGA) به طور پیوسته و سرعت متغیر (تعیین توسط ورودی speed) به صورت زیر است:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity stepper_motor is
   Port ( clk : in std_logic;
          reset : in std_logic;
          dir : in std_logic;
          speed : in std_logic_vector(7 downto 0);
          motor_abcd : out std_logic_vector(3 downto 0));
end stepper_motor;

architecture Behavioral of stepper_motor is
   signal motor_abcd_i : std_logic_vector(3 downto 0);
   signal  count : std_logic_vector(23 downto 0);
```

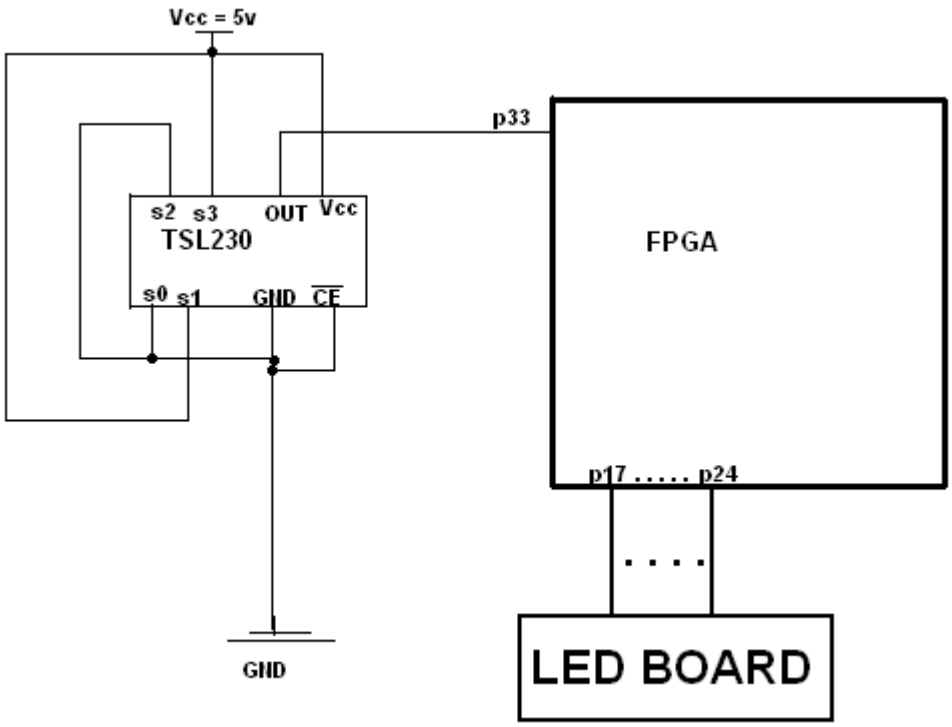
```
begin
   process (clk, reset)
   begin
      if reset = '1' then
         motor_abcd_i <= "0011";
         count <= (others => '0');
      elsif clk = '1' and clk'event then
         if count < speed & "0000000000000000" then
            count <= count + 1;
         else
            if dir = '0' then
               motor_abcd_i <= motor_abcd_i(2 downto 0) &
                               motor_abcd_i (3);
            else
               motor_abcd_i <= motor_abcd_i (0) &
                               motor_abcd_i(3 downto 1);
            end if;
            count <= (others => '0');
         end if;
      end if;
   end process;

   motor_abcd <= motor_abcd_i;

end Behavioral;
```

۳- اتصال TSL230 (سنسور نور) به FPGA

هدف آزمایش

در این آزمایش با اعمال نور به سنسور مربوطه نتایج را بر روی LED مشاهده مـی کنـیم لازم به ذکر است که با اعمال نور خروجی TSL بصورت فرکانس می باشد.

شماتیک مدار:



برنامه:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_unsigned.ALL;
--=========================================================
entity tsl230 is port (dataout :out std_logic_vector(11 downto 0)
:="000000000000";reset:in STD_LOGIC ;data:in STD_LOGIC;clk:in
std_logic);
end tsl230;
--=========================================================
```

```
architecture dlaymachin of tsl230 is
   signal tmer: integer:=0;
   signal dataold : STD_LOGIC :='1';
   signal sh : integer :=0 ;
BEGIN
process (clk)
  begin
       if clk = '1' and clk'event then
              tmer <= tmer + 1;

       if (data = '1' and dataold = '0') then
           sh <= sh + 1 ;
           dataold <= '1';

       end if ;
       if (data= '0' and dataold = '1' ) then
               dataold <= '0';
       end if ;
       if (tmer = 20000000 )  then
         for i in 0 to dataout'length - 1 loop
              dataout(i) <= '0' ;
              if (sh > 2**i) then
                 if i>4 then
                    dataout(i-5) <= '1' ;
                 end if ;
              end if;
          end loop;
       end if ;

       if (reset='1')   then  tmer<=0;sh<=0; end if;
       if (tmer = 20000010)   then  tmer<=0;sh<=0 ; end if;
       end if ;
end process;
end dlaymachin;
```

برنامه‌ی دیگری از tsl230:

```
library ieee;
use ieee.std_logic_1164.all;
entity tsl230 is
         port ( clk,data,reset : in std_logic ;
                   output : out std_logic_vector ( 11 downto 0 );
                   flag : inout std_logic);
end tsl230;
architecture behavioral of tsl230 is
type state is ( start, compare);
signal current : state;
type freque is array( 0 to 12 ) of std_logic_vector(11 downto 0);
constant table : freque := ( "000000000000" , "100000000000" ,
"110000000000" , "111000000000" , "111100000000" , "111110000000" ,
"111111000000" , "111111100000" , "111111110000" , "111111111000" ,
"111111111100" , "111111111110" , "111111111111");
begin
process (clk,reset)
variable count,counter,timer,i : integer;
begin
```

```
if reset = '1' then
       counter := 0;
       output <= "000000000000";
elsif clk = '1' and clk'event  then
timer := timer +1;
if timer = 500000 then  -- در مدت ۵۰۰۰۰۰ کلاک عمل شمارش انجام میشود
      count := counter;
      counter := 0;
      current <= compare;
elsif timer = 507000 then
      timer := 0;
end if;
case current is
when start =>
i := 0;
if data = '1' and flag = '1' then                -- تشخیص خروجی سنسور
       counter := counter +1;
       flag <= '0';
elsif data = '0' and flag = '0' then
      flag <= '1';
end if;
when compare =>
if count < 2**i then         -- برای اینکه خروجی اعدادی از توان ۲ را
 نشان دهد ( اعداد داخل جدول)
      output <= table(i);
      current <= start;
else
      current <= compare;
      i := i +1;
end if;
end case;
end if;
end process;
end behavioral;
```

۴- دور شمار موتور

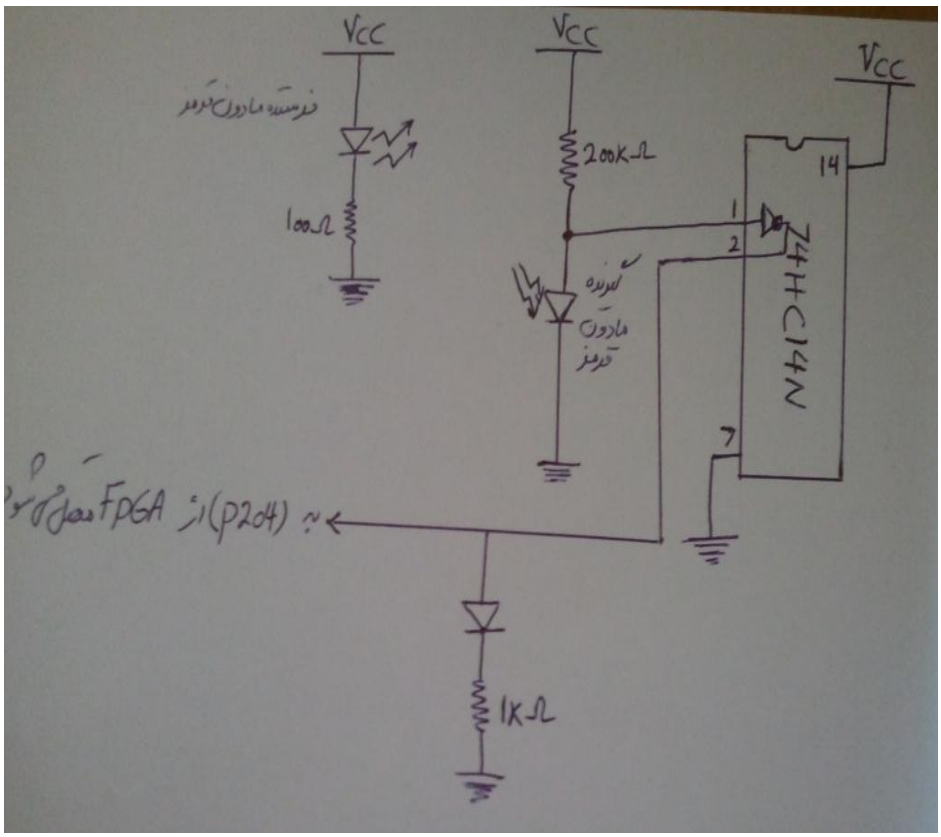
هدف از بستن این مدار اندازه گیری تعداد دورهای یک موتور DC در زمانی معـین مـثلا یـک ثانیـه است.

توضیحات مدار

این مدار از دو بخش جداگانه که یکی برای تشخیص گردش موتـور توسـط سنسـورهای فرسـتنده وگیرنده است که روی برد بورد بسته شده است و بخش دیگر شمارش تعداد دور های موتـور توسـط برنامه نویسی در FPGA است .

در بخش برد بورد پره ای که به موتور وصل شده است با عبور از بین سنسـور هـا کـه شـکل ان در ادامه امده است باعث ایجاد یک پالس میشود که این پالس برای شمارش به FPGA داده میشود.

شماتیک مدار:



برنامه:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY counter IS
PORT(clk,input,reset: in std_logic;
sev_seg:out std_logic_vector(6 downto 0);
sel :out std_logic_vector(2 downto 0);
flag :inout std_logic);
end counter;
architecture behavioral of counter is
     type state is ( start,count );
        signal current : state;
begin process (clk,reset)
        variable
digit_iii_old,digit_ii_old,digit_i_old,digit_iii,digit_ii,digit_i :
integer:=0;
        variable output : integer range 0 to 9;
        variable counter1,counter2 : integer :=0;
begin
if    reset = '1'   then
                current <= start;
                digit_iii_old:=0;
                digit_ii_old:=0;
                digit_i_old := 0;
                digit_iii := 0;
                digit_ii :=0;
                digit_i :=0;
                counter1:=0;
                counter2:=0;
elsif clk='1' and clk'event then
                counter1:=counter1+1;
                counter2:= counter2+1;
case current is
when start =>
   if input = '0' and flag = '0'  then    -- چک کردن ورودی
            flag <='1';
            current <= count;
   elsif  input = '1' and flag = '1' then
            flag <='0';
   end if;
when count =>     -- شمارش لبه های ورودی
    current <= start;
    digit_i_old := digit_i_old + 1;
    if   digit_i_old = 10   then
          digit_i_old := 0;
          digit_ii_old :=  digit_ii_old + 1;
    end if;
    if   digit_ii_old = 10  then
          digit_ii_old:=  0;
          digit_iii_old := digit_iii_old + 1;
    end if;
end case;
if counter1 = 50000000   then
      counter1:= 0;
      digit_i := digit_i_old;
      digit_ii :=digit_ii_old;
```

```
      digit_iii := digit_iii_old;
      digit_i_old := 0;
      digit_ii_old:=0;
      digit_iii_old := 0;
end if;
   if counter2 < 10000  then  -- برای سوئیچ کردن سون سگمنت ها
         output := digit_i;   -- انتساب رقم یکان به خروجی
         sel <= "001";
   elsif counter2 < 20000  then
         output := digit_ii;  --  انتساب رقم دهگان به خروجی
         sel <= "010";
   elsif counter2 < 30000 then
         output := digit_iii; -- انتساب رقم صدگان به خروجی
         sel <= "100";
   elsif counter2 = 30000 then
         counter2 := 0;
   end if;
  end if;
case output is
                when 0   => sev_seg <=   "1000000";
                when 1   => sev_seg <=   "1111001";
                when 2   => sev_seg <=   "0100100";
                when 3   => sev_seg <=   "0110000";
                when 4   => sev_seg <=   "0011001";
                when 5   => sev_seg <=   "0010010";
                when 6   => sev_seg <=   "0000010";
                when 7   => sev_seg <=   "1111000";
                when 8   => sev_seg <=   "0000000";
                when 9   => sev_seg <=   "0010000";
end case;
end process;
end behavioral;
```

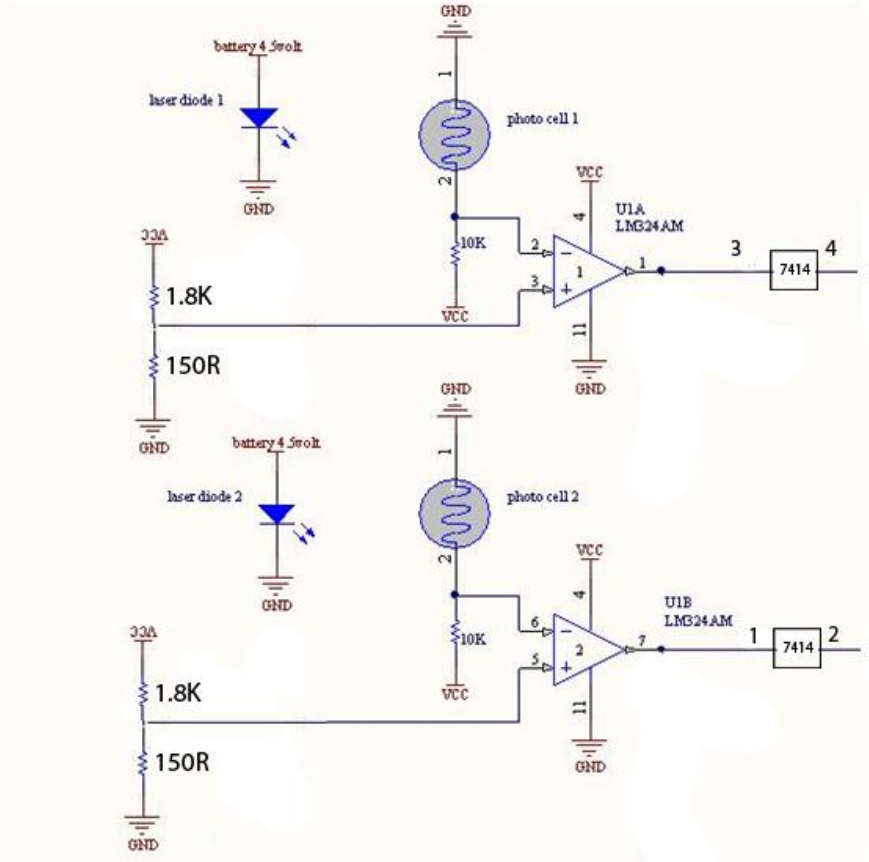
۵- شمارشگر افراد با لیزر

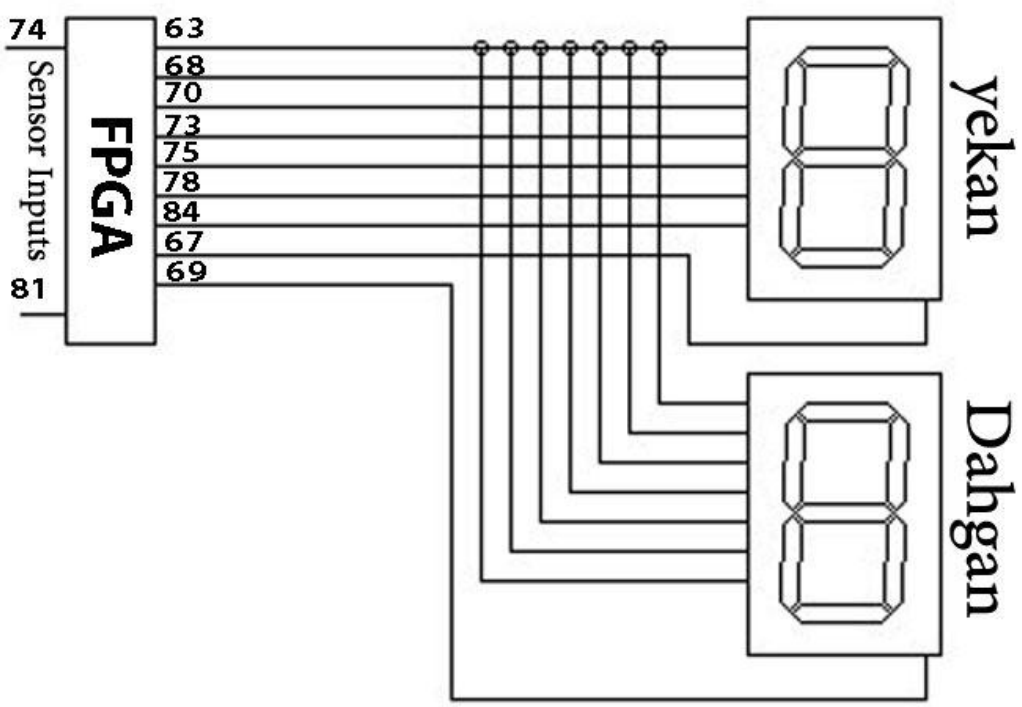
این پروژه ساخت دستگاه شمارش افراد می باشد که برای شمارش تعـداد افـراد ورودی و همچنـین خروجی از یک درگاه به کار می رود.

این دستگاه شامل ۲ لیزر می باشد که ورود و خروج افراد را مشخص می کند ، بدین صورت که اگـر ابتدا لیزر شماره ا(بیرونی) خاموش و روشن شود و سپس لیزر شماره ۲(درونی)، نشان دهنـده اضـافه شدن یک فرد و و برعکس اگر ابتدا لیزر شماره ۲ و سپس لیزر شـماره ۱ خـاموش و روشـن شـود ، نشان دهنده خروج یک فرد می باشد..

در این قسمت از قطعاتی که در این پروژه از آنها استفاده شده نام برده و اطلاعات مختصری از جمله بلوک دیاگرام های آنان مورد بررسی قرار می گیرد.

شماتیک:





برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
ENTITY counter IS
PORT (  clk,reset : in std_logic;
            sens_i,sens_o : in std_logic;
            sev_seg : out std_logic_vector ( 6 downto 0 );
            sel : out std_logic_vector ( 2 downto 0 )
             );
END counter;
ARCHITECTURE behavioral OF counter IS
TYPE state IS ( start,input,output,wait_i,wait_o,count_i,count_o);
signal current : state;
BEGIN
PROCESS ( clk,reset)
        variable digit_ii,digit_i : integer;
        variable final : integer range 0 to 9;
        variable changer,timer : integer;
BEGIN
if reset = '1' then
        current <= start;
        digit_i := 0;
        digit_ii := 0;
        final := 0;
        changer := 0;
elsif clk='1' and clk'event then
timer := timer +1;
```

```
changer := changer + 1;
if timer = 500 then
timer := 0;
case current is
when start =>
      if sens_i = '1' then
         current <= input;
      elsif sens_o = '1'then
         current <= output;
      end if;
when input =>     -- هنگامی که حالت سنسورها "۰۱" است
      if sens_o = '1' then
      if sens_i = '0' then
             current <=count_i;
      else
             current <= wait_i;
      end if;
      end if;
when output =>    --  هنگامی که حالت سنسورها "۱۰" است
      if sens_i = '1' then
      if sens_o = '0' then
            current <=count_o;
      else
            current <= wait_o;
      end if;
      end if;
when wait_i =>    --  هنگامی که حالت سنسورها از سمت ورودی "۱۱" است
   if sens_i = '0' then
         current <= count_i;
   elsif sens_o = '0' then
         current <= input;
   end if;
when wait_o =>    -- هنگامی که حالت سنسورها از سمت ورودی "۱۱" است
    if sens_o = '0' then
          current <= count_o;
     elsif sens_i = '0' then
          current <= output;
     end if;
when count_i =>   -- شمارش ورودی
if sens_o = '0' then
      current <= start;
      digit_i:= digit_i + 1;
      if   digit_i = 10   then
           digit_i := 0;
           digit_ii :=  digit_ii + 1;
      end if;
      if digit_ii > 9 then          --  هنگامی که عدد بزرگتر از ۹۹
نشود
           digit_ii := 9;
           digit_i := 9;
      end if;
end if;
when count_o =>   --  شمارش خروجی
if sens_i = '0' then
current <= start;
digit_i := digit_i - 1;
      if digit_i = -1 then
         digit_i := 9;
         digit_ii := digit_ii - 1;
      end if;
      if digit_ii < 0 then    -- هنگامی که عدد کوچکتر از ۰۰ نشود
```

```
         digit_ii := 0;
         digit_i := 0;
      end if;
end if;
end case;
end if;
   if changer < 10000  then
         final := digit_i;
         sel <= "001";
   elsif changer < 20000  then
         final := digit_ii;
         sel <= "010";
   elsif changer = 20000 then
        changer := 0;
   end if;
end if;
case final is
                 when 0   =>sev_seg <=   "1000000";
                 when 1   =>sev_seg <=   "1111001";
                 when 2   =>sev_seg <=   "0100100";
                 when 3   =>sev_seg <=   "0110000";
                 when 4   =>sev_seg <=   "0011001";
                 when 5   =>sev_seg <=   "0010010";
                 when 6   =>sev_seg <=   "0000010";
                 when 7   =>sev_seg <=   "1111000";
                 when 8   =>sev_seg <=   "0000000";
                 when 9   =>sev_seg <=   "0010000";
end case;
end process;
end behavioral;
```

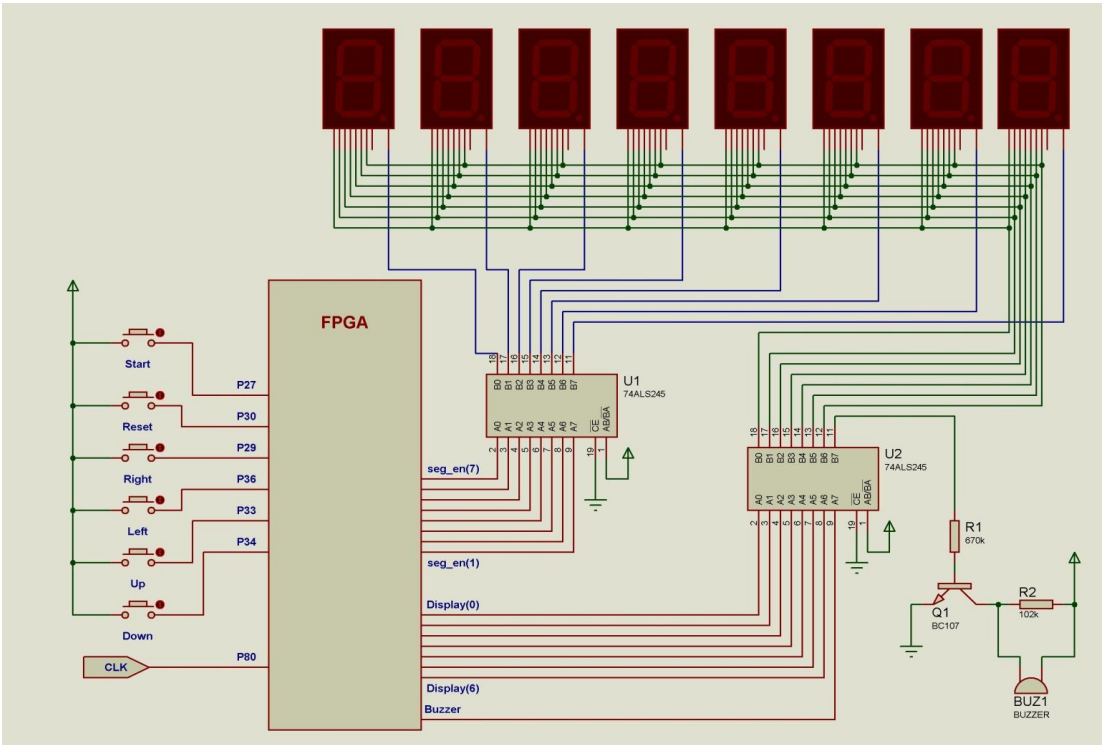
۶- آلارم

هـدف: پیاده ساز ی شمارنده معکوس (down counter) با fpga

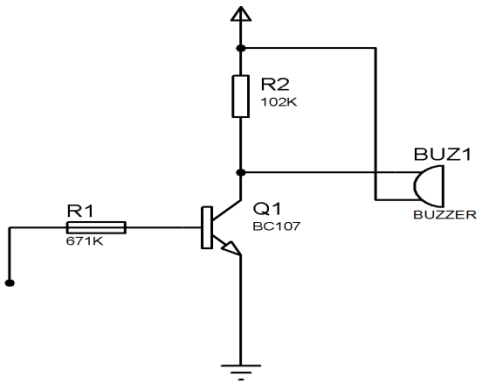
شمارنده ای است که عددی را که شامل ساعت و دقیقه و ثانیه و صدم ثانیـه اسـت بعنـوان ورودی دریافت میکند و بصورت معکوس می شمارد وقتی به صفر رسید زنگی به صدا در می آید . کارهـایی که این شمارنده معکوس می خواهد انجام دهد شامل موارد ذکر شده است :

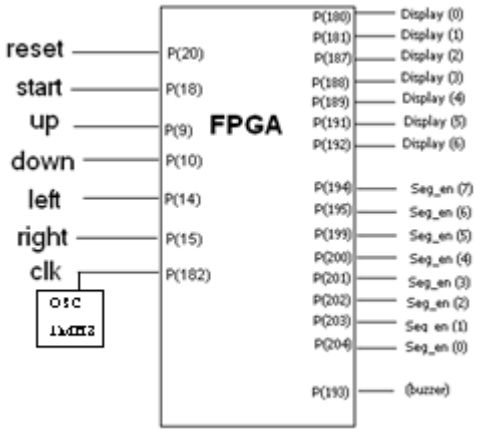
- reset (برای دادن مقدار اولیه)
- دگمه هایی برای انتخاب سون سگمنت مورد نظر
- دگمه هایی برای تنظیم عدد روی سون سگمنت مورد نظر
- ضمن تنظیم عدد روی سون سگمنت انتخابی عدد روی آن سون سگمنت چشمک زن باشد .
- دگمه ای برای شروع و توقف شمارش معکوس
- به صورت معکوس بشمارد
- در نهایت و قتی به صفر رسید buzzer به صدا در آید و اتمام کار را اعلام کند .

شکل مدار:



برای تقویت صدا buzzer پایه buzzer را به پایه base ترانزیستور وصل می کنیم.





برای تقویت صدا buzzer پایه buzzer را به پایه base ترانزیستور وصل می کنیم

توضیح برنامه:

این برنامه شامل یک entity می باشد(که پورتهای ورودی و خروجی ما در آن تعریف می شـود) و یک architecture می باشد که یک پروسس دارد که به کلاک فعال می شود در reset به آن مقـدار اولیه دادیم و زنگ را غیر فعال کردیم .

تاخیرهایی برای دگمه های start ، left، right ، up، down برای debounc(حل مشکل bounc) در نظر گرفته شد که نویزهای اضافی را در هنگام فشردن کلید به عنوان لبه در نظر نگیرد.

سیگنالهای کمکی هم به نام فلگ برای هر کدام از کلید های فشاری در نظر گرفته شد که در قسمت تعریف کار هر دگمه استفاده شد و مربوط به لبه آن است و یکبار لبه را در نظر می گیرد .

Left برای حرکت به سمت چپ روی سگمنتها و انتخاب سگمنت مورد نظر است که اگرشماره سگمنت انتخابی کوچکتر از ۹ بود به طرف چپ حرکت میکند .

Right برای حرکت به سمت چپ روی سگمنتها و انتخاب سگمنت مورد نظر است که اگرشماره سگمنت انتخابی بزرگتر از ۰بود به طرف راست حرکت میکند .

Up برای تنظیم رو به بالای عدد در صورتیکه عدد روی سون سگمنت انتخابی کوچکتر از ۹ یا ۵ باشد تعریف شده است .

Down برای تنظیم رو به پایین عدد در صورتیکه عدد روی سون سگمنت انتخابی بزرگتر یا مساوی یک باشد تعریف شده است.

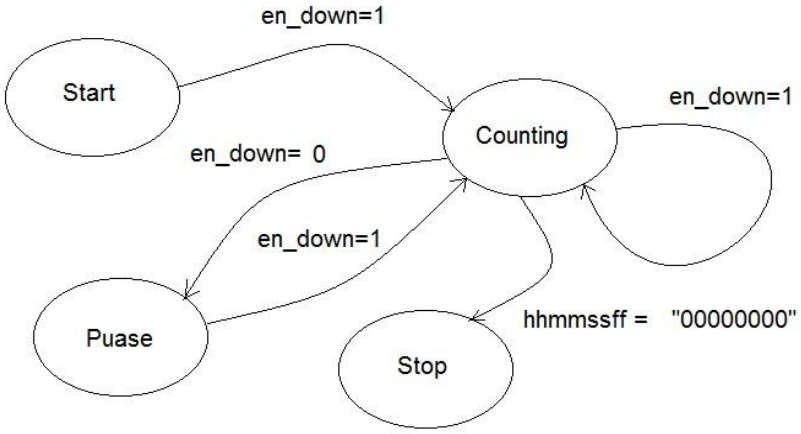
Blink برای چشمک زدن سون سگمنت انتخابی در نظر گرفته شد که هر یک چهارم ثانیه تغییر وضعیت میدهد یعنی در هر ثانیه دو بار خاموش و روشن می شود.

خطوط ۱۱۲تا ۱۷۰ در این محدوده قسمت if آن مربوط به نمایش عدد روی سون سگمنت انتخاب شده به صورت چشمک زن است و قسمت elseآن مربوط به نمایش عادی عدد است .

خطوط ۱۸۳تا ۲۰۹ کیسهایی روی ساعت ، دقیقه ، ثانیه و صدم ثانیه گذاشته شد تا بگوید با تعیین مقدار کیس چه چیزی در سیگنالهای مربوطه قرار گیرد .چون سون سگمنتها آند مشترک هستند هر قطعه روشن با صفر فعال میشود و هر قطعه خاموش با یک فعال میشود .

خطوط ۳۶۲تا ۳۶۹ دگمه شروع و توقف عملیات شمارش در حلقه ای تعریف شده است .

خط ۳۷۳ از آنجایی که هر یک صدم ثانیه باید یکی از عدد نمایشی شمارنده معکوس کم می شود باید این زمان را برای شمارش معکوس بدست آورد که این کار هم با تقسیم فرکانس بر عدد ۱۰۰ بدست می آید و از این قسمت به بعد برنامه مربوط به شمارش معکوس است و در آخر کار هم وقتی به عدد به صفر رسید buzzer را یک قرار می دهیم .



کد :

```
-- Company:
-- Engineer:
--
-- Create Date:    16:30:09 12/25/11
-- Design Name:
-- Module Name:    alarm - Behavioral
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity alarm is
port(
```

```
------ALARM Ports
start:in std_logic;     --p27
up: in std_logic;       --p33
down:in std_logic;      --p34
left:in std_logic;      --p36
right:in std_logic;     --p35
clk:in std_logic;       --p80 clk
reset:in std_logic;     --p29
buzzer:out std_logic;  --p168
display:out std_logic_vector(6 downto 0);  --p173, p175, p178, p180,
p187, p189, p192
seg_en:out std_logic_vector(7 downto 0)   --p172, p174, p176, p179,
p181, p188, p191, p193


);


end alarm;

architecture Behavioral of alarm is



--alarm signals-----------------------------------------------------

signal  counter : integer range 0 to 50;       --counteri baray
taghsim ferekans chon barname ba ferekanse 1meg kar mikard
signal            seg_f1: std_logic_vector(6 downto 0);
signal            seg_f2: std_logic_vector(6 downto 0);
signal            seg_s1: std_logic_vector(6 downto 0);
signal            seg_s2: std_logic_vector(6 downto 0);
signal            seg_m1: std_logic_vector(6 downto 0);
signal            seg_m2: std_logic_vector(6 downto 0);
signal            seg_h1: std_logic_vector(6 downto 0);
signal            seg_h2: std_logic_vector(6 downto 0);
signal            blink: std_logic_vector(7 downto 0);
signal            en_down: std_logic;
signal left_flag:std_logic:='0' ;
signal right_flag:std_logic:='0' ;
signal up_flag:std_logic:='0' ;
signal down_flag:std_logic:='0' ;
signal start_flag:std_logic:='0' ;
signal blink_flag: std_logic:='0';




begin

process(clk)


-----------alarm vars
variable c:integer ;
variable c1:integer;
variable c2:integer;
variable c3:integer;
variable c4:integer;
variable c5:integer;
```

```
variable c6:integer;
variable c7:integer;
variable c8:integer;
variable set_count:integer range 0 to 10;
variable f1:integer range 0 to 10;
variable f2:integer range 0 to 10;
variable s1:integer range 0 to 10;
variable s2:integer range 0 to 10;
variable m1:integer range 0 to 10;
variable m2:integer range 0 to 10;
variable h1:integer range 0 to 10;
variable h2:integer range 0 to 10;




begin
if(reset='1')then           --meghdar avaliye be seven segmentha va
kanter va bazer

counter <= 0 ;
blink_flag<='0';

c8 := 0;

f1:=6;
f2:=5;
s1:=4;
s2:=3;
m1:=2;
m2:=1;
h1:=0;
h2:=0;
set_count:=0;
buzzer<='0';
en_down<='0';

elsif rising_edge(clk)then

counter <= counter - 1 ; -- taghsime ferekanse baraye bedast avardane
ferekanse 1MHz az ferekanse 50 MHz

if counter=0 then


c6:=c6+1;
if c6=100000 then  --taghsime ferekanse baraye bedast avardane
ferekanse 10 Hz, yani 10 bar dar sanie in kar ra anjam midahad
c6:=0;
if (left='1' and left_flag='0') then      --harkat be samte chap
left_flag<='1';   --parcham baraye negahdarie feshorde shodane
dokmeye left, baraye inke dobare varede in block nashavad

if set_count<9 then  --agar set_count kochaktar az 9 bud set_count ra
yeki kam kon
set_count :=set_count+1;
end if;
end if;
end if;
```

```
if (left='0' and left_flag='1') then --vaghti dokme azad shod flag ra
sefr mikonim
left_flag<='0';
end if;

c2:=c2+1;
if c2=100000 then  --taghsime ferekanse baraye bedast avardane
ferekanse 10 Hz
c2:=0;
if (right='1' and right_flag='0') then     --harkat be samte rast
right_flag<='1'; --parcham baraye negahdarie feshorde shodane dokmeye
right, baraye inke dobare varede in block nashavad


if set_count>0 then --agar set_sount bozorgtar az 0 bud ejazeye
harkat bede
set_count :=set_count-1;
end if;
end if;
end if;
if (right='0' and right_flag='1') then--vaghti dokme azad shod flag
ra sefr mikonim

right_flag<='0';
end if;

c7:=c7+1;
if c7=250000 then  --taghsime ferekans baraye bedast avardane
ferekanse 4Hz az ferekanse 1MHz
c7:=0;
blink<=not blink; --blink ra dar har sanie 2 bar be halat haye tamam
sefr va tamam yek mibarad
if blink_flag = '1' then --agar shomaresh be sefr resid blink_flag ra
yek kon, va dar in ghesmat agar blink_flag yek shod ta 25 bar
cheshmak zadan beshmor va  bad az An blink_flag rasefr kon va buzzer
ra ham khamush kon
c8:= c8+1;
if c8 = 50  then
blink_flag <= '0';
buzzer<='0';
end if;              --dar har saniye 2 bar cheshmak bezan
end if;
end if;



c:=c+1; -- in kanter ta 400 mishemarad var dar har bazeye 50 taei
yeki az 7segment ha ra ba tavajjoh be hAlatash roshan kon, agar
set_count shomareye 7segmenti ra dashte bashad An 7segment cheshmak
mizanad, va agar blink_flag  yek bashad hameye 7segment ha cheshmak
mizanand.
if    (c>0 and c<=50)then
if set_count= 1 or blink_flag = '1' then
seg_en<="00000001" and  blink  ;  -- namayeshe adad roye sevensegment
ba cheshmak zadan
display<=seg_f1 ; --meghdare seg_f1 ra be port haye display befrest
else seg_en<="00000001";     --namayesh addi bar roye seven segment
display<=seg_f1;
end if;
elsif(c>50 and c<=100)then
if set_count= 2 or blink_flag = '1'then
```

```
seg_en<="00000010" and blink ;
display<=seg_f2 ;
else
seg_en<="00000010";
display<=seg_f2;
end if;
elsif(c>100 and c<=150)then
if set_count= 3 or blink_flag = '1'then
seg_en<="00000100" and blink ;
display<=seg_s1 ;
else seg_en<="00000100";
display<=seg_s1;
end if;
elsif(c>150 and c<=200)then
if set_count= 4 or blink_flag = '1'then
seg_en<="00001000" and blink ;
display<=seg_s2 ;
else seg_en<="00001000";
display<=seg_s2;
end if;
elsif(c>200 and c<=250)then
if set_count= 5 or blink_flag = '1'then
seg_en<="00010000" and blink  ;
display<=seg_m1 ;
else seg_en<="00010000";
display<=seg_m1;
end if;
elsif(c>250 and c<=300)then
if set_count= 6 or blink_flag = '1'then
seg_en<="00100000" and blink ;
display<=seg_m2 ;
else seg_en<="00100000";
display<=seg_m2;
end if;
elsif(c>300 and c<=350)then
if set_count= 7 or blink_flag = '1'then
seg_en<="01000000" and blink ;
display<=seg_h1 ;
else seg_en<="01000000";
display<=seg_h1;
end if;
elsif(c>350 and c<=400)then
if set_count= 8 or blink_flag = '1'then
seg_en<="10000000" and blink  ;
display<=seg_h2 ;
else seg_en<="10000000";
display<=seg_h2;
end if;
elsif(c>= 400 )then -- agar be 400 resid sefr kon ta shomaresh az
ebteda surat girad
c:=0;
end if;




c3:=c3+1;
if c3=100000 then -- taghsime ferekans baraye ferekanse 10 Hz
c3:=0;
```

```
if (up='1' and up_flag='0') then  -- dar sorati ke adade entekhabi
roye seven segment kochektaar az 9 ya 5 bod yeki ezafe kon, dahgAne
sAnie va daghighe ta 5 meghdar migirand!,   up_flag baraye in ast ke
dar hengame feshordan dokmeye up dobare varede in block nashavad
up_flag<='1';

case set_count is
when 1 =>   if f1<9 then
f1:=f1+1;
end if;
when 2 =>   if f2<9  then

f2:=f2+1;
end if;
when 3 =>   if s1<9 then
s1:=s1+1;
end if;
when 4 =>   if s2<5 then
s2:=s2+1;
end if;
when 5 =>   if m1<9 then
m1:=m1+1;
end if;
when 6 =>   if m2<5 then
m2:=m2+1;
end if;
when 7 =>   if h1<9 then
h1:=h1+1;
end if;
when 8 =>   if h2<9 then
h2:=h2+1;
end if;
when others=> null; -- dar halat haye digar hich kari anjam nadeh
end case;
end if;
end if;
if (up='0' and up_flag='1') then --agar dast ra az dokmeye  up
bardarim up flag ra sefr mikonad
up_flag<='0';
end if;


c4:=c4+1;
if c4=100000 then --taghsime ferekanse baraye ferekanse 10Hz
c4:=0;
if (down='1'and down_flag='0') then    --agar seven segment bozorgtar
ya mosaviye 1 bashad mitavan Anha ra kam kard
down_flag<='1';

case set_count is                   --ba taine meghdar case meghdar
seven segment  ra tain mikonad
when 1 =>   if f1>=1 then
f1:=f1-1;
end if;
when 2 =>   if f2>=1 then
f2:=f2-1;
end if;
when 3 =>   if s1>=1 then
s1:=s1-1;
end if;
when 4 =>   if s2>=1 then
```

```
s2:=s2-1;
end if;
when 5 =>   if m1>=1 then
m1:=m1-1;
end if;
when 6 =>   if m2>=1 then
m2:=m2-1;
end if;
when 7 =>   if h1>=1 then
h1:=h1-1;
end if;
when 8 =>   if h2>=1 then
h2:=h2-1;
end if;
when others=> null; -- dar halat haye digar hich kari anjam nadeh
end case;
end if;
end if;

if (down='0' and down_flag='1') then --agar dokmeye down ra
nafesharim, down_flag ra sefr mikonim
down_flag<='0';
end if;




--in ghesmat maghadire 7segment hara tabdil be namyeshe LED haye
7segment mikonad

case f1 is                              --meghdar sadome saniye baraye
seven segment
when 0      => seg_f1<="1000000"; --1 yani khamush va sefr yani
roshan
when 1      => seg_f1<="1111001";
when 2      => seg_f1<="0100100";
when 3      => seg_f1<="0110000";
when 4      => seg_f1<="0011001";
when 5      => seg_f1<="0010010";
when 6      => seg_f1<="0000010";
when 7      => seg_f1<="1111000";
when 8      => seg_f1<="0000000";
when 9      => seg_f1<="0010000";
when others=> null;
end case;
case f2 is                              --meghdar sadome saniye baraye
seven segment
when 0      => seg_f2<="1000000";
when 1      => seg_f2<="1111001";
when 2      => seg_f2<="0100100";
when 3      => seg_f2<="0110000";
when 4      => seg_f2<="0011001";
when 5      => seg_f2<="0010010";
when 6      => seg_f2<="0000010";
when 7      => seg_f2<="1111000";
when 8      => seg_f2<="0000000";
when 9      => seg_f2<="0010000";
when others=> null;
end case;
```

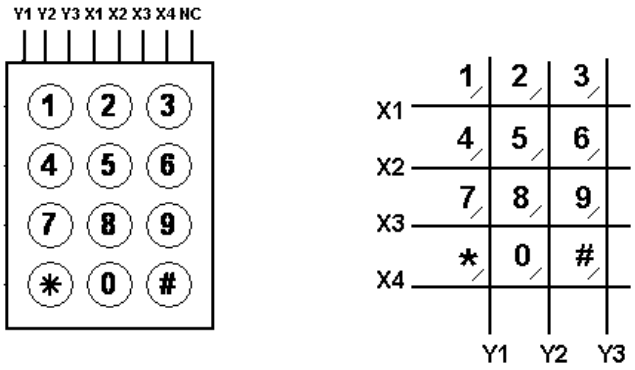
```
case s1 is                          --meghdar  saniye baraye seven
segment
when 0      => seg_s1<="1000000";
when 1      => seg_s1<="1111001";
when 2      => seg_s1<="0100100";
when 3      => seg_s1<="0110000";
when 4      => seg_s1<="0011001";
when 5      => seg_s1<="0010010";
when 6      => seg_s1<="0000010";
when 7      => seg_s1<="1111000";
when 8      => seg_s1<="0000000";
when 9      => seg_s1<="0010000";
when others=> null;
end case;
case s2 is                          --meghdar  saniye baraye seven
segment
when 0      => seg_s2<="1000000";
when 1      => seg_s2<="1111001";
when 2      => seg_s2<="0100100";
when 3      => seg_s2<="0110000";
when 4      => seg_s2<="0011001";
when 5      => seg_s2<="0010010";
when others=> null;
end case;
case m1 is                          --meghdar  daghighe baraye seven
segment
when 0      => seg_m1<="1000000";
when 1      => seg_m1<="1111001";
when 2      => seg_m1<="0100100";
when 3      => seg_m1<="0110000";
when 4      => seg_m1<="0011001";
when 5      => seg_m1<="0010010";
when 6      => seg_m1<="0000010";
when 7      => seg_m1<="1111000";
when 8      => seg_m1<="0000000";
when 9      => seg_m1<="0010000";
when others=> null;
end case;
case m2 is                          --meghdar  daghighe baraye seven
segment
when 0      => seg_m2<="1000000";
when 1      => seg_m2<="1111001";
when 2      => seg_m2<="0100100";
when 3      => seg_m2<="0110000";
when 4      => seg_m2<="0011001";
when 5      => seg_m2<="0010010";
when others=> null;
end case;
case h1 is                          --meghdar  saat baraye seven
segment
when 0      => seg_h1<="1000000";
when 1      => seg_h1<="1111001";
when 2      => seg_h1<="0100100";
when 3      => seg_h1<="0110000";
when 4      => seg_h1<="0011001";
when 5      => seg_h1<="0010010";
when 6      => seg_h1<="0000010";
when 7      => seg_h1<="1111000";
when 8      => seg_h1<="0000000";
when 9      => seg_h1<="0010000";
when others=> null;
```

```
end case;
case h2 is                          --meghdar  saat baraye seven
segment
when 0      => seg_h2<="1000000";
when 1      => seg_h2<="1111001";
when 2      => seg_h2<="0100100";
when 3      => seg_h2<="0110000";
when 4      => seg_h2<="0011001";
when 5      => seg_h2<="0010010";
when 6      => seg_h2<="0000010";
when 7      => seg_h2<="1111000";
when 8      => seg_h2<="0000000";
when 9      => seg_h2<="0010000";
when others=> null;
end case;

c5:=c5+1;
if c5=100000 then -- taghsime ferekans be 10 Hz
c5:=0;
if (start='1' and start_flag='0') then      --dokmeye shoro va
ist,agar en_down yek shaved shomaresh anjam migardad va agar sefr
shaved maks ijad migardad
start_flag<='1';
en_down <=not en_down;
end if;
end if;
if (start='0' and start_flag='1') then --agar start digar feshorde
nashavad  star_flag ra sefr kon
start_flag<='0';
end if;



if    en_down='1' then  --agar en_down yek shavad ejezeye shomaresh
midahad

c1:=c1+1;
if c1=10000 then         -- baraye be dast amadane sadome saniye,
taghsime ferekans 1MHz be 100Hz
c1:=0;
if (f1>0) then   --dar in ghesmat vaghti sadom saniye be 0 resid mire
saniye ra kam mikonad ta be 0 beresad in kar ta sa'at edame darad ta
hame sefr va buzzer be seda dar ayad
f1:=f1-1;

elsif (f1=0 and f2>0) then  --agar sadome sanie sef shaved va dahome
sanie bozorgtar az sefr bud , f2 ra yeki kam kon, f1 ra 9 bede
f2:=f2-1;
f1:=9;

elsif (f1=0 and s1>0) then --agar f1 sefr bud va f2 niz sefr bud va
s1 bozorgtar az sefr bud   s1 ra yeki kam kon, f2 ra 9 bede va f1 ra
niz 9 bede, in amaliat ta enteha edame darad
s1:=s1-1;
f2:=9;
f1:=9;

elsif (f1=0 and s2>0) then
s2:=s2-1;
s1:=9;
f2:=9;
```

```
f1:=9;

elsif (f1=0 and m1>0) then
m1:=m1-1;
s2:=5; --chon dahome sanie ast
s1:=9;
f2:=9;
f1:=9;

elsif (f1=0 and m2>0) then
m2:=m2-1;
m1:=9;
s2:=5;
s1:=9;
f2:=9;
f1:=9;

elsif (f1=0 and h1>0) then
h1:=h1-1;
m2:=5;
m1:=9;
s2:=5;
s1:=9;
f2:=9;
f1:=9;

elsif (f1=0 and h2>0) then
h2:=h2-1;
h1:=9;
m2:=5;
m1:=9;
s2:=5;
s1:=9;
f2:=9;
f1:=9;

else --agar hame sefr shodand buzzer roshan shavad en_down sefr
shaved blink_flag yek shaved va set_count niz sefr gardad
buzzer<='1';
en_down<='0';
blink_flag<='1';
set_count := 0;
end if;
end if;
end if;
end if;

end if;




end process;

end Behavioral;
```

۷- اتصال صفحه کلید ماتریسی به FPGA

صفحه کلیدی که در اینجا استفاده می‌کنیم، از نوع صفحه کلیدهای ماتریسی است. ساختمان داخلی این نوع صفحه کلیدها به صورتی که در شکل ۱ نشان داده شده است، می‌باشد. این شکل، یک صفحه کلید ۱۲ کلیدی را نشان می‌دهد. نمونه آن را در شماره گیرهای تلفن هم دیده‌اید. تنوع صفحه کلیدهای ماتریسی زیاد، ولی اصول کار اکثر آن‌ها یکسان است. در بعضی صفحه کلیدهای ماتریسی، ممکن است به جای X ها، به حرف R و به جای Y ها، به حرف C بر بخورید.



شکل ۱ صفحه کلید ماتریسی (KeyPad)

وقتی هیچ کلیدی فشار داده نشده باشد در خطوط Y1، Y2، Y3 و Y4 مقدار '1' منطقی می‌بینیم چون Y1 تا Y4 با مقاومت Pull-up به Vcc متصل شده‌اند و اگر تمام سطرها (X1 تا X3) را به GND وصل کنیم با فشرده شدن یک کلید در یک ستون، خط Y متناظر آن، '0' منطقی می‌شود.

اصول کار این صفحه کلیدها به این صورت است که هر بار یکی از سطرهای آن را '0' و بقیه را '1' می‌کنیم. برای نمونه در ابتدای کار به X1، '0' منطقی و به X2، X3 و X4، '1' منطقی می‌دهیم.

حال اگر کلیدی در سطر X1 (یکی از کلیدهای 0، 1، 2 و 3) فشار داده شده باشد، ستون متناظر آن '0' خواهد شد و دیگر ستون‌ها '1' می‌ماند.

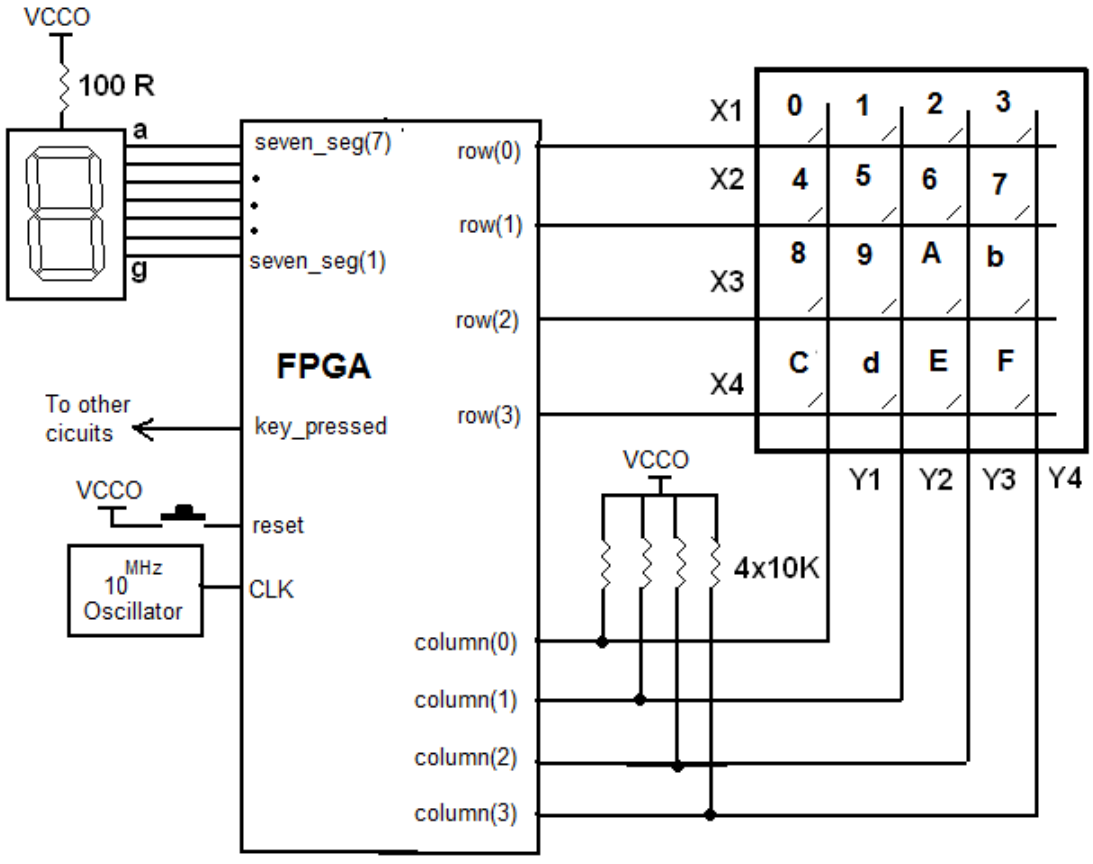
پس کافی است پس از '0' کردن سطر X1، ستون‌های Y1 تا Y4 را خوانده هرکدام '0' شد، کلید متناظر آن را به عنوان کلید فشرده شده در نظر بگیریم.

اگر هر چهار ستون '1' ماندند، در این سطر کلیدی فشرده نشده و سراغ سطر بعدی می‌رویم و همین کار را برای آن سطر انجام می‌دهیم. منتها برای این سطر، ستون‌های Y1، Y2، Y3 و Y4 متناظر کلیدهای 4، 5، 6 و 7 است.

این کار را برای تمام سطرها انجام می‌دهیم و دو باره به سطر اول بر می‌گردیم و این کار را به طور متناوب تکرار می‌کنیم.

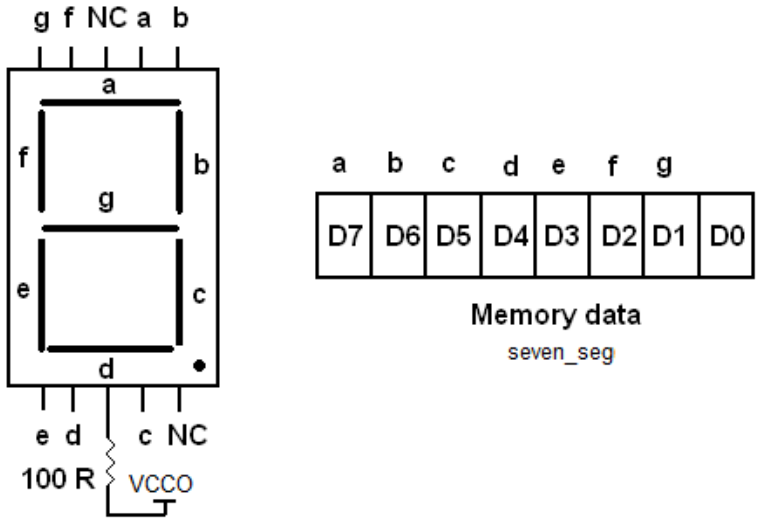
در ضمن اگر تشخیص داده شد که ستونی '0' است باید یک مدت زمانی صبر کرد تا این که پارازیت‌های احتمالی به عنوان فشار کلید تعبیر نشوند. اگر پس از این مدت زمانی (چیزی حدود $250^{\mu s}$) باز هم ستون مربوط '0' بود، می‌فهمیم که این سیگنال پارازیت نیست.

مطابق شکل ۲-۲ صفحه کلید را به FPGA متصل کنید. توجه داشته باشید که X1، X2، X3 و X4 به عنوان خروجی به پایه‌های row(0) تا row(3) وصل می‌شوند و پین‌های Y1 تا Y4 به عنوان ورودی به column(0) تا column(3) وصل می‌شوند.



شکل ۲ مدار اتصال KeyPad به FPGA

شکل ۲-۳ هم پایه‌های 7-Segment آند مشترک و تناظر سگمنت‌ها با بیت‌های کلمات حافظه را نشان می‌دهد.



شکل ۳ پایه‌های 7-segment آند مشترک و سگمنتها در کلمات حافظه

در ادامه برنامه مربوط به صفحه کلید آمده است.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

 ----Uncomment the following library declaration if instantiating
 ----any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity keypad is
   Port ( row : out std_logic_vector(3 downto 0);
          column : in std_logic_vector(3 downto 0);
          led : out std_logic_vector(3 downto 0);
          seven_seg : out std_logic_vector(7 downto 0);
          clk : in std_logic;
          reset : in std_logic;
          key_pressed : out std_logic);
end keypad;

architecture Behavioral of keypad is
   signal debounce_flag : std_logic;
   constant debounce_time : integer := 200000;
   signal row_i, column_encoded, row_encoded ,
   column_before, digit: std_logic_vector (3 downto 0);
   type state is (start, waiting, prob_pressed, delay);
   signal current : state;

begin
--------------
```

```
      process (clk, reset)
        variable count, count1: integer;
      begin
      if reset = '1' then
            current <= start;
         key_pressed <= '0';
      elsif clk = '1' and clk'event then
            case current is
            when start  =>
                  row_i <= "0000";
                  key_pressed <= '0';
                  count := 0;
                  if column = "1111" then
                        current <= waiting;
                        row_i <= "1110";
               end if;
            when waiting =>
                  count := count + 1;
                  if count = 1000 then
                        row_i <= row_i(2 downto 0) & row_i(3);
                        count := 0;
                  end if;
                  if column /= "1111" then
                        current <= prob_pressed;
                        count1 := debounce_time;
                        column_before <= column;
               end if;
            when prob_pressed  =>
                  count1 := count1 - 1;
                  if count1 = 0 then
                        if column_before = column then
                              digit <= row_encoded + column_encoded;
                              key_pressed <= '1';
                              current <= delay;
                              count1 := 10000;
                        else
                        current <= start;
                        end if;
                  end if;
            when delay  =>
                  count1 := count1 -1;
                  if count1 = 0 then
                        current <= start;
                  end if;
            when others =>
                  current <= start;
        end case;

      end if;
end process;
   column_encoded <= "0000" when column = "1110" else
                     "0100" when column = "1101" else
                     "1000" when column = "1011" else
                     "1100" when column = "0111" else
                     "0000";

   row_encoded <= "0000" when row_i = "1110" else
                        "0001" when row_i = "1101" else
                  "0010" when row_i = "1011" else
                  "0011" when row_i = "0111" else
                  "0000";
```

```
   row <= row_i;
----------------------
   led <= digit;
   seven_seg <= "00000010" when digit = "0000" else
            "10011110"    when digit = "0001" else
            "00100100"    when digit = "0010" else
            "00001100"    when digit = "0011" else
            "10011000"    when digit = "0100" else
            "01001000"    when digit = "0101" else
            "01000000"    when digit = "0110" else
            "00011110"    when digit = "0111" else
            "00000000"    when digit = "1000" else
            "00001000"    when digit = "1001" else
            "00010000"    when digit = "1010" else
            "11000000"    when digit = "1011" else
            "01100010"    when digit = "1100" else
            "10000100"    when digit = "1101" else
            "01100000"    when digit = "1110" else
            "01110000"    when digit = "1111";
end Behavioral;
```

۸- تولید موسیقی

هدف تولید نت های موسیقی به وسیله ی FPGA می باشد. در مدار شکل زیر تنها به یک عدد بلندگوی کوچک احتیاج است و می خواهیم با فشردن هر کدام از کلید های تعبیه شده روی برد FPGA نت های هفتگانه موسیقی را تولید کنیم. همچنین یک button برای reset از buttonهای موجود بر روی برد fpga استفاده می کنیم.

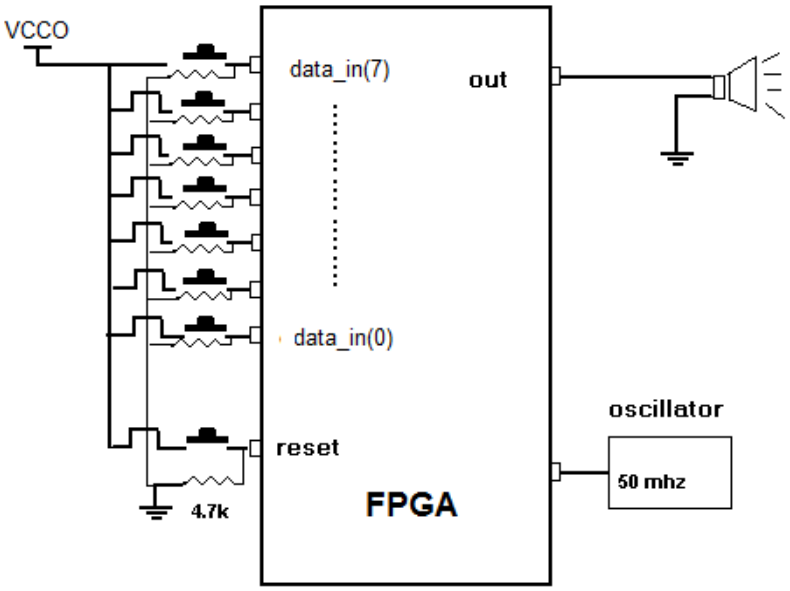
همچنین از اسیلاتور موجود بر روی برد fpga استفاده شده است.

فرکانس مرجع 50mhz می باشد برای تولید فرکانس نت های موسیقی می بایست فرکانس مرجع را با تقیسم بر عددی به فرکانس مربوطه تبدیل کرد و آن فرکانس را به خروجی هدایت نمود.

جهت تولید فرکانس می بایست ۵۰۰۰۰۰۰۰ را بر فرکانس استاندار هر نت تقسیم نموده و مقدار بدست آمده را به عنوان delay جهت شمارش counter تا آن مقدار استفاده نمود.

به این ترتیب با ۱ شدن پرچم ،شمارنده تا مقدار delay به شمارش ادامه خواهد داد و با توجه به if_else مقدار Delay مربوط به کلید فشرده شده جهت شمارش با counter مقایسه خواهد شد.

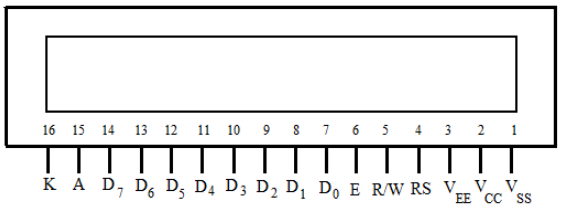
شکل کلی مدار :



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity music is
      port(clk, reset : in STD_LOGIC;
           speaker : buffer STD_LOGIC;
           data : in STD_LOGIC_VECTOR(6 downto 0));
end music;

architecture Behavioral of music is
signal delay : integer:=0;
signal count,flag:integer:=0;
begin

      process (clk,reset)
      begin

            if(reset = '1') then
            speaker<='0';
                  count<=0;
                  flag<=0;
            elsif(clk'event and clk='1') then
                  if (flag=1) then
                        count<=count+1;
                        if (count=delay) then
                              speaker<=not(speaker);
                              count<=0;
                              flag<=0;
                        end if;
                  else
                        if(data(0)='1')then
                              delay<=64102;
                              flag<=1;
                        elsif(data(1)='1')then
                              delay<=57077;
                              flag<=1;
                        elsif(data(2)='1')then
                              delay<=50864;
                              flag<=1;
                        elsif(data(3)='1')then
                              delay<=47984;
                              flag<=1;
                        elsif(data(4)='1')then
                              delay<=42783;
                              flag<=1;
                        elsif(data(5)='1')then
                              delay<=38080;
                              flag<=1;
                        elsif(data(6)='1')then
                              delay<=33944;
                              flag<=1;
                        end if;
                  end if;
            end if;
      end process;

end Behavioral;
```

۹- اتصال LCD به FPGA

یک نمونه از صفحه نمایش‌های LCD ساده را در شکل ۱ می‌بینید.



شکل ۱ پایه‌های LCD (دید از روبرو)

دو پایه ۱۵ و ۱۶ مربوط به نور Backlight می‌باشند که در صورت عدم نیاز می‌توان آنها را به جایی وصل نکرد.

پایه‌های V_{SS} و V_{CC} را باید به ترتیب به GND و V_{CC} وصل نمود. پایه V_{EE} برای کنترل شدت روشنایی LCD به کار می‌رود. می‌توان با اتصال یک پتانسیومتر (حدود 10^K) به این پایه شدت روشنایی آن را کنترل نمود. پایه‌های D0 تا D7 برای ورود اطلاعات می‌باشند که می‌تواند داده (کد اسکی کاراکتری که می‌خواهیم نمایش داده شود) یا دستور (عملی که باید انجام شود) باشد.

برای این که به LCD بفهمانیم که می‌خواهیم داده بفرستیم خط RS را برابر '1' و برای فرستادن دستور این خط را برابر '0' قرار می‌دهیم.

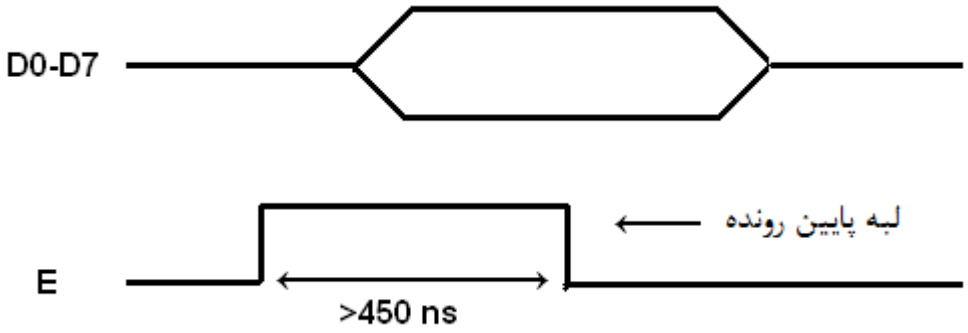
از پایه D7 می‌توان برای خواندن وضعیت کنونی LCD، که این که آیا مشغول انجام دادن کاری می‌باشد یا خیر استفاده نمود. برای این کار باید خط R/W که در حالت نوشتن (در حالت فرستادن داده ودستور) باید آن را برابر '0' قرار داد، در این حالت (حالت خواندن) آن را برابر '1' قرار دهیم. پس از آن باید D7 را بخوانیم. اگر D7 برابر '1' بود، LCD مشغول انجام کار قبلی می‌باشد (Busy می‌باشد) و اطلاعات (دستور یا داده) تازه را نمی‌پذیرد. ولی اگر '0' =D7 بود می‌توان اطلاعات جدید را فرستاد.

در هر حالت برای فرستادن داده یا دستور پس از اعمال اطلاعات به D0 تـا D7 بایـد مطـابق شـکل ۲ یک لبه پایین رونده به پایه E اعمال کرد تا اطلاعات ترتیب اثر داده شوند. همچنین برای خواندن وضعیت نیز باید یک لبه پایین رونده به E اعمال نمود. حداقل مدت زمان '1' بودن پالس اعمال شده به پایه E باید 450^{ns} باشد.

دستورات اصلی که اول کار برای آماده‌سازی LCD باید فرستاد به شرح زیر اند:

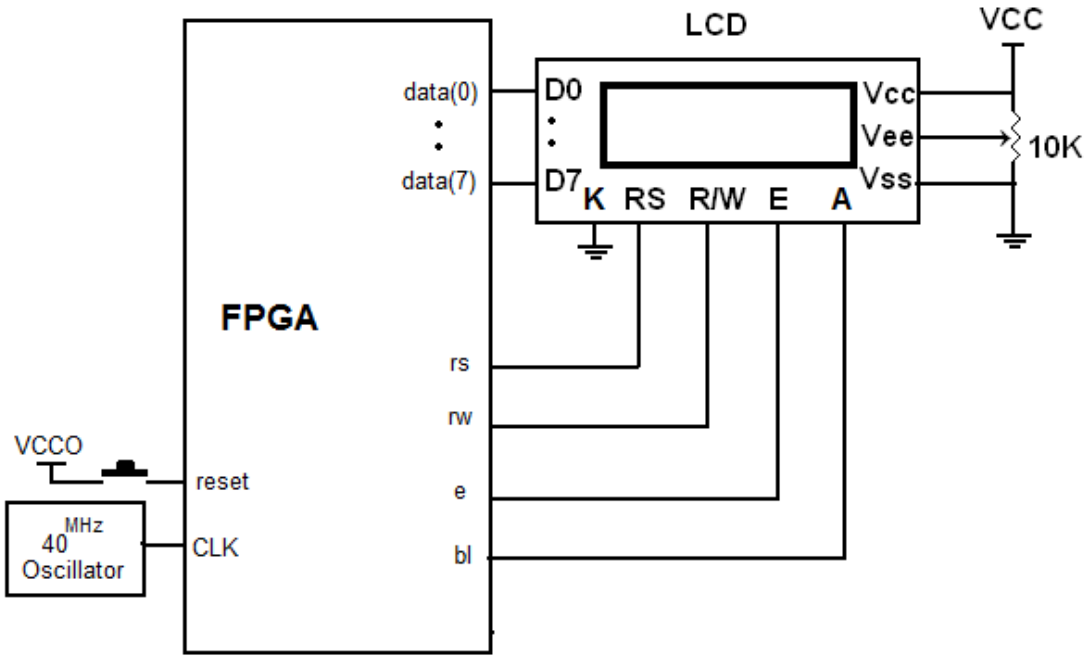
برای آماده‌سازی و تعریف نوع LCD باید دستور 30H (LCD هـای یـک سـطری) یـا دسـتور 38H (LCDهای دو سطری) را فرستاد. پس از آن دستور 0EH که LCD و مکان‌نما را روشن می‌کند باید فرستاده شود و همچنین باید با فرستادن دستور 01H صفحه LCD را پاک کرد.

اگر خواستیم مکان‌نما را به سمت راست شیفت دهیم از دستور 06H استفاده می‌کنیم و برای شیفت به چپ از دستور 04H بهره می‌گیریم.



شکل ۲ نحوه اعمال اطلاعات به LCD

مدار شکل ۳ را ببندید و نتیجه را مشاهده کنید.



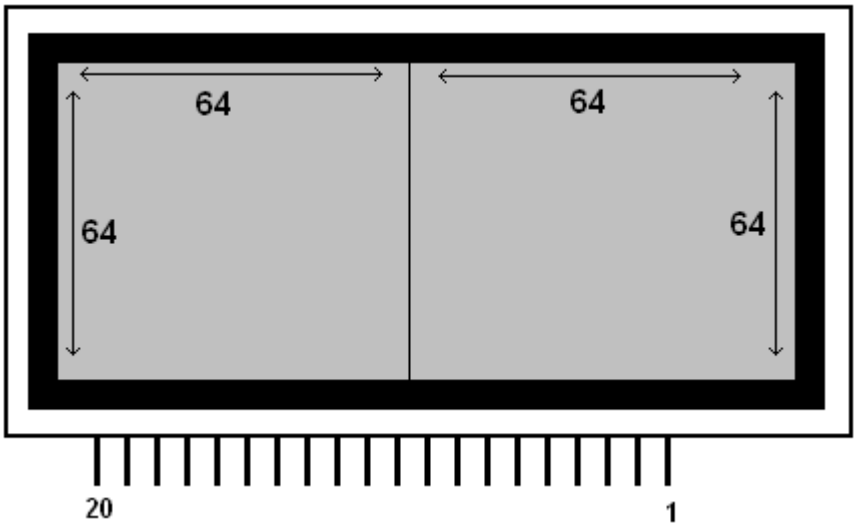
شکل ۳ مدار اتصال LCD به FPGA

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity lcd is
   Port ( data : out std_logic_vector(7 downto 0);
          rs : out std_logic;
          rw : out std_logic;
          e : out std_logic;
          bl : out std_logic;
          clk : in std_logic;
          reset : in std_logic);
end lcd;

architecture Behavioral of lcd is
   constant number_of_data : integer := 14;
   constant delay_for_e : integer := 18;  --18 * 25 NS = 450 NS
   constant delay_between_data : integer := 100000;
   type two_dimen_array is array (0 to number_of_data - 1)
                           of std_logic_vector (8 downto 0);
   constant table : two_dimen_array :=
                       ("000111000", "000001110", "000000001",
                        "101001000", "101100101", "101101100",
                        "101101100", "101101111", "110100000",
                        "101110111", "101101111", "101110010",
```

```
                        "101101100", "101100100");    -- Hello world
   type state is (start, write, delay, stop);
   signal current : state;
   signal part : integer;
begin

   process (clk, reset)
      variable i, count : integer;
   begin
      if reset = '1' then
         current <= start;
            rs <= '0';
            rw <= '0';
            e <= '0';
            bl <= '1';
            count := 0;
            i := 0;
            part <= 0;
      elsif clk = '1' and clk'event then
         case current is
         when start =>
            data <= table(i)(7 downto 0);
            rs <= table (i)(8);
            i := i + 1;
            if i < (number_of_data + 1) then
               current <= write;
            else
               current <= stop;
            end if;
         when write =>
            if part = 0 then
               e <= '1';
               count := delay_for_e;
               part <= part + 1;
               current <= delay;
            elsif part = 1 then
               e <= '0';
               count := delay_between_data;
               current <= delay;
               part <= part + 1;
            else
               current <= start;
               part <= 0;
            end if;
         when delay =>
            count := count - 1;
            if count = 0 then
               current <= write;
            end if;
         when stop =>
         end case;
      end if;
   end process;

end Behavioral;
```

۱۰- اتصال LCD گرافیکی به FPGA

شکل ۱ پایه‌های GLCD (Graphic LCD) و توضیحات آنها را نشان می‌دهد.



Pin #	Symbol	Function	Pin #	Symbol	Function
1	VSS	GND	11	DB4	Data Bus Line 4
2	VDD	Power Supply (5V)	12	DB5	Data Bus Line 5
3	Vo	Contrast Adjustment	13	DB6	Data Bus Line 6
4	D/I	Data/Instruction	14	DB7	Data Bus Line 7
5	R/W	Data Read/Write	15	CS1	Chip Select for IC1
6	E	H → L Enable Signal	16	CS2	Chip Select for IC2
7	DB0	Data Bus Line 0	17	RST	Reset
8	DB1	Data Bus Line 1	18	Vee	Negative Voltage Output
9	DB2	Data Bus Line 2	19	A	Power Supply for LED (4.2V)
10	DB3	Data Bus Line 3	20	K	Power Supply for LED (0V)

شکل ۱ پایه‌های LCD گرافیکی و توضیحات آن

GLCD استفاده شده در این مدار دارای ۱۲۸×۶۴ پیکسل می‌باشد، صفحه GLCD دارای دو قسمت ۶۴×۶۴ پیکسل است. هر قسمت GLCD با پایه‌های CS1 و CS2 کنترل می‌شود، اگر این دوپایه همزمان فعال باشند، نمایش در هر دو صفحه به صورت یکسان انجام می‌گیرد. برای استفاده از تمام پیکسل ها، باید در هر لحظه فقط یکی از این دو پایه فعال باشند.

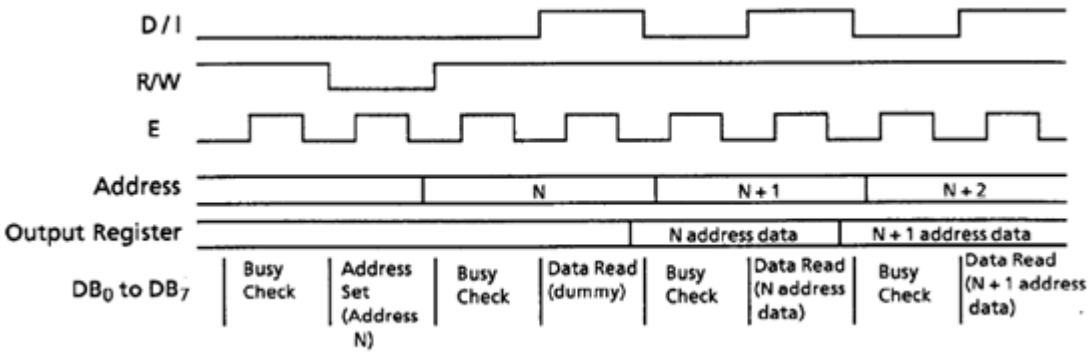
خطوط DATA (DB0-DB7) برای فرستادن دستورات و یا تعیین وضعیت ۸ پیکسل جاری استفاده می‌شوند. خط D/I این تفاوت را مشخص می‌کند هنگامی که این خط '0' باشد، GLCD آماده پذیرش دستورات است و با '1' کردن این خط GLCD پیکسل های مورد نظر را خاموش و یا روشن می‌کند. خط R/W برای خواندن رجیستر وضعیت و یا محتویات RAM داخلی GLCD باید '1' باشد و در بقیه حالتها '0' است.

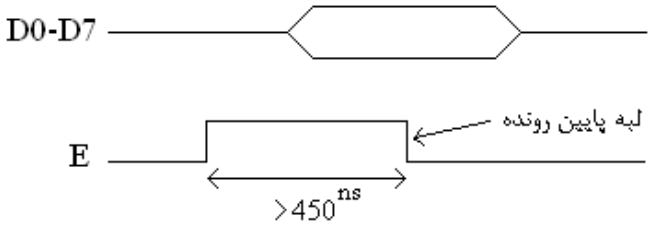
رجیستر وضعیت اطلاعاتی از قبیل مشغول بودن GLCD، روشن و یا خاموش بودن صفحه و همچنین وضعیت RESET را نشان می‌دهد. به عنوان مثال اگر در وضعیت خواندن (R/W = '1') خط D/I هم '0' باشد، DB7 وضعیت مشغول (Busy) بودن GLCD را نشان می‌دهد (در صورت مشغول بودن، DB7 = '1' می‌شود) بنابراین تا '0' شدن این پایه باید صبر کرد.

با اعمال یک لبه پایین رونده به خط E داده‌ها به GLCD منتقل می‌شوند و یا از GLCD خوانده می‌شوند. حداقل مدت زمان '1' بودن پالس اعمال شده به پایه E باید 450ns باشد.

پایه‌های A و K (۱۹و ۲۰) نور پس زمینه را فراهم می‌کنند با قرار دادن یک مقاومت 100 اهم بین پایه A و Vcc و متصل نمودن پایه K به زمین نور پس زمینه فعال می‌شود.

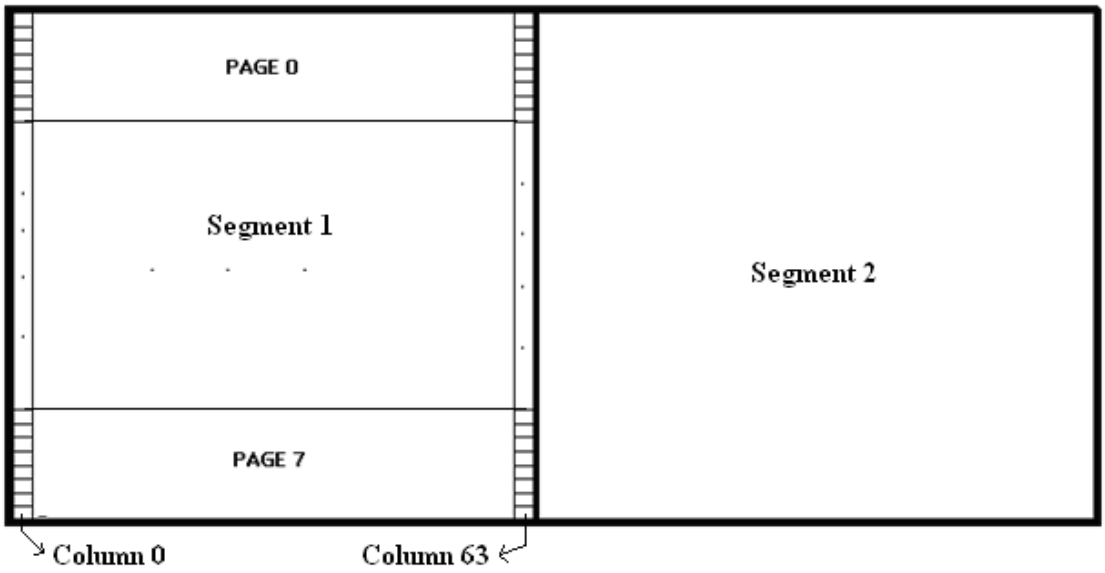
شکل ۲، شکل موجهای مربوط به پایه‌های کنترلی و داده GLCD را نشان می‌دهد.



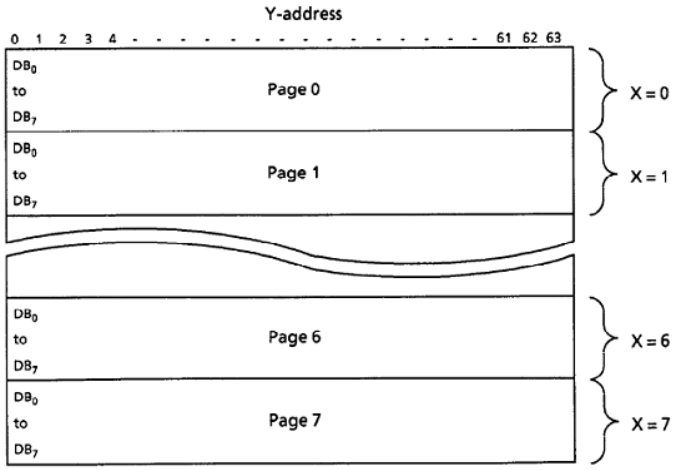


شکل ۲ شکل موجهای پایه‌های LCD گرافیکی

همانگونه که در شکل ۳ مشاهده **می**کنید، صفحه نمایش GLCD به دو قسمت تقسیم شده و هر قسمت نیز دارای ۸ صفحه (PAGE) و ۶۴ ستون می‌باشد، هر ستون در صفحه دارای ۸ پیکسل است. شکل ۴ هم یک Segment را به همراه Page هایش نشان می‌دهد.



شکل ۳ قسمتهای یک LCD گرافیکی



شکل ۴ یک Segment از LCD و Page هایش

دستور 3FH صفحه نمایش را روشن می‌کند.

دستور شماره C0H+Shift مقدار شیفت کل صفحه GLCD به بالا را مشخص می‌کند (حاصل جمع عدد C0H با عدد Shift، عددی است که شماره دستور را مشخص می‌کند.)

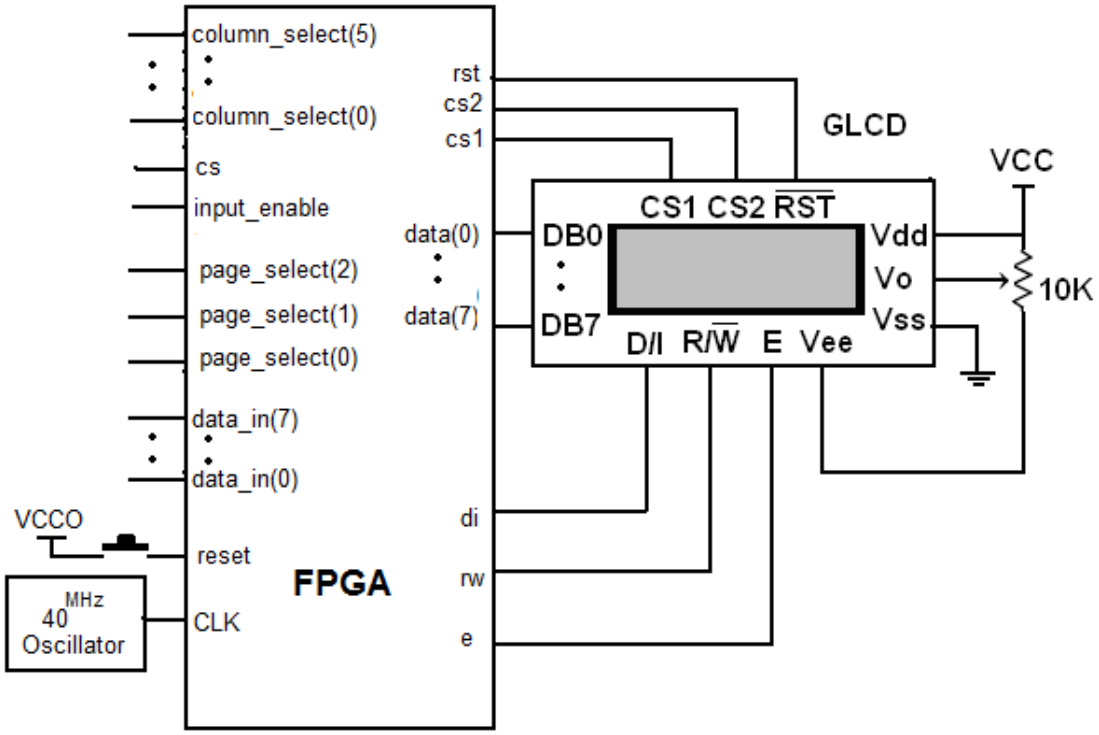
دستور B8H+PAGE Number انتخاب صفحه مورد نظر (PAGE Number = 0-7).

دستور 40H+Column Number انتخاب ستون مورد نظر (Column Number = 0-63).

حال در این صفحه و ستون ۸ پیکسل در اختیار داریم که وضعیت روشن و یا خاموش بودن آنها در حالت '1' = D/I انجام می‌گیرد. هنگامی‌که این خط '1' است ۸ بیت داده ای که روی خطوط DATA قرار می‌دهیم وضعیت پیکسل‌ها را مشخص می‌کند.

برای روشن شدن پیکسل، بیت موردنظر را '1' و برای خاموش شدن '0' می‌کنیم. به عنوان مثال مقدار 01010101 هشت پیکسل را یکی در میان روشن می‌کند.

مدار شکل ۵ را ببندید و نتیجه را مشاهده کنید. در ادامه برنامه اتصال GLCD به FPGA آمده است.



شکل ۵ مدار اتصال GLCD به FPGA

در برنامه زیر با استفاده از سه خط page_select(0)-page_select(2) صفحه مورد نظر انتخاب می‌شود، ستونی که در آن پیکسل‌ها روشن می‌شوند، با پنج خط column_select تعیین می‌شود. با استفاده از data_in می‌توانید پیکسل های مورد نظر در صفحه و ستون انتخاب شده را به دلخواه روشن و یا خاموش کنید. با ورودی cs هم می‌توانید segment مربوطه را تعیین کنید. همه این ورودی‌ها در صورتی تاثیر می‌گذارند که ورودی input_enable فعال باشد.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity glcd is
   Port ( data : out std_logic_vector(7 downto 0);
          di : out std_logic;
          rw : out std_logic;
          e : out std_logic;
          cs1, cs2, rst : out std_logic;
```

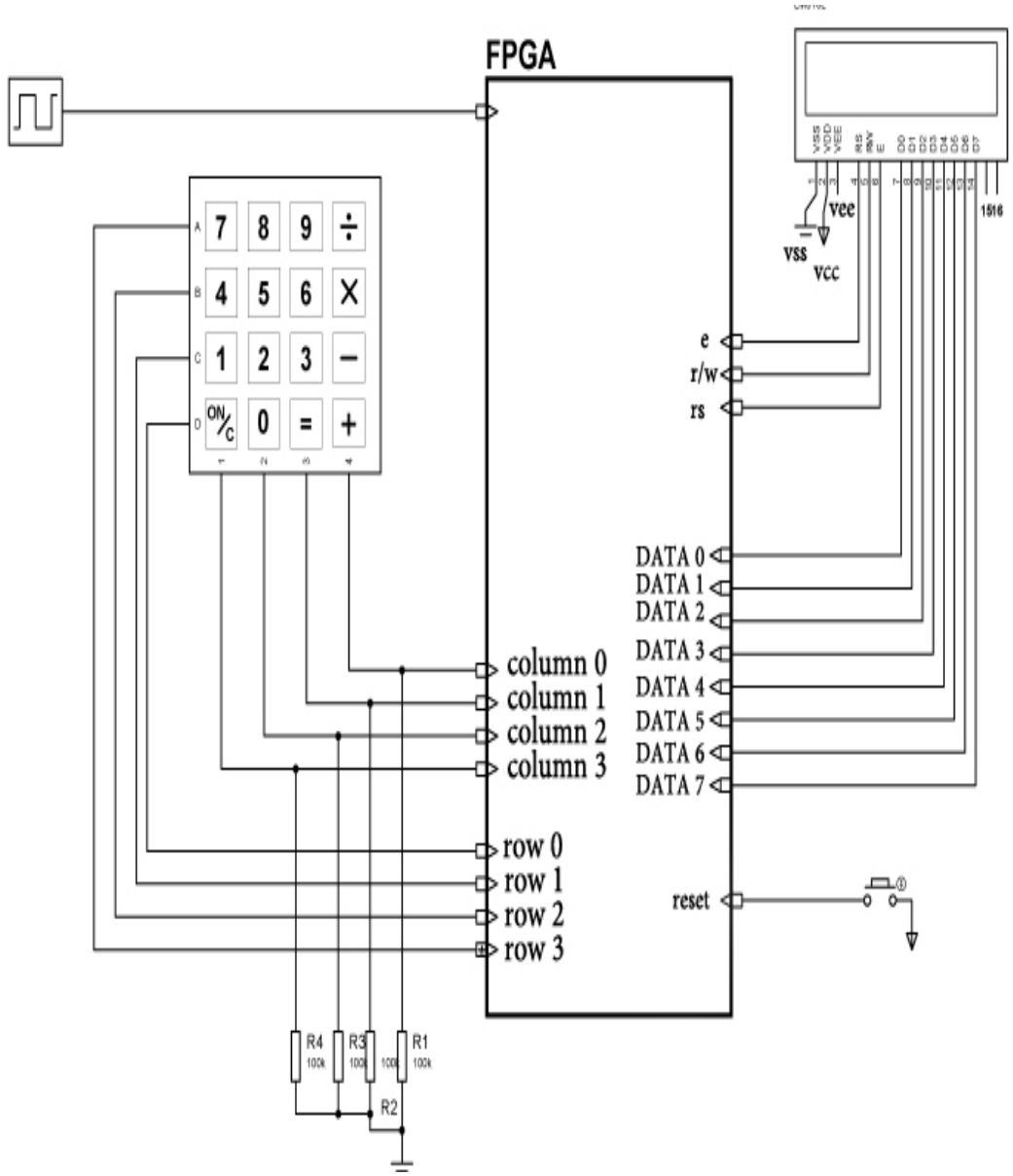
```
          cs, input_enable : in std_logic;
          page_select : in std_logic_vector(2 downto 0);
          column_select : in std_logic_vector(5 downto 0);
          data_in : in std_logic_vector(7 downto 0);
          clk : in std_logic;
          reset : in std_logic );
end glcd;

architecture Behavioral of glcd is
   signal page : std_logic_vector (7 downto 0);
   constant number_of_data : integer := 15;
   constant delay_for_e : integer := 18;  --18 * 25 NS = 450 NS
   constant delay_between_data : integer := 100000;
   type state is (start, write, delay, clear, normal);
   signal current, caller : state;
   signal part, part1 : integer;

begin

   process (clk, reset)
      variable i, count : integer;
   begin
      if reset = '1' then
         current <= start;
         di <= '0';
         rw <= '0';
         e <= '0';
         cs1 <= '1';
         cs2 <= '1';
         rst <= '1';
         count := 0;
         i := 0;
         part <= 0;
         part1 <= 0;
         page <= "00000000";
      elsif clk = '1' and clk'event then
         case current is
         when start =>
            data <= "00111111";
            di <= '0';
            caller <= clear;
            current <= write;
         when write =>
            if part = 0 then
               e <= '1';
               count := delay_for_e;
               part <= part + 1;
               current <= delay;
            elsif part = 1 then
               e <= '0';
               count := delay_between_data;
               current <= delay;
               part <= part + 1;
            else
               current <= caller;
               part <= 0;
            end if;
         when delay =>
            count := count - 1;
            if count = 0 then
```

```
            current <= write;
         end if;
      when clear =>
         if i = 0 then
            data <= "10111000" + page;
            di <= '0';
            caller <= clear;
            current <= write;
            i := i + 1;
         elsif i = 1 then
            data <= "01000000";
            di <= '0';
            caller <= clear;
            current <= write;
            i := i + 1;
         elsif i > 1 and i < 66 then
            data <= "00000000";
            di <= '1';
            caller <= clear;
            current <= write;
            i := i + 1;
         elsif i = 66 then
            page <= page + 1;
            i := 0;
            if page = 7 then
               current <= normal;
            else
               caller <= clear;
            end if;
         end if;
      when normal =>
         if input_enable = '1' then
            cs1  <= cs;
            cs2 <= not cs;
            if part1 = 0 then
               data <= "10111000" + ("00000" & page_select);
               di <= '0';
               caller <= normal;
               current <= write;
               part1 <= part1 + 1;
            elsif part1 = 1 then
               data <= "01000000" + ("00" & column_select);
               di <= '0';
               caller <= normal;
               current <= write;
               part1 <= part1 + 1;
            elsif part1 = 2 then
               data <= data_in;
               di <= '1';
               caller <= normal;
               current <= write;
               part1 <= 0;
            end if;
         end if;
      end case;
   end if;
end process;
end Behavioral;
```

۱۱- قفل رمزی

برای انجام این پروژه در ابتدای کار باید بورد LCD را با استفاده از نقشه زیر طراحی کرده سپس این بورد را توسط یک کابل ارتباطی(جی تگ)به بورد FPGA وصل کنیم تا توسط این LCD بتوانیم ۴ عددی را که توسط برنامه حدس زده شده است ببینیم.نحوه کار به این گونه می باشد که بعد از پروگرام کردن این برنامه روی FPGA، ۴ عدد به صورت random در LCD نشان داده می شود سپس خود کاربر این عدد تولید شده را به صورت ذهنی یک به یک و جداگانه از رمز در آورده و با دکمه های key pad وارد FPGA می کنیم و بعد از وارد کردن هر ۴ عدد یک دکمه دیگر فرضا menu را فشار می دهیم ،در پایان کار اگر این ۴ عدد تولید شده را درست و صحیح از رمز در آوردیم و وارد کردیم باید LED که مربوط به این کار تعریف شده است روشن شود در غیر این صورت اگر به اشتباه اعداد را وارد کنیم LED دیگری که مربوط به غیر صحیح بودن است روشن می شود.

مدار شکل زیر را ببندید و نتیجه را مشاهده کنید.



کد برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity lcd is
   Port (LedOk,LedNotOk: out std_logic;
                 -- LCD Data
```

```
                  data : out std_logic_vector(7 downto 0);
                  -- satr key pad
                row : out std_logic_vector(3 downto 0);
                   -- soton key pad
                column : in std_logic_vector(3 downto 0);
                   led:buffer std_logic_vector(3 downto 0);
                   -- rs LCD
                   rs : out std_logic;
                   --RW LCD
          rw : out std_logic;
                   -- Enable LCD
          e : out std_logic;
                   clk : in std_logic;
          reset : in std_logic);
end lcd;

architecture Behavioral of lcd is
-- component keypad baraye daryaft adad
 component keypad port(clk_keypad : in std_logic;
                                     reset_keypad : in std_logic;
                                     row : out std_logic_vector(3
downto 0);
                                     column : in std_logic_vector(3
downto 0);
                                     led : buffer std_logic_vector(3
downto 0);
                                                 digit : buffer
std_logic_vector(3 downto 0);
                                     key_pressed : out std_logic);
end component;
      -- tedad arayeh hafezeh
   constant number_of_data : integer := 19;
   constant delay_for_e : integer := 2;   --25 * 20 NS = 500  ns >450
NS
   constant delay_between_data : integer := 2000;-- 100,000 *20 ns =
2.4 MS
   constant Delay_1sec:integer:=    1000000; -- 50,000,000 * 20ns = 1
Second
   constant Delay_500ms:integer:=   500000; -- 25,000,000 * 20ns =
500ms
  type init_array is array(0 to 2)of std_logic_vector(7 downto 0);
  signal init: init_array:=("00111000", "00001110", "00000001");
   type two_dimen_array is array (0 to number_of_data - 1)
                           of std_logic_vector (7 downto 0);
   -- hafezaeh baraye namayesh lcd -
      --                            -3 bayte init lcd
      --                            - 4 bayte ramz
      --                            - 4 bayte codeshodeh ramz
      --             - 4 byte character code
      --             - 4 byte vorodi karbar

      signal table : two_dimen_array :=
                        ("00111000", "00001110", "00000001",
                     "00000000","00000000","00000000","00000000",
                     "00000000","00000000","00000000","00000000",
                     "00000000","00000000","00000000","00000000",
                     "00000000","00000000","00000000","00000000"
                  );
      -- FSM asli barnameh
```

```
   type state is (Start,SetKeys, Write_Change,GetKeys,CheckKeys,
Stop);
   signal current : state:=Start;
      -- FSM LCD
   type Delay_Type is(None,Enable,WriteData);
      signal Delay_State : Delay_Type:=None;
      signal Ok:std_logic:='0';
      signal L:std_logic:='0';
      -- Eyjad adad tasadofi
      signal Rand_num:std_logic_vector(3 downto 0):="1001";
      signal Rand_bit:std_logic:='0';
      signal l2: std_logic_vector(7 downto 0):="00000000";
      -- Ragham Vared shodeh
      signal Digit:  std_logic_vector(3 downto 0);

      signal key_press:std_logic;

begin
      keypad0: keypad port map(clk, reset, row, column, led, digit,
key_press);
   process (clk, reset,Key_press)
      variable i,Waitfor,KeyCount: integer:=0;
            variable Delay:integer:=123;
            variable c1,c2,c3,c4:integer:=0;
   begin
      if reset = '1' then
            current <= start;
         rs <= '0';
            rw <= '0';
            e <= '0';
            l<='0';
            keyCount:=0;
            LedOk <= '0';
            LedNotOk <= '0';
            Ok<='0';
            Delay := 0;
            i := 0;
            --led<="0000";
            Delay_State<= None;
            Delay:=delay_1Sec;
            waitfor:=0;
      elsif clk = '1' and clk'event then
                  -- eyjad adad tasadofi
            if c1=3 then
                     c1:=0;
                  else
                     c1:=c1+1;
                  end if;
            if c2=7 then
                     c2:=0;
                  else
                     c2:=c2+1;
                  end if;
            if c3=11 then
                     c3:=0;
                  else
                     c3:=c3+1;
                  end if;
            if c4=13 then
                     c4:=0;
```

```
                  else
                     c4:=c4+1;
                  end if;
                  rand_bit <= rand_num(3) xor rand_num(2);
                  rand_num(3 downto 1) <= rand_num(2 downto 0);
                  rand_num(0) <= rand_bit;
            -- delay baraye zaman bandi haye system
         if Delay>0 then
                  Delay:=Delay-1;
         else
                  case Delay_State is
                        -- Enable LCD
                        when Enable =>
                              e<='0';
                              Delay_State<= WriteData;
                              Delay :=delay_between_data;
                         -- Wait for write data  dar lcd
                        when WriteData =>
                           Delay_State <=None;
                        when None =>
                           if WaitFor>0 then
                                 WaitFor:=WaitFor-1;
                              else
                              case current is
                                    when start =>
                                          -- init lcd
                                          data(7 downto 0) <=
init(i)(7 downto 0);
                                          rs <= '0';
                                          rw<='0';
                                          e<='1';
                                          i := i + 1;
                                          Delay := delay_for_e;
                                          Delay_State<=Enable;
                                          if i >= 3 then
                                                waitfor:=delay_1Sec;
                                                current <= SetKeys;
                                                i:=0;
                                          end if;
                                    when SetKeys =>
                                                -- set random numbers
                                       if rand_Num<"1010" then
                                             table(i+3)(3 downto
0)<=  rand_num(3 downto 0);
                                             table(i+3)(7 downto
4)<="0000";
                                                 if
Rand_Num>"0100" then
      table(i+7)(3 downto 0)<= Rand_Num-5;
                                                 else
      table(i+7)(3 downto 0)<= Rand_NUM+5;
                                                 end if;
                                                 table(i+7)(7
downto 4)<="0000";
      delay:=c1*c2*c3*c4;
                                                 i:=i+1;
```

```
                                                        if i>=4 then


      waitfor:=delay_1Sec;
                                                                current
<= write_Change;
                                                                i:=0;
                                                        end if;
                                                    end if;
                                            when write_Change =>
                                                  -- sakht ramz az roye adad
tasadofi eyjad shodeh
                                                  --waitfor:=delay_500ms;
                                                  data<=table(i+7)(7 downto
0)+48;
                                                  rs <= '1';
                                                  rw<='0';
                                                  e<='1';
                                                  i:= i + 1;
                                                  Delay := delay_for_e;
                                                  Delay_State<=Enable;
                                                  l2<=l2+1;
                                                  if i >= 4 then
                                                        --data<="00100000";
                                                        i:=0;
                                                        current <= GetKeys;
                                                        KeyCount:=0;
                                                  end if;
                                             when GetKeys =>
                                                        --  gereftan adad
vorodi karbar
                                                        if Key_Press='1' then
                                                          -- l2<=l2+1;
                                                              if KeyCount<4
then
                                                                 case Digit
is

      when "0000" => Table(11+KeyCount)<="00000001";

      when "0001" => Table(11+KeyCount)<="00000010";

      when "0010" => Table(11+KeyCount)<="00000011";

      when "0100" => Table(11+KeyCount)<="00000100";

      when "0101" => Table(11+KeyCount)<="00000101";

      when "0110" => Table(11+KeyCount)<="00000110";

      when "1000" => Table(11+KeyCount)<="00000111";

      when "1001" => Table(11+KeyCount)<="00001000";

      when "1010" => Table(11+KeyCount)<="00001001";

      when "1101" => Table(11+KeyCount)<="00000000";

      when others => Table(11+KeyCount)<="11111111";
```

```
                                                                    end
case;

      KeyCount:=KeyCount+1;

      Delay:= delay_500ms;
                                                          else

      current<=CheckKeys;
                                                          end if;
                                                    end if;

                                        when CheckKeys =>
                                                    -- cheke adad vorodi
karbar ba ramz zakhireh shodeh dar hafezeh
                                                    Ok<='1';
                                                    if
table(i+3)=table(i+11) then
                                                       i:=i+1;
                                                          if i>=4 then
                                                             current<=
stop;
                                                                --i:=0;
                                                          end if;
                                                    else
                                                       Ok<='0';
                                                          --i:=0;
                                                          current<=Stop;
                                                    end if ;

                                       when Stop =>
                                                    -- payen FSM  va
namayesh vazeyat Ramz
                                            if Ok='1' then
                                                          LedOk<='1';
                                                    else
                                                       ledNotOk<='1';
                                                    end if;
                                                    --i:=0;
                                                    current<=Stop;

                                         when others=> --i:=0;
                                                    current<=Stop;
                                 end case;
                                 end if;
                        end case;
            end if;
               --led<=l2(3 downto 0);
         end if;
      end process;

end Behavioral;


--############################--> keypad <--
############################--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

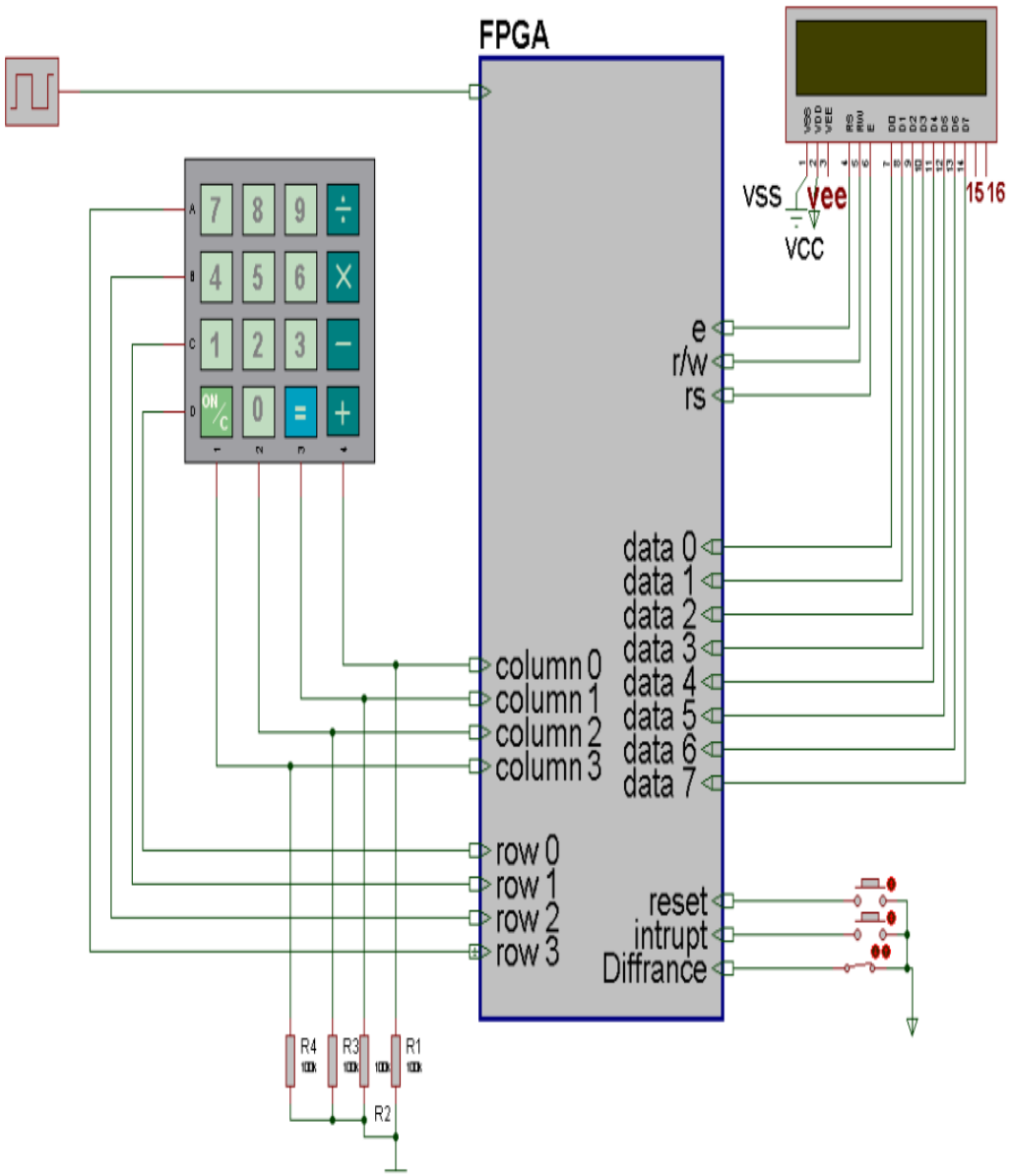
```
entity keypad is
      Port( clk_keypad : in std_logic;
        reset_keypad : in std_logic;
        row : out std_logic_vector(3 downto 0);
        column : in std_logic_vector(3 downto 0);
        led : out std_logic_vector(3 downto 0);
        digit : buffer std_logic_vector(3 downto 0);
        key_pressed : out std_logic);
end keypad;

architecture Behavioral of keypad is
-- faseleh beyn 2 key
constant debounce_time : integer := 300000;
signal debounce_flag : std_logic;
-- shomarandeh satr , code soton, code satr, soton ghabli
signal row_i, column_encoded, row_encoded, column_before :
std_logic_vector (3 downto 0);
signal count : integer;

begin
   process (clk_keypad, reset_keypad)
   begin
      if reset_keypad = '1' then
                count <= 0;
                digit <= "1111";
                row_i <= "1110";
                debounce_flag <= '0';
                          Key_pressed<='0';
     elsif clk_keypad= '1' and clk_keypad'event then
                          -- time beyne 2 key
               if debounce_flag = '1' then
                        count <= count + 1;
        else
                              count <= 0;
        end if;
              key_pressed <= '0';
         if column /= "1111" and debounce_flag = '0' then
                    -- if key preesed
                         column_before <= column;
                          debounce_flag <= '1';
           elsif count = debounce_time then
                          if column_before = column then
                                          -- if soton = soton ghabli
                            digit <= column_encoded + row_encoded;
                            key_pressed <= '1';
                            debounce_flag <= '0';
           else
                    debounce_flag <= '0';
           end if;
           elsif debounce_flag = '0' then
                             -- charkhesh satr
                          row_i <= row_i(2 downto 0) & row_i(3);
           end if;
      end if;
   end process;
      -- code Soton
   column_encoded <= "0000" when column = "1110" else
                                              "0001" when column =
"1101" else
```

```
                                                "0010" when column =
"1011" else
                                                "0011" when column =
"0111" else
                                                "0000";
      -- code satr
   row_encoded <= "0000" when row_i = "1110" else
                           "0100" when row_i = "1101" else
                                                "1000" when row_i = "1011"
else
                                                "1100" when row_i = "0111"
else
                                                "0000";

   row <= row_i;
      -- Namayesh ragham Key feshordeh shodeh
      led <= "0000" when digit = "1101" else
                     "0001" when digit = "0000" else
                     "0010" when digit = "0001" else
                     "0011" when digit = "0010" else
                     "0100" when digit = "0100" else
                     "0101" when digit = "0101" else
                     "0110" when digit = "0110" else
                     "0111" when digit = "1000" else
                     "1000" when digit = "1001" else
                     "1001" when digit = "1010";

end Behavioral;
```

۱۲- بازی حدس عدد

هدف، اتصال یک keypad و Lcd به FPGA می‌باشد تا بتوانیم از طریق یک کلید یک عدد تصادفی دو رقمی تولید کنیم و از طریق keypad بتوانیم آن عدد تصادفی تولید شده دو رقمی را حدس بزنیم، توسط Lcd نیز اعداد وارد شده توسط keypad و پیغام های مربوطه نمایش داده می شوند. این پروژه با دریافت عدد وارد شده از keypad آن را با عدد تصادفی تولید شده مقایسه می کند، در صورتیکه عدد وارد شده کوچکتر از عدد تصادفی بود، پیغام "is Small" را در Lcd نمایش می دهد، همچنین اگر عدد وارد شده بزرگتر بود پیغام "is Large" را نمایش می دهد، این کار آنقدر ادامه پیدا می کند تا عدد ورودی با آن عدد تصادفی تولید شده برابر شود، که در این صورت ابتدا پیغام "You Win" نمایش داده می شود و بعد از ۲ ثانیه آن عدد تولید شده همراه با تعداد دفعات حدس ما را نشان می دهد و پایان می یابد. در واقع این پروژه همان بازی حدس عدد می باشد و همچنین یک دکمه ای در fpga با شماره ۳۰ تعیین کرده ایم که برایمان تعداد اختلاف باینری را هم در کنار بزرگتر ، کوچکتر یا مساوی نشان می دهد.

مدار شکل زیر را ببندید و نتیجه را مشاهده کنید:



برنامه در ادامه آورده شده است. توضیحات لازم در مورد چگونگی کار برنامه و زیر برنامه های آن در خود برنامه قید شده است.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity guess_number is
      Port( clk : in std_logic;
        reset : in std_logic;
                -- LCD data
                   data : buffer std_logic_vector(7 downto 0);
                   -- KeyPad Rows
```

```
                row : out std_logic_vector(3 downto 0);
                   -- KeyPad Column
                column : in std_logic_vector(3 downto 0);
                   -- LCd RS
                rs : buffer std_logic;
                   -- LCD RW
        rw : out std_logic;
              -- LCD Enable
                e : out std_logic;
              -- Intrupt for key Pressed
        key_press : inout std_logic;
        led, digit: buffer std_logic_vector(3 downto 0);
              -- diff for show bit differ
                 Diff:in std_logic;
                   -- baraye eyjad yek adad tasadofi
                intrupt : in std_logic);
end guess_number;

architecture Behavioral of guess_number is
-- component key pad baraye daryaft adad ha
component keypad port(clk_keypad : in std_logic;
                                   reset_keypad : in std_logic;
                                   row : out std_logic_vector(3
downto 0);
                                   column : in std_logic_vector(3
downto 0);
                                   led : buffer std_logic_vector(3
downto 0);
                                              digit : buffer
std_logic_vector(3 downto 0);
                                   key_pressed : out std_logic);
end component;
-- component LCD baraye namayesh adad ha
component lcd port( clk_lcd : in std_logic;
                              reset_lcd : in std_logic;
                                          data : buffer
std_logic_vector(7 downto 0);
                              rs : buffer std_logic;
                              rw : out std_logic;
                              e : out std_logic;
                                       Diff:in std_logic;
                                              n : in
std_logic_vector(3 downto 0);
                                 key, intrupt :in std_logic);
end component;

begin
-- portmap KeyPad
keypad0: keypad port map(clk, reset, row, column, led, digit,
key_press);
-- Portmap LCD
lcd0: lcd port map(clk, reset, data, rs, rw, e,Diff, digit,
key_press, intrupt);

end Behavioral;

--############################--> keypad Component <--
############################--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity keypad is
      Port( clk_keypad : in std_logic;
        reset_keypad : in std_logic;
        row : out std_logic_vector(3 downto 0);
        column : in std_logic_vector(3 downto 0);
        led : out std_logic_vector(3 downto 0);
        digit : buffer std_logic_vector(3 downto 0);
        key_pressed : out std_logic);
end keypad;

architecture Behavioral of keypad is
-- faseleh bayn 2 kelid
constant debounce_time : integer := 100000;
-- zaman beyn 2 kelid
signal debounce_flag : std_logic;
-- satr , code soton ,code satr, soton ghably
signal row_i, column_encoded, row_encoded, column_before :
std_logic_vector (3 downto 0);
signal count : integer;

begin
   process (clk_keypad, reset_keypad)
   begin
      if reset_keypad = '1' then
               count <= 0;
               digit <= "1111";
               row_i <= "1110";
               debounce_flag <= '0';
      elsif clk_keypad= '1' and clk_keypad'event then
               -- agar kelidi feshar dadeh shodeh ta 0 shodan count
sabr kon
                        if debounce_flag = '1' then
                     count <= count + 1;
         else
                          count <= 0;
         end if;
             key_pressed <= '0';
          -- if Column <>1111 va dar halat beyn do kelid nistim kelid
jadid ra daryaft kon
                   if column /= "1111" and debounce_flag = '0' then
                     column_before <= column;
                           debounce_flag <= '1';
            elsif count = debounce_time then
                              -- if soton ghabli barabar soton feli
hast ragham soton ra mohasebeh kon
                           if column_before = column then
                             digit <= column_encoded + row_encoded;
                             key_pressed <= '1';
                             debounce_flag <= '0';
            else
                    debounce_flag <= '0';
            end if;
                        -- if beyn 2 kelid nist satr ra shift bedeh
            elsif debounce_flag = '0' then
                           row_i <= row_i(2 downto 0) & row_i(3);
            end if;
      end if;
```

```
   end process;
      -- code nemodan soton
   column_encoded <= "0000" when column = "1110" else
                                                "0001" when column =
"1101" else
                                                "0010" when column =
"1011" else
                                                "0011" when column =
"0111" else
                                                "0000";
      -- code nemodan satr
   row_encoded <= "0000" when row_i = "1110" else
                           "0100" when row_i = "1101" else
                                                "1000" when row_i = "1011"
else
                                                "1100" when row_i = "0111"
else
                                                "0000";

      row <= row_i;

      -- namayesh argham bar rouye led
      led <= "0000" when digit = "1101" else
                     "0001" when digit = "0000" else
                     "0010" when digit = "0001" else
                     "0011" when digit = "0010" else
                     "0100" when digit = "0100" else
                     "0101" when digit = "0101" else
                     "0110" when digit = "0110" else
                     "0111" when digit = "1000" else
                     "1000" when digit = "1001" else
                     "1001" when digit = "1010";

end Behavioral;

--############################--> lcd <--
############################--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity lcd is
      Port( clk_lcd : in std_logic;
        reset_lcd : in std_logic;
        data : buffer std_logic_vector(7 downto 0);
        rs : buffer std_logic;
        rw : out std_logic;
        e : out std_logic;
        n : in std_logic_vector(3 downto 0);
              Diff: in std_logic;
        key, intrupt:in std_logic);
end lcd;

architecture Behavioral of lcd is

constant delay_for_e : integer := 1;       --1 u s > 450ns
constant delay_between_data_1 : integer := 10000; -- 10000 1 us=10ms
constant delay_between_data_2 : integer := 2000000;   --2000000* 1us=
2s
```

```
type state is (init, start, enter, write, compare,Diffrence, delay,
win, stop);
signal current : state;
signal i , j , Counts, count, part, part_write , part_win ,
delay_data : integer;
signal n1, n2, c1, c2 : std_logic_vector(3 downto 0);
signal enter_n: std_logic;

type rom_2_4 is array (0 to 1) of std_logic_vector (3 downto 0);
type rom_2_8 is array (2 to 0) of std_logic_vector (7 downto 0);
type rom_3_8 is array (0 to 2) of std_logic_vector (7 downto 0);
type rom_7_8 is array (0 to 6) of std_logic_vector (7 downto 0);
type rom_8_8 is array (0 to 7) of std_logic_vector (7 downto 0);
type rom_11_8 is array (0 to 10) of std_logic_vector (7 downto 0);
type rom_14_8 is array (0 to 13) of std_logic_vector (7 downto 0);

constant table : rom_3_8 := ("00111000", "00001110", "00000001");  --
command lcd (two rows, set curser, clear)
constant you_win : rom_7_8 :=
("01011001","01101111","01110101","00100000","01010111","01101001","0
1101110");  -- You Win
constant is_small : rom_8_8 :=
("01001001","01110011","00100000","01010011","01101101","01100001","0
1101100","01101100");  -- Is Small
constant is_large : rom_8_8 :=
("01001001","01110011","00100000","01001100","01100001","01110010","0
1100111","01100101");  -- Is Large
Constant number : rom_11_8 :=
("01001110","01110101","01101101","01100010","01100101","01110010","0
0100000","01001001","01110011","00111101","00100000"); -- Number Is:
constant guess : rom_14_8 :=
("01001110","01110101","01101101","00100000","01000111","01110101","0
1100101","01110011","01110011","00100000","01001001","01110011","0011
1101","00100000");  -- Num Guess is =
constant enter_number : rom_14_8 :=
("01000101","01101110","01110100","01100101","01110010","00100000","0
1001110","01110101","01101101","01100010","01100101","01110010","0011
1010","00100000");  -- Enter Number =
signal Dif1, dif2 :std_logic_vector(7 downto 0);
begin
   process (clk_lcd, reset_lcd)
       variable t : rom_2_4;   -- for save input number
   begin
--############################--> Reset <--
############################--
            if reset_lcd = '1' then
                  current <= init;
                  rs <= '0';
                  rw <= '0';
                  e <= '0';
                  i <= 0;
                  j <= 0;
                  count <= 0;
                  enter_n <= '0';
                  Counts <= 0;
                  part <= 0;
                  part_write <= 0;
                  part_win <= 0;
                  n1 <= "0000";
```

```
      n2 <= "0000";
      c1 <= "0000";
      c2 <= "0000";
                  data <= "00000000";
                  t := (others=>(others=>'1'));

      elsif clk_lcd = '1' and clk_lcd'event then
            case current is
--############################--> init <--
############################--
                  when init =>   -- Lcd Define
                        j<=0;
                        t := (others=>(others=>'1'));
                        part_write <= 0;
                        data <= table(i);
            rs <= '0';
                        if i<2 then
                              i<=i+1;
                        else
                              part_write <= 1;
                              i<=0;
                        end if;
                        delay_data <= delay_between_data_1;
                        current<=write;
--############################--> start <--
############################--
                  when start =>    -- send "enter_number:" to Lcd
                        data <= enter_number(i);
                        rs <= '1';
                        if i<13 then
                              i<=i+1;
                        else
                              part_write <= 3;
                              i<=0;
                        end if;
                        delay_data <= delay_between_data_1;
                        current<=write;
--############################--> enter <--
############################--
            when enter =>    -- Wait for Press Key from Keypad
            if (c2<"1001")then   -- counter Random Number
                  c2<= c2+1;
            else
                  c1<= c1+ 1;
                  c2<= "0000";
                  if c1 = "1001" then
                        c1 <= "0000";
                  end if;
            end if;

                        if ( intrupt ='1' )then  -- output Random
Number
                  n1 <= c1;    -- dahgan
                  n2 <= c2;    -- yekan
                        end if;

                        rs <= '1';
                        part_write <= 3;

                        case n is
```

```
                                    when "1101" => data <="00110000";
      enter_n <= '1';
                                    when "0000" => data <="00110001";
      enter_n <= '1';
                                    when "0001" => data <="00110010";
      enter_n <= '1';
                                    when "0010" => data <="00110011";
      enter_n <= '1';
                                    when "0100" => data <="00110100";
      enter_n <= '1';
                                    when "0101" => data <="00110101";
      enter_n <= '1';
                                    when "0110" => data <="00110110";
      enter_n <= '1';
                                    when "1000" => data <="00110111";
      enter_n <= '1';
                                    when "1001" => data <="00111000";
      enter_n <= '1';
                                    when "1010" => data <="00111001";
      enter_n <= '1';
                                    when "1110" =>
                                          data <= "00000001";
                                          rs <= '0';
                                          enter_n <= '0';
                                          part_write <= 2;
                                    when "1100" =>
                                          data <= "00000001";
                                          rs <= '0';
                                          enter_n <= '0';
                                          part_write <= 0;
                                    when others => current <= enter;
enter_n <= '0';
                        end case;

                        delay_data <= 100000;

                  if key ='1' then
                        current <= write;
                  else
                        current <= enter;
                  end if;
--############################--> write <--
############################--
            when write =>
                  if part = 0 then
                              if enter_n ='1' then   -- Wait for
Enter Number from Keypad
                                    if j<2 then
                                          t(j)(3 downto 0) := data(3
downto 0);  -- got number from krypad
                                    end if;
                                    j<=j+1;
                           end if;

                              e <= '1';
                           count <= delay_for_e;
                           part <= part + 1;
                           current <= delay;

                        elsif part = 1 then
```

```
                        e <= '0';
                        count <= delay_data;
                        current <= delay;
                        part <= part + 1;

                        else
                              part <= 0;
                              case part_write is     -- select state
from part_write
                                    when 0 => current <= init;
                                    when 1 => current <= start;
                                    when 2 => current <= compare;  rs
<= '1';
                                    when 3 => current <= enter;
                                    when 4 => current <= win;
                                    when 5 => current <= Diffrence;
                                    when others => null;
                              end case;
                        end if;
--############################--> compare <--
############################--
                  when compare =>
                        part_write<=2;

                        if (t(0)/="1111" and t(1)="1111") then  --
for numbers < 10
                              t(1) := t(0);
                              t(0) := "0000";
                        end if;

                        delay_data <= delay_between_data_1;


            if (t(0) = n1 and t(1) = n2) then   -- you win
                  if (i<7) then  -- send "you_win" to lcd
                                    data <= you_win(i);
                                    i<=i+1;
                                    current <= write;
                                    if i=6 then   -- after send
"you_win" to lcd, wait 2s
                                          delay_data <=
delay_between_data_2;
                                    end if;
                              else
                                    Counts <= Counts +1;  -- INC
guess counter
                                    i<=0;
                                    current<=win;
                              end if;

            elsif ((t(0) < n1) or (t(0) = n1 and t(1) < n2)) then --
your number is small
                  if (i<8) then    -- send "is_Small" to lcd
                                    data <= is_small(i);
                                    i<=i+1;
                                    current <= write;
                                    if i=7 and Diff='0' then   --
after send "is_Small" to lcd, wait 2s
                                          delay_data <=
delay_between_data_2;
```

```
                                    end if;
                              else

                                    Counts <= Counts +1;  -- INC
guess counter
                                    -- if Diff=1 then show byte
Diffrence
                                    if Diff='0' then
                                          current <= Init;
                                    else
                                          current <= Diffrence;
                                    end if ;
                                    Dif1<="00000000";
                                    Dif2<="00000000";
                                    i<=0;
                                    j<=0;
                              end if;

            elsif ((t(0) > n1) or (t(0) = n1 and t(1) > n2)) then --
your number is large

                              if (i<8) then    -- send "is_Large" to
lcd
                                    data <= is_large(i);
                                    current <= write;
                                    if i=7  and Diff='0' then   --
after send "is_Large" to lcd, wait 2s
                                          delay_data <=
delay_between_data_2;
                                    end if;
                                    i<=i+1;
                              else

                                    Counts <= Counts +1;  -- INC
guess counter
                                    -- if Diff=1 then show byte
Diffrence
                                    if Diff='0' then
                                          current <= Init;
                                    else
                                          current <= Diffrence;
                                    end if ;
                                    Dif1<="00000000";
                                    Dif2<="00000000";
                                    i<=0;
                                    j<=0;

                              end if;

            end if;
                        -- Mohasebeh ekhtelaf bitha
                  when Diffrence=>
                              Part_Write<=5;
                           if i=0 then
                              if j<4 then
                                    if t(0)(j)/=n1(j) then
                                          Dif1<=Dif1+1;
                                     end if;
                                    if t(1)(j)/=n2(j) then
                                          Dif2<=Dif2+1;
```

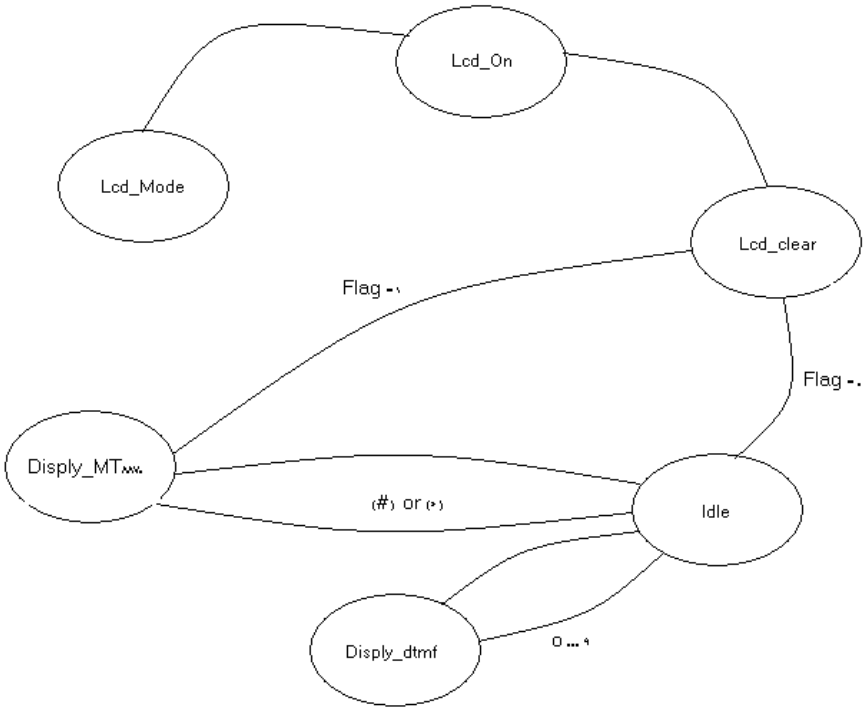
```
                                        end if;
                                        j<=j+1;
                                  else --j>=4
                                      j<=0;
                                            i<=i+1;
                                  end if;
                            else -- i>=2
                                if i=1 then
                                     Data<=48+Dif1+dif2;
                                        current<=write;
                                                  delay_data <=
2*delay_between_data_2;
                                                   i<=i+1;
                                  else
                                     current<=Init;
                                        i<=0;
                                     end if;
                            end if;
--############################--> delay <--
############################--
            when delay =>
                        count <= count - 1;
                        if (count = 0) then
                        current <= write;
                              count<=0;
                  end if;
--############################--> win <--
############################--
                  when win =>
                        part_write<=4;
                        case part_win is
                        when 0 =>
                              data <= "00000001";  -- clear lcd
                        rs <= '0';
                              part_win <= part_win + 1;
                              current<=write;

                        when 1 =>
                              if i<11 then   -- send "Number is =" to
lcd
                                    data <= number(i);
                              rs <= '1';
                                    i<=i+1;
                                    current<=write;
                              else
                                    part_win <= part_win + 1;
                                    i<=0;
                              end if;

                        when 2 =>
                              case t(0) is   -- send random
number(dahgan) to lcd
                                    when "0000" => data<="00110000";
                                    when "0001" => data<="00110001";
                                    when "0010" => data<="00110010";
                                    when "0011" => data<="00110011";
                                    when "0100" => data<="00110100";
                                    when "0101" => data<="00110101";
                                    when "0110" => data<="00110110";
                                    when "0111" => data<="00110111";
```

```
                                    when "1000" => data<="00111000";
                                    when "1001" => data<="00111001";
                                    when others => null;
                              end case;
                              rs <= '1';
                              part_win <= part_win + 1;
                              current<=write;

                        when 3 =>
                              case t(1) is   -- send random
number(yekan) to lcd
                                    when "0000" => data<="00110000";
                                    when "0001" => data<="00110001";
                                    when "0010" => data<="00110010";
                                    when "0011" => data<="00110011";
                                    when "0100" => data<="00110100";
                                    when "0101" => data<="00110101";
                                    when "0110" => data<="00110110";
                                    when "0111" => data<="00110111";
                                    when "1000" => data<="00111000";
                                    when "1001" => data<="00111001";
                                    when others => null;
                              end case;
                              rs <= '1';
                              part_win <= part_win + 1;
                              current<=write;

                        when 4 =>
                              data<= "11000000";   -- next line lcd
                              rs <= '0';
                              current<=write;
                              part_win <= part_win + 1;

                        when 5 =>
                              if i<14 then   -- send "Num Guess is ="
to lcd
                                    data<= guess(i);
                                    rs <= '1';
                                    current<=write;
                                    i<=i+1;
                              else
                                    part_win <= part_win + 1;
                              end if;

                        when 6 =>
                              case Counts is    -- send Countse guess
number(dahgan) to lcd
                                    when 1 | 10 | 11 | 12 | 13 | 14 |
15 | 16 | 17 | 18 | 19 => data<="00110001";
                                    when 2 | 20 | 21 | 22 | 23 | 24 |
25 | 26 | 27 | 28 | 29 => data<="00110010";
                                    when 3 | 30 | 31 | 32 | 33 | 34 |
35 | 36 | 37 | 38 | 39 => data<="00110011";
                                    when 4 | 40 | 41 | 42 | 43 | 44 |
45 | 46 | 47 | 48 | 49 => data<="00110100";
                                    when 5 | 50 => data<="00110101";
                                    when 6 => data<="00110110";
                                    when 7 => data<="00110111";
                                    when 8 => data<="00111000";
                                    when 9 => data<="00111001";
```

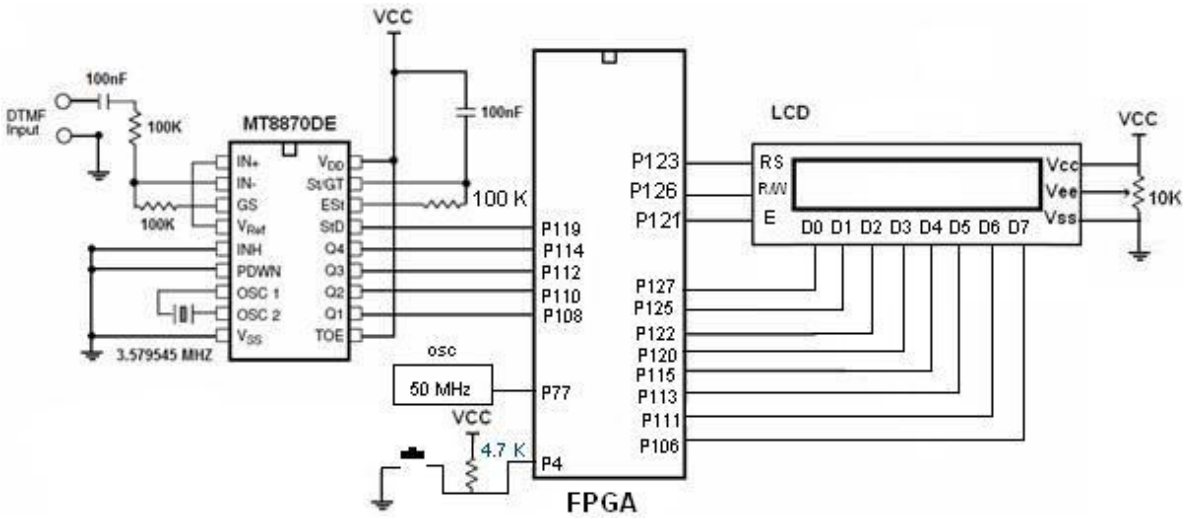
```
                                    when others => null;
                                end case;

                                if (Counts > 9) then
                                    part_win <= part_win + 1;
                                elsif (Counts<10) then    -- if Counts
< 10 goto Stop
                                    part_win <= part_win + 2;

                                end if;
                                rs <= '1';
                                current<=write;

                            when 7 =>
                                case Counts is    -- send Countse guess
number(yekan) to lcd
                                    when 10 | 20 | 30 | 40 | 50 =>
data <="00110000";
                                    when 11 | 21 | 31 | 41 =>
data<="00110001";
                                    when 12 | 22 | 32 | 42 =>
data<="00110010";
                                    when 13 | 23 | 33 | 43 =>
data<="00110011";
                                    when 14 | 24 | 34 | 44 =>
data<="00110100";
                                    when 15 | 25 | 35 | 45 =>
data<="00110101";
                                    when 16 | 26 | 36 | 46 =>
data<="00110110";
                                    when 17 | 27 | 37 | 47 =>
data<="00110111";
                                    when 18 | 28 | 38 | 48 =>
data<="00111000";
                                    when 19 | 29 | 39 | 49 =>
data<="00111001";
                                    when others => data <="00110000";
                                end case;
                                part_win <= part_win + 1;
                                rs <= '1';
                                current<=write;

                            when others => current <= stop;

                            end case;

                            delay_data <= delay_between_data_1;
--############################--> stop <--
############################--
                        when stop =>null;
                    end case;
            end if;
            end process;
end Behavioral;
```

۱۳- اتصال 8870 به FPGA برای دریافت اطلاعات تن تلفن

State_Diagram:



مدار زیر را ببندید و نتیجه را مشاهده کنید.



شکل ۱۲-۱ مدار تشخیص تن صدای تلفن با FPGA

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity dtmf_fpga is
port(reset:in std_logic;
            clk:in std_logic;
            dtmf_in:in std_logic_vector(3 downto 0);
            std_in:in std_logic;
            lcd_data:out std_logic_vector(7 downto 0);
            lcd_rs:out std_logic;
            lcd_rw:out std_logic;
            lcd_e:out std_logic);
end dtmf_fpga;

architecture Behavioral of dtmf_fpga is
type dtmf_state

   is(lcd_mode,lcd_on,lcd_clear,idle,display_dtmf,display_mt8870de);

signal flag:std_logic;
begin
lcd_rw<='0';
dtmfff:process(std_in,reset,clk)
variable state:dtmf_state:=idle;
variable c:integer;
variable c2:integer;
      begin
            if(reset='1')then
                  state:=lcd_mode;
                  c:=0;
                  c2:=0;
                  flag<='1';
      elsif (clk='1' and clk'event) then
                  c2:=c2+1;
                  if (c2=10000)then
                  c2:=0;
                  c:=c+1;
                  end if;
      case  state is
      when lcd_mode     <=    lcd_data<=X"30";
                  lcd_rs<='0; '
      if   c<1000 then
                  lcd_e<='1; '
      Elsif  (c>1000 and c<2000) then
                  lcd_e<='0; '
      Elsif  c=2000  then
                  c:=0;
                  state:=lcd_on;
      end if;
      when lcd_on       <=          lcd_data<=X"0E; "
                  lcd_rs<='0; '


      if  c<1000 then
                  lcd_e <='1; '
      elsif   (c>1000 and c<2000) then
                  lcd_e<='0; '
      Elsif  c=2000 then
                  c:=0;
                  state:=lcd_clear;
```

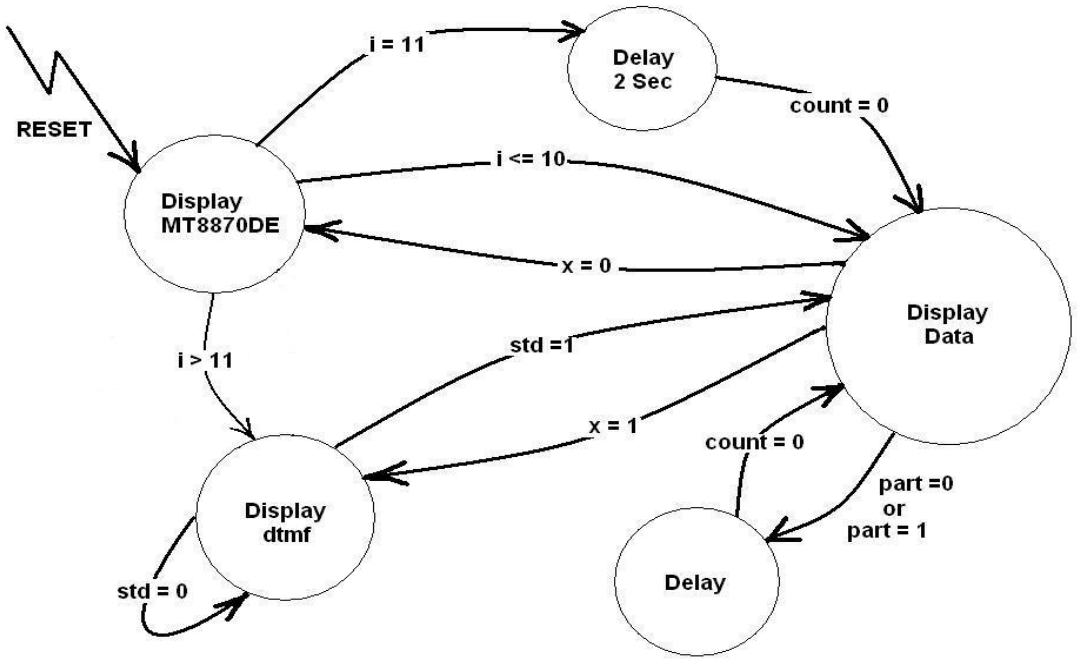
```
end if;
when lcd_clear    =>    lcd_data<=X"01; "
            lcd_rs<='0; '
if  c<1000 then
            lcd_e <='1; '
elsif  (c>1000 and c<2000) then
            lcd_e<='0; '
Elsif  c=2000 then
            if flag='1' then
            state:=display_mt8870de;
            c:=0;
            flag<='0';
            else
            c:=0;
            state:=idle;
            end if;
end if;
when idle   <=    if( std_in='1')then

      if    dtmf_in="1010"then
            lcd_data<=X"30; "
            state:=display_dtmf;
            c:=0;
      elsif dtmf_in="0001"then
            lcd_data<=X"31; "
            state:=display_dtmf;
            c:=0;
      elsif dtmf_in="0010"then
            lcd_data<=X"32; "
            state:=display_dtmf;
            c:=0;
      elsif dtmf_in="0011"then
            lcd_data<=X"33; "
            state:=display_dtmf;
            c:=0;
      elsif dtmf_in="0100"then
            lcd_data<=X"34; "
            state:=display_dtmf;
            c:=0;
      elsif dtmf_in="0101"then
            lcd_data<=X"35; "
            state:=display_dtmf;
            c:=0;
      elsif dtmf_in="0110"then
            lcd_data<=X"36; "
            state:=display_dtmf;
            c:=0;
      elsif dtmf_in="0111"then
            lcd_data<=X"37; "
            state:=display_dtmf;
            c:=0;
      elsif dtmf_in="1000"then
            lcd_data<=X"38; "
            state:=display_dtmf;
            c:=0;
      elsif dtmf_in="1001"then
            lcd_data<=X"39; "
            state:=display_dtmf;
            c:=0;
      elsif dtmf_in="1011"then
```

```
                  c:=0;
                  state:=display_mt8870de;
            elsif dtmf_in="1100"then
                  c:=0;
                  state:=display_mt8870de;
            end if;
      end if;
when display_dtmf      =>     lcd_rs<='1; '
            if    c<1000      then
                  lcd_e<='1; '
            elsif )c>1000 and c<2000) then
                  lcd_e<='0; '
            elsif c=2000 then
                  c:=0;
                  state:=idle;
      end if;
            when display_mt8870de<=
            if  c<500   then
                  lcd_data<=X"01";
                  lcd_rs<='0; '
                  lcd_e<='1;'
                  elsif (c>=500 and c<1250) then
                  lcd_e<='0; '
            elsif (c>=1250 and c<1500) then
                  lcd_data<=X"4D";        ---m
                  lcd_rs<='1;'
                  lcd_e<='1; '
            elsif )c>=1500 and c<1750) then
                  lcd_e<='0; '
            elsif (c>=1750 and c<2000) then
                  lcd_data<=X"54";        ---t
                  lcd_rs<='1; '
                  lcd_e<='1; '
            elsif )c>=2000 and c<2250) then
                  lcd_e<='0; '
            elsif (c>=2250 and c<2500)then
                  lcd_data<=X"38";        ---۸
                  lcd_rs<='1; '
                  lcd_e<='1; '
            elsif )c>=2500 and c<2750) then
                  lcd_e<='0; '
            elsif (c>=2750 and c<3000)then
                  lcd_data<=X"38";        ---۸
                  lcd_rs<='1; '
                  lcd_e<='1; '
            elsif )c>=3000 and c<3250) then
                  lcd_e<='0; '
            elsif (c>=3250 and c<3500)then
                  lcd_data<=X"37";        ---۷
                  lcd_rs<='1; '
                  lcd_e<='1; '
            elsif(c>=3500 and c<3750) then
                  lcd_e<='0; '
            elsif (c>=3750 and c<4000) then
                  lcd_data<=X"30";        ---۰
                  lcd_rs<='1; '
                  lcd_e<='1; '
            elsif )c>=4000 and c<4250 ) then
                  lcd_e<='0; '
            elsif (c>=4250 and c<4500) then
```

```
                lcd_data<=X"44";        ---d
                lcd_rs<='1; '
                lcd_e<='1; '
          elsif )c>=4500 and c<4750 ) then
                lcd_e<='0; '
          elsif (c>=4750 and c<5000 ) then
                lcd_data<=X"45";        ---e
                lcd_rs<='1; '
                lcd_e<='1; '
          elsif )c>=5000 and c<10000) then
                lcd_e<='0; '
          elsif (c>=10000 and c<10250 ) then
                lcd_data<=X"01;"
                lcd_rs<='0; '
                lcd_e<='1; '
          elsif )c>=10250 and c<10550) then
                lcd_e<='0; '
          elsif c=10550 then
                c:=0;
                state:=idle;
          end if;
    end case;
    end if;
    end process;
end Behavioral;
```

کد دیگر:

دیاگرام حالت این کد به صورت زیر است.



دیاگرام حالت برای برنامه پروژه MT8870DE

کد برنامه نرم افزاری مدار پروژه به زبان VHDL در زیر آورده شده است :

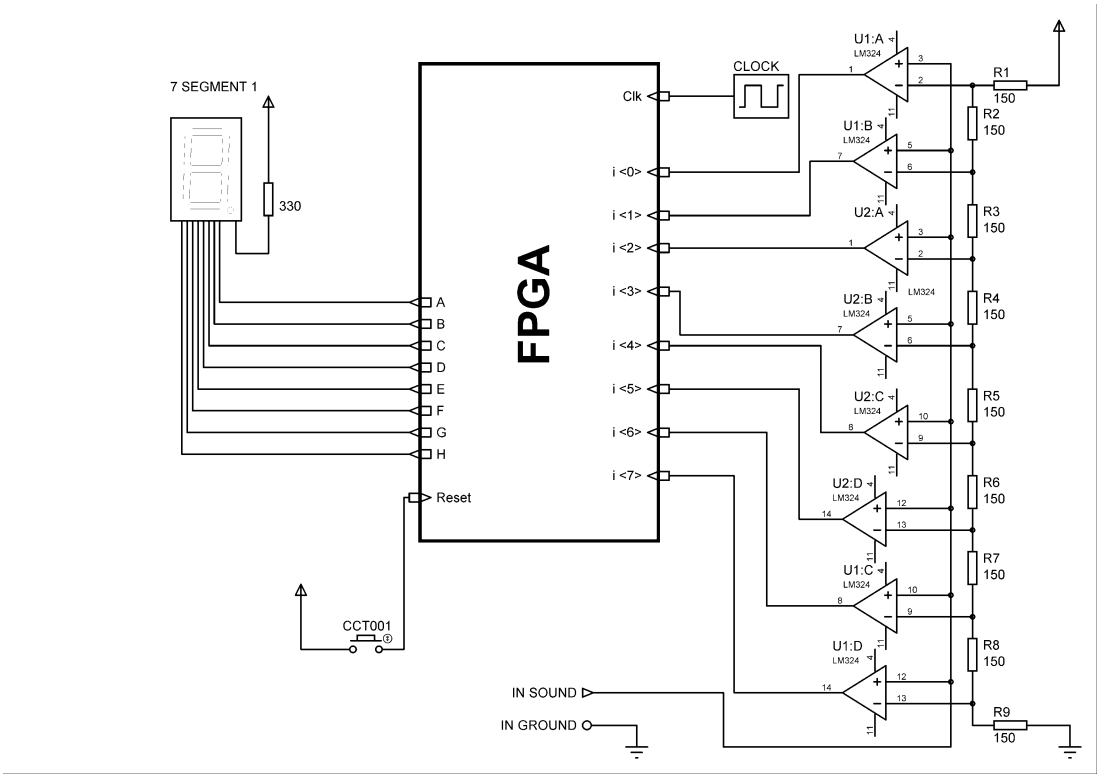
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity modified_8870 is
port(reset,clk,std_in:in std_logic;          -- "std_in" is input
from MT8870DE
dtmf_in:in std_logic_vector(3 downto 0);  --4 Bit input from MT8870DE
lcd_data:out std_logic_vector(7 downto 0);  --8 Bit output to LCD
lcd_rs,lcd_rw,lcd_e:out std_logic   --Output control pins for LCD
       );
end modified_8870;
Architecture Behavioral of modified_8870 is
Constant number_of_data : integer := 11;  -- Number of array
elements,3 commands + 8 data.
Constant delay_for_e : integer := 23; -- 23 * 20ns = 460 ns .
Constant delay_between_data : integer := 1000000; -- 1000000 * 20ns =
20ms.
Constant delay_for_show : integer := 100000000; -- 100000000 * 20ns =
2 sec;
Constant delay_for_std : integer := 10000000;--10000000 * 20ns =
200ms.
Type show_array is array (0 to number_of_data - 1) of
std_logic_vector (8 downto 0);    --prototype of 1D*1D array.
constant table : show_array := ("000111000", "000001110",
"000000001",
                                                    "101001101",
"101010100", "100111000",
                                                    "100111000",
"100110111", "100110000",
                                                    "101000100",
"101000101" ); -- MT8870DE
type state is
(display_mt8870de,display_dtmf,display_data,delay,delay_2sec); --
Prototype the States.
signal current : state;                    -- Define a signal from
states.
signal part : integer range 0 to 3;  -- Declear the part signal for
partitioning the "display_data"
begin
process(reset,clk)
variable x,i,count : integer := 0;
begin
if (reset = '1') then
current<= display_mt8870de ;
lcd_rs<= '0 ;'
lcd_rw<= '0 ;'
lcd_e<= '0;'
part<= 0;
elsif (clk = '1' and clk'event) then        --Cheking the rising edge
of input clock pulse.
case current is
when display_mt8870de =>--in "display_MT8870DE" state
if i <= (number_of_data-1) then   --if showing the content of the
table is incompelete,
lcd_data<= table(i)(7 downto 0);  -- i th content of table goes to
LCD "data" pins.
```

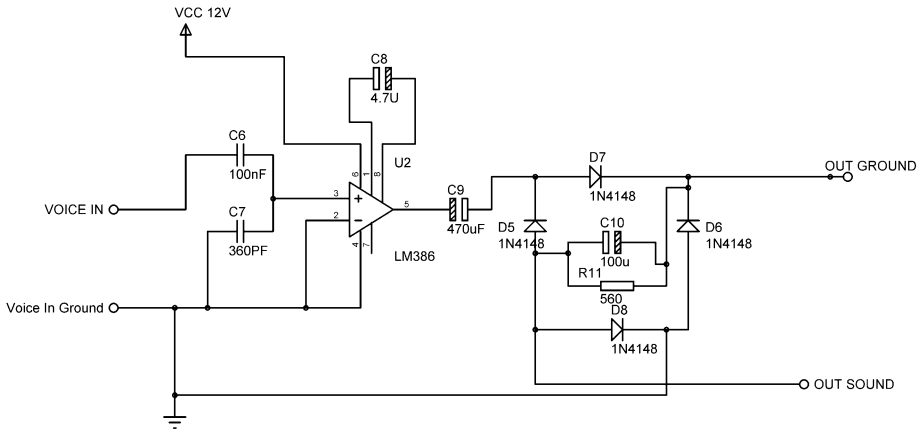
```
lcd_rs<= table (i)(8);    -- MSB goes to LCD "rs" pin.
x := 0;                        -- identifier "x" must be (0) in this
state.
i := i+1;              -- increament the counter i for the next data.
current<= display_data; -- go to "display_data" for showing the i th
contenton LCD
elsif i = number_of_data then       -- if all of data are shown,
count := delay_for_show;          -- assign the delay_for_show to
signal "count."
current <= delay_2sec;    -- go to "delay_2sec" that lead to display
the "MT8870DE" on LCD for 2 seconds and then clear the screen.
i := i+1;        -- increament the counter i , because the program
shouldn't come in to this state again until the next reset.
else        -- when the "MT8870DE"is shown on LCD and screen cleared.
i :=0;            -- reset the counter i for next use of this state.
current<= display_dtmf; -- go to "display_dtmf" for checking the
inputs from 8870
end if;
when display_dtmf =>    -- in "display_dtmf" state,
x := 1;     -- identifier "x" must be  (1) in this state.
if (std_in ='1') then    -- if 4-bit data is available form IC 8870,
lcd_rs<= '1';    -- assume all of these inputs as DATA
if (dtmf_in = "1010") then   -- if number 0 is pressed on telephone
keypad ,
lcd_data<= X"30";   -- ASCII code for number 0 goes to LCD "data"
pins.
current<= display_data; -- go to "display_data" for showing the
character 0.
elsif (dtmf_in  = "1011") then   -- if * key is pressed on telephone
keypad,
lcd_data<= X"2A";  -- ASCII code for character * goes to LCD "data"
pins   .
current<= display_data; -- go to "display_data" for showing the
character.*
elsif (dtmf_in = "1100") then  -- if # key is pressed on telephone
keypad,
lcd_data<= X"23";  -- ASCII code for character # goes to LCD "data"
pins.
current<= display_data; -- go to "display_data" for showing the
character.#
else      -- means that a number between 1-9 is pressed on thelephone
keypad,
lcd_data<= dtmf_in + X"30";  -- add this value with 0's ASCII code
for generate the ASCII code for that number.
current<= display_data;  -- go to "display_data" for showing that
number.
end if;
else               -- if none of the keys are pressed
current<= display_dtmf;     -- stay in this state until one key
pressed.
end if;
when display_data =>        -- in "display_data"state,
if (part = 0) then         -- in the first partition of this state,
lcd_e<= '1';             -- activate the "enable" pin of LCD.
count := delay_for_e;   -- assign the delay_for_e to signal "count."
part<= part + 1;     -- increament the signal part,because in the
next reference to this state,state must begin from second partition
current<= delay; -- go to "delay" to apply delay for 450 ns .
elsif (part = 1) then   -- in the second partition of this state,
```

```
lcd_e<= '0';          -- deactivate the "enable" pin of LCD after 450
ns and show the content on the LCD.
count := delay_between_data; -- assign the delay_between_data to
signal "count."
current<= delay; -- go to "delay" to apply  delay for 20ms .
part<= part + 1;    -- increament the signal part,because in the next
reference to this state,state must begin from 3rd partition .
elsif (part = 2) then  -- in the 3rd partition of this state,
if (x = 0) then   -- check the value of x that if it is 0 then.
            part<= 0;       -- reset the part value,
current<= display_mt8870de; -- and come back to state
"display_mt8870de."
elsif (x = 1) then -- check the value of x that if it is 1 then.
            count := delay_for_std; -- assign the delay_for_std to
signal "count" for avoid receive repeatitive data.
            current<= delay; -- go to "delay" to apply  delay for
200ms .
part<= part + 1;  -- increament the signal part,
end if;
else                           -- when Part = 3
part<= 0;
current<= display_dtmf; -- go back to state "display_dtmf."
end if;
when delay =>              -- in state "delay"
count := count - 1;    -- decreament the count value.
if (count = 0) then         -- when count = 0
current<= display_data;-- go back to state "display_data."
end if;
when delay_2sec =>     -- in state "delay_2sec"
count := count - 1;          -- decreament the count value.
if (count = 0) then       -- when count = 0
lcd_data<= X"01";    -- clear screen command goes on LCD data pins.
lcd_rs<= '0';    -- assume this value as a command
current<= display_data; -- go to state "display_data" to apply this
command on LCD.
end if;
end case;
end if;
end process;
end behavioral;
```

۱۴- رقص نور

شکل مدار:





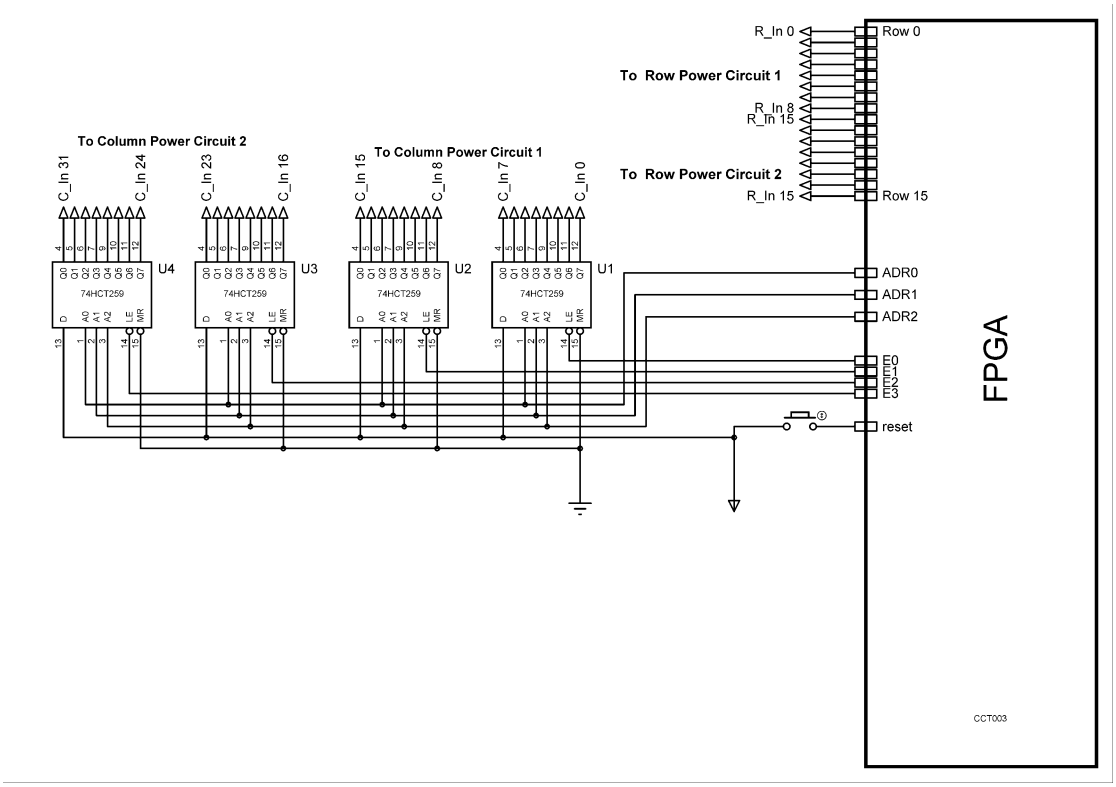
برنامه:

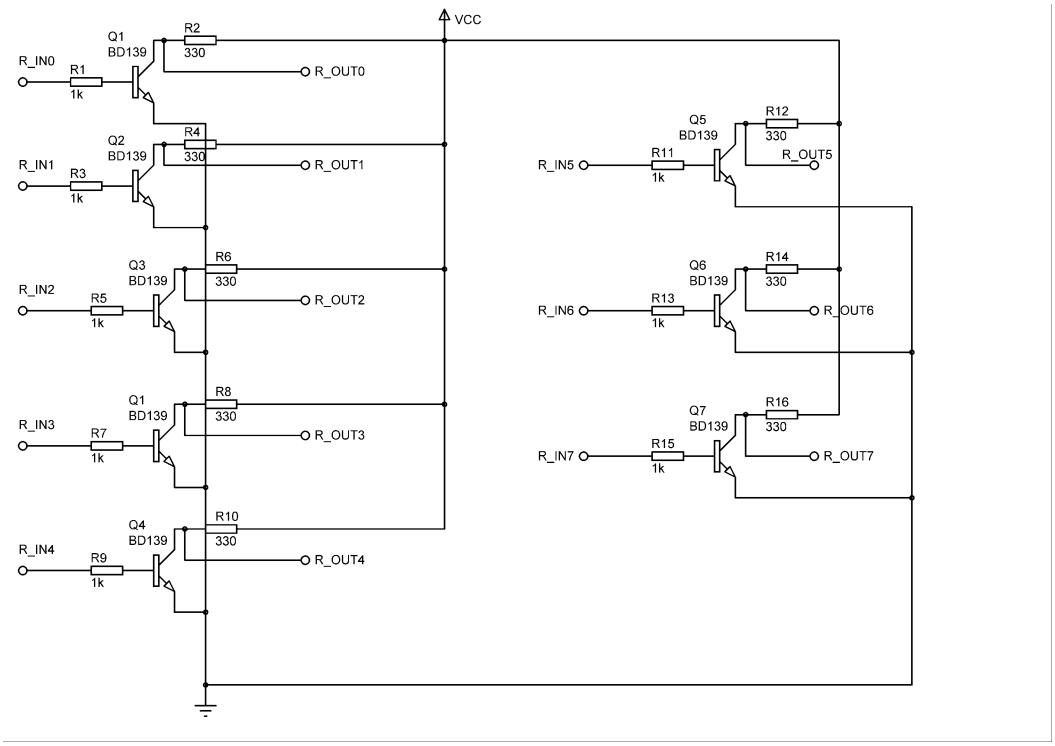
```
----------------------------------------------------------------------
-----------
-- Company:
-- Engineer:
--
-- Create Date:    16:54:02 11/30/10
-- Design Name:
-- Module Name:    LiteDancer - Behavioral
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------
-----------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity LiteDancer is
    Port ( clk :in std_logic;
              i : in std_logic_vector(7 downto 0);
           Seg : out std_logic_vector (7 downto 0);
                    led : out std_logic_vector(7 downto 0)
           );
       signal Dig:bit_vector(3 downto 0);
end LiteDancer;

architecture Behavioral of LiteDancer is
begin

      process(clk)
            variable j:integer:=0;
            begin
                  if ( clk'event and clk='1') then
                        if j<100000 then
                        j:=j+1;
                        else
                              j:=0;
                        if i(7)='1' then    -- show 8
                        Dig<="1000";
                        elsif i(6)='1' then  --- show 7
                              Dig<="0111";
                       elsif i(5)='1' then  --- show 6
                              Dig<="0110";
```

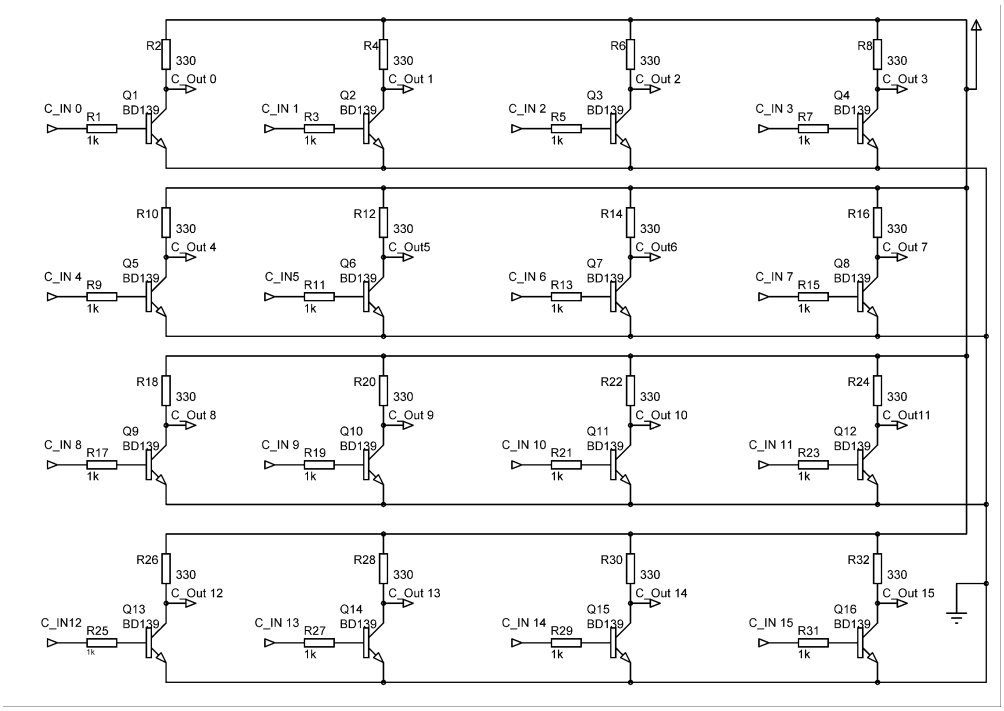
```
                              elsif i(4)='1' then  ---show 5
                                     Dig<="0101";
                                   elsif i(3)='1' then  --- show 4
                                          Dig<="0100";
                                        elsif i(2)='1' then  --show 3
                                               Dig<="0011";
                                             elsif i(1)='1' then --
show 2
                                                      Dig<="0010";
                                                   elsif i(0)='1'
then -- show 1

Dig<="0001";
                                            else  ---show 0

Dig<="0000";
                                                                  end if;
                             LEd<=i;

            end if;
            end if;
   end process;

       with Dig select
       Seg <=     "11000000" when "0000",  -- turn on abcdef on 7seg
to show 0
                        "11111001" when "0001",  -- turn on bc on
7seg to show 1
                        "10100100" when "0010",  -- turn on abged on
7seg to show 2
                        "10110000" when "0011",  -- turn on abgcd on
7seg to show 3
                        "00011001" when "0100",  -- turn on fgbc on
7seg to show 4
                        "10010010" when "0101",  -- turn on afgcd on
7seg to show 5
                        "10000010" when "0110",  -- turn on afedcg on
7seg to show 6
                        "11111000" when "0111",  -- turn on abc on
7seg to show 7
                        "10000000" when "1000",  -- turn on abcdefg
on 7seg to show 8
                        "11000110" when others;  -- show E on other
end Behavioral;
```

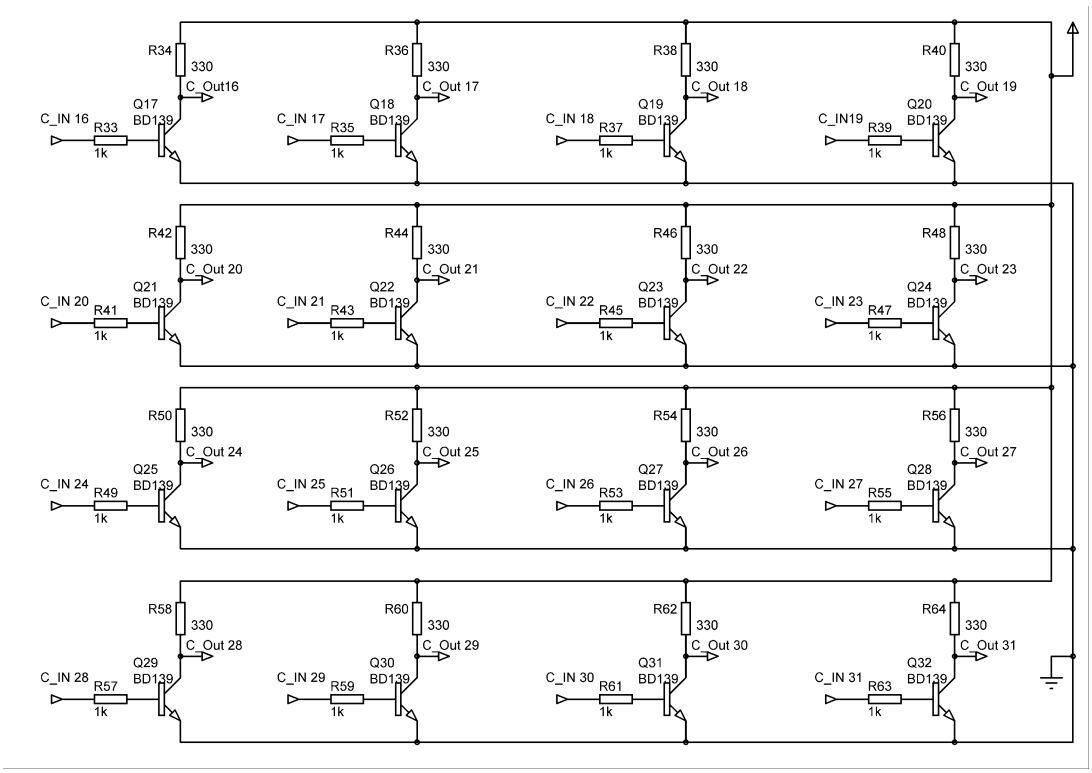
۱۵- تابلو روان

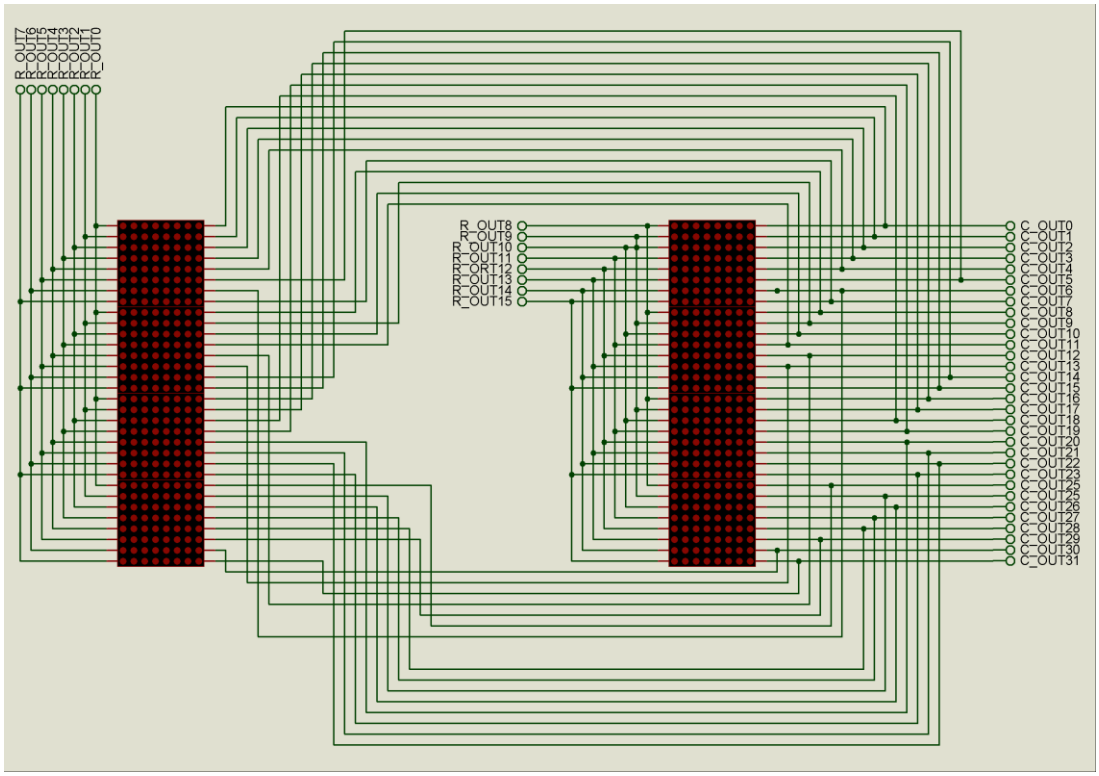
شکل مدار:






R2 330 C_Out 0 Q1 BD139 C_IN 0 R1 1k
R4 330 C_Out 1 Q2 BD139 C_IN 1 R3 1k
R6 330 C_Out 2 Q3 BD139 C_IN 2 R5 1k
R8 330 C_Out 3 Q4 BD139 C_IN 3 R7 1k
R10 330 C_Out 4 Q5 BD139 C_IN 4 R9 1k
R12 330 C_Out5 Q6 BD139 C_IN5 R11 1k
R14 330 C_Out6 Q7 BD139 C_IN 6 R13 1k
R16 330 C_Out 7 Q8 BD139 C_IN 7 R15 1k
R18 330 C_Out 8 Q9 BD139 C_IN 8 R17 1k
R20 330 C_Out 9 Q10 BD139 C_IN 9 R19 1k
R22 330 C_Out 10 Q11 BD139 C_IN 10 R21 1k
R24 330 C_Out11 Q12 BD139 C_IN 11 R23 1k
R26 330 C_Out 12 Q13 BD139 C_IN12 R25 1k
R28 330 C_Out 13 Q14 BD139 C_IN 13 R27 1k
R30 330 C_Out 14 Q15 BD139 C_IN 14 R29 1k
R32 330 C_Out 15 Q16 BD139 C_IN 15 R31 1k


R34 330 C_Out16 Q17 BD139 C_IN 16 R33 1k
R36 330 C_Out 17 Q18 BD139 C_IN 17 R35 1k
R38 330 C_Out 18 Q19 BD139 C_IN 18 R37 1k
R40 330 C_Out 19 Q20 BD139 C_IN19 R39 1k
R42 330 C_Out 20 Q21 BD139 C_IN 20 R41 1k
R44 330 C_Out 21 Q22 BD139 C_IN 21 R43 1k
R46 330 C_Out 22 Q23 BD139 C_IN 22 R45 1k
R48 330 C_Out 23 Q24 BD139 C_IN 23 R47 1k
R50 330 C_Out 24 Q25 BD139 C_IN 24 R49 1k
R52 330 C_Out 25 Q26 BD139 C_IN 25 R51 1k
R54 330 C_Out 26 Q27 BD139 C_IN 26 R53 1k
R56 330 C_Out 27 Q28 BD139 C_IN 27 R55 1k
R58 330 C_Out 28 Q29 BD139 C_IN 28 R57 1k
R60 330 C_Out 29 Q30 BD139 C_IN 29 R59 1k
R62 330 C_Out 30 Q31 BD139 C_IN 30 R61 1k
R64 330 C_Out 31 Q32 BD139 C_IN 31 R63 1k



برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity tablo_k is
port( clk,res:in std_logic;
row:out std_logic_vector(15 downto 0);
column_a:out std_logic_vector(2 downto 0);
e:buffer std_logic_vector(3 downto 0));
end tablo_k;


architecture Behavioral of tablo_k is
type state is (start,delay);
signal current : state;
signal z,w:std_logic_vector(2 downto 0);
signal b,l,k,i,c:integer;
constant row_counter:integer:=32;
```

```
type Memory is array(0 to row_counter-1) of std_logic_vector(0 to
15);
signal t,table:Memory;

begin
process(clk,res)
variable temp_a:std_logic_vector(2 downto 0);
variable Row_Count:std_logic_vector(2 downto 0);  --soton ra ebteda
dar temp mirizim va yeki be an ezafe mikonim
variable counter,n,j: integer;
begin
if(res='1')then
table<=(
 "0000011111000000"
,"0000011111000000"
,"0000011000011000"
,"0000011000011000"
,"0000011111000110"
,"0000011111000110"
,"0000011000011000"
,"0000011000011000"
,"0000011111000000"
,"0000011111000000"
,"0000011000000000"
,"0000011000000000"
,"0000011110000000"
,"0000011111000000"
,"0000011011000000"
,"0000011011000000"
,"0000011111000000"
,"0000011111000000"
,"0000011000000000"
,"0000011000000000"
,"0000011000000000"
,"0000011111111100"
,"0000011111111100"
,"0000000000000000"
,"0111111111111100"
,"0111111111111100"
,"0110000000000000"
,"0110000000000000"
,"0110000000000000"
,"0111111100000000"
,"0111111100000000"
,"0000000000000000" );

t<=table;
counter:=1000;
temp_a:="000";
Row_Count:="000";           --32 column ra mishomarad
e<="1110";   --decoder aval ra faal mi konad
z<="000";         --
i<=0;             --shomarande hafeze
w<="000";
n:=0;
b<=0;
j:=0;
k<=0 ;      --baraye raftan be decoder badi bayad 8 shavad
c<=0;      --
row<="0000000000000000";
```

```
current<=start;
elsif (clk'event and clk='1')then


case current is
--*******************************start*****************************

when start=>
j:=j+1;
if(j=1000)then
j:=0;
table <=  table(1 to 31)& table(0);
end if;
temp_a:=Row_Count;
Row_Count:=Row_Count+1;

if(c<row_counter-1)then
row<=table(i);
i<=i+1;
b<=i;   --baraye didan i dar modelsim
k<=k+1;
c<=c+1;
counter:=1000;
current<=delay;
if(k=8)then
e<=e(2 downto 0)&e(3);
k<=1;
end if;
else

i<=0;
c<=0;
k<=8;
counter:=1000;
current<=delay;

end if;
--
*********************************************delay*****************
**************
when delay=>
counter:=counter-1;
if(counter=0)then
current<=start;
end if;

end case;
column_a<=temp_a;

end if;
end process;
end Behavioral;
```

۱۶- اتصال 8255 (PPI) به FPGA

8255 شامل سه پورت می‌باشد که هرکدام می‌توانند به عنوان ورودی یا خروجی برنامه‌ریزی شوند. این پورت‌ها PA و PB و PC نام دارند و می‌توان به وسیله A0 و A1 آن‌ها را آدرس داد. بدین صورت که اگر A1A0=00 شود، PA انتخاب می‌شود. اگر A1A0=01 شود PB و اگر A1A0=10 شود، PC انتخاب می‌شود.

اگر A1A0=11 شود رجیستر کنترلی (CONTROL REGISTER) انتخاب می‌شود. که این رجیستر وضعیت پورت‌ها را مشخص می‌کند.

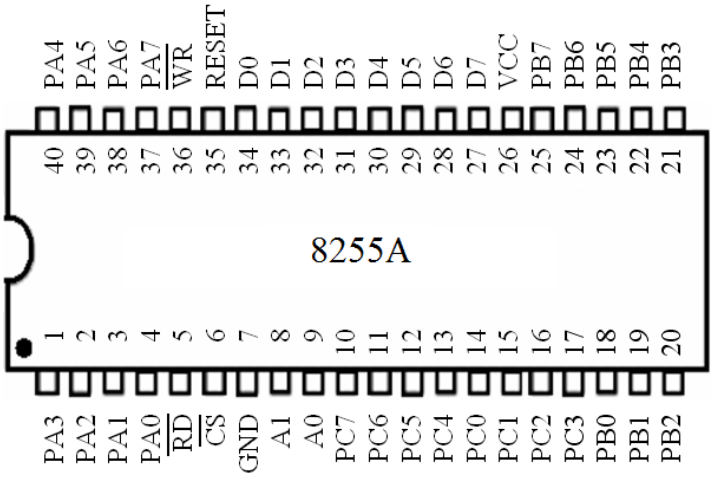
ما در اینجا با یک حالت ساده برنامه‌ریزی (Mode 0) کار می‌کنیم. که در این حالت رجیستر کنترلی به صورت شکل ۱ در می‌آید.

1	0	0	PA ورودی: 1 خروجی: 0	PC7-PC4 ورودی: 1 خروجی: 0	0	PB ورودی: 1 خروجی: 0	PC3-PC0 ورودی: 1 خروجی: 0
MSB							LSB

شکل ۱ بیتهای رجیستر کنترلی 8255

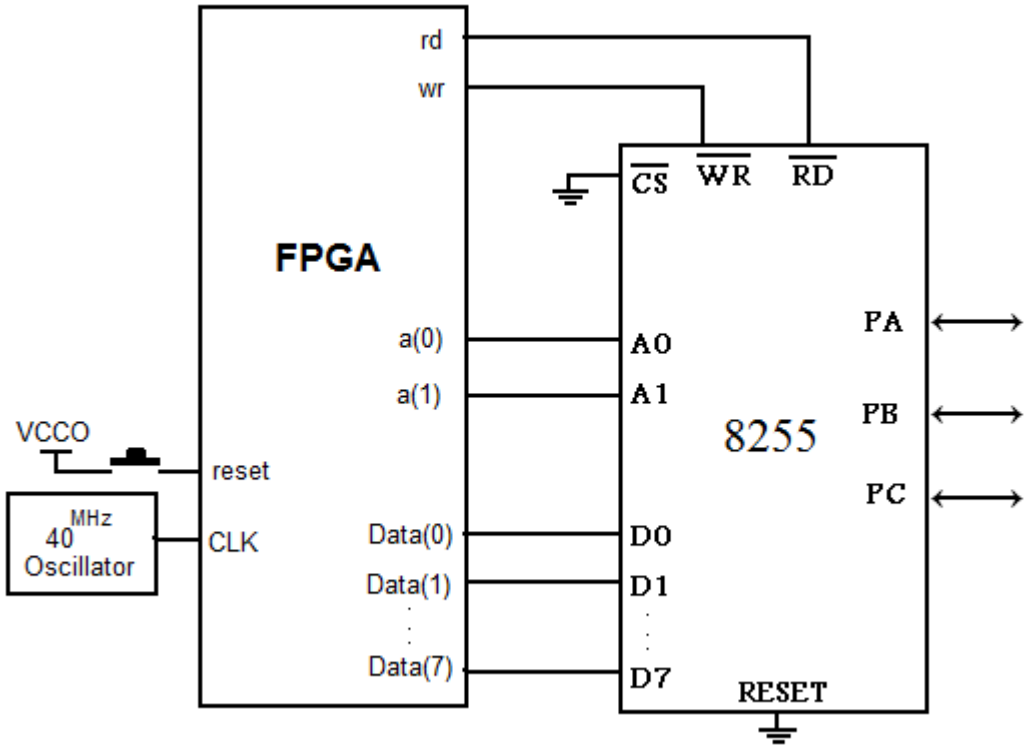
برای نمونه اگر بخواهیم 4 بیت پایینی پورت C به عنوان ورودی و 4 بیت بالایی آن را به عنوان خروجی و پورت B را به عنوان خروجی و پورت A را به عنوان ورودی تعریف کنیم باید مقدار 10010001 را به آدرس A1A0=11 بفرستیم.

شکل ۲ پایه‌های تراشه 8255 را نشان می‌دهد.



شکل ۲ پایه‌های تراشه 8255

خروجی‌های wr و rd از FPGA به ترتیب برای فعال کردن پایه $\overline{WR}$ و $\overline{RD}$ به کار می‌روند. همچنین خط RESET تراشه 8255 را به زمین وصل می‌کنیم تا در هر حالت غیر فعال باشد. شکل ۳ این اتصالات را نشان می‌دهد. مدار شکل ۳ را ببندید.



شکل ۳ مدار اتصال 8255 به FPGA

برنامه زیر پورت A را به عنوان ورودی تعریف مـی‌کنـد و مقـدار خوانـده شـده از آن را در خروجـی پورت‌های B و C قرار می‌دهد.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- synopsys translate_off
library UNISIM;
use UNISIM.Vcomponents.ALL;
-- synopsys translate_on

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ppi is
    Port (  -- Data to 8255
                        data : inout std_logic_vector(7 downto 0);
            -- program mode
                     a : out std_logic_vector(1 downto 0);
            rd : out std_logic;
            wr : out std_logic;
            clk : in std_logic;
            reset : in std_logic);

end ppi;

architecture Behavioral of ppi is

   component IOBUF
      port ( I  : in    std_logic;
             IO : inout std_logic;
             O  : out   std_logic;
             T  : in    std_logic);
   end component;

   constant rd_wr_delay : integer := 80; -- 10 us
   type state is (init, write, read, normal);
   signal current : state;
   signal part : integer;
   signal direction : std_logic;
   signal temp, data_in, data_out : std_logic_vector (7 downto 0);

begin
   process (clk, reset)
      variable count : integer;
   begin
      if reset = '1' then
                  -- reet all datas
         rd <= '1';
         wr <= '1';
         part <= 0;
         current <= init;
         count := 0;
```

```
elsif clk = '1' and clk'event then

   case current is
   when init =>
      -- set program mode
            a <= "11";

      direction <= '0';
      data_out <= "10010000";  -- pa : in pb, pc : out
      count := rd_wr_delay;
      current <= write;
   when normal =>
      -- programming
      if part = 0 then
         a <= "00";
         direction <= '1';
         count := rd_wr_delay;
         current <= read;
         part <= 1;
      elsif part = 1 then
         a <= "01";
         direction <= '0';
         data_out <= temp;
         count := rd_wr_delay;
         current <= write;
         part <= 2;
      elsif part = 2 then
         a <= "10";
         direction <= '0';
         data_out <= temp;
         count := rd_wr_delay;
         current <= write;
         part <= 0;
      end if;
      -- write program byte
   when write =>
      if count /= 0 then
         if count < rd_wr_delay / 2 then
            wr <= '1';
         else
            wr <= '0';
         end if;
            count := count - 1;
      else
         current <= Normal;
      end if;
   when read =>
      if count /= 0 then
         if count < rd_wr_delay / 2 then
            rd <= '1';
         else
            rd <= '0';
            temp <= data_in;
         end if;
         count := count - 1;
      else
         current <= normal;
      end if;
   end case;
end if;
```

```
   end process;
   io_block_8 : for j in 0 to 7 generate
      io_block : iobuf port map (data_out(j),data(j),
data_in(j), direction);
   end generate;
end Behavioral;
```

۱۷- اتصال مبدل دیجیتال به آنالوگ DAC08 به FPGA

در این مبدل (DAC08)، مقدار دیجیتال ورودی (B1 (با ارزشترین بیت) تا B8) به یک جریان خروجی تبدیل می‌شود که با اتصال یک مقاومت به خروجی می‌توان این جریان را تبدیل به ولتاژ نمود. خروجی مذکور از پایه I_{OUT} گرفته می‌شود. همچنین باید یک جریان مرجع به تراشه اعمال کرد. این جریان از طریق پایه‌های $V_{REF(+)}$ و $V_{REF(-)}$ توسط دو مقاومت 12^{K} و ولتاژ مرجع 2.5^{V} (ولتاژ مرجع اصلی برای کار کردن درست تراشه $\pm 15^{V}$ می‌باشد؛ ولی در اینجا برای امتحان تراشه از ولتاژ $\pm 2.5^{V}$ استفاده شده است) به تراشه داده می‌شود. ($I_{REF}=2.5/12=0.2^{mA}$) پس می‌توان جریان I_{REF} را در صورتی که مدار درست کار نمی‌کند، اندازه گرفت تا از درستی کار آن مطمئن شد. رابطه I_{OUT} با I_{REF} از طریق فرمول زیر به دست می‌آید:

$$I_{OUT} = I_{REF}\left(\frac{B1}{2}+\frac{B2}{4}+\frac{B3}{8}+\ldots+\frac{B8}{256}\right)$$

برای حداکثر مقدار دیجیتال ورودی (11111111) جریان خروجی تقریباً برابر 0.2^{mA} می‌شود:

$$I_{OUT} = 0.2^{mA}\left(\frac{1}{2}+\frac{1}{4}+\ldots+\frac{1}{256}\right) \cong 0.2^{mA}$$

پس حداکثر ولتاژ خروجی با مقاومت 4.7^{K} تقریباً برابر 1^{V} (0.98^{V}) و حداقل آن هم 0^{V} می‌باشد.

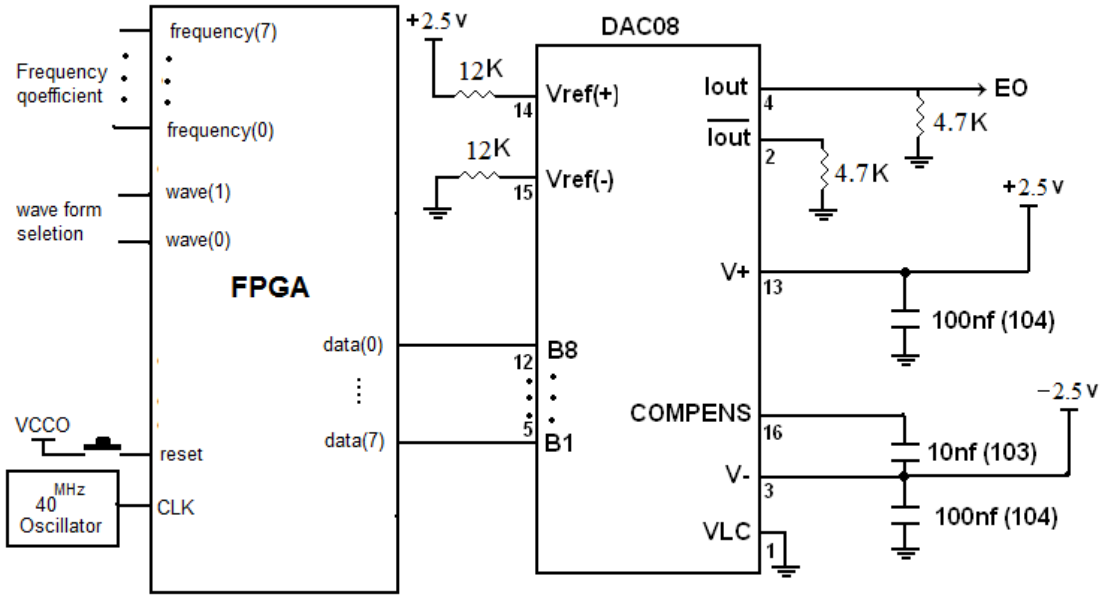
می‌خواهیم یک موج سینوسی ایجاد کنیم. همان طور که گفتیم مقدار ولتاژ خروجی می‌تواند میان 0^{V} تا 1^{V} باشد.پس باید مقدار سینوس زاویه‌هایی را که بین 1- تا 1 اند به 0 تا 1^{V} نگاشت کنیم. برای این کار از فرمول $V_{OUT} = 1/2^{V} + (1/2 \times \text{Sin}\ \theta)$ استفاده می‌کنیم.

برای نمونه برای نقطه $\theta = 0^{o}$، مقدار V_{OUT} برابر 1/2 خواهد شد و برای $\theta = 30^{o}$ مقدار V_{OUT} برابر 1/4 + 1/2 می‌شود.

حداکثر مقدار آنالوگ خروجی (1^{V}) مربوط به مقدار دیجیتال B1-B8=11111111=255 می‌باشد. پس برای $1/2^{V}$، باید مقدار 128 و برای $1/2^{V} + 1/4^{V}$ مقدار 192 محاسبه شود.

همچنین توجه داشته باشید که علاوه بر ولتاژ 5^V، نیاز به دو ولتاژ 2.5^V+ و 2.5^V- هم دارید. برای داشتن این دو ولتاژ، نیاز به دو منبع دارید. ترمینال منفی منبع اول را به ترمینال مثبت منبع دوم وصل کنید و آن‌ها را به زمین مدار وصل نمایید. از ترمینال مثبت منبع اول، 2.5^V+ و از ترمینال منفی منبع دوم، 2.5^V- را بگیرید.

مطابق شکل ۱ مبدل را به FPGA وصل کنید.



شکل ۱ مدار اتصال DAC08 به FPGA

برای این مدار نیازی نیست که مقاومت ها یا خازن‌هایی با همان اندازه که در شکل آمده را به کار بگیرید. برای سادگی می‌توانید از اندازه‌های نزدیک به آن مقادیر استفاده کنید. ولی توجه داشته باشید که در جایی که نیاز به دقت دارید باید از همان مقادیر استفاده کنید.

برنامه زیر برای تولید موج مربعی، دندان اره‌ای، مثلثی و سینوسی طراحی شده است. خروجی مبدل (EO) را به اسیلوسکپ وصل کنید و نتیجه را مشاهده نمایید.

انتخاب نوع موج خروجی به وسیله عددی که به wave می‌دهید، انجام می‌شود:

wave = 00 موج خروجی مربعی

wave = 01 موج خروجی دندان اره‌ای

wave = 10 موج خروجی مثلثی

wave = 11 موج خروجی سینوسی

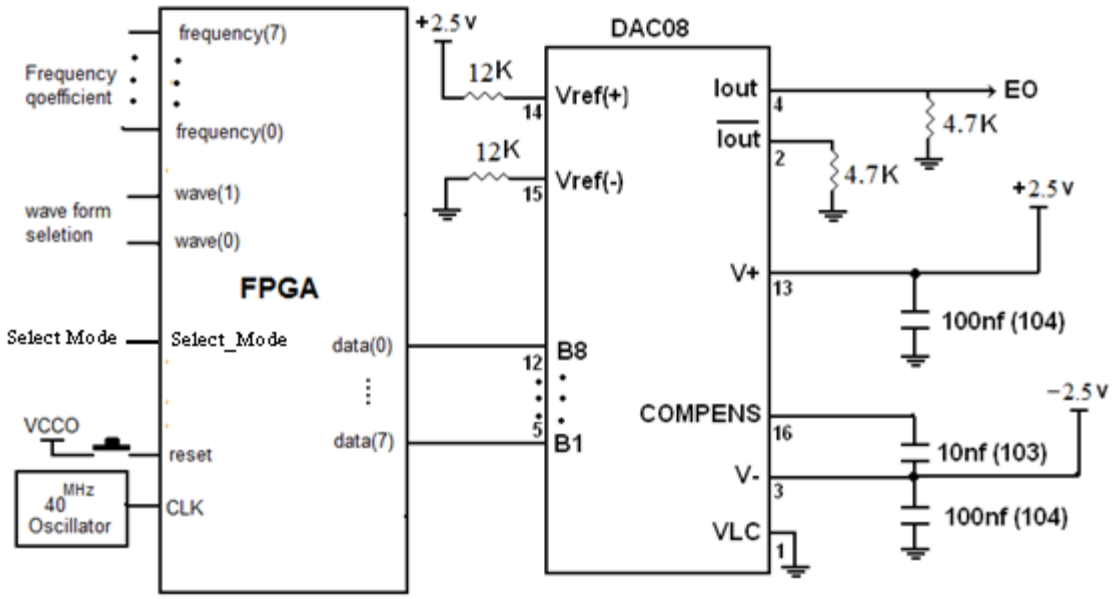
فرکانس را هم با ورودی frequency می‌توان عوض کرد.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

 ----Uncomment the following library declaration if instantiating
 ----any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity dac is
   Port ( data : out std_logic_vector (7 downto 0);
          wave : in std_logic_vector(1 downto 0);
          frequency : in std_logic_vector (7 downto 0);
          clk : in std_logic;
          reset : in std_logic)
end dac;

architecture Behavioral of dac is

   type state is (square, saw, triangle, sine, delay);
   signal current : state;
   signal period : std_logic_vector (15 downto 0);
   signal data_i : std_logic_vector (7 downto 0);
   type two_dim  is array (0 to 23) of std_logic_vector (7 downto 0);
   constant table : two_dim := ("10000000", "10100000", "11000000",
                                "11011001", "11101110", "11111010",
                                "11111111", "11111010", "11101110",
                                "11011001", "11000000", "10100000",
                                "10000000", "01011110", "01000000",
                                "00100101", "00010001", "00000100",
                                "00000000", "00000100", "00010001",
                                "00100101", "01000000", "01011110");
begin
   process (clk, reset)
      variable i, j : integer;
      variable down : std_logic;
      variable count : std_logic_vector (15 downto 0);
   begin
      if reset = '1' then
         current <= square;
         data_i <= "00000000";
         i := 0;
         j := 0;
         down := '0';
      elsif clk = '1' and clk'event then
         case current is
         when square  =>
            if i = 0 then
               data_i <= "00000000";
               i := 1;
```

```
            else
               data_i <= "11111111";
               i := 0;
            end if;
            count := period;
            current <= delay;
         when saw  =>
            data_i <= data_i + 1;
            count := period;
            current <= delay;
         when triangle  =>

            if down = '0' then
               data_i <= data_i + 1;
               if data_i = "11111110" then
                  down := '1';
               end if;
            else
               data_i <= data_i - 1;
               if data_i = "00000001" then
                  down := '0';
               end if;
            end if;
            count := period;
            current <= delay;
         when sine  =>
            if j < 24 then
               data_i <= table(j);
               j := j + 1;
            else
               j := 0;
            end if;
            count := period;
            current <= delay;
         when delay  =>
            count := count - 1 ;
            if count = 0 then
               case wave is
               when "00"
                  current <= square;
               when "01" =>
                  current <= saw;
               when "10" =>
                  current <= triangle;
               when others  =>
                  current <= sine;
               end case;
               period <= frequency & "00000000";
            end if;
         end case;
      end if;
   end process;
   data <= data_i;

end Behavioral;
```

مدار دیگر با حالت اتوماتیک تغییر فرکانس:

مطابق شکل ۲ مبدل را به FPGA وصل کنید.



شکل ۲ مدار اتصال DAC08 به FPGA

در این مدار فرکانس را می توان با دو حالت دستی و اتوماتیک عوض کرد.

برای انتخاب حالت دستی باید مقدار '0' را به پایه ی select_mode بدهید و فرکانس را هم با ورودی frequency می‌توان عوض کرد. برای انتخاب حالت اتوماتیک باید مقدار '1' (Vcc) را به پایه ی select_mode بدهید.

برنامه مربوط به تبدیل دیجیتال به آنالوگ:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
```

```
entity dac is
   Port (data : out std_logic_vector (7 downto 0);
         wave : in std_logic_vector(1 downto 0);
            frequency : in std_logic_vector (7 downto 0);
         clk : in std_logic;
         reset : in std_logic;
                  select_mode: in std_logic);
end dac;

architecture Behavioral of dac is

   type state is (square, saw, triangle, sine, delay);
   signal current : state;
   signal period : std_logic_vector (15 downto 0);
   signal data_i : std_logic_vector (7 downto 0);
   type two_dim  is array (0 to 23) of std_logic_vector (7 downto 0);
   constant table : two_dim := ("10000000", "10100000", "11000000",
                                "11011001", "11101110", "11111010",
                                "11111111", "11111010", "11101110",
                                "11011001", "11000000", "10100000",
                                "10000000", "01011110", "01000000",
                                "00100101", "00010001", "00000100",
                                "00000000", "00000100", "00010001",
                                "00100101", "01000000", "01011110");
begin
   process (clk, reset)
      variable i, j,k,l : integer;
            variable frequency_auto : std_logic_vector (7 downto 0);
            variable down : std_logic;
            variable count : std_logic_vector (15 downto 0);
   begin
            if reset = '1' then
            current <= square;
            data_i <= "00000000";
            i := 0;
            j := 0;
                  l := 0;
                  k := 0;
            down := '0';
                  frequency_auto:="00000000";
            elsif clk = '1' and clk'event then
            case current is
            when square  =>                        --  نمایش موج مربعی
                  if i = 0 then                    --  نمایش "صفر" موج مربعی
                        data_i <= "00000000";
                        i := 1;
                        else                       --  نمایش "یک" موج مربعی
                        data_i <= "11111111";
                        i := 0;
                        end if;
                        count := period;       -- انتساب مقدار فرکانس به شمارنده
                  current <= delay;
            when saw  =>                           --  نمایش موج دندان اره ای
                  data_i <= data_i + 1;
                   -- هر گاه متغییر به ۲۵۵ برسد و یک واحد به آن اضافه شود مقدار متغییر برابر صفر خواهد شد
                        count := period;
                        current <= delay;
            when triangle  =>                --  نمایش موج مثلثی
```

```
            if down = '0' then
            data_i <= data_i + 1;
            if data_i = "11111110" then
                              --
                  down := '1';
            end if;
            else
            data_i <= data_i - 1;
            if data_i = "00000001" then
                              --
                  down := '0';
            end if;
      end if;
            count := period;
            current <= delay;
when sine  =>                    --
            if j < 24 then
            data_i <= table(j);    --
            j := j + 1;
            else
            j := 0;
            end if;
            count := period;
            current <= delay;
when delay  =>  --
            count := count - 1;
            if count = 0 then
            case wave is           --
            when "00" =>
                  current <= square;
            when "01" =>
            current <= saw;
            when "10" =>
            current <= triangle;
            when others  =>
            current <= sine;
            end case;
      if select_mode = '0' then     --
      period <= frequency & "00000000";--
      else                    --
            l := l + 1;
            if l = 50 then  --
                  l := 0;
                  if k = 0 then --
                  period <= frequency_auto & "00000000";
                  frequency_auto :=frequency_auto +1;
      if frequency_auto ="11111111" then
                        --
                        k := 1;
                  end if;
            else                          --
                  period <= frequency_auto & "00000000";
                  frequency_auto :=frequency_auto -1;
                  if frequency_auto ="00000000" then
                        k := 0;
                  end if;
```

تذکر: متغیر از نوع سیگنال می باشد

تذکر: متغیر از نوع سیگنال می باشد

نمایش موج سینوسی

فراخوانی از جدول بالا

مشخص کردن نوع موج، حالت تغییر فرکانس و مقدار فرکانس

مشخص کردن نوع موج از ورودی

فرکانس بصورت دستی مقدار میگیرد

بزرگ کردن مقدار فرکانس

فرکانس بصورت اتوماتیک مقدار میگیرد

تاخیر در عوض کردن مقدار فرکانس

اضافه کردن تاخیر

چک کردن فرکانس برای رسیدن به ۲۵۵

کاهش تاخیر

```
                                end if;
                            end if;
                end if;
                            end if;
                            end case;
          end if;
      end process;
      data <= data_i;

end Behavioral;
```

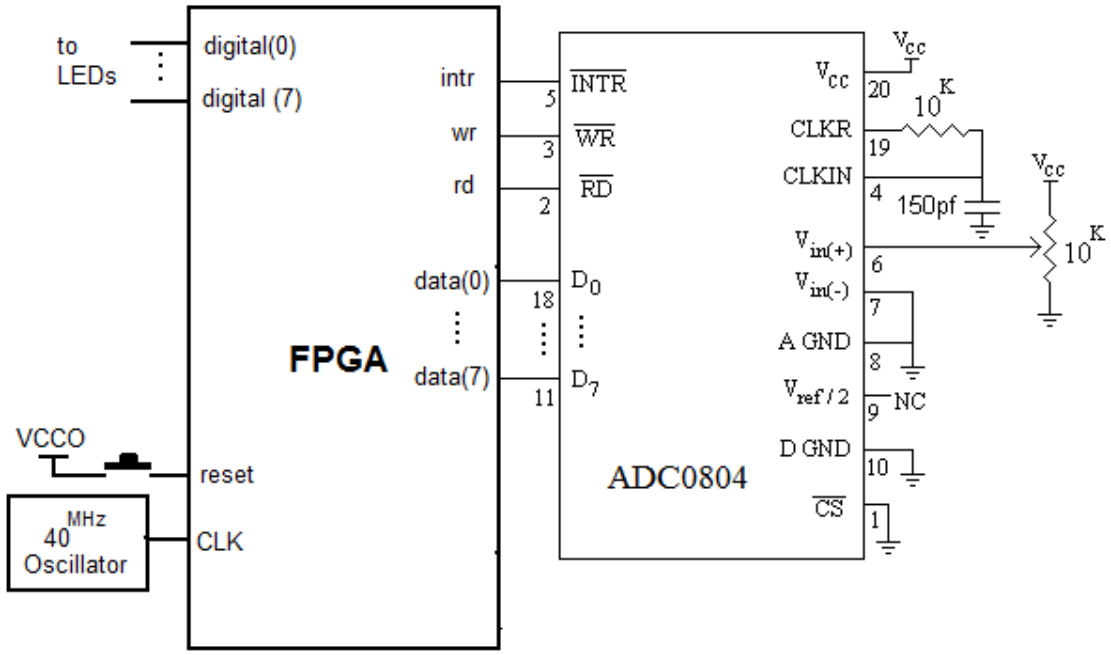
۱۸- اتصال مبدل آنالوگ به دیجیتال ADC0804 به FPGA

تراشه ADC0804 برای تبدیل سیگنال آنالوگ (که به پایه‌های $V_{in(+)}$ و $V_{in(-)}$ آن اعمال می‌شود) به سیگنال دیجیتال (که در پایه‌های D_0 تا D_7 آماده می‌شوند) به کار می‌رود.

برای انتخاب تراشه $\overline{CS}$ باید برابر '0' قرار داده شود. شروع کار توسط خط $\overline{WR}$ به تراشه اعلام می‌شود و پایان کار توسط پایه $\overline{INTR}$ از طرف تراشه مشخص می‌شود. پس از پایان کار، مقدار دیجیتال را با فعال کردن پایه $\overline{RD}$ می‌توانیم از تراشه بخوانیم. عملیات زمان‌بندی در شکل ۱ نشان داده شده است.

شکل ۱ زمان‌بندی سیگنالهای مبدل ADC0804

می‌خواهیم مطابق شکل ۲ توسط FPGA این تراشه را کنترل کنیم. در مدار شکل ۲، ورودی به جای این که به صورت اختلاف ولتاژ بین Vin(+) و Vin(-) (Vin(+) و Vin(-)) اعمال شود، بطور مستقیم به Vin(+) وصل می‌شود. در نتیجه باید Vin(-) را به زمین آنالوگ (A GND) وصل نمود.



شکل ۲ مدار اتصال ADC0804 به FPGA

در این شکل همچنین مشاهده می‌کنید که زمین آنالوگ با زمین دیجیتال (D GND) یکی در نظر گرفته شده است. مدار شکل ۲ را وصل کنید و با یک مالتی‌متر ولتاژ Vin(+) را بخوانید. در خروجی digital از FPGA، باید مقدار تبدیل شده آن به دیجیتال حاضر باشد. برای نمونه برای حداکثر ولتاژ ورودی (5^V) در پورت ۳، مقدار 11111111 را باید داشته باشیم و برای حداقل (0^V) مقدار 00000000 و برای 2.5^V، مقدار حدود 01111111 (127) را باید داشته باشیم. به طور کلی برای (x^V)، مقدار $\frac{x}{5} \times 255$ را در پورت ۳ باید بخوانیم. برای ولتاژ ورودی برابر0، 1، 2، 2.5، 3، 4 و 5 ولت مدار را امتحان کنید.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity adc is
```

```
   Port ( wr : out std_logic;
          rd : out std_logic;
          data : in std_logic_vector(7 downto 0);
          intr : in std_logic;
          digital : out std_logic_vector(7 downto 0);
          clk : in std_logic;
          reset : in std_logic);
end adc;

architecture Behavioral of adc is

   constant delay_between_read : integer := 1000; -- 25 us
   type state is (start, read, wait_for_intr, delay);
   signal current : state;

begin

   process (clk, reset)
      variable count : integer;
   begin
      if reset = '1' then
         rd <= '1';
         wr <= '0';
         current <= start;
         count := 0;
      elsif clk = '1' and clk'event then
         case current is
         when start =>          -- اعلام شروع کار به تراشه
            wr <= '1';
            current <= wait_for_intr;
         when wait_for_intr =>  -- انتظار برای دریافت اعلام پایان کار توسط تراشه
            if intr = '0' then
               current <= read;
            end if;
         when read =>           -- خواندن مقدار دیجیتال از تراشه
            rd <= '0';
            count := delay_between_read;
            current <= delay;
         when delay =>        -- انتظار برای انتقال داده گرفته شده از تراشه و تنظیم لبه ها
            if count /= 0 then
               if count = delay_between_read / 2 then
                  digital <= data;
                  wr <= '0';
               end if;
               count := count - 1;
            else
               rd <= '1';
               current <= start;
            end if;
         end case;
      end if;
   end process;
end Behavioral;
```

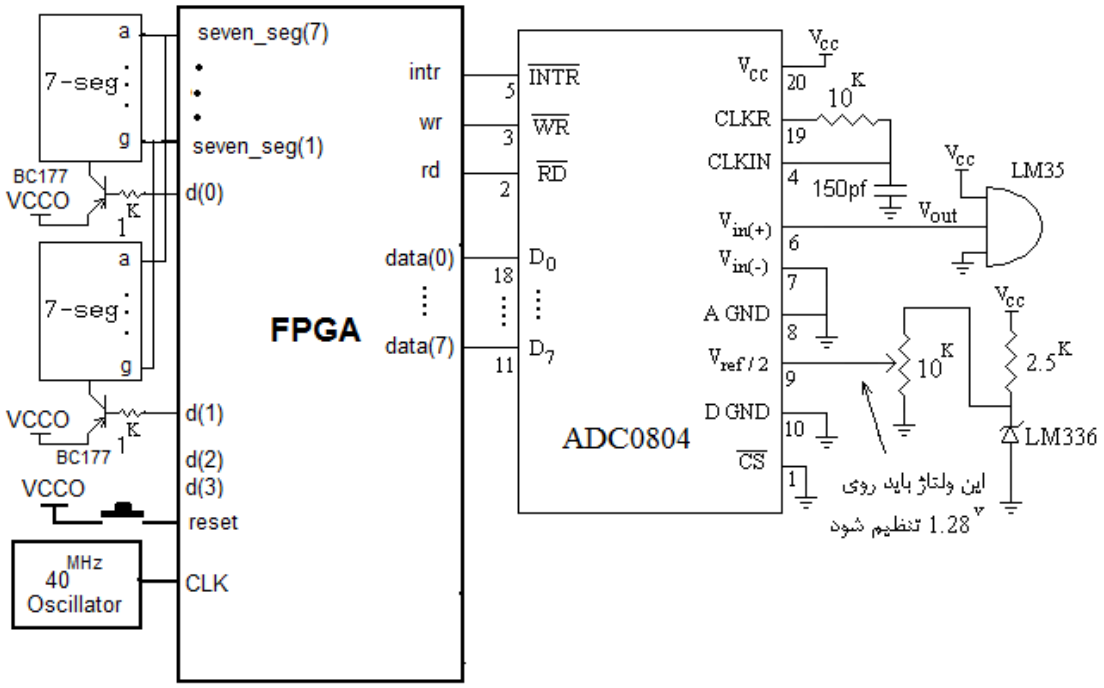
۱۹- اندازه‌گیری دما با مبدل آنالوگ به دیجیتال

می‌خواهیم یک سنسور حرارتی (LM35) را به عنوان ورودی آنالوگ به ADC0804 وصل کنیم. چون حداکثر مقدار V_{out} در LM35 برابر حاصل ضرب $2^8=256$ پله در 10^{mv} (اختلاف پتانسیل بین دو پله) است، باید ولتاژ مرجع (V_{ref}) برابر 2560^{mv} (2.56^{v}) قرار دهیم. برای این کار باید به پایه $V_{ref/2}$، نصف این ولتاژ یعنی 1.28^{v} را اعمال کنیم. این کار را با استفاده از یک پتانسیومتر می‌توانیم انجام دهیم؛ منتها برای دقت بیش‌تر، از یک ولتاژ ثابت استفاده می‌کنیم که توسط تراشه LM336 ایجاد می‌شود. (این تراشه در دو نوع 2.5^{V} و 5^{V} موجود است).

مدار شکل ۱ را ببندید. در این شکل خروجی seven_seg را به دو 7-segment وصل کرده‌ایم. مقاومتهای 7-seg ها را فراموش نکنید. خروجی‌های d(0) و d(1) برای فعال کردن یک 7-segment در هر لحظه به کار می‌رود. برنامه هم باید طوری تغییر داده شود که به جای خروجی باینری ۸ بیتی، دو خروجی BCD داشته باشید. یعنی از برنامه دوم که در انتها آورده شده، استفاده کنید.

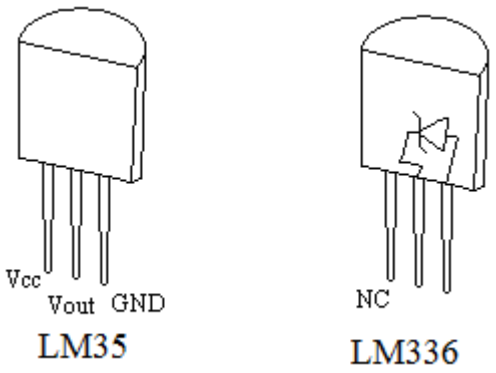
با یک هویه داغ که نزدیک سنسور LM35 نگه می‌دارید دما را تغییر دهید و در همان حال توسط یک دماسنج معمولی دمای آن ناحیه را بخوانید. برای چند مقدار دما، مقداری که توسط دماسنج می‌خوانید را با مقدار روی 7-seg ها مقایسه کنید.

توجه داشته باشید که مقدار $V_{ref/2}$ را باید با پیچاندن پتانسیومتر در حد 1.28^{v} قرار دهید.



شکل ۱ مدار اندازه‌گیری دما با FPGA

پایه‌های LM35 و LM336 در شکل ۲ نشان داده شده‌اند.



شکل ۲ پایه‌های تراشه‌های LM35 و LM336

برنامه‌ی دماسنج:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity temperature is
   Port ( wr : out std_logic;
          rd : out std_logic;
          data : in std_logic_vector(7 downto 0);
          intr : in std_logic;
          seven_seg : out std_logic_vector(7 downto 0);
          d : out std_logic_vector(3 downto 0);
          clk : in std_logic;
          reset : in std_logic);
end temperature;

architecture Behavioral of temperature is

   constant delay_between_read : integer := 10000; -- 25 us
   type state is (start, read, wait_for_intr, delay, convert);
   signal current : state;
   signal digital_i, digital_im : std_logic_vector(7 downto 0);

   signal digital_1, digital_m_1, digit :
          std_logic_vector(3 downto 0);
   signal end_flag : std_logic;
begin

   process (clk, reset)
      variable count : integer;
   begin
      if reset = '1' then
         rd <= '1';
         wr <= '0';
         current <= start;
         count := 0;
         digital_im <= "00000000";
         digital_i <= "00000000";
         end_flag <= '0';
      elsif clk = '1' and clk'event then
         case current is
         when start =>
            wr <= '1';
            current <= wait_for_intr;
         when wait_for_intr =>
            if intr = '0' then
               current <= read;
            end if;
         when read =>
            rd <= '0';
            count := delay_between_read;
            current <= delay;
         when delay =>
            if count /= 0 then
               if count = delay_between_read / 2 then
                  digital_i <= data;
                  wr <= '0';
               end if;
               count := count - 1;
```

```
         else
            rd <= '1';
            current <= convert;

            digital_im <= "00000000";
         end if;
      when convert =>
         if end_flag = '0' then
            digital_i <= digital_i + "11110110"; -- x - 10
            digital_im <=  digital_im + 1;
            if digital_i(7) = '1' then
               end_flag <= '1';
               digital_i <= digital_i + "00001010"; -- x + 10
               digital_im <= digital_im - 1;
            end if;
         else
            current <= start;
            end_flag <= '0';
            digital_1 <= digital_i(3 downto 0);
            digital_m_1 <= digital_im(3 downto 0);
         end if;
      end case;
   end if;
end process;

process (clk, reset)
   variable count : integer;
begin
   if reset = '1' then
      count := 0;
      digit <= "0000";
      d <= "0000";
   elsif clk = '1' and clk'event then
      if count < 100 then
         digit <= digital_1;
         count := count + 1;
         d <= "1101";
      elsif count < 200 then
         digit <= digital_m_1;
         count := count + 1;
         d <= "1110";
      else
         count := 0;
      end if;
   end if;
end process;

seven_seg <= "00000010" when digit = "0000" else
             "10011110" when digit = "0001" else
             "00100100" when digit = "0010" else
             "00001100" when digit = "0011" else
             "10011000" when digit = "0100" else
             "01001000" when digit = "0101" else
             "01000000" when digit = "0110" else
             "00011110" when digit = "0111" else
             "00000000" when digit = "1000" else
             "00001000" when digit = "1001" else
             "00010000" when digit = "1010" else
             "11000000" when digit = "1011" else
             "01100010" when digit = "1100" else
```

```
                "10000100" when digit = "1101" else
                "01100000" when digit = "1110" else
                "01110000" when digit = "1111";

end Behavioral;
```

برنامه‌ی دیگری برای دماسنج (با یک پروسس):

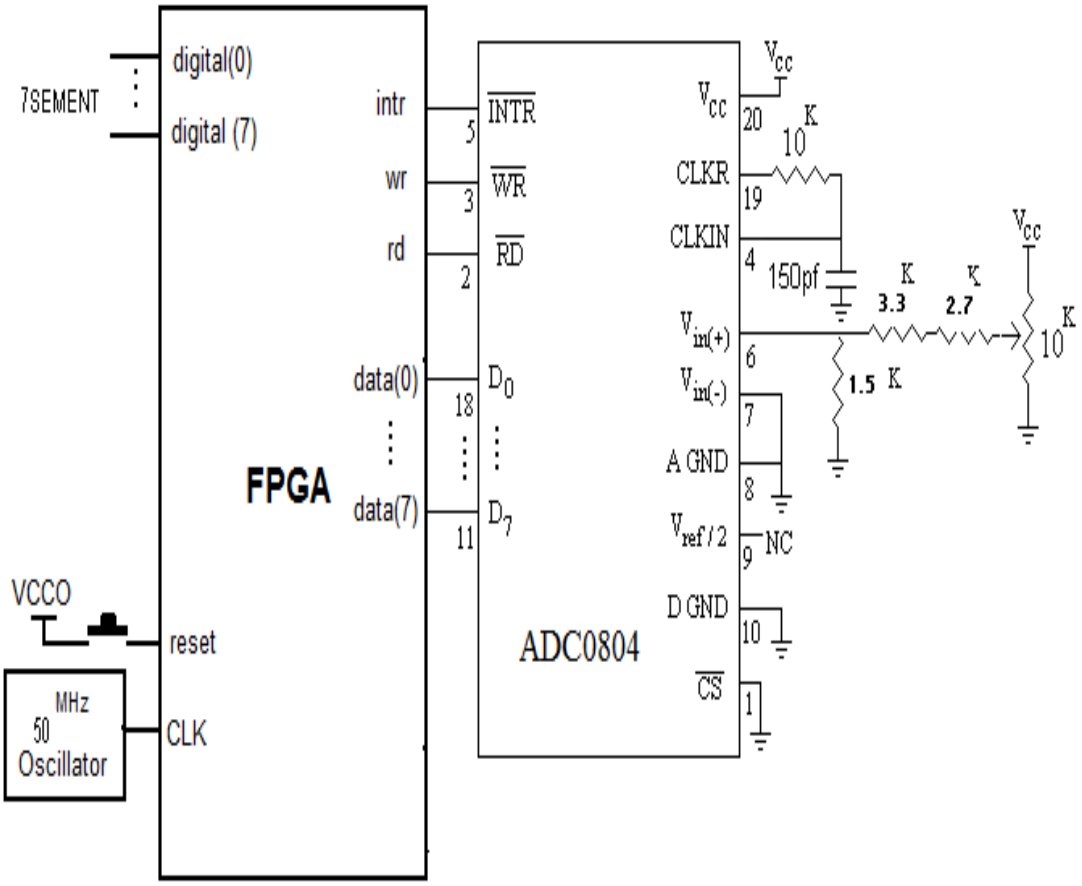
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity temperature is
   Port ( wr : out std_logic;
          rd : out std_logic;
          data : in std_logic_vector(7 downto 0);
          intr : in std_logic;
          seven_seg : out std_logic_vector(7 downto 0);
          d : out std_logic_vector(1 downto 0);
          clk : in std_logic;
          reset : in std_logic);
end temperature;

architecture Behavioral of temperature is

   constant delay_between_read : integer := 10000; -- 25 us
   type state is (start, read, wait_for_intr, delay, convert);
   signal current : state;
   signal digital_i, digital_im : std_logic_vector(7 downto 0);
   signal  digit :  std_logic_vector(3 downto 0);
   signal end_flag : std_logic;

begin

   process (clk, reset)
      variable count : integer;
            variable count1 : integer;
   begin
       if reset = '1' then
         rd <= '1';
         wr <= '0';
         current <= start;
         count := 0;
         digital_im <= "00000000";
         digital_i <= "00000000";
         end_flag <= '0';
        count1 := 0;
         digit <= "0000";
         d <= "00";
      elsif clk = '1' and clk'event then
         case current is
         when start =>              -- اعلام شروع کار به تراشه
                 wr <= '1';
                 current <= wait_for_intr;
         when wait_for_intr =>     -- انتظار برای دریافت اعلام پایان کار توسط تراشه
            if intr = '0' then
```

```
                current <= read;
            end if;
        when read =>            -- خواندن مقدار دیجیتال از تراشه
            rd <= '0';
            count := delay_between_read;
            current <= delay;
        when delay =>      -- انتظار برای انتقال داده گرفته شده از تراشه و تنظیم لبه ها
            if count /= 0 then
                if count = delay_between_read / 2 then
                    digital_i <= data;
                    wr <= '0';
                end if;
                count := count - 1;
            else
                rd <= '1';
                current <= convert;
                digital_im <= "00000000";
            end if;
        when convert =>                    -- تبدیل باینری به دسیمال
            if end_flag = '0' then          -- در صورت منفی نبودن داده
                digital_i <= digital_i + "11110110"; -- x - 10
                digital_im <=  digital_im + 1;
                                  --با هر بار کم کردن از مقدار داده یک واحد به دهگان اضافه می شود
                if digital_i(7) = '1' then
                                -- منفی شدن داده را چک می کند
                    end_flag <= '1';
                    digital_i <= digital_i + "00001010"; -- x + 10
                    digital_im <= digital_im - 1;
                                -- بدلیل اینکه یک واحد بیشتر به رقم دهگان اضافه شده یک واحد کم می کنیم
                end if;
            else           -- در صورت منفی شدن داده کار تبدیل تمام شده و به حالت شروع می رود
                current <= start;
                end_flag <= '0';
            end if;
        end case;
            if count1 < 100 then            -- بازه نمایش رقم یکان
            digit <= digital_i(3 downto 0);
            count1 := count1 + 1;
            d <= "01";
        elsif count1 < 20000 then             --  بازه نمایش رقم دهگان
            digit <= digital_im(3 downto 0);
            count1 := count + 1;   -- not count1:=count1 + 1;
            d <= "10";
        else
            count1 := 0;
        end if;
      end if;
  end process;

--نمایش در سون سگمنت
  seven_seg <= "00000010" when digit = "0000" else
               "10011110" when digit = "0001" else
               "00100100" when digit = "0010" else
               "00001100" when digit = "0011" else
               "10011000" when digit = "0100" else
               "01001000" when digit = "0101" else
               "01000000" when digit = "0110" else
```

```
                "00011110" when digit = "0111" else
                "00000000" when digit = "1000" else
                "00001000" when digit = "1001" else
                "00010000" when digit = "1010" else
                "11000000" when digit = "1011" else
                "01100010" when digit = "1100" else
                "10000100" when digit = "1101" else
                "01100000" when digit = "1110" else
                "01110000" when digit = "1111";

end Behavioral;
```

۲۰- ولت‌متر

شماتیک مدار:

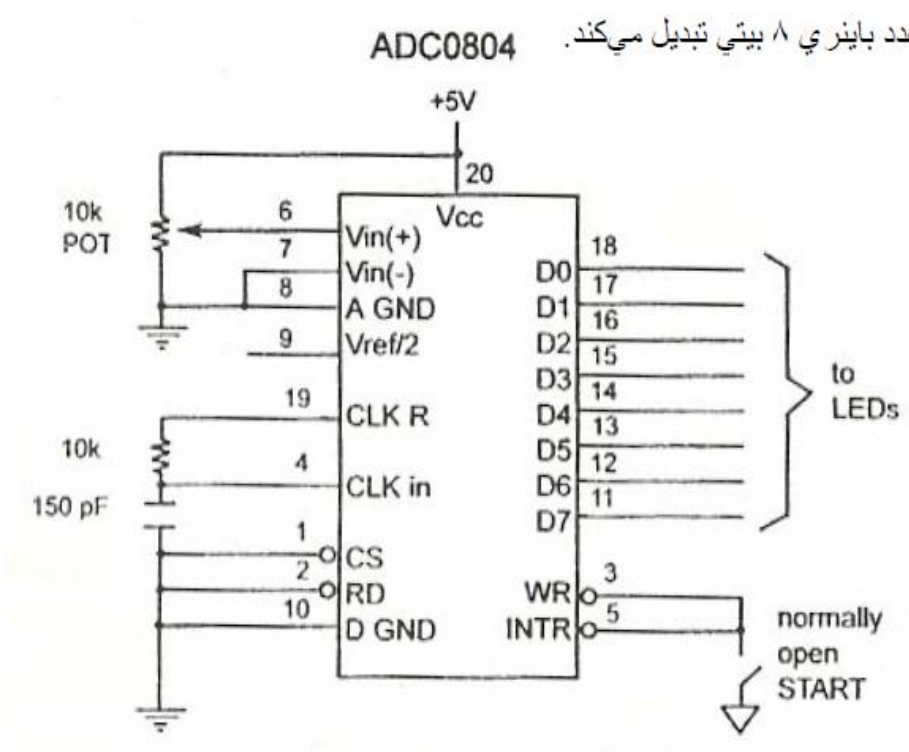


توضیحات در مورد مدار:

هدف ، تشخیص مقدار ولتاژ ورودی با استفاده از ADC0804 و نمایش آن توسط برد FPGA می باشد

در این پروژه ما از مبدل آنالوگ به دیجیتال ADC0804 جهت محاسبه ی مقدار ولتاژ ورودی و تبدیل آن به مقدار دیجیتال و نمایش آن در توسط برد FPGA استفاده کردیم.

متغير هاي اطراف ما يا بصورت مقادير بهم پيوسته است و يا بصورت مقادير گسسته كه اصطلاحاً آنها را متغير هاي آنالوگ و ديجيتال مي‌گويند. البته تعداد بسيار زيادي از متغير ها در زندگي روزمره ما از نوع آنالوگ هستند و در عين حال، وقتي كه اين متغير ها مي‌خواهند توسط سيستمهاي ديجيتالي و يا كامپيوتري مورد پردازش قرار گيرند، بايستي به فرم ديجيتال و عدد درآيند. از اينرو عناصر و قطعاتي وجود دارند كه كميتهاي الكتريكي پيوسته نظير ولتاژ و جريان را به اعداد باينري يا BCD تبديل مي‌كنند.انجام اين عمل به روشهاي گوناگوني انجام مي‌پذيرد و دو روش بسيار مشهور يكي روش انتگرال‌گيري و روش دوم روش تقريبهاي متوالي است و البته هر يك مزايا و معايب خاص خود را دارند. (مزايا و معايب هر كدام چيست؟) در اين آزمايش با مبدلي آشنا مي‌شويد كه ولتاژ الكتريكي را با روش تقريبهاي متوالي به يك عدد باينري ۸ بيتي تبديل مي‌كند.



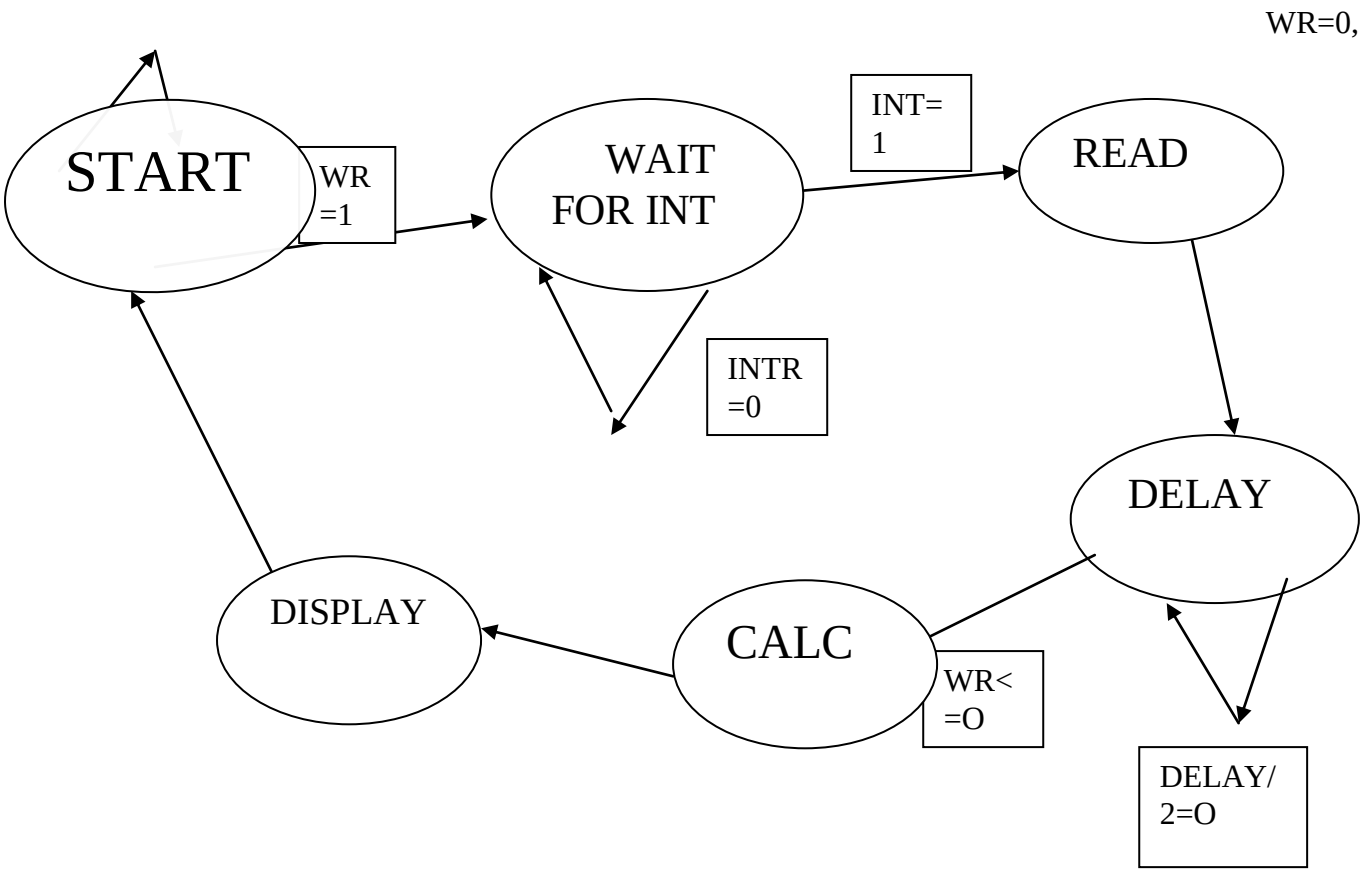
ICهاي مبدل A/D عموماً داراي يك ورودي براي ورود ولتاژ و تعدادي پايه خروجي عدد حاصل از تبديل هستند. CLK in یک پایه ورودی است که به یک منبع ساعت خارجی وصل می شود تا زمانبندی تراشه بصورت خارجی انجام شود. اما از آنجایی که804 داراي يك مولد پالس داخلي نيز می باشد برای استفاده از مولد پالس داخلی ADC804 ، پایه های CLK R,CLKin به يك خازن و مقاومت مطابق شكل وصل می شوند. در این صورت فرکانس پالس ساعت تراشه از روی فرمول(F=1/(1.1RC محاسبه مي شود. چنانچه R=10K و C=150Pf باشد فرکانس تراشه 606KHz و زمان تبدیل برابر 110 μ S خواهد بود. پایه INTR یک پایه خروجی و فعال صفر می باشد. این پایه در حالت عادی یک بوده و وقتی عمل تبدیل انجام می شود در سطح صفر قرار می گیرد. پس از اینکه INTR صفر می شود پایه CS مساوی صفر قرار داده شده و یک پالس یک به صفر روی پایه RD اعمال می شود تا داده تبدیل شده دریافت گردد. Vin(+) و Vin(-) ورودی های آنالوگ تفاضلی می باشند که در آن Vin=Vin(+)-Vin(-) برقرار است. اغلب پایه Vin(-) به زمین وصل می شود و پایه Vin(+) به ولتاژورودی آنالوگ جهت تبدیل به دیجیتال وصل می شود.Vcc به منبع تغذیه 5V+ وصل می شود. این پایه در صورتیکه ورودی Vref/2 (پایه 9 تراشه) باز باشد بعنوان ولتاژ مرجع بکار می رود. Vref/2 یک ولتاژ ورودی است که ولتاژ مرجع تراشه را تامین می کند. اگر این پایه باز باشد(بدون اتصال) ، ولتاژ آنالوگ ورودی در محدوده صفر تا 5V+ است(برابر با پایه Vcc). اما کاربردهای زیادی وجود دارند که در آنها ولتاژ آنالوگ ورودی نیاز به محدوده ای غیر از صفر تا5V+ دارد.بعنوان مثال اگر محدوده ولتاژ آنالوگ ورودی مورد نیاز بین صفر تا 4V+ باشد پایه Vref/2 به ولتاژ 2V+ وصل می شود.

جدول زیر محدوده Vin به ازای مقادیر مختلف Vref/2 را نشان می دهد.

Vref/2(V)	Vin(V)
بی اتصال	0 تا5
2.0	0 تا4
1.5	0 تا3
1.28	0 تا2.56
1.0	0 تا2
0.5	0 تا1

D0 تا D7 (D7 با ارزشترین بیت و D0 کم ارزشترین بیت) پایه های خروجی داده دیجیتال هستند. این پایه ها از بافرهای سه حالته تشکیل شده اند و داده تبدیل شده وقتی در دسترس است که پایه CS=0 و RD نیز در سطح صفر قرار گیرد. برای محاسبه ولتاژ خروجی از فرمول (Vin(+)/Vref)*255 و نهایتاً تبدیل آن به فرم باینری استفاده می شود. زمین آنالوگ و زمین دیجیتال ورودی زمین سیگنالهای آنالوگ و دیجیتال را فراهم می کنند. زمین آنالوگ به زمین سیگنال آنالوگ ورودی Vin و زمین دیجیتال به زمین پایه Vcc وصل می شود. دلیل داشتن دو زمین جدا برای سیگنالهای آنالوگ و دیجیتال جدا سازی سیگنال آنالوگ ورودی از ولتاژ های گذرایی است که در اثر تغییر وضعیت پایه های خروجیD0 تا D7 تولید می شوند و این جدا سازی دقت داده خروجی دیجیتال را تضمین می کند. شایان ذکر است که جدا سازی زمین های آنالوگ و دیجیتال در مدارات بسیار حساس صورت می پذیرد و در مدارات معمولی می توان این دو زمین را به هم وصل کرد.

STATE DIAGRAM:



نحوه عملکرد مدار:

داده ها از طریق پایه های ۱۱-۱۸ مبدل به صورت آنالوگ از طریق پتانسیومتر تعبیه شـده در مـدار وارد شده تبدیل به مقدار دیجیتال متناظر شده و روی سون سگمنت های تعبیه شده روی FPGA نمایش داده می شود.

پایه های READ,WRITE,INT را نیزمطابق شماتیک متصل می کنیم.

برای تامین برق مدار نیز از منبع تغذیه جداگانه استفاده می کنیم.

کد:

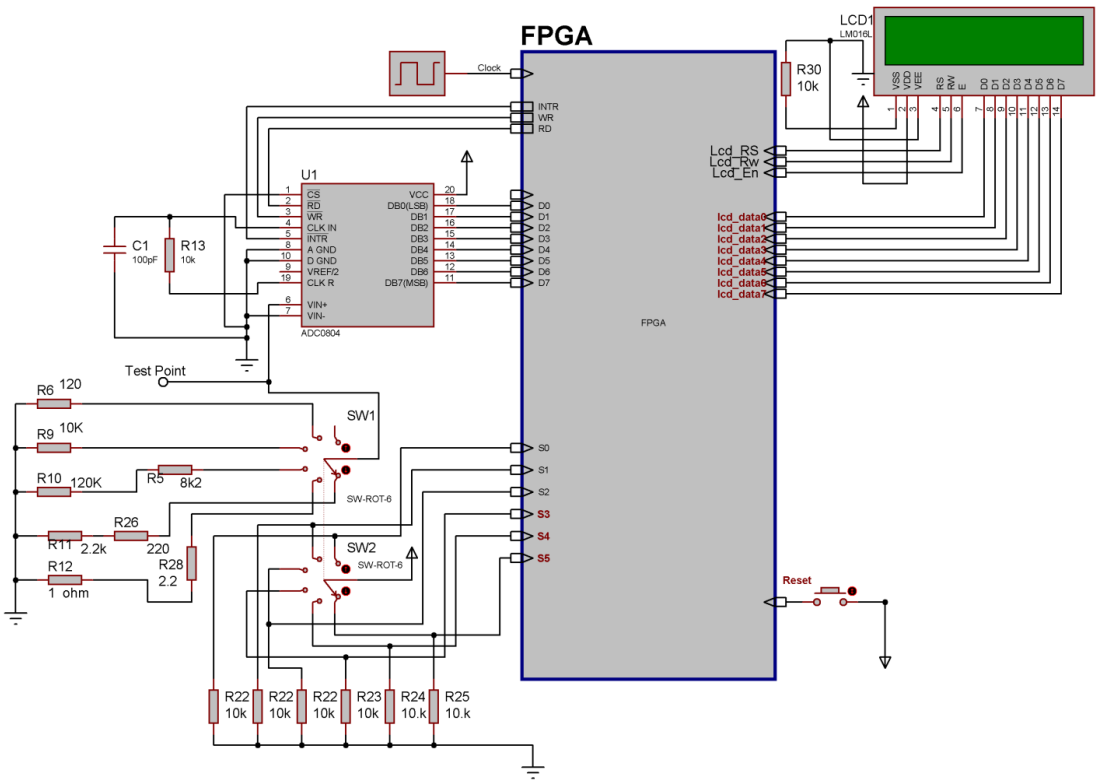
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity vol is
  port(clk,reset:in std_logic;
       SevenSeg:out std_logic_vector(7 downto 0);
       Sd : out std_logic_vector(2 downto 0);
       ADC:in std_logic_vector(7 downto 0);
             wr : out std_logic;
             rd : out std_logic;
             intr : in std_logic);
  end vol;

ARCHITECTURE structural OF vol IS

constant delay_between_read:integer := 1000; -- 25 us
type state is (start, read, wait_for_intr, delay, Calc, Display);
SIGNAL Dig0,Dig1,Dig2: std_logic_vector(7 downto 0);
SIGNAL volt:std_logic_vector(7 downto 0);
signal current : state:=Start;
begin
  process(clk,reset)
  variable count : integer;
  variable i:integer:=0;
  Variable Show:integer:=0;
    begin
      if reset = '1' then
                  SD<="111";
                  rd <= '1';
            wr <= '0';
            current <= start;
            count := 0;
                  i:=0;
                  Show:=0;
       elsif clk = '1' and clk'event then
             case current is
                when start =>
                   SD<="111";
                                wr <= '1';
                    current <= wait_for_intr;
                when wait_for_intr =>
                      if intr = '0' then
                         current <= read;
                         end if;
                when read =>
                     rd <= '0';
                     count := delay_between_read;
                     current <= delay;
                when delay =>
                     if count /= 0 then
                        if count = delay_between_read / 2 then
                           Volt <= ADC;
                              wr <= '0';
```

```
                        end if;
                      count := count - 1;
                    else
                      rd <= '1';
                      Dig0<="00000000";
                      Dig1<="00000000";
                      Dig2<="00000000";
                                  current <= Calc;
                                  i:=100;
                    end if;
                            when Calc=>
                                 if i=0 then
                                    if Volt>99 then
                                       Volt<=Volt-100;
                                          Dig2<=Dig2+1;
                                        else
                                           if Volt>9 then
                                                    Volt<=Volt-10;
                                                    Dig1<=Dig1+1;
                                              else
                                                 Dig0<=Volt;
                                                    Current
<=Display;
                                                    SD<="111";
                                                    Show:=0;
                                                    i:=0;
                                              end if;
                                        end if;
                                        else
                                           i:=i-1;
                                        end if;
                                when Display=>
                                   if Show<20000 then
                                         case i is
                                                  when 0 =>

      SD<="110";

      Volt<=Dig0;
                                                  when 150=>
                                                        SD<="111";

      Volt<="11000011";
                                                  when 200 =>

      SD<="101";

      Volt<=Dig1;
                                                  when 350=>
                                                        SD<="111";

      Volt<="11000011";
                                                  when 400 =>

      SD<="011";

      Volt<=Dig2;
                                                  when 550=>
                                                        SD<="111";
```

```
      Volt<="11000011";
                                                    when Others=>null;

                                              end case;
                                              i:=i+1;
                                              if i=600 then
                                                 SD<="111";
                                                    Show:=show+1;
                                                    i:=0;
                                              end if;
                                        CASE volt IS
                                              WHEN "00000000" =>
SevenSeg<="11000000";
                                              WHEN "00000001" =>
SevenSeg<="11110011";
                                              WHEN "00000010" =>
SevenSeg<="10001001";
                                              WHEN "00000011" =>
SevenSeg<="10100001";
                                              WHEN "00000100" =>
SevenSeg<="10110010";
                                              WHEN "00000101" =>
SevenSeg<="10100100";
                                              WHEN "00000110" =>
SevenSeg<="10000100";
                                              WHEN "00000111" =>
SevenSeg<="11110001";
                                              WHEN "00001000" =>
SevenSeg<="10000000";
                                              WHEN "00001001" =>
SevenSeg<="10100000";
                                              WHEN OTHERS=>
SevenSeg<="11111111";
                                         END CASE;

                                        else
                     Show:=0;
                     i:=0;
                     current<=Start;
                                        end if;
                   end case;
              end if;
    end process;
END structural;
```

۲۱- مالتی متر

شماتیک:



برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity vol is
  port(clk,reset:in std_logic;
       -- seven segement "hgfedcba"
       --SevenSeg:out std_logic_vector(7 downto 0);
             -- 7==seg select
       --Sd : out std_logic_vector(3 downto 0);
             -- mode selecte
       S  : in std_logic_vector(5 downto 0);
             -- adc data in
       ADC:in std_logic_vector(7 downto 0);

             lED:out std_logic_vector(7 downto 0);
             -- lcd Pins connector
             Lcd_Data:out std_logic_vector(7 downto 0);
             lcd_rs: out std_logic;
             Lcd_En: out std_logic;
             lcd_Wr: out std_logic;

             -- adc rd, rw, intr
             wr : out std_logic;
             rd : out std_logic;
```

```
             intr : in std_logic);
  end vol;

ARCHITECTURE structural OF vol IS
constant delay_between_read:integer := 30; -- 30 us
type Mainstate is (lcd_init,lcd_clear,Start, read, wait_for_intr,
delay_Read, CalcVolt,CalcOhm,CalcAmper,Showlcd);
type lcd_State is(none, Enable,Write);
-- main FSM
signal State : Mainstate:=Lcd_Init;
signal Lcd:Lcd_State:=None;
-- digit for 7 seg
-- volt read from adc
SIGNAL volt:std_logic_vector(9 downto 0);
SIGNAL TEST:std_logic_vector(7 downto 0);
-- volt & ohm & Amper
signal V1,V2,Ohm:std_logic_vector(19 downto 0);
--lcd signals
type init_array is array(0 to 2)of std_logic_vector(7 downto 0);
signal init: init_array:=("00111000", "00001110", "00000001");
type lcd_array is array(0 to 15)of std_logic_vector(7 downto 0);
signal Lcd_Line: lcd_array:=("00100000", "00100000", "00100000",
"00100000",
                                                    "00100000",
"00100000", "00100000", "00100000",
                                                    "00100000",
"00100000", "00100000", "00100000",
                                                    "00100000",
"00100000", "00100000", "00100000"
                                                    );
constant delay_for_e : integer := 1;--25 * 20 NS = 500  ns >450 NS
constant delay_for_write : integer := 5000;-- 5,000 *20 ns = 2.0 MS
      > 1.53Ms
begin
  process(clk,reset)
  variable count,Part,Delay : integer:=0;
  variable i,j,k:integer:=0;
    begin
      if reset = '1' then
                  --SD<="0000";
                  rd <= '1';
            wr <= '0';
            count := 0;
                  i:=0;
                  Part:=0;
                  Ohm<=(Others=>'0');
                  v1<=(Others=>'1');
                  v2<=(others=>'1');
                  TEST<=(OTHERS=>'1');
            State <= lcd_init;
                  lcd_rs<='0';
                  lcd_en<='0';
                  lcd_wr<='0';
                  lcd<=None;
                  -- preset variabels & signals
       elsif clk = '1' and clk'event then
                        if Delay>0 then
                              Delay:=Delay-1;
                        else
                              case lcd is
```

```
when Enable=>
            lcd_En<='1';
            Delay:= Delay_For_E;
            lcd<=Write;

when Write=>
            Lcd_En<='0';

      Delay:=Delay_for_Write;
            Lcd<=None;
when None=>
      case State is
            when lcd_init=>
                  if Part<2
then

      lcd_data<=init(Part);

      lcd_rs<='0';

      lcd<=Enable;

      Part:=Part+1;
                        else

      State<=Lcd_Clear;

      part:=0;
                        end if;
            when
lcd_clear=>
                              if
part<16 then

      lcd_Line(Part)<="00100000";

      Part:=Part+1;

      else

      lcd_Data<=init(2);

      lcd_rs<='0';

      State<=Start;

      lcd<=Enable;

      Part:=0;
                              end
if;
      -- Start Read from
adc
      when Start =>

      Count:=0;
                              wr
<= '1';
            State <=
wait_for_intr;
```

```
                                        -- wait for
intr =0
                                    when wait_for_intr =>
                                                    if
intr = '0' then
                                                        State <=
read;
                                                    end if;
                                                    -- read adc
state
                                    when read =>
                                                        rd <=
'0';
                                                        count :=
delay_between_read;
                                                        State <=
delay_read;
                                    when delay_Read =>
                                                    if count > 0
then
                                                    if count =
delay_between_read / 2 then
            Volt <= "0"&ADC&"0";
            test<=ADC;
            -- read data
            wr <= '0';
                                                        end if;
                                                    count := count
- 1;
                                                    else
      rd <= '1';
      -- select Show State
      If S(0)='1' then
            -- Show 00.00 in Left of lcd
            lcd_Line(0)<="00110000";
            lcd_Line(1)<="00110000";
            lcd_Line(2)<="00101110"; -- dot
            lcd_Line(3)<="00110000";
            lcd_Line(4)<="00110000";
            -- show Volt in Write of lcd
            lcd_line(12)<="01010110";
            lcd_line(13)<="01101111";
```

```
        lcd_line(14)<="01101100";
        lcd_line(15)<="01110100";
        State <= CalcVolt;
        Part:=0;
elsif S(3 downto 1)>"000" then
-- v1= 4.88 v ref base
              -- show 000.0 in left of lcd
              lcd_Line(0)<="00110000";
              lcd_Line(1)<="00110000";
              lcd_Line(2)<="00110000";
              lcd_Line(3)<="00101110"; -- dot
              lcd_Line(4)<="00110000";

              -- Show Ohm in Write of Lcd
              lcd_Line(13)<="01001111";
              lcd_Line(14)<="01101000";
              lcd_Line(15)<="01101101";

              V1(19 downto 0)<="00000000001000000000";
              i:=0;
              Part:=0;
              State <= CalcOhm;
        elsif S(5 downto 4)>"00" then
                    -- show 000.0 in left of lcd
                    lcd_Line(0)<="00110000";
                    lcd_Line(1)<="00110000";
                    lcd_Line(2)<="00110000";
                    lcd_Line(3)<="00101110"; -- dot
                    lcd_Line(4)<="00110000";
                    -- Show Amp in write of lcd
                    lcd_Line(13)<="01000001";
```

```
                        lcd_Line(14)<="01101101";
                        lcd_Line(15)<="01110000";
                        V1(19 downto 0)<= "0000000000"&Volt(9 downto
0);
                        Part:=0;
                        State <= CalcAmper;
                  else
                        State<=CalcVolt;
      end if;
      i:=0;
                                                        end if;
                                                        when CalcVolt
=>
                                                                    --
divide digits
      delay:=10;
                                                                    if
Volt>999 then
      Volt<=Volt-1000;
      Lcd_Line(0)<=Lcd_Line(0)+1;
      elsif Volt>99 then
      Volt<=Volt-100;
            Lcd_Line(1)<=lcd_Line(1)+1;
      elsif Volt>9 then
                  Volt<=Volt-10;
                  Lcd_Line(3)<=lcd_Line(3)+1;
            else
                  Lcd_Line(4)<=lcd_Line(4)+Volt(7 downto 0);
                  rd <= '1';
      wr <= '0';
                  i:=0;
                  Part:=0;
                  State <=Showlcd;
                                                                    end
if;
```

```
                                                    when CalcOhm =>

delay:=20;
                                                                    if
Part=0 then
      V2(19 downto 0)<="0000000000" & Volt(9 downto 0);
      Ohm<="00000000000000000000";
      Part:=1;
      elsif Part=1 then
            v1<=v1-v2;
            if v2>0 then
                  part:=2;
            else
                  part:=5;
            end if;
      elsif part=2 then
                  -- select Ohm meter renge
            case S(3 downto 1)is
                        when "001"=>  -- 127 ohm
                                    v1(19 downto 0)<= V1(12 downto
0)&"0000000" ;
                        when "010"=>      -- 10 k ohm
                                    lcd_line(12)<="01001011";
                                    v1(19 downto 0)<= V1(9 downto
0)&"0000000000" ;
                        when "100"=>      -- 120 k==k ohm
                                    lcd_line(12)<="01001011";
                                    v1(19 downto 0)<= V1(9 downto
0)&"0000000000" ;
                        when others=>  Ohm<="00000000000000000000";
                  end case;
                  part:=3;
                  i:=0;
```

```
        elsif Part=3 then
                    if v1>v2 then
                    v1<=v1-v2;
                    Ohm<=Ohm+1;
                    else

                          Part:=4;
                    end if;
                elsif Part=4 then
                          part:=5;
                          -- divide ohm digits
                    elsif Part=5 then
                                if Ohm>999 then
                                Ohm<=Ohm-1000;
                                      lcd_line(0)<=Lcd_Line(0)+1;
                                elsif Ohm>99 then
                                      Ohm<=Ohm-100;

lcd_line(1)<=Lcd_Line(1)+1;
                                      elsif Ohm>9 then
                                                  Ohm<=Ohm-10;

lcd_line(2)<=Lcd_Line(2)+1;
                                            else

Lcd_Line(4)<="00110000"+Ohm(7 downto 0);
                                                  rd <= '1';
                                            wr <= '0';
                                                  Part:=6;
                                 end if;
                          elsif part=6 then
                                      case s(3 downto 1) is
```

```
                                        when "001"=>
Lcd_Line(3)<=Lcd_Line(4);
      LCd_Line(4)<="00100000";
                                        when "010" =>
      Lcd_Line(3)<=Lcd_Line(2);
Lcd_Line(2)<="00101110";
                                         when "100"=>null;
                                         when Others=>
      lcd_Line(0)<="01000101";
      lcd_Line(1)<="01110010";
      lcd_Line(2)<="01110010";
      lcd_Line(3)<="01101111";
      lcd_Line(4)<="01110010";
                                    end case;
                                    part:=7;
                               else
                                    Part:=0;
                                    i:=0;
                                    State <= ShowLcd;
                                                          end
if;                                          when
CalcAmper=>
      delay:=10;
                                          if Part=0
then
                                          -- select
Amper type
```

```
      case S(5 downto 4) is
            when "01"=>   -- Amper and  mili amper
                        v1(19 downto 0)<="0"&v1(19 downto 1)+v1(17
downto 0)&"0";
                        lcd_line(9) <="01101101";
                        lcd_line(10)<="01101001";
                        lcd_line(11)<="01101100";
            when others=>     -- micro amper
                        v1(19 downto 0)<=v1(17 downto 0)&"00";
                        lcd_line(7) <="01101101";
                        lcd_line(8) <="01101001";
                        lcd_line(9) <="01100011";
                        lcd_line(10)<="01110010";
                        lcd_line(11)<="01101111";
      end case;
      Part:=1;
      --divide amper digits
      elsif Part=1 then
            part:=2;
      elsif part=2 then
                  if V1>999 then
                  V1<=V1-1000;
                        lcd_line(0)<=Lcd_Line(0)+1;
                  elsif V1>99 then
                        V1<=V1-100;
                              lcd_line(1)<=Lcd_Line(1)+1;
                        elsif V1>9 then
                                    V1<=V1-10;
                                    lcd_line(2)<=Lcd_Line(2)+1;
                              else
```

```
                                    lcd_line(4)<="00110000"+ V1(7
downto 0);
                                    rd <= '1';
                        wr <= '0';
                                    i:=0;
                                    Part:=3;
                  end if;
            elsif part=3 then
                  --if s(5)='1' then
                        lcd_line(3)<=lcd_line(4);
                        lcd_line(4)<="00100000";
                  --end if;
                  part:=0;
                  State <= ShowLcd;
            end if;
when Showlcd=>
      if Part<16 then
Lcd_Data<=Lcd_Line(Part);
      lcd_Rs<='1';
      Lcd<=Enable;
      Part:=Part+1;
      else
      Part:=0;
      State<=Lcd_Clear;
      Delay:=1000000;
            end if;
      when others=>     State <= lcd_init;
      test<="11111111";
                                                end case;
                                                end case;
                                    end if;
                  end if;
                        LED<=TEST;
    end process;

end structural;
```

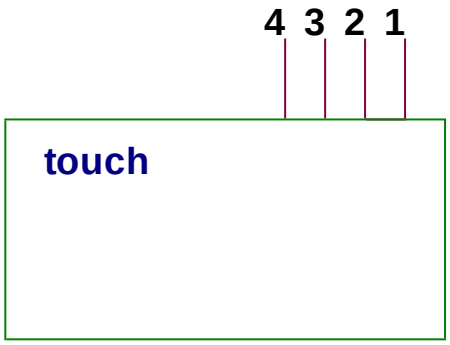
۲۲- اتصال touch

میدانیم که Touch با استفاده از مقاومتهایی که در خود دارد دو مقدار مختلف بـه مـا میدهـد یکـی برای مقدارx دیگری برای y که به صورت ولتاژ آنالوگ می باشد بنابراین برای این پروژه در ابتدا باید یک مدار تبدیل آنالوگ به دیجیتال داشته باشیم زیرا FPGA مقادیر آنالوگ را شناسـایی نمیکنـد و سپس مقادیر دیجیتال بدست آمده را تبدیل به چهار مقدار segment,page ,column و Data تبدیل کنیم که بتوانیم نقطه متناظر فشرده شده در Touch را در GLCD نمایش دهیم.

برنامه ماشامل State های مختلفی می باشد که هر کدام وظیفه منحصر به خود را دارد که برای هر کدام در خود برنامه کامنت گذاری شده و توضیح داده شده است.

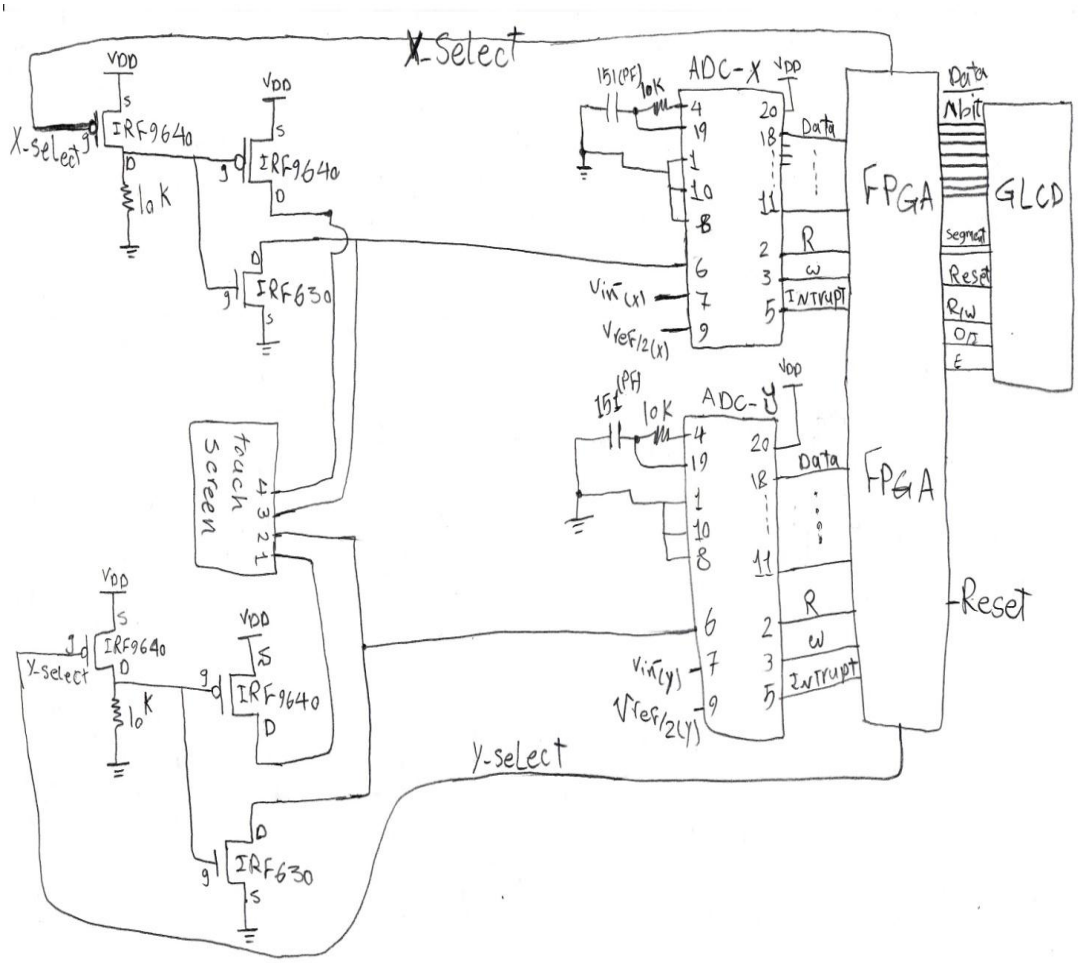
شکل مدار این برنامه را در قسمت های زیر مشاهده می کنیم:

شکل ظاهری صفحه لمسی و تشریح پایه های آن.

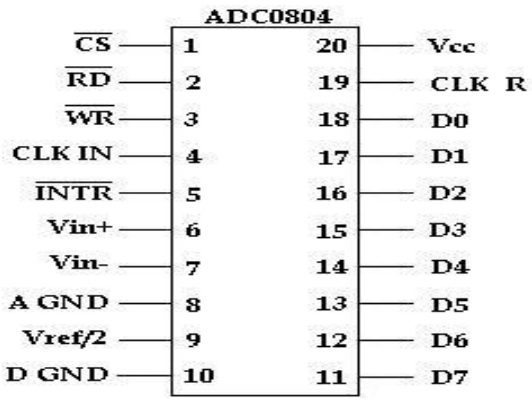


جدول زیر مربوط به محاسبه ولتاژ خروجی x و خروجی y در صفحه لمسی می باشد.

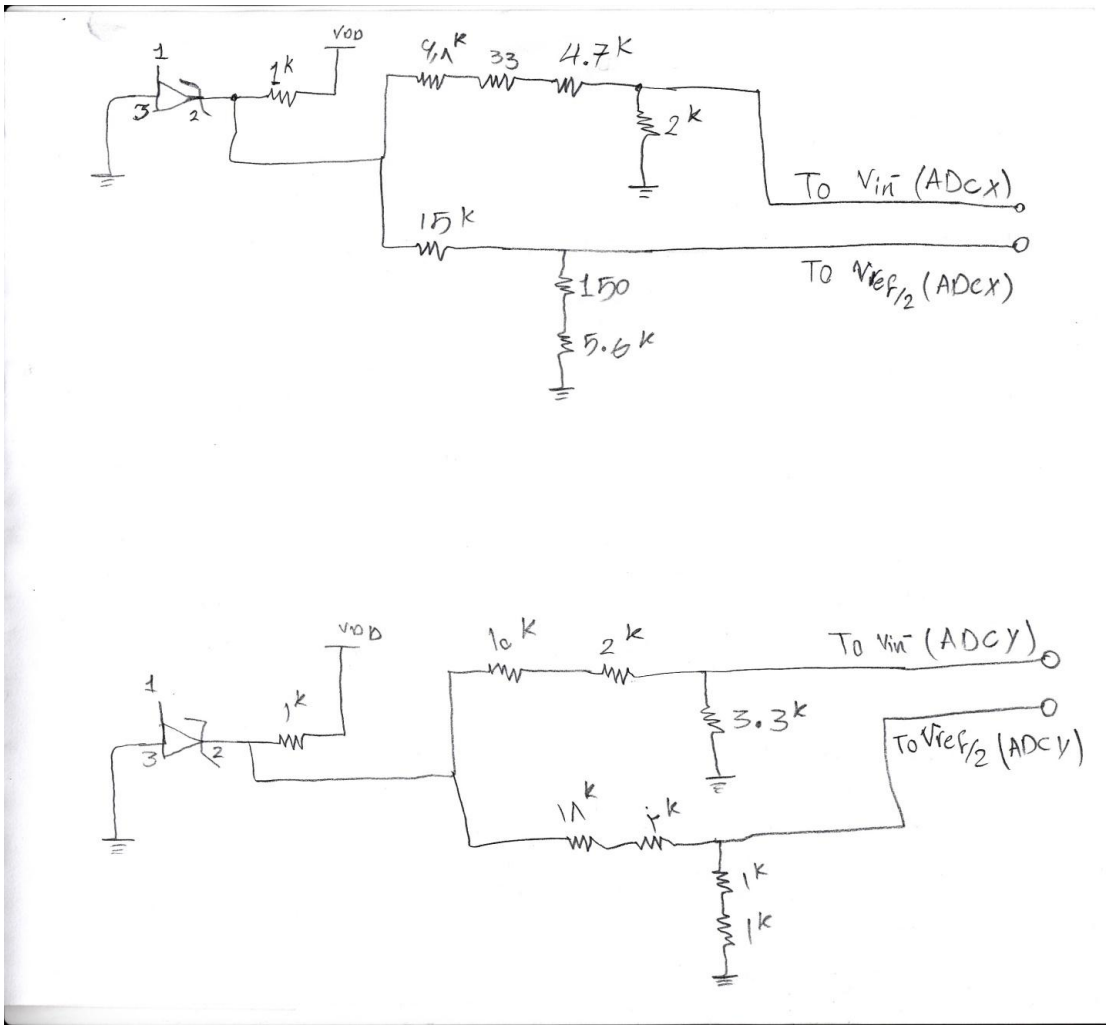
pin_1	pin_2	pin_3	pin_4
z	GND	v_x	v_{cc}
v_{cc}	v_y	GND	z



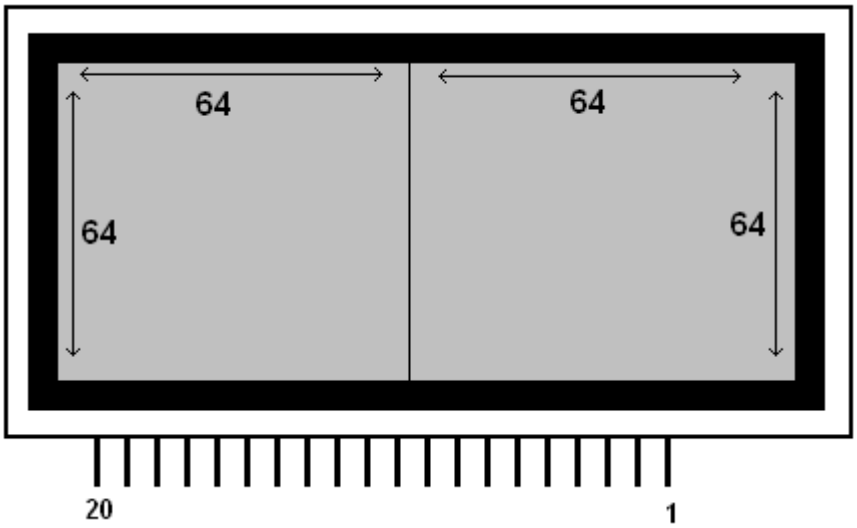
اتصال قطعات مدار به FPGA:



مدار شکل زیر مربوط به اعمال ولتاژهای مناسب به پایه های V_{in-} و $V_{ref/2}$ در adc های x و y می باشد تا بتوانیم خروجی صفحه لمسی را در صفحه نمایش کالیبره نماییم.

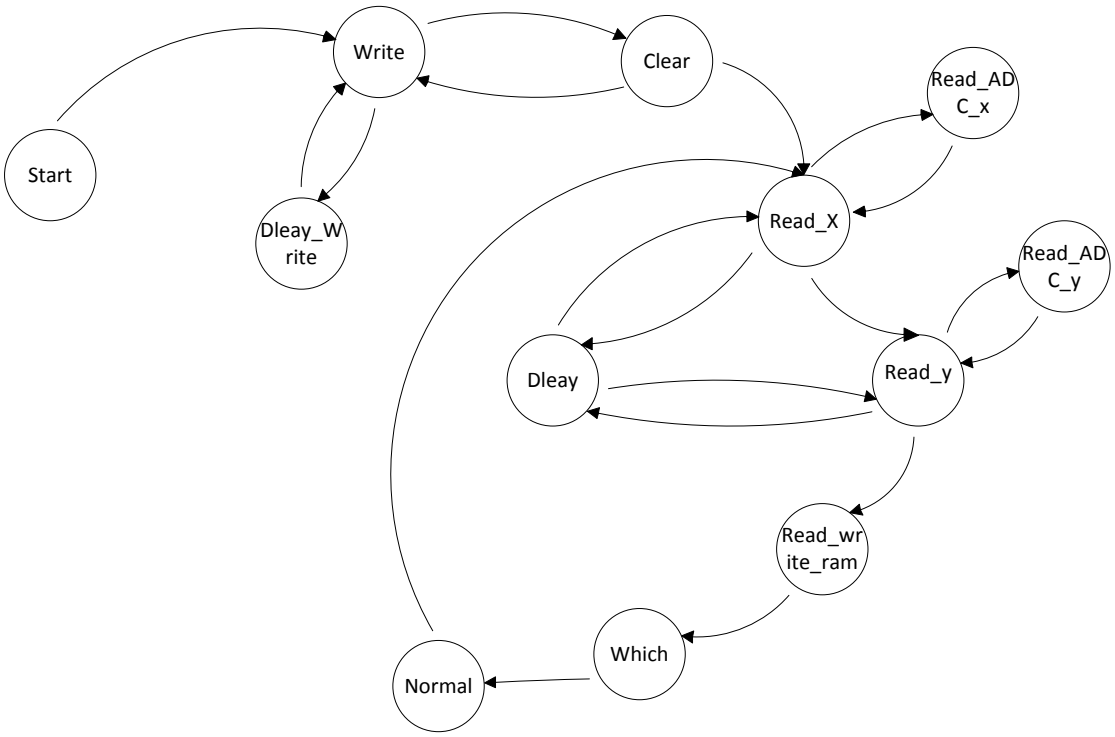


شکل GLCD و شرح پایه هایش :



Pin #	Symbol	Function	Pin #	Symbol	Function
1	VSS	GND	11	DB4	Data Bus Line 4
2	VDD	Power Supply (5V)	12	DB5	Data Bus Line 5
3	Vo	Contrast Adjustment	13	DB6	Data Bus Line 6
4	D/I	Data/Instruction	14	DB7	Data Bus Line 7
5	R/W	Data Read/Write	15	CS1	Chip Select for IC1
6	E	H → L Enable Signal	16	CS2	Chip Select for IC2
7	DB0	Data Bus Line 0	17	RST	Reset
8	DB1	Data Bus Line 1	18	Vee	Negative Voltage Output
9	DB2	Data Bus Line 2	19	A	Power Supply for LED (4.2V)
10	DB3	Data Bus Line 3	20	K	Power Supply for LED (0V)

State Diagram برنامه در زیر آمده است.



همانطور که مشاهده می شود این برنامه بعد از اینکه به انتهـا رسـید دو بـاره بـه State خوانـدن X برگشت میکند تا نقطه بعدی را در یافت کند و مکـان آنـرا نمـایش دهـد.در State هـای Read_x وRead_y به تعداد ۲۰ بار مقدار x و y خوانده می شود و با مقدار اولیه مقایسه مـی شـود اگـر برابـر بودند به Stateبعدی می رود اگر برابر نبودند دوباره شروع به خواندن مقادیر و مقایسه مـی کنـد تـا حداقل ۲۰ بار با هم برابر باشند .در برنامه یک خروجی با نام Result وجود دارد که نشان مـی دهـد مدار برنامه در طول اجرا تا کنون از کدام State ها خارج شده است،تاکید می گردد از کـدام State ها خارج شده است.

در قسمت زیر برنامه به همراه توضیحات آورده شده است.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity touch is
   Port ( data : out std_logic_vector(7 downto 0);--<= GLCD
          di : out std_logic;--<= GLCD
```

```
          rw : out std_logic;--<= GLCD
          e : out std_logic;--<= GLCD
          rst : out std_logic;--<= GLCD
          clk : in std_logic;--<= GLCD
          reset : in std_logic;--<= GLCD
          x_select : out std_logic;--<= GLCD
          y_select : out std_logic;--<= GLCD
          segment : out std_logic_vector(1 downto 0);--<= GLCD
 ---read from adc for x and y ------------
          wr_adc_x,rd_adc_x:out std_logic;--<= Read X & Y
          intr_x : in std_logic;--<= Read X & Y
          data_adc_x :in std_logic_vector(7 downto 0);--<= Read X & Y
          wr_adc_y,rd_adc_y:out std_logic;--<= Read X & Y
          intr_y : in std_logic;--<= Read X & Y
          data_adc_y :in std_logic_vector(7 downto 0);--<= Read X & Y
          result : out std_logic_vector(8 downto 0)--<= For Check
State's
          );
end touch;
architecture Behavioral of touch is
   signal page : std_logic_vector (7 downto 0);  --jame B8H +
y_digital (7 downto 5)
   signal column : std_logic_vector(7 downto 0); --jame 40H +
x_digital(6 downto 1)
   signal Pixel : std_logic_vector(7 downto 0);  --decod shode
y_digital(4 downto 2)
   constant delay_for_e : integer := 23;   --23 * 19 NS = 450 NS
baraye ijade labe
   constant delay_between_data : integer := 100000;    --baraye ijade
delay monaseb ta data roye khat gharar girad
   type state is (start, write, delay_write, delay, read_x,
read_y,Read_Write_SRAM,Read_ADC_x,Read_ADC_y, which, clear, normal);
--read_GLCD
   signal current, caller : state;
   signal part,part_mul_read_RAM_2,test_part,
part1,par,part_adr_RAM_1,part_Read_RAM_2,part_mul_RAM_2,part_Write_RA
M_1,part_Write_RAM_2,part_x,part_y : integer;--
part=>write,part1=>Normal,Part2=>read_GLCD,Part_X=>read_x,Part_y=>rea
d_y
   type state_which is (page_column,on_Pixel);--state haye lazem
baraye state which
   signal current_which : state_which;
   signal x_digital : std_logic_vector(7 downto 0);--maghadir khandeh
shodeh az adc
   signal y_digital : std_logic_vector(7 downto 0);--maghadir khandeh
shodeh az adc
   signal data_temp : std_logic_vector(7 downto 0);
   constant delay_between_read: integer :=1250;--for read from Analog
Digital Convertor
   type state_adc_x is
(start_adc_x,read_adc_x,wait_for_intr_x,delay_adc_x);--state haye
lazem baraye state Read_ADC_X
   signal current_adc_x : state_adc_x;
   type state_adc_y is
(start_adc_y,read_adc_y,wait_for_intr_y,delay_adc_y);--state haye
lazem baraye state Read_ADC_Y
   signal current_adc_y : state_adc_y;
   --baraye check kardane maghadire x & y
   signal check_part_x ,l :integer;
   signal x_1,x_for_check : std_logic_vector(7 downto 0);
```

```
   signal check_part_y ,m :integer;
   signal y_1,y_for_check : std_logic_vector(7 downto 0);
   type state_RAM is (check,Write_RAM_1,Read_RAM_2,Write_RAM_2);
   signal Current_RAM : state_RAM;
   signal adr_ram_1 : std_logic_vector(8 downto 0);
   signal adr_ram_2 : std_logic_vector(9 downto 0);
   signal dati_RAM_1 : std_logic_vector(8 downto 0);
      signal dato_RAM_1 : std_logic_vector(8 downto 0);
   signal dati_RAM_2 : std_logic_vector(7 downto 0);
      signal dato_RAM_2 : std_logic_vector(7 downto 0);
   signal wen_RAM_1, wen_RAM_2 : std_logic ;
   signal r1_add_counter : integer range 0 to 511;
      signal c :integer;
      signal mul :std_logic_vector(9 downto 0);
------expalin RAM_2--------------
  type mem_array_2 is array (0 to 1023) of std_logic_vector(7 downto
0);
    signal ram_2 : mem_array_2;
------explain RAM_1--------------
    type mem_array_1 is array (0 to 511) of std_logic_vector(8 downto
0);
    signal ram_1 : mem_array_1;
begin
   process (clk, reset)
     variable i, count : integer;
     variable count_adc_x:integer;
     variable count_adc_y:integer;
    begin

          if reset = '1' then
           current <= start;
              Current_RAM<=check;
           di <= '0';
           rw <= '0';
           e <= '0';
              c<=0;
           segment <= "11";
           rst <= '1';
           count := 0;
           i := 0;
           l<=0;
           m<=0;
           part_x <= 0;
           part_y <= 0;
           part <= 0;
           part1 <= 0;
              par<= 0;
              test_part<=0;
              part_mul_read_RAM_2 <=0;
              part_adr_RAM_1 <=0;
              part_Read_RAM_2<=0;
       part_Write_RAM_1 <=0;
       part_Write_RAM_2 <=0;
       part_mul_RAM_2<=0;
       r1_add_counter<=0;
          check_part_x<=0;
          check_part_y<=0;
          page <= "00000000";
          current_which<= page_column;
          rd_adc_x<= '1';
```

```
            wr_adc_x<= '0';
            current_adc_x<= start_adc_x;
            count_adc_x :=0;
            rd_adc_y<= '1';
            wr_adc_y<= '0';
            current_adc_y<= start_adc_y;
            count_adc_y :=0;
          result<="000000000";
           elsif clk = '1' and clk'event then
            case current is
             when start =>
               data <= "00111111";
               di <= '0';
               caller <= clear;
               current <= write;
-------------------------write----------------------
             when write =>
                   if part = 0 then
                     e <= '1';
                     count := delay_for_e;
                     part <= part + 1;
                     current <= delay_write;
                   elsif part = 1 then
                     e <= '0';
                     count := delay_between_data;
                     current <= delay_write;
                     part <= part + 1;
                   else
                     current <= caller;
                     part <= 0;
                   end if;
-------------------------delay_write----------------------
          when delay_write =>
                count := count - 1;
                   if count = 0 then
                     current <= write;
           end if;
-------------------------clear----------------------
             when clear =>
                   if i = 0 then
                     data <= "10111000" + page;
                     di <= '0';
                  caller <= clear;
                      current <= write;
                      i := i + 1;
                    elsif i = 1 then
                      data <= "01000000";
                      di <= '0';
                      caller <= clear;
                      current <= write;
                      i := i + 1;
                    elsif i > 1 and i < 66 then
                      data <= "00000000";
                      di <= '1';
                      caller <= clear;
                      current <= write;
                      i := i + 1;
                    elsif i = 66 then
                      page <= page + 1;
                      i := 0;
```

```
                    if page = 7 then
                      current <= read_x;
                    end if;
                  end if;
-------------------------read_x-----------------
              when read_x =>
                if part_x= 0 then
                  x_select <= '1';
                  y_select <= '0';
                  part_x <= part_x+1;
                  count := 10000; --delay monaseb ta voltage ha az
transistor ha obur konad va touch voltage khorouji bedahad
                  current <= delay;
                  caller <= read_x;
                elsif part_x= 1 then
               if check_part_x=0 then
                 l<=l+1;
                    current <=Read_ADC_x ;
                    caller<=read_x;
                 check_part_x<=check_part_x+1;
               elsif check_part_x=1 then
                 if l=1 then
                   x_1<=x_for_check;
                 end if;
                 l<=l+1;
                 current <=Read_ADC_x;
                 check_part_x<=check_part_x+1;
               elsif check_part_x=2 then
                 if x_1 = x_for_check then
                                        if l<14 then

                                          check_part_x<=1;
                                          l<=l+1;
                     elsif l=14 then
                                          l<=0;
                                          check_part_x<=0;
                                          part_x<=Part_x+1;
                                    x_digital<="11111111" - x_1;
                     end if;
                  else
                   check_part_x<=0;
                   l<=0;
                end if;
              end if;
                elsif part_x= 2 then
                  current <= read_y;
                  part_x <= 0;
                end if;
-------------------------delay-----------------
              when delay =>
                count := count - 1;
                if count = 0 then
                  current <= caller;
                end if;
-------------------------read_y-----------------
              when read_y =>
                if part_y= 0 then
                  y_select <= '1';
                  x_select <= '0';
                  part_y <= part_y+1;
```

```
                count := 10000; --delay monaseb ta voltage ha az
transistor ha obur konad va touch voltage khorouji bedahad
                current <= delay;
                caller <= read_y;
              elsif part_y= 1 then
             if check_part_y=0 then
               m<=m+1;
                  current <=Read_ADC_y;
               check_part_y<=check_part_y+1;
             elsif check_part_y=1 then
                              if m=1 then
                 y_1<=y_for_check;
                              end if;
               m<=m+1;
               current <=Read_ADC_y;
               check_part_y<=check_part_y+1;
             elsif check_part_y=2 then
               if y_1 = y_for_check then
                                    if m<16 then
                                      check_part_y<=1;
                                      m<=m+1;
                 elsif m=16 then
                                      m<=0;
                                      check_part_y<=0;
                                      part_y<=Part_y+1;
                                  y_digital<=y_1;
                end if;
               else
                check_part_y<=0;
                m<=0;
              end if;
            end if;
              elsif part_y= 2 then
                current <= which;
                part_y <= 0;
              end if;
-------------------------Read_ADC_x-----------------
            when Read_ADC_x =>
              case current_adc_x is
                when start_adc_X =>
                  wr_adc_x<='1';
                  current_adc_x<= wait_for_intr_x;
                when wait_for_intr_x =>
                  if(intr_x='0') then
                    current_adc_x<=read_adc_x;
                  end if;
                when read_adc_x =>
                  rd_adc_x<='0';
                  count_adc_x:=delay_between_read;
                  current_adc_x<=delay_adc_x;
                when delay_adc_x =>
                  if(count_adc_x/=0)then
                    if(count_adc_x=delay_between_read/2)then
                      x_for_check<= data_adc_x;
                      wr_adc_x<='0';
                    end if;
                    count_adc_x:=count_adc_x-1;
                  else
                    rd_adc_x<='1';
                    current_adc_x <= start_adc_x;
```

```
                    current <= read_x;
                  end if;
                end case;
-------------------------Read_ADC_y-----------------
              when Read_ADC_y =>
                case current_adc_y is
                  when start_adc_y =>
                    wr_adc_y<='1';
                    current_adc_y<= wait_for_intr_y;
                  when wait_for_intr_y =>
                    if(intr_y='0') then
                      current_adc_y<=read_adc_y;
                    end if;
                  when read_adc_y =>
                    rd_adc_y<='0';
                    count_adc_y:=delay_between_read;
                    current_adc_y<=delay_adc_y;
                  when delay_adc_y =>
                    if(count_adc_y/=0)then
                      if(count_adc_y=delay_between_read/2)then
                        y_for_check<= data_adc_y;
                        wr_adc_y<='0';
                      end if;
                      count_adc_y:=count_adc_y-1;
                    else
                      rd_adc_y<='1';
                      current_adc_y <= start_adc_y;
                      current <= read_y;
                    end if;
                end case;
-------------------------which-----------------
              when which =>
                    if (x_digital="11111111"  and
y_digital="00000000" )then--BARAY gheyre fa'al kardan noqte (0,0)
               current<=read_x;
                       else
                case current_which is
                  when page_column =>
                    if ( x_digital(7)= '0' ) then
                      segment <= "01"; --- baray moshakhas kardane
Segment aval
                    elsif ( x_digital(7)= '1' ) then
                      segment <= "10"; --- baray moshakhas kardane
Segment dovom
                    end if;
                    column <= "01" & x_digital(6 downto 1); --- baray
moshakhas kardane kodam soton az page
                    page <= "10111" & y_digital (7 downto 5); ---
baray moshakhas kardane kodam page az segment
                    current_which<=on_Pixel;
                  when on_Pixel =>
                    case y_digital(4 downto 2) is
                      when "000" =>
                        pixel <= "00000001";
                        current <= Read_Write_SRAM;
                        current_which<= page_column;
                      when "001" =>--<=Roshan shodane 1th pixcel
                        pixel <= "00000010";
                        current <= Read_Write_SRAM;
                        current_which<= page_column;
```

```
                    when "010" =>--<=Roshan shodane 2nd pixcel
                      pixel <= "00000100";
                      current <= Read_Write_SRAM;
                      current_which<= page_column;
                      result(6) <= '1';
                    when "011" =>--<=Roshan shodane 3th pixcel
                      pixel <= "00001000";
                      current <= Read_Write_SRAM;
                      current_which<= page_column;
                    when "100" =>--<=Roshan shodane 4th pixcel
                      pixel <= "00010000";
                      current <= Read_Write_SRAM;
                      current_which<= page_column;
                    when "101" =>--<=Roshan shodane 5th pixcel
                      pixel <= "00100000";
                      current <= Read_Write_SRAM;
                      current_which<= page_column;
                    when "110" =>--<=Roshan shodane 6th pixcel
                      pixel <= "01000000";
                      current <= Read_Write_SRAM;
                      current_which<= page_column;
                    when "111" =>--<=Roshan shodane 7th pixcel
                      pixel <= "10000000";
                      current <= Read_Write_SRAM;
                      current_which<= page_column;
                    when others  =>
                      pixel <= "00000000";
                      current <= Read_Write_SRAM;
                      current_which<= page_column;
                  end case;
               end case;
     end if;-----
-------------------------Read_Write_SRAM---------
   when Read_Write_SRAM =>
case Current_RAM is
   when check =>
          if c < r1_add_counter then
               if part_adr_RAM_1 = 0 then
               adr_RAM_1<=Conv_std_logic_vector(c,9);
                        part_adr_RAM_1 <= 1;
                  elsif part_adr_RAM_1 = 1 then
              if dato_RAM_1 = ( y_digital(7 downto 5) & X_digital(6
downto 1) ) then
             Current_RAM <= Read_RAM_2;
             c<=0;                          --reset counter
                        part_adr_RAM_1 <= 0; --reset if
                     result(0)<='1';            --allert for
outgoing
           else
                     c <=c+1;  --reset counter
                     part_adr_RAM_1 <= 0; --reset if
                   end if;
         end if;
       else
               Current_Ram<=Write_RAM_1;  --agar sabet shavad ghablan
dar ram2 dar satr morede nazar naneveshteh eim;
               c<=0;                                                --
reset counter
               result(1)<='1';
             end if;
```

```
   when Write_RAM_1 =>
     if part_Write_RAM_1=0 then
       wen_RAM_1<='1';
       adr_RAM_1<=Conv_std_logic_vector(r1_add_counter,9);
       dati_RAM_1<=( y_digital(7 downto 5) & X_digital(6 downto 1) );
       r1_add_counter<=r1_add_counter+1;
       part_Write_RAM_1<=1;
     elsif part_Write_RAM_1=1 then
       wen_RAM_1<='0';
       part_Write_RAM_1<=0;
       Current_Ram<=Write_RAM_2;
             result(2)<='1';
       data_temp<=Pixel;        --baraye in ke pixel haman dadeh E
ast ke bayad dar ram2 zakhireh shavad
     end if;
   when Read_RAM_2 =>
        if part_mul_read_RAM_2 < conv_integer(y_digital(7 downto 5))
then--baraye sakhtane adress ram 2
         mul <= mul +("00" & "10000000") ; --mohasebeye adress
"ghesmate kodam page"
            part_mul_read_RAM_2<=part_mul_read_RAM_2+1;
     else
          if part_Read_RAM_2 = 0 then
        adr_RAM_2<=(mul + ("0000" & X_digital(6 downto 1)));      --
adress for read from ram 2
                part_Read_RAM_2 <= 1;
       elsif part_Read_RAM_2 = 1    then
         data_temp <= ( dato_RAM_2 or pixel ); --macke or new and old
data
         Current_Ram<=Write_RAM_2;  --next hop
               part_Read_RAM_2 <= 0;       --reset if
                  part_mul_read_RAM_2 <= 0;     --reset if
                  mul<="0000000000";            --reset value of mul
               result(3)<='1';                  --allert for outgoing
       end if;
     end if;
   when Write_RAM_2 =>
     if part_Write_RAM_2=0 then
          if part_mul_RAM_2 < conv_integer(y_digital(7 downto 5))
then
            mul <= mul +("00" & "10000000") ;
                  part_mul_RAM_2<=part_mul_RAM_2+1;
       else
               if par = 0 then
           wen_RAM_2<='1';
           adr_RAM_2<= mul + X_digital (6 downto 1);
           dati_RAM_2<=data_temp;
                    par <= 1;
                  elsif par = 1 then
           part_Write_RAM_2<=1;
                    part_mul_RAM_2<=0;
                    par<=0;
                  end if;
       end if;
     elsif part_Write_RAM_2=1 then
       wen_RAM_2<='0';
       part_Write_RAM_2<=0;
       Current_Ram<=check;
       Current<=normal;
             mul<="0000000000";
```

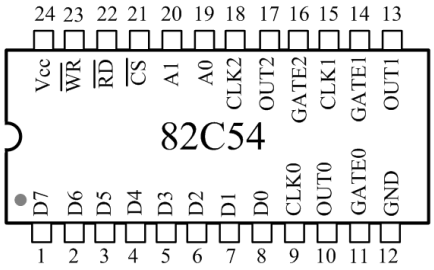
```
              result(7) <= '1';
              result(4)<='1';
      end if;
    end case;
-------------------------normal-----------------
               when normal =>
                 if part1 = 0 then
               rw <= '0';
                   data <= page;
                   di <= '0';
                   caller <= normal;
                   current <= write;
                   part1 <= part1 + 1;
                 elsif part1 = 1 then
                   data <= column;
                   di <= '0';
                   caller <= normal;
                   current <= write;
                   part1 <= part1 + 1;
                 elsif part1 = 2 then
                   data <= pixel or data_temp;
                   di <= '1';
                   caller <= normal;
                   current <= write;
                   part1 <= part1+ 1;
                 elsif part1 = 3 then
                   part1 <= 0;
                   current <= read_x;
                   result(8) <= '1';
                 end if;
             end case;
           end if;
             if clk = '1' and clk'event then
           if (wen_RAM_2='1') then
            ram_2( conv_integer(adr_RAM_2) ) <= dati_RAM_2;
           end if;
           if (wen_RAM_1='1') then
            ram_1( conv_integer(adr_RAM_1) ) <= dati_RAM_1;
           end if;
        end if;
     end process;
        dato_RAM_2 <= ram_2(conv_integer(adr_RAM_2));
     dato_RAM_1 <= ram_1(conv_integer(adr_RAM_1));
end Behavioral;
```

۲۳- اتصال 8254 (تایمر قابل برنامه‌ریزی) به FPGA

هدف، اتصال تراشه تایمر قابل برنامه‌ریزی 8254 به FPGA می‌باشد. در این مدار با استفاده از تایمر، فرکانس یک موج مربعی با فرکانس مشخص (همان موج پایه‌های کریستال میکرو) را بر اعداد مختلف تقسیم می‌کنیم و نتیجه را روی اسیلوسکپ مشاهده می‌نماییم.

آشنایی با تایمر PIT

شکل ۱ پایه‌های تراشه 82C54 را نشان می‌دهد.



شکل ۱ پایه‌های تراشه 8254

شرح پایه‌های تراشه 8254:

D0-D7: هشت خط برای ارتباط دوجهته با میکروکنترلر

RD: خواندن از رجیسترهای تراشه

WR: نوشتن در رجیسترهای تراشه

CS: فعال سازی تراشه

CLK0-2: پایه‌های ورودی کلاک

GATE0-2: فعال سازی کانترهای تراشه

OUT0-2: خروجی حاصل تقسیم

A0 و A1 خطوط آدرس:

00 : انتخاب Counter 0

01: انتخاب Counter 1

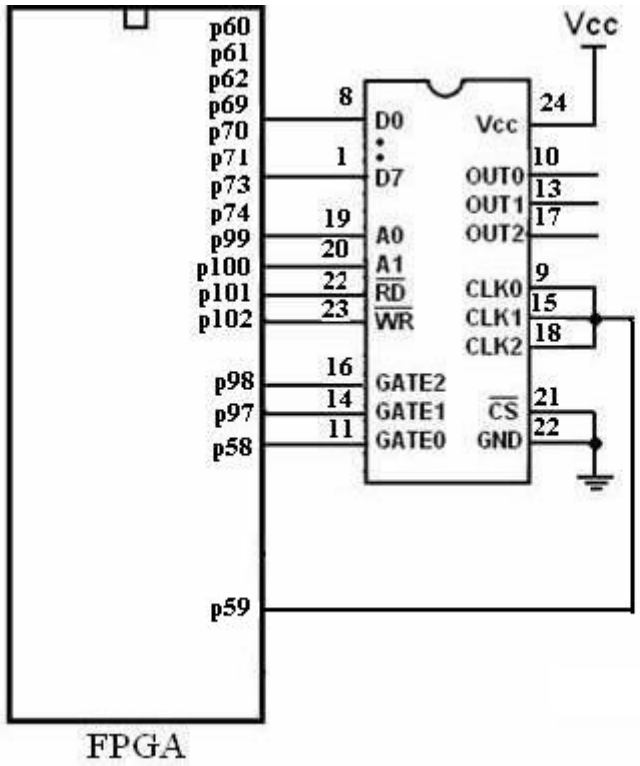
10 : انتخاب Counter 2

11: انتخاب رجیستر کنترلی

برای اطلاعات بیشتر در مورد رجیسترهای تراشه به DATASHEET تراشه مراجعه کنید.

عملکرد تراشه 8254:

میکروکنترلر پس از برنامه ریزی رجیستر کنترلی، عددی که باید کلاک ورودی بر آن تقسیم شود را در دو بایت به تراشه ارسال می‌کند.



شکل ۲ مدار اتصال ۸۲۵۴ به FPGA

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
D_7–D_0	1–8	I/O	**DATA:** Bi-directional three state data bus lines, connected to system data bus.
CLK 0	9	I	**CLOCK 0:** Clock input of Counter 0.
OUT 0	10	O	**OUTPUT 0:** Output of Counter 0.
GATE 0	11	I	**GATE 0:** Gate input of Counter 0.
GND	12		**GROUND:** Power supply connection.
V_{CC}	24		**POWER:** +5V power supply connection.
$\overline{WR}$	23	I	**WRITE CONTROL:** This input is low during CPU write operations.
$\overline{RD}$	22	I	**READ CONTROL:** This input is low during CPU read operations.
$\overline{CS}$	21	I	**CHIP SELECT:** A low on this input enables the 8254 to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and $\overline{WR}$ are ignored otherwise.
A_1, A_0	20–19	I	**ADDRESS:** Used to select one of the three Counters or the Control Word Register for read or write operations. Normally connected to the system address bus.

A_1	A_0	Selects
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Word Register

CLK 2	18	I	**CLOCK 2:** Clock input of Counter 2.
OUT 2	17	O	**OUT 2:** Output of Counter 2.
GATE 2	16	I	**GATE 2:** Gate input of Counter 2.
CLK 1	15	I	**CLOCK 1:** Clock input of Counter 1.
GATE 1	14	I	**GATE 1:** Gate input of Counter 1.
OUT 1	13	O	**OUT 1:** Output of Counter 1.

جدول تشریح پایه های IC ۸۲۵۴

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Output	Null Count	RW1	RW0	M2	M1	M0	BCD

D_7 1 = OUT Pin is 1
0 = OUT Pin is 0

D_6 1 = Null Count
0 = Count available for reading

D_5–D_0 Counter programmed mode (see Figure 7)

Figure 11. Status Byte

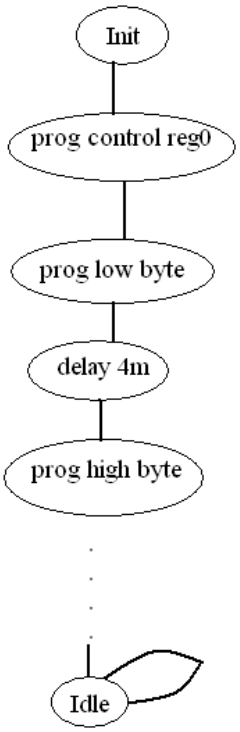
D0 =نوع شمارنده(باینری یا BCD)

D1,D2,D3=موج مربعی

D4,D5 =ابتدا خواندن LSB وسپس MSB

D6,D7 = مربوط بهشماره کانال

State diagram for timer:



برنامه تایمر :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity timer is
            port(
                  -- 82C54 port
                  Data        : out std_logic_vector(7 downto 0);
                  Address     : out std_logic_vector(1 downto 0);
                  RDn         : out std_logic;
```

```
                  WRn         : out std_logic;
                  GATE        : out std_logic_vector(2 downto 0);
                  CLK         : out std_logic;
                  -- General
                  CLKin       : in std_logic
                  );
end timer;

architecture Behavioral of timer is
-- CLKin is 11.059 MHz
constant delay_4us: integer:= 12*4;
constant delay_1us: integer:= 12;
typestate_type
is(init,ControlReg_0,delay4u_c0,delay1u_c0,delay4u_t0L,delay1u_t0L,de
lay4u_t0H,delay1u_t0H,
      ControlReg_1,delay4u_c1,delay1u_c1,delay4u_t1L,delay1u_t1L,dela
y4u_t1H,delay1u_t1H,
      ControlReg_2,delay4u_c2,delay1u_c2,delay4u_t2L,delay1u_t2L,dela
y4u_t2H,delay1u_t2H,idle);

signal state:state_type:=init;
signal CLK_D4:std_logic:='0';
signal tcnt:std_logic:='0';
signal dcnt:unsigned(7 downto 0):=(others=>'0');

begin
process(CLKin)
begin
      if(rising_edge(CLKin)) then
            tcnt<= not tcnt;
            if(tcnt='0') then
                  CLK_D4<= not CLK_D4;
            else
                  CLK_D4<= CLK_D4;
            end if;
      end if;
end process;
CLK<=CLK_D4;
RDn<='1';
process(CLKin)
begin
      if(rising_edge(CLKin)) then
            if(state=init) then
                  Data<=X"00";
                  Address<="11";
                  WRn<='1';
                  GATE<="111";
                  state<=ControlReg_0;
                  --------------------------
            elsif(state=ControlReg_0) then
                  Address<="11";
                  Data<=X"36";
                  WRn<='0';
                  dcnt<=(others=>'0');
                  state<=delay4u_c0;
            elsif(state=delay4u_c0) then
                  if(to_integer(dcnt)<delay_4us) then
                        dcnt<=dcnt+1;
                        state<=state;
                  else
```

```
            WRn<='1';
            dcnt<=(others=>'0');
            state<=delay1u_c0;
      end if;
elsif(state=delay1u_c0) then
      if(to_integer(dcnt)<delay_1us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            Address<="00";
            -- LOW Byte
            Data<=X"82";
            WRn<='0';
            dcnt<=(others=>'0');
            state<=delay4u_t0L;
      end if;
elsif(state=delay4u_t0L) then
      if(to_integer(dcnt)<delay_4us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            WRn<='1';
            dcnt<=(others=>'0');
            state<=delay1u_t0L;
      end if;
elsif(state=delay1u_t0L) then
      if(to_integer(dcnt)<delay_1us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            -- HIGH Byte
            Data<=X"42";
            WRn<='0';
            dcnt<=(others=>'0');
            state<=delay4u_t0H;
      end if;
elsif(state=delay4u_t0H) then
      if(to_integer(dcnt)<delay_4us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            WRn<='1';
            dcnt<=(others=>'0');
            state<=delay1u_t0H;
      end if;
elsif(state=delay1u_t0H) then
      if(to_integer(dcnt)<delay_1us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            state<=ControlReg_1;
      end if;
      ---------------------------
elsif(state=ControlReg_1) then
      Address<="11";
      Data<=X"76";
      WRn<='0';
      dcnt<=(others=>'0');
      state<=delay4u_c1;
elsif(state=delay4u_c1) then
```

```
		if(to_integer(dcnt)<delay_4us) then
			dcnt<=dcnt+1;
			state<=state;
		else
			WRn<='1';
			dcnt<=(others=>'0');
			state<=delay1u_c1;
		end if;
	elsif(state=delay1u_c1) then
		if(to_integer(dcnt)<delay_1us) then
			dcnt<=dcnt+1;
			state<=state;
		else
			Address<="01";
			-- LOW Byte
			Data<=X"41";
			WRn<='0';
			dcnt<=(others=>'0');
			state<=delay4u_t1L;
		end if;
	elsif(state=delay4u_t1L) then
		if(to_integer(dcnt)<delay_4us) then
			dcnt<=dcnt+1;
			state<=state;
		else
			WRn<='1';
			dcnt<=(others=>'0');
			state<=delay1u_t1L;
		end if;
	elsif(state=delay1u_t1L) then
		if(to_integer(dcnt)<delay_1us) then
			dcnt<=dcnt+1;
			state<=state;
		else
			-- HIGH Byte
			Data<=X"21";
			WRn<='0';
			dcnt<=(others=>'0');
			state<=delay4u_t1H;
		end if;
	elsif(state=delay4u_t1H) then
		if(to_integer(dcnt)<delay_4us) then
			dcnt<=dcnt+1;
			state<=state;
		else
			WRn<='1';
			dcnt<=(others=>'0');
			state<=delay1u_t1H;
		end if;
	elsif(state=delay1u_t1H) then
		if(to_integer(dcnt)<delay_1us) then
			dcnt<=dcnt+1;
			state<=state;
		else
			state<=ControlReg_2;
		end if;
	---------------------------------
	elsif(state=ControlReg_2) then
		Address<="11";
		Data<=X"B7";
```

```
      WRn<='0';
      dcnt<=(others=>'0');
      state<=delay4u_c2;
elsif(state=delay4u_c2) then
      if(to_integer(dcnt)<delay_4us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            WRn<='1';
            dcnt<=(others=>'0');
            state<=delay1u_c2;
      end if;
elsif(state=delay1u_c2) then
      if(to_integer(dcnt)<delay_1us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            Address<="10";
            -- LOW Byte
            Data<=X"99";
            WRn<='0';
            dcnt<=(others=>'0');
            state<=delay4u_t2L;
      end if;
elsif(state=delay4u_t2L) then
      if(to_integer(dcnt)<delay_4us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            WRn<='1';
            dcnt<=(others=>'0');
            state<=delay1u_t2L;
      end if;
elsif(state=delay1u_t2L) then
      if(to_integer(dcnt)<delay_1us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            -- HIGH Byte
            Data<=X"99";
            WRn<='0';
            dcnt<=(others=>'0');
            state<=delay4u_t2H;
      end if;
elsif(state=delay4u_t2H) then
      if(to_integer(dcnt)<delay_4us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            WRn<='1';
            dcnt<=(others=>'0');
            state<=delay1u_t2H;
      end if;
elsif(state=delay1u_t2H) then
      if(to_integer(dcnt)<delay_1us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            state<=idle;
      end if;
```

```
            elsif(state=idle) then
                  state<=state;
            else
                  state<=init;

            end if;
      end if;
end process;

end Behavioral;
```

برنامه‌ی دیگری از تایمر:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity timer is
port(
            data   :out  std_logic_vector(7 downto 0);
            address :out std_logic_vector(1 downto 0);
            gate      :out std_logic_vector(2 downto 0);
            rdn     :out std_logic;
            wrn     : out std_logic;
            rst     :in std_logic;
            clock : out std_logic;
            clkin   : in std_logic
      );
end timer;
architecture behavioral of timer is
      type show_array is array (0 to 8) of std_logic_vector (7 downto
0);
      type addr_array is array (0 to 2) of std_logic_vector (1 downto
0);
      --constant delay_4us  :integer:=200;-- 4 us;
      --constant delay_1us  :integer:=50;-- 1 us;
      constant delay_4us  :integer:=4000;--400;-- 4 us;
      constant delay_1us  :integer:=1000;--100;-- 1 us;
      type state is(subdevide,enable_wrn,delay,start,stop);
      signal table : show_array := (x"36",x"24",x"3c",  --
2.5Mhz/15396  --> 162.38 hz

      x"76",x"12",x"1e",  -- 2.5Mhz/7698   --> 324.7  hz

      x"B6",x"52",x"23"); -- 2.5Mhz/9042   --> 276.5  hz


      -- signal  address_i : addr_array:=("00","01","10");
      signal  address_i :show_array :=(x"03",x"00",x"00",

      x"03",x"01",x"01",

      x"03",x"02",x"02");
      signal  current:state;
      signal  clock_i :std_logic;
      signal  part : integer range 0 to 3;
      begin
      gate<="111"; -- enable all chanels
      process(clkin,rst)
            variable count,srq,q :integer:=0;
```

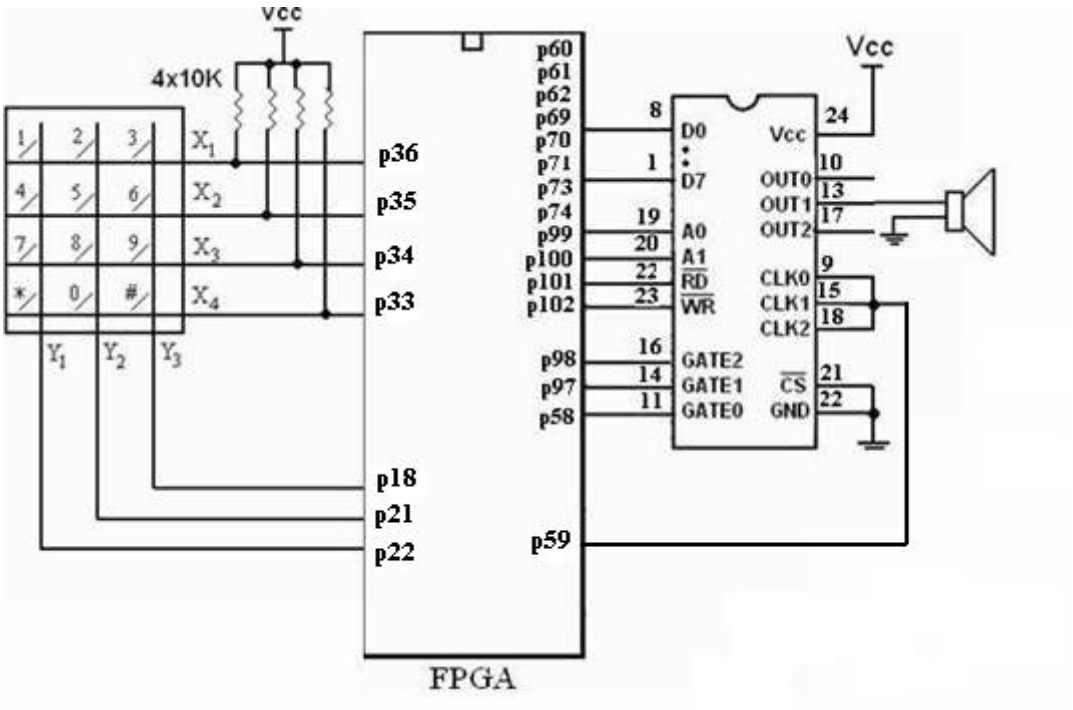
```
      begin
            if (rst='1') then -- asynchronous reset
                  q:=0;

                  wrn<='1';
                  rdn<='1';
                  current<=subdevide;
                  clock_i<='0';

            elsif(clkin'event and clkin='1')then
                  srq:=srq+1;                    -- prescaler
                  if (srq=2) then
                        clock_i<=not clock_i; -- 10Mhz /4  => 2.5Mhz
                     srq :=0;
                  end if;

                  case current is
                        when subdevide=>
                         -- select a chanel or the cw and set data
                              address <= address_i(q)(1 downto 0) ;
-- and programm all chanel

                              data <= table(q)(7 downto 0);
                              q:=q+1;
                              current <= enable_wrn;
                              if(q=10)then
                                    current     <=stop;
                              end if;
                        when stop=>
                        when enable_wrn =>-- after set address and
data write them with WR
                              if (part =0) then               --
enabling the Write pin
                                    wrn <= '0';
                                    count := delay_4us;
                                    current <= delay;
                                    part <= part +1;
                              elsif (part =1) then           --
Disabling the Write Pin
                                    wrn <= '1';
                                    count := delay_1us;
                                    current <= delay;
                                    part <= part +1;
                              else
                                    part <=0;
                                    current <= subdevide;
                              end if;
                        when delay=>                   -- use for make
correct delay between commands for 82c54
                              count := count-1;
                              if(count=0)then
                                    current<= enable_wrn;
                              end if;
                        when Start =>
                              rdn <= '1';
                  end case;
                  clock<=clock_i;
            end if;
      end process;
end behavioral;
```

۲۴- تولید موسیقی با استفاده از تراشه 8254

هدف اتصال تایمر 82c54 به FPGA می باشد تا اینکه یک فرکانس مرجع را بر تعدادی عدد تقسیم کند تا اینکه فرکانس نت های موسیقی یک اکتاو پدید آید . یک کی پد هم به FPGA متصل شده تا با فشردن کلید های آن (هفت کلید) FPGA با هر کلید یکی از اعدادی که باید برای تقسیم به تایمر بفرستد را انتخاب کند . در حقیقت با این مدار یک کیبورد موسیقی الکترونیکی ساده می سازیم.

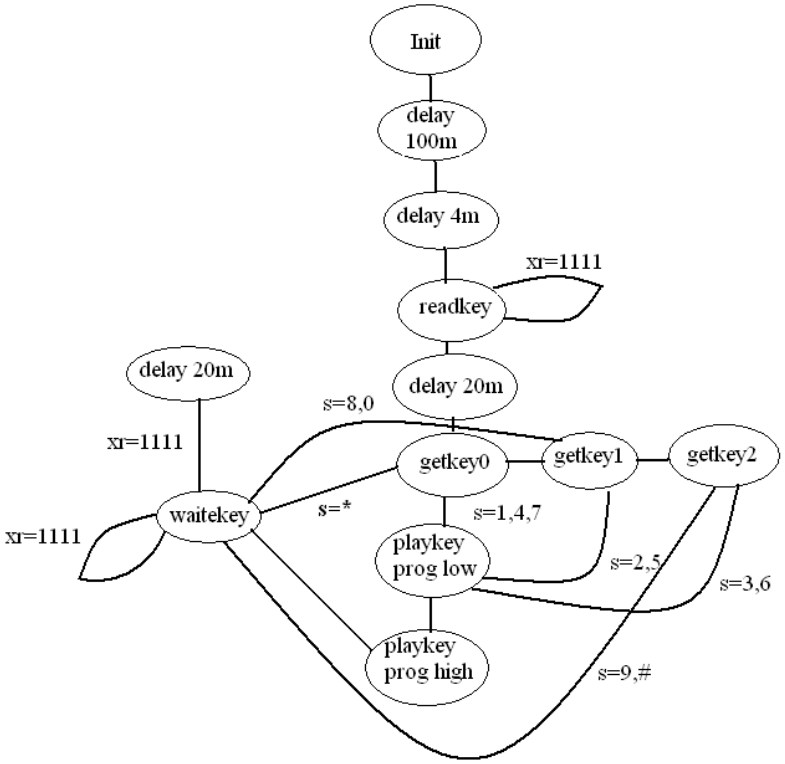
برای اینکه یک مدار ساده تولید نت های موسیقی داشته باشیم از تراشه 82c54 که یک تراشه تایمر قابل برنامه ریزی می باشد استفاده کرده ایم .

مدار شکل ۱ را ببندید .



شکل ۱

State for music:



برنامه موسیقی:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity music is
port(
                  -- 82C54 port
                  Data        : out std_logic_vector(7 downto 0);
                  Address     : out std_logic_vector(1 downto 0);
                  RDn         : out std_logic;
                  WRn         : out std_logic;
                  GATE        : out std_logic_vector(2 downto 0);
                  CLK         : out std_logic;
                  -- keypad
                  Xr                : in std_logic_vector(3 downto
0);
                  Yc                : out std_logic_vector(2 downto
0);
                  -- General
                  led         : out std_logic_vector(1 downto 0);
                  CLKin       : in std_logic
                  );
end music;
```

```
architecture Behavioral of music is
-- CLKin is 11.059 MHz
constant delay_100ms    : integer:=12000*100;
constant delay_20ms     : integer:=12000*20;
constant delay_4us: integer:=12*4;
constant delay_1us: integer:=4;
type state_type
is(init,delay100m_init,delay4u_init,readkey,delay20m_getkey,
      getkey_0,getkey_1,getkey_2,PlayKey,delay4u_playkeyL,delay1u_pla
ykey,delay4u_playkeyH,
      waitkey,delay20m_waitkey);

signal state:state_type:=init;
signal CLK_D4:std_logic:='0';
signal tcnt:std_logic:='0';
signal low:std_logic_vector(7 downto 0);
signal high:std_logic_vector(7 downto 0);
signal dcnt:unsigned(23 downto 0):=(others=>'0');

begin

-- divide CLKin on 4 for CLKs of 82C54
process(CLKin)
begin
      if(rising_edge(CLKin)) then
            tcnt<= not tcnt;
            if(tcnt='0') then
                  CLK_D4<= not CLK_D4;
            else
                  CLK_D4<= CLK_D4;
            end if;
      end if;
end process;
CLK<=CLK_D4;
RDn<='1';

process(CLKin)
begin
      if(rising_edge(CLKin)) then
            if(state=init) then
                  Yc<="000";
                  Data<=X"00";
                  Address<="11";
                  WRn<='1';
                  GATE<="111";
                  dcnt<=(others=>'0');
                  state<=delay100m_init;

            elsif(state=delay100m_init) then
                  if(to_integer(dcnt)<delay_100ms) then
                        dcnt<=dcnt+1;
                        state<=state;
                  else
                        Address<="11";
                        Data<=X"B6";
                        WRn<='0';
                        dcnt<=(others=>'0');
                        state<=delay4u_init;
                  end if;
```

```
elsif(state=delay4u_init) then
      if(to_integer(dcnt)<delay_4us) then
            dcnt<=dcnt+1;
            state<=state;
      else
            WRn<='1';
            GATE<="000";
            -- Counter 2
            Address<="10";
            state<=readkey;
      end if;

elsif(state=readkey) then
      Yc<="000";
      led<="11";
      if(Xr="1111") then
            state<=state;
      else
            dcnt<=(others=>'0');
            state<=delay20m_getkey;
      end if;

elsif(state=delay20m_getkey) then
      if(to_integer(dcnt)<delay_20ms) then
            dcnt<=dcnt+1;
            state<=state;
      else
            Yc<="110";
            state<=getkey_0;
      end if;

elsif(state=getkey_0) then
      led<="10";
      if(Xr="1110") then                        --s=1
            low<=X"C3"; high<=X"0D";
            state<=PlayKey;
            Yc<="000";
      elsif(Xr="1101") then                  --s=4
            low<=X"4F"; high<=X"0A";
            state<=PlayKey;
            Yc<="000";
      elsif(Xr="1011") then                      --s=7
            low<=X"4A"; high<=X"07";
            state<=PlayKey;
            Yc<="000";
      elsif(Xr="0111") then
            state<=waitKey;
            Yc<="000";
      else
            Yc<="101";
            state<=getkey_1;
      end if;

elsif(state=getkey_1) then
      led<="01";
      if(Xr="1110") then              --s=2
            low<=X"42"; high<=X"0C";
            state<=PlayKey;
            Yc<="000";
```

```
        elsif(Xr="1101") then                --s=5
              low<=X"2E"; high<=X"09";
              state<=PlayKey;
              Yc<="000";
        elsif(Xr="1011") then           --s=8
              state<=waitKey;
              Yc<="000";
        elsif(Xr="0111") then                --s=0
              state<=waitKey;
              Yc<="000";
        else
              Yc<="011";
              state<=getkey_2;
        end if;

  elsif(state=getkey_2) then
        led<="00";
        if(Xr="1110") then                   --s=3
              low<=X"EC"; high<=X"0A";
              state<=PlayKey;
              Yc<="000";
        elsif(Xr="1101") then           --s=6
              low<=X"2E"; high<=X"08";
              state<=PlayKey;
              Yc<="000";
        elsif(Xr="1011") then                --s=9
              state<=waitKey;
              Yc<="000";
        elsif(Xr="0111") then
              state<=waitKey;
              Yc<="000";
        else
              Yc<="000";
              state<=waitkey;
        end if;

  elsif(state=PlayKey) then
        GATE(2)<='1';
        -- low byte
        Data<=low;
        WRn<='0';
        dcnt<=(others=>'0');
        state<=delay4u_playkeyL;

  elsif(state=delay4u_playkeyL) then
        if(to_integer(dcnt)<delay_4us) then
              dcnt<=dcnt+1;
              state<=state;
        else
              WRn<='1';
              dcnt<=(others=>'0');
              state<=delay1u_playkey;
        end if;

  elsif(state=delay1u_playkey) then
        if(to_integer(dcnt)<delay_1us) then
              dcnt<=dcnt+1;
              state<=state;
        else
              -- high byte
```

```
                        Data<=high;
                        WRn<='0';
                        dcnt<=(others=>'0');
                        state<=delay4u_playkeyH;
                  end if;

            elsif(state=delay4u_playkeyH) then
                  if(to_integer(dcnt)<delay_4us) then
                        dcnt<=dcnt+1;
                        state<=state;
                  else
                        WRn<='1';
                        state<=waitkey;
                  end if;

            elsif(state=waitkey) then
                  if(Xr/="1111") then
                        state<=state;
                  else
                        dcnt<=(others=>'0');
                        state<=delay20m_waitkey;
                  end if;

            elsif(state=delay20m_waitkey) then
                  if(to_integer(dcnt)<delay_20ms) then
                        dcnt<=dcnt+1;
                        state<=state;
                  else
                        if(Xr/="1111") then
                              state<=waitkey;
                        else
                              GATE(2)<='0';
                              state<=readkey;
                        end if;
                  end if;

            else
                  state<=init;
            end if;
      end if;
end process;

end Behavioral;
```

برنامه‌ی دیگر موسیقی:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity music_pit is
port(
```

```
        data,test   :out  std_logic_vector(7 downto 0);
        address :out std_logic_vector(1 downto 0);
        gate      :out std_logic_vector(2 downto 0);
        rdn     :out std_logic;
        wrn     : out std_logic;
        rst     :in std_logic;
        clock : out std_logic;
        clkin   : in std_logic;
        row : out std_logic_vector(3 downto 0);
        column : in std_logic_vector(2 downto 0)

       );
end music_pit;

architecture Behavioral of music_pit is
   constant debounce_time : integer := 10000;--(10000)
   constant delay_4us  :integer:=40;--(200) 4 us;
   constant delay_1us  :integer:=10;--(50) 1 us;

   type show_array is array (0 to 20) of std_logic_vector (7 downto
0);
   type state is (Idel,subdevide,enable_wrn,delay,Play);

   signal debounce_flag,clock_i : std_logic;
   signal row_i,digit,column_encoded, row_encoded,t :
std_logic_vector (3 downto 0);
   signal column_before : std_logic_vector (2 downto 0);

   signal counter,Keys,flag,part : integer;
   signal current:state;
      signal r:integer;
   signal table : show_array := (
         x"B6",x"A6",x"4A", --5Mhz/19110.9582234453 --> 261.63 hz
do
         x"B6",x"81",x"42", --5Mhz/17025.9134402561 --> 293.67 hz
re
         x"B6",x"40",x"3B", --5Mhz/15168.5222825592 --> 329.63 hz
mi
         x"B6",x"ED",x"37", --5Mhz/14317.2121524497 --> 349.23 hz
fa
         x"B6",x"D3",x"31", --5Mhz/12755.1020408163 --> 392 hz
sol
         x"B6",x"63",x"2C", --5Mhz/11363.6363636364 --> 440 hz
la
         x"B6",x"8B",x"27");--5Mhz/10123.9167409087 --> 493.88 hz
si
   begin
      row_encoded <= "1000" when row_i = "1110" else
                     "1100" when row_i = "1101" else
                     "0000" when row_i = "1011" else
                     "0100" when row_i = "0111" ;
      column_encoded <= "0010" when column = "110" else
                        "0001" when column = "101" else
                        "0000" when column = "011"  ;
      Keys <= 0 when Digit = "1101" else
                  1 when Digit = "0000" else
                  2 when Digit = "0001" else
                  3 when Digit = "0010" else
                  4 when Digit = "0100" else
                  5 when Digit = "0101" else
```

```
                  6 when Digit = "0110" else
                  7 when Digit = "1000" else
                  8 when Digit = "1001" else
                  9 when Digit = "1010" else
                  10 when Digit = "1100" else
                  11 when Digit = "1110" ;

      process (clkin, rst)
            variable srq,count,i:integer :=0;
      Begin
            if (rst='1') then
                  counter <= 0;
                  gate<="111";
                  Address<="11";
                  --Activating channel 2 only.
                  wrn<='1';
                  rdn<='1';
                  current<=idel;--subdevide;
                  clock_i<='0';
                  flag <= 0;
                  part <=0;
                  t<="0111";
                  digit <= "1111";
                  row_i <= "1110";

                  debounce_flag <= '0';
            elsif(clkin'event and clkin='1')then
                  --clock out maker
--                srq:=srq+1;
--                if (srq=2) then -- 10Mhz /4  => 2.5Mhz
                        clock_i<=not clock_i;
--                      srq :=0;
--                end if;

                  -- debounce controler
                  if counter=0 then
                        Row_i<=Row_i(2 downto 0)& row_i(3);
                        debounce_flag<='0';
                        test(7 downto 4)<= row_i;
                        i:=0;
                        counter<=200000;
                  else
                        counter<=counter -1;
                  end if;

                  if (column /= "111" and debounce_flag = '0') then
                        counter<=4000000;
                        i:=i+1;
                        digit <= column_encoded + row_encoded;
                        if i>10 then
                              debounce_flag <= '1';
                              Current<=subdevide;
                              r<=0;
                              flag<=0;
                              test(3 downto 0)<= digit;
                        end if;
                  end if;

            case current is
                  when subdevide =>
```

```
                        --address <= "11";
--Channel
                        Gate<="111";
                        if ((keys >0) and (Keys<8))then
                                    if (r=0) then
                                          address <= "11";
                                    else
                                          address <= "10";
                                    end if;
                        data <= table((Keys-1)*3+r)(7 downto 0);

                              r<=r+1;
                              current <= enable_wrn;
                              if r=3 then
                                    current<=Play;
                                    Count:=5000000;
                              end if;
                        else
                              current<=Idel;
                        end if;
                  when enable_wrn =>
                        if (part =0) then  -- enabling the Write pin
                              wrn <= '0';
                              count := delay_4us;
                              current <= delay;
                              part <= 1;
                        elsif (part =1) then--Disabling the Write Pin
                              wrn <= '1';
                              count := delay_4us;
                              current <= delay;
                              part <= 2;
                        else
                              part <=0;
                              current <= subdevide;
                        end if;

                  when delay=>
                        count := count-1;
                              if(count=0)then
                              current<= enable_wrn;
                        end if;
                  when Play=>
                        if Count>0 then
                              Count:=Count-1;
                        else
                              current<=Idel;
                        end if;
                        rdn<='1';
                        wrn<='1';
                  when idel =>
                        Gate<="000";
                  when others =>
                        current<=Idel;
                  end case;
            end if;
      end process;

      clock<=clock_i;
      row <=row_i;
end behavioral;
```

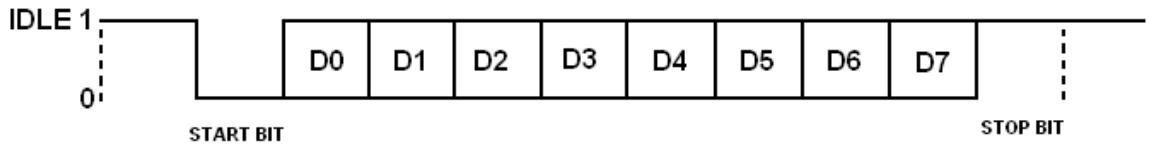
۲۵- ارتباط سریال توسط FPGA

اصول ارتباط سریال آسنکرون:

یک ارتباط دوجهته سریال آسنکرون نیاز به ۳ خط ارسال داده، دریافت داده و زمین مشترک (GND) دارد.

برای فرستادن داده‌ها خط ارسال داده را که در حالت بیکار (IDLE) '1' منطقی است به '0' منطقی تبدیل می‌کنیم گیرنده این عمل را شروع ارسال داده تعبیر کرده و آن را به عنوان START BIT می‌شناسد. سپس داده‌ها را یکی پس از دیگری می‌فرستیم و در پایان بیت توازن یا پریتی ارسال می‌شود، پایان فرستادن داده‌ها با '1' کردن خط ارسال داده به STOP BIT تعبیر می‌شود.

به طور مثال برای ارسال ۸ بیت داده بدون داشتن پریتی شکل موج خط ارسال در شکل ۱ نشان داده شده است.



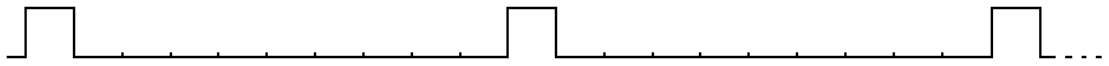
شکل ۱ پروتکل ارتباط سریال آسنکرون

نکته مهمی که در این مورد وجود دارد مدت زمان انجام کل این عمل است، تا بتوان داده‌ها را از هم تشخیص داد، برای این کار طبق قرار داد فرستنده و گیرنده با یک نرخ ثابت عمل ارسال و دریافت را انجام می‌دهند. (مثلاً با سرعت 1200 بیت در ثانیه (1200bps))

برنامه فرستنده از پایه send اطلاعات موجود در ورودی data_in (۸ بیت) را با نرخ ارسال 1200bps بدون پریتی و با STOP BIT = '1' (به طول یک بیت) ارسال می‌کند.

برای دیدن چگونگی انجام گرفتن این کار می‌توان خروجی send را به یک اسیلوسکپ متصل کرده و موج حاصل را بررسی کرد.

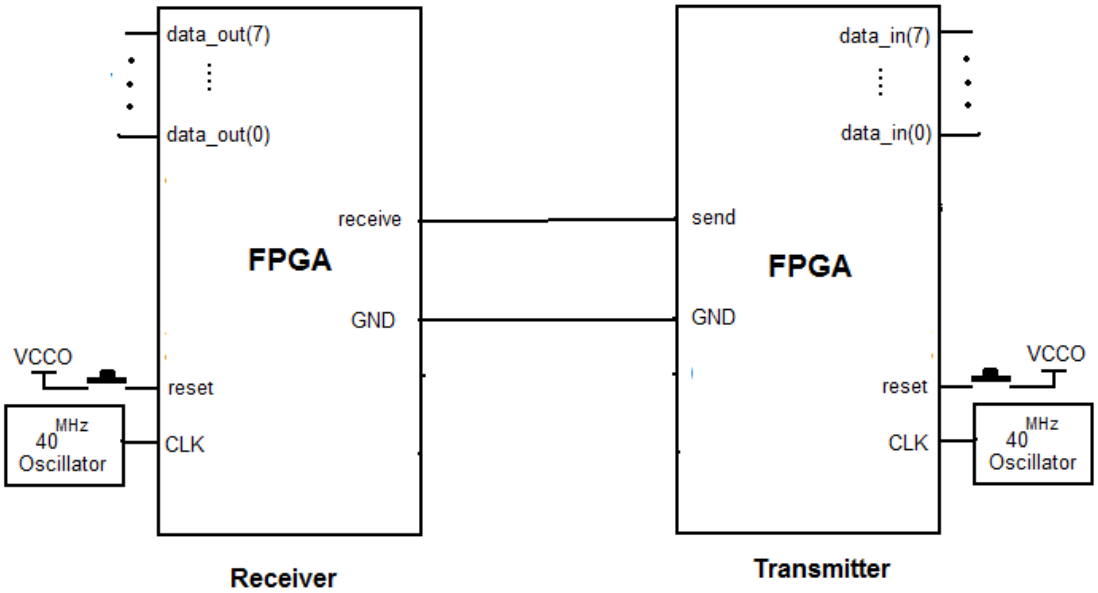
به طور مثال اگر به data_in مقدار 00H بدهید، شکل موج حاصل بصورت شکل ۲ در می‌آید.



شکل ۲ شکل موج خروجی send از FPGA برای ارسال 00H

با دادن مقادیر مختلف شکل موج حاصل را ارزیابی کنید.

برای ارتباط دو FPGA به هم و ارسال داده‌ها مطابق شکل ۳ به صورت سریال اتصال بین پایه send از FPGA فرستنده و پایه receive از FPGA گیرنده را با یک سیم بر قرار کنید و GND های دو FPGA را نیز به هم متصل کنید. اینک داده‌های داده شده به data_in از FPGA فرستنده را روی data_out از FPGA گیرنده مشاهده کنید.



شکل ۳ اتصال سریال دو FPGA به هم

برنامه FPGA فرستنده:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity serial_sender is
   Port ( clk : in std_logic;
          reset: in std_logic;
          data_in : in std_logic_vector (7 downto 0);
          send : out std_logic);
end serial_sender;

architecture Behavioral of serial_sender is
   constant bit_time : integer := 33333; --1200bps
   type state is (start, send_data, delay);
   signal current, caller : state;
   signal out_reg : std_logic_vector (9 downto 0);
begin
   process (clk, reset)
      variable i, count : integer;
   begin
      if reset = '1' then
         current <= start;
         send <= '1';
         i := 0;
      elsif clk = '1' and clk'event then
         case current is
         when start =>
            out_reg <= '1' & data_in & '0';
            current <= send_data;
            i := 0;
            send <= '1';
         when send_data =>
            if i < 10 then
               send <= out_reg (0);
               i := i + 1;
               caller <= send_data;
               count := bit_time;
               current <= delay;
               out_reg <= out_reg(0) & out_reg(9 downto 1);
            else
               out_reg <= '1' & data_in & '0';
               current <= send_data;

               i := 0;
            end if;
         when delay =>
```

```
            count := count - 1;
            if count = 0 then
               current <= caller;
            end if;
         when others =>
            current <= start;
         end case;
      end if;
   end process;
end Behavioral;
```

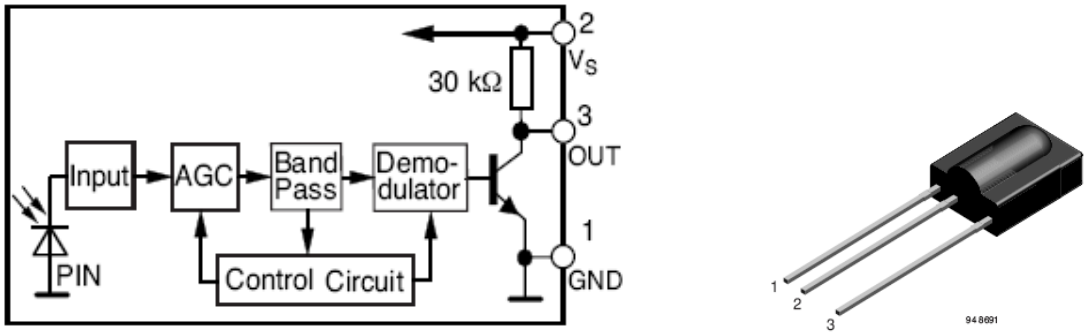
برنامه FPGA گیرنده:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity serial_receiver is
   Port ( clk : in std_logic;
          reset : in std_logic;
          receive : in std_logic;
          data_out : out std_logic_vector(7 downto 0));
end serial_receiver;

architecture Behavioral of serial_receiver is
   constant bit_time : integer := 33333; --1200bps
   constant max_delay : integer := bit_time + 1;
   type state is (start, receive_data, delay);
   signal current, caller : state;
   signal in_reg : std_logic_vector (9 downto 0);
begin
   process (clk, reset)
      variable i : integer;
      variable count : integer range 0 to max_delay;
      variable part : std_logic;
   begin
      if reset = '1' then
         current <= start;
         i := 0;
         part := '0';
      elsif clk = '1' and clk'event then
         case current is
         when start =>
            if part = '0' then
               if receive = '1' then
                  part := '1';
               end if;
            else
               if receive = '0' then
                  i := 0;
                  part := '0';
                  caller <= receive_data;
```

```
                count := bit_time / 2;

                current <= delay;
             end if;
          end if;
       when receive_data =>
          if i < 9 then
             i := i + 1;
             caller <= receive_data;
             count := bit_time;
             in_reg <= receive & in_reg (9 downto 1);
             current <= delay;
          else
             if receive = '1' and in_reg (1) = '0' then
                in_reg <= receive & in_reg (9 downto 1);
                data_out <= in_reg (9 downto 2);
                current <= start;
             else
                caller <= start;
                count := bit_time;
                current <= delay;
             end if;
          end if;
       when delay =>
          count := count - 1;
          if count = 0 then
             current <= caller;
           end if;
       when others =>
          current <= start;
       end case;
    end if;
 end process;
end Behavioral;
```

۲۶- انتقال اطلاعات بصورت مادون قرمز با استفاده از FPGA

در این مدار از سنسور TSOP1238 برای دریافت اشعه مادون قرمز استفاده شده است. (شکل ۱)

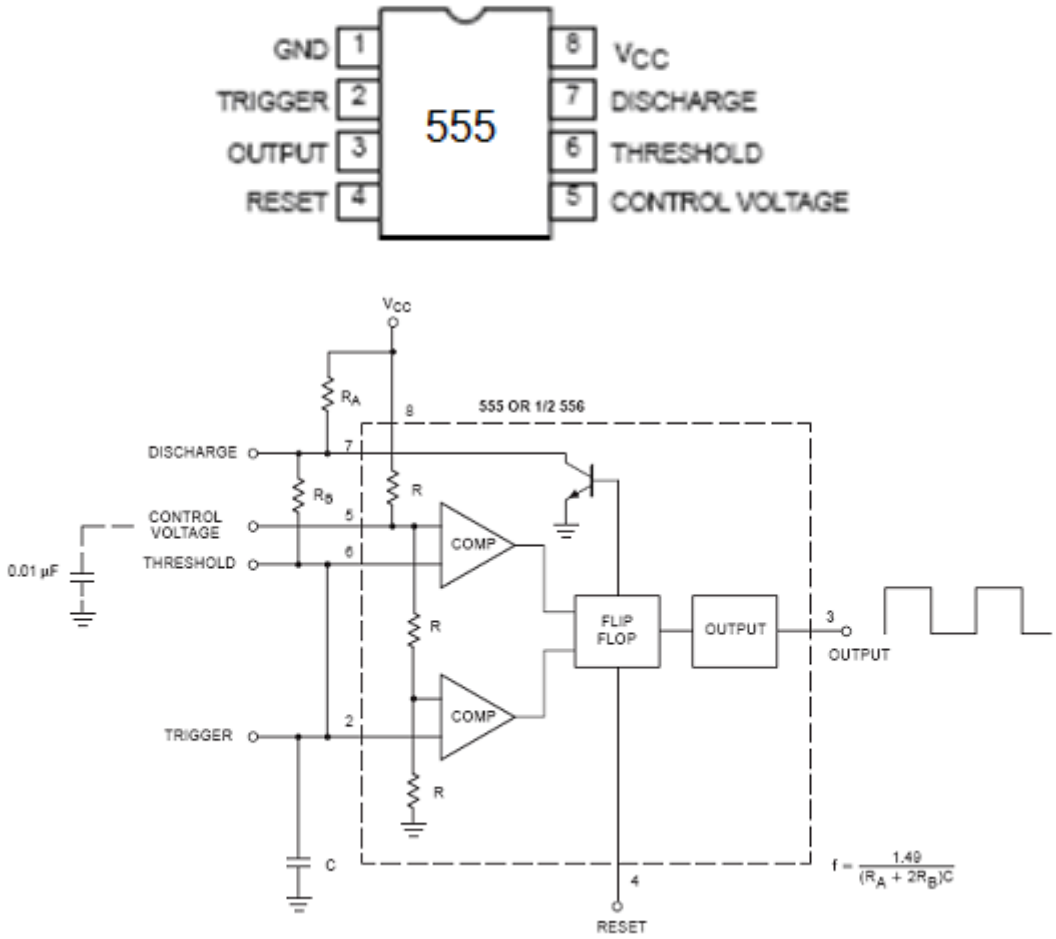


شکل ۱ پایه‌های TSOP1238 و بلاک دیاگرام داخلی آن

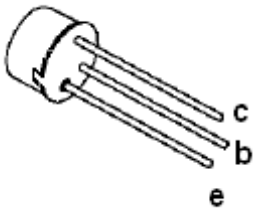
خروجی TSOP1238 با دریافت موجی با فرکانس 38^{KHz} فعال ('0') می‌شود بنابراین در قسمت فرستده از یک مولد پالس 38^{KHz} استفاده می‌کنیم. این عمل را با استفاده از تراشه 555 (شکل ۲) انجام می‌دهیم.

بدلیل اینکه برای فرستادن عدد '1' باید مدار مولد پالس غیر فعال شود و برای فرستادن عدد '0' این مولد باید پالس تولید کند. از یک ترانزیستورNPN مثل BC107 (شکل ۳) برایRESET کردن 555 استفاده شده است. هرگاه عدد '1' فرستاده شود ترانزیستور فعال شده و تراشه 555 غیر فعال می‌شود وچون پالسی ارسال نمی‌گردد، در طرف گیرنده، خروجی TSOP1238، '1' باقی می‌ماند. با فرستادن مقدار '0' ترانزیستور غیر فعال شده و تراشه شروع به تولید پالس 38^{KHz} می‌نماید، درنتیجه خروجی گیرنده (TSOP1238) '0' می‌شود.

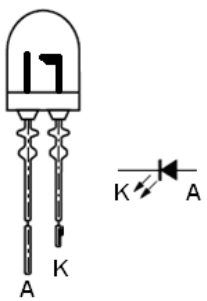
در قسمت فرستنده از یک دیود فرستنده مادون قرمز (شکل ۴) استفاده شده است. برد این دیود کم می‌باشد و برای تقویت اشعه خروجی IR می‌توان از یک ترانزیستور جهت افزایش طول برد موثر مدار استفاده کرد.



شکل ۲ پایه‌های تراشه 555 و بلاک دیاگرام داخلی آن

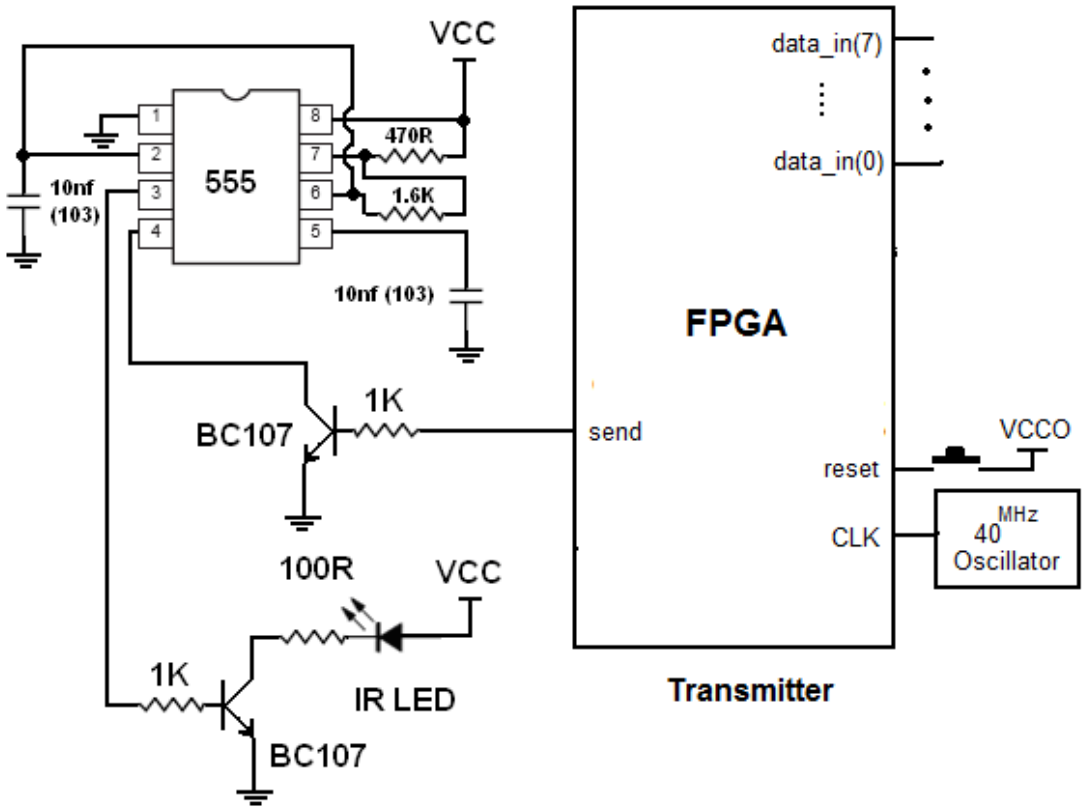


شکل ۳ پایه‌های ترانزیستور BC107



شکل ۴ شکل پایه‌های دیود فرستنده

مدار شکل ۵ اتصالات FPGA فرستنده را نشان می‌دهد.



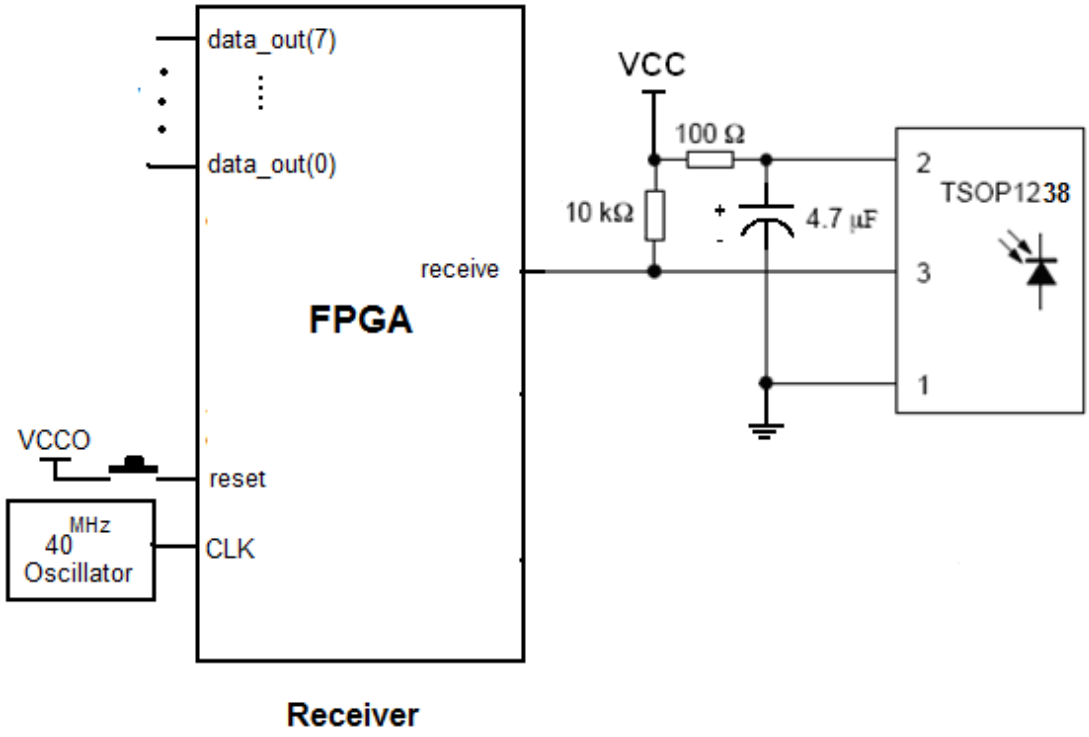
شکل ۵ مدار اتصال فرستنده مادون قرمز به FPGA

برنامه زیر از خروجی send اطلاعات موجود در data_in را با نرخ ارسال 1200bps بدون پریتی و با STOP BIT = '1' (به طول یک بیت) ارسال می‌کند.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity serial_sender is
   Port ( clk : in std_logic;
          reset: in std_logic;
          data_in : in std_logic_vector (7 downto 0);
          send : out std_logic);
end serial_sender;

architecture Behavioral of serial_sender is
   constant bit_time : integer := 33333; --1200bps
   type state is (start, send_data, delay);
   signal current, caller : state;
```

```
   signal out_reg : std_logic_vector (9 downto 0);
begin
   process (clk, reset)
      variable i, count : integer;
   begin
      if reset = '1' then
         current <= start;
         send <= '1';
         i := 0;
      elsif clk = '1' and clk'event then
         case current is
         when start =>
            out_reg <= '1' & data_in & '0';
            current <= send_data;
            i := 0;
            send <= '1';
         when send_data =>
            if i < 10 then
               send <= out_reg (0);
               i := i + 1;
               caller <= send_data;
               count := bit_time;
               current <= delay;
               out_reg <= out_reg(0) & out_reg(9 downto 1);
            else
               out_reg <= '1' & data_in & '0';
               current <= send_data;

               i := 0;
            end if;
         when delay =>
            count := count - 1;
            if count = 0 then
               current <= caller;
            end if;
         when others =>
            current <= start;
         end case;
      end if;
   end process;
end Behavioral;
```

مدار شکل ۶ اتصالات FPGA گیرنده را نشان می‌دهد.



شکل ۶ مدار اتصال گیرنده مادون قرمز به FPGA

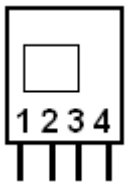
برنامه زیر هم مربوط به FPGA گیرنده می‌باشد که اطلاعات دریافت شده را روی data_out نشان می‌دهد.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity serial_receiver is
   Port ( clk : in std_logic;
          reset : in std_logic;
          receive : in std_logic;
          data_out : out std_logic_vector(7 downto 0));
end serial_receiver;

architecture Behavioral of serial_receiver is
   constant bit_time : integer := 33333; --1200bps
   constant max_delay : integer := bit_time + 1;
   type state is (start, receive_data, delay);
   signal current, caller : state;
   signal in_reg : std_logic_vector (9 downto 0);
begin
   process (clk, reset)
      variable i : integer;
      variable count : integer range 0 to max_delay;
```

```
      variable part : std_logic;
   begin
      if reset = '1' then
         current <= start;
         i := 0;
         part := '0';
      elsif clk = '1' and clk'event then
         case current is
         when start =>
            if part = '0' then
               if receive = '1' then
                  part := '1';
               end if;
            else
               if receive = '0' then
                  i := 0;
                  part := '0';
                  caller <= receive_data;
                  count := bit_time / 2;

                  current <= delay;
               end if;
            end if;
         when receive_data =>
            if i < 9 then
               i := i + 1;
               caller <= receive_data;
               count := bit_time;
               in_reg <= receive & in_reg (9 downto 1);
               current <= delay;
            else
               if receive = '1' and in_reg (1) = '0' then
                  in_reg <= receive & in_reg (9 downto 1);
                  data_out <= in_reg (9 downto 2);
                  current <= start;
               else
                  caller <= start;
                  count := bit_time;
                  current <= delay;
               end if;
            end if;
         when delay =>
            count := count - 1;
            if count = 0 then
               current <= caller;
             end if;
         when others =>
            current <= start;
         end case;
      end if;
   end process;
end Behavioral;
```

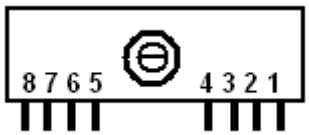
۲۷- انتقال اطلاعات بصورت رادیویی با استفاده از FPGA

دو پکیج مورد نظر TX-C1 (Hybird Transmitter) و RXD1 (Hybird Receiver) ساخت شـرکت Keymark می‌باشند. شکل‌های ۱ و ۲ پایه‌های این دو را نشان می‌دهند.



Pin	Connections
1	GND
2	DATA
3	VCC
4	ANT

شکل ۱ پایه‌های ماژول فرستنده TX-C1



Pin	Connections	Pin	Connections
1	ANT	5	VCC
2	GND	6	DATA
3	GND	7	DATA
4	VCC	8	GND

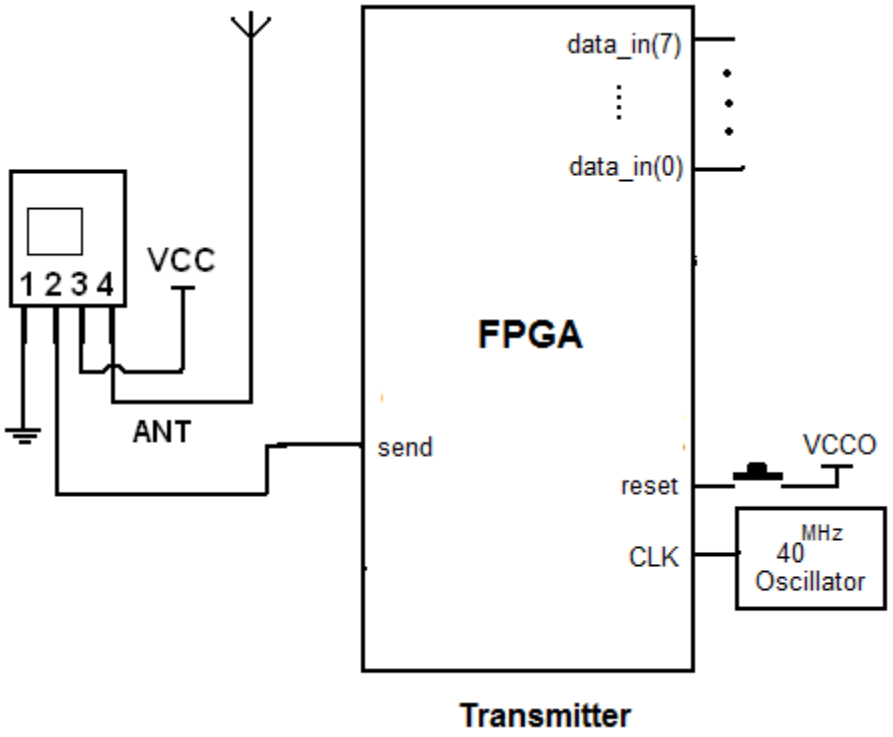
شکل ۲ پایه‌های ماژول گیرنده RXD1

برای این پکیج ها دو فرکانس کاری 315^{MHz} و 434^{MHz} موجود می‌باشد، برای هرکدام از فرکانس ها طول آنتن برابر مقادیر زیر است:

434^{MHz} : 17.2 cm

315^{MHz} : 22.6 cm

مدار شکل ۳ اتصالات FPGA فرستنده را نشان می‌دهد.



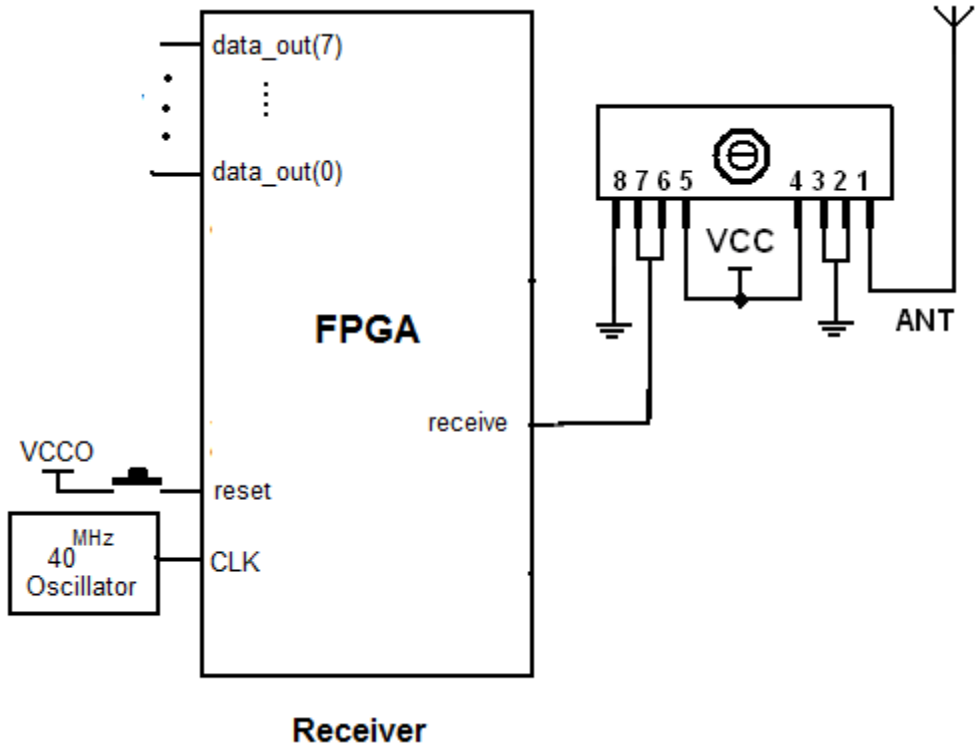
شکل ۳ مدار اتصال فرستنده رادیویی به FPGA

برنامه زیر از خروجی send اطلاعات موجود در data_in را با نرخ ارسال 1200bps بدون پریتی و با یک STOP BIT ارسال می‌کند.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity serial_sender is
   Port ( clk : in std_logic;
          reset: in std_logic;
          data_in : in std_logic_vector (7 downto 0);
          send : out std_logic);
end serial_sender;
```

```
architecture Behavioral of serial_sender is
   constant bit_time : integer := 33333; --1200bps
   type state is (start, send_data, delay);
   signal current, caller : state;
   signal out_reg : std_logic_vector (9 downto 0);
begin
   process (clk, reset)
      variable i, count : integer;
   begin
      if reset = '1' then
         current <= start;
         send <= '1';
         i := 0;
      elsif clk = '1' and clk'event then
         case current is
         when start =>
            out_reg <= '1' & data_in & '0';
            current <= send_data;
            i := 0;
            send <= '1';
         when send_data =>
            if i < 10 then
               send <= out_reg (0);
               i := i + 1;
               caller <= send_data;
               count := bit_time;
               current <= delay;
               out_reg <= out_reg(0) & out_reg(9 downto 1);
            else
               out_reg <= '1' & data_in & '0';
               current <= send_data;
               i := 0;
            end if;
         when delay =>
            count := count - 1;
            if count = 0 then
               current <= caller;
            end if;
         when others =>
            current <= start;
         end case;
      end if;
   end process;
end Behavioral;
```

مدار شکل ۴ اتصالات FPGA گیرنده را نشان می‌دهد. دقت کنید که فرکانس کاری پکیج های فرستنده و گیرنده یکی باشد.



شکل ۴ مدار اتصال گیرنده رادیویی به FPGA

برنامه زیر هم برای FPGA گیرنده نوشته شده است کـه اطلاعـات دریافـت شـده از فرسـتنده را در data_out نشان می‌دهد.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity serial_receiver is
   Port ( clk : in std_logic;
          reset : in std_logic;
          receive : in std_logic;
          data_out : out std_logic_vector(7 downto 0));
end serial_receiver;

architecture Behavioral of serial_receiver is
   constant bit_time : integer := 33333; --1200bps
   constant max_delay : integer := bit_time + 1;
   type state is (start, receive_data, delay);
   signal current, caller : state;
   signal in_reg : std_logic_vector (9 downto 0);
begin
   process (clk, reset)
```

```
      variable i : integer;
      variable count : integer range 0 to max_delay;
      variable part : std_logic;
   begin
      if reset = '1' then
         current <= start;
         i := 0;
         part := '0';
      elsif clk = '1' and clk'event then
         case current is
         when start =>
            if part = '0' then
               if receive = '1' then
                  part := '1';
               end if;
            else
               if receive = '0' then
                  i := 0;
                  part := '0';
                  caller <= receive_data;
                  count := bit_time / 2;

                  current <= delay;
               end if;
            end if;
         when receive_data =>
            if i < 9 then
               i := i + 1;
               caller <= receive_data;
               count := bit_time;
               in_reg <= receive & in_reg (9 downto 1);
               current <= delay;
            else
               if receive = '1' and in_reg (1) = '0' then
                  in_reg <= receive & in_reg (9 downto 1);
                  data_out <= in_reg (9 downto 2);
                  current <= start;
               else
                  caller <= start;
                  count := bit_time;
                  current <= delay;
               end if;
            end if;
         when delay =>
            count := count - 1;
            if count = 0 then
               current <= caller;
             end if;
         when others =>
            current <= start;
         end case;
      end if;
   end process;
end Behavioral;
```

۲۸- اتصال تراشه‌ی touch و adc آن

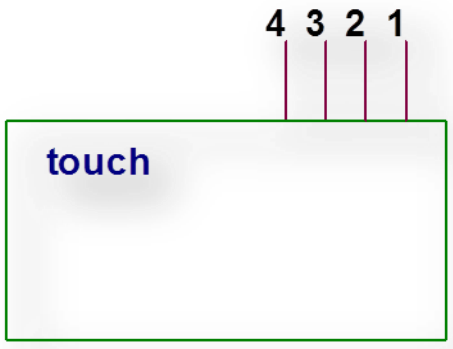
مقدمه:

میدانیم که Touch با استفاده از مقاومتهایی که در خود دارد دو مقدار مختلف به ما میدهد یکی برای مقدارx دیگری برای y که به صورت ولتاژ آنالوگ می باشد بنابراین برای این پروژه در ابتدا باید یک مدار تبدیل آنالوگ به دیجیتال داشته باشیم زیرا FPGA مقادیر آنالوگ را شناسایی نمیکند و سپس مقادیر دیجیتال بدست آمده را تبدیل به چهار مقدار column, segment,page و Data تبدیل کنیم که بتوانیم نقطه متناظر فشرده شده در Touch را در GLCD نمایش دهیم.

برنامه ماشامل State های مختلفی می باشد که هر کدام وظیفه منحصر به خود را دارد که برای هر کدام در خود برنامه کامنت گذاری شده و توضیح داده شده است.

شکل ظاهری صفحه لمسی و تشریح پایه های آن:

تاچ اسکرین مقاومتی(Touch Screen)



جدول زیر مربوط به محاسبه ولتاژ خروجی x و خروجی y در صفحه لمسی می باشد.

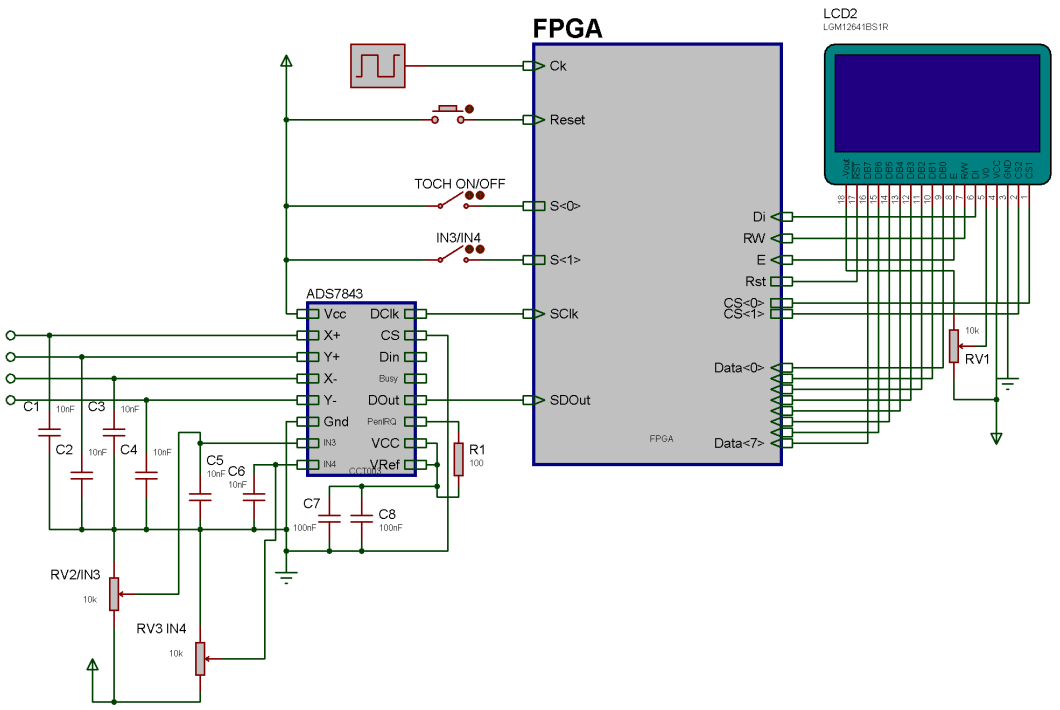
pin_1	pin_2	pin_3	pin_4
z	GND	v_x	v_{cc}
v_{cc}	v_y	GND	z

همانطور که در جدول فوق مشاهده می شود هنگامیکه پایه شماره ۲ تاچ را به GND و پایه شماره ۴ آنرا به Vcc متصل کنیم از پایه شماره ۳ ولتاژی بین 0 تا 5 ولت خارج می شود که این ولتاژ در اثر فشار آوردن به تاچ در انحنای محور x ها بوجود می آید.همچنین هنگامیکه پایه ۳ تاچ را به GND و پایه ۱ تاچ را به Vcc متصل میکنیم با فشار آوردن به تاچ در امتداد محور Y از پایه شماره ۲ تاچ ولتاژ مورد نظر ایجاد می شود.

حال قبل اینکه تاچ را به مدار اعمال کنیم ،میتوانیم با استفاده از آنچه که در قسمت بالا ذکر شد قطعه را تست کنیم و از سالم بودن آن اطمینان حاصل نماییم.

در هر دو طرف تاچ مقاومتی ای که در این مدار استفاده میکنیم از یک لایه پلاستیکی پوشیده شده که توصیه اکید میشود این لایه ها را از تاچ جدا ننمایید،چون باعث میشود تاچ حالت ولتاژ دهی مناسب خود را از دست بدهد و در مدار نویز ایجاد کند،لذا خروجی مورد نظر نیز از بین خواهد رفت و نقاطی علاوه بر آنچه که ما ایجاد می کنیم در صفحه نمایش ایجاد می کند.

شکل مداری:



کد برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity touch is
   Port ( data : out std_logic_vector(7 downto 0);--<= GLCD
          di : out std_logic;--<= GLCD
          rw : out std_logic;--<= GLCD
          e : out std_logic;--<= GLCD
          rst : out std_logic;--<= GLCD
          clk : in std_logic;--<= system clock
          reset : in std_logic;--<= system reset
                   sw: in std_logic_vector(1 downto 0);
--          x_select : out std_logic;--<= GLCD
--          y_select : out std_logic;--<= GLCD
          Cs : out std_logic_vector(1 downto 0);--<= GLCD
 ---ADS 7843 Interface ------------
                   SClk : out std_logic;-- Serial Clock for ADS
                   SDIn : out std_logic; -- Serial Data inpout for ADS
                   SDOut: in std_logic; -- Serial Data outpout from ADS
--                 Sw   : in std_logic;
          --wr_adc_x,rd_adc_x:out std_logic;--<= Read X & Y
          --intr_x : in std_logic;--<= Read X & Y
          --data_adc_x :in std_logic_vector(7 downto 0);--<= Read X & Y
          --wr_adc_y,rd_adc_y:out std_logic;--<= Read X & Y
          --intr_y : in std_logic;--<= Read X & Y
          --data_adc_y :in std_logic_vector(7 downto 0);--<= Read X & Y
------- Leds for Check
          Leds : out std_logic_vector(7 downto 0)--<= For Check State's
          );
end touch;
architecture Behavioral of touch is
   signal page : std_logic_vector (7 downto 0);  --jame B8H + y_digital (7
downto 5)
   signal tpage:std_logic_vector(2 downto 0);
      signal column,tcol : std_logic_vector(7 downto 0); --jame 40H +
x_digital(6 downto 1)
   signal DePixel,Out_Pixel : std_logic_vector(7 downto 0);  --decod shode
y_digital(4 downto 2)
      Signal Pixel: std_logic_vector(2 downto 0);
   constant Delay_for_e : integer := 25;  --23 * 19 NS = 450 NS  baraye
eijade labe
   constant delay_between_data : integer := 500; --baraye eijade delay
monaseb ta data roye khat gharar girad
   type Main_state is
(Init,Pattern,Clear_GLcd,Select_Mode,Read_IN3,Read_IN4, READ_x, READ_Y,
Calc,Make_Adr, WRITE_Ram, Set_Page,Set_Column,Write_Pixel); -- main state
   type Glcd_state is (None,Enable, Write); --write  to GLCD
   signal State : Main_state;
      Signal GLCD  : GLCD_State;
      signal iData :std_logic_vector(7 downto 0);
--------expalin RAM_2--------------
  --type mem_array_2 is array (0 to 7,0 to 127) of std_logic_vector(7
downto 0);   -- must be 1024
  --signal ram : mem_array_2;
--------explain RAM_1--------------
```

```
       signal C_Byte : std_logic_vector(7 downto 0);
       signal ADS_Data : std_logic_vector(11 downto 0);
       signal Ads_Sum : std_logic_vector(15 downto 0);
       Signal X,Y,y1,y2,y4,y8,yt: Std_Logic_vector(7 downto 0);
       Signal ADS_Enable: std_logic:='0';
       Signal xt,x1,x2,x4,x8:std_logic_vector(11 downto 0);
begin
   process (clk, reset)
     variable i,c,CIndx,BIndx,DIndx,Delay,Part, count,iPage,iCol :
integer:=0;

    begin
          if reset = '1' then
           di <= '0';
           rw <= '0';
           e <= '0';
              c:=0;
           Cs <= "11";
           rst <= '1';
           i := 0;
              Part:=0;
           page <= "00000000";
        leds<="11111111";
              cIndx:=0;
              BIndx:=1;
              DIndx:=0;
              C_Byte<="10010111";
              State<=Init;
              GLCD<=None;
              Delay:=0;
              iPage:=0;
              iCol:=0;
         elsif clk = '1' and clk'event then
                   if Delay>0 then
                      delay:=Delay-1;
                   else
                         case GLcd is
                                   when Enable=>
                                      Delay:=Delay_For_E;
                                      E<='1';
                                      GLCD<= Write;
                                   when Write=>
                                      Delay:=delay_between_data;
                                      E<='0';
                                      GLCD <= None;
                                   when none=>
                                        case state is
                                                     when Init =>
--
      if Part=0 then
--
      if iPage<8 then
--
         if iCol<127 then
--
            Ram(iPage,iCol)<="00000000";
--
                  iCol:=iCol+1;
--
            else
```

```
--
               iPage:=ipage+1;
--
                  iCol:=0;
--
            end if;
--
      else
--
            i:=0;
--
            part:=1;
--
      end if;
--
      else
      part:=0;
                                                      data <=
"00111111";
                                                      di <= '0';
      i:=0;
      Page<="00000000";
      Glcd<=Enable;
      State<=Pattern;
--
      end if;
                                                  when Pattern=>
      cs<="11";
      if i = 0 then
      data <= "10111000" + page;
      di <= '0';
      glcd <= Enable;
                                                                i
:= i + 1;
                                                         elsif i
= 1 then
      data <= "01000000";
      di <= '0';
      glcd <= enable;
      i := i + 1;
      elsif i < 66 then
                  data <= "10101010";
                  di <= '1';
```

```
                glcd <= enable;
                i := i + 1;
            else
                    page <= page + 1;
                i := 0;
                if page = 7 then
                State <= clear_GLCD;
                        delay:=100000000;
                end if;
                                                        end if;
                                                when Clear_GLCD=>
                                                        if i =
0 then
      data <= "10111000" + page;
      di <= '0';
      glcd <= Enable;
                                                                i
:= i + 1;
                                                        elsif i
= 1 then
      data <= "01000000";
      di <= '0';
      glcd <= enable;
      i := i + 1;
      elsif i < 66 then
                data <= "00000000";
                di <= '1';
                glcd <= enable;
                i := i + 1;
            else
                    page <= page + 1;
                i := 0;
                if page = 7 then
                                State <= Select_Mode;
```

```
                end if;
                                                        end if;
                                                    when Select_Mode =>
      Part:=0;
      i:=0;
      CS<="00";
      Count:=32;
      ADS_Enable<='1';
      Ads_Sum<="0000000000000000";
      if Sw(1)='0' then
      State <= read_x;
      C_Byte<="10010011";
      elsif sw(0)='0' then
            State <= read_IN3;
            C_Byte<="10100111";
      else
            State <= read_IN4;
            C_Byte<="11100111";
      end if;
Delay:=5000000;
                                                    when Read_IN3=>
                                                            if
ADS_Enable='0' then
                                                                    if
Ads_Data(11 downto 1)>"0010001000" then
leds<="11111111";
                                                                    elsif
ads_Data(11 downto 1)<="0000001000" then
Leds<="00000000";
                                                                    else
      leds<=Ads_Data(8 downto 1)-"00001000";
                                                                    end
if;
State<=Select_Mode;
                                                            end if    ;
                                                    When Read_IN4=>
                                                            if
ADS_Enable='0' then
```

```
leds<=Ads_Data(10 downto 3);
State<=Select_Mode;
                                                        end if;
                                                  when Read_X =>
                                                        if
Ads_Enable='0' then
      if part=0 then
      Xt<=ADS_sum(11 downto 0)- "00010110000";
      --leds<=Ads_sum(11 downto 4)- "00001001";
      Part:=1;
      elsif Part=1 then
            x1<="00000"&xt(11 downto 5);
            x2<="0000"&xt(11 downto 4);
            x4<="000"&xt(11 downto 3);
            x8<="00"&xt(11 downto 2);
            Part:=2;
       elsif Part=2 then
                        Xt    <= Xt+X8;
                        Part:=3;
             elsif Part=3 then
                  --leds<=xt(11 downto 4);
                  if Xt>"011111110000" then
                     X<="01111111";
                  elsif Xt<"000000001111" then
                      X<="00000000";
                        else
                     X<=Xt(11 downto 4);
             end if;
                   part:=0;
                   C_Byte<="11010011";
                  ADS_Enable<='1';
                  Ads_Sum<="0000000000000000";
```

```
                Delay:=10;
                State<=Read_Y;
                Count:=32;
      end if;
                                                              end if;
                                                  when Read_Y =>
                                                              if
Ads_Enable='0' then
      if Part=0 then
            Yt<= ADS_Sum(11 downto  4)- "00011010";
            --leds<=Ads_Sum(11 downto 4)- "00010011";
            Part:=1;
      elsif Part=1 then
            y1<="00000"&yt(7 downto 5);
            y2<="0000"&yt(7 downto 4);
            y4<="000"&yt(7 downto 3);
            y8<="00"&yt(7 downto 2);
            Part:=2;
       elsif Part=2 then
                        yt    <= yt+y8+y8+y8;
                        Part:=3;
             elsif Part=3 then
--
                  if yt>"11110000" then
--
                     y<="00000000";
--
                  els
                 if yt>"01111110" then
                     y<="01111110";
                  elsif yt<"00000000" then
                      y<="00000000";
                        else
                     y<=yt;
                  end if;
```

```
                    part:=0;
                   C_Byte<="10010011";
                   State<=Calc;
      end if;
                                                              end if;
                                                        when Calc=>
                                                             if X<64 then
CS<="01";
Column<=64+x;
TCol<=x+64;
                                                                   Else
      Cs<="10";
      column<=x;
      Tcol<=x;
                                                              end if;
      leds<=y;
      Page<="10111"& Y(6 downto 4);
      TPage<=y(6 downto 4);
      Pixel<=Y(3 downto 1);
      part:=0;
      State<=Make_adr;
                                                        when make_adr=>
--                                                             if part=0
then
--
part:=1;
--                                                                   if
TPage>0 then
--
Tpage<=TPage-1;
--
iPage:=iPage+1;
--
Part:=0;
--                                                        end if;
--                                              if TCol>0 then
--
TCol<=TCol-1;
--
iCol:=iCol+1;
--
Part:=0;
--                                                   end if;
--                                                              else
```

```
--
      Part:=0;

      State<=Write_Ram;
--                                                              end if;
                                                          when write_ram=>
--
      if Part=0 then
--
      Ram(iPage,iCol)<=Ram(iPage,iCol)or dePixel;
--
      Part:=1;
--
      else
--
      Out_Pixel<=Ram(iPage,iCol);
--
      State<= Set_Page;
--
      part:=0;
--
      end if;
                                                                        -
--/*
--                                                                 else

      Out_Pixel<=depixel;
      State<= Set_Page;
      part:=0;
--                                                                 end if;
                                                                   ---*/
                                                          when Set_page=>
                                                             rw <= '0';
                                                    data <= page;
                                                    di <= '0';
                                                    state <= Set_Column;
                                                    GLCD <= Enable;
                                                          when Set_column=>
                                                    data <= column;
                                                    di <= '0';
                                                    state <= write_Pixel;
                                                    GLCD <= Enable;
                                                          when Write_pixel=>
                                                          data <= out_Pixel;
                                                    di <= '1';
                                                    GLCD <= Enable;

      State<=Select_Mode;
                                              when others=>
      State<=Select_Mode;
                                                end case;
                                 end case;
                           end if;
                           if ADS_Enable='1' then
                                c:=c+1;
                                if Cindx<8 then
                                      if C=2 then
                                if BIndx=1 then  -- Send Control Byte To ADS
                                                  SDIn<= C_Byte(7-Cindx);
                                            end if;
```

```
                              end if;
                              if c=200 then
                                    -- Read 12 Bit Data from ADS
                                    if Bindx>1 then
                                          if DIndx<12 then
                                             ADs_Data(11-DIndx)<=SDOut;
                                                Dindx:=Dindx+1;
                                          end if;
                                    end if;
                              end if;
                              -- Make Clock out For ADS
                              if c<200 then
                                    SClk<='0';
                                    elsif c<400 then
                                          SClk<='1';
                                          else
                                                C:=0;
                                                SClk<='0';
                                                CIndx:=CIndx+1;
                              end if;
                        else
                        SdIn<='0';
                              if C=1600 then
                                 Cindx:=0;
                                    if BIndx=3 then
                                       BIndx:=1;
                                          CIndx:=0;
                                          DIndx:=0;
                                          if Count>0 then
                                                if Count<=16 then

      Ads_Sum<=Ads_Sum+"0000"&Ads_Data;
                                                end if;
                                             count:=count-1;
                                          else
                                                Ads_Enable<='0';
                                                delay:=100;
                                          end if;
                                    else
                                       BIndx:=BIndx+1;
                                    end if;
                                    C:=0;
                              end if;
                        end if;

                  end if;

       end if;
    end process;
Depixel<="00000001" when Pixel="000"
      else  "00000010" when Pixel="001"
      else  "00000100" when Pixel="010"
      else  "00001000" when Pixel="011"
      else  "00010000" when Pixel="100"
      else  "00100000" when Pixel="101"
      else  "01000000" when Pixel="110"
      else  "10000000" when Pixel="111";
end Behavioral;
```

۲۹- اتصال مبدل دیجیتال به آنالوگ AD7564

تراشه AD7564 شامل ۴ مبدل دیجیتال به آنالوگ ۱۲ بیتی می باشد که برای هر یک از مبدل ها ترمینال های VREF, IOUT1, IOUT2و RFB بصورت مجزا وجود دارد و با ولتاژ ۳.۳ ولت یا ۵ ولت کار می کند. این تراشه بـه وسـیله یـک ارتباط سریال قابل برنامه ریزی است . برنامه زیر برای تولید موج مربعی، دندان اره‌ای، مثلثی و سینوسی طراحی شده است. خروجی مدار را به اسیلوسکپ وصل کنید و نتیجه را مشاهده نمایید.

انتخاب نوع موج خروجی به وسیله عددی که به wave می‌دهید، انجام می‌شود:

wave = 00 موج خروجی مربعی

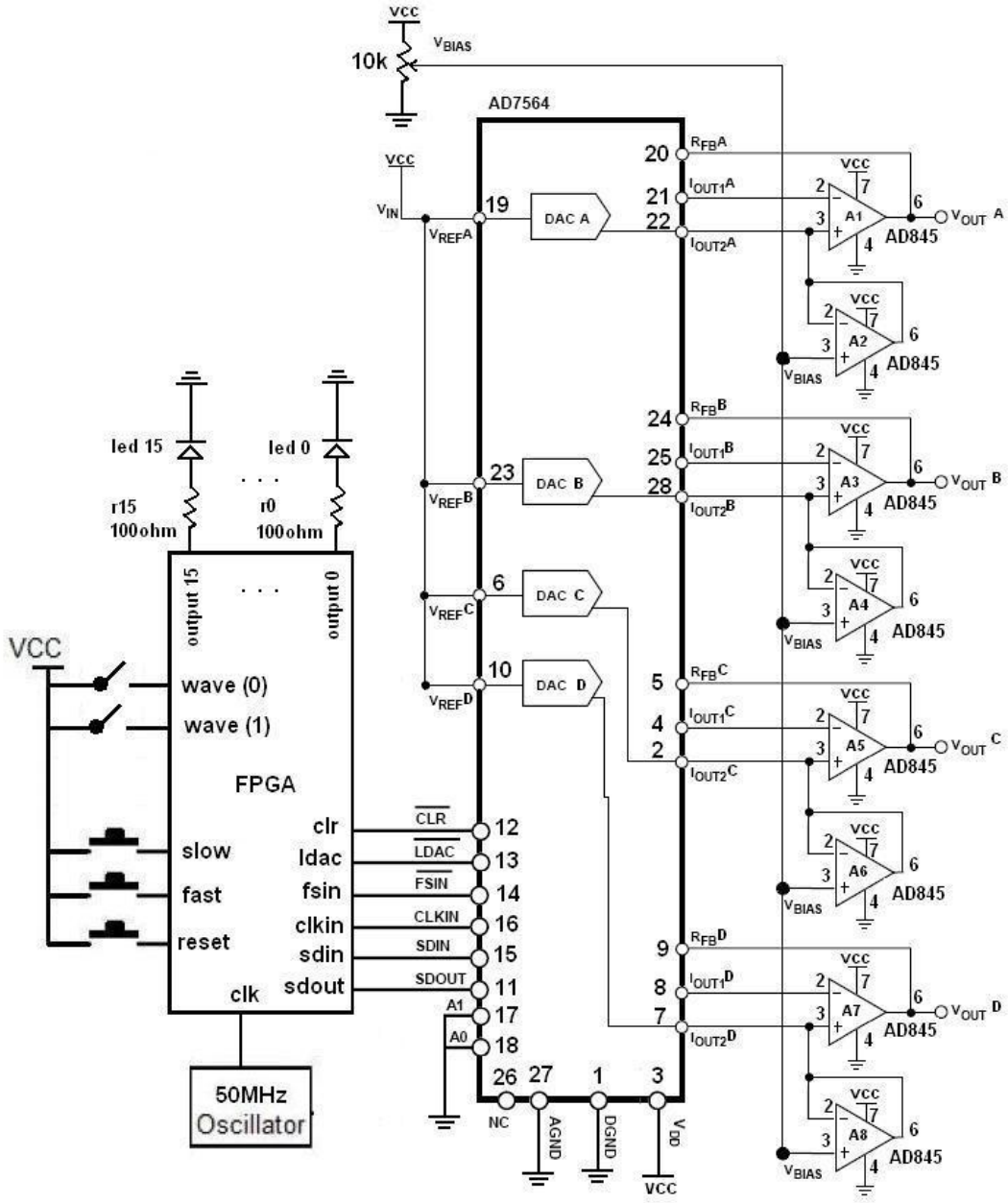
wave = 01 موج خروجی دندان اره‌ای

wave = 10 موج خروجی مثلثی

wave = 11 موج خروجی سینوسی

فرکانس هم بوسیله کلید های fast و slow قابل تنظیم است

شکل مدار به صروت زیر می باشد



کد برنامه :

```
library IEEE;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity ad7564 is
    Port ( clk:       in std_logic;
           reset:     in std_logic;
                  slow:     in std_logic;
                  fast:     in std_logic;
                  --data : in std_logic_vector(11 downto 0);
                  wave : in std_logic_vector (1 downto 0 );
```

```
                --ds : in std_logic_vector (1 downto 0 );
                output : out std_logic_vector(15 downto 0);
        clkin :    out std_logic;
        clr : out std_logic;
        ldac :   out std_logic;
        fsin :   out std_logic;
                sdin: out std_logic;
        sdout: in std_logic);
end ad7564;

architecture Behavioral of ad7564 is
type two_dim is array (0 to 179) of std_logic_vector (11 downto 0);   --
jadvale zarayebe sinusi

constant table : two_dim := (
"100000000000","100001000111","100010001110","100011010110","100100011100
","100101100011","100110101001","100111101111",
"101000110100","101001111000","101010111100","101011111111","101101000000
","101110000001","101111000001","101111111111",
"110000111101","110001111000","110010110011","110011101100","110100100100
","110101011010","110110001110","110111000000",
"110111110001","111000100000","111001001101","111001111000","111010100001
","111011001000","111011101101","111100001111",
"111100110000","111101001110","111101101010","111110000100","111110011011
","111110110000","111111000010","111111010010",
"111111100000","111111101011","111111110100","111111111010","111111111110
","111111111111","111111111110","111111111010",
"111111110100","111111101011","111111100000","111111010010","111111000010
","111110110000","111110011011","111110000100",
"111101101010","111101001110","111100110000","111100001111","111011101101
","111011001000","111010100001","111001111000",
"111001001101","111000100000","110111110001","110111000000","110110001110
","110101011010","110100100100","110011101100",
"110010110011","110001111000","110000111101","101111111111","101111000001
","101110000001","101101000000","101011111111",
"101010111100","101001111000","101000110100","100111101111","100110101001
","100101100011","100100011100","100011010110",
"100010001110","100001000111","100000000000","011110111000","011101110001
","011100101001","011011100011","011010011100",
"011001010110","011000010000","010111001011","010110000111","010101000011
","010100000000","010010111111","010001111110",
"010000111110","010000000000","001111000010","001110000111","001101001100
","001100010011","001011011011","001010100101",
"001001110001","001000111111","001000001110","000111011111","000110110010
","000110000111","000101011110","000100110111",
"000100010010","000011110000","000011001111","000010110001","000010010101
","000001111011","000001100100","000001001111",
"000000111101","000000101101","000000011111","000000010100","000000001011
","000000000101","000000000001","000000000000",
"000000000001","000000000101","000000001011","000000010100","000000011111
","000000101101","000000111101","000001001111",
"000001100100","000001111011","000010010101","000010110001","000011001111
","000011110000","000100010010","000100110111",
"000101011110","000110000111","000110110010","000111011111","001000001110
","001000111111","001001110001","001010100101",
"001011011011","001100010011","001101001100","001110000111","001111000010
","010000000000","010000111110","010001111110",
"010010111111","010100000000","010101000011","010110000111","010111001011
","011000010000","011001010110","011010011100",
"011011100011","011100101001","011101110001","011110111000");
```

```
type state is (square, saw, triangle, sine, delay);
--signal data : std_logic_vector(11 downto 0);
signal current : state;
signal data_i : std_logic_vector (11 downto 0);
signal data :  std_logic_vector(11 downto 0); ----------------------------
-
signal ds :  std_logic_vector (1 downto 0 );-----------------------------
begin


process(clk,reset)
variable c,c1,i, j : integer;
    variable down : std_logic;
    variable count,period : std_logic_vector (15 downto
0):="0000000100000000";
variable mode,count1,sending : integer;
begin

      if (reset='1') then

            ldac<='1';
            clr<='1';
            c:=0;
            c1:=0;
            clkin<='0';
            sending:=0;
            data<="000000000000";--------------------
            ds<="00";--------------------


            current <= square;
            data_i <= "000000000000";
            i := 0;
            j := 0;
            down := '0';
      elsif(clk='1' and clk'event) then
                        if slow='1' then         -- kelid slow baraye
kaheshe frequecy
                c:=c+1;
                if c>=2500000 then          -- kelid slow baraye kaheshe
frequecy
                    c:=0;
                    if period<"1000000000000000" then
                        period:=std_logic_vector (unsigned(period) sll
1); --shifte chap baraye afzayeshe ta'khir => kaheshe frequency
                    end if;
                end if;
            end if;
            if fast='1' then              -- kelid fast baraye afzayeshe
frequecy
                c:=c+1;
                if c>=2500000 then          -- ta'khire nim saniye
                    c:=0;
                    if period>"0000000000000001" then
                        period:=std_logic_vector (unsigned(period) srl
1); --shifte rast baraye kaheshe ta'khir => afzayeshe frequency
                    end if;
                end if;
            end if;
```

```
            c1:=c1+1;
            if(c1=4)then
                  c1:=0;
                  if sending<=64 then
                  sending:=sending+1;
                  end if;


            case sending is
                  when 0=>
                        ldac<='1'; -- baryae shorue ersal payeh
ldac,clr,bayad gheire fa'al bashand
                        clr<='1';
                        clkin<='1';
                        fsin<='1';-- baryae shorue ersal yek labe paeen
ravande roye payeh fsin bayad emal shavad
                        sdin<=data(11);
                  when 1=>
                        clkin<='1';
                        fsin<='0';-- baryae shorue ersal yek labe paeen
ravande roye fsin bayad emal shavad
                        sdin<=data(11); -- bit 11(MSB) roye payeh sdin
migozarim va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 2=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 3=>
                        clkin<='1';
                        sdin<=data(10);-- bit 10 roye payeh sdin
migozarim va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 4=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 5=>
                        clkin<='1';
                        sdin<=data(9);-- bit 9 roye payeh sdin migozarim
va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 6=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 7=>
                        clkin<='1';
                        sdin<=data(8);-- bit 8 roye payeh sdin migozarim
va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 8=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 9=>
                        clkin<='1';
                        sdin<=data(7);-- bit 7 roye payeh sdin migozarim
va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 10=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 11=>
                        clkin<='1';
                        sdin<=data(6);-- bit 6 roye payeh sdin migozarim
va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 12=>
```

```
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 13=>
                        clkin<='1';
                        sdin<=data(5);-- bit 5 roye payeh sdin migozarim
va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 14=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 15=>
                        clkin<='1';
                        sdin<=data(4);-- bit 4 roye payeh sdin migozarim
va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 16=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 17=>
                        clkin<='1';
                        sdin<=data(3);-- bit 3 roye payeh sdin migozarim
va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 18=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 19=>
                        clkin<='1';
                        sdin<=data(2);-- bit 2 roye payeh sdin migozarim
va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 20=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 21=>
                        clkin<='1';
                        sdin<=data(1);-- bit 1 roye payeh sdin migozarim
va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 22=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 23=>
                        clkin<='1';
                        sdin<=data(0);-- bit 0(LSB) roye payeh sdin
migozarim va bad yek labeye paeen ravande roye payeh clkin emal mikonim
                  when 24=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 25=>
                        clkin<='1';
                        sdin<='0';        --data(3); adrese sakhtafzariye
tarashe A1=0,Payeh A1 be zamin vasl shode
                  when 26=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 27=>
                        clkin<='1';
                        sdin<='0';        --data(2);adrese sakhtafzariye
tarashe A0=0,Payeh A0 be zamin vasl shode
                  when 28=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 29=>
                        clkin<='1';
```

```
                        sdin<=ds(1);      --be vasileye bit haye ds1, ds0
yeki az 4 dac dakhele tarashe ghabele entekhab mibashad
                  when 30=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 31=>
                        clkin<='1';
                        sdin<=ds(0);      --be vasileye bit haye ds1, ds0
yeki az 4 dac dakhele tarashe ghabele entekhab mibashad
                  when 32=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                  when 33=>
                        fsin<='1'; --payane ersale 16 bit be tarasheye
7564
                        clkin<='1';
                        output(15)<= sdout; -- shorue daryafte 16bit az
7564 be fpga va namayesh bit ha roye led ha
                  when 34=>
                        clkin<='0';--yek labeye paeen ravande roye payeh
clkin emal mikonim
                        ldac<='0';-- baraye bargozariye(load) adade 12
biti dar dac entekhab shodeh
                  when 35=>
                        ldac<='1';-- baraye bargozariye(load) adade 12
biti dar dac entekhab shodeh
                        clkin<='1';
                        output(14)<= sdout;
                  when 36=>
                        clkin<='0';
                  when 37=>
                        clkin<='1';
                        output(13)<= sdout;
                  when 38=>
                        clkin<='0';
                  when 39=>
                        clkin<='1';
                        output(12)<= sdout;
                  when 40=>
                        clkin<='0';
                  when 41=>
                        clkin<='1';
                        output(11)<= sdout;
                  when 42=>
                        clkin<='0';
                  when 43=>
                        clkin<='1';
                        output(10)<= sdout;
                  when 44=>
                        clkin<='0';
                  when 45=>
                        clkin<='1';
                        output(9)<= sdout;
                  when 46=>
                        clkin<='0';
                  when 47=>
                        clkin<='1';
                        output(8)<= sdout;
                  when 48=>
                        clkin<='0';
```

```
  when 49=>
        clkin<='1';
        output(7)<= sdout;
  when 50=>
        clkin<='0';
  when 51=>
        clkin<='1';
        output(6)<= sdout;
  when 52=>
        clkin<='0';
  when 53=>
        clkin<='1';
        output(5)<= sdout;
  when 54=>
        clkin<='0';
  when 55=>
        clkin<='1';
        output(4)<= sdout;
  when 56=>
        clkin<='0';
  when 57=>
        clkin<='1';
        output(3)<= sdout;
  when 58=>
        clkin<='0';
  when 59=>
        clkin<='1';
        output(2)<= sdout;
  when 60=>
        clkin<='0';
  when 61=>
        clkin<='1';
        output(1)<= sdout;
  when 62=>
        clkin<='0';
  when 63=>
        clkin<='1';
        output(0)<= sdout;
  when 64=>
        clkin<='0';

        --sending:=0;
        --sending:=0;
        --data<=data+'1';
  when 65=>
        --count:=count+1;
        --if count=period then
        --count:="0000000000000000";
        --sending:=0;
        --data<=data+'1';
        --end if;
  data <= data_i;

        case current is
when square =>   --shekle moje morabaee
    if i <= 100  then
            i:=i+1;
            data_i <= "000000000000";
    elsif i<=200 then
        i:=i+1;
```

```
                    data_i <= "111111111111";
                else
                    i:=0;
                end if;
                count := period;
                current <= delay;
            when saw =>     --shekle moje dandane arehee
                data_i <= data_i + 1;  --baraye shekle moje dandane
arehee az 0 ta 4095 ra ba surate afzayeshi be porte khoruji ersal mikonad
                count := period;       --va pas az shomaresh az 4096
be 0  overflew rokh midahad kedobareh az 0 ta 4096 tekrar mishavad
                current <= delay;
            when triangle => --shekle moje mosallasi
                if down = '0' then     --shomaresh az 0 ta 4095
(so'udi)
                    data_i <= data_i + 1;
                    if data_i = "111111111110" then
                        down := '1';
                    end if;
                else                   --shomaresh az 4095 ta 0
(nozuli)
                    data_i <= data_i - 1;
                    if data_i = "000000000001" then
                    down := '0';
                    end if;
                end if;
                count := period;
                current <= delay;
            when sine =>     --shekle moje sinusi
                if j < 180 then
                    data_i <= table(j); -- jaygozariye maghdire
zaribe sinusi az table be registere data_i baraye enteghal be porte
khoruji
                    j := j + 1;
                else
                    j := 0;
                end if;
                count := period;
                current <= delay;     -- ta'khire beyne ersale har
yek az maghadire jadvale zaribe sinus
            when delay =>             -- halate "delay" baraye ijad
ta'khir beyen har yek az ersal ha be porte khorujiye fpga mibashad
                count := count - 1 ;  -- ke ba tagheere in delay az
tarighe "period" va "count" mitavan frekans ra tagheer dad
                if count = 0 then
                    case wave is
                        when "00" =>
                            current <= square;
                                                          sending:=0;
                        when "01" =>
                            current <= saw;
                                                          sending:=0;
                        when "10" =>
                            current <= triangle;
                                                          sending:=0;
                        when others =>
                            current <= sine;
                                                          sending:=0;
                    end case;
                end if;
```

```
                end case;
                  when others =>
                        null;
                        --sending:=0;
            end case;
            end if;
      end if;
end process;
end architecture;
```

۳۰- اتصال مبدل دیجیتال به آنالوگ AD667

برنامه زیر برای تولید موج مربعی، دندان اره‌ای، مثلثی و سینوسی طراحی شده است. خروجی مبدل (EO) را به اسیلوسکپ وصل کنید و نتیجه را مشاهده نمایید.

انتخاب نوع موج خروجی به وسیله عددی که به wave می‌دهید، انجام می‌شود:

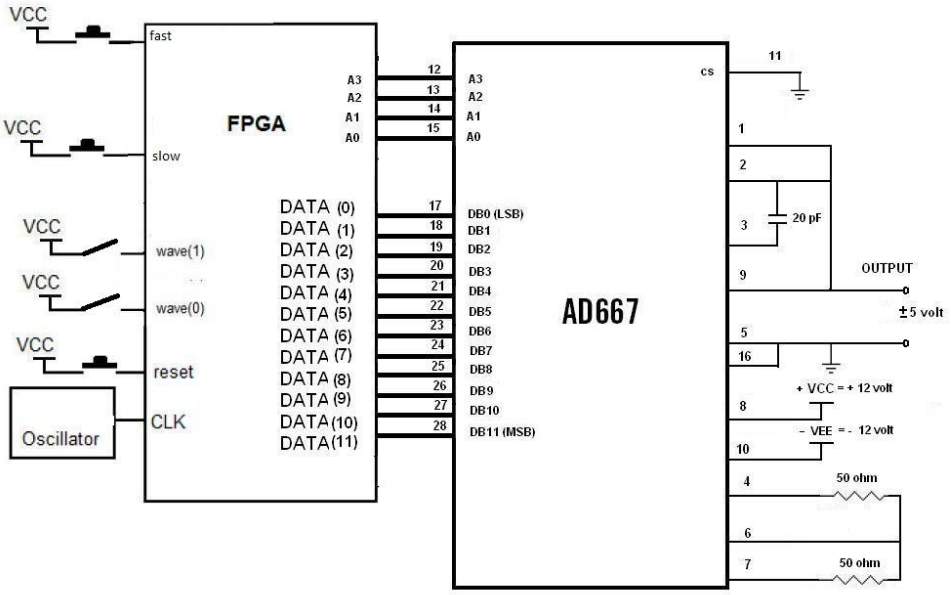
wave = 00 موج خروجی مربعی

wave = 01 موج خروجی دندان اره‌ای

wave = 10 موج خروجی مثلثی

wave = 11 موج خروجی سینوسی

شماتیک:



کد برنامه:

```
library IEEE;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity dac is
Port ( data : out std_logic_vector (11 downto 0 );
a : out std_logic_vector (3 downto 0 );
wave : in std_logic_vector (1 downto 0 );
```

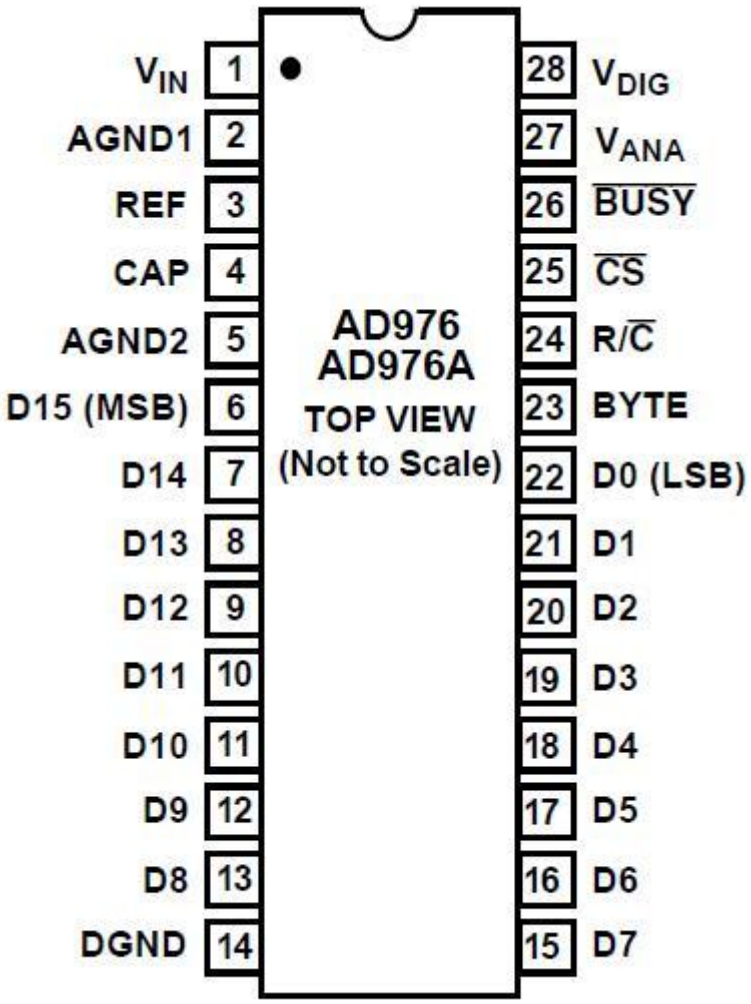
```
clk : in std_logic;
slow : in std_logic;
fast : in std_logic;
reset : in std_logic);
end dac;
architecture Behavioral of dac is
type state is (square, saw, triangle, sine, delay);
signal current : state;
signal data_i : std_logic_vector (11 downto 0);
type two_dim is array (0 to 179) of std_logic_vector (11 downto 0);   --
jadvale zarayebe sinusi
constant table : two_dim :=
("100000000000","100001000111","100010001110","100011010110","10010001110
0","100101100011","100110101001","100111101111",
"101000110100","101001111000","101010111100","101011111111","101101000000
","101110000001","101111000001","101111111111",
"110000111101","110001111000","110010110011","110011101100","110100100100
","110101011010","110110001110","110111000000",
"110111110001","111000100000","111001001101","111001111000","111010100001
","111011001000","111011101101","111100001111",
"111100110000","111101001110","111101101010","111110000100","111110011011
","111110110000","111111000010","111111010010",
"111111100000","111111101011","111111110100","111111111010","111111111110
","111111111111","111111111110","111111111010",
"111111110100","111111101011","111111100000","111111010010","111111000010
","111110110000","111110011011","111110000100",
"111101101010","111101001110","111100110000","111100001111","111011101101
","111011001000","111010100001","111001111000",
"111001001101","111000100000","110111110001","110111000000","110110001110
","110101011010","110100100100","110011101100",
"110010110011","110001111000","110000111101","101111111111","101111000001
","101110000001","101101000000","101011111111",
"101010111100","101001111000","101000110100","100111101111","100110101001
","100101100011","100100011100","100011010110",
"100010001110","100001000111","100000000000","011110111000","011101110001
","011100101001","011011100011","011010011100",
"011001010110","011000010000","010111001011","010110000111","010101000011
","010100000000","010010111111","010001111110",
"010000111110","010000000000","001111000010","001110000111","001101001100
","001100010011","001011011011","001010100101",
"001001110001","001000111111","001000001110","000111011111","000110110010
","000110000111","000101011110","000100110111",
"000100010010","000011110000","000011001111","000010110001","000010010101
","000001111011","000001100100","000001001111",
"000000111101","000000101101","000000011111","000000010100","000000001011
","000000000101","000000000001","000000000000",
"000000000001","000000000101","000000001011","000000010100","000000011111
","000000101101","000000111101","000001001111",
"000001100100","000001111011","000010010101","000010110001","000011001111
","000011110000","000100010010","000100110111",
"000101011110","000110000111","000110110010","000111011111","001000001110
","001000111111","001001110001","001010100101",
"001011011011","001100010011","001101001100","001110000111","001111000010
","010000000000","010000111110","010001111110",
"010010111111","010100000000","010101000011","010110000111","010111001011
","011000010000","011001010110","011010011100",
"011011100011","011100101001","011101110001","011110111000");
begin
process (clk,reset)
    variable c,i, j : integer;
```

```
    variable down : std_logic;
    variable count,period : std_logic_vector (15 downto
0):="0000000100000000";
    begin
        if reset = '1' then
            current <= square;
            data_i <= "000000000000";
            i := 0;
            j := 0;
            down := '0';
        elsif clk = '1' and clk'event then
            if slow='1' then                    -- kelid slow baraye
kaheshe frequecy
                c:=c+1;
                if c>=25000000 then             -- ta'khire nim saniye
                    c:=0;
                    if period<"1000000000000000" then
                        period:=std_logic_vector (unsigned(period) sll
1); --shifte chap baraye afzayeshe ta'khir => kaheshe frequency
                    end if;
                end if;
            end if;
            if fast='1' then                    -- kelid fast baraye
afzayeshe frequecy
                c:=c+1;
                if c>=25000000 then             -- ta'khire nim saniye
                    c:=0;
                    if period>"0000000000000001" then
                        period:=std_logic_vector (unsigned(period) srl
1); --shifte rast baraye kaheshe ta'khir => afzayeshe frequency
                    end if;
                end if;
            end if;
            case current is
                when square =>   --shekle moje morabaee
                    if i <= 100  then
                            i:=i+1;
                            data_i <= "000000000000";
                    elsif i<=200 then
                        i:=i+1;
                        data_i <= "111111111111";
                    else
                        i:=0;
                    end if;
                    count := period;
                    current <= delay;
                when saw =>     --shekle moje dandane arehee
                    data_i <= data_i + 1;  --baraye shekle moje dandane
arehee az 0 ta 4095 ra ba surate afzayeshi be porte khoruji ersal mikonad
                    count := period;       --va pas az shomaresh az 4096
be 0  overflew rokh midahad kedobareh az 0 ta 4096 tekrar mishavad
                    current <= delay;
                when triangle => --shekle moje mosallasi
                    if down = '0' then     --shomaresh az 0 ta 4095
(so'udi)
                        data_i <= data_i + 1;
                        if data_i = "111111111110" then
                            down := '1';
                        end if;
```

```
                    else                    --shomaresh az 4095 ta 0
(nozuli)
                        data_i <= data_i - 1;
                        if data_i = "000000000001" then
                        down := '0';
                        end if;
                    end if;
                    count := period;
                    current <= delay;
                when sine =>     --shekle moje sinusi
                    if j < 180 then
                        data_i <= table(j); -- jaygozariye maghdire
zaribe sinusi az table be registere data_i baraye enteghal be porte
khoruji
                        j := j + 1;
                    else
                        j := 0;
                    end if;
                    count := period;
                    current <= delay;     -- ta'khire beyne ersale har
yek az maghadire jadvale zaribe sinus
                when delay =>             -- halate "delay" baraye ijad
ta'khir beyen har yek az ersal ha be porte khorujiye fpga mibashad
                    count := count - 1 ;  -- ke ba tagheere in delay az
tarighe "period" va "count" mitavan frekans ra tagheer dad
                    if count = 0 then
                        case wave is
                            when "00" =>
                                current <= square;
                            when "01" =>
                                current <= saw;
                            when "10" =>
                                current <= triangle;
                            when others =>
                                current <= sine;
                        end case;
                    end if;
                end case;
        end if;
    end process;
a<="0000"; -- fa'al kardane paye haye A0,A1,A2,A3 dar tarasheye ad667
data <= data_i;
end Behavioral;
```

۳۱- اتصال مبدل آنالوگ به دیجیتال ad976

نمایی از آی سی AD976 :



برای این منظور پورت های زیر را در نظر می گیریم :

RC:که دو عمل READ و CONVERT را انجام میدهد.

CS : برای فعال کردن تراشه

BUSY : برای تشخیص مشغول بودن تراشه

DATA : داده ی تبدیل شده ی آنالوگ به دیجیتال که خروجی ۱۶ بیتی است.

Seven segment : پورت خروجی FPGA

Sevensegment_en : پایه ی فعال کننده ی 7seg

ما در این کد از ۵ حالت یا state استفاده کردیم :

start, read, wait_for_busy, convert_pulse_delay,convert2decimal

سیگنالهای digit_1,digit_10,digit_100,digit_1000,digit_10000 برای نشان دادن یکان و دهگان تا ده هزارگان است.

متغیر data_registerکه پورتی ۱۶ بیتی است از data میخواهد و یک داده ازش میخوانی و در برنامه میخواهی از آن استفاده کنی هر لحظه ممکن است مقدار پایه ها عوض شود که مقدار آنالوگ عوض میشود هرلحظه ، برای اینکه عوض نشود مقداری را گرفته و میبریم تغییر میدهیم.

Convert_delay : شمردن زمان تاخیر برای تبدیل

عدد خروجی ما ۱۶ بیتی است که این به درد 7seg نمیخورد و منطق آن باینری است ما باید معادل یک عدد دسیمال کنیم آن راو باید تبد یل کنیم آن را...

ما در این پروژه از الگوریتم دابل دبل استفاده کردیم که الگوریتمی است که با شیفت دادن و جمع کردن و مقایسه کردن عمل میکند.

یک متغیر ۱۶ بیتی را میخواهد تبدیل کند ۵ تا رقم دهدهی دهد از ۰ تا ۶۵۵۳۵....

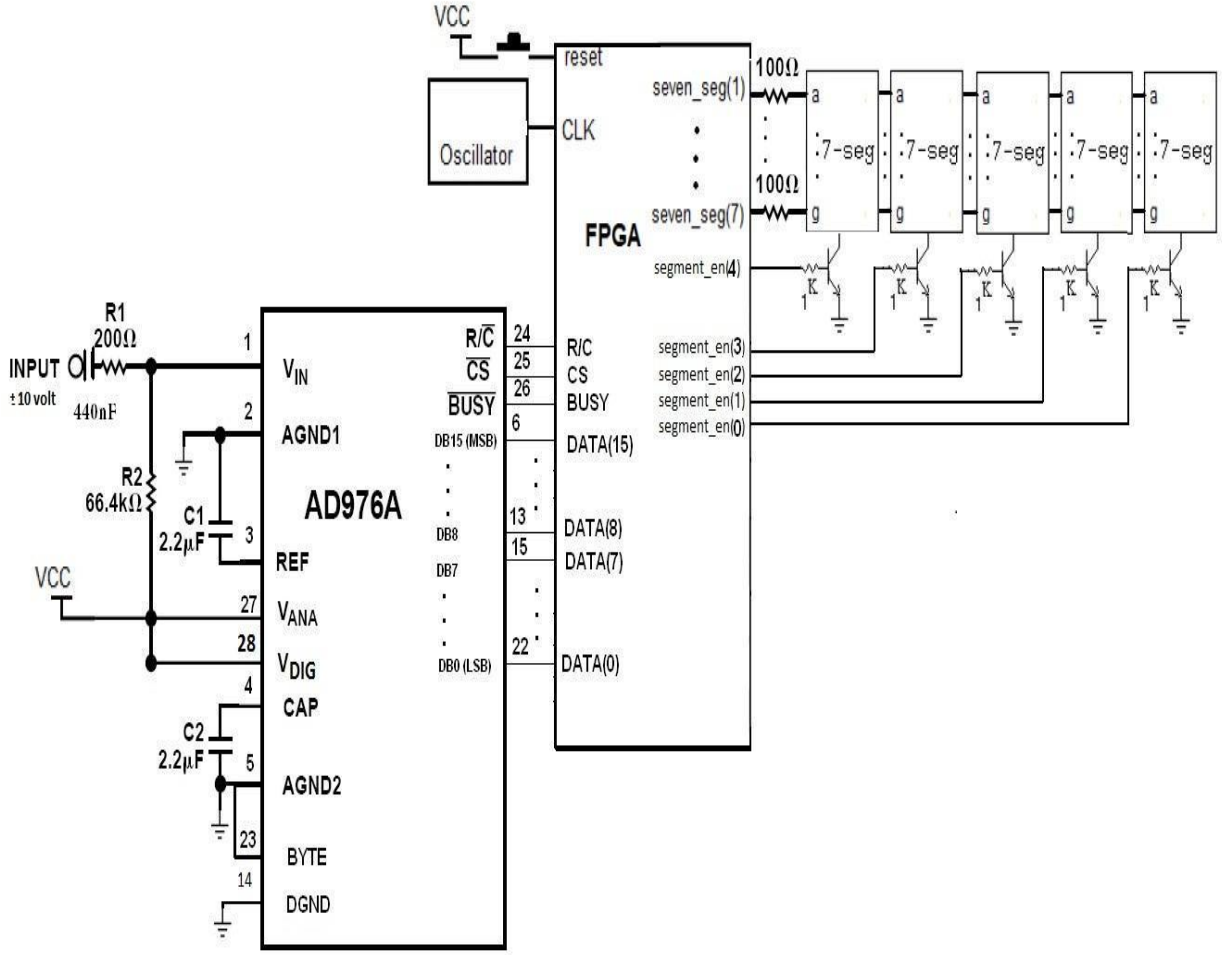
ما در این پروژه ۵ متغیر دسیمال در نظر گرفتیم که هر کدام ۴ بیتی است به نام های one,ten,hund,thou,tthouکه از یکان تا ده هزارگان هستند و در حالت پیش فرض به آنها صفر دادیم. عدد را یک بیت یک بیت به سمت چپ شیفت داده بیت اول وارد متغیر می شود بعد از مقایسه با عدد ۵ اگر عدد بزرگتر از ۴ بود آن وقت باید با عدد ۳ جمع بسته شود. ۱۶ بار این کار را انجام می دهد اگر عدد ما ۱۶ بیتی باشد باید ۱۶ شیفت انجام شود که همه این بیت ها تک تک وارد این متغیرها بشوند بعد از اتمام این ۱۶ بار ما عددهایی داریم که از صفر تا ۹ هستند که به حالت BCD نشان داده شده اند که ما این اعداد را می فرستیم به سون سگمنت تا اعداد تبدیل شده را نشان دهد حال ۱۶ مرحله تبدیل ۱۶ بیت

شیفت داده شده و مقایسه ها انجام شده اند. ما در کد برنامه ۵ if پشت سر هم داریم که در یک کلاک انجام می شود و همزمان است. بیت آخر دیتا رجیستر همیشه وصل است به بیت کم ارزش ONE.

زمانی که ۱۶ شیفت انجام شده اعداد بدست آمده معادل اعداد باینری است. در هر تبدیل باید تاخیری ایجاد کنیم تا سریع این تبدل انجام نشود چون قابل دیدن و رویت برای ما نیست. در مدار هر ۵ سون سگمنت با هم مقدار می گیرد ولی به شرطی نشان می دهد که ترانزیستور مربوط به آن روشن باشد پس ما ترانزیستور اول را روشن می کنیم و مقدار یکان را داخل آن می ریزیم و بنا بر کد یک پنجم ثانیه منتظر می ماند و حال که این زمان سپری شد این ترانزیستور خاموش و دومی را روشن می کند. (دهگان). اعداد به دست آمده در کد برای مقدار دهی تاخیر زمانی به صورت تجربی بدست آمده اند. اگر کمتر از این مقدار باشند لرزش به چشم نمی آید ولی اگر بیشتر باشد روند کار ما کند می شود، سرعت کم و تاخیر زیاد می شود و با چشم می توان این تاخیر را مشاهده نمود و اگر هم این مقدار را خیلی کم کنیم نور سون سگمنت کم می شود.

در نهایت باید گفت که به دلیل آنکه این ۵ سون سگمنت به همدیگر وصل هستند و در هر لحظه یک مقدار را نشان می دهند باید در آن لحظه یک سون سگمنت را برای نشان دادن آن مقدار فعال کنیم و چون این تاخیر بین نشان دادن در سون سگمنت ها در حد میلی ثانیه است ما این تغییر مقدار دهی در سون سگمنت ها را با لرزش مشاهده می کنیم و این تغییر به چشم نمی آید و در آن واحد میتوانیم در هر سون سگمنت یک مقدار متفاوت داشته باشیم با توجه به فعال یا غیرفعال شدن آن سون سگمنت در آن رنج زمانی.

شماتیک ic976a به این شکل می باشد:



کد برنامه بدین شکل می باشد :

```
library IEEE;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity a2d is
Port (
rc : out std_logic;
cs : out std_logic;
busy : in std_logic;
data : in std_logic_vector(15 downto 0);
seven_segment : out std_logic_vector(6 downto 0);
segment_en : out std_logic_vector (4 downto 0);
clk : in std_logic;
reset : in std_logic);
end a2d;
architecture Behavioral of a2d is
type state is (start, read, wait_for_busy,
convert_pulse_delay,convert2decimal);
signal current : state;
```

```
signal digit_1,digit_10,digit_100,digit_1000,digit_10000 :
std_logic_vector(6 downto 0);
begin
process (clk, reset)
variable c1,i,j,convert_delay : integer;
variable one,ten,hund,thou,tthou: std_logic_vector(3 downto 0) := "0000;"
variable data_register: std_logic_vector(15 downto 0) :=
"0000000000000000"; --16 BIT
begin
if reset = '1' then
i:=0;
rc <= '1;'
cs <= '0;'
current <= start;
elsif clk = '1' and clk'event then
case current is
when start<=
cs <= '0;'
rc <= '0;'
convert_delay:=0;
current <= convert_pulse_delay;
when convert_pulse_delay =>            -- dar in marhale pulse monaseb
baraye aghaze tabdil voltage analog be yek adade binery be ad976 frestade
mishavad
convert_delay:=convert_delay+1;
if convert_delay <= 2 then                ----convert_delay:=0,1,2
=>3*20ns=60ns
current <= convert_pulse_delay;
else
current <= wait_for_busy;
rc<='1;'
end if;
when wait_for_busy =>----4us   --barname dar in state montazer mimanad ta
tarshe ad976 az halate "busy" kharej shavad
if busy = '0' then                                ----Data Valid Delay
after R/C Low=> 4us tebghe datasheet(t2 dar jadvale safheye 4(
current <= wait_for_busy;
elsif busy = '1' then        --agar tarashe ad976 az halat "busy" kharej
shod barname be state "read" miravad
current <=read;
end if;
when read =>                   ----۲۰ns
data_register := data ;  --enteghale adad binerye 16biti be ye
moteghayere dakheli baraye anjame amaliate tabdil be decimal
current<=convert2decimal;-- sepas be marhaleye tabdil az binery be
decimal miravad
i:=0;
when convert2decimal =>----18clk*20ns=>360ns baraye tabdile bin to dec
if i=0 then       ----ta inja kollan:4.440us:
60ns+4000ns+20ns+360ns=4440ns
i:=i+1;
tthou := "0000"; --pak kardane moteghayerha ke hanuz maghadire  marhaleye
ghabli ra darand
thou := "0000";  --pak kardane moteghayerha ke hanuz maghadire  marhaleye
ghabli ra darand
hund := "0000";  --pak kardane moteghayerha ke hanuz maghadire  marhaleye
ghabli ra darand
ten :=  "0000";  --pak kardane moteghayerha ke hanuz maghadire  marhaleye
ghabli ra darand
```

```
one :=  "0000";  --pak kardane moteghayerha ke hanuz maghadire  marhaleye
ghabli ra darand
elsif i=1 then
i:=i+1;
data_register(15):=not data_register(15)    ;   -- tabdile moteghayere
alamat dar be moteghayere bedune alamat
elsif i<=17 then                                    -- tabdile binery be
decimal (bcd(
i:=i+1;
current <= convert2decimal;
if (tthou >= "0101") then
tthou := std_logic_vector (unsigned(tthou) +3;(
end if;
if (thou >= "0101") then
thou := std_logic_vector (unsigned(thou) +3);
end if;
if (hund >= "0101") then
hund := std_logic_vector (unsigned(hund) +3);
end if;
if (ten >= "0101") then
ten := std_logic_vector (unsigned(ten) +3);
end if;
if (one >= "0101") then
one := std_logic_vector (unsigned(one) +3);
end if;
tthou := tthou(2 downto 0) & thou(3);          -- THE SHIFTING
thou := thou(2 downto 0)& hund(3);         -- THE SHIFTING
hund := hund(2 downto 0)& ten(3);          -- THE SHIFTING
ten := ten(2 downto 0)& one(3);            -- THE SHIFTING
one := one(2 downto 0)& data_register(15); -- THE SHIFTING
data_register := std_logic_vector (unsigned(data_register) sll 1);   --
THE SHIFTING
else
if c1<=28000000 then   ----28*20ns=560ns+4440ns=5000ns=5us  -- Ta'khire
beyne har tabdil
c1:=c1+1;                           -- barname be andazeye ta'khir dar
halat "convert2decimal" mimanad
current<=convert2decimal;
else                                    -- pas az tamamshodan ta'khir
barname be halat start bazmigardad
current<=start;
i:=0;
c1:=0;
end if;
end if;
end case;
case one IS
 --------------------codhaye marbut be seven-segment ha---------------
when "0000" => digit_1 <= "0111111" ; -- 0
when "0001" => digit_1 <= "0000110" ; -- 1
when "0010" => digit_1 <= "1011011" ; -- 2
when "0011" => digit_1 <= "1001111" ; -- 3
when "0100" => digit_1 <= "1100110" ; -- 4
when "0101" => digit_1 <= "1101101" ; -- 5
when "0110" => digit_1 <= "1111101" ; -- 6
when "0111" => digit_1 <= "0000111" ; -- 7
when "1000" => digit_1 <= "1111111" ; -- 8
when "1001" => digit_1 <= "1101111" ; -- 9
when others => digit_1 <= "0000000-- ; "
end case;
```

```
case ten IS
when "0000" => digit_10 <= "0111111" ; -- 0
when "0001" => digit_10 <= "0000110" ; -- 1
when "0010" => digit_10 <= "1011011" ; -- 2
when "0011" => digit_10 <= "1001111" ; -- 3
when "0100" => digit_10 <= "1100110" ; -- 4
when "0101" => digit_10 <= "1101101" ; -- 5
when "0110" => digit_10 <= "1111101" ; -- 6
when "0111" => digit_10 <= "0000111" ; -- 7
when "1000" => digit_10 <= "1111111" ; -- 8
when "1001" => digit_10 <= "1101111" ; -- 9
when others => digit_10 <= "0000000-- ; "
end case;
case hund IS
when "0000" => digit_100 <= "0111111" ; -- 0
when "0001" => digit_100 <= "0000110" ; -- 1
when "0010" => digit_100 <= "1011011" ; -- 2
when "0011" => digit_100 <= "1001111" ; -- 3
when "0100" => digit_100 <= "1100110" ; -- 4
when "0101" => digit_100 <= "1101101" ; -- 5
when "0110" => digit_100 <= "1111101" ; -- 6
when "0111" => digit_100 <= "0000111" ; -- 7
when "1000" => digit_100 <= "1111111" ; -- 8
when "1001" => digit_100 <= "1101111" ; -- 9
when others => digit_100 <= "0000000-- ; "
end case;
case thou IS
when "0000" => digit_1000 <= "0111111" ; -- 0
when "0001" => digit_1000 <= "0000110" ; -- 1
when "0010" => digit_1000 <= "1011011" ; -- 2
when "0011" => digit_1000 <= "1001111" ; -- 3
when "0100" => digit_1000 <= "1100110" ; -- 4
when "0101" => digit_1000 <= "1101101" ; -- 5
when "0110" => digit_1000 <= "1111101" ; -- 6
when "0111" => digit_1000 <= "0000111" ; -- 7
when "1000" => digit_1000 <= "1111111" ; -- 8
when "1001" => digit_1000 <= "1101111" ; -- 9
when others => digit_1000 <= "0000000-- ; "
end case;
case tthou IS
when "0000" => digit_10000 <= "0111111" ; -- 0
when "0001" => digit_10000 <= "0000110" ; -- 1
when "0010" => digit_10000 <= "1011011" ; -- 2
when "0011" => digit_10000 <= "1001111" ; -- 3
when "0100" => digit_10000 <= "1100110" ; -- 4
when "0101" => digit_10000 <= "1101101" ; -- 5
when "0110" => digit_10000 <= "1111101" ; -- 6
when "0111" => digit_10000 <= "0000111" ; -- 7
when "1000" => digit_10000 <= "1111111" ; -- 8
when "1001" => digit_10000 <= "1101111" ; -- 9
when others => digit_10000 <= "0000000-- ; "
end case;
j:=j+1;
case j is
when 1          => segment_en<="00001";     -- transistor marbut be
seven-segmente "yakan" fa'al mishavad , va sayere transistor ha gheire
fa'al mishavand
seven_segment<=digit_1;     -- raghame yekan roye bus moshtarak(porte
seven_segment) gozashte mishavad
when 50000          => segment_en<="00000;"
```

```
seven_segment<="0000000;"
when 100000     => segment_en<="00010";             -- transistor marbut
be seven-segmente "dahgan" fa'al mishavad ,va sayere transistor ha gheire
fa'al mishavand
seven_segment<=digit_10;    -- raghame dahgan roye bus moshtarak(porte
seven_segment) gozashte mishavad
when 150000          => segment_en<="00000;"
seven_segment<="0000000;"
when 200000    => segment_en<="00100";             -- transistor marbut
be seven-segmente "sadgan" fa'al mishavad ,va sayere transistor ha gheire
fa'al mishavand
seven_segment<=digit_100;   -- raghame sadgan roye bus moshtarak(porte
seven_segment) gozashte mishavad
when 250000          => segment_en<="00000;"
seven_segment<="0000000;"
when 300000    => segment_en<="01000";             -- transistor marbut
be seven-segmente "hezargan" fa'al mishavad ,va sayere transistor ha
gheire fa'al mishavand
seven_segment<=digit_1000;  -- raghame hezargan roye bus moshtarak(porte
seven_segment) gozashte mishavad
when 350000          => segment_en<="00000;"
seven_segment<="0000000;"
when 400000    => segment_en<="10000";             -- transistor marbut
be seven-segmente "dah hezargan" fa'al mishavad ,va sayere transistor ha
gheire fa'al mishavand
seven_segment<=digit_10000; -- raghame dah hezargan roye bus
moshtarak(porte seven_segment) gozashte mishavad
when 450000          => segment_en<="00000;"
seven_segment<="0000000;"
when 500000    => j:=0;
when others     => null;
end case;
end if;
end process;
end Behavioral;
```

pwm -۳۲

مدار پروژه جاری به وسیله تکنیک مدولاسیون عرض پالس باعث تغییر در Duty Cycle یک پالس ورودی شده و از این طریق موجب تغییر در سرعت چرخش یک موتور DC می شود. Duty Cycle یک پالس، نسبت یک بودن پالس به صفر بودن آن در یک دوره کامل (Period) است. همانطور که می دانیم بر خلاف موتور پله ای که سرعت و جهت چرخش آن بنا به نظر طراح مدار به سادگی قابل تغییر است، سرعت و جهت چرخش موتور DC به طور معمول قابل کنترل نیست. یکی از راه های کنترل سرعت چرخش موتور DC استفاده از مدار PWM است. مدار PWM امروزه در صنعت برای کنترل سرعت چرخش موتورهای صنعتی ، در رباتیک و... استفاده گسترده ای دارد.

شرح پروژه :

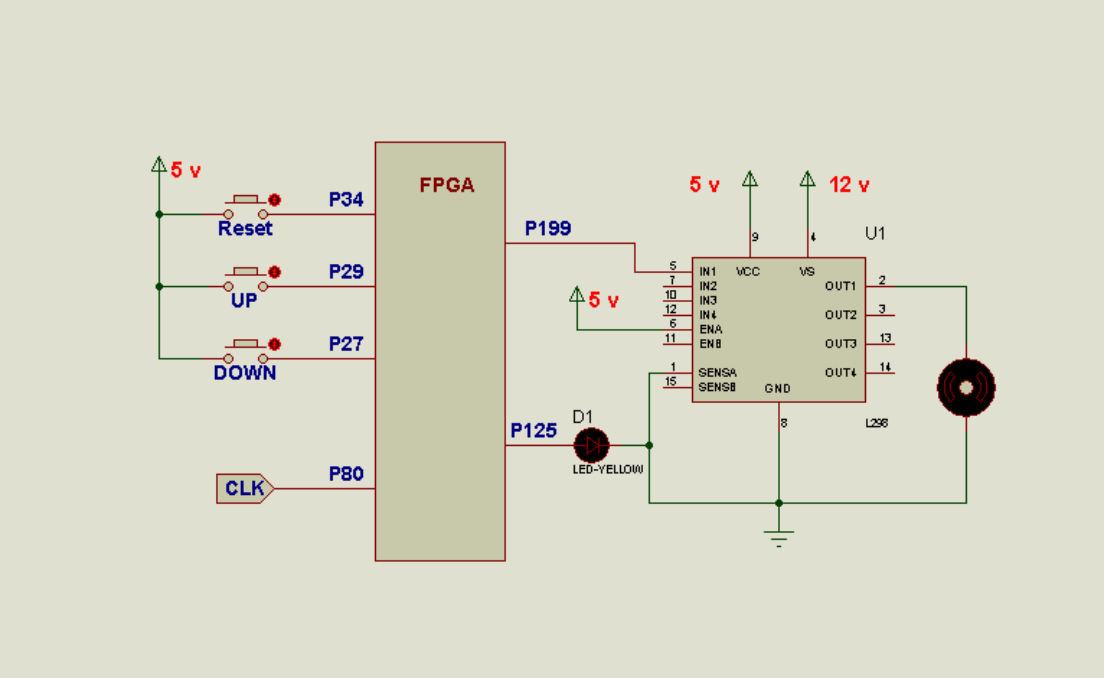
مدار این پروژه کلا دارای دو ورودی به نام های up و down و یک خروجی برای اتصال به موتور است. با فشردن کلید up سرعت موتور افزایش و با فشردن کلید down سرعت موتور کاهش می یابد. متغیر اصلی برنامه x است که با هر بار فشردن کلید up یک واحد به آن افزوده و با هر بار فشردن کلید down یک واحد از آن کاسته می شود. فرکانس کاری برنامه 500 HZ است. متغیر y یک متغیر کمکی است که هر $\frac{1}{500}$ ثانیه مقدار آن یک واحد اضافه می شود. برنامه نرم افزاری این پروژه به شیوه RTL نوشته شده و دارای دو state به نام های counting و waiting است.

۱. در حالت counting مقدار حال حاضر x با مقدار y مقایسه می شود اگر x از مقدار y کوچکتر بود ، به تعداد پالس های عدد x ، خروجی برنامه ۱ می شود. تا زمانیکه $x > y$ است این روند ادامه می یابد.

۲. زمانیکه مقدار عددی دو متغیر x و y برابر شد برنامه به حالت waiting رفته و خروجی برنامه صفر می شود و به اندازه ای که متغیر y به حد نهایت عددی متغیر x برسد خروجی برنامه صفر می ماند.

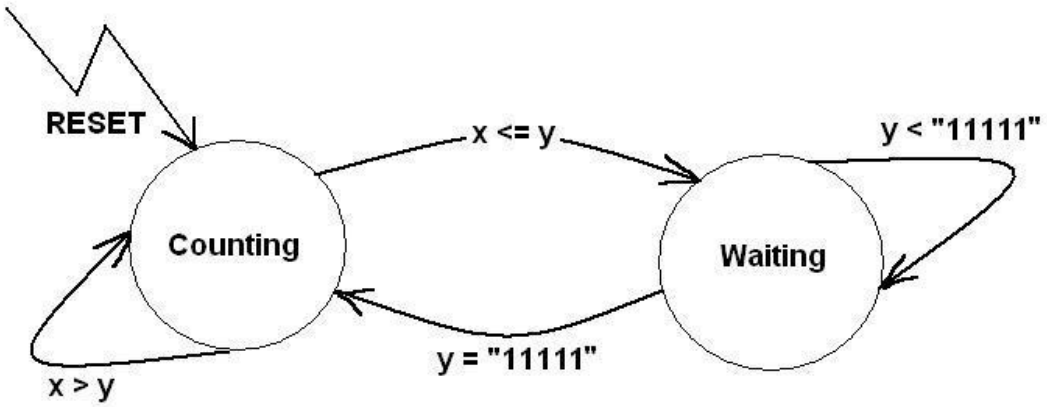
لازم به توضیح است که حد نهایت عددی متغیر x ، به طول بردار متغیر x بستگی دارد. در این پروژه متغیر x از نوع Std_logic_vector و به طول پنج بیت است. پس اگر به عنوان مثال متغیر x دارای عـدد 01001 یا عدد ۹ باشد، به اندازه هشت بار اضافه شدن متغیر y خروجی مدار پروژه را ’1‘ کرده و به اندازه اخـتلاف آن با نهایت عددی متغیر x یعنی ۲۳=۹-۳۲ بار اضافه شدن متغیر y خروجی مدار را ’0‘ مـی کنـد. بـدین صورت Duty Cycle پالس خروجی با فشردن دو کلید Up و Down و تغییر مقادیر دو متغیر x و y عـوض شده و موجب تغییر در سرعت چرخش موتور DC می شود.

در صفحه بعد شکل مدار این پروژه آمده است :



شکل مدار پروژه مدولاسیون عرض پالس PWM

و دیاگرام حالت مدار این پروژه به صورت شکل زیر می باشد.



دیاگرام حالت برای برنامه پروژه PWM

کد برنامه نرم افزاری مدار پروژه به زبان VHDL در زیر آورده شده است :

```
-- Company:
-- Engineer:
--
-- Create Date:    16:18:39 12/25/11
-- Design Name:
-- Module Name:    pwm - Behavioral
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
-------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity pwm is
port(

clk,pwm_reset,pwm_up,pwm_down : in std_logic;                          --
up and down are the main inputs.
```

```
pwm_out,pwm_led_out : out std_logic                          -- pwm_out is
the main output.

);


end pwm;

architecture Behavioral of pwm is


constant pwm_debounce_time : integer := 25000;                --  500 us.
Baraye zamane debounce
constant pwm_led_time : integer := 25000000;                     --  500
ms. Modat zamane roshan budane LED
type pwm_state is (pwm_counting,pwm_waiting); -- tarife hAlathaye PWM
signal pwm_current : pwm_state;
signal pwm_x : std_logic_vector(4 downto 0) ;      -- main signal that
change with up or down. Meghdare marja baraye amalkarde PWM
signal
pwm_flag_up,pwm_flag_down,pwm_before,pwm_debounce_flag,pwm_led_flag :
std_logic;
signal pwm_count,pwm_temp : integer;  --tarife timer haye debounce va LED


begin




-- PWM Process ----------------------------------------------------


process (clk)

variable pwm_frequency : integer := 0;                -- variable for
dividing frequency. Taghsime ferekanse baraye bedast avardane ferekanse
500Hz
variable pwm_y : std_logic_vector(4 downto 0) := "00000";     -- this
variable should compare with "x".

begin

if (pwm_reset='1') then --meghdar dehie avvalie be tamame parameter haye
zir
pwm_out <= '0';
pwm_led_out <= '0';
pwm_x <= "00011";                -- initial variable "x" with "00011"
that begins at 3rd step.
pwm_current <= pwm_counting;
pwm_flag_up <= '0';
pwm_flag_down <= '0';
pwm_debounce_flag <= '0';
pwm_led_flag <= '0';
elsif (clk'event and clk='1') then
pwm_frequency := pwm_frequency+1;                          -- increament
the frequency variable.

-- pwm up button -------------------------------------------------
```

```
if (pwm_up = '1' and pwm_flag_up='0')then        -- Detecting the input
"UP".Input is Low Active.
if (pwm_debounce_flag = '0') then
pwm_before <= pwm_up;  --hAlate ghablie pwm_up ra negah midarad
pwm_debounce_flag <= '1';  --ba inkar timer count fa'al migardad
pwm_flag_up <='1';  -- baraye jelogiri az vorude mojaddad be in block
end if;

else
if (pwm_count = pwm_debounce_time) then                    -- Debouncing
the input "UP". Debounc anjam shod

If (pwm_before = pwm_up ) then   --agar hanuz dokmeye up paein ast
pwm_x <= pwm_x+1;  --meghdare marja ra yeki ezafe kon
pwm_led_out <= '1';--By activating this output,user can realize that the
key is pressed once. LED ra roshan mikonad ta karbar motevajehe feshorde
shodane dokme shavad
pwm_debounce_flag <= '0';   --count ra sefr mikonad
pwm_led_flag <='1';    --timer LED ra fa'al mikonad
else
pwm_debounce_flag <= '0';   --yani yek noise baese fa'al shodane pwm_up
shode pas timere count ra sefr kon
end if;
end if;


if (pwm_temp = pwm_led_time)then  --agar timer LED be 500ms resid LED ra
khamush kon va timere LED ra ba sefr kardane led_flag, sefr kon
pwm_led_out <= '0';
pwm_led_flag <='0';
end if;


end if;


if (pwm_up='0') then  --agar digar dokmeye up feshorde shode nist flag_up
ra sefr kon
pwm_flag_up<= '0';
end if;


------ pwm down button-----------------------------------------------------
-------
-- Tozihate in bakhsh daghighan mesle up button ast
if (pwm_down = '1' and pwm_flag_down='0')then   -- Detecting the input
"DOWN".Input is Low Active.

if (pwm_debounce_flag = '0') then
pwm_before <= pwm_down;
pwm_debounce_flag <= '1';
pwm_flag_down <='1';
end if;
else

if (pwm_count = pwm_debounce_time) then                -- Debouncing the
input "DOWN".
If (pwm_before = pwm_down ) then
pwm_x <= pwm_x-1;
```

```
pwm_led_out <= '1';    --By activating this output,user can realize that
the key is pressed once.
pwm_debounce_flag <= '0';
pwm_led_flag <='1';
else
pwm_debounce_flag <= '0';
end if;
end if;


if (pwm_temp = pwm_led_time)then
pwm_led_out <= '0';
pwm_led_flag <='0';
end if;

end if;


if (pwm_down='0') then
pwm_flag_down<= '0';
end if;

----- pwm main-----------------------------------------------------------
------

case pwm_current is
-->
when pwm_counting =>
if (pwm_frequency = 100000) then                          -- program
frequency is 500 Hz.
if ( pwm_x>pwm_y ) then                       --agar y kamtar x ast khoruji
ra 1 negahdar va y ra yeki ezafe kon
pwm_y := pwm_y+1;
pwm_out <= '1';                                    -- activate the main
output
pwm_current <= pwm_counting;                      -- dar hAlate counting
baghi beman
else                                                              -- dar
gheire insurat khoruji ra sefr va be hAlate waiting boro
pwm_out <= '0';                                    -- deactivate the
output.
pwm_current <= pwm_waiting;                                -- go to state
"waiting".
end if;

pwm_frequency := 0;                                          -- ferekans
ra sefr kon
end if;
-->
when pwm_waiting =>
if (pwm_frequency = 100000) then                       -- program
frequency is 500 Hz.
if ( pwm_y = "11111" ) then                  -- agar y be hadde aksare
meghdare khod reseid, An ra sefr kon va be hAlate counting boro
pwm_y := "00000";
pwm_current <= pwm_counting;                             -- go to state
"counting"
else                                       -- agar be hadde aksare meghdare
khod naresid An ra yeki ezafe kon va dar hAlate waiting baghi beman
pwm_y := pwm_y+1;
```

```
pwm_current <= pwm_waiting;                    -- return to state "Waiting"
(this state).
end if;
pwm_frequency := 0;                                          -- reset the
frequency variable.
end if;
end case;

end if;

-- in ghesmat marbut be shomarande hast----
if (pwm_reset ='1')then --agar reset zade shod timer ha ra sefr kon
pwm_count <= 0;
pwm_temp <= 0;

elsif (clk'event and clk='1') then

if (pwm_debounce_flag = '1') then --agar debounce_flag yek bud count be
shomaresh edame dahad, dar gheire in surat An ra sefr kon
pwm_count <= pwm_count + 1;
else
pwm_count <= 0;
end if;


if (pwm_led_flag='1') then
pwm_temp <= pwm_temp+1;        --agar led_flag yek bud, temp be shomaresh
edame dahad, dar gheire in surat An ra sefr kon
else
pwm_temp <= 0;
end if;

end if;

end process;
end Behavioral;
```

logic probe -۳۳

مقدمه‌ای بر logic probe

نوسان سازهای مونواستابل را" نوسان سازهای تک پالس "مینامند که مدت زمان این پالس بوسیله شبکه خارجی RC مشخص میشود.

درحالت پایدار معمولاً خروجی مدار در سطح صفر قرار دارد .زمانی که پالس ورودی اعمال میگردد خروجی به سطح یک میرود(Vcc).

مدت زمانی که خروجی در سطح یک باقی میماند به شبکه RC بستگی دارد .بعد از این مدت زمان سیگنال خروجی به سطح پایدار صفربرمیگردد .خروجی مدار تا رسیدن پالس ورودی بعدی در سطح صفر سطح پایدار باقی میماند.

از آنجا که اینگونه مدارها تنها یک سطح پایداری(Stable) دارند مدارات Monostable نامیده میشوند.

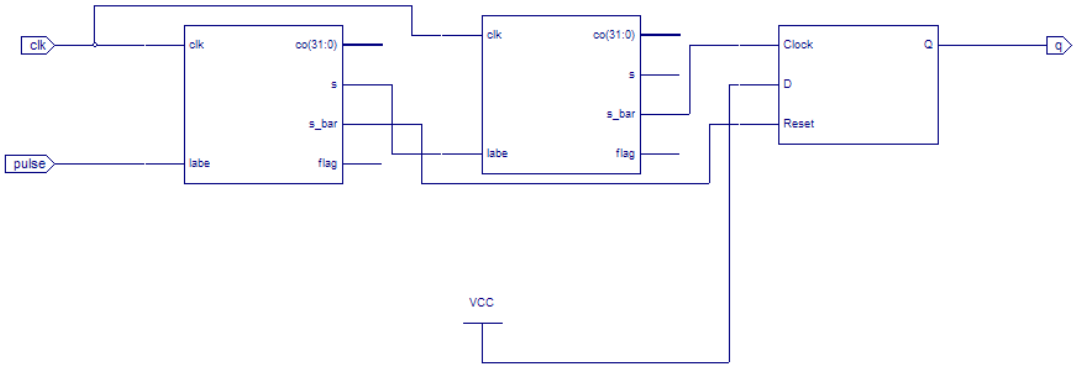
حال چگونه میتوان این مدارات را شبیه سازی کرد .

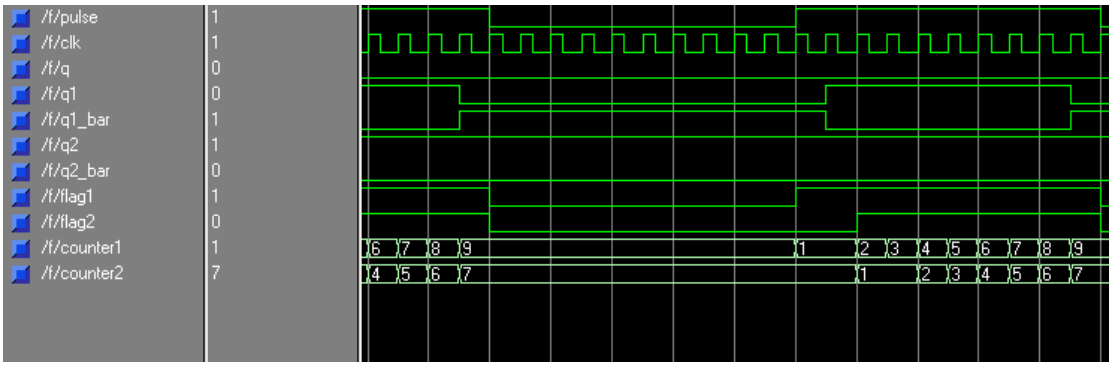
در این پروژه با استفاده از کانتر وکلاک ورودی اعمال شده که کانتر وابسته به آن است شمارش میکند که با استفاده از این کانتر مدت زمان لازم برای بالا نگه داشتن لبه فراهم شده است . در این پروژه از دو مونواستابل استفاده شده که ورودی اولی، پالس ورودی است و ورودی دومی سیگنال q ی مونواستابل اولی است q_bar مونواستابل دوم به کلاک dff رفته و q ی مونواستابل اولی به پایه ی ریست dff رفته است، اگر عرض پالس سیگنال pulse به اندازه ی زمان سطح پایداری مونواستابل اول برسد، q1 را 1 نموده و اگر q1 یا همان ورودی مونواستابل دوم، همیشه در ۱ باقی بماند، q2_bar آن همیشه ۱ خواهد شد، چون هرگز لبه ای رخ نمی دهد تا مونواستابل دوم آن را تشخیص دهد، و در شرط برنامه(قسمت ستاره دار) مقدار 1 به آن تخصیص داده میشود.

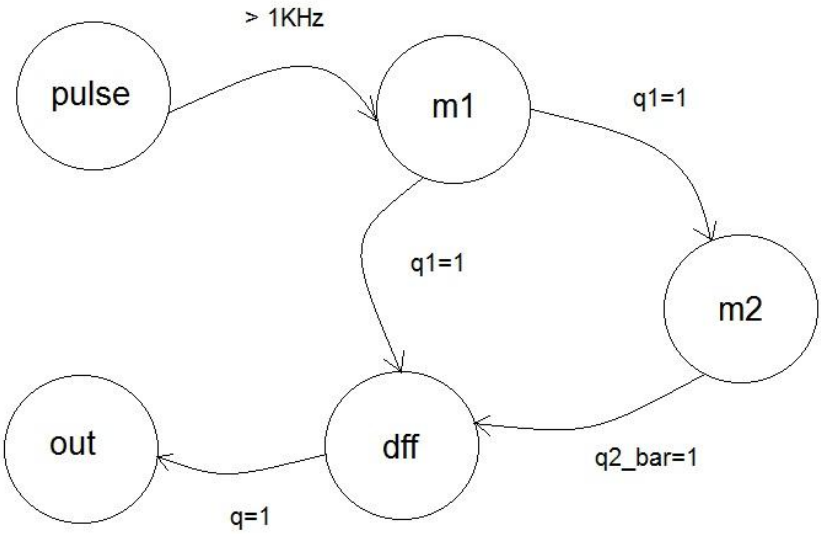
بنابراین سیگنال کلاک dff (یا q2_bar) در حالی که ریست (یا q1) سطح ۱ است، از ۰ به ۱ رفته و خروجی dff را 1 می نماید و چون کلاک dff (یا همان q1) دیگر تغییر نمیکند(حداقل تا تغییر pulse) و

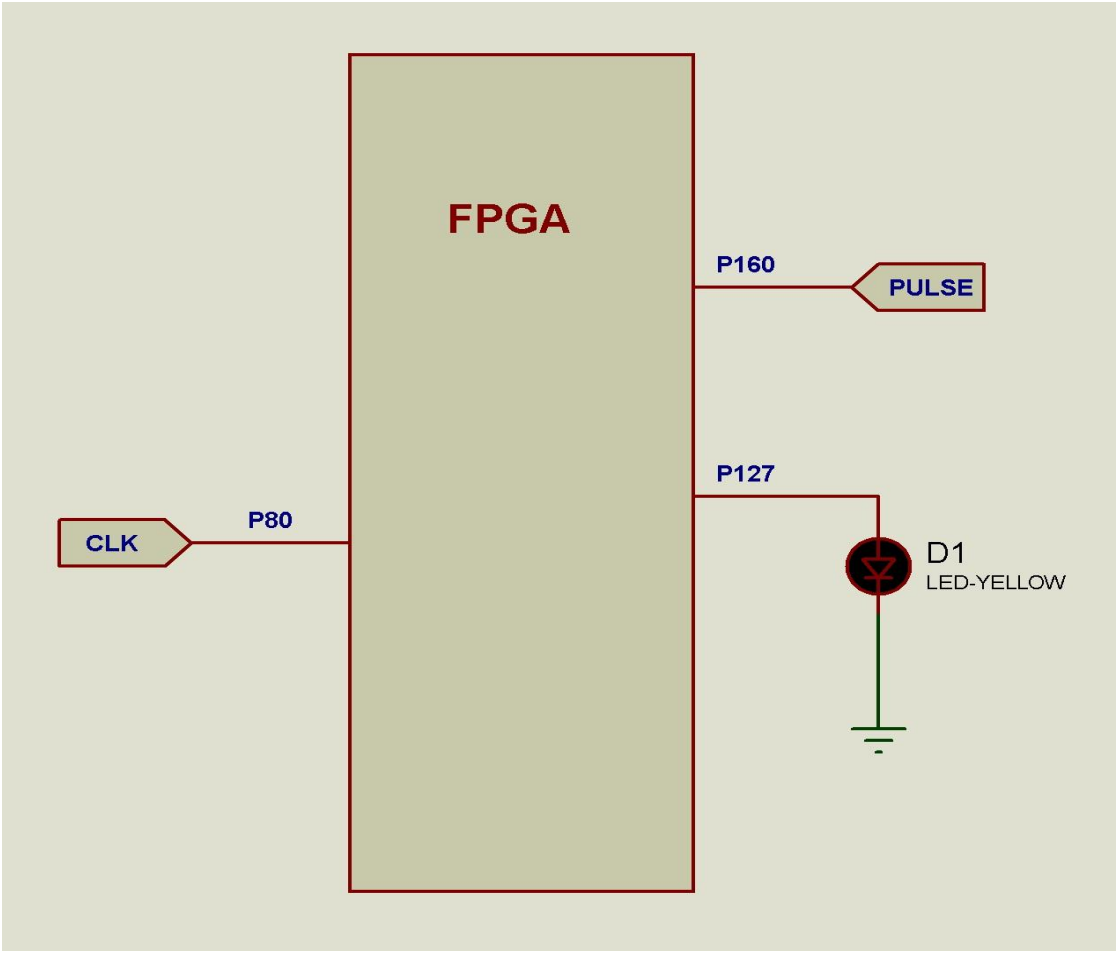
کلاک dff (یا q2_bar) درسطح منطقی 1 باقی میماند ، خروجی dff نیز 1 باقی مانده و تا تغییر پالس ورودی در مونواستابل اول در همین حالت قرار خواهد داشت.

شماتیک کلی :






> 1KHz
pulse
m1
q1=1
q1=1
m2
out
dff
q2_bar=1
q=1


FPGA
P160
PULSE
P127
CLK
P80
D1
LED-YELLOW

برنامه:

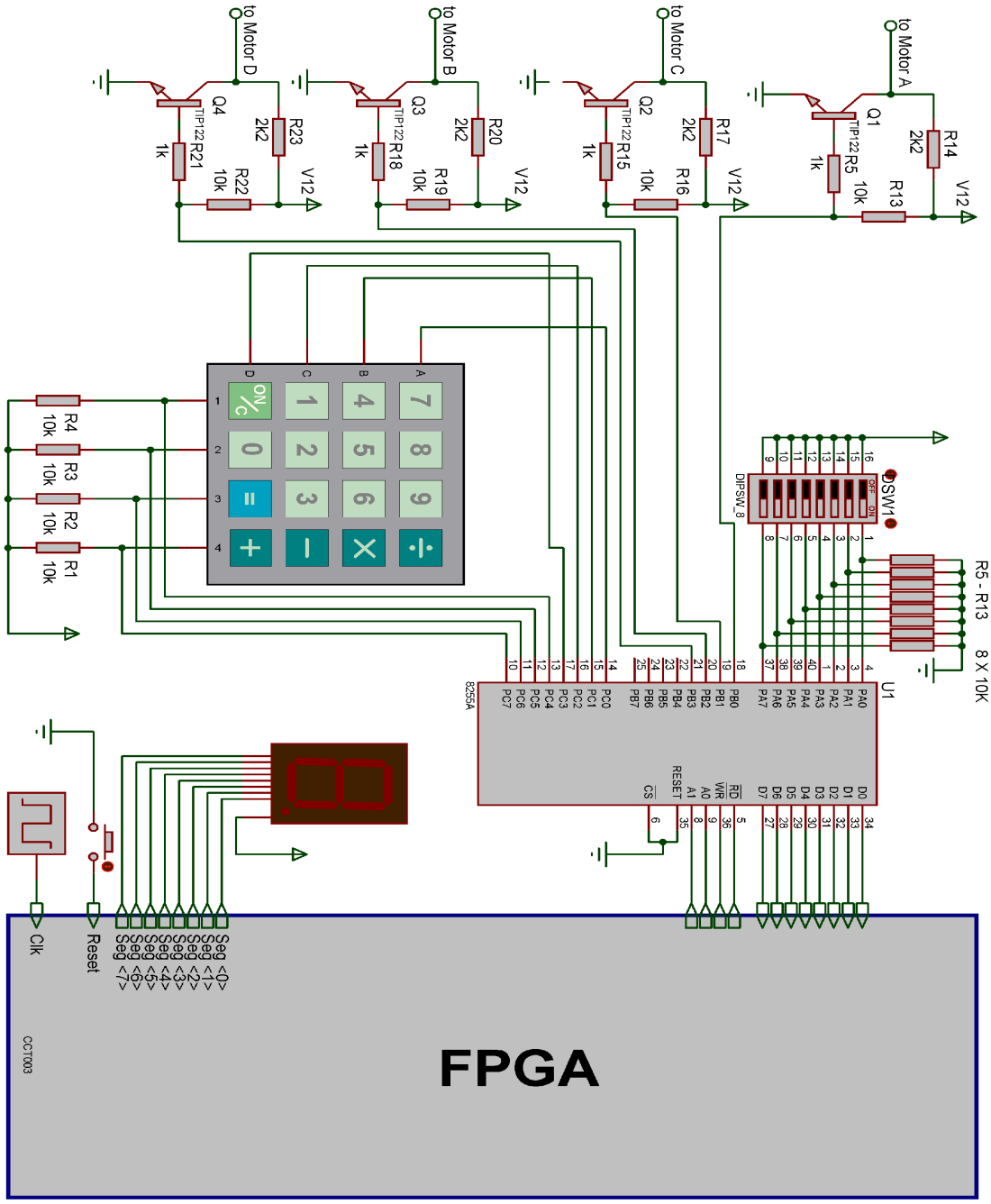
```
 ----------------------------------------------------------------------------
--------
-- Company:
-- Engineer:
--
-- Create Date:    16:10:58 12/25/11
-- Design Name:
-- Module Name:    probe - Behavioral
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
-------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity probe is
port(


clk:in std_logic;
pulse : in std_logic;
q : out std_logic


);


end probe;

architecture Behavioral of probe is



--logic probe signals ----------------------------------------------------

signal counter1,counter2 : integer range 0 to 50000; -- baraye taghsime
ferekans, ta ferekanse 1KHz bedast Ayad
signal flag1,flag2: std_logic;
signal q1,q2_bar : std_logic;
signal decide1,decide2 : std_logic_vector(1 downto 0) :="00";
```

```
begin




-- Probe Process -----------------------------------------------------


process (clk)

begin

-- monostabale 1


if (clk='1' and clk'event) then
decide1(1) <= decide1(0); --jahate tashkhise labe
decide1(0) <= pulse;


case decide1 is     --agar labe tashkhis dade shod flag ra yek kon
when "01" =>
flag1 <= '1';
when others =>
flag1 <= '0';
end case;



--  agar flag yek shaved yek palse sathe balaye farzi ( q1 = 1 )  ijad
kon va be andazeye yek periode yek signal 1KHz An ra edAme bede.
if flag1 = '1' then
counter1 <= 50000; --q1  ra yek kon
q1 <= '1';
else
if counter1 = 0 then  -- agar countere 1  sefr shod q1 ra sefr kon
q1 <= '0';
else    -- dar gheire in surat countere 1 ra yeki kam kon va q1 ra dar
halite yek negah dar
counter1 <= counter1 - 1 ;
q1 <= '1';
end if;
end if;

-- monostable 2 , mesle monostable 1
decide2(1) <= decide2(0);
decide2(0) <= q1;


case decide2 is
when "01" =>
flag2 <= '1';
when others =>
flag2 <= '0';
end case;
```

```
--  agar flag yek shaved yek palse sathe paeine farzi ( q2_bar = 0 )
ijad kon va be andazeye yek periode yek signal 1KHz An ra edAme bede.



if flag2 = '1' then
counter2 <= 50000; --q2_bar  ra sefr kon
q2_bar <= '0';
else
if counter2 = 0 then -- agar countere 2 sefr shod q2_bar ra yek kon

q2_bar <= '1';
else -- dar gheire in surat countere 2 ra yeki kam kon va q2_bar ra dar
halte sefr negah dar
counter2 <= counter2 - 1 ;
q2_bar <= '0';
end if;
end if;



end if;


------- D flip flop     , zamani ke periode signal vorudi kamtar az
zamane  fa'aliate mono stable shaved q1 dar sathe yek baghi mimanad, va
q1-bar sefr migardad, va chon q1 be vorudie pulse monostable 2 vared
migardad, ba tavajjoh be khassiate monostable agar vorudi hamishe 1 ya
hamishe 0 bashad q_bar 1 shode va q sefr migardad ke ba tavajjoh be proje
agar q1 hamishe 1 shavad q2_bar hamishe 1 migardad va agar in sharayet
bargharar shod dff khorujiash yek mishavad, chon q2_bar be clk e dff va
q1 be reset e dff mottasel ast

if (clk='1' ) then
if (q2_bar = '1' ) then   -- vorudie clk
if (q1 = '1') then --vorudie reset
q <= '1';
end if;
else
q<='0';
end if;
end if;


end process;

end Behavioral;
```

۳۴- کی پد و PPI و موتور پله ای

شماتیک مدار:



برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
```

```
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- synopsys translate_off
library UNISIM;
use UNISIM.Vcomponents.ALL;
-- synopsys translate_on

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ppi is
    Port (
                          -- Motor rotate direction
                       dir : in std_logic;
                         -- seven sgement Output
                         Seg : out std_logic_vector(7 downto 0);
                         --led out for digit & Steper motor out
                         led : out std_logic_vector(7 downto 0);
                         -- Data port to 8255
                         data : inout std_logic_vector(7 downto 0);
           --PPI 8255 program mode
                    a : out std_logic_vector(1 downto 0);
           rd : out std_logic;
           wr : out std_logic;
           clk : in std_logic;
           reset : in std_logic);

end ppi;

architecture Behavioral of ppi is

   component IOBUF
      port ( I  : in    std_logic;
             IO : inout std_logic;
             O  : out   std_logic;
             T  : in    std_logic);
   end component;

   constant rd_wr_delay : integer := 10; -- 10 us
   type state is (init, write, read, normal);
   signal current : state;
   signal part : integer;
   signal direction : std_logic;
   signal temp, data_in, data_out : std_logic_vector (7 downto 0);
             --motor speed data
      signal  Speed :  std_logic_vector(7 downto 0);
      -- Motor connection dcba
   signal  motor_abcd : std_logic_vector(3 downto 0);
   signal  MCount : std_logic_vector(23 downto 0);
   constant debounce_time : integer := 200000; --200 ms
   signal debounce_flag : std_logic;
   signal  column_encoded, row_encoded,column,Column_i,column_before,
digit : std_logic_vector (3 downto 0);
   signal row,row_i: std_logic_vector (3 downto 0):="1110";
   signal Keycount,RowCount : integer;
   -- keybord presed indicator
   signal key_pressed : std_logic:='0';
```

```
begin
   process (clk, reset)
      variable PPIcount : integer;
   begin
      if reset = '1' then
                -- reet all datas
         rd <= '1';
         wr <= '1';
         part <= 0;
         current <= init;
         PPIcount := 0;
            motor_abcd <= "0011";
         MCount <= (others => '0');
                -- set digit to 0
            digit <= "0000";
                --Set Keyboard Scanner
            row <= "1110";
                -- Debounce bitwin 2 key press
            debounce_flag <= '0';
                --Debounce Counter
            Keycount <= 0;

      elsif clk = '1' and clk'event then
            -- steper motor controler
         if Mcount < "0000"&speed & "000000000000" then
            Mcount <= mcount + 1;
         else
                  if dir = '0' then
                        --if  dir is low make clock wise     rotation
                        motor_abcd <= motor_abcd(2 downto 0) &
                              motor_abcd (3);
                  else
                        -- make none clock wise rotation
                        motor_abcd <= motor_abcd (0) &
                              motor_abcd(3 downto 1);
                  end if;
                  Mcount <= (others => '0');
         end if;
            -- keypad driver
                   if KeyCount>0 then
                         key_Pressed<='1';
                         KeyCount<=KeyCount-1;
                   else
                      Debounce_flag<='0';
                         Key_Pressed<='0';
                   end if;
                   if Debounce_flag='0' then
                      if Column/="1111" then
                        --encode   keboard row and column to its key code
digit
                        digit <= column_encoded + row_encoded;
                           Column_i<=Column;
                              Row_i<=Row;
                              Debounce_Flag<='1';
                              KeyCount<=Debounce_time;
                         end if;
                   end if;
            -- PPI 8255 driver
         case current is
         when init =>
```

```
                -- set program mode
                      a <= "11";
                      direction <= '0';
                      data_out <= "10011000";  -- pa,Pc Hi : in   pb,
pc Low : out
                      PPIcount := rd_wr_delay;
                      current <= write;
            when normal =>
                      -- programming
                      if part = 0 then
                            -- read Speed Value from Port A
                      a <= "00";
                      direction <= '1';
                      PPIcount := rd_wr_delay;
                      current <= read;
                      part <= 1;
                      elsif part = 1 then
                                  -- Set Speed Value
                                  Speed<=Data_In;
                                  -- Set Steper motor drive value to
port B
                            a <= "01";
                            direction <= '0';
                            data_out <= "0000"&motor_abcd;
                            PPIcount := rd_wr_delay;
                            current <= write;
                            part <= 2;
                                   row <= row(2 downto 0) & row(3);
                            elsif part = 2 then
                                        --write key Rows to Port c
                                  a <= "10";
                                  direction <= '0';
                                  data_out <= "1111"&Row;
                                  PPIcount := rd_wr_delay;
                                  current <= write;
                                  part <= 3;
                                  elsif part = 3 then
                                              -- Read Key Colomn from
port C
                                        a <= "10";
                                        direction <= '1';
                                        PPIcount := rd_wr_delay;
                                        current <= read;
                                        part <= 4;
                                        elsif Part=4 then
                                                    Column<=temp(7
downto 4);
                                                    Part<=0;
                                        end if;
                -- write program byte
            when write =>
                      if PPIcount /= 0 then
                      if PPIcount < rd_wr_delay / 2 then
                            wr <= '1';
                      else
                                  wr <= '0';
                      end if;
                            PPIcount := PPIcount - 1;
                      else
                      current <= Normal;
```

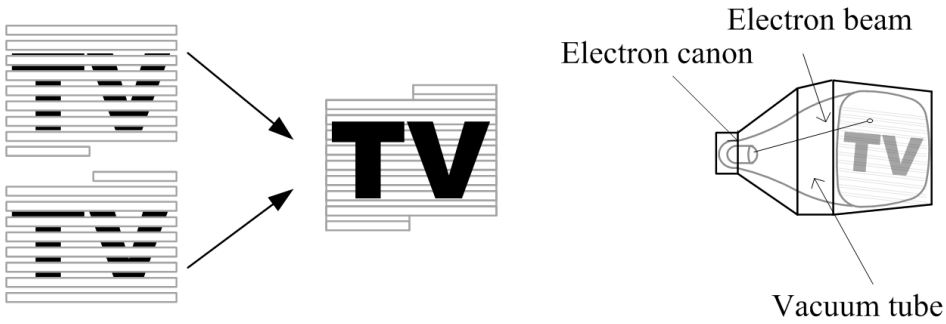
```
            end if;
            when read =>
                        if PPIcount /= 0 then
                        if PPIcount < rd_wr_delay / 2 then
                              rd <= '1';
                        else
                                    rd <= '0';
                                    temp <= data_in;
                        end if;
                        PPIcount := PPIcount - 1;
                        else
                        current <= normal;
            end if;
         end case;
      end if;
   end process;
   io_block_8 : for j in 0 to 7 generate
      io_block : iobuf port map (data_out(j),data(j),
data_in(j), direction);
   end generate;
      -- colomun encoding
   column_encoded <= "0000" when column = "1110" else
                     "0001" when column = "1101" else
                     "0010" when column = "1011" else
                     "0011" when column = "0111" else
                     "0000";
      -- make row encoding
   row_encoded <= "0000" when row = "1110" else
                  "0100" when row = "1101" else
                  "1000" when row = "1011" else
                  "1100" when row = "0111" else
                  "0000";
   --show keypad presed  digit 0n led
      led(3 downto 0) <= digit;
   --show Steper motor abcd out on led
      led(7 downto 4) <= Motor_abcd;
      --Send seven segement code for digit
   seg <= "11000000" when digit = "0000" else
                   "11111001" when digit = "0001" else
                         "10100100" when digit = "0010" else
                         "10110000" when digit = "0011" else
                         "10011001" when digit = "0100" else
                         "10010010" when digit = "0101" else
                         "10000010" when digit = "0110" else
                         "11111000" when digit = "0111" else
                         "10000000" when digit = "1000" else
                         "10010000" when digit = "1001" else
                         "01000000" when digit = "1010" else
                         "01111001" when digit = "1011" else
                         "00100100" when digit = "1100" else
                      "00110000" when digit = "1101" else
                      "00011001" when digit = "1110" else
                      "00010010" when digit = "1111";

end Behavioral;
```

۳۵- اتصال تلویزیون به FPGA و نمایش روی آن

هدف، اتصال یک تلویزیون از طریق ورودی Video به میکرو می‌باشد تا میکرو بتواند در صفحه چیزی را نشان دهد. در این مدار، تلویزیون به صورت سیاه و سفید راه‌اندازی شده که همان طور که می‌دانید، در سیستم سیاه و سفید، علاوه بر این دو رنگ، رنگهای دیگری بین ایندو داریم که انواع رنگهای خاکستری را تشکیل می‌دهند که شدت نورشان متفاوت است (بین خاکستری خیلی روشن که تقریباً در سطح سفید است و خاکستری خیلی تیره که در سطح سیاه است). این مدار، روی صفحه تلویزیون، پنج ستون سیاه و سفید نشان می‌دهد که دو ستون سفید آن، بین درجه‌های مختلف خاکستری، تغییر رنگ می‌دهند.

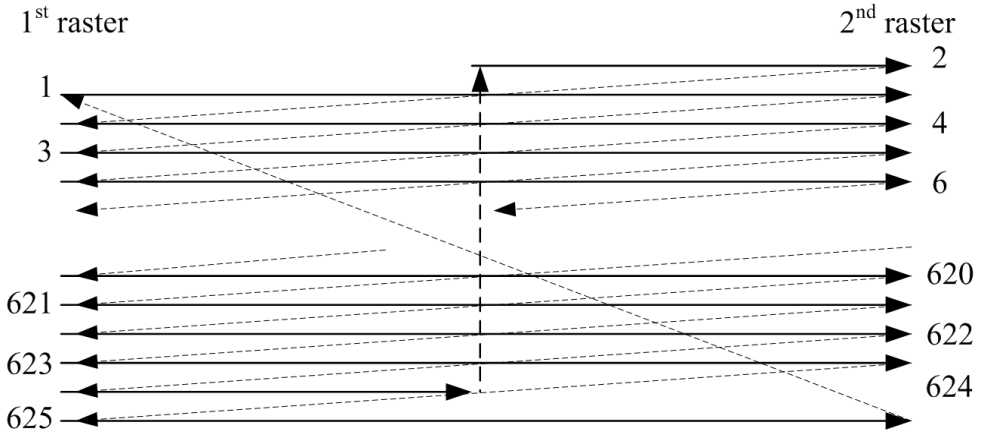
آشنایی با نحوه کار تلویزیون:

تفنگ الکترونی صفحه تلویزیون را در دو نوبت پیمایش می‌کند (روش Interlaced)، همانگونه که در شکل ۱ مشخص است، تصویر در دو مرحله جداگانه تولید می‌شود.



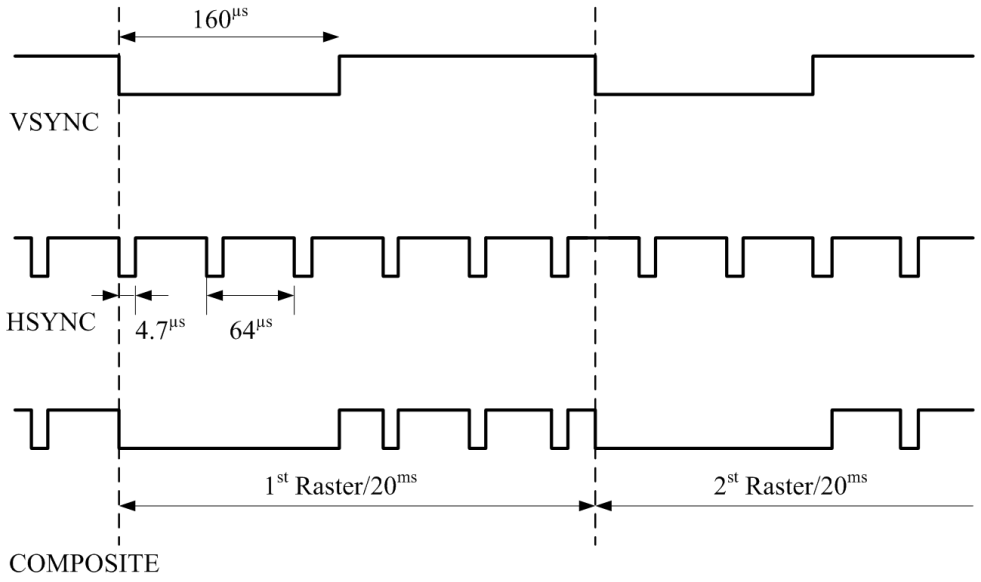
شکل ۱ ساختار داخلی تلویزیون و نحوه پیمایش صفحه

مطابق شکل ۲، ابتدا خطوط فرد پیمایش شده سپس تفنگ الکترونی به بالای صفحه بازگشته و خطوط زوج را پیمایش می‌کند، بدین ترتیب تصویر شکل می‌گیرد.

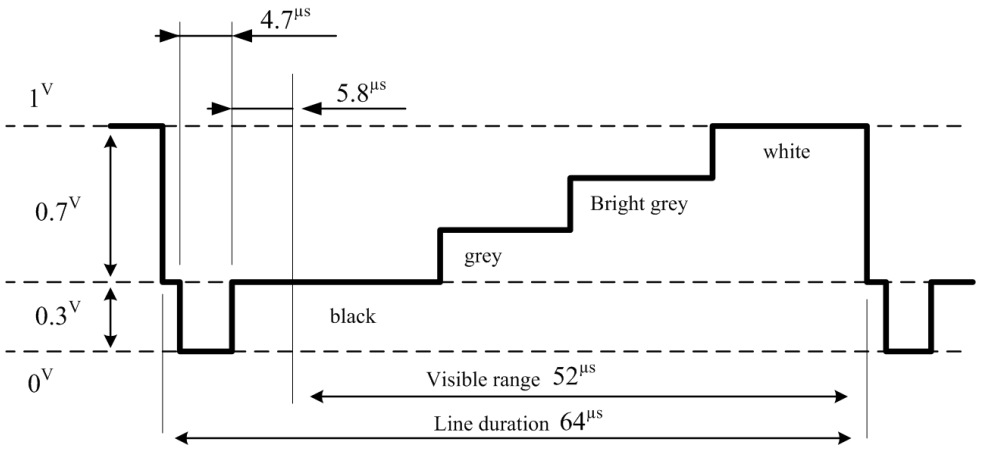


شکل ۲ خطوط فرد و زوج و نحوه پیمایش

شکل ۳ نحوه ارسال سیگنالهای مورد نیاز را نشان می‌دهد و سطوح ولتاژ برای تولید سیگنال‌ها و شدت روشنایی در شکل ۴ نشان داده شده است.



شکل ۳ سیگنالها و فرکانسهای مورد نیاز برای تولید تصویر



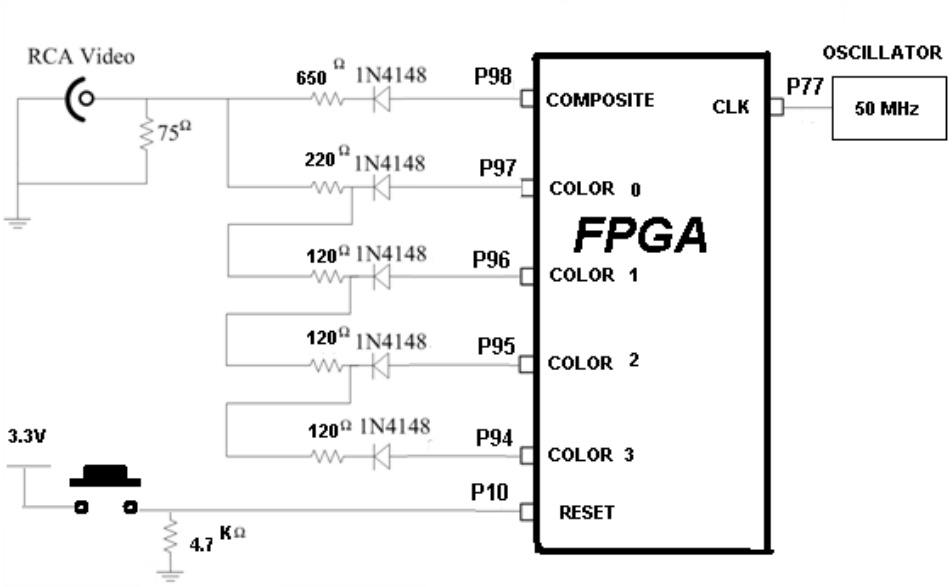
شکل ۴ ولتاژهای مربوط به تنظیم شدت نور

تلویزیون فقط از یک پایه برای دریافت فرکانس HSYNC، VSYNC و رنگها استفاده می‌کند. هنگامی که یک سطر به تلویزیون ارسال می‌شود، ابتدا به مدت 4.7µs خط 0V شده، سپس به مدت 5.8µs مقدار 0.3V روی خط قرار می‌گیرد (HSYNC). سپس در مدت 52µs رنگها در سطح ولتاژ 0.3V تا 1V ارسال می‌گردند (پیکسلها). مجموع زمان HSYNC برابر 64µs است.

در انتهای صفحه اول (خطوط فرد) خط آخر $32^{\mu s}$ زمان دارد و خط اول صفحه دوم نیز $32^{\mu s}$ زمان دارد. بنابراین بین دو فریم که یک تصویر را می‌سازند باید $32^{\mu s}$ اختلاف فاز وجود داشته باشد.

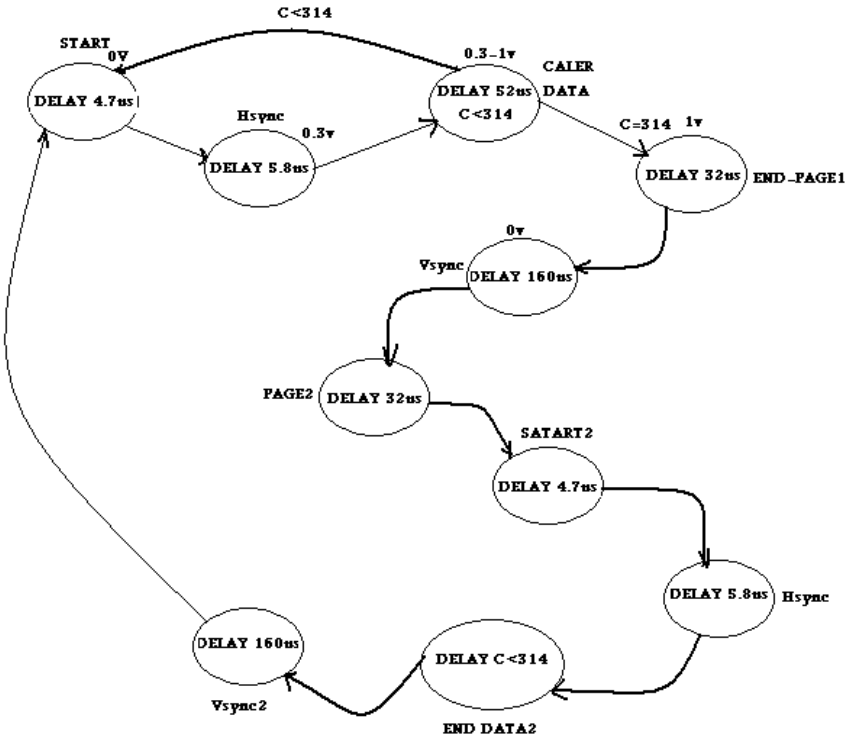
برای شروع هر صفحه $160^{\mu s}$ خط صفر می‌شود. (VSYNC)

در این مدار برای تولید فرکانس های افقی و عمودی تلویزیون از تایمر صفر استفاده شده است. در زیر برنامه color صفحه تلویزیون به پنج ستون (دو ستون سفید و سه ستون مشکی) تقسیم شده و رنگ دو ستون سفید بعد از مدت زمان معینی از سفید به خاکستری تغییر می‌کند و این کار تکرار می‌شود.



شکل ۵ مدار اتصال تلویزیون به FPGA

نمودار STATE DIAGRAM:



برنامه در ادامه آورده شده است. توضیحات لازم در مورد چگونگی کار برنامه و زیر برنامه‌هـای آن در خـود برنامه قید شده است.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity TV is
    Port ( clk : in std_logic;
           reset : in std_logic;
           color : out std_logic_vector(3 downto 0);
           composite : out std_logic);
end TV;

architecture Behavioral of TV is
type tv_state
is(start_page1,hsync_1,color_data1,end_page1,vsync_1,start_page2,page2_47
us,hsync_2,color_data2,vsync_2);--
signal bri:std_logic_vector(3 downto 0);

begin
process(clk,reset)
variable state:tv_state:=start_page1;
variable count:integer;
variable bri_counter:integer;

variable line_counter:integer;
begin
      if(reset='1')then
      bri<="1000";
      bri_counter:=0;
      state:=start_page1;
      count:=0;
      elsif (clk='1' and clk'event)then
      count:=count+1;
      bri_counter:= bri_counter+1;
      if bri_counter=0 then
            bri<="1000";
      elsif bri_counter=50000000 then
            bri<="0100";
      elsif bri_counter=100000000 then
            bri<="0010";
      elsif bri_counter=150000000 then
            bri<="0001";
      elsif bri_counter=200000000 then
            bri_counter:=0;
      end if;

      case  state is
      when  start_page1 =>    --4.7us
```

```
        if(count<235)then
              state:=start_page1;
                    composite<='0';
              color<="0000";
        elsif(count=235)then
              state:=hsync_1;
        end if;
  when  hsync_1           =>    --5.8us
        if(count>235 and count<525)then
              state:=hsync_1;
              composite<='1';
              color<="0000";
        elsif(count=525)then
              state:=color_data1;
              count:=0;
        end if;
  when  color_data1 =>    --53.5us
        if(count>0 and count<2675)then
              state:=color_data1;
              composite<='1';
              case count is
              when 0      =>    color<="0000";
              when 525=>  color<=bri;
              when 1070=> color<="0000";
              when 1605=> color<=bri;
              when 2140=> color<="0000";
              when others => null;
        end case;

        elsif(count=2675)then
              line_counter:=line_counter+1;
              color <= "0000";
              composite<='1';         --
              if (line_counter<314)then
                    state :=    start_page1;
                    count := 0;
              elsif (line_counter=314)then
                    state :=    end_page1;
                    count :=    0;
                    line_counter := 0;
              end if;
        end if;
  when  end_page1   =>
        if(count<1600 )then
              state :=    end_page1;
              composite   <=    '1';  --or '0'
              color <=    "0000";           --or"1111"
         elsif(count=1600)then  --or >=
              state := vsync_1;
              composite   <=    '0';
              color <= "0000";
              count :=    0;
         end if;
  when  vsync_1           =>
        if (count<8000)then
              state := vsync_1;
              composite   <=    '0';
              color <= "0000";
        elsif(count=8000)then
              state :=    start_page2;
```

```
            composite   <=    '1';
            color <= "0000";
            count := 0;
      end if;
when  start_page2 =>
      if(count<1600)then
            state :=    start_page2;
            composite   <=    '1';
            color <=    "0000";
      elsif(count=1600)then
            state :=    page2_47us;
            composite   <=    '0';
            color <=    "0000";
            count := 0;
      end if;
when  page2_47us  =>
      if(count<235)then
            state :=    page2_47us;
            composite   <=    '0';
            color <=    "0000";
      elsif(count=235)then
            state :=    hsync_2;
            composite   <=    '1';
            color <=    "0000";
            count :=    0;
      end if;
when  hsync_2           =>
      if(count<290)then
            state :=    hsync_2;
            composite   <=    '1';
            color <=    "0000";
      elsif(count=290)then
            state :=    color_data2;
            composite   <=    '1';
            color <=    "0000";
            count :=    0;
      end if;
when  color_data2 =>
      if(count>0 and count<2675)then
            state:=color_data2;
            composite<='1';
            case count is
            when 0      =>    color<="0000";
            when 525    =>    color<=bri;
            when 1070=> color<="0000";
            when 1605=> color<=bri;
            when 2140=> color<="0000";
            when others => null;
            end case;
      elsif(count=2675)then
            line_counter:=line_counter+1;
            color <= "0000";
            composite<='1';
            if (line_counter<314)then
                  state :=    page2_47us;
                  count := 0;
            elsif (line_counter=314)then
                  state :=    vsync_2;
                  count :=    0;
                  line_counter := 0;
```

```
                  end if;
            end if;
      when  vsync_2           =>
            if (count<8000)then
                  state := vsync_2;
                  composite   <=    '0';
                  color <= "0000";
            elsif(count=8000)then
                  state :=    start_page1;
                  composite   <=    '1';
                  color <= "0000";
                  count := 0;
            end if;
      end case;
      end if;
end process;
end Behavioral;
```

۳۶- اتصال تلویزیون به FPGA و فرستادن متن به آن

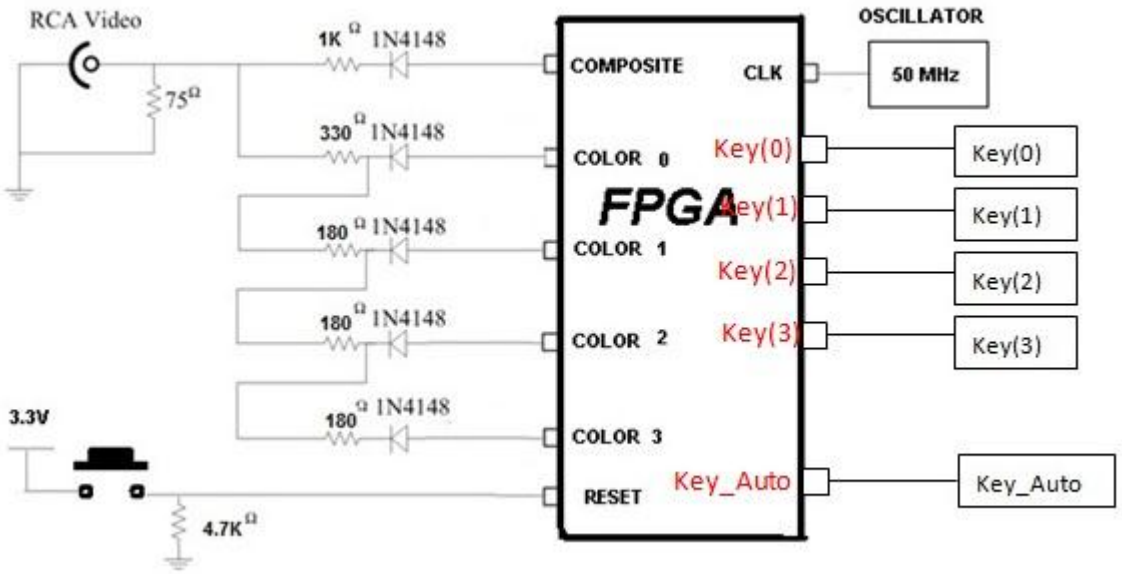
مقدمه :

در این فصل به بررسی و تجزیه و تحلیل برنامه فرستادن متن به از FPGA بـه TV بـا زبـان VHDL مـی پردازیم همچنین به بررسی نحوه روشن کردن پیکسـل هـای تلویزیـون و چگـونگی پیمـایش سـطری در تلویزیون (روش Interlaced) می پردازیم .

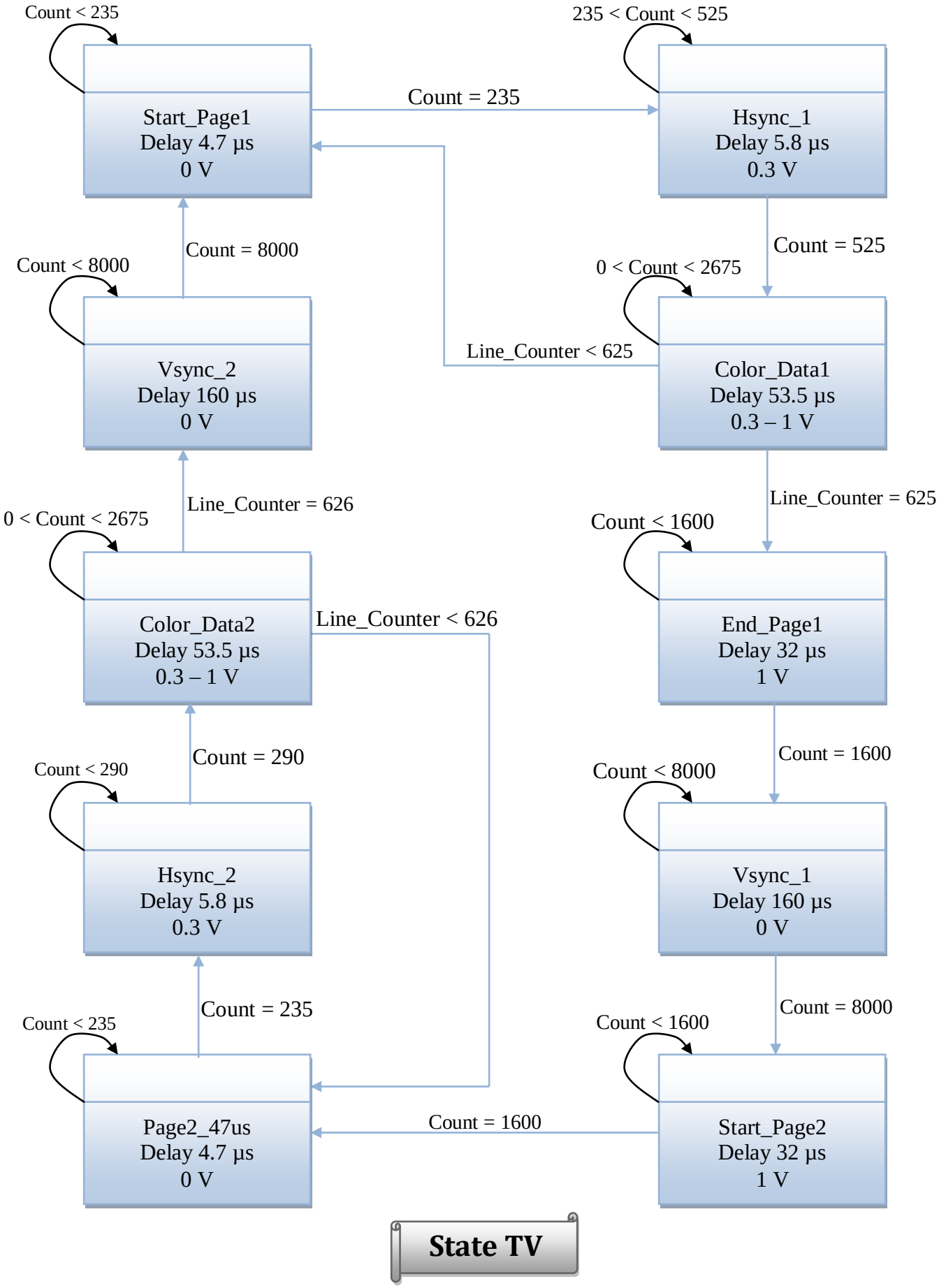
اتصال تلویزیون و نمایش روی آن :

هدف از پروژه، اتصال یک تلویزیون از طریق پورت ورودی Video به FPGA می باشد تا FPGA بتوانـد در صفحه متن های مختلفی را در نقاط مختلف (نقاط دلخواه) نمایش دهد. در این مدار، تلویزیون به صـورت سیاه و سفید را ه اندازی شده که همان طور که می دانید ، در سیستم سیاه و سفید ، علاوه بر این دو رنگ ، رنگهای دیگری بین این دو داریم که انواع رنگهای خاکستری را تشکیل مـی دهنـد کـه شـدت نورشـان متفاوت است (بین خاکستری خیلی روشن که تقریباً در سطح سفید است و خاکستری خیلی تیـره کـه در سطح سیاه است) . این مدار ، روی صفحه تلویزیون ، متون مورد نظر را در نقاط دلخواه نشان می‌دهد بدین صورت که در ابتدا تمام حروف و اعداد لاتین (A-Z و 0-9) در Character Box هـایی در انـداره ۷*۹ در آرایه ها ذخیره گردیده است برنامه پس از دیدن متن مورد این آرایه ها را یافته و پس از ترکیب آنها با هم در صفحه تلویزیون نمایش می دهد .

مدار شکل زیر را ببندید و نتیجه را مشاهده کنید.



کلید key: برای تغییر رنگ بصورت دستی
کلید key_Auto:برای تغییر رنگ بصورت اتوماتیک


Count < 235
Start_Page1
Delay 4.7 µs
0 V
Count = 235
235 < Count < 525
Hsync_1
Delay 5.8 µs
0.3 V
Count = 525
Count < 8000
Count = 8000
Vsync_2
Delay 160 µs
0 V
0 < Count < 2675
Color_Data1
Delay 53.5 µs
0.3 – 1 V
Line_Counter < 625
Line_Counter = 625
0 < Count < 2675
Line_Counter = 626
Color_Data2
Delay 53.5 µs
0.3 – 1 V
Count < 1600
End_Page1
Delay 32 µs
1 V
Line_Counter < 626
Count = 1600
Count < 290
Count = 290
Hsync_2
Delay 5.8 µs
0.3 V
Count < 8000
Vsync_1
Delay 160 µs
0 V
Count = 8000
Count < 235
Count = 235
Page2_47us
Delay 4.7 µs
0 V
Count < 1600
Start_Page2
Delay 32 µs
1 V
Count = 1600
State TV

برنامه در ادامه آورده شده است. توضیحات لازم در مورد چگونگی کار برنامه و زیر برنامه هـای آن در خـود برنامه قید شده است.

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.STD_LOGIC_ARITH.ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;
----------------------------------------------------------------------------------
--
```

--برای اتصال TV به FPGA ما نیاز به چندین ورودی و خروجی داریم که در قسمت entity برنامه تعریف می شوند.یک ورودی به نام clk و reset و یک خروجی چهار بیتی به نام color که این متغیر بـرای ایجـاد رنگ (روشن کردن پیکسل) به کار می رود. ویک خروجی تک بیتی به نام composite برای سطح ولتاژ بـه کار می رود.کلید key برای تغییـر رنـگ بصـورت دسـتی و کلیـدkey_Auto بـرای تغییـر رنـگ بصـورت اتوماتیک--

```
Entity tv is
        Port (clk: in std_logic;
                reset: in std_logic;
                color: out std_logic_vector(3 downto 0);
                composite: out std_logic);
                key_Auto:in std_logic;
           key:std_logic_vector(3 downto 0);
End tv;
----------------------------------------------------------------------------------
--
Architecture Behavioral of tv is
```

--در این قسمت ما چندین state تعریف کرده ایم که کاربرد این state هـا را در طـول برنامـه بـه طـور

کامل توضیح می دهیم. --

type tv_state is(start_page1, hsync_1, color_data1, end_page1, vsync_1, start_page2, page2_47us, hsync_2, color_data2, vsync_2);

--در این قسمت دو ثابت به نامهای m و n تعریف شده که از m برای تعیین تعداد سطرهای TV و از n برای تعیین تعداد ستون های TV استفاده می شود که توسط کاربر وارد برنامه می شوند . تعداد سطرهای واقعی استاندارد TV در اینجا ۶۲۵ می باشد و ستون های واقعی استاندارد TV در اینجا ۵۳۵ می باشد. (تعداد سطر به ستون به نسبت ۴/۳ در نظر گرفته شده است)--

Constant m : integer := 627; --

Row

Constant n : integer := 535; --

Column

--ثابتهای m و n در این قسمت وارد آرایه می شوند و آرایه ای با ابعاد m و n ساخته می شود که m سطر و n ستون می باشد.--

type rom_type is array (0 to m) of std_logic_vector(0 to n);

--در این قسمت یک آرایه دو بعدی ۹ در ۷ ساخته می شود.--

type charbox_2d is array (0 to 8) of std_logic_vector(0 to 6);

--در این قسمت یک آرایه سه بعدی ساخته می شود که از ۳۶ آرایه ۹ در ۷ تشکیل شده است که برای قرار گرفتن ۲۶ حرف انگلیسی و ارقام ۰ تا ۹ (۱۰ رقم) و ۱ فضای خالی استفاده می شود.--

type charbox_3d is array (0 to ۳۶) of charbox_2d;

--در این قسمت چون مشخص نیست که کاربر چند کلمه را می خواهد وارد کند از آرایه نامحدود استفاده شده است که در ادامه توضیحات لازم داده می شود--

type str_type is array (natural range<>, natural range<>) OF CHARACTER;

--در این قسمت چون مشخص نیست که طول کلمه ای را که کاربر می خواهد وارد کند چقدر اسـت ، از آرایه نامحدود استفاده شده است که در ادامه توضیحات لازم داده می شود--

type len is array (natural range<>) OF integer;

--این بخش اول ورودی برنامه می باشد، یعنی بخشی که کاربر مشخص می کند که کلمات چـه چیـزی و طول آنها چقدر باشد ، در تابع str_type کلماتی را که می خواهیم را می نویسیم و با (,) از همدیگر جدا می کنیم و در تابع str_len طول کلماتی را که در تابع str_type نوشته ایم را بـه همـان ترتیـب وارد مـی کنیم .--

---------- < User Input part 1 > ----------

--

constant str: str_type := ("This","is","a","test"); -- Araye characterie 2boadi namahdod

constant str_len: len := (4,2,1,4); -- Arayei namahdod az noe integer

---

--در این قسمت طول آرایه str در آرایه _array قرار می دهد.--

constant array_len : integer := str'length; -- toole arayeye str dar str_len gharar migirad

type int is array (0 to array_len-1) of integer;

--این بخش دوم ورودی برنامه می باشد ، یعنی بخشی که کاربر آدرس سطر و ستون (نقاط شروع کلمه)هر کلمه ای را که وارد کرده را در این آرایه ها وارد می کند کـه آرایـه row1 مربـوط بـه سـتون هـا و آرایـه column1 مربوط به سطر ها --

----------<User Input part 2>----------

--

constant row1 : int :=(50,50,80,80); -- Addresse satrhaye noghate shoro

```
constant column1 : int :=(0,90,0,60);          --      Addresse      sotunhaye
noghate shoro
-------------------------------------------------------------------------------------
--
signal rom : rom_type;                                          -- Rom
628*535
signal array_3d:charbox_3d;                                     -- Array Character
Box
signal state:tv_state;
signal type_color:std_logic_vector(3 downto 0);
signal clock:std_logic:='0';
```

--Count : یک شمارنده که از نوع integer می باشد.

line_counter : یک شمارنده که از نوع integer می باشد که برای شمارش سطرها به کار می رود.

counter1 : شمارنده ای که برای صفحه اول TV به کار می رود.

counter2 : شمارنده ای که برای صفحه دوم TV به کار می رود.

pixel_counter : شمارنده ای است که برای شمارش پیکسل ها می باشد.--

```
signal counter1,counter2,pixel_counter,count,line_counter:integer;
```

--در این قسمت می توان حروف الفبای انگلیسی را که تعداد آن ۳۶ حرف و ۱۰ رقم (رقم های ۰ تا ۹) همراه با یک فضای خالی که در حالت کلی ۳۶ کاراکتر باکس (Character Box) می باشد را در آرایه هایی که تعداد ستون های آنها ۷ ستون و تعداد سطر های آنها ۹ سطر می باشد را که بصورت بیت به بیت مقداردهی شده را مشاهده کرد.--

```
Begin
array_3d<= (
        ("0000000",    -- null
         "0000000",
         "0000000",
         "0000000",
         "0000000",
```

```
        "0000000",
        "0000000",
        "0000000",
        "0000000"),

       ("0011100",    -- A
        "0100010",
        "1000001",
        "1000001",
        "1111111",
        "1000001",
        "1000001",
        "1000001",
        "1000001"),

    …


       ("0111110",          -- 0
        "1000001",
        "1000001",
        "1000001",
        "1000001",
        "1000001",
        "1000001",
        "0111110"),
…
       ("1111111",          -- 9
        "1000001",
        "1000001",
        "1000001",
        "1111111",
        "0000001",
        "0000001",
        "0000001",
        "1111111"));
```

Process (clk, reset)

i-- و j و k در سه حلقه for تو در تو استفاده شده است که در ادامه بررسی می شوند.

row : آدرس سطر نقطه ای که می خواهیم متن را در آن نقطه نمایش دهیم.

column : آدرس ستون نقطه ای که می خواهیم متن را در آن نقطه نمایش دهیم.

```
variable row, column : int;
variable  i,c,j,k,str_counter,char_index:integer;
variable        secend :integer;
begin
--در این قسمت از برنامه در صورت یک بودن  reset (reset=1)به تمام سیگنال ها مقدار اولیه می دهیم..-
-


if(reset='1')then
--آدرس نقطه های شروع سطر ها
   row:=row1;                    --      Addresse     satrhaye noghate shoro

--آدرس نقطه های شروع ستون ها

   column:=column1;              --      Addresse     sotunhaye noghate shoro


   clock<='0';
   c:=0;
   i:=0;
   j:=0;
   k:=0;
   str_counter:=0;
   counter1<=0;
   counter2<=0;
   pixel_counter<=0;
   count<=0;
```

```
line_counter<=1;                                    -- First Odd Line
state<=start_page1;


 for str_counter in 0 to array_len-1 loop   -- be tedade matnhaye vared shode tavassote karbar tekrar mishavad


for i in 0 to str_len(str_counter)-1 loop

   case str(str_counter,i) is
    -- Checking Character For Specifying Index
    when 'A'|'a'=> char_index:=1;
    when 'B'|'b'=> char_index:=2;
    …
    when 'Z'|'z'=> char_index:=26;
    when '0'=> char_index:=27;
    …
    when '9'=> char_index:=36;
    when others=> char_index:=0;
   end case;
```

--در این قسمت کاراکتر باکس ها بیت به بیت درون rom کپی می شوند.--

```
   for j in 0 to 8 loop            -- Find Character and Send It to Rom
    for k in 0 to 6 loop
```

--چک کردن آخر خط که در صورتی که به انتهای خط برسد به بقیه نوشته ها را به خط بعدی ببرد--

```
        if (column(str_counter) < n-9) then   -- ta zamani ke be akhare khat nareside in ghesmat ejra mishavad
          rom(j+row(str_counter))(k+column(str_counter)) <= array_3d(char_index)(j)(k);
```

--این شرط زمانی رخ می دهد که به آخر خط رسیده باشد که بقیه نوشته ها را به خط بعدی می برد--

```
        else      -- zamani ke be akhare khat resid baghie matn dar khatte ba'd namayesh dade mishavad
          column(str_counter) := 0;
          row(str_counter) := row(str_counter) + 12;
          rom(j+row(str_counter))(k+column(str_counter)) <= array_3d(char_index)(j)(k);
        end if;
      end loop;
    end loop;


    column(str_counter):=column(str_counter)+9;    -- Column tv + 9 , for next character
  end loop;
  end loop;
```

--این قسمت با هر کلاک بالارونده فعال می شود و به متغییر count با هر کلاک یکی اضافه می شود.--

```
      elsif (clk='1' and clk'event)then
            count<=count+1;
```

--قبل از اینکه یک سطر به تلویزیون ارسال کنیم ابتدا باید به مدت 4.7us خط 0v شده،سپس به مدت 5.8us مقدار 0.3v روی خط قرار گیرد. با توجه به توضیح داده شده در state- start_page1 ابتدا باید سطح ولتاژ های مورد نظر را تولید کنیم تا بتوانیم یک سطر را ارسال کنیم که در قسمت ابتدای برنامه به

مدت 4.7us خط 0v تولید شده و در state=hsync_1 نیز به مدت 5.8us مقدار 0.3v روی خـط قـرار مـی گیرد.--

Case state is

---------- < **Page1 = Odd Lines** > ----------

--دراین state بـه مـدت 4.7us (میکروثانیـه) ولتـاژ 0v بـه TV ارسـال مـی شـود.('0'=<composite و color<="0000")--

when start_page1 => -- 4.7 µs

if (count<235)then

composite<='0';

color<="0000";

elsif(count=235)then

state<=hsync_1;

end if;

--دراین state به مـدت 5.8us (میکروثانیـه) ولتـاژ 0.3v بـه TV ارسـال مـی شـود. ('1'=<composite و color<="0000")--

when hsync_1=> --5.8µs

if(count>235 and count<525)then

composite<='1';

color<="0000";

elsif (count=525)then

state<=color_data1;

count<=0;

end if;

--در این قسمت از برنامه رنگ های تولید شده را می خواهیم در فاصله های مختلفی از هم ارسال کنیم.کل زمان فرستادن یک سطر از ۰ تا ۲۶۷۵ می باشد که معادل 53.5us (میکروثانیه) می باشد. زمانی که بخواهیم data یا داده را بفرستیم composite را یک می کنیم و از آنجایی که طول مانیتور ۲۶۷۵ می باشد برای اینکه رنگها را ارسال کنیم ابتدا ۲۶۷۵ را بر ۵ تقسیم کرده تا فاصله بین رنگها مشخص شود. در اینجا برای تولید رنگ سفید ولتاژ 0.3v (composite<='1' و color<="0000") و برای تولید رنگ مشکی ولتاژ 1v (composite<='1' و color<="1111") را استفاده کرده ایم.--

```
when color_data1 =>                                    -- 53.5
µs
if(count>0 and count<2675)then
    counter1<=counter1+1;
    composite<='1';
```

--در این قسمت بعد از هر ۵ کلاک (زمانی که counter1 از ۰ تا ۴ طی می کند) یک واحد به pixel_counter اضافه می شود، که pixel_counter نشان دهنده شماره پیکسل ها می باشد.--

```
if (counter1=4) then                                   -- 5 Clk For Each
Pixel
    counter1<=0;
    pixel_counter<=pixel_counter+1;                    -- Next
Pixel
if (pixel_counter<=n and line_counter<m) then
if (rom(line_counter)(pixel_counter)='1') then
    color <= type_color
else
    color <= "0000";                                   -- Black
Color
```

```
end if;
end if;
end if;
```

--در این قسمت از برنامه ،زمانی که count برابر ۲۶۷۵ شد یعنی یک سطر پوشش داده شده است و ما Counter مربوط به شمارش سطر را یک واحد افزایش می دهیم.--

ما در تلویزیون باید خطوط را دو بار پیمایش کنیم که ابتدا خطوط فرد و سپس خطوط زوج پیمایش می شود و از آنجایی که خطوط تلویزیون ۶۲۵ سطر می باشد ما این خطوط را تقسیم بر ۲ می کنیم تا تعداد خط های فرد و زوج مشخص شود. که در این قسمت از برنامه خطوط فرد پیمایش می شود و در انتهای این قسمت line_counter را ۲ می کنیم یعنی خطوط فرد کاملاً پوشش داده شده و دفعه بعد باید خطوط زوج را پوشش دهد.--

```
elsif(count=2675)then
line_counter<=line_counter+2;                    -- Next Odd
Line
        color <= "0000";
        composite <='1';
        pixel_counter <=0;
        count <= 0;
        counter1 <= 0;
if (line_counter<625)then
state <= start_page1;
elsif (line_counter=625)then
state <= end_page1;
        line_counter <= 2;                       -- First Even
Line
```

```
            end if;
            end if;
```

--دراین state بـه مـدت 32us (میکروثانیـه) ولتـاژ 0.3v بـه TV ارسـال مـی شـود. ('1'=>composite و "0000"=>color) (زمانی که نیاز است در انتهای صفحه اول کـه بـرای خطهـای فـرد مـی باشـد رعایـت شود)—

```
            when end_page1 =>
            if(count<1600 )then                                    -- 32
µs
                  composite <= '1';                                -- or
'0'
                  color <= "0000";                                 --
or"1111"
            elsif (count=1600)then                                 -- or
>=
            state <= vsync_1;
                  composite <= '0';
                  color <= "0000";
                  count <=     0;
            end if;
```

--در این state به مدت 160 میکرو ثانیه صبر می کنیم تا از آخرین سطر فرد به اولین سطر زوج بـرود. و ولتاژ 0v به TV ارسال می شود. ('0'=>composite و "0000"=>color)--

```
            when  vsync_1 =>
            if (count<8000)then                                    -- 160
µs
```

```
                composite <= '0';
                color <= "0000";
            elsif(count=8000)then
            state <=start_page2;
                composite <= '1';
                color <= "0000";
                count <= 0;
            end if;
```

---------- < Page2 = Even Lines > ----------

--

--دراین state بـه مـدت 32us (میکروثانیـه) ولتـاژ 0.3v بـه TV ارسـال مـی شـود. ('1'=>composite و "0000"=>color) (زمانی که نیاز است در ابتدای صفحه دوم که برای خطهای زوج می باشد رعایت شود)-

-

```
            when start_page2 =>
            if(count<1600)then                                    -- 32
µs
                composite <= '1';
                color <= "0000";
            elsif (count=1600)then
            state <= page2_47us;
                composite <= '0';
                color <= "0000";
                count <= 0;
            end if;
```

--دراین state به مدت 4.7us (میکروثانیه) ولتاژ 0v به TV ارسال می شود.('0'<=composite و color<="0000")--

```
when page2_47us =>                                    -- 4.7 µs
if (count<235)then
        composite <= '0';
        color <= "0000";
elsif(count=235)then
state <= hsync_2;
        composite <= '1';
        color <= "0000";
        count <= 0;
end if;
```

--دراین state به مدت 5.8us (میکروثانیه) ولتاژ 0.3v به TV ارسال می شود. ('1'<=composite و (color<="0000"--

```
when  hsync_2 =>                                      -- 5.8 µs
if(count<290)then
        composite<='1';
        color<="0000";
elsif(count=290)then
state<=color_data2;
        composite <= '1';
        color <= "0000";
        count <= 0;
end if;
```

--د ر این قسمت از برنامه رنگ های تولید شـده را مـی خـواهیم در فاصـله هـای مختلفـی از هـم ارسـال کنیم.کل زمان فرستادن یک سطر از ۰ تا ۲۶۷۵ می باشد که معادل 53.5us (میکروثانیه) می باشد. زمانی که بخواهیم data یا داده را بفرستیم composite را یک می کنیم و از آنجایی که طول مانیتور ۲۶۷۵ مـی باشد برای اینکه رنگها را ارسال کنیم ابتدا ۲۶۷۵ را بر ۵ تقسیم کرده تا فاصله بین رنگها مشخص شود. در اینجا برای تولید رنگ سفید ولتاژ 0.3v (composite<='1' و color<="0000") و برای تولید رنگ مشـکی ولتاژ 1v (composite<='1' و color<="1111") را استفاده کرده ایم.--

```
when color_data2 =>                                    -- 53.5
µs
if(count>0 and count<2675)then
    counter2 <=counter2+1;
    composite <='1';
```

--در این قسمت بعـد از هـر ۵ کـلاک (زمـانی کـه counter1 از ۰ تـا ۴ طـی مـی کنـد) یـک واحـد بـه pixel_counter اضافه می شود، که pixel_counter نشان دهنده شماره پیکسل ها می باشد.--

```
if (counter2=4) then                                   -- 5 Clk For Each
Pixel
    counter2<=0;
    pixel_counter<=pixel_counter+1;                    -- Next
Pixel
if (pixel_counter<n and line_counter<=m) then
if (rom(line_counter)(pixel_counter)='1') then
    color <= type_color
else
```

```
color <= "0000";                    -- Black
Color
end if;
end if;
end if;
```

--در این قسمت از برنامه ،زمانی که count برابر ۲۶۷۵ شد یعنی یک سطر پوشش داده شده است و ما Counter مربوط به شمارش سطر را یک واحد افزایش می دهیم.--

در این قسمت از برنامه خطوط زوج پیمایش می شود و در انتهای این قسمت line_counter را ۱ می کنیم یعنی خطوط زوج کاملاً پوشش داده شده و دفعه بعد باید خطوط فرد را پوشش دهد.—

```
elsif(count=2675)then
        line_counter<=line_counter+2;        -- Next Even
Line
        color <= "0000";
        composite<='1';
        pixel_counter<=0;
        count <= 0;
        counter2 <= 0;
if (line_counter<626)then
state <= page2_47us;
elsif (line_counter=626)then
state <= vsync_2;
        line_counter <= 1;                   -- First Odd
Line
end if;
end if;
```

--در این state به مدت 160 میکرو ثانیه صبر می کنیم تا از آخرین سطر زوج به اولین سطر فرد بـرود. و ولتاژ 0v به TV ارسال می شود. (composite<='0' و color<="0000")

```
            when vsync_2 =>                                    -- 160
µs
            if (count<8000)then
                  composite <= '0';
                  color <= "0000";
            elsif (count=8000)then
            state <= start_page1;
                  composite <= '1';
                  color <= "0000";
                  count <= 0;
            end if;
            End Case;


End if;
-----clk generation. For Osilator 50 MHz clock----

 if rising_edge(clock) then        --bedast avardan clk ba meghdare 1 sanie


      secend:=secend+1;
  if key_Auto ='1' then
      if secend<3 then  -- after 3 secend
            type_color<="0001";
            elsif secend<6 then -- after 6 secend
              type_color<="0010";  --change color
             elsif secend<9 then          -- after 9 secend
```

```
                type_color<="0100"; --change colo
                elsif secend<12 then -- after 12 secend
                type_color<="1000";  --change colo
                elsif secend<15 then -- after 15 secend
                type_color<="0000"; --change color
    else
        secend:=0;
        end if;


                    else
                        case key is     --change by key
                        when "1000" =>
                    type_color<="0000";
                         when "0100" =>
                    type_color<="1000" ;
                         when "0010" =>
                    type_color<="0100" ;
                          when "0001" =>
                    type_color<="0010" ;
                         when "0000" =>
                    type_color<="1111";
                        when others=> null;
                        end case;
                   end if;
                   end if;
End process;
End Behavioral;
```

کد کامل برنامه :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
----------------------------------------------------
entity tv is
    Port ( clk : in std_logic;
           reset : in std_logic;
           color : out std_logic_vector(3 downto 0);
                    key_Auto:in std_logic;
                    key:std_logic_vector(3 downto 0);
           composite : out std_logic);
end tv;
----------------------------------------------------
architecture Behavioral of tv is

type tv_state
is(start_page1,hsync_1,color_data1,end_page1,vsync_1,start_page2,page2_47
us,hsync_2,color_data2,vsync_2);

constant m : integer := 100;  -- Row  real value = 628 for
example(test)=100
constant n : integer := 100;  -- Column  real value = 535 ,for
example(test)=100

type rom_type is array (0 to m) of std_logic_vector(0 to n);
type charbox_2d is array (0 to 8) of std_logic_vector(0 to 6);
type charbox_3d is array (0 to 36) of charbox_2d;
type str_type is array (natural range<>, natural range<>) OF CHARACTER;
type len is array (natural range<>) OF integer;

--------------------------- < User Input part 1 > ----------------------
-------------------
constant str: str_type := ("This","is","a","test");  -- Araye characterie
2boadi namahdod
constant str_len: len := (4,2,1,4); -- Arayei namahdod az noe integer

----------------------------------------------------------------------------
---------------------


constant array_len : integer := str'length;     -- toole arayeye str dar
str_len gharar migirad
type int is array (0 to array_len-1) of integer;


----------------------------------<User Input part 2>--------------------
------------------
constant row1 : int :=(50,50,80,80);            --    Addresse
      satrhaye noghate shoro
constant column1 : int :=(0,90,0,60);           --    Addresse
      sotunhaye noghate shoro
----------------------------------------------------------------------------
----------------------
signal rom : rom_type;                                  -- rom 628*535
```

```
signal array_3d:charbox_3d;                              -- array character
box
signal state:tv_state;
signal counter1,counter2,pixel_counter,count,line_counter:integer;
signal type_color:std_logic_vector(3 downto 0);
signal clock:std_logic:='0';
begin

array_3d<=(

   ("0000000",       -- null
    "0000000",
    "0000000",
    "0000000",
    "0000000",
    "0000000",
    "0000000",
    "0000000",
    "0000000"),

    ("0011100",      -- A
    "0100010",
    "1000001",
    "1000001",
    "1111111",
    "1000001",
    "1000001",
    "1000001",
    "1000001"),

   ("1111100",       -- B
    "1000010",
    "1000001",
    "1000001",
    "1111110",
    "1000001",
    "1000001",
    "1000010",
    "1111100"),

   ("0011100",          -- C
    "0100010",
    "1000001",
    "1000000",
    "1000000",
    "1000000",
    "1000001",
    "0100010",
    "0011100"),

   ("1111100",          -- D
    "1000010",
    "1000001",
    "1000001",
    "1000001",
    "1000001",
    "1000001",
    "1000010",
    "1111100"),
```

```
("1111111",          -- E
 "1000000",
 "1000000",
 "1000000",
 "1111110",
 "1000000",
 "1000000",
 "1000000",
 "1111111"),

("1111111",          -- F
 "1000000",
 "1000000",
 "1000000",
 "1111110",
 "1000000",
 "1000000",
 "1000000",
 "1000000"),

("0011100",          -- G
 "0100010",
 "1000001",
 "1000000",
 "1000000",
 "1000111",
 "1000001",
 "0100011",
 "0011101"),

("1000001",          -- H
 "1000001",
 "1000001",
 "1000001",
 "1111111",
 "1000001",
 "1000001",
 "1000001",
 "1000001"),

("0111110",          -- I
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0111110"),

("0111110",          -- J
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "1001000",
 "0110000"),
```

```
("1000001",          -- K
 "1000010",
 "1000100",
 "1001000",
 "1110000",
 "1001000",
 "1000100",
 "1000010",
 "1000001"),

("0100000",          -- L
 "0100000",
 "0100000",
 "0100000",
 "0100000",
 "0100000",
 "0100000",
 "0111110"),

("1000001",          -- M
 "1100011",
 "1010101",
 "1001001",
 "1001001",
 "1001001",
 "1000001",
 "1000001",
 "1000001"),

("1000001",          -- N
 "1000001",
 "1100001",
 "1010001",
 "1001001",
 "1000101",
 "1000011",
 "1000001",
 "1000001"),

("0011100",          -- O
 "0100010",
 "1000001",
 "1000001",
 "1000001",
 "1000001",
 "1000001",
 "0100010",
 "0011100"),

("1111100",          -- P
 "1000010",
 "1000001",
 "1000001",
 "1111110",
 "1000000",
 "1000000",
 "1000000",
 "1000000"),
```

```
("0011100",          -- Q
 "0100010",
 "1000001",
 "1000001",
 "1000001",
 "1001001",
 "1000101",
 "0100010",
 "0011101"),

("1111100",          -- R
 "1000010",
 "1000001",
 "1000001",
 "1111110",
 "1001000",
 "1000100",
 "1000010",
 "1000001"),

("0111100",          -- S
 "1000010",
 "1000001",
 "0100000",
 "0011100",
 "0000010",
 "1000001",
 "0100001",
 "0011110"),

("0111110",          -- T
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0001000"),

("1000001",          -- U
 "1000001",
 "1000001",
 "1000001",
 "1000001",
 "1000001",
 "1000001",
 "0100010",
 "0011100"),

("1000001",          -- V
 "1000001",
 "1000001",
 "1000001",
 "1000001",
 "1000001",
 "0100010",
 "0010100",
 "0001000"),
```

```
("1000001",          -- W
 "1000001",
 "1000001",
 "1000001",
 "1000001",
 "1001001",
 "1001001",
 "1010101",
 "0100010"),

("1000001",          -- X
 "1000001",
 "0100010",
 "0010100",
 "0001000",
 "0010100",
 "0100010",
 "1000001",
 "1000001"),

("1000001",          -- Y
 "0100010",
 "0010100",
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0001000",
 "0001000"),

("1111111",          -- Z
 "1000001",
 "0000010",
 "0000100",
 "0001000",
 "0010000",
 "0100000",
 "1000001",
 "1111111"),

    ("0111110",           -- 0
     "1000001",
     "1000001",
     "1000001",
     "1000001",
     "1000001",
     "1000001",
     "1000001",
     "0111110"),

    ("0001000",           -- 1
     "1111000",
     "0001000",
     "0001000",
     "0001000",
     "0001000",
     "0001000",
     "0001000",
     "1111111"),
```

```
 ("1111111",           -- 2
 "0000001",
 "0000001",
 "0000001",
 "1111111",
 "1000000",
 "1000000",
 "1000000",
 "1111111"),

("1111111",           -- 3
 "0000001",
 "0000001",
 "0000001",
 "1111111",
 "0000001",
 "0000001",
 "0000001",
 "1111111"),

("1000001",           -- 4
 "1000001",
 "1000001",
 "1000001",
 "1111111",
 "0000001",
 "0000001",
 "0000001",
 "0000001"),

("1111111",           -- 5
 "1000000",
 "1000000",
 "1000000",
 "1111111",
 "0000001",
 "0000001",
 "0000001",
 "1111111"),

("1111111",           -- 6
 "1000000",
 "1000000",
 "1000000",
 "1111111",
 "1000001",
 "1000001",
 "1000001",
 "1111111"),


 ("1111111",          -- 7
 "1000001",
 "0000010",
 "0000100",
 "0001000",
 "0010000",
 "0100000",
 "1000000",
 "1000000"),
```

```
       ("1111111",            -- 8
        "1000001",
        "1000001",
        "1000001",
        "1111111",
        "1000001",
        "1000001",
        "1000001",
        "1111111"),

       ("1111111",            -- 9
        "1000001",
        "1000001",
        "1000001",
        "1111111",
        "0000001",
        "0000001",
        "0000001",
        "1111111"));



process(clk,reset,clock)
variable row, column : int;
variable i,c,j,k,str_counter,char_index:integer;
variable    secend :integer;

begin

  if(reset='1')then

    row:=row1;                                          --    Addresse
      satrhaye noghate shoro

    column:=column1;                                  --    Addresse
      sotunhaye noghate shoro
       clock<='0';
    c:=0;
    i:=0;
    j:=0;
    k:=0;
    str_counter:=0;
    counter1<=0;
    counter2<=0;
    pixel_counter<=0;
    count<=0;
    line_counter<=1;                                  -- First Odd Line
    state<=start_page1;


   for str_counter in 0 to array_len-1 loop   -- be tedade matnhaye vared
shode tavassote karbar tekrar mishavad

    for i in 0 to str_len(str_counter)-1 loop

        case str(str_counter,i) is
            -- Checking Character For Specifying Index
          when 'A'|'a'=> char_index:=1;
          when 'B'|'b'=> char_index:=2;
```

```
          when 'C'|'c'=> char_index:=3;
          when 'D'|'d'=> char_index:=4;
          when 'E'|'e'=> char_index:=5;
          when 'F'|'f'=> char_index:=6;
          when 'G'|'g'=> char_index:=7;
          when 'H'|'h'=> char_index:=8;
          when 'I'|'i'=> char_index:=9;
          when 'J'|'j'=> char_index:=10;
          when 'K'|'k'=> char_index:=11;
          when 'L'|'l'=> char_index:=12;
          when 'M'|'m'=> char_index:=13;
          when 'N'|'n'=> char_index:=14;
          when 'O'|'o'=> char_index:=15;
          when 'P'|'p'=> char_index:=16;
          when 'Q'|'q'=> char_index:=17;
          when 'R'|'r'=> char_index:=18;
          when 'S'|'s'=> char_index:=19;
          when 'T'|'t'=> char_index:=20;
          when 'U'|'u'=> char_index:=21;
          when 'V'|'v'=> char_index:=22;
          when 'W'|'w'=> char_index:=23;
          when 'X'|'x'=> char_index:=24;
          when 'Y'|'y'=> char_index:=25;
          when 'Z'|'z'=> char_index:=26;
      when '0'=> char_index:=27;
          when '1'=> char_index:=28;
          when '2'=> char_index:=29;
          when '3'=> char_index:=30;
          when '4'=> char_index:=31;
          when '5'=> char_index:=32;
          when '6'=> char_index:=33;
          when '7'=> char_index:=34;
          when '8'=> char_index:=35;
          when '9'=> char_index:=36;
          when others=> char_index:=0;
        end case;

        for j in 0 to 8 loop              -- Find Character and Send It
to Rom
          for k in 0 to 6 loop
            if (column(str_counter) < n-9) then   -- ta zamani ke be
akhare khat nareside in ghesmat ejra mishavad
              rom(j+row(str_counter))(k+column(str_counter)) <=
array_3d(char_index)(j)(k);
            else      -- zamani ke be akhare khat resid baghie matn dar
khatte ba'd namayesh dade mishavad
              column(str_counter) := 0;
              row(str_counter) := row(str_counter) + 12;
              rom(j+row(str_counter))(k+column(str_counter)) <=
array_3d(char_index)(j)(k);
            end if;
          end loop;
        end loop;

        column(str_counter):=column(str_counter)+9;       -- Column tv +
9 , for next character
    end loop;
    end loop;
```

```
--------------------------------------------------------------------------
----------------------

  elsif     (clk='1' and clk'event)then
        c :=c+1;
      if(c = 25000000) then
        clock <= not clock;
        c :=1;
      end if;




    count<=count+1;

    case    state is

---------------------------------- < Page1 = Odd Lines > -----------------
-----------------

      when  start_page1 =>
      -- 4.7 us
        if(count<235)then
          composite<='0';
          color<="0000";
        elsif(count=235)then
          state<=hsync_1;
        end if;

      when  hsync_1           =>
            -- 5.8us
        if(count>235 and count<525)then
          composite<='1';
          color<="0000";
        elsif(count=525)then
          state<=color_data1;
          count<=0;
        end if;

      when  color_data1 =>
      -- 53.5 us
        if(count>0 and count<2675)then
          counter1<=counter1+1;
          composite<='1';
          if (counter1=4) then                        -- 5 Clk For Each
Pixel
            counter1<=0;
            pixel_counter<=pixel_counter+1;           -- Next Pixel

            if (pixel_counter<=n and line_counter<m) then

              if (rom(line_counter)(pixel_counter)='1') then
                color <= type_color;
              else
                 color <= "0000";                           -- Black
Color
              end if;
            end if;
```

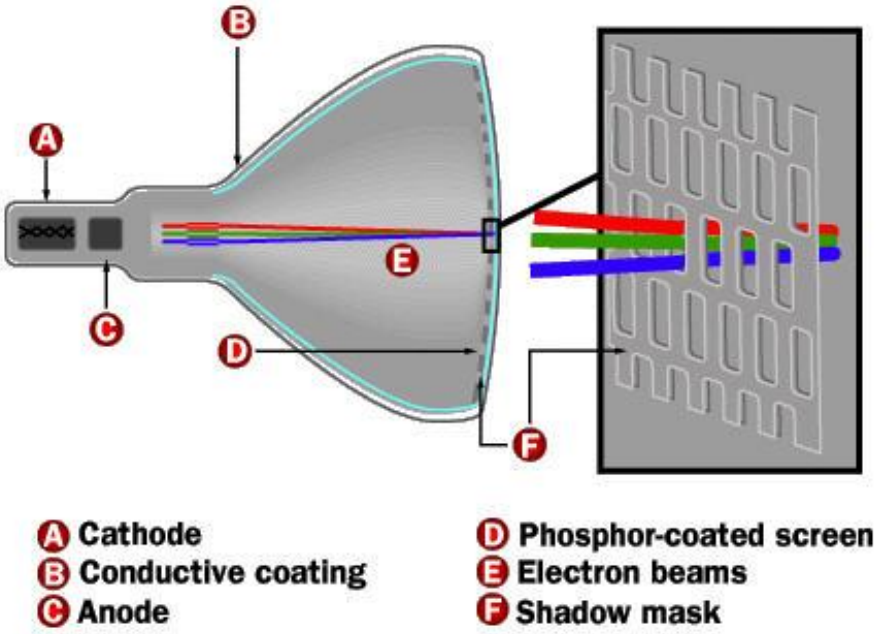
```
          end if;
        elsif(count=2675)then
          line_counter<=line_counter+2;                    -- Next Odd
Line
          color   <= "0000";
          composite<='1';
          pixel_counter<=0;
          count <= 0;
          counter1 <= 0;
          if (line_counter<625)then
            state <=    start_page1;
          elsif   (line_counter=625)then
            state <=    end_page1;
            line_counter <= 2;                          -- First Even Line
          end if;
        end if;

      when  end_page1   =>
        if(count<1600 )then                                 -- 32 us
          composite     <=    '1';                          -- or '0'
          color   <=    "0000";                           -- or"1111"
         elsif(count=1600)then                              -- or >=
          state <= vsync_1;
          composite     <=    '0';
          color <= "0000";
          count   <=    0;
        end if;

      when  vsync_1           =>
        if (count<8000)then                                 -- 160 us
          composite     <=    '0';
          color <= "0000";
        elsif(count=8000)then
          state <=      start_page2;
          composite     <=    '1';
          color <= "0000";
          count <= 0;
        end if;

-------------------------------- < Page2 = Even Lines > -----------------
-----------------

      when  start_page2 =>
        if(count<1600)then                                          -
- 32 us
          composite     <=    '1';
          color   <=    "0000";
        elsif(count=1600)then
          state   <=    page2_47us;
          composite     <=    '0';
          color   <=    "0000";
          count   <= 0;
        end if;

      when  page2_47us  =>
      -- 4.7 us
        if(count<235)then
          composite     <=    '0';
          color   <=    "0000";
        elsif(count=235)then
```

```
          state   <=    hsync_2;
          composite     <=    '1';
          color   <=    "0000";
          count   <=    0;
        end if;

      when  hsync_2           =>
            -- 5.8 us
        if(count<290)then
          composite     <=    '1';
          color   <=    "0000";
        elsif(count=290)then
          state   <=    color_data2;
          composite     <=    '1';
          color   <=    "0000";
          count   <=    0;
        end if;

      when  color_data2 =>
      -- 53.5 us
        if(count>0 and count<2675)then
          counter2<=counter2+1;
          composite<='1';
          if (counter2=4) then                          -- 5 Clk For Each
Pixel
            counter2<=0;
            pixel_counter<=pixel_counter+1;             -- Next Pixel
            if (pixel_counter<n and line_counter<=m) then
              if (rom(line_counter)(pixel_counter)='1') then
                color <= type_color;
              else
                 color <= "0000";                               -- Black
Color
              end if;
            end if;
          end if;
        elsif(count=2675)then
          line_counter<=line_counter+2;                 -- Next Even
Line
          color   <= "0000";
          composite<='1';
          pixel_counter<=0;
          count <= 0;
          counter2 <= 0;
          if (line_counter<626)then
            state <=    page2_47us;
          elsif   (line_counter=626)then
            state <=    vsync_2;
            line_counter <= 1;                          -- First Odd Line
          end if;
        end if;
      when  vsync_2           =>
            --    160 us
        if (count<8000)then
          composite     <=    '0';
          color <= "0000";
        elsif(count=8000)then
          state <=      start_page1;
          composite     <=    '1';
          color <= "0000";
```

```
          count <= 0;
        end if;
    end case;
  end if;

-----clk generation. For Osilator 50 MHz clock----

  if rising_edge(clock) then   --bedast avardan clk ba meghdare 1 sanie

      secend:=secend+1;
   if key_Auto ='1' then
      if secend<3 then  -- after 3 secend
            type_color<="0001";
            elsif secend<6 then -- after 6 secend
              type_color<="0010";  --change color
             elsif secend<9 then      -- after 9 secend
              type_color<="0100"; --change colo
              elsif secend<12 then -- after 12 secend
              type_color<="1000";  --change colo
              elsif secend<15 then -- after 15 secend
              type_color<="0000"; --change color
    else
       secend:=0;
       end if;

                  else
                        case key is  --change by key
                        when "1000" =>
                     type_color<="0000";
                         when "0100" =>
                     type_color<="1000" ;
                         when "0010" =>
                     type_color<="0100" ;
                          when "0001" =>
                     type_color<="0010" ;
                         when "0000" =>
                     type_color<="1111";
                        when others=> null;
                        end case;



                    end if;
                    end if;
end process;

end Behavioral;
```

۳۷- اتصال مانیتور به FPGA

اساس مانیتور ها و آداپتورهای تصویر :



شکل ۱: اشعه الکترونی در مانیتور

روش پیمایش:

در این روش برای نمایش تصویر در صفحه نمایش ازشعاع الکترونی استفاده می کنند. به وسیله این شعاع نقاط فسفری روی صفحه نمایش یا همان پیکسل ها روشن می شوند .برای این منظور شعاع الکترونی مسیر خود را از گوشه بالا سمت چپ آغاز می کند و با سمت راست پایین خاتمه می یابد . با خاموش شدن تفنگ الکترونی دوباره به ابتدای خط باز می گردد.

که این عمل را (HORIZONTAL RETRACE) می نامند.

به هنگام رسیدن به گوشه سمت راست پایین تفنگ خاموش شده وشعاع به گوشه سمت چپ بالا برده می شود . این خاموش شدن و بازگشت به بالا بازگشت عمودی (VERTICAL RETRACE) خوانده می شود.

مانیتور های رنگی:

دراین نوع مانیتورها هر نقطه فسفری از سه رنگ تشکیل شده است :

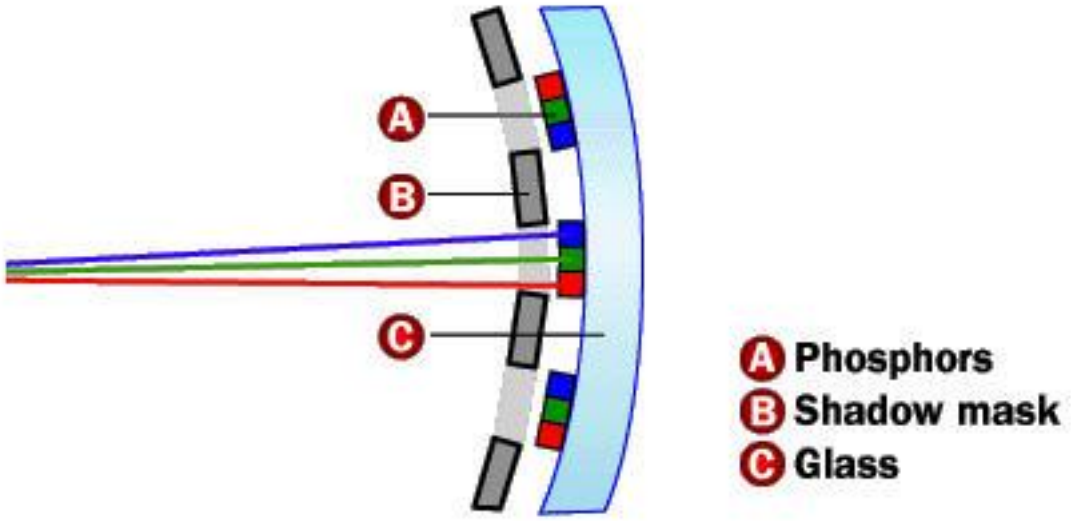
- قرمز
- آبی
- سبز

به همین جهت نام این مانیتور ها را RGB گذاشته اند. بـر خـلاف مانیتورهـای ترکیبـی در مانیتورهـای معمولی برای حمل هر شعاع الکترونی یک نوار مجزا وجود دارد که هر کدام رنگ مخصوص به خود را نشان می دهد. در مانیتورهای ترکیبی یک نوار هر سه رنگ RGB را ایجاد مـی کنـد . در ایـن ترکیـب و جـدا سازی RGB از کیفیت تصویر می کاهد.

اختلاف دیگر بین مانیتور های رنگی وتک رنگ وجود پوشش سایه دار مانیتورهای رنگی است.

پوشش سایه :

قبل ازصفحه فسفری یک صفحه فلزی سوراخ دار قرار گرفته که بدین وسیله هر پرتاب شـعاع الکترونـی از یک سوراخ عبور می کند . و باعث تفکیک رنگ ها می شود یعنی ایجاد هر رنگ توسط تفنگ همـان رنـگ تضمین خواهد شد.



شکل ۲ : اشعه الکترونی در مانیتور های رنگی

VGAپورت:

این پورت به منظور اتصال به مانیتور کامپیتور مورد استفاده قرار می گیرد و از پنج سیگنال زیر تشکیل شده است :

۱) سیگنال همزمانی افقی

۲) سیگنال همزمانی عمودی

۳) سه سیگنال رنگ: آبی و سبز و قرمز (RGB)

نمایشگرهای (CRT) از یک اشعه الکترونی برای نمایش تصاویر به روی صفحه ای فسفری استفاده می کنند.

در حالی که نمایشگرهای LCD از تعدادی پیکسل تشکیل شده اند که با تغییر ولتاژ اعمالی به آنها شدت نور ساطع شده از آنها تغییر می یابد.

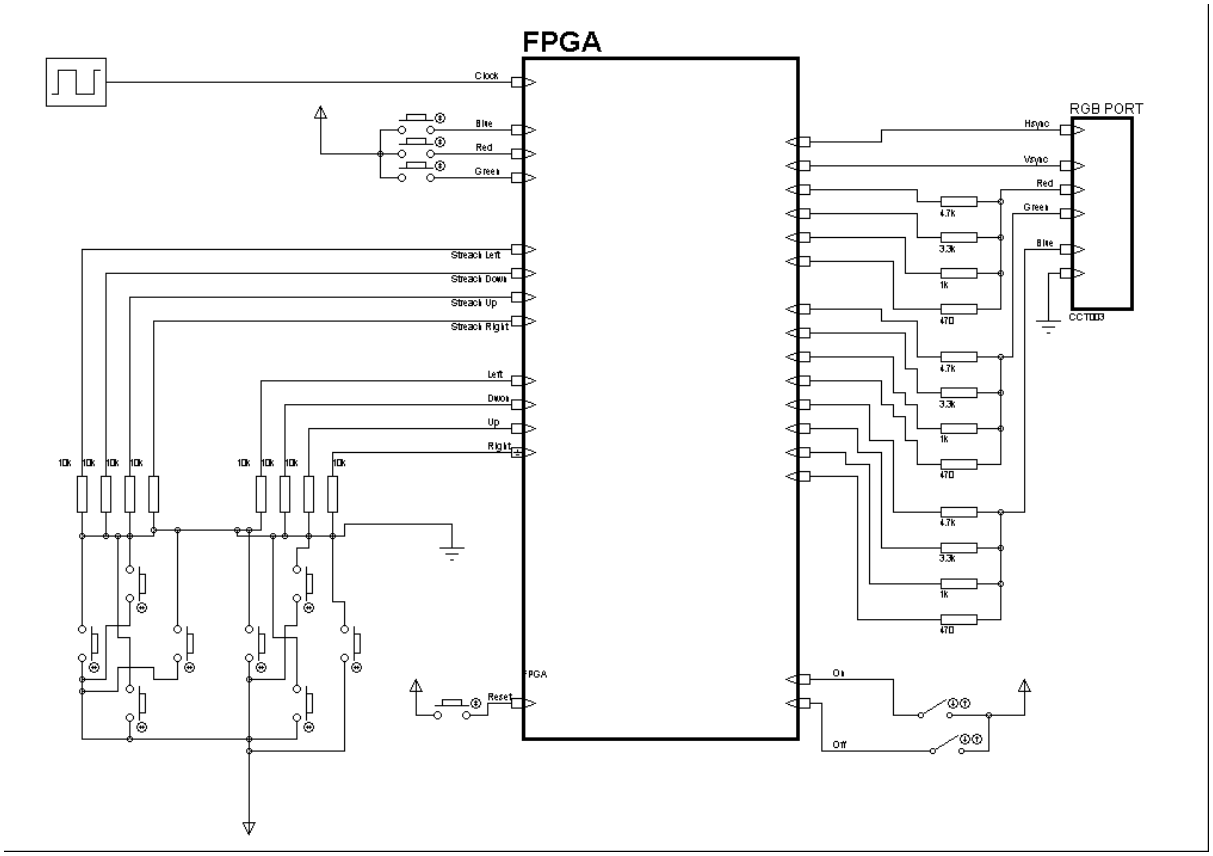
تصویری که باید نمایش داده شود از فریم های متعددی تشکیل شده است که هر فریم نیز به خطوط متعددی تقسیم می شود.

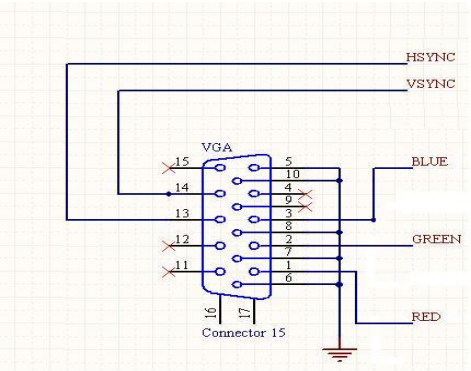
هر خط از تعدادی نقطه تشکیل می شود که ترکیب اشعه های آبی و سبز و قرمز در هر نقطه رنگ آنرا مشخص می کنند.

به منظور سنکرون سازی اشعه الکترونی با اطلاعات تصویر متناظر با هر خط نمایش داده شده یک پالس به روی سیگنال همزمانی افقی (HSYNC) و به ازای نمایش کلیه خطوط تصویر (یک فریم) یک پالس به روی سیگنال همزمانی عمودی (VSYNC) ایجاد می شود .

فرکانس سیگنال های همزمانی افقی و عمودی و همچنین تعداد شمارش های کلاک اسیلاتور ۵۰ مگا هرتز جهت به کار گیری مانیتور در حالت نمایش ۴۸۰*۶۴۰ (تعداد خط های تصویر برابر ۴۸۰ و تعداد پیکسل ها در هر خط ۶۴۰)

شکل مدار:





برنامه:

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.ALL;
ENTITY vga IS
PORT (
                -- keybaraye move vataghirandazehmorabavataghir rang
morabahvasafheyenamayesh
                screen, square, left, right, up, down, stretch_left,
stretch_right,
                stretch_up, stretch_down, clk, rst: IN std_logic;
                -- keybarayetaghir rang red, green & blue
```

```
                  Btn_R,Btn_G,Btn_B:in std_logic;
                  -- signal hamzamaniofoghi
                  hsync: BUFFER BIT;
                  -- signal hamzamaniamodi
                  vsync: OUT BIT;
                  led: out std_logic_vector(7 downto 0);
                  --omghe rang red, green , blue
                  Red,Green,Blue: out std_logic_vector(3 downto 0));
END vga ;
-----------------------------------------------
ARCHITECTURE behav OF vga IS
Signal Square_Red,Square_Green,Square_Blue: std_logic_vector(3 downto
0):="0000";
Signal Screen_Red,Screen_Green,Screen_Blue: std_logic_vector(3 downto
0):="1111";
SHARED VARIABLE x1: INTEGER RANGE 0 TO 2000:=200;
SHARED VARIABLE x2: INTEGER RANGE 0 TO 2000:=400;
SHARED VARIABLE y1: INTEGER RANGE 0 TO 1000:=200;
SHARED VARIABLE y2: INTEGER RANGE 0 TO 1000:=400;
CONSTANT h: INTEGER :=  809;
CONSTANT v: INTEGER := 530;
SIGNAL clr1: BIT_VECTOR(2 DOWNTO 0);
SIGNAL clr2: BIT_VECTOR(2 DOWNTO 0);
BEGIN
-----------------------------------------------
PROCESS (clk, rst)
      VARIABLE hcnt: INTEGER RANGE 0 TO h;
      VARIABLE vcnt: INTEGER RANGE 0 TO v;
      VARIABLE c,l: INTEGER RANGE 0 TO 3;
      ------------------------------------------------
      VARIABLE dbnc1 : INTEGER RANGE 0 TO  200000000 ;

BEGIN
      IF (rst='1') THEN
            hcnt :=0;
            -------------------
            Screen_Red<="1111";
            Screen_Green<="0000";
            Screen_blue<="0000";
            Square_Red<="0000";
            Square_Green<="0000";
            Square_Blue<="1111";
            x1:=400;
            x2:=600;
            y1:=200;
            y2:=400;
            ---------------------
      ELSIF (clk='1' AND clk'EVENT ) THEN
                  c := c+1;
                  l:=l+1;
             -- clock is 50MH/2=25mh
            IF (c=1) THEN
                  c := 0;
if  hcnt< 799  then
hcnt := hcnt+ 1;
else



hcnt := 0;
```

```
if  vcnt< 524  then -- less than oh
vcnt := vcnt + 1;
else
vcnt := 0;       -- is set to zero
end if;
end if;
  -- define H pulse
if  hcnt>= 662  and  hcnt< 755  then
hsync<= '0';
else
hsync<= '1';
end if;

  -- define V pulse
if  vcnt>= 491  and  vcnt< 493  then
vsync<= '0';
else
vsync<= '1';
end if;
                  -- ifh_counter and v_counterdarmantaghehmorababashand
rang morabahnamayeshdadehshavad.
                  IF (y1<vcnt AND vcnt<y2 AND x1<hcnt AND hcnt<x2) THEN
                        Red<=Square_Red;
                        Green<=Square_Green;
                        Blue<=Square_Blue;
                        -- else if
h_conter&v_counterdarmahdodehnamayeshbashand rang
zaminehnamayeshdadehshavad
                  ELSif (480>vcnt and 640 >hcnt) then
                              Red<=Screen_Red;
                              Green<=Screen_Green;
                              Blue<=Screen_Blue;
                        -- elsedarmantagegkhamosh (h return bak  or v
Reterbak) rang 0 yanikhamoshferestadehshavad
                        else
                           Red<="0000";
                              Green<="0000";
                              Blue<="0000";
                  END IF;
            END IF;
             -- clock is 50MH/2=25mh
            IF (l=1) THEN
                  l:=0;
                  -- if time beyn 2 key
                  if (dbnc1>0)THEN
                        dbnc1:=dbnc1-1;
                  else
                                    ---------------------------- ifmoraba
selected

                        IF(square='1')THEN
                              --   ifBtn Red pushed
                              if(Btn_R='1') then
                                    IF (Square_Red<"1111" )THEN
                                          Square_Red<=Square_Red+1;
                                    ELSE
                                          Square_Red<="0000";
                                    END if;
                                    dbnc1:=10000000;
                                    led(3 downto 0)<=Square_Red;
```

```
                              --   ifBtn green pushed
                              elsif Btn_G='1' Then
                                          IF (Square_Red<"1111" )THEN

      Square_Green<=Square_Green+1;
                                          ELSE
                                                Square_Green<="0000";
                                          END if;
                                          dbnc1:=10000000;
                                          led(3 downto 0)<=Square_Green;
                              --   ifBtn Blue pushed
                               elsif Btn_B='1' Then
                                          IF (Square_Blue<"1111"
)THEN

      Square_Blue<=Square_Blue+1;
                                          ELSE

      Square_Blue<="0000";
                                          END if;
                                          dbnc1:=10000000;
                                          led(3 downto
0)<=Square_Blue;
                                    else
                                        dbnc1:=0;
                                    END If;
                        END IF;
                              ----------------------------
                        --   ifBtn Screen Selected
                        IF (screen='1')THEN
                              --   ifBtn Red pushed
                              if(Btn_R='1') then
                                    IF (Screen_Red<"1111" )THEN
                                          Screen_Red<=Screen_Red+1;
                                    ELSE
                                          Screen_Red<="0000";
                                    END if;
                                    dbnc1:=10000000;
                                    led(3 downto 0)<=Screen_Red;
                              --   ifBtn green pushed
                              elsif Btn_G='1' Then
                                          IF (Screen_green<"1111" )THEN

      Screen_Green<=Screen_Green+1;
                                          ELSE
                                                Screen_Green<="0000";
                                          END if;
                                          dbnc1:=10000000;
                                          led(3 downto 0)<=Screen_Green;
                                    --   ifBtn Blue pushed
                                    elsif Btn_B='1' Then
                                          IF (Screen_Blue<"1111"
)THEN

      Screen_Blue<=Screen_Blue+1;
                                          ELSE

      Screen_Blue<="0000";
                                          END if;
                                          dbnc1:=10000000;
```

```
                                        led(3 downto
0)<=Screen_Blue;
                                    else
                                        dbnc1:=0;
                                    END If;
                            END IF;
                                    ------------------------------left
                                --   ifBtn Left pushed
                                IF (left='1') AND (x1>0) THEN
                                      x1 := x1-1;
                                      x2 := x2-1;
                                      dbnc1 := 200000;
                                END IF;
                                      ---------------------------------
right
                                --   ifBtn Right pushed
                                IF (right='1') and (x2<639) THEN
                                      x1 := x1+1;
                                      x2 := x2+1;
                                      dbnc1 := 200000;
                                END IF;
                                      ---------------------------------up
                                --   ifBtn Up pushed
                                IF (up='1') and (y1>0)THEN
                                      y1 := y1-1;
                                      y2 := y2-1;
                                      dbnc1 := 200000;
                                END IF;
                                      ---------------------------------down
                                --   ifBtn Down pushed
                                IF (down='1') and (y2<479)THEN
                                      y1 := y1+1;
                                      y2 := y2+1;
                                      dbnc1 := 200000;
                                END IF;
                                      ---------------------------------
stretch left
                                --   ifBtnStrech_left pushed  width
morabakamshavad
                                IF (stretch_left='1') and (x2>(x1+10))THEN
                                      x2 := x2-1;
                                      dbnc1 := 200000;
                                END IF;
                                      ---------------------------------
stretch right
                                --   ifBtnStrech Right pushed  width
morabaezafehshavad
                                IF (stretch_right='1') and (x2<639)THEN
                                      x2 := x2+1;
                                      dbnc1 := 200000;
                                END IF;
                                      ---------------------------------
stretch up
                                --   ifBtnStrech_uP pushed Height
morabakamshavad
                                IF (stretch_up='1') and (y2>y1+10)THEN
                                      y2 := y2-1;
                                      dbnc1 := 200000;
                                END IF;
```

```
                                        --------------------------------
stretch down
                                --   ifBtnStrech_down pushed height
morabaezafehshavad
                                IF (stretch_down ='1') and (y2<479) THEN
                                      y2 := y2+1;
                                      dbnc1 := 200000;
                                END IF;
                    END IF; --dbnc1
               END IF;
         END IF;       --clk
END PROCESS;
led(4)<=btn_r;-- Btn Red Pushed
led(5)<=btn_g;-- Btn Green Pushed
led(6)<=btn_b;-- Btn Blue Pushed
led(7)<=Screen; -- btn Screen Pushed
END behav;
```

۳۸- اتصال کارت SD

این نوع حافظه ها که از نوع FLASHمیباشند و به صورت مرسوم در دستگاههای قابل حملی مانند MP3 PLAYERها COOL DISKها و موبایلها استفاده میشوند .از خصوصیات بارز آن میتوان به سرعت بالا و حجم بسیار بالای آن اشاره کرد .این حجم بالا که از چندین مگابایت تا چندین گیگابایت میباشد

این پروتکل ارتباطی به طور معمول دارای سه خط اصلی ارتباطی میباشد که یکی از این سه خط وظیفه ارسال داده، دیگری دریافت داده و خط سوم حامل کلاک میباشد.

تغذیه مموری بین ۲.۷ ولت تا ۳.۶ ولت میباشد که برای اطمینان از صحت کار آن توصیه شده با ولتاژ ۳.۶ ولت تغذیه شود و حافظه بسیار به ولتاژ کاری خود حساس است و امکان آسیب دیدن آن بدلیل اضافه ولتاژ و نویز بسیار زیاد است

یکی از خارق العاده ترین تکنولوژیهایی که شرکت مایکروسافت ارائه داده FATمی باشد این استاندارد که برای مکان حافظههای بسیار حجیم بوجود آمد در ابتدا با استانداردFAT12ارائه شد که قابلیت آدرس دهی ۱۲بیتی کلاسترها را داشت و تا ۱۶ مگابایت مموری یا هارددیسک را پشتیبانی می کرد وبعد از آن FAT 16 به دلیل محدودیت های ظرفیت FAT32 آمد .FAT32 قابلیت آدرس دهی ۳۲ بیتی کلاسترها را دارا بود

Sector: فضای مموری کارتها یا هارد دیسکها به تعدادی سکتور تقسیم میشوندکه برای مموری کارتهای Sd و mmc سکتور شامل ۵۱۲ بایت میباشد و مموری کارت درهنگام نوشتن و خواندن بطور مستقیم با بایتها سر و کار ندارد بلکه با سکتورها کار میکند .

CLUSTER: از دید سخت افزاری کلاستر وجود خارجی ندارد و فقط در استاندارد fatفایل سیستم بجای اینکه با سکتورها کار کند که تعداد زیادی میباشند با کلاستر کار میکند، و کلاستر به مجموعهای از سکتورها اطلاق میشود که برای هر نوع fat اعم از fat12,fat16,fat32 متفاوت و وابسته به ظرفیت حافظه میباشد

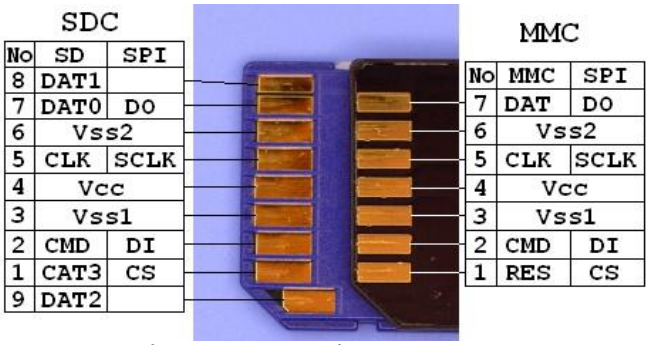
MBR (MASTER BOOT RECORD): MBR یک کد اجرایی کوچکی است که در بخشی از حافظه قرار میگیرد تا سیستم از آن به اصطلاح BOOT شود.

برای ارسال هر داده یا دستور به مموری باید ابتدا csفعال شود .

این نوع حافظه شامل چندید سکتور است که همه سکتورها در همه انواع مموری کارت ۵۱۲ بایت است.

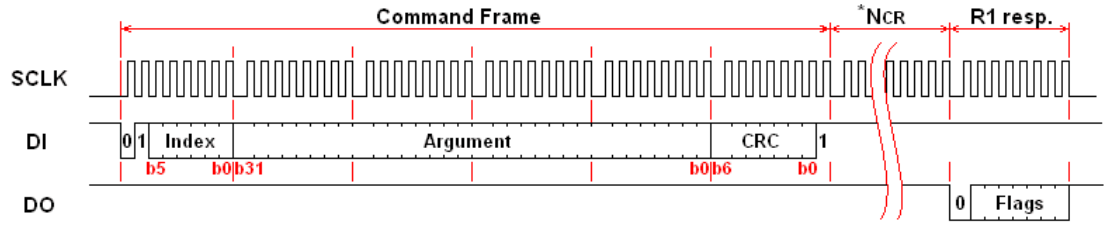
نکته مهم دیگر اینکه در این ارتباط میتوان انتخاب کرد که ابتداMSBارسال شود یاLSBکه در بحث مموری کارت MSBابتدا ارسال می شود.

شکل زیر وضعیت پایه های مموری کارت وابعاد فیزیکی مموری را نشان می دهد

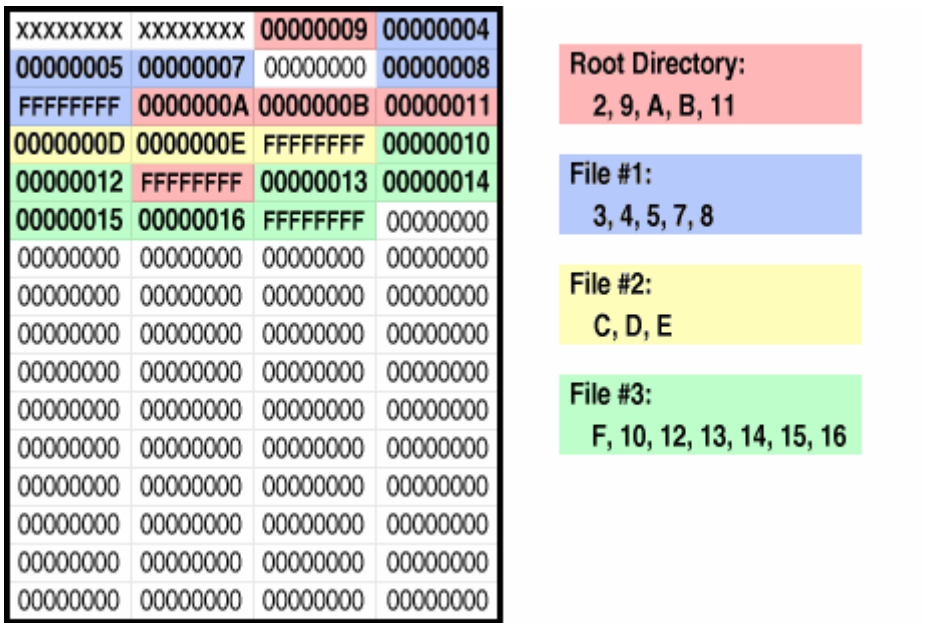


Pin No.	Name	Type[1]	Description
SD Mode			
1	CD/DAT3[2]	I/O[3], PP	Card detect/Data line [Bit 3]
2	CMD	I/O, PP	Command/Response
3	V_{SS1}	S	Supply voltage ground
4	V_{DD}	S	Supply voltage
5	CLK	I	Clock
6	V_{SS2}	S	Supply voltage ground
7	DAT0	I/O, PP	Data line [Bit 0]
8	DAT1	I/O, PP	Data line [Bit 1]
9	DAT2	I/O, PP	Data line [Bit 2]
SPI Mode			
1	CS	I	Chip Select (active low)
2	DataIn	I	Host-to-card Commands and Data
3	V_{SS1}	S	Supply voltage ground
4	V_{DD}	S	Supply voltage
5	CLK	I	Clock
6	V_{SS2}	S	Supply voltage ground
7	DataOut	O	Card-to-host Data and Status
8	RSV[4]	---	Reserved
9	RSV[5]	---	Reserved

Command and Response



FAT



xxxxxxxx	xxxxxxxx	00000009	00000004
00000005	00000007	00000000	00000008
FFFFFFFF	0000000A	0000000B	00000011
0000000D	0000000E	FFFFFFFF	00000010
00000012	FFFFFFFF	00000013	00000014
00000015	00000016	FFFFFFFF	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Root Directory:
2, 9, A, B, 11

File #1:
3, 4, 5, 7, 8

File #2:
C, D, E

File #3:
F, 10, 12, 13, 14, 15, 16

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity sd is

   Port(
        clk,reset_ms:in std_logic;
              mosi: out std_logic;
        ss : out std_logic;
        miso : in std_logic;
        sclk : out std_logic;
              -------lcd
              data_lcd : out std_logic_vector(7 downto 0);
          rs_lcd : out std_logic;
          rw_lcd : out std_logic;
          e_lcd : out std_logic;
          bl_lcd : out std_logic;
        led:out std_logic_vector(13 downto 0);
       pin :out bit := '0');

End sd;

Architecture Gate_Level_1 of sd is
             type sector is array(0 to 160) of std_logic_vector(7 downto
0);
       Type State is
(spi_start,Start_Sample,Start_Sample2,sd_state,lcd_cmd,start_lcd,write_lc
d,delay_lcd,stop_lcd,sel_arg);
       signal sdc_state : state:=sd_state;
             signal arg:std_logic_vector(31 downto 0):=(others=>'0');
```

```
       signal bit_cnt : integer := 7;
       signal b:integer:=0;
       signal c,q:std_logic:='0';
             signal data_in:std_logic_vector(7 downto 0);
             signal sec:sector:=(others=>"00100000");
             constant delay_for_e : integer := 2;     --18 * 25 NS = 450
NS
       constant delay_between_data : integer := 1250;
             type two_array is array (0 to 2)of std_logic_vector (7
downto 0);
             signal clust:std_logic_vector(22 downto 0):=(others=>'0');
       constant table_lcd : two_array :=("00111000", "00001110",
"00000001");--dastoorate lcd
             signal boot_address:std_logic_vector(31 downto
0):=(others=>'0');
             signal num_boot_sec:std_logic_vector(16 downto
0):=(others=>'0');
             signal fat_byte:std_logic_vector(25 downto
0):=(others=>'0');
             signal first_cluster:std_logic_vector(31 downto
0):=(others=>'0');
             type file_name is array (0 to 2) of std_logic_vector(7
downto 0);

             constant name:file_name:=("01001101","01000101","01000101");
--mee

             type root_name is array (0 to 5) of std_logic_vector(7
downto 0);
             signal root: root_name
:=("01001101","01010011","01000100","01001111","01010011","00110101"); --
MSDOS5

Begin

 process(clk)
  begin
   if clk'event and clk='1' then

    if b=100 then
     b<=0;
     q<=c;
     c<=not c;
    else
     b<=b+1;

       end if;

  end if;
 end process;




Process(q,reset_ms)
    variable falling,rising:std_logic:='0';
       variable
x,y,aa,bb,level,cnt,cntff,cntloop,cmd_cnt,part,s_level,s_cnt:integer:=0;
    variable i_lcd, count_lcd,m : integer;
```

```
begin
--
*****************************************************************************

              if q'event and q='1' then
--
*****************************************************************************
if (reset_ms = '1') then
  sdc_state <= sd_state;

            level:=0;

      e_lcd<='0';
      rw_lcd<='0';
      bl_lcd<='1';
           --
else
case sdc_state is
--
*****************************************************************************



              when sd_state =>
                  if level=0 then
                     cnt:=10;
                         level:=1;
         elsif level=1 then
                      if cnt>0 then
                         data_in<="11111111";
                         ss<='1';
                         level:=1;
                         cnt:=cnt-1;
                         sdc_state <= spi_start;
                         else
                         level:=2;
                         end if;
          elsif level=2 then
                      data_in<="01000000";
                         ss<='0';
                         level:=3;
                         sdc_state <= spi_start;
                         cnt:=4;
                    elsif level=3 then
                      if cnt>0 then
                         data_in<="00000000";
                         ss<='0';
                         level:=3;
                         cnt:=cnt-1;
                         sdc_state <= spi_start;
                         else
                         level:=4;
                         end if;
                   elsif level=4 then
                      data_in<="10010101";
                         ss<='0';
                         level:=5;
                         sdc_state <= spi_start;
                elsif level=5 then
                       if data_in="00000001" then
```

```
                              ss<='1';
                             level:=6;
            else
                             data_in<="11111111";
                                level:=5;
                                sdc_state <= spi_start;
                                ss<='0';
                            end if;
                           ----------------------------------------RESET
      command (cmd0)

                    elsif level=6 then
                         data_in<="01000001";
                           ss<='0';
                           level:=7;
                           sdc_state <= spi_start;
                           cnt:=4;
                     elsif level=7 then
                        if cnt>0 then
                           data_in<="00000000";
                           ss<='0';
                           level:=7;
                           cnt:=cnt-1;
                           sdc_state <= spi_start;
                           else
                           level:=72;
                           cntff:=8;
                           end if;
                     elsif level=72 then
                         data_in<="11111111";
                           ss<='0';
                           level:=8;
                           sdc_state <= spi_start;
                     elsif level=8 then
                        if data_in="00000000" then
                    --     arg<=(others=>'0');
                             level:=9;
            else
                             if cntff>0 then
                                cntff:=cntff-1;
                             data_in<="11111111";
                                level:=8;

                                sdc_state <= spi_start;
                                ss<='0';
                                else
                                level:=6;
                                end if;
                            end if;
                            ----------------------------------------INIT
command       (cmd41)
                      elsif level=9 then
                        data_in<="01010001";
                           ss<='0';
                           level:=10;
                           if s_level=0 then
                           arg(31 downto 0)<=clust(22 downto 0) &
"000000000";
                           end if;
                           sdc_state <= spi_start;
```

```
                    -- cnt:=4;
                elsif level=10 then
              --  data_in<="00000000";   ---addrese morede nazar
                    data_in<=arg(31 downto 24);
                    ss<='0';
                    sdc_state <= spi_start;
                    level:=101;
                elsif level=101 then
              -- data_in<="00000001";   ---addrese morede nazar
                    data_in<=arg(23 downto 16);
                    ss<='0';
                    sdc_state <= spi_start;
                    level:=102;
                elsif level=102 then
              -- data_in<="00001010";   ---addrese morede nazar
                    data_in<=arg(15 downto 8);
                    ss<='0';
                    sdc_state <= spi_start;
                    level:=103;
                elsif level=103 then
             --   data_in<="00000000";   ---addrese morede nazar
                    data_in<=arg(7 downto 0);
                    ss<='0';
                    sdc_state <= spi_start;
                    level:=11;
                                        ------------------------------
-----------
                elsif level=11 then
                  data_in<="00000000";
                    ss<='0';
                    level:=12;
                    sdc_state <= spi_start;
                    cntff:=8;
                elsif level=12 then

                  if data_in="11111110" then      --    if clust >
"00000000000000000000000" then      pin<='1';   end if;
                     level:=13;
                        cnt:=0;
            else
                    if cntff>0 then
                     data_in<="11111111";
                        level:=12;
                        sdc_state <= spi_start;
                        ss<='0';
                        cntff:=cntff-1;
                        else
                         level:=9;
                        end if;
                     end if;
                   elsif level=13 then
                        sdc_state <= spi_start;
                         data_in<="11111111";
                         -- clust<=clust+1;
                        level:=14;
                elsif level=14 then
                        if cnt=511 then
                    --    led(7 downto 0)<=sec(0);
              --          i_lcd:=0;
```

```
                                --           arg(31 downto 0)<=clust(22
downto 0) & "000000000";
                    --           sdc_state <= lcd_cmd;
                                  sdc_state <=sel_arg;  --    led(7)<='1';
                                 cmd_cnt:=0;
                                 ss<='1';
                                 sec(160)<=data_in;
                                elsif cnt<511 then
                                if cnt<160 then
                                 sec(cnt)<=data_in;
                                 level:=15;
                                 else
                                 sec(160)<=data_in;
                                  level:=15;
                                 end if;
                                 end if;
                          elsif level=15 then
                                cnt:=cnt+1;
                                level:=13;

                          end if;
                    ---------------------------------------- cmd(51)
                    when sel_arg =>
                          if s_level=0 then
                                        if sec(3 to 8) =
("01001101","01010011","01000100","01001111","01010011","00110101") then
                         --msdos5
                                          -- led(8 downto 0)<=clust(8 downto
0);
                           boot_address(31 downto 0)<=arg(31 downto 0);
pin<='1';
                                           s_level:=1;
                                           else
                                         clust<=clust+1;
                                                level:=0;   ----read next block
                                                s_level:=0;
                                                sdc_state <= sd_state;
                                           end if;
                                elsif s_level=1 then
                                    led(13)<='1';
                                     --  num_boot_sec<=sec(14) & "000000000";--
-root sector
                                    --           fat_byte<=sec(22) & sec(23) &
"0000000000";     --tedade fat
                                    --           arg(31 downto 0)<=boot_address
+ num_boot_sec + fat_byte;
                                          arg(31 downto 0)<=boot_address +
(sec(14) & "000000000") + (sec(23) & sec(22) & "0000000000");

                                           s_level:=2;
                                           s_cnt:=0;
                                           level:=0;
                                               sdc_state <= sd_state;
                                elsif s_level=2 then
                        --    if (s_cnt < 130) then
                                          if sec(s_cnt)=name(0) and sec(s_cnt
+ 1)=name(1) and  sec(s_cnt+2)=name(2) then                 ----mem
                                            arg(31 downto 0)<=((sec(s_cnt +
26) &  sec(s_cnt + 27) & "000000000") + (boot_address));
                                                        level:=0;
```

```
                                                    sdc_state <=
sd_state;          led(12)<='1';
                                                    s_level:=22;
                                              else
                                                   s_cnt:=s_cnt + 32;     -
-       led(11)<='1';
                                                   s_level:=2;
                                                   end if;
                                 --  else

                            --    end if;
                                   elsif s_level=22 then
                                              led(11 downto 0)<=arg(19 downto
8);
                                               s_level:=3;

                                  elsif s_level=3 then

                                                   i_lcd:=0;
                                              --   led(10)<='1';
                                               sdc_state <= lcd_cmd;
                                               cmd_cnt:=0;
                                  end if;
             ---------------------------------- sd controler
              when spi_start=>
                    bit_cnt <= 7;
                 sclk<='0';
                 sdc_state <= Start_Sample;

              when Start_Sample =>
                              sclk<='0';
                            mosi <= data_in(7);
                              for i in 7 downto 1 loop
                                  data_in(i) <= data_in(i-1);
                              end loop;
                                    sdc_state <= Start_Sample2;



                   when Start_Sample2 =>
                               sclk<='1';
                                data_in(0) <=miso;

                                 bit_cnt <=bit_cnt - 1;


                               if bit_cnt=0 then

                      sdc_state <= sd_state;

                   else
                    sdc_state <= Start_Sample;


                           end if;


--*****************************************************************--spi
             -- when lcd_start=>
```

```
               when lcd_cmd =>
                                -- led(9)<='1';

                                   if cmd_cnt<3 then
                                       rs_lcd <= '0';
                                       data_lcd(7 downto 0) <=
table_lcd(cmd_cnt)(7 downto 0);
                                        sdc_state <= write_lcd;
                                        cmd_cnt:=cmd_cnt+1;
                                        else
                                            sdc_state <= start_lcd;
                                            end if;

                           when start_lcd =>
                           --    led(8)<='1';
                             data_lcd(7 downto 0) <= sec(i_lcd)(7 downto
0);
                                --    data_lcd <= table_test(i_lcd)(7
downto 0);
                                      rs_lcd <= '1';
                                      i_lcd := i_lcd + 1;
                                      if i_lcd < 8 then
                                         sdc_state <= write_lcd;
                                      else
                                            sdc_state <= stop_lcd;
                                end if;

                          when write_lcd =>
                               if part = 0 then
                                          e_lcd <= '1';
                                             count_lcd := delay_for_e;
                                          part := part + 1;
                                             sdc_state <= delay_lcd;
                                      elsif part = 1 then
                                      e_lcd <= '0';
                                      count_lcd := delay_between_data;
                                      sdc_state <= delay_lcd;
                                      part := part + 1;
                                      else
                                      sdc_state <= lcd_cmd;
                                      part := 0;
                                      end if;

                          when delay_lcd =>
                                      count_lcd := count_lcd - 1;
                                      if count_lcd = 0 then
                                      sdc_state <= write_lcd;
                        end if;

                          when stop_lcd =>
end case;
end if;

--*****************************************************************--lcd
 end if;
end process;


End Gate_Level_1;
```

Write data:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity sd is

   Port(
        clk,reset_ms:in std_logic;
              mosi: out std_logic;
        ss : out std_logic;
        miso : in std_logic;
        sclk : out std_logic;
             led:out std_logic_vector(13 downto 0);
       pin :out bit := '0');

End sd;

Architecture Gate_Level_1 of sd is
             type sector is array(0 to 159) of std_logic_vector(7 downto
0);
       Type State is
(spi_start,Start_Sample,Start_Sample2,sd_state,sel_arg);
       signal sdc_state : state:=sd_state;
             signal arg:std_logic_vector(31 downto 0):=(others=>'0');
       signal bit_cnt : integer := 7;
       signal b:integer:=0;
       signal c,q:std_logic:='0';
             signal data_in:std_logic_vector(7 downto 0);
             signal sec:sector:=(others=>"00100000");
             constant delay_for_e : integer := 2;     --18 * 25 NS = 450
NS
       constant delay_between_data : integer := 1250;
             type two_array is array (0 to 2)of std_logic_vector (7
downto 0);
             signal clust:std_logic_vector(22 downto 0):=(others=>'0');
       constant table_lcd : two_array :=("00111000", "00001110",
"00000001");--dastoorate lcd
             signal boot_address:std_logic_vector(31 downto
0):=(others=>'0');
             signal num_boot_sec:std_logic_vector(16 downto
0):=(others=>'0');
             signal fat_byte:std_logic_vector(25 downto
0):=(others=>'0');
             signal first_cluster:std_logic_vector(31 downto
0):=(others=>'0');
             type file_name is array (0 to 2) of std_logic_vector(7
downto 0);
             constant name:file_name:=("01001101","01000101","01000101");
--mee
             type root_name is array (0 to 5) of std_logic_vector(7
downto 0);
             type name_sec is array(0 to 159) of std_logic_vector(7
downto 0);
```

```
            signal root: root_name
:=("01001101","01010011","01000100","01001111","01010011","00110101"); --
MSDOS5
            signal
new_data:sector:=("01101000","01100101","01101100","01101100","01101111",
"01101111", "01101110",others=>"00000000"); --
("01010011","01000001","01001100","01000001","01010011","01010011");
Begin

 process(clk)
  begin
   if clk'event and clk='1' then

    if b=100 then
     b<=0;
     q<=c;
     c<=not c;
    else
     b<=b+1;

       end if;

  end if;
 end process;




Process(q,reset_ms)
    variable falling,rising:std_logic:='0';
       variable
x,y,aa,bb,level,cnt,cntff,cntloop,cmd_cnt,part,s_level,s_cnt,write_sd,wr_
cnt,offset, data_cnt:integer:=0;
    variable i_lcd, count_lcd,m : integer;
begin
--
*************************************************************************

              if q'event and q='1' then
--
*************************************************************************
if (reset_ms = '1') then
  sdc_state <= sd_state;

            level:=0;

--    e_lcd<='0';
--    rw_lcd<='0';
--    bl_lcd<='1';
           --
else
case sdc_state is
--
*************************************************************************



              when sd_state =>
                  if level=0 then
```

```
                cnt:=10;
                    level:=1;
    elsif level=1 then
                 if cnt>0 then
                    data_in<="11111111";
                    ss<='1';
                    level:=1;
                    cnt:=cnt-1;
                    sdc_state <= spi_start;
                    else
                    level:=2;
                    end if;
     elsif level=2 then
                 data_in<="01000000";
                    ss<='0';
                    level:=3;
                    sdc_state <= spi_start;
                    cnt:=4;
               elsif level=3 then
                 if cnt>0 then
                    data_in<="00000000";
                    ss<='0';
                    level:=3;
                    cnt:=cnt-1;
                    sdc_state <= spi_start;
                    else
                    level:=4;
                    end if;
            elsif level=4 then
                 data_in<="10010101";
                    ss<='0';
                    level:=5;
                    sdc_state <= spi_start;
          elsif level=5 then
                 if data_in="00000001" then
                        ss<='1';
                     level:=6;
       else
                     data_in<="11111111";
                        level:=5;
                        sdc_state <= spi_start;
                        ss<='0';
                    end if;
                  ----------------------------------------RESET
command (cmd0)

            elsif level=6 then
                 data_in<="01000001";
                   ss<='0';
                   level:=7;
                   sdc_state <= spi_start;
                   cnt:=4;
              elsif level=7 then
                 if cnt>0 then
                    data_in<="00000000";
                    ss<='0';
                    level:=7;
                    cnt:=cnt-1;
                    sdc_state <= spi_start;
                    else
```

```
                        level:=72;
                        cntff:=8;
                        end if;
                    elsif level=72 then
                        data_in<="11111111";
                        ss<='0';
                        level:=8;
                        sdc_state <= spi_start;
                    elsif level=8 then
                       if data_in="00000000" then
                   --     arg<=(others=>'0');

                    if write_sd=1 then
                        cnt:=10;
                        level:=16;
                           elsif write_sd=0 then
                            level:=9;
                               end if;
               else
                            if cntff>0 then
                               cntff:=cntff-1;
                            data_in<="11111111";
                               level:=8;

                               sdc_state <= spi_start;
                               ss<='0';
                               else
                               level:=6;
                               end if;
                            end if;
                            ----------------------------------------INIT
command       (cmd41)
                     elsif level=9 then

                       data_in<="01010001";
                          ss<='0';
                          level:=10;
                          if s_level=0 then
                          arg(31 downto 0)<=clust(22 downto 0) &
"000000000";
                          end if;
                          sdc_state <= spi_start;
                         -- cnt:=4;
                    elsif level=10 then
                   --  data_in<="00000000";   ---addrese morede nazar
                          data_in<=arg(31 downto 24);
                          ss<='0';
                          sdc_state <= spi_start;
                          level:=101;
                      elsif level=101 then
                    -- data_in<="00000001";   ---addrese morede nazar
                          data_in<=arg(23 downto 16);
                          ss<='0';
                          sdc_state <= spi_start;
                          level:=102;
                       elsif level=102 then
                    -- data_in<="00001010";   ---addrese morede nazar
                          data_in<=arg(15 downto 8);
                          ss<='0';
                          sdc_state <= spi_start;
```

```
                    level:=103;
                elsif level=103 then
            --   data_in<="00000000";   ---addrese morede nazar
                    data_in<=arg(7 downto 0);
                    ss<='0';
                    sdc_state <= spi_start;
                    level:=11;
                                        ------------------------------
-----------
                elsif level=11 then
                  data_in<="00000000";
                    ss<='0';
                    level:=12;
                    sdc_state <= spi_start;
                    cntff:=8;
                elsif level=12 then

                  if data_in="11111110" then      --    if clust >
"00000000000000000000000" then      pin<='1';   end if;
                     level:=13;
                        cnt:=0;
            else
                     if cntff>0 then
                      data_in<="11111111";
                        level:=12;
                        sdc_state <= spi_start;
                        ss<='0';
                        cntff:=cntff-1;
                        else
                         level:=9;
                        end if;
                    end if;
                  elsif level=13 then
                        sdc_state <= spi_start;
                         data_in<="11111111";
                         -- clust<=clust+1;
                         data_cnt:=0;
                        level:=14;
                elsif level=14 then
                        if cnt=511 then
                    --    led(7 downto 0)<=sec(0);
              --          i_lcd:=0;
                        --           arg(31 downto 0)<=clust(22
downto 0) & "000000000";
              --           sdc_state <= lcd_cmd;
                            sdc_state <=sel_arg;  --    led(7)<='1';
                         cmd_cnt:=0;
                         ss<='1';
                         sec(159)<=data_in;
                        elsif cnt<511 then

                              if cnt<159 then
                               sec(cnt)<=data_in;
                               level:=15;
                               else
                               sec(159)<=data_in;
                                level:=15;
                               end if;

                         end if;
```

```
                    elsif level=15 then
                          cnt:=cnt+1;
                          level:=13;

                    elsif level=151 then
                          cnt:=cnt+1;
                          data_cnt:=data_cnt+1;
                          level:=13;
                    ----------------------------------


                        ----------------write
                        elsif level=16 then

                              if cnt>0 then
                         data_in<="11111111";
                         ss<='1';
                         level:=16;
                         cnt:=cnt-1;
                         sdc_state <= spi_start;
                         else
                        level:=161;
                         end if;

                        elsif level=161 then           -- led(11 downto
0)<=arg(19 downto 8);
                               data_in<="01011000";
---0x58
                         ss<='0';
                         level:=17;
                         sdc_state <= spi_start;
                        elsif level=17 then
                               data_in<=arg(31 downto 24);
                         ss<='0';
                         level:=18;
                         sdc_state <= spi_start;
                        elsif level=18 then
                               data_in<=arg(23 downto 16);
                         ss<='0';
                         level:=19;
                         sdc_state <= spi_start;
                        elsif level=19 then
                               data_in<=arg(15 downto 8);
                         ss<='0';
                         level:=20;
                         sdc_state <= spi_start;
                        elsif level=20 then
                               data_in<=arg(7 downto 0);
                         ss<='0';
                         level:=200;
                         sdc_state <= spi_start;

                        elsif level=200 then
                               data_in<="11111111";
                         ss<='0';
                         level:=201;
                         sdc_state <= spi_start;
                        cnt:=2;     --15

                        elsif level=201 then
```

```
                        if data_in/="00000000" then
                        level:=21;
                        else
                        level:=161;
                        end if;

                      elsif level=21 then
               --           if data_in /= "11111111" then
                                    led(7 downto 0)<=data_in;
                                    led(11)<='1';
                                    led(10)<='1';
                                    led(9)<='0';
                                    led(8)<='0';
                           --              level:=22;
                                    --     else
                                    if cnt>0 then
                         data_in<="11111111";
                                           ss<='0';
                                     level:=21;
                                           cnt:=cnt-1;
                                           sdc_state <= spi_start;
                                     else
                                    --    cnt:=0;
                                    --    level:=16;
                                    level:=22;
                                 end if;
                               -- end if;
                      elsif level=22 then
                      data_in<="11111110";
                       ss<='0';
                       data_cnt:=0;
                       level:=23;
      --   wr_l_h_cnt:=0;
                       sdc_state <= spi_start;
                       cnt:=0;
                      elsif level=23 then
                           if cnt<512 then
                                          if cnt <159  then
                                                    --
                                                      level:=24;
----baraye ezafe kardan be cnt
                                                   ss<='0';

data_in<=sec(cnt);
                                                         sdc_state <=
spi_start;
                                                  else

data_in<="00000000";
                                                        level:=24;
----baraye ezafe kardan be cnt
                                                         ss<='0';
                                                         sdc_state <=
spi_start;
                                              end if;
                                    else
                                   level:=25;
                                   cnt:=2;
                                  end if;
                      elsif level=24 then
```

```
                                cnt:=cnt+1;
                                level:=23;
                                      elsif level=241 then
                                cnt:=cnt+1;
                                data_cnt:=data_cnt+1;
                                level:=23;


                        elsif level=25 then
                            if cnt>0 then
                                   data_in<="11111111";
                                   ss<='0';
                                   level:=25;
                                   cnt:=cnt-1;
                                   sdc_state <= spi_start;
                                   else
                                   sdc_state <= sel_arg;
                                   ss<='1';
                            --     if wr_cnt=1 then
                            --     level:=16;
                            --     cnt:=10;
                            --     wr_cnt:=wr_cnt +  1;
                            --     else
                                              pin<='1';
                            --     end if;
                                   end if;

                        end if;
                    ------------------------------------------ cmd(51)
                    when sel_arg =>
                            if s_level=0 then
                                          if sec(3 to 8) =
("01001101","01010011","01000100","01001111","01010011","00110101") then
                        --msdos5
                                           -- led(8 downto 0)<=clust(8 downto
0);
                         boot_address(31 downto 0)<=arg(31 downto 0);
                                           s_level:=1;
                                           else
                                          clust<=clust+1;
                                               level:=0;   ----read next block
                                               s_level:=0;
                                               sdc_state <= sd_state;
                                           end if;
                                elsif s_level=1 then
                                    led(13)<='1';
                                     --  num_boot_sec<=sec(14) & "000000000";--
-root sector
                                    --          fat_byte<=sec(22) & sec(23) &
"0000000000";     --tedade fat
                                    --          arg(31 downto 0)<=boot_address
+ num_boot_sec + fat_byte;
                                          arg(31 downto 0)<=boot_address +
(sec(14) & "000000000") + (sec(23) & sec(22) & "0000000000");
                                               offset:=0;
                                           s_level:=2;
                                           s_cnt:=0;
                                           level:=0;
                                               sdc_state <= sd_state;
                                elsif s_level=2 then
```

```
                                    if sec(s_cnt)=name(0) and sec(s_cnt
+ 1)=name(1) and  sec(s_cnt+2)=name(2) then                ----mem
                                      arg(31 downto 0)<=((sec(s_cnt +
26) &  sec(s_cnt + 27) & "000000000") + (boot_address));
                                             --   level:=0;
                                        cnt:=10;
                                             ss<='1';
                                    --    level:=16;
                                       level:=16;
                                                   write_sd:=1;
                                    --
      sec(0)<=new_data(0);
                                    --
      sec(1)<=new_data(1);
                                    --
      sec(2)<=new_data(2);
                                    --
      sec(3)<=new_data(3);
                                    --          sec(4)<=new_data(4);
                                    --
      sec(5)<=new_data(5);
                                                          wr_cnt:=1;
                                                          sec(0 to 159)
<= new_data(0 to 159);
                                                    sdc_state <=
sd_state;          led(12)<='1';
                                                    s_level:=22;
                                          else
                                               s_cnt:=s_cnt + 32;     -
-       led(11)<='1';
                                               s_level:=2;
                                               end if;


                                  elsif s_level=22 then


                                       s_level:=3;

                                        level:=0;

                                             sdc_state <= sd_state;

                                             write_sd:=1;

                                   elsif s_level=3 then

                                             i_lcd:=0;

                                      --     sdc_state <= lcd_cmd;
                                             cmd_cnt:=0;
                                   end if;
            ---------------------------------- sd controler
             when spi_start=>
                   bit_cnt <= 7;
                sclk<='0';
                sdc_state <= Start_Sample;

             when Start_Sample =>
```

```
                        sclk<='0';
                      mosi <= data_in(7);
                        for i in 7 downto 1 loop
                            data_in(i) <= data_in(i-1);
                        end loop;
                              sdc_state <= Start_Sample2;


              when Start_Sample2 =>
                          sclk<='1';
                           data_in(0) <=miso;

                            bit_cnt <=bit_cnt - 1;


                          if bit_cnt=0 then

                 sdc_state <= sd_state;

              else
               sdc_state <= Start_Sample;


                      end if;


--*********************************************************************--spi


end case;
end if;

--*********************************************************************--lcd
 end if;
end process;


End Gate_Level_1;
```

Write file name:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity sd is

   Port(
        clk,reset_ms:in std_logic;
              mosi: out std_logic;
        ss : out std_logic;
        miso : in std_logic;
        sclk : out std_logic;

       led:out std_logic_vector(13 downto 0);
```

```
       pin :out bit := '0');

End sd;

Architecture Gate_Level_1 of sd is
             type sector is array(0 to 159) of std_logic_vector(7 downto
0);
       Type State is
(spi_start,Start_Sample,Start_Sample2,sd_state,sel_arg);
       signal sdc_state : state:=sd_state;
             signal arg:std_logic_vector(31 downto 0):=(others=>'0');
       signal bit_cnt : integer := 7;
       signal b:integer:=0;
       signal c,q:std_logic:='0';
             signal data_in:std_logic_vector(7 downto 0);
             signal sec:sector:=(others=>"00100000");
             constant delay_for_e : integer := 2;     --18 * 25 NS = 450
NS
       constant delay_between_data : integer := 1250;
             type two_array is array (0 to 2)of std_logic_vector (7
downto 0);
             signal clust:std_logic_vector(22 downto 0):=(others=>'0');
       constant table_lcd : two_array :=("00111000", "00001110",
"00000001");--dastoorate lcd
             signal boot_address:std_logic_vector(31 downto
0):=(others=>'0');
             signal num_boot_sec:std_logic_vector(16 downto
0):=(others=>'0');
             signal fat_byte:std_logic_vector(25 downto
0):=(others=>'0');
             signal first_cluster:std_logic_vector(31 downto
0):=(others=>'0');
             type file_name is array (0 to 5) of std_logic_vector(7
downto 0);
             constant
name:file_name:=("01001101","01000101","01000101","00100000","00100000","
00100000"); --mee
              constant
new_name:file_name:=("01001101","01000101","01000101",
"01001101","00100000","00100000"); --mem
             type root_name is array (0 to 5) of std_logic_vector(7
downto 0);
             signal root: root_name
:=("01001101","01010011","01000100","01001111","01010011","00110101"); --
MSDOS5

Begin

 process(clk)
  begin
   if clk'event and clk='1' then

    if b=100 then
     b<=0;
     q<=c;
     c<=not c;
    else
     b<=b+1;

       end if;
```

```
  end if;
 end process;



Process(q,reset_ms)
    variable falling,rising:std_logic:='0';
       variable
x,y,aa,bb,level,cnt,cntff,cntloop,cmd_cnt,part,s_level,s_cnt,write_sd,wr_
cnt,offset, data_cnt:integer:=0;
    variable i_lcd, count_lcd,m : integer;
begin
--
**************************************************************************

              if q'event and q='1' then
--
**************************************************************************
if (reset_ms = '1') then
  sdc_state <= sd_state;

            level:=0;

--    e_lcd<='0';
--    rw_lcd<='0';
--    bl_lcd<='1';
           --
else
case sdc_state is
--
**************************************************************************



              when sd_state =>
                  if level=0 then
                     cnt:=10;
                         level:=1;
         elsif level=1 then
                      if cnt>0 then
                         data_in<="11111111";
                         ss<='1';
                         level:=1;
                         cnt:=cnt-1;
                         sdc_state <= spi_start;
                         else
                         level:=2;
                         end if;
          elsif level=2 then
                      data_in<="01000000";
                         ss<='0';
                         level:=3;
                         sdc_state <= spi_start;
                         cnt:=4;
                    elsif level=3 then
                      if cnt>0 then
                         data_in<="00000000";
```

```
                    ss<='0';
                    level:=3;
                    cnt:=cnt-1;
                    sdc_state <= spi_start;
                    else
                    level:=4;
                    end if;
                elsif level=4 then
                    data_in<="10010101";
                    ss<='0';
                    level:=5;
                    sdc_state <= spi_start;
            elsif level=5 then
                    if data_in="00000001" then
                          ss<='1';
                      level:=6;
        else
                      data_in<="11111111";
                         level:=5;
                         sdc_state <= spi_start;
                         ss<='0';
                      end if;
                    ----------------------------------------RESET
command (cmd0)

                elsif level=6 then
                     data_in<="01000001";
                       ss<='0';
                       level:=7;
                       sdc_state <= spi_start;
                       cnt:=4;
                 elsif level=7 then
                    if cnt>0 then
                       data_in<="00000000";
                       ss<='0';
                       level:=7;
                       cnt:=cnt-1;
                       sdc_state <= spi_start;
                       else
                       level:=72;
                       cntff:=8;
                       end if;
                 elsif level=72 then
                     data_in<="11111111";
                       ss<='0';
                       level:=8;
                       sdc_state <= spi_start;
                 elsif level=8 then
                    if data_in="00000000" then
                --     arg<=(others=>'0');

                 if write_sd=1 then
                     cnt:=10;
                     level:=16;
                        elsif write_sd=0 then
                         level:=9;
                            end if;
        else
                         if cntff>0 then
                            cntff:=cntff-1;
```

```
                          data_in<="11111111";
                             level:=8;

                             sdc_state <= spi_start;
                             ss<='0';
                             else
                             level:=6;
                             end if;
                           end if;
                           ----------------------------------------INIT
command       (cmd41)
                 elsif level=9 then

                   data_in<="01010001";
                      ss<='0';
                      level:=10;
                      if s_level=0 then
                      arg(31 downto 0)<=clust(22 downto 0) &
"000000000";
                      end if;
                      sdc_state <= spi_start;
                     -- cnt:=4;
                elsif level=10 then
              --  data_in<="00000000";   ---addrese morede nazar
                      data_in<=arg(31 downto 24);
                      ss<='0';
                      sdc_state <= spi_start;
                      level:=101;
                elsif level=101 then
               -- data_in<="00000001";   ---addrese morede nazar
                      data_in<=arg(23 downto 16);
                      ss<='0';
                      sdc_state <= spi_start;
                      level:=102;
                 elsif level=102 then
               -- data_in<="00001010";   ---addrese morede nazar
                      data_in<=arg(15 downto 8);
                      ss<='0';
                      sdc_state <= spi_start;
                      level:=103;
                elsif level=103 then
             --   data_in<="00000000";   ---addrese morede nazar
                      data_in<=arg(7 downto 0);
                      ss<='0';
                      sdc_state <= spi_start;
                      level:=11;
                                          ----------------------------
-----------
                elsif level=11 then
                  data_in<="00000000";
                      ss<='0';
                      level:=12;
                      sdc_state <= spi_start;
                      cntff:=8;
                elsif level=12 then

                   if data_in="11111110" then      --    if clust >
"00000000000000000000000" then      pin<='1';   end if;
                       level:=13;
                          cnt:=0;
```

```
              else
                           if cntff>0 then
                            data_in<="11111111";
                               level:=12;
                               sdc_state <= spi_start;
                               ss<='0';
                               cntff:=cntff-1;
                               else
                                level:=9;
                               end if;
                             end if;
                          elsif level=13 then
                                sdc_state <= spi_start;
                                 data_in<="11111111";
                                 -- clust<=clust+1;
                                 data_cnt:=0;
                                level:=14;
                      elsif level=14 then
                                if cnt=511 then
                           --     led(7 downto 0)<=sec(0);
                    --           i_lcd:=0;
                                --           arg(31 downto 0)<=clust(22
downto 0) & "000000000";
                    --           sdc_state <= lcd_cmd;
                                  sdc_state <=sel_arg;  --    led(7)<='1';
                                 cmd_cnt:=0;
                                 ss<='1';
                                 sec(159)<=data_in;
                                elsif cnt<511 then
                                      if cnt<159 then
                                       sec(cnt)<=data_in;
                                       level:=15;
                                       else
                                       sec(159)<=data_in;
                                        level:=15;
                                       end if;
                             end if;
                      elsif level=15 then
                            cnt:=cnt+1;
                            level:=13;

                      elsif level=151 then
                            cnt:=cnt+1;
                            data_cnt:=data_cnt+1;
                            level:=13;
                      ------------------------------------


                          ----------------write
                          elsif level=16 then

                                if cnt>0 then
                           data_in<="11111111";
                           ss<='1';
                           level:=16;
                           cnt:=cnt-1;
                           sdc_state <= spi_start;
                           else
                          level:=161;
                           end if;
```

```
                        elsif level=161 then           -- led(11 downto
0)<=arg(19 downto 8);
                               data_in<="01011000";
---0x58
                         ss<='0';
                         level:=17;
                         sdc_state <= spi_start;
                        elsif level=17 then
                               data_in<=arg(31 downto 24);
                         ss<='0';
                         level:=18;
                         sdc_state <= spi_start;
                        elsif level=18 then
                               data_in<=arg(23 downto 16);
                         ss<='0';
                         level:=19;
                         sdc_state <= spi_start;
                        elsif level=19 then
                               data_in<=arg(15 downto 8);
                         ss<='0';
                         level:=20;
                         sdc_state <= spi_start;
                        elsif level=20 then
                               data_in<=arg(7 downto 0);
                         ss<='0';
                         level:=200;
                         sdc_state <= spi_start;

                        elsif level=200 then
                               data_in<="11111111";
                         ss<='0';
                         level:=201;
                         sdc_state <= spi_start;
                        cnt:=2;     --15

                        elsif level=201 then
                          if data_in/="00000000" then
                          level:=21;
                          else
                          level:=161;
                          end if;

                        elsif level=21 then
                  --           if data_in /= "11111111" then
                                        led(7 downto 0)<=data_in;
                                        led(11)<='1';
                                        led(10)<='1';
                                        led(9)<='0';
                                        led(8)<='0';
                               --              level:=22;
                                        --     else
                                        if cnt>0 then
                         data_in<="11111111";
                                              ss<='0';
                                         level:=21;
                                              cnt:=cnt-1;
                                              sdc_state <= spi_start;
                                         else
                                        --    cnt:=0;
```

```
                                        --    level:=16;
                                        level:=22;
                                    end if;
                                -- end if;
                            elsif level=22 then
                            data_in<="11111110";
                             ss<='0';
                             data_cnt:=0;
                             level:=23;
      --   wr_l_h_cnt:=0;
                             sdc_state <= spi_start;
                             cnt:=0;
                            elsif level=23 then
                                  if cnt<512 then
                                                  if cnt <159  then
                                                             --
                                                               level:=24;
----baraye ezafe kardan be cnt
                                                          ss<='0';

data_in<=sec(cnt);
                                                                sdc_state <=
spi_start;
                                                        else

data_in<="00000000";
                                                               level:=24;
----baraye ezafe kardan be cnt
                                                                ss<='0';
                                                                sdc_state <=
spi_start;
                                                    end if;
                                     else
                                      level:=25;
                                      cnt:=2;
                                     end if;
                            elsif level=24 then
                                  cnt:=cnt+1;
                                  level:=23;
                                        elsif level=241 then
                                  cnt:=cnt+1;
                                  data_cnt:=data_cnt+1;
                                  level:=23;


                        elsif level=25 then
                            if cnt>0 then
                                   data_in<="11111111";
                                   ss<='0';
                                   level:=25;
                                   cnt:=cnt-1;
                                   sdc_state <= spi_start;
                                   else
                                   sdc_state <= sel_arg;
                                   ss<='1';
                            --     if wr_cnt=1 then
                            --     level:=16;
                            --     cnt:=10;
                            --     wr_cnt:=wr_cnt +  1;
                            --     else
```

```
                                        pin<='1';
                        --      end if;
                                end if;

                    end if;
            ------------------------------------------ cmd(51)
            when sel_arg =>
                    if s_level=0 then
                                    if sec(3 to 8) =
("01001101","01010011","01000100","01001111","01010011","00110101") then
                    --msdos5
                                     -- led(8 downto 0)<=clust(8 downto
0);
                     boot_address(31 downto 0)<=arg(31 downto 0);
                                      s_level:=1;
                                      else
                                    clust<=clust+1;
                                         level:=0;   ----read next block
                                         s_level:=0;
                                         sdc_state <= sd_state;
                                      end if;
                            elsif s_level=1 then
                                led(13)<='1';
                                 --  num_boot_sec<=sec(14) & "000000000";--
-root sector
                                --          fat_byte<=sec(22) & sec(23) &
"0000000000";     --tedade fat
                                --          arg(31 downto 0)<=boot_address
+ num_boot_sec + fat_byte;
                                   arg(31 downto 0)<=boot_address +
(sec(14) & "000000000") + (sec(23) & sec(22) & "0000000000");
                                         offset:=0;
                                      s_level:=2;
                                      s_cnt:=0;
                                      level:=0;
                                         sdc_state <= sd_state;
                            elsif s_level=2 then

                                    if sec(s_cnt)=name(0) and sec(s_cnt
+ 1)=name(1) and  sec(s_cnt+2)=name(2) then                 ----mem
                                --        arg(31 downto 0)<=((sec(s_cnt +
26) &  sec(s_cnt + 27) & "000000000") + (boot_address));
                                                --   level:=0;
                                         cnt:=10;
                                               ss<='1';
                                    --    level:=16;
                                       level:=16;
                                                     write_sd:=1;

      sec(s_cnt)<=new_name(0);

      sec(s_cnt+1)<=new_name(1);

      sec(s_cnt+2)<=new_name(2);

      sec(s_cnt+3)<=new_name(3);

      sec(s_cnt+4)<=new_name(4);

      sec(s_cnt+5)<=new_name(5);
```

```
                                                    wr_cnt:=1;
                                                    sdc_state <=
sd_state;          led(12)<='1';
                                                    s_level:=22;
                                              else
                                                   s_cnt:=s_cnt + 32;     -
-       led(11)<='1';
                                                   s_level:=2;
                                                   end if;


                              elsif s_level=22 then
                             s_level:=3;
                                      level:=0;
                                          sdc_state <= sd_state;
                                   write_sd:=1;

                               elsif s_level=3 then

                                                  i_lcd:=0;

                                  --      sdc_state <= lcd_cmd;
                                          cmd_cnt:=0;
                               end if;
            ----------------------------------- sd controler
             when spi_start=>
                   bit_cnt <= 7;
                sclk<='0';
                sdc_state <= Start_Sample;

             when Start_Sample =>
                              sclk<='0';
                            mosi <= data_in(7);
                              for i in 7 downto 1 loop
                                  data_in(i) <= data_in(i-1);
                              end loop;
                                    sdc_state <= Start_Sample2;



                  when Start_Sample2 =>
                              sclk<='1';
                               data_in(0) <=miso;

                                bit_cnt <=bit_cnt - 1;


                             if bit_cnt=0 then

                     sdc_state <= sd_state;

                  else
                   sdc_state <= Start_Sample;


                          end if;
```

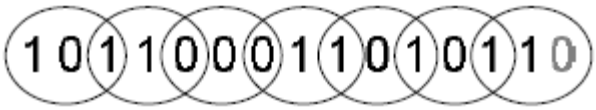
```
--*****************************************************************--spi

--               when lcd_cmd =>


      --                           if cmd_cnt<3 then
      --                               rs_lcd <= '0';
      --                               data_lcd(7 downto 0) <=
table_lcd(cmd_cnt)(7 downto 0);
      --                                sdc_state <= write_lcd;
      --                                cmd_cnt:=cmd_cnt+1;
      --                                else
      --                                    sdc_state <= start_lcd;
      --                                    end if;

      --                 when start_lcd =>

      --                           data_lcd(7 downto 0) <= sec(i_lcd)(7
downto 0);
      --
      --                              rs_lcd <= '1';
      --                              i_lcd := i_lcd + 1;
      --                              if i_lcd < 8 then
      --                                 sdc_state <= write_lcd;
      --                              else
      --                                    sdc_state <= stop_lcd;
     --                         end if;
          --
         --        when write_lcd =>
           --            if part = 0 then
                  --                    e_lcd <= '1';
                  --                       count_lcd := delay_for_e;
                  --                    part := part + 1;
                  --                       sdc_state <= delay_lcd;
                  --                 elsif part = 1 then
               --                    e_lcd <= '0';
               --                    count_lcd := delay_between_data;
               --                    sdc_state <= delay_lcd;
               --                    part := part + 1;
                  --                 else
               --                    sdc_state <= lcd_cmd;
               --                    part := 0;
                  --                 end if;

            --     when delay_lcd =>
                  --                 count_lcd := count_lcd - 1;
                  --                 if count_lcd = 0 then
              --                     sdc_state <= write_lcd;
         --       end if;

            --     when stop_lcd =>
end case;
end if;

--*****************************************************************--lcd
 end if;
end process;


End Gate_Level_1;
```

۳۹- ضرب بوث

مکانیزم‌های زیادی برای ضرب‌کردن اعداد باینری وجود دارد که اساس آنها تقریباً ثابت است و متشکل از یک‌سری ضرب‌های جزئی و جمع می‌باشد. اما شیوه‌هائی وجود دارد که سرعت و تعداد این ضرب‌ها را کاهش داده و موجب تسریع ضرب باینری می‌شود. در یکی از این روش‌ها که booth نام دارد تعداد ضرب‌های جزئی را کاهش می‌دهد و باعث تسریع ضرب می‌شود .

در روش booth که یکی از روش‌های ضرب سریع محسوب می‌شود ما ضرب‌های جزئی کمتری خواهیم داشت که می‌توان این ضرب‌های جزئی را به روش‌های مختلف با هم جمع کرده این روش هر چند روند ضرب را سریع‌تر می‌کند ولی آن را پیچیده‌تر خواهد کرد .

در روش کد کردن بوث به جای اینکه تک تک بیتهای ضرب کننده را در ضربشونده ضرب کنیم،برای کاهش تعداد حاصلضربهای جزئی به ترتیب زیر عمل میکنیم .ابتدا یک بیت صفر باید به سمت راست بیتهای ضرب کننده اضافه کنیم سپس بیتهای ضرب کننده را به دسته های سه تایی تقسیم می کنیم به طوری که یک بیت با یکدیگر همپوشانی داشته باشند و بیتها را از همان سمت راست دسته بندی میکنیم.(به شکل زیر)



سپس بر اساس الگوی دسته های به دست آمده در جدول زیر بیتهای ضربشونده را با هم جمع می کنیم.

با این روش تعداد ردیفهای حاصلضرب های جزئی نصف میشود.

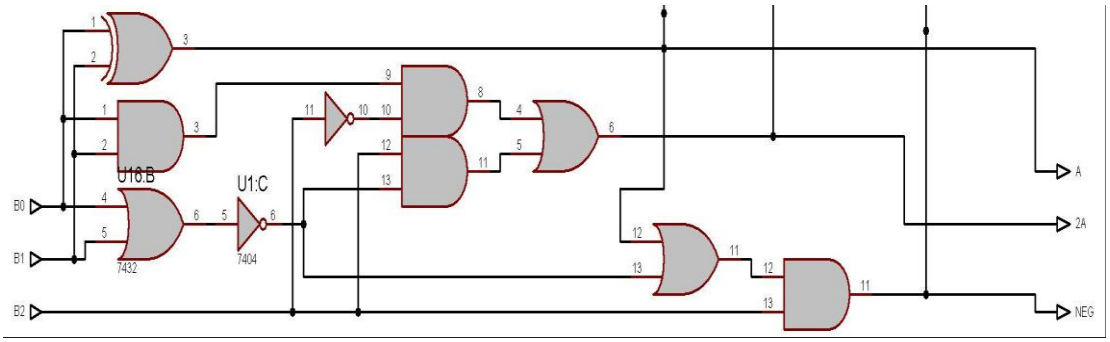
X_{i+1}	X	X_{i-1}	$Z_{i/2}$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

کد کردن بوث در دو مرحله انجام میشود :کدکننده ی بوث با امتحان کردن سه بیت ضرب کننده تصمیم میگیرد

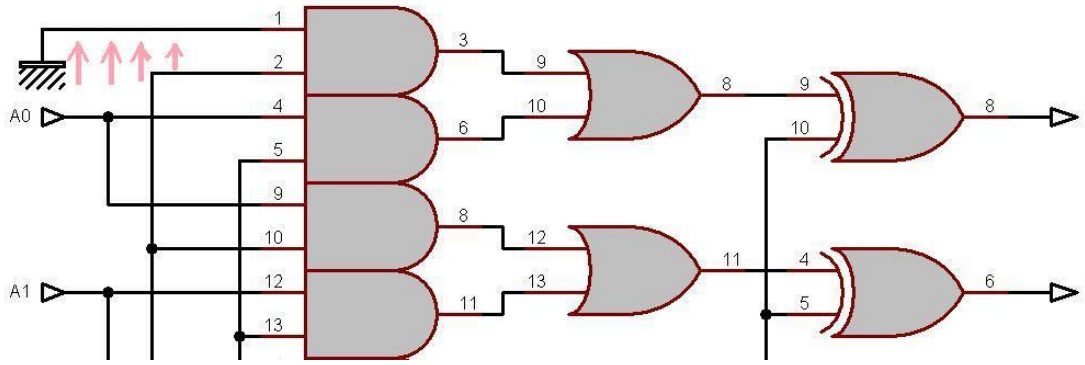
ضرب شونده، معکوس آن، دو برابر آن، و یا دو برابر معکوس آن و یا صفر باید با ردیف قبلی جمع شود، و انتخاب کننده ی بوث این عمل را انجام میدهد.

مدار کد کننده ی بوث در شکل زیر نشان داده شده است. در این شکل خروجی a سیگنال کنترلی است که نشان دهنده ی این است که خود ضرب شونده باید جمع شود.

خروجی 2a نشان دهنده ی این است که دو برابر ضرب شونده باید جمع شود، و خروجی NEG نشان دهنده ی این است که معکوس ضرب شونده باید جمع شود.



این سیگنال های کنترلی به زیر مدار انتخاب کننده ی بوث اعمال می شوند که آن مدار نیز در ورودی خود دو بیت ضرب شونده را دارد و در خروجی یک بیت از حاصلضربهای جزیی را تولید میکند .قسمتی از زیر مدار انتخاب کننده ی بوث در شکل فوق نشان داده شده است. که مربوط به a ،NEG، 2a است.



قابل ذکر است که یک بیت صفر در اینجا نیز باید به سمت راست ضرب شونده اضافه شود .در صورتی که 2a فعال باشد (برای ۲ برابر کردن).

مثال : $X=169\ ,\ Y=107$

Decimal

(4)

0 1 1

169 .0

× 107

1183

000

169

18083

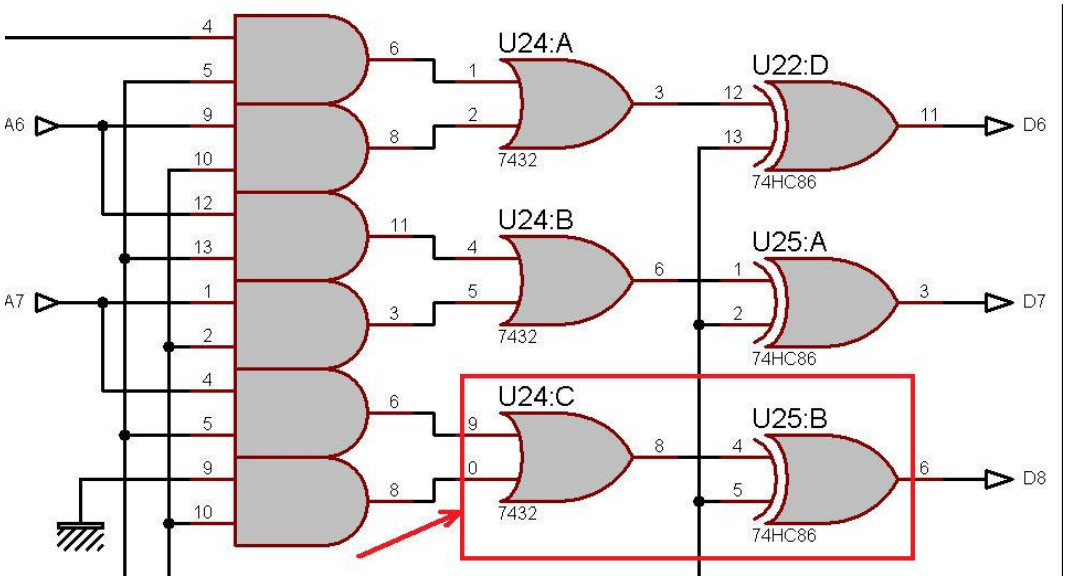
ابتدا به سمت راست Y یک صفر اضافه میکنیم سپس سه تا سه تا دسته بندی میکنیم . پس داریم

(1)	110	$-X$	1	-169	-169
(2)	101	$-X$	4	-169×4	-676
(3)	101	$-X$	16	-169×16	-2704
(4)	011	$+2X$	64	$2\times 169\times 64$	21632
				total action	18083

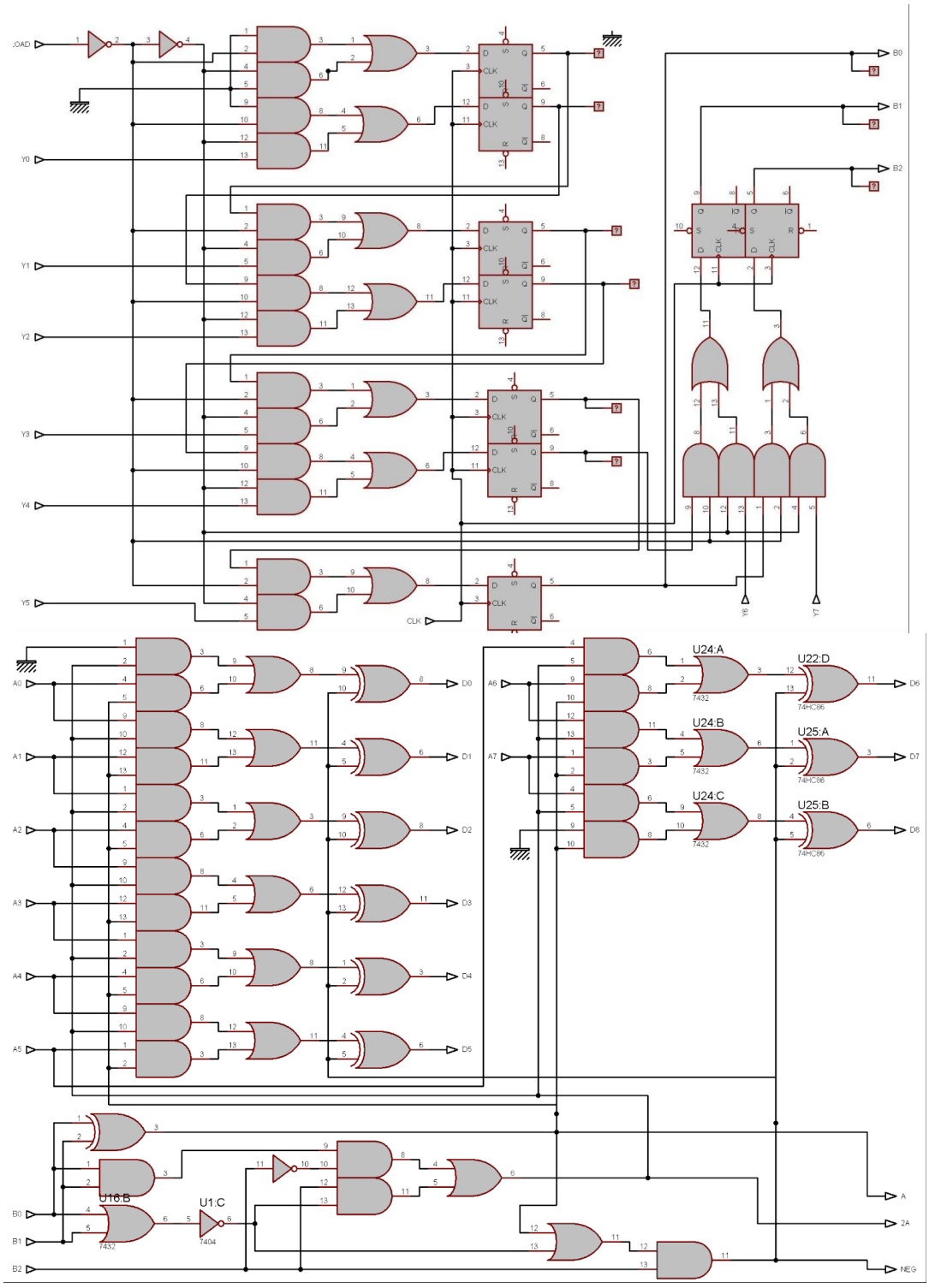
دقت میکنیم که بار اول ۰ برابر ، بار دوم ۴ برابر وبار سوم ۱۶ برابر و بار چهارم ۶۴ برابر میکنیم سپس آنها را جمع میکنیم . در این برنامه از شیفت برای چند برابر کردن استفاده کرده ایم .

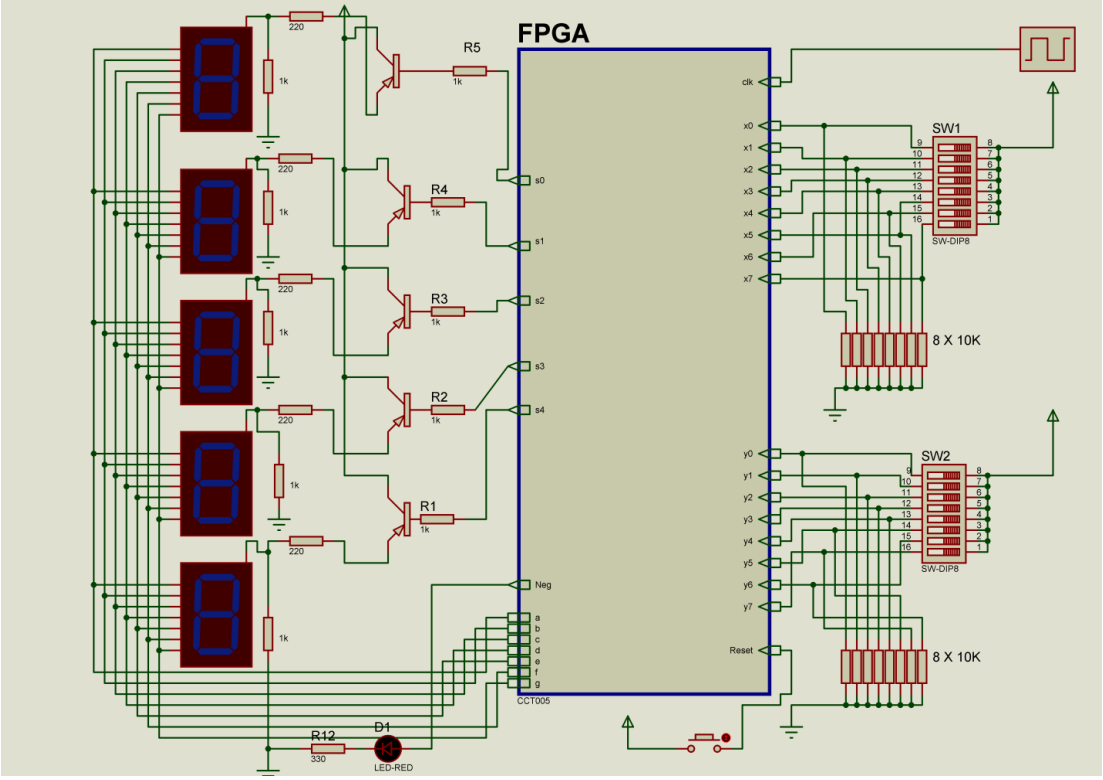
	10101001	
×	01101011	
	1111111101010111	$-X$
	11111101010111	$-X \times 4$
	111101010111	$-X \times 16$
	0101010010	$+2X \times 64$
	0100011010100011	

و قسمت زیر از مدار وظیفه گسترش بیت علامت عدد دوم را بر عهده دارد



درنهایت شکل کلی مدار از ترکیب این دو شکل حاصل می شود:


LOAD
Y0
Y1
Y2
Y3
Y4
Y5
CLK
B0
B1
B2
Y6
Y7
A0
A1
A2
A3
A4
A5
A6
A7
D0
D1
D2
D3
D4
D5
D6
D7
D8
U24:A
U24:B
U24:C
U22:D
U25:A
U25:B
7432
74HC86
U16:B
U1:C
7404
A
2A
NEG



کد vhdl:

----------Register 16 bit

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity reg16 is

port( I:    in std_logic_vector(15 downto 0;(
          clock:  in std_logic;
            set: in std_logic;

         Q: Buffer std_logic_vector(15 downto 0):="0000000000000000;("
end reg16;
architecture register16 of reg16 is
begin
    process(clock,I,set(
    begin
      if(clock='1' and clock'event) then
 --    if set =1 set outpot ="000000000000000"
      if set='1' then
            Q <= "0000000000000000;"
            else
             --set outpout = inpout
            Q <= I;
            end if;
        end if;
    end process;
```

```
end Register16;

 ----------------------------------------Register 8 bit
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity reg8 is
port( I:    in std_logic_vector(7 downto 0;(
          clock:  in std_logic;
            set: in std_logic;

         Q: Buffer std_logic_vector(7 downto 0):="00000000;("
end reg8;
architecture register8 of reg8 is
begin
    process(clock,I,set(
    begin

      if(clock='1' and clock'event) then
 --   if set =1 set outpot ="000000000"
   if set='1' then
            Q <= (others => '0;('
            else
             --set outpout= inpout
            Q <= I;
            end if;

         end if;
    end process;
end Register8;

 ---------------------------------------FullAdder 1 bit
library ieee;
use ieee.std_logic_1164.all;
entity FAdder1 is
  port (a, b, Cin : in std_ulogic; Sum, Cout: out std_ulogic;(
end entity FAdder1;

architecture FullAdder1 of FAdder1 is
begin
--anjame  amaliate  f.a  1bit
  Sum <= a xor b xor Cin;
  Cout <= (a and b) or (a and Cin) or (b and Cin;(
end architecture FullAdder1;

 ------------------------------------FullAdder 16 Bit
library ieee;
use ieee.std_logic_1164.all;
entity fadder16 is
  port (a:in std_logic_vector(15 downto 0;(
            b:in std_logic_vector(15 downto 0;(
            cin :in std_logic;
        o:out std_logic_vector(15 downto 0;((
end entity fadder16;
architecture FullAdder16 of fadder16 is
      SIGNAL c: std_logic_vector(0 to 14); -- internal carry
      signal cout : std_logic;
begin
       ١٦ --bit start bit + 14 internal bit + end bit
```

```
stage_1   :entity WORK.FAdder1 port map(a(0),b(0),cin,o(0),c(0;((
      jame :  for i in 1 to 14 generate
      oo : entity WORK.FAdder1 port map(a(i),b(i),c(i-1),o(i),c(i;((
                                    end generate jame ;

stage_2   :entity WORK.FAdder1 port map(a(15),b(15),c(14),o(15),cout;(

end architecture FullAdder16;

  ----------------------------------------Counter 4 Bit
library ieee;
use ieee.std_logic_1164.all, ieee.numeric_std.all;
entity counter4 is
  port(clk : in std_ulogic;
       reset : in std_logic ;
       count : out std_ulogic_vector(3 downto 0;((
end counter4;
architecture rtl of counter4 is
begin
 --counter 4 bity 0000 to 1111
p0: process (clk,reset (
      variable cnt : unsigned(3 downto 0):= (others => '0;('
    begin
      if (reset='1') then
        cnt := "0000;"
     elsIF (clk = '1' AND clk'EVENT) THEN
        cnt := cnt + 1;
      end if;
      count <= std_ulogic_vector(cnt ;(
    end process p0;
end rtl;

 ----------------------------------------Both Detector
--3 bit b ra migirad moayan mikonad ke bayad dar 2a,-2a,a,-a,0 zarbe badi
anjam girad
library IEEE ;
use IEEE.std_logic_1164.all ;
ENTITY Booth_Calc IS
     PORT(B: in std_logic_vector(2 downto 0;(
     A1,A2,Neg :  out std_logic := '0 ;(   '
END Booth_Calc ;
ARCHITECTURE behavioral OF Booth_Calc IS
       SIGNAL S: std_logic_vector(6 downto 0;(
      BEGIN
            S(0)<= B(0) Xor B(1;(
            S(1)<= B(0) And B(1;(
            S(2)<= B(0) Nor B(1;(
            S(3)<= Not B(2;(
            S(4)<= S(1)And S(3;(
            S(5)<= B(2) And S(2;(
            A2  <= S(4) Or S(5;(
            S(6)<= S(0) Or S(2;(
            Neg <= B(2) And S(6;(
            A1<=S(0;(
END behavioral;

 ----------------------------------Booth Activer
library IEEE ;
use IEEE.std_logic_1164.all ;
ENTITY BoothAct IS
```

```
     PORT(A1,A2,Neg: in std_logic;
        A : in Std_Logic_vector(7 downto 0;(
     D :  BUFFER std_logic_vector(15 downto 0 ;(  (
--d hasele khoruji a,2a,-2a,-a,0 bar asas vurudihaye neg, a1,a2 mibashad
END BoothAct ;
 --Make X, 2X, 0, -x, -2x by Bouth detector result
ARCHITECTURE behavioral OF BoothAct IS
       SIGNAL S1: std_logic_vector(8 downto 0):="000000000;"
       SIGNAL S2: std_logic_vector(8 downto 0):="000000000;"
       SIGNAL S3: std_logic_vector(8 downto 0):="000000000;"
      BEGIN
      S1(0)<= '0;'
      S2(0)<= A(0)  and A1;
      S3(0)<= '0'   Or  S2(0;(
      D(0) <= S3(0) XOr Neg;

      S1(1)<= A(0)  and A2;
      S2(1)<= A(1)  and A1;
      S3(1)<= S1(1) Or  S2(1;(
      D(1) <= S3(1) XOr Neg;

      S1(2)<= A(1)  and A2;
      S2(2)<= A(2)  and A1;
      S3(2)<= S1(2) Or  S2(2;(
      D(2) <= S3(2) XOr Neg;

      S1(3)<= A(2)  and A2;
      S2(3)<= A(3)  and A1;
      S3(3)<= S1(3) Or  S2(3;(
      D(3) <= S3(3) XOr Neg;

      S1(4)<= A(3)  and A2;
      S2(4)<= A(4)  and A1;
      S3(4)<= S1(4) Or  S2(4;(
      D(4) <= S3(4) XOr Neg;

      S1(5)<= A(4)  and A2;
      S2(5)<= A(5)  and A1;
      S3(5)<= S1(5) Or  S2(5;(
      D(5) <= S3(5) XOr Neg;

      S1(6)<= A(5)  and A2;
      S2(6)<= A(6)  and A1;
      S3(6)<= S1(6) Or  S2(6;(
      D(6) <= S3(6) XOr Neg;

      S1(7)<= A(6)  and A2;
      S2(7)<= A(7)  and A1;
      S3(7)<= S1(7) Or  S2(7;(
      D(7) <= S3(7) XOr Neg;

      S1(8)<= A(7)  and A2;
      S2(8)<= '0'  and A1;
      S3(8)<= S1(8) Or  S2(8;(
      D(8) <= S3(8) XOr Neg;
            D(9)  <= Neg;
            D(10) <= Neg;
            D(11) <= Neg;
            D(12) <= Neg;
            D(13) <= Neg;
```

```
            D(14) <= Neg;
            D(15) <= Neg;

END behavioral;

-------------------------------------------—-----------------------------------
 ---------Add Shift Register
--16 bit fa + 16 bit shift adder ya vorudi clk shift reg shod era *4
karde +vurudi
library IEEE ;
use IEEE.std_logic_1164.all ;
ENTITY Add_Sh_Reg IS
     PORT(clk,Load,Neg : in std_logic;
     D : in std_logic_vector(15 downto 0;(
        Qout :  BUFFER std_logic_vector(15 downto 0 ;(  (
END Add_Sh_Reg ;
 ١١ --bit full adder + 16 bit shif register
ARCHITECTURE behavioral OF Add_Sh_Reg IS
       SIGNAL Add,B: std_logic_Vector(15 downto 0):="0000000000000000;"
       SIGNAL O: std_logic_Vector(15 downto 0):="0000000000000000;"
           signal Reset : std_logic  ;
      BEGIN
      Reset<= Load;
             --move result 2 bit left (shiftleft 2 bit)= * 4
            B(15 downto 0)<= O(13 downto 0)&"00;"
      A : entity WORK.fAdder16 port map(D(15 downto 0),B,Neg,Add;(
      C : entity WORK.Reg16 port map(Add,Clk,Reset,O;(
            QOut<=O;
END behavioral;
-------------------------------------------------------------------------------
 ------------Booth Spliter
--tavasot shift 2 biti meghdar vurudi tashkhis dahande zarbe booth ra
faraham mikonad
library IEEE ;
use IEEE.std_logic_1164.all ;
ENTITY Booth_Split IS
     PORT(clk,Load : in std_logic;
        Y : in std_logic_vector(7 downto 0  ;(
                  B : BUFFER std_logic_Vector(2 downto 0) := "000 ;(   "
END Booth_Split ;
ARCHITECTURE behavioral OF Booth_Split IS
       SIGNAL S,O: std_logic_Vector(7 downto 0):="00000000;"
      BEGIN
       --   split 3 bit from left to right of y
            S <= Y when Load='1' else O(5 downto 0)&"00;"
      a : entity WORK.Reg8 port map(S,Clk,'0',O;(
      B(2 downto 0)<= O(7 downto 5;(
END behavioral;

-------------------------------------------------------------------------------
 ------------Clock Controler
library IEEE ;
use IEEE.std_logic_1164.all ;
ENTITY Clk_Control IS
     PORT(clk,Reset : in std_logic;
     Clkout,Load :  Out std_logic ;(
END Clk_Control ;
ARCHITECTURE behavioral OF Clk_Control IS
       constant MaxCount:integer:=5;
           signal EnClk,clk2,Max : std_logic:='0  ;'
```

```
       Signal C : std_ulogic_vector(3 downto 0):="0000;"
      BEGIN
       --   make only 4 clock for calclute after load
      Load <= '1' when Reset='1' else '0;'
      Max<='1' when C(2)='1' else '0;'
      a : entity WORK.Counter4 port map(EnClk,Reset,C;(
      EnClk<= (Not Max) Nand Clk2;
      ClkOut<=EnClk;
      process (clk(
      variable Count:integer:=0 ;
      begin
            if (clk='1' and clk'event) then
                        if Count <MaxCount then
                              Count:=Count+1;
                        else
                        Count:=0;
                        Clk2<= Not Clk2;
                                                end if;
            end if;
      end process;
END behavioral;

 --------------------------------Posetive Maker 8 bit
library IEEE ;
use IEEE.std_logic_1164.all ;
use ieee.std_logic_unsigned.all;

ENTITY Posetive8 IS
     PORT(res,clk  : in std_logic;
             Xin  :  in  std_logic_vector(7 downto 0;(
          xOut :  out std_logic_vector(7 downto 0 ;(  (
END Posetive8 ;
ARCHITECTURE behavioral OF Posetive8 IS
BEGIN
             --make Posetive number if it is negetive
            process (clk,res(
      variable Count:integer:=0 ;
      begin
--          mokamel 2
               if (clk='1' and clk'event) then
            if Xin(7)='1' then
                        XOut<= (Xin xor "11111111")+1;
                  else
                        XOut<= XIn;
                     end if;

                end if;
      end process;
END behavioral;


 ---------------------------------Posetive Maker 16 bit
library IEEE ;
use IEEE.std_logic_1164.all ;
use ieee.std_logic_unsigned.all;

ENTITY Posetive16 IS
     PORT(clk  : in std_logic;
             Neg  : in std_logic:='0;'
             Xin  :  in  std_logic_vector(15 downto 0;(
```

```
          xOut :  buffer std_logic_vector(15 downto 0 ;(  (
END Posetive16 ;
ARCHITECTURE behavioral OF Posetive16 IS
      BEGIN
             --make Posetive number if it is negetive
      process (clk(
      variable Count:integer:=0 ;
      begin
                if (clk='1' and clk'event) then

             --      Mokamel 2
            if Neg='1' then
                              XOut<= (XIn xor "0111111111111111")+1;
                  else
                           XOut<= XIn;
                         end if;

                  end if;
      end process;
END behavioral;


 ------------------------------------------Main Madule

library IEEE ;
use IEEE.std_logic_1164.all ;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity bmul8 is
 port (x : in  std_logic_vector(7 downto 0):=(others=>'0   ;('
       y : in  std_logic_vector(7 downto 0):=(others=>'0 ;('
             Negetive:out std_logic;
             reset : in std_logic;
             clk : in std_logic ;
       S: out std_logic_vector(4 downto 0;(
             Seg: Out std_logic_vector(7 downto 0;(
             led: out std_logic_vector(7 downto 0;((
--       outpot: buffer std_logic_vector(15 downto 0  ;((
end entity bmul8 ;
architecture B of bmul8 is
   type State_Type is (Start,Calc,Show;(
      signal State:State_Type:=Start;
      Signal D: std_logic_vector (15 downto 0):="0000000000000000;"
   Signal XP,YP: std_logic_vector(7 downto 0):="00000000;"
   Signal B: std_logic_vector(2 downto 0):="000;"
   Signal Clkout,Load,A1,A2,Neg: std_logic:='0;'
   Signal PP: std_logic_vector(15 downto 0;(
   signal p : std_logic_vector(15 downto 0;(
      signal Digit,dig0, dig1, dig2, dig3,dig4 : std_logic_vector(3
downto 0;(
      Signal NegOut:Std_logic:='0;'
begin
            process (clk,reset(
            variable i:integer:=0;
            begin
            if reset='1' then
                  S<="11111;"
                  Dig0<="0000;"
                  Dig1<="0000;"
                  Dig2<="0000;"
```

```
                Dig3<="0000;"
                Dig4<="0000;"
          elsif (clk='1' and clk'event) then
                case State is
                     when Start<=
                             led<= xp(7 downto 0;(
                                   P(15 downto 0)<= '&'·pp(14 downto
0;(
--                                  P(15 downto 0)<= ";"·····\·····
                                  S<="11111;"
                                  Dig0<="0000;"
                                  Dig1<="0000;"
                                  Dig2<="0000;"
                                  Dig3<="0000;"
                                  Dig4<="0000;"
                                  State<=Calc;
                           when Calc<=
                                   if p > 9999 then
                                         P<=P-10000;
                                      Dig4<=Dig4+1;
                                  elsif P>999 then
                                               P<=P-1000;
                                            Dig3<=Dig3+1;
                                         elsif p>99 then
                                                   P<=P-100;
                                              Dig2<=Dig2+"0001;"
                                           elsif p>9 then
                                                      P<=P-10;

Dig1<=Dig1+"0001;"
                                                    else
                                                       Dig0<=P(3
downto 0;(

State<=Show;
                                                          i:=0;
                                                    end if;
                                  when Show<=
                                       if i<2000 then
                                                  Digit<=Dig0;
                                                  S<="11110;"
                                                         i:=i+1;
                                         elsif i<4000 then
                                                       Digit<=Dig1;
                                                       S<="11101;"
                                                        i:=i+1;

                                                elsif i<6000 then
                                                       Digit<=Dig2;
                                                       S<="11011;"
                                                        i:=i+1;
                                                     elsif i<8000 then

      Digit<=Dig3;

      S<="10111;"
                                                             i:=i+1;
                                                    elsif i<10000
then
```

```
      Digit<=Dig4;

      S<="01111;"

      i:=i+1;
                                                                  elsif
i<12000 then

S<="11111"  ;

 i:=i+1;

      else

  i:=0;

State<=Start;
                                                                  end if;
                                  when others<=
                                  State<=Start;
                  end case;
                  NegOut<= x(7) xor y(7;(
            end if;
            end process ;
            PX: entity work.Posetive8 port map (reset,clk,X,Xp;(
            PY: entity work.Posetive8 port map (reset,clk,y,yp;(
            ClockControler : entity WORK.Clk_Control      port map
(clk,Reset,ClkOut,Load;(
            BoothSpliter   : entity WORK.Booth_split      port map
(clkOut,Load,Yp,B;(
      BoothDetector  : entity WORK.Booth_Calc       port map
(B,A1,A2,Neg;(
      BoothActor                    : entity WORK.BoothAct   port map
(A1,A2,Neg,Xp,D;(
      Adder_Shift_Register : entity WORK.Add_Sh_Reg port map
(clkOut,Load,Neg,D,PP;(
            Negetive<=NEgOut;
            --p(14 downto 0)<=PP(14 downto 0;(
            --Pout: entity work.Posetive16 port map
(clk,NegOut,p,outpot;(
            seg <= "11000000" when digit = "0000" else
                   "١١١١١٠٠١" when digit = "0001" else
                         "١٠١٠٠١٠٠" when digit = "0010" else
                         "١٠١١٠٠٠٠" when digit = "0011" else
                         "١٠٠١١٠٠١" when digit = "0100" else
                         "١٠٠١٠٠١٠" when digit = "0101" else
                         "١٠٠٠٠١٠" when digit = "0110" else
                         "١١١١١٠٠٠" when digit = "0111" else
                         "١٠٠٠٠٠٠٠" when digit = "1000" else
                         "١٠٠١٠٠٠٠" when digit = "1001" else
                         "١١١١١١١١" when digit = "1010" else
                         "١١١١١١١١" when digit = "1011" else
                         "١١١١١١١١" when digit = "1100" else
             "١١١١١١١١"          when digit = "1101" else
             "١١١١١١١١"          when digit = "1110" else
             "١١١١١١١١"          when digit = "1111;"
end architecture;
```

۴۰- دمو برای بورد آموزشی

در این برنامه، اجزاء خروجی بورد شامل LCD، LED ها، Seven segment، بازر و حالت خروجی پورت-های کاربر تست می‌شوند.

با همان شروع کار روی LCD اطلاعاتی از نام بورد و وب سایت به صورت روان می‌آید و در ادامه اجزاء بورد و امکانات آن پشت سر هم معرفی می‌شوند. در حین معرفی LCD و در قسمت dimming آن، نور Backlight به تدریج کم می‌شود (با روش PWM) و بعد به حالت عادی بر می‌گردد. در معرفی seven segment، نور آن که از ابتدا کم نگه داشته شده بود، زیاد می‌شود و دونقطه (colon) وسط آن چشمک می‌زند و سپس در قسمت dimming، نور به تدریج کم می‌شود. در معرفی LED ها، در ابتدا تمام LED ها که قبلاً با نور کم به حالت رقص نور روشن و خاموش می‌شدند، با نور کامل روشن می‌شوند و سپس در قسمت dimming، نورشان به تدریج کم شده و سپس به حالت عادی بر می‌گردند. در قسمت معرفی Buzzer هم ابتدا بازر دو بار با قدرت زیاد به صدای در می‌آید و پس از آن صدای چند تن (tone) پشت سر هم با قدرت کم می‌آید. باز هم از روش PWM برای کنترل قدرت صدا استفاده شده است. اگر صدای بلند بازر موجب ناراحتی می‌شود یا کلاً نیازی به بازر ندارید می‌توانید با کلیدی که در بالای آن قرار داده شده، آن را از حالت standby (آماده به کار و منتظر دستور از FPGA)، به حالت off ببرید و در موقع نیاز برگردانید. در برنامه‌هایی که نیاز به بازر ندارند، برای اینکه نویز باعث ایجاد صداهای احتمالی ریز مزاحم نشود، بهتر است آن را داخل برنامه غیر فعال کنید (با فرستادن '1' منطقی) و به عهده‌ی کاربر و خاموش کردن سخت‌افزاری (توسط کلید) نگذارید.

در کل مدت زمان اجراء، پورت‌ها شامل یک پورت ۱۰ پینی باکسی (با ۸ پایه ورودی/خروجی)، یک پورت ۲۰ پینی باکسی (با ۱۸ پایه ورودی/خروجی) و یک پورت مادگی نظامی ۱۰ پینی با فرکانس 1^{Hz}، '0' و '1' می‌شوند. پورت پین گردی هم با فرض قرار دادن اسیلاتور 20^{MHz} در سوکت اسیلاتور به طریق مشابه نوسان می‌کند.

در این برنامه همچنین پورت‌های TV، VGA(monitor)، Serial و دو پورت PS/2 فقط در حد اطمینان از درست بودن اتصالات و نه امتحان کامل چک می‌شوند. با این کار می‌توان از صحت این پورت‌ها (برای بعضی از آنها که هم ورودی و هم خروجیند، صحت خروجیشان) اطمینان حاصل کرد. عملکرد اصلی آنها را در مثال‌های دیگر تست کرده‌ایم.

برای پورت TV، تک خروجی پورت که composite نام دارد به صورت آنالوگ، بین 0V تا 1V با فاصله‌ی 0.3V در ابتدا (بین 0V تا 0.3V) و از 0.3V تا 1V با فاصله‌ی 0.1V در فاصله‌های زمانی ۱ ثانیه تغییر می‌کند و این کار تکرار می‌شود. برای چک کردن این قسمت، از مالتی‌مترهای کند استفاده نکنید. در غیر اینصورت باید فاصله‌ی زمانی را از داخل برنامه زیاد کنید و برای چک کردن، زمان بیشتری معطل بمانید!

برای پورت VGA پایه‌های R، G و B (پایه‌های ۱، ۲ و ۳) آن به صورت آنالوگ بین 0V تا 3.3V با فاصله-های زمانی ۱ ثانیه تغییر می‌کنند (در ۱۶ پله) و پایه‌های hsync و vsync (۱۳ و ۱۴) آن هم با فرکانس 1^{Hz}، '0' و '1' می‌شوند (این پایه‌ها دیجیتال هستند). برای تشخیص پایه‌ها اگر از روبرو به پورت نگاه کنید پایه‌ی سمت راستی ردیف بالا پایه‌ی ۱، پایه‌ی سمت راستی ردیف وسط پایه‌ی و پایه‌ی سمت چپی ردیف پایین پایه‌ی ۱۵ می‌باشد.

برای پورت سریال، پایه‌ی ۳ (TXD) با فرکانس 1^{Hz}، '0' و '1' می‌شود (البته با استاندارد RS232 که مقادیر منفی را هم شامل می‌شود) و اگر آن را با سیم به پایه‌ی ۲ (RXD) متصل کنید در طول زمان اتصال، به شرط اینکه کلید فشاری متصل به پورت 8 را فشار دهید و نگه دارید، بازر با فرکانس 1^{Hz} قطع و وصل می-شود (در این حالت، ورودی این پورت هم چک می‌شود). برای تشخیص پایه‌ها اگر از روبرو به پورت نگاه کنید پایه‌ی سمت چپ ردیف بالا (ردیف ۵ تایی) پایه‌ی شماره‌ی ۱ می‌باشد و پایه‌ی سمت چپ ردیف پایین (ردیف ۴ تایی) پایه‌ی شماره‌ی ۶ می‌باشد. پایه‌ی ۵ هم به زمین متصل شده است.

برای پورت‌های PS/2، دو پایه‌ی data و clock (پایه‌های ۱ و ۳) با فرکانس 1^{Hz}، '0' و '1' می‌شوند. برای تشخیص پایه‌ها اگر از روبرو به پورت نگاه کنید، پایه‌ی بالا سمت راست، پایه‌ی ۱ و پایه‌ی پایین سمت

راست، پایه‌ی ۳ می‌باشد (شمارش پایه‌ها را در جهت حرکت عقربه‌های ساعت انجام دهید). پایه‌های ۲ و ۵ هم مربوط به زمین و تغذیه می‌باشند.

کلید فشاری متصل به پایه‌ی 41، برای ریست می‌باشد که لزومی ندارد که در ابتدای کار حتماً فشرده شود.

برای پالس ساعت هم از هر سه اسیلاتور داخل بوردی 1^{MHz}، 10^{MHz} و 50^{MHz} و همچنین سوکت اسیلاتور که در اختیار کاربر است و در بعضی بوردها یک اسیلاتور (معمولاً مربعی 20^{MHz}) در آن قرار داده شده استفاده شده است. برای پورت‌های VGA، TV، Serial و PS/2 ها از فرکانس 1^{MHz}، برای پورت‌ها بجز پورت پین گرد که از سوکت کاربر (20^{MHz}) استفاده شده، از فرکانس 50^{MHz} و برای بقیه‌ی اجزاء بورد از فرکانس 10^{MHz} استفاده شده است.

برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity demo is
   Port (
                    --lcd--
                    data : out std_logic_vector(7 downto 0);
                    rs : out std_logic;
                    rw : out std_logic;
                    e : out std_logic;
                    bl : out std_logic;

                    --general--
                    clk_1M : in std_logic;
                    clk_10M : in std_logic;
                    clk_50M : in std_logic;
                    clk_20M : in std_logic;
                    reset : in std_logic;

                    --seven segment--
                    seg: out std_logic_vector (3 downto 0);
                    seven_seg : out std_logic_vector (7 downto 0);

                    --leds--
                    leds : out std_logic_vector (15 downto 0);

                    --buzzer--
                    pwm_buz : out std_logic;

                    --tv--
                    tv : out std_logic_vector(7 downto 0);
```

```
                    --vga--
                    hsync, vsync : out std_logic;
                    blue, green, red : out std_logic_vector(3 downto 0);

                    --ports--
                    port18: out std_logic_vector(17 downto 0);
                    port8: out std_logic_vector(7 downto 0);
                    port10: out std_logic_vector(9 downto 0);
                    port1: out std_logic;

                    --serial--
                    txd :   out std_logic;
                    rxd:   in std_logic;
                    serial_test_pb:     in std_logic;
                    --ps2--
                           ps2data1, ps2clock1, ps2data2, ps2clock2:       out
std_logic

             );

end demo;

architecture Behavioral of demo is

   constant datamarq : integer := 437;   --number of data to be showed in LCD
   constant delay_FFFF : integer := 65536;
   constant delay_0_2s : integer := 2000000;
   constant delay_2s : integer := 20000000;
   constant delay_4000 : integer := 16384;
   constant delay_0_5s : integer := 5000000;
   constant delay_3s : integer := 30000000;
   constant delay_4_5s : integer := 45000000;
   constant delay_0_4s : integer := 4000000;
   signal colon, pwm_led:  std_logic;
   signal seven_seg1 :  std_logic_vector (6 downto 0);
   signal pwm_led_vec :  std_logic_vector (15 downto 0);
   signal e1 : std_logic;
   signal digit, pwm_7seg:  std_logic_vector (3 downto 0);
   signal leds1:  std_logic_vector (15 downto 0);
      signal buzzer_signal, buzzer_pwm :      std_logic;
      signal blink_1M :   std_logic;
      signal blink_20M :  std_logic;
      signal blink_50M :  std_logic;
      signal color : std_logic_vector(3 downto 0);

      type state is (start, write, delay, stop);
   signal current, return_state : state;
   type arraymarq is array (0 to datamarq - 1)
                           of std_logic_vector (8 downto 0);
   constant tablemarq : arraymarq := (
               "000111000", "000001100", "000000001",       "010000111",
               "101000001", "000011000", "101101101", "000011000", --Am
                           "101001111", "000011000", "101001100",  --OL
               "000011000", "100100000", "000011000", "101100101", -- e
            "000011000", "101101100", "000011000", "101100101",  --le
            "000011000", "101100011", "000011000", "101110100", --ct
            "000011000", "101110010", "000011000", "101101111", --ro
               "000011000", "101101110", "000011000", "101101001", --ni
            "000011000", "101100011", "000011000", "101110011", --cs
            "000011000", "100100000", "000011000", "100100000", -- 2 spaces
      --    "000011000", "100100000", "000011000", "100100000", -- 2 spaces
            "000011000", "101110111", "000011000", "101110111", --ww
            "000011000", "101110111", "000011000", "100101110", --w.
               "000011000", "101100001", "000011000", "101101101", --am
            "000011000", "101101111", "000011000", "101101100",  --ol
```

```
"000011000", "100101101", "000011000", "101100101",  -- -e
"000011000", "100101110", "000011000", "101100011", --.c
"000011000", "101101111", "000011000", "101101101", --om
"000011000", "100100000", "000011000", "100100000", -- 2 spaces
"000011000", "100100000", "000011000", "100100000", -- 2 spaces
"000011000", "100100000", "000011000", "100100000", -- 2 spaces
"000011000", "100100000", "000011000", "100100000", -- 2 spaces
       --     "000011000" --shift

              "000000001", --83
"101001100", "101000011", "101000100", "100100000", --LCD
"100100000", "100100000", "100100000", "100100000",
              "011000000", --92
"100100000", "100110010", "100101010", "100111000", --2*8
              "100100000", "100100000", "100100000", "100100000",
              "011000000",  --101
 "101100100", "101101001", "101101101", "101101101", --dimm
"101101001", "101101110", "101100111",  "100100000", --ing

              "000000001", --110
"100110111", "101010011", "101100101", "101100111", --7Seg
"101101101", "101100101", "101101110", "101110100", --ment
              "011000000", --119
                           --2nd line
"100100000", "100110100", "100101101", "101100100", -- 4-d
"101101001", "101100111", "101101001", "101110100", --igit
              "011000000", --128
                           --2nd line
"100100000", "101100011", "101101111", "101101100", -- col
"101101111", "101101110", "100100000", "100111010", --on :
              "000010000", "000001101", --blink
          "000001100", --no blink  --139
              --"100100000",
              "011000000",
"101100100", "101101001", "101101101", "101101101",  --dimm
"101101001", "101101110", "101100111", "100100000", --ing

              "000000001", --149
"100110001", "100110110", "100100000", "101001100", --16 L
"101000101", "101000100", "101110011", "100100000", --EDs
              "011000000", --158
"101001000", "101101001", "101000010", "101110010", --HiBr
"101101001", "101100111", "101101000", "101110100", --ight
              "011000000", --167
"101100100", "101101001", "101101101", "101101101", --dimm
"101101001", "101101110", "101100111",  "100100000", --ing

              "000000001", --176
"101000010", "101110101", "101111010", "101111010", --Buzz
"101100101", "101110010", "100100000", "100100000", --er
              "011000000", --185
"101110100", "101101111", "101101110", "101100101", --tone
"101110011", "100100000", "100100000", "100100000", --s

              "000000001", --194
"101010100", "101010110", "100100000", "100100000", --TV
              "100100000", "100100000", "100100000", "100100000", --
              "011000000", --203
"100111000", "100100000", "101100111", "101110010", --8 gr
"101100001", "101111001", "100100000", "100100000", --ay

              "000000001",  --212
"101001101", "101101111", "101101110", "101101001", --Moni
"101110100", "101101111", "101110010", "100100000", --tor
              "011000000",  --221
"100110100", "100110000", "100111001", "100110110", --4096
```

```
"100100000", "101100011", "101101111", "101101100", -- col


              "000000001", --230
"100111000", "100100000", "101010000", "101110101", --8 Pu
"101110011", "101101000", "100100000", "100100000", --sh
              "011000000",
"100100000", "101000010", "101110101",  "101110100", -- But
"101110100", "101101111", "101101110", "101110011", --tons

              "000000001",  --248
"100111000", "100100000", "101101111", "101101110", --8 on
"100101111", "101101111", "101100110", "101100110", --/off
              "011000000",
"101010011", "101110111", "101101001", "101110100", --Swit
"101100011", "101101000", "101100101", "101110011", --ches

              "000000001",  --266
"101000100", "101001001", "101010000", "100100000", --DIP
"100100000", "100100000", "100100000", "100100000", --
              "011000000",
"100100000", "100100000", "101010011", "101110111", --  Sw
"101101001", "101110100", "101100011", "101101000", --itch

              "000000001",  --284
"101001011", "101100101", "101111001", "100100000", --Key
"101110000", "101100001", "101100100", "100100000", --pad
              "011000000",
"100100000", "100100000", "100110100", "100101010", --  4*
"100110100", "100100000", "100100000", "100100000", --4

          "000000001",  --302
"100110011", "100110111", "100100000", "101010000", --37 P
"101101111", "101110010", "101110100", "101110011", --orts
              "011000000",  --311
"100100000", "100111000", "100101011", "100110001", -- 8+1
"100111000", "100101011", "100110001", "100110001", --8+11

              "000000001",  --320
"101010011", "101100101", "101110010", "101101001", --Seri
"101100001", "101101100", "100100000", "100100000", --al
              "011000000",
              "100100000", "100100000", "100100000", "100100000",
"101010000", "101101111", "101110010", "101110100",-- Port

              "000000001",  --338
"101010000", "101010011", "100101111",  "100110010", --PS/2
              "100100000", "100100000", "100100000", "100100000",
              "011000000",
"100100000", "100100000", "100100000", "101001101", --   M
"101101111", "101110101", "101110011", "101100101", --ouse

              "000000001", --356
"101010000", "101010011", "100101111",  "100110010", --PS/2
              "100100000", "100100000", "100100000", "100100000",
              "011000000",
"101001011", "101100101", "101111001", "101100010", --Keyb
"101101111", "101100001", "101110010", "101100100", --oard

              "000000001",  --374
"101001111", "101110011", "101100011", "101101001", --Osci
"101101100", "101101100", "100101110",  "101110011", --ll.s
              "011000000", --383
"100110001", "100101100", "100110001", "100110000", --1,10
"100101100", "100110101", "100110000",  "101001101", --,50M
              "011000000",  --392
```

```
            "100101011", "100100000", "101010011", "101101111",  --+ So
            "101100011", "101101011", "101100101", "101110100",  --cket

                         "000000001",  --401
            "101010000", "101101100", "101100001", "101110100", --Plat
            "101100110", "101101111", "101110010", "101101101", --form
                         "011000000",
            "100100000", "100100000", "101000110", "101101100", --  Fl
            "101100001", "101110011", "101101000", "100100000", --ash

                         "000000001",  --419
            "101010010", "101101001", "101100111", "101101001", --Rigi
                         "101100100", "100100000", "100100000", "100100000", --d
                         "011000000",
                         "100100000", "100100000", "100100000", "100100000", --
                  "101100011", "101100001", "101110011", "101100101"  --case
                      );

begin

   process (clk_10M, reset)
      variable i, count, count1, c, c1, pwm_c, pwm_counter,
                                 count_7seg, count1_7seg, count3, count4,
                                 count5, count6, count7, count8, count9,
                                 count10, count11, count12, count13,
                                 count14, count15, count16, round : integer;
      variable one, ten, hun, tou : std_logic_vector (3 downto 0);
            variable ud, ud1, ud2, sil : std_logic;
   begin
       if reset = '1' then
      current <= start;
          rs <= '0';
              rw <= '0';
              e1 <= '1';
              bl <= '1';
              count := 0;
              i := 0;
              c1 := 0;
              c:=0;
              pwm_c :=0;
              pwm_counter := 16384;
              count_7seg:= 0;
              count3 := 0;
              count4 := 1000;
              count5 := 0;
              count6 := 0;
              count7 := 0;
              count8 := 0;
              count9 := 1000;
              count10 := 0;
              count11 := 0;
              count13 := 0;
              count14 := 0;
              count15 := 0;
              count16 := 0;
              ud := '0';
              ud1 := '0';
              ud2 := '0';
              sil := '1';
              round := 0;
              buzzer_signal <= '1';
              buzzer_pwm <= '1';
    elsif clk_10M = '1' and clk_10M'event then

----------LCD's back light dimming------------
              if c = 4096 then
```

```
                c := 0;
            else
                c := c + 1;
            end if;
            pwm_c := pwm_c + 1;
            if pwm_c = 20000 then
                if pwm_counter < 16384 and i < 111 and i >108 then
                    pwm_counter := pwm_counter + 1;
                else
                    pwm_counter := 0;
                end if;
                pwm_c := 0;
            end if;

            if c < pwm_counter and i < 111 and i >108 then
                bl <= '0';
            else
                bl <= '1';
            end if;

------------------7seg---------------
            count_7seg:= count_7seg + 1;
            if count_7seg =  1000000 then
                count_7seg := 0;
                one := one + 1;
                if one = 10 then
                    one := "0000";
                    ten := ten + 1;
                    if ten = 10 then
                        ten := "0000";
                        hun := hun + 1;
                        if hun = 10 then
                            hun := "0000";
                            tou := tou + 1;
                            if tou = 10 then
                                tou := "0000";
                            end if;
                        end if;
                    end if;
                end if;
            end if;

-----------4 digit multiplexing------------
            count1_7seg := count1_7seg + 1;
            if count1_7seg < 10000 then
                digit <= one;
                seg <= "1110" or pwm_7seg;
            elsif count1_7seg < 20000 then
                digit <= ten;
                seg <= "1101" or pwm_7seg;
            elsif count1_7seg < 30000 then
                digit <= hun;
                seg <= "1011" or pwm_7seg;
            elsif count1_7seg < 40000 then
                digit <= tou;
                seg <= "0111" or pwm_7seg;
            elsif count1_7seg = 40000 then
                count1_7seg := 0;
            end if;

            if i >121 and i <140 then
                count4 := 511;
            elsif i >139 and i <150 then
               count3:=count3 + 1;
            else
                count4 := 50; -- 7seg's regular intensity
```

```
            end if;
            if count3 = 100000 then
                  if ud = '0' then
                     count4 := count4 + 1;
                        if count4 = 512 then
                        ud := '1';
                        end if;
                  else
                        count4 := count4 - 1;
                        if count4 = 0 then
                        ud := '0';
                        end if;
                  end if;
                  count3:= 0;
            end if;
            count5:= count5 + 1;
            if count5 = 4096 then
                  count5 := 0;
            elsif count5 < count4 then
                  pwm_7seg <= "0000";
            else
               pwm_7seg <= "1111";
            end if;

            if i >128 and i <141 then
                  if count6 = 5000000 then
                        colon <= not colon;
                        count6 := 0;
                  else
                        count6 := count6 + 1;
                  end if;
            else
                  colon <= '0';
            end if;
----------------7seg end-----------------------
----------------leds---------------------------
                        if i > 156 and i < 177 then
                              leds1 <= "1111111111111111";
                              sil := '0';
                              count7 := 0;
                              round := 0;
                        elsif (count7 = 1000000 and
                        (round <8 or (round >11 and round <20))) or
            (count7 = 5000000 and
            ((round >7 and round <12 )or (round >19 and round < 25))) then
                        if round < 8 then
                              leds1 <= sil & leds1 (15 downto 1);
                              if leds1 = "1111111111111111" then
                                    sil := '0';
                                    round := round + 1;
                              elsif leds1 = "0000000000000000" then
                                    sil := '1';
                                    round := round + 1;
                              end if;
                     elsif round < 12 then
                        leds1 <= not leds1;
                              round := round + 1;
                        elsif round < 20 then
                              leds1 <=  leds1 (14 downto 0) & sil;
                              if leds1 = "1111111111111111" then
                                    sil := '0';
                                    round := round + 1;
                              elsif leds1 = "0000000000000000" then
                                    sil := '1';
                                    round := round + 1;
                              end if;
```

```
                elsif round < 24 then
                   leds1 <= not leds1;
                          round := round + 1;
                    else
                          round := 0;
                    end if;
                    count7 := 0;
              else
                    count7 := count7 + 1;
              end if;

        if i >158 and i <168 then
              count9 := 511;
        elsif i >167 and i <177 then
           count8:=count8 + 1;
        else
              count9 := 50;  -- led's regular intensity
        end if;
        if count8 = 100000 then
              if ud1 = '0' then
                 count9 := count9 + 1;
                    if count9 = 512 then
                    ud1 := '1';
                    end if;
              else
                    count9 := count9 - 1;
                    if count9 = 0 then
                    ud1 := '0';
                    end if;
              end if;
              count8:= 0;
        end if;
        count10:= count10 + 1;
        if count10 = 4096 then
              count10 := 0;
        elsif count10 < count9 then
              pwm_led <= '1';
        else
           pwm_led <= '0';
        end if;

----------------leds end--------------------
----------------buzzer----------------------
        if i <176 then
              count12 := 3000;
        elsif i <196 and i > 185 then
              if count11 = 3000000  then
                    if ud2 = '0' then
                       count12 := count12 + 1000;
                          if count12 >= 19000 then
                          ud2 := '1';
                          end if;
                    else
                          count12 := count12 - 1000;
                          if count12 <= 6000 then
                          ud2 := '0';
                          end if;
                    end if;
                    count11:= 0;
              else
                    count11:=count11 + 1;
              end if;
        end if;
        count13:= count13 + 1;
        if (i <186 and i > 177 and count >= 2500000) then
          buzzer_signal <= '0';
```

```
            elsif i <196 and i > 193 then
                  if count13 > count12 then
                        buzzer_signal <= not buzzer_signal;
                        count13 := 0;
                  end if;
            else
              buzzer_signal <= '1';
            end if;

            if count14 < 100 then
                  count14 := count14 + 1;
            else
                  count14 := 0;
            end if;
            if count14 <= 30 and i <196 and i > 185 then
                  buzzer_pwm <= '1';
            else
                  buzzer_pwm <= '0';
            end if;

----------------buzzer end------------------
          case current is
              when start =>
              data <= tablemarq(i)(7 downto 0);
                    rs <= tablemarq(i)(8);
                    if i < datamarq  then
                          current <= write;
                          if (tablemarq(i)(8 downto 0) = "000011000") then
                                count1 := delay_FFFF;
                          elsif i = 100 or i = 127 or i = 138 or i = 157 or
                                       i = 166 or i = 202 or i = 211 or
                          i = 220 or i = 229 or i = 247 or i = 265 or
                          i = 283 or i = 301 or i = 310 or i = 319 or
                          i = 337 or i = 355 or i = 373 or i = 382 or
                          i = 391 or i = 400 or i = 418 or i = 436 then
                                count1 := delay_2s;
                          elsif i = 91 or i = 118 then
                                count1 := delay_0_5s;
                          elsif i = 147 or i = 175 then
                                count1 := delay_3s;
                          elsif  i = 109 or i = 193  then
                                count1 := delay_4_5s;
                          elsif i = 184  then
                                count1 := delay_0_4s;
                          elsif i > 83 then
                                count1 := delay_4000;
                          else
                                count1 := delay_0_2s;
                          end if;
                          i := i + 1;
                    else
                          i := 1;
                    current <= start;
         end if;
          when write =>
                e1 <= not e1;
                count := count1;
                    if e1 = '1' then
                          return_state <= write;
                    else
                          return_state <= start;
                    end if;
                current <= delay;
          when delay =>
                    count := count - 1;
                    if count = 0 then
```

```
                    current <= return_state;
        end if;

         when others =>
         end case;
     end if;
 end process;
     e <= e1;
seven_seg1 <= "0000001" when digit = "0000" else
                         "1001111" when digit = "0001" else
                             "0010010" when digit = "0010" else
                             "0000110" when digit = "0011" else
                             "1001100" when digit = "0100" else
                             "0100100" when digit = "0101" else
                             "0100000" when digit = "0110" else
                             "0001111" when digit = "0111" else
                             "0000000" when digit = "1000" else
                             "0000100" when digit = "1001" else
                             "0001000" when digit = "1010" else
                             "1100000" when digit = "1011" else
                             "0110001" when digit = "1100" else
                             "1000010" when digit = "1101" else
                             "0110000" when digit = "1110" else
                             "0111000" when digit = "1111";
 seven_seg <=   seven_seg1 & colon;
     pwm_led_vec <= (others => pwm_led);
 leds <= leds1 and pwm_led_vec;
     pwm_buz <= buzzer_signal or buzzer_pwm when rxd='1' else
                '0'  when rxd='0' and serial_test_pb = '1';


 process (clk_20M, reset)
    variable count : integer;
 begin
            if reset = '1' then
                   count := 0;
            elsif clk_20M = '1' and clk_20M'event then
                   if count = 20000000 then
                          count := 0;
                          blink_20M <= not blink_20M;
                   else
                          count := count + 1;
                   end if;
            end if;
     end process;

 process (clk_50M, reset)
    variable count : integer;
 begin
            if reset = '1' then
                   count := 0;
            elsif clk_50M = '1' and clk_50M'event then
                   if count = 50000000 then
                          count := 0;
                          blink_50M <= not blink_50M;
                   else
                          count := count + 1;
                   end if;
            end if;
     end process;

     --ports--
     port18 <= (others => blink_50M);
     port8 <= (others => blink_50M);
     port10 <= (others => blink_50M);
     port1 <= blink_20M;
```

```
	--vga & tv--
	process (clk_1M, reset)
		variable count : integer;
	begin
		if reset = '1' then
			count := 0;
			color <= "0000";
			tv <= "00000000";
		elsif (clk_1M'event and clk_1M = '1') then
			count := count + 1;
			if count = 1000000 then
				color <= color + 1;
				blink_1M <= not blink_1M;
				case tv is
				when "00000000" =>
					tv <= "10000000";
				when "10000000" =>
					tv <= "11000000";
				when "11000000" =>
					tv <= "10100000";
				when "10100000" =>
					tv <= "10010000";
				when "10010000" =>
					tv <= "10001000";
				when "10001000" =>
					tv <= "10000100";
				when "10000100" =>
					tv <= "10000010";
				when "10000010" =>
					tv <= "10000001";
				when "10000001" =>
					tv <= "00000000";
				when others =>
					tv <= "00000000";
			    end case;
				count := 0;
			end if;
		end if;
	end process;
	hsync <= blink_1M;
	vsync <= blink_1M;
	blue<= color;
	green<= color;
	red<= color;

	--serial--
	txd <= blink_1M;

	--ps2--
	ps2data1 <=  blink_1M;
	ps2clock1 <= blink_1M;
	ps2data2 <=  blink_1M;
	ps2clock2 <= blink_1M;

end Behavioral;
```

۴۱- دموی ورودی برای بورد آموزشی

در این برنامه، اجزاء ورودی بورد شامل کلیدهای فشاری (Push buttons)، کلیدهای 0/1 (On/Off Switches)، keypad، Dip switch و حالت ورودی پورت‌های کاربر (۳ پورت) تست می‌شوند. برای نمایش نتایج هم از LCD، LED ها و Seven segment و همچنین بازر استفاده شده است.

با همان شروع کار روی LCD اطلاعاتی از نام بورد و وب سایت به صورت روان می‌آید و تا زمانی که تغییری در ورودی‌ها ایجاد نشود این کار ادامه پیدا می‌کند. در صورت ایجاد تغییر هم پس از آنکه تغییر ایجاد شد و جزء مورد وقوع تغییر نشان داده شد، روال دوباره به حالت فوق‌الذکر بر می‌گردد.

در سطر اول LCD جزء مورد تغییر نشان داده می‌شود (در LCD، Push Bt.، 0/1 Swi:، Key pad:، DIP Swi:، Port 8P، Port 10P و Port 18P را می‌بینید). در سطر دوم هم یک عدد باینری ۸ بیتی نشان داده می‌شود. در مورد کلیدهای فشاری، کلیدهای 0/1 و Dip switch، هر بیت به یک کلید مرتبط است. در مورد Keypad هم چهار بیت سمت راست، شماره‌ی کلید فشرده شده را نشان می‌دهند و چهار بیت سمت چپ همیشه صفرند. در مورد پورت‌ها هم برای پورت ۸ پینی هر بیت مرتبط با یک پین می‌باشد. در مورد پورت ۱۰ پینی بیت‌ها متناظر ۸ بیت پر ارزش‌تر پورت و در مورد پورت ۱۸ پینی هم متناظر ۸ بیت کم ارزش‌تر پورت می‌باشد.

از LED های روی بورد هم به موازات سطر دوم LCD برای نمایش تغییرات استفاده می‌شود. در مورد کلیدهای فشاری، کلیدهای 0/1، Dip switch، Keypad و پورت ۸ پینی، هشت LED سمت راست بورد به همان صورتی که در LCD نمایش داده می‌شود، تغییرات را نشان می‌دهند. برای پورت ۱۰ پینی، از ده LED سمت راست بورد برای نمایش تغییرات ۱۰ پین بورد استفاده شده و برای پورت ۱۸ پینی، از ده LED سمت راست بورد برای نمایش تغییرات ۱۰ پین پر ارزش‌تر بورد استفاده شده (۸ پین کم‌ارزش آن در LCD نمایش داده می‌شود).

با هر تغییر، یک لحظه بازر به صدا در می‌آید (صدای Beep) و همچنین به عدد نشان داده شده در Seven segment که در ابتدا 0000 بوده، یک واحد اضافه می‌شود. توجه داشته باشید که با فشردن هر کلید فشاری، دو تغییر حساب می‌شود. یکی برای فشار دادن آن و دیگری برای رها سازی آن. در نتیجه اگر می‌خواهید تغییرات به سرعت از دید شما خارج نشود، کلید را سریع فشار ندهید.

پایه‌ی ۸۸ (پورت متصل به پین گرد)، برای ریست می‌باشد (چون کلیدهای فشاری برای تست خودشان استفاده می‌شوند) و لزومی ندارد که در ابتدای کار حتماً فعال ('1') شود. برای اینکه نویزها روی آن اثر نگذارند، آن را به صورت داخلی Pulldown کرده‌ایم (در فایل UCF این کار را انجام داده‌ایم). برای پالس ساعت هم از اسیلاتور داخل بوردی 10^{MHz} استفاده شده است.

برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity demo_in is
   Port (
                    --lcd--
                    data : out std_logic_vector(7 downto 0);
                    rs : out std_logic;
                    rw : out std_logic;
                    e : out std_logic;
                    bl : out std_logic;

                    --general--
                    clk_10M : in std_logic;
                    reset : in std_logic;

                    --seven segment--
                    seg: out std_logic_vector (3 downto 0);
                    seven_seg : out std_logic_vector (7 downto 0);

                    --leds--
                    leds : out std_logic_vector (15 downto 0);

                    --buzzer--
                    buzzer : out std_logic;

                    --ports--
```

```
                    port18: in std_logic_vector(17 downto 0);
                    port8: in std_logic_vector(7 downto 0);
                    port10: in std_logic_vector(9 downto 0);

                    --push buttons--
                    pb : in std_logic_vector(7 downto 0);

                    --0/1 keys--
                    keys : in std_logic_vector(7 downto 0);

                    --DIP switch--
                    dip : in std_logic_vector(7 downto 0);

                    --keypad--
            row : out std_logic_vector(3 downto 0);
            column : in std_logic_vector(3 downto 0)

);

end demo_in;

architecture Behavioral of demo_in is
   constant delay_3FFF : integer := 16384;
   constant delay_0_2s : integer := 2000000;
   constant delay_5s : integer := 50000000;
   signal e1 : std_logic;
   signal pb_old, keys_old,
                    dip_old, port8_old : std_logic_vector (7 downto 0);
   signal port10_old : std_logic_vector (9 downto 0);
   signal port18_old : std_logic_vector (17 downto 0);
   constant datamarq : integer := 83;    --number of data to be showed in LCD
      type state is (start, marq, write, delay, changed);
   signal current, return_state : state;
   constant debounce_time : integer := 200000;
   signal row_i, column_encoded, row_encoded,
                    column_before, digit: std_logic_vector (3 downto 0);
   signal key_pressed : std_logic;
      type state1 is (start, waiting, prob_pressed, delay);
      signal current1 : state1;
   signal beep_en : std_logic;
   signal digit1: std_logic_vector (3 downto 0);
   signal seven_seg1: std_logic_vector (6 downto 0);

   type arraymarq is array (0 to datamarq - 1)
                           of std_logic_vector (8 downto 0);
   type array_inputs is array (0 to 69)
                           of std_logic_vector (8 downto 0);
   constant tablemarq : arraymarq := (
               "000111000", "000001100", "000000001",      "010000111",
               "101000001", "000011000", "101101101", "000011000", --Am
                           "101001111", "000011000", "101001100",  --OL
               "000011000", "100100000", "000011000", "101100101", -- e
            "000011000", "101101100", "000011000", "101100101",  --le
            "000011000", "101100011", "000011000", "101110100", --ct
            "000011000", "101110010", "000011000", "101101111", --ro
              "000011000", "101101110", "000011000", "101101001", --ni
            "000011000", "101100011", "000011000", "101110011", --cs
            "000011000", "100100000", "000011000", "100100000", -- 2 spaces
            "000011000", "101110111", "000011000", "101110111", --ww
            "000011000", "101110111", "000011000", "100101110", --w.
              "000011000", "101100001", "000011000", "101101101", --am
            "000011000", "101101111", "000011000", "101101100",  --ol
            "000011000", "100101101", "000011000", "101100101",  -- -e
            "000011000", "100101110", "000011000", "101100011", --.c
            "000011000", "101101111", "000011000", "101101101", --om
            "000011000", "100100000", "000011000", "100100000", -- 2 spaces
```

```
            "000011000", "100100000", "000011000", "100100000", -- 2 spaces
            "000011000", "100100000", "000011000", "100100000", -- 2 spaces
            "000011000", "100100000", "000011000", "100100000"); -- 2 spaces

   constant table_inputs : array_inputs := (
               "000000001",
               "101010000", "101110101", "101110011","101101000",  --Push
               "100100000", "101000010", "101110100","100111010",  -- Bt:
                           "011000000",
               "000000001",
               "100110000", "100101111", "100110001","100100000",  --0/1
               "101010011", "101110111", "101101001","100111010",  --Swi:
                           "011000000",
               "000000001",
               "101000100", "101001001", "101010000", "100100000",  --DIP
               "101010011", "101110111", "101101001", "100111010",  --Swi:
                           "011000000",
               "000000001",
               "101010000", "101101111", "101110010", "101110100",  --Port
               "100100000", "100100000", "100111000", "101010000",  --  8P
                           "011000000",
               "000000001",
               "101010000", "101101111", "101110010", "101110100",  --Port
               "100100000", "100110001", "100110000", "101010000",  -- 10P
                           "011000000",
               "000000001",
               "101010000", "101101111", "101110010", "101110100",  --Port
               "100100000", "100110001", "100111000", "101010000",  -- 18P
                           "011000000",
               "000000001",
            "101001011", "101100101", "101111001", "100100000", --Key
            "101110000", "101100001", "101100100", "100111010", --pad:
                           "011000000"

            );

begin
   process (clk_10M, reset)
      variable i, j, pointer, count, count1 : integer;
            variable anychanged : std_logic_vector (3 downto 0);
            variable changed_value : std_logic_vector (7 downto 0);
            variable first_pb, first_keys, first_dip,
                  first_port8, first_port10, first_port18,
                  first_keypad, first : std_logic;
   begin
            if reset = '1' then
            current <= start;
               rs <= '0';
                  rw <= '0';
                  e1 <= '1';
                  bl <= '1';
                  count := 0;
                  i := 0;
                  anychanged := "0000";
                  first := '0';
         elsif clk_10M = '1' and clk_10M'event then
               case current is
                  when start =>
                        if anychanged = "0001" then
                              if first_pb = '1' then
                                    j := 0;
                                    pointer := 0;
                                    first_pb := '0';
                                    changed_value := pb_old;
                                    leds <= "00000000" & pb_old(0)&
pb_old(1)& pb_old(2)&
```

```
                                        pb_old(3)& pb_old(4)& pb_old(5)&
pb_old(6)& pb_old(7);
                                end if;
                                current <= changed;
                        elsif anychanged = "0010" then
                                if first_keys = '1' then
                                        j := 0;
                                        pointer := 10;
                                        first_keys := '0';
                                        changed_value := keys_old;
                                        leds <= "00000000" & keys_old(0)&
keys_old(1)& keys_old(2)&
                                        keys_old(3)& keys_old(4)& keys_old(5)&
                                        keys_old(6)& keys_old(7);
                                end if;
                                current <= changed;
                        elsif anychanged = "0011" then
                                if first_dip = '1' then
                                        j := 0;
                                        pointer := 20;
                                        first_dip := '0';
                                        changed_value := dip_old;
                                        leds <= "00000000" & dip_old(0)&
dip_old(1)& dip_old(2)&
                                        dip_old(3)& dip_old(4)& dip_old(5)&
dip_old(6)& dip_old(7);
                                end if;
                                current <= changed;
                        elsif anychanged = "0100" then
                                if first_port8 = '1' then
                                        j := 0;
                                        pointer := 30;
                                        first_port8 := '0';
                                        changed_value := port8_old;
                                        leds <= "00000000" & port8_old(0)&
port8_old(1)&
                                         port8_old(2)& port8_old(3)&
port8_old(4)&
                                         port8_old(5)& port8_old(6)&
port8_old(7);
                                end if;
                                current <= changed;
                        elsif anychanged = "0101" then
                                if first_port10 = '1' then
                                        j := 0;
                                        pointer := 40;
                                        first_port10 := '0';
                                        changed_value := port10_old(7 downto 0);
                                        leds <= "000000" & port10_old(0)&
port10_old(1)&
                                        port10_old(2)& port10_old(3)&
port10_old(4)&
                                        port10_old(5)& port10_old(6)&
                                          port10_old(7)& port10_old(8)&
port10_old(9);
                                end if;
                                current <= changed;
                        elsif anychanged = "0110" then
                                if first_port18 = '1' then
                                        j := 0;
                                        pointer := 50;
                                        first_port18 := '0';
                                        changed_value := port18_old(7 downto 0);
                                        leds <= "000000" & port18_old(17 downto
8);
                                end if;
```

```
                        current <= changed;
                    elsif anychanged = "0111" then
                        if first_keypad = '1' then
                            j := 0;
                            pointer := 60;
                            first_keypad := '0';
                            changed_value := digit(0)& digit(1)&

digit(2)& digit(3)& "0000";
                            leds <= "000000000000" & digit;
                        end if;
                        current <= changed;
                    else
                        current <= marq;
                    end if;
                when marq =>
                    leds <= "0000000000000000";
                data <= tablemarq(i)(7 downto 0);
                    rs <= tablemarq(i)(8);
                    if i < datamarq  then
                        current <= write;
                        if (tablemarq(i)(8 downto 0) = "000011000") then
                            count1 := delay_3FFF;
                        else
                            count1 := delay_0_2s;
                        end if;
                        i := i + 1;
                    else
                        i := 1;
                    current <= start;
            end if;
              when changed=>
                data <= table_inputs(j+pointer)(7 downto 0);
                    rs <= table_inputs(j+pointer)(8);
                    if j <= 9  then
                        count1 := delay_3FFF;
                        current <= write;
                        j := j + 1;
                    elsif j <=17 then
                        if changed_value(j-10) = '1' then
                           data <= "00110001";
                        else
                           data <= "00110000";
                        end if;
                        rs <= '1';
                        count1 := delay_3FFF;
                        current <= write;
                        j := j + 1;
                    else
                        anychanged := "0000";
                        return_state <= start;
                        count := delay_5s;
                        current <= delay;
                        i := 1;
                    end if;
            when write =>
                   e1 <= not e1;
                   count := count1;
                    if e1 = '1' then
                        return_state <= write;
                    else
                        return_state <= start;
                    end if;
                   current <= delay;
            when delay =>
                    count := count - 1;
```

```
                        if count = 0 or (anychanged /= "0000" and count >
20000) then
                        current <= return_state;
                end if;
                  when others =>
                 end case;

                    if pb_old /= pb then
                           pb_old <= pb;
                           anychanged := "0001";
                           first_pb := '1';
                    end if;

                    if keys_old /= keys then
                           keys_old <= keys;
                           anychanged := "0010";
                           first_keys := '1';
                    end if;

                    if dip_old /= dip then
                           dip_old <= dip;
                           anychanged := "0011";
                           first_dip := '1';
                    end if;

                    if port8_old /= port8 then
                           port8_old <= port8;
                           anychanged := "0100";
                           first_port8 := '1';
                    end if;

                    if port10_old /= port10 then
                           port10_old <= port10;
                           anychanged := "0101";
                           first_port10 := '1';
                    end if;

                    if port18_old /= port18 then
                           port18_old <= port18;
                           anychanged := "0110";
                           first_port18 := '1';
                    end if;

                    if key_pressed = '1' then
                           anychanged := "0111";
                           first_keypad := '1';
                    end if;

                    if first = '0' then
                           anychanged := "0000";
                           first := '1';
                    end if;
                    if anychanged /= "0000" then
                           beep_en <= '1';
                    else
                           beep_en <= '0';
                    end if;
          end if;
   end process;
       e <= e1;

---------keypad--------------
       process (clk_10M, reset)
         variable count, count1: integer;
       begin
       if reset = '1' then
```

```
            current1 <= start;
         key_pressed <= '0';
      elsif clk_10M = '1' and clk_10M'event then
            case current1 is
            when start =>
                  row_i <= "0000";
                  key_pressed <= '0';
                  count := 0;
                  if column = "1111" then
                        current1 <= waiting;
                        row_i <= "1110";
               end if;
            when waiting =>
                  count := count + 1;
                  if count = 1000 then
                        row_i <= row_i(2 downto 0) & row_i(3);
                        count := 0;
                  end if;
                  if column /= "1111" then
                        current1 <= prob_pressed;
                        count1 := debounce_time;
                        column_before <= column;
               end if;
            when prob_pressed =>
                  count1 := count1 - 1;
                  if count1 = 0 then
                        if column_before = column then
                              digit <= row_encoded + column_encoded;
                              key_pressed <= '1';
                              current1 <= delay;
                              count1 := 10000;
                        else
                        current1 <= start;
                        end if;
                  end if;
            when delay =>
                  count1 := count1 -1;
                  if count1 = 0 then
                        current1 <= start;
                  end if;
            when others =>
                  current1 <= start;
        end case;

      end if;
end process;
   column_encoded <= "0000" when column = "1110" else
                     "0100" when column = "1101" else
                     "1000" when column = "1011" else
                     "1100" when column = "0111" else
                     "0000";

   row_encoded <= "0000" when row_i = "1110" else
             "0001" when row_i = "1101" else
                    "0010" when row_i = "1011" else
                    "0011" when row_i = "0111" else
                    "0000";
   row <= row_i;

--seven segment & buzzer --
      process (clk_10M, reset)
        variable count, count1_7seg: integer;
        variable part: std_logic;
      variable one, ten, hun, tou : std_logic_vector (3 downto 0);

      begin
```

```
	if reset = '1' then
		count := 0;
		part :=	'0';
		buzzer <= '1';
		count1_7seg := 0;
		one := "0000";
		ten := "0000";
		hun := "0000";
		tou := "0000";

	elsif clk_10M = '1' and clk_10M'event then
		 if part = '0' then
			if beep_en = '1' then
				part := '1';
			end if;
		else
			one := one + 1;
			if one = 10 then
				one := "0000";
				ten := ten + 1;
				if ten = 10 then
					ten := "0000";
					hun := hun + 1;
					if hun = 10 then
						hun := "0000";
						tou := tou + 1;
						if tou = 10 then
							tou := "0000";
						end if;
					end if;
				end if;
			end if;

			buzzer <= '0';
			if count = 1000000 then
				buzzer <= '1';
				count := 0;
				part := '0';
			else
				count := count + 1;
			end if;
		end if;

----------4 digit multiplexing------------
		count1_7seg := count1_7seg + 1;
		if count1_7seg < 10000 then
			digit1 <= one;
			seg <= "1110" ;
		elsif count1_7seg < 20000 then
			digit1 <= ten;
			seg <= "1101" ;
		elsif count1_7seg < 30000 then
			digit1 <= hun;
			seg <= "1011" ;
		elsif count1_7seg < 40000 then
			digit1 <= tou;
			seg <= "0111";
		elsif count1_7seg = 40000 then
			count1_7seg := 0;
		end if;

	end if;
	end process;

  seven_seg1 <= "0000001" when digit1 = "0000" else
                "1001111" when digit1 = "0001" else
```

```
                              "0010010" when digit1 = "0010" else
                              "0000110" when digit1 = "0011" else
                              "1001100" when digit1 = "0100" else
                              "0100100" when digit1 = "0101" else
                              "0100000" when digit1 = "0110" else
                              "0001111" when digit1 = "0111" else
                              "0000000" when digit1 = "1000" else
                              "0000100" when digit1 = "1001" else
                              "0001000" when digit1 = "1010" else
                              "1100000" when digit1 = "1011" else
                              "0110001" when digit1 = "1100" else
                              "1000010" when digit1 = "1101" else
                              "0110000" when digit1 = "1110" else
                              "0111000" when digit1 = "1111";
   seven_seg <=   seven_seg1 & '1';       -- colon off

end Behavioral;
```

۴۲- مانیتور برای بورد آموزشی

مانیتور را به پورت VGA روی بورد وصل کنید (دقت کنید که بعضی از جک‌های مانیتور کمی بزرگ هستند و ممکن است در حین اتصال به بورد فشار وارد کنند؛ در این موارد کمی با احتیاط عمل کنید). بلافاصله پس از اتصال، در وسط مانیتور عبارت AmOL *electronics* را می‌توانید مشاهده کنید. رنگ نوشته به صورت پیش فرض آبی روی زمینه‌ی قرمز می‌باشد. توسط سه کلید فشاری متصل به پورت-های 8، 10 و 15 می‌توانید به ترتیب شدت رنگ‌های قرمز، سبز و آبی را برای متن و زمینه (که به ترتیب توسط کلیدهای متصله به پورت‌های 35 «متن»و 37 «زمینه» تعیین می‌شود) تغییر دهید. با هر بار فشار هر یک از کلیدها به شدت رنگ مربوطه، یک واحد اضافه می‌شود و با رسیدن به حداکثر (عدد 15) شدت بعدی 0 می‌شود. با ترکیب‌های متفاوت شدت این سه رنگ می‌توانید تا 4096 رنگ متفاوت به وجود آورید. دوازده LED سمت راست روی بورد در هر لحظه بسته به کلیدهای 35 و 37، شدت هر کدام از رنگ‌های قرمز، سبز و آبی را برای متن یا زمینه نشان می‌دهند (هر رنگ، چهار LED). دو LED سمت چپ هم نشان می‌دهند که برای تغییر رنگ، متن انتخاب شده یا زمینه.

کلید فشاری متصل به پایه‌ی 41، برای ریست می‌باشد که لزومی ندارد که در ابتدای کار حتماً فشرده شود. برای پالس ساعت هم از اسیلاتور 50^{MHz} روی بورد استفاده شده است. بازر بورد هم غیر فعال شده تا نویزهای موجود باعث تولید احتمالی صداهای مزاحم نشوند.

برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity monitor is
      port(
                  screen, text, clk_50M, reset: in std_logic;
                  btn_r, btn_g, btn_b: in std_logic;
```

```
                    hsync: out std_logic;
                    vsync: out std_logic;
                    vgaled: out std_logic_vector(15 downto 0);
                    red, green, blue: out std_logic_vector(3 downto 0);
                    buzzer: out std_logic
                );
end monitor;

architecture Behavioral of monitor is

        signal screen_red, screen_green, screen_blue : std_logic_vector(3 downto
0);
        signal text_red, text_green, text_blue : std_logic_vector(3 downto 0);
        constant h: integer := 809;
        constant v: integer := 530;
        type pic_rom is array (0 to 9999) of std_logic;    -- AmOL electronics
        constant pic_matrix : pic_rom :=
('0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','1','1','1
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
      '0','0','0','0','0','0','1','1','1','1','1','1','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
      '0','0','0','0','0','0','1','1','1','1','1','1','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','1','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
      '0','0','0','0','0','0','1','1','0','1','1','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','1','1','1','0','0','0','0','0','0','1','1','1','0','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
      '0','0','0','0','0','0','1','1','1','0','1','1','1','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','1','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
```

```
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','1','1','0','0','0','1','1','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
',0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
',0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','1','1','1','0','0','0','1','1','1','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
',0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
',0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','1','1','0','0','0','0','0','1','1','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
',0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
',0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','0','1','1','1','0','0','0','0','0','1','1','1','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
',0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
',0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','0','1','1','1','1','1','1','1','1','1','1','1','1','1','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
',0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
',0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','1','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
```

','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','1','1','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0'
,'1','1','1','0','0','1','1','1','0','0','0','0','1','1','0','0','0','0','0','0',
'0','1','1','1','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','1','1','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0'
,'1','1','1','0','1','1','1','1','1','0','0','1','1','1','1','1','0','0','0','0',
'0','0','1','1','1','0','0','0','0','0','0','1','1','1','0','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','1','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0'
,'1','1','1','1','1','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0',
'0','0','1','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','1','
1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','1','0','0','0','1','1','1','0','0','0',
'0','0','0','0','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','1','
1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','1','
1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',

```
      '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
      '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
      '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
      '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
      '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
      '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','
1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
      '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
```

'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1'
,'1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1'
,'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','1','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','1'
,'1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','1','1'
,'1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','1','1','0','
0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0
','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','1','1','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','0',
'0','0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','0'
,'0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',

'0','0','0','1','1','0','0','0','1','1','1','0','0','0','0','0','0','1','1','0','
0','0','0','0','0','0','0','0','0','1','1','0','0','0','1','1','1','0','0','0','0
',0','0','0','1','1','0','0','0','0','1','1','1','0','0','0','0','0','1','1','1','0'
,'0','0','0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','1','1','1','1','1',
'0','0','0','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','1','
1','0','0','0','0','0','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0
',0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','1','1'
,'0','0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','1','1','0','0','0','0','0','1','1','0','0','0','0','0','0','1','1','0','0','
0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','1','1','0','0','0','0','0
',0','0','1','1','0','0','0','0','0','1','1','0','0','0','0','0','1','1','0','0','0'
,'0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','1','1','1','1','1','0',
'0','0','1','1','1','0','0','0','0','1','1','1','1','0','0','0','0','0','1','1','1','
1','0','0','0','0','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','1','1
',1','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','1','1'
,'0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','1','1','1','
0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','1','1','0','0','0','0
',0','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0'
,'0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','1','1','0','0','0','0',
'0','1','1','1','0','0','0','0','0','0','0','1','1','1','1','1','1','1','1','1','1','
1','0','0','0','1','1','1','0','0','1','1','0','0','0','0','0','0','0','0','0','1','1
',1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','1','1','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','0','0','0','0','0','1','1','1','0','0','0','0','0','1','1','1','0','0','0','
0','0','0','0','0','0','1','1','1','0','0','0','0','0','1','1','0','0','0','0','0','0
',1','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0'
,'0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','1','1','1','0','0','0','0','0','1','1','1','1','0','
0','0','0','0','0','1','1','1','0','0','0','0','0','1','1','0','0','0','0','0','0','1
',1','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0'
,'0','0','0','0','0','0','0','0','1','1','0','0','1','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','1','
1','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','1
',1','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0'
,'0','0','0','0','0','0','0','0','1','1','0','1','0','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','1','1','
1','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','1

```
','0','0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0'
,'0','0','1','1','0','0','0','1','1','1','0','1','1','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','1','1','1','0','0','
0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1
',1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','1','1','0','0','0'
,'0','0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','1',
'1','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','1','1','
1','1','1','0','0','0','0','0','0','1','1','1','0','0','0','0','1','1','1','1','1
',0','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','0'
,'0','1','1','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','0','0','1','1','0','0','1','1','1','0','0','
0','0','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','1','1','0','0
',1','1','0','0','0','0','0','0','0','0','0','1','1','0','0','1','1','1','0','0'
,'0','0','0','1','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','1','1','
1','1','0','0','0','0','0','0','0','1','1','1','1','0','0','0','1','1','1','1','1
',0','0','0','0','0','0','1','1','1','0','0','1','1','1','0','0','0','0','0','0'
,'1','1','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','0','1','1','0','0','0','1','1','1','0','0','
0','1','1','1','0','1','1','1','0','0','0','0','0','0','0','0','1','1','0','0','0
',1','1','1','0','0','0','0','0','0','0','1','1','0','0','0','1','1','1','0','0'
,'0','0','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','0','1','1','
1','1','0','0','0','0','0','0','0','1','1','1','0','0','1','1','1','0','1','1','1
',0','0','0','0','1','1','0','0','0','1','1','1','0','0','0','0','0','0','0','1'
,'1','0','0','0','0','1','1','1','0','1','1','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','1','1','1','0','0','0','0','1','1','1','1','1','
1','1','1','0','0','0','1','1','1','0','0','0','0','0','1','1','1','0','0','0','0
',1','1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','1','1'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','0','0','0','0','0','0',
'1','1','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','0','1','1','
1','0','0','0','0','0','0','0','0','0','1','1','1','1','1','1','1','0','0','1','1','1
',0','0','0','0','1','1','0','0','0','0','1','1','1','1','0','0','0','0','1','1','1'
,'0','0','0','0','0','1','1','0','0','0','0','1','1','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','1','1','1','1','
1','1','0','0','0','0','0','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0
',0','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','1','1','1','1'
,'1','1','0','0','0','0','1','1','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','1','1','0','0','0','0','0','0','0','0','0','0','1','1','
0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','1','0','0','0','0','1','1
',1','1','1','1','1','0','0','0','0','0','0','1','1','1','1','1','1','1','1','1','0','0'
,'0','0','0','1','1','0','0','0','0','1','1','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1','
0','0','0','0','0','0','0','0','0','1','1','1','1','1','1','0','0','0','0','0','0','0','0
',0','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','1','1','1'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1','
0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0','1
',1','1','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0'
,'0','0','0','1','1','0','0','0','1','1','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
```

```
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0');

begin

       process (clk_50M, reset)
              VARIABLE hcnt: integer RANGE 0 TO h;
              VARIABLE vcnt: integer RANGE 0 TO v;
              VARIABLE dbnc1, x: integer RANGE 0 TO 200000000;

       begin
              if reset='1' then
```

```
                        hcnt := 0;
                        vcnt := 0;
                        screen_red <= "1111";
                        screen_green <= "0000";
                        screen_blue <= "0000";
                        text_red <= "0000";
                        text_green <= "0000";
                        text_blue <= "1111";
                        buzzer <= '1';
                elsif (clk_50M = '1' and clk_50M'event) then
                        if hcnt < 799 then
                                hcnt := hcnt + 1;
                        else
                                hcnt := 0;
                                if vcnt < 524 then
                                        vcnt := vcnt + 1;
                                else
                                        vcnt := 0;
                                end if;
                        end if;
                        if hcnt >= 662 and hcnt < 755 then
                                hsync <= '0';
                        else
                                hsync <= '1';
                        end if;
                        if vcnt >= 491  and vcnt < 493 then
                                vsync <= '0';
                        else
                                vsync <= '1';
                        end if;
                        if (215 < vcnt and vcnt < 266 and 220 < hcnt and hcnt < 421)
then
                                if x < 9999 then
                                        x := x + 1;
                                else
                                        x := 0;
                                end if;
                                if pic_matrix(x) = '0' then
                                        red <= screen_red;
                                        green <= screen_green;
                                        blue <= screen_blue;
                                else
                                        red <= text_red;
                                        green <= text_green;
                                        blue <= text_blue;
                                end if;
                        elsif (480 > vcnt and 640 > hcnt) then
                                red <= screen_red;
                                green <= screen_green;
                                blue <= screen_blue;
                        else
                           red <= "0000";
                               green <= "0000";
                               blue <= "0000";
                        end if;
                        if dbnc1 > 0 then
                                dbnc1 := dbnc1 - 1;
                        else
                                if text = '1' then
                                        if btn_r = '1' then
                                                text_red <= text_red + 1;
                                                dbnc1 := 10000000;
                                        elsif btn_g = '1' then
                                                text_green <= text_green + 1;
                                                dbnc1 := 10000000;
                                        elsif btn_b = '1' then
```

```
                                text_blue <= text_blue + 1;
                                dbnc1 := 10000000;
                        else
                                dbnc1 := 0;
                        end if;
                end if;
                                ----------------------------
                if screen = '1' then
                        if btn_r = '1' then
                                screen_red <= screen_red + 1;
                                dbnc1 := 10000000;
                        elsif btn_g = '1' then
                                screen_green <= screen_green + 1;
                                dbnc1 := 10000000;
                        elsif btn_b='1' then
                                screen_blue <= screen_blue + 1;
                                dbnc1 := 10000000;
                        else
                                dbnc1 := 0;
                        end if;
                end if;
        end if; --dbnc1
        if screen = '1' then
                vgaled(11 downto 8) <= screen_red;
                vgaled(7 downto 4) <= screen_green;
                vgaled(3 downto 0) <= screen_blue;
        elsif text = '1' then
                vgaled(11 downto 8) <= text_red;
                vgaled(7 downto 4) <= text_green;
                vgaled(3 downto 0) <= text_blue;
        end if;
    end if;
    vgaled(13 downto 12)<=     "00";
    vgaled(14)<=  screen;
    vgaled(15)<=   text;
end process;

end Behavioral;
```

۴۳- کی بورد PS/2 برای بورد آموزشی

کی‌بورد PS/2 را به پورت PS/2 شماره‌ی ۲ (بنفش رنگ) روی بورد وصل کنید. با فشردن هر کلید روی آن make code آن روی هشت LED سمت چپ بورد نشان داده می‌شود. البته در حقیقت بایت‌های ارسالی از کی‌بورد به همان صورتی که از طرف آن فرستاده می‌شوند روی LED ها نشان داده می‌شوند. برای اکثر کلیدها سه بایت شامل یک بایت make code و دو بایت break code که خود آن شامل بایت F0H و دوباره همان make code می‌باشد فرستاده می‌شود که در نتیجه در آخر همان make code را می‌بینیم و چون این کار با سرعت بالایی انجام می‌شود می‌توانیم بگوییم که فقط همان make code را برای اکثر کلیدها مشاهده می‌کنیم.

کلید فشاری متصل به پایه‌ی 41، برای ریست می‌باشد که لزومی ندارد که در ابتدای کار حتماً فشرده شود. برای پالس ساعت هم از اسیلاتور 1^{MHz} روی بورد استفاده شده است. بازر بورد هم غیر فعال شده تا نویزهای موجود باعث تولید احتمالی صداهای ریز مزاحم نشوند.

برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ps2keyboard is
  port (reset:  in  std_logic;
        clk_1M:   in  std_logic;                    -- system clk_1M
        ps2_clk: in  std_logic;                     -- PS/2 clk line
        ps2_dta: in  std_logic;                     -- PS/2 data line
               leds:    out std_logic_vector(7 downto 0); -- LED outputs
               buzzer : out std_logic
      );
end ps2keyboard;

architecture Behavioral of ps2keyboard is
      type state_type is (IDLE, START, DATA, PARITY);    -- FSM states
      signal ps2_dv: std_logic;                          -- PS/2 data valid
      signal prv_ps2_clk, act_ps2_clk: std_logic;        -- auxiliary signals
      signal recdata: std_logic_vector(7 downto 0);      -- read data
```

```
	signal shift: std_logic;                          -- enable for shift reg.
	signal n_shift: std_logic;                        -- auxiliary signal
	signal latch: std_logic;                          -- latch read data
	signal n_latch: std_logic;                        -- auxiliary signal
	signal err: std_logic;                            -- parity or stop error
	signal n_err: std_logic;                          -- auxiliary signal
	signal parset: std_logic;                         -- preset for parity
check
	signal n_parset: std_logic;                       -- auxiliary signal
	signal c_state, n_state: state_type;              -- current & next states
	signal cntval: std_logic_vector(2 downto 0);      -- counter of data bits
	signal zero: std_logic;                           -- counter is zero
	signal parbit: std_logic;                         -- odd parity of data

begin
	-- Provides a one-shot pulse after every falling edge of PS/2 clk
	PS_CLK_SYNC: process(clk_1M, reset)
	begin
		if (reset = '1') then
			prv_ps2_clk <= '1';
			act_ps2_clk <= '1';
			buzzer <= '1';
		elsif (clk_1M'event and clk_1M = '1') then
			act_ps2_clk <= ps2_clk;
			prv_ps2_clk <= act_ps2_clk;
		end if;
	end process;

	ps2_dv <= (not act_ps2_clk) and prv_ps2_clk;

	-- Serial input, parallel output shift register
	SIPO: process(clk_1M, reset)
	begin
		if (reset = '1') then
			recdata <= (others => '0');
		elsif (clk_1M'event and clk_1M = '1') then
			if (shift = '1') then
				recdata <= ps2_dta & recdata(7 downto 1);
			end if;
		end if;
	end process;

	-- Counter of data bits
	COUNT8: process(reset, clk_1M)
	begin
		if (reset = '1') then
			cntval <= (others => '0');
		elsif (clk_1M'event and clk_1M = '1') then
			if (shift = '1') then
				cntval <= cntval + 1;
			end if;
		end if;
	end process;

	zero <= not (cntval(0) or cntval(1) or cntval(2));

	-- Parity check of received data
	PARITY_CHECK: process(clk_1M, parset)
	begin
		if (parset = '1') then
			parbit <= '1';
		elsif (clk_1M'event and clk_1M = '1') then
			if (shift = '1' and ps2_dta = '1') then
				parbit <= not parbit;
			end if;
		end if;
```

```
end process;

-- Synchronous process of control state machine
FSM_SYNC: process(clk_1M, reset)
begin
	if (reset = '1') then
		c_state <= IDLE;
		shift <= '0';
		latch <= '0';
		err <= '0';
		parset <= '1';
	elsif (clk_1M'event and clk_1M = '1') then
		c_state <= n_state;
		shift <= n_shift;
		latch <= n_latch;
		err <= n_err;
		parset <= n_parset;
	end if;
end process;

-- Combinatorial process of control state machine
FSM_COMB: process(c_state, ps2_dv, ps2_dta, zero)
begin
	-- default values
	n_shift <= '0';
	n_latch <= '0';
	n_err <= '0';
	n_parset <= '0';
	case c_state is
	when IDLE =>                        -- wait to receive data
		if ((ps2_dv and (not ps2_dta)) = '1') then
			n_state <= START;
			n_parset <= '1';
		else
			n_state <= IDLE;
		end if;
	when START =>               -- receive first data bit
		if (ps2_dv = '0') then
			n_state <= START;
		else
			n_state <= DATA;
			n_shift <= '1';
		end if;
	when DATA =>                -- receive remaining data bits and parity
		if (ps2_dv = '0') then
			n_state <= DATA;
		elsif (zero = '0') then
			n_state <= DATA;
			n_shift <= '1';
		else
			n_state <= PARITY;
			if (parbit /= ps2_dta) then
				n_err <= '1';
			end if;
		end if;
	when PARITY =>      -- receive stop bit
		if (ps2_dv = '0') then
			n_state <= PARITY;
		else
			n_state <= IDLE;
			n_latch <= '1';
			n_err <= not ps2_dta;
		end if;
	end case;
end process;
```

```
	-- Output latch
	LED_OUTPUTS: process(reset, clk_1M)
	begin
		if (reset = '1') then
			leds <= (others => '1');
		elsif (clk_1M'event and clk_1M = '1') then
			if (err = '1') then
				leds <= (others => '1');
			elsif (latch = '1') then
				leds <= recdata;
			end if;
		end if;
	end process;

end Behavioral;
```

۴۴- موس PS/2 برای بورد آموزشی

موس PS/2 را به پورت PS/2 شماره‌ی ۱ (سبز رنگ) روی بورد وصل کنید. با فشردن هر یک از سه کلید روی موس و حرکت موس به یکی از چهار جهت اصلی، یکی از LED های سمت چپ بورد روشن می‌شود. توجه کنید چون حرکات موس دقیق هستند شاید سخت باشد که از LED های مربوط به حرکت، هر دفعه فقط یکی روشن شود، مگر اینکه دقت را بالا ببرید و موس را دقیقاً به همان جهت اصلی حرکت دهید.

LED های سمت راست بورد هم برای نمایش اندازه‌ی حرکت موس به کار می‌رود (عدد نشان داده شده متناسب با سرعت تغییر می‌باشد). این عدد تا تغییر حرکت بعدی باقی می‌ماند.

کلید فشاری متصل به پایه‌ی 41، برای ریست می‌باشد که لزومی ندارد که در ابتدای کار حتماً فشرده شود.

برای پالس ساعت هم از اسیلاتور 10^{MHz} روی بورد استفاده شده است. بازر بورد هم غیر فعال شده تا نویزهای موجود باعث تولید احتمالی صداهای ریز مزاحم نشوند.

برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;


ENTITY ps2mouse IS
       PORT (
                     clk_10M: in std_logic;
                     reset:in std_logic;
                     led :out std_logic_vector(7 downto 0);
            mov_LEFT       : out   std_logic;
            mov_Right      : out   std_logic;
            Mov_Up           : out   std_logic;
            Mov_Down         : out   std_logic;
                     Click_Left      : out   std_logic;
            Click_Middel    : out   std_logic;
            Click_Right     : out   std_logic;
            mPS2_CLK     : inout std_logic;
            mPS2_DATA    : inout std_logic;
                     buzzer : out std_logic
               );
END ps2mouse;
----------------------------------------------
ARCHITECTURE Behavioral OF ps2mouse IS
```

```
component mouse port(CLK        : in    std_logic;
  RESOLUTION : in    std_logic;
  Reset        : in    std_logic;
  SWITCH     : in    std_logic;
  LEFT       : out   std_logic;
  MIDDLE     : out   std_logic;
  --NEW_EVENT  : out   std_logic;
  RIGHT      : out   std_logic;
  XPOS       : out   std_logic_vector (9 downto 0);
  YPOS       : out   std_logic_vector (9 downto 0);
  --ZPOS       : out   std_logic_vector (3 downto 0);
  PS2_CLK    : inout std_logic;
  PS2_DATA   : inout std_logic);
      end component;

    signal X, Y, xpos, yPos : std_logic_vector(9 downto 0);
    signal mRESOLUTION :     std_logic:='1';
 signal mSWITCH     :    std_logic:='1';
    begin
           mouse1:mouse port map(
    CLK=>clk_10M,RESOLUTION=>mRESOLUTION,Reset=>Reset,

    SWITCH=>mSWITCH,LEFT=>Click_Left,MIDDLE=>Click_Middel,

    RIGHT=>Click_Right,XPOS=>X,YPOS=>Y,
                                                        --ZPOS=>mZPOS,

    PS2_CLK=>mPS2_CLK,PS2_DATA=>mPS2_DATA);
    process (clk_10M, Reset)
           variable Delay: integer;
    begin
           if Reset = '1' then
                  Led <= "11111111";
                  delay := 1000000;
                  Xpos <= "1000000000";
                  YPos <= "1000000000";
                  mSWITCH <= '1';
                  mRESOLUTION <= '1';
                  buzzer <= '1';
           else
              if (clk_10M='1' and clk_10M'event)   then
                         mSWITCH <= '0';
                         led <= x(9 downto 2);
                         if Delay > 0 then
                                delay := delay - 1;
                         else
                                delay := 1000000;
                                if XPos < X then
                                       Mov_Right <= '1';
                                else
                                       Mov_Right <= '0';
                                end if;
                                if XPos > X then
                                       Mov_Left <= '1';
                                else
                                       Mov_Left <= '0';
                                end if;
                                if YPos > Y then
                                       Mov_Up <= '1';
                                else
                                       Mov_Up <= '0';
                                end if;
                                if YPos < Y then
                                       Mov_Down <= '1';
                                else
```

```
                                        Mov_Down <= '0';
                                    end if;
                                    XPOs <= X;
                                    YPOs <= Y;
                                end if;
                            end if;
                        end if;
                end process;
end Behavioral;

                    ----------------------------------------------------------
----------------------
-- Company: Digilent RO
-- Engineer: Mircea Dabacan
--
-- Create Date:    12:57:12 03/01/2008
-- Design Name:
-- Module Name:    MouseRefComp - Structural
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description: This is the structural VHDL code of the
--              Digilent Mouse Reference Component.
--              It instantiates three components:
--                - ps2interface
--                - mouse_controller
--                - resolution_mouse_informer
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity mouse is
   port ( CLK        : in    std_logic;
          RESOLUTION : in    std_logic;
          Reset        : in    std_logic;
          SWITCH     : in    std_logic;
          LEFT       : out   std_logic;
          MIDDLE     : out   std_logic;
          NEW_EVENT  : out   std_logic;
          RIGHT      : out   std_logic;
          XPOS       : out   std_logic_vector (9 downto 0);
          YPOS       : out   std_logic_vector (9 downto 0);
          ZPOS       : out   std_logic_vector (3 downto 0);
          PS2_CLK    : inout std_logic;
          PS2_DATA   : inout std_logic);
end mouse;

architecture Structural of mouse is

   signal TX_DATA    : std_logic_vector (7 downto 0);
   signal bitSetMaxX    : std_logic;
   signal vecValue    : std_logic_vector (9 downto 0);
   signal bitRead   : std_logic;
   signal bitWrite   : std_logic;
   signal bitErr   : std_logic;
```

```
    signal bitSetX   : std_logic;
    signal bitSetY   : std_logic;
    signal bitSetMaxY   : std_logic;
    signal vecRxData   : std_logic_vector (7 downto 0);

    component mouse_controller
       port ( clk       : in    std_logic;
              rst       : in    std_logic;
              read      : in    std_logic;
              write     : out   std_logic;
              err       : in    std_logic;
              setx      : in    std_logic;
              sety      : in    std_logic;
              setmax_x  : in    std_logic;
              setmax_y  : in    std_logic;
              value     : in    std_logic_vector (9 downto 0);
              rx_data   : in    std_logic_vector (7 downto 0);
              tx_data   : out   std_logic_vector (7 downto 0);
              left      : out   std_logic;
              middle    : out   std_logic;
              right     : out   std_logic;
              xpos      : out   std_logic_vector (9 downto 0);
              ypos      : out   std_logic_vector (9 downto 0);
              zpos      : out   std_logic_vector (3 downto 0);
              new_event : out   std_logic);
    end component;

    component resolution_mouse_informer
       port ( clk        : in    std_logic;
              rst        : in    std_logic;
              resolution : in    std_logic;
              switch     : in    std_logic;
              setx       : out   std_logic;
              sety       : out   std_logic;
              setmax_x   : out   std_logic;
              setmax_y   : out   std_logic;
              value      : out   std_logic_vector (9 downto 0));
    end component;

    component ps2interface
       port ( clk      : in    std_logic;
              rst      : in    std_logic;
              read     : out   std_logic;
              write    : in    std_logic;
              rx_data  : out   std_logic_vector (7 downto 0);
              tx_data  : in    std_logic_vector (7 downto 0);
              busy     : out   std_logic;
              err      : out   std_logic;
              ps2_clk  : inout std_logic;
              ps2_data : inout std_logic);
    end component;

begin

    MouseCtrlInst : mouse_controller
       port map (clk=>CLK,
                 rst=>Reset,
                 read=>bitRead,
                 write=>bitWrite,
                 err=>bitErr,
                 setmax_x=>bitSetMaxX,
                 setmax_y=>bitSetMaxY,
                 setx=>bitSetX,
                 sety=>bitSetY,
                 value(9 downto 0)=>vecValue(9 downto 0),
                 rx_data(7 downto 0)=>vecRxData(7 downto 0),
```

```
                tx_data(7 downto 0)=>TX_DATA(7 downto 0),
                left=>LEFT,
                middle=>MIDDLE,
                right=>RIGHT,
                xpos(9 downto 0)=>XPOS(9 downto 0),
                ypos(9 downto 0)=>YPOS(9 downto 0),
                zpos(3 downto 0)=>ZPOS(3 downto 0),
                new_event=>NEW_EVENT);

   ResMouseInfInst : resolution_mouse_informer
      port map (clk=>CLK,
                resolution=>RESOLUTION,
                rst=>Reset,
                switch=>SWITCH,
                setmax_x=>bitSetMaxX,
                setmax_y=>bitSetMaxY,
                setx=>bitSetX,
                sety=>bitSetY,
                value(9 downto 0)=>vecValue(9 downto 0));

   Pss2Inst : ps2interface
      port map (clk=>CLK,
                rst=>Reset,
                tx_data(7 downto 0)=>TX_DATA(7 downto 0),
                read=>bitRead,
                write=>bitWrite,
                busy=>open,
                err=>bitErr,
                rx_data(7 downto 0)=>vecRxData(7 downto 0),
                ps2_clk=>PS2_CLK,
                ps2_data=>PS2_DATA);

end Structural;


----------------------------------------------------------------------
-- mouse_controller.vhd
----------------------------------------------------------------------
-- Author : Ulrich Zolt?n
--          Copyright 2006 Digilent, Inc.
----------------------------------------------------------------------
-- Software version : Xilinx ISE 7.1.04i
--                    WebPack
-- Device             : 3s200ft256-4
----------------------------------------------------------------------
-- This file contains a controller for a ps/2 compatible mouse device.
-- This controller is a client for the ps2interface module.
----------------------------------------------------------------------
--  Behavioral description
----------------------------------------------------------------------
-- Please read the following article on the web for understanding how
-- to interface a ps/2 mouse:
-- http://www.computer-engineering.org/ps2mouse/

-- This controller is implemented as described in the above article.
-- The mouse controller receives bytes from the ps2interface which, in
-- turn, receives them from the mouse device. Data is received on the
-- rx_data input port, and is validated by the read signal. read is
-- active for one clock period when new byte available on rx_data. Data
-- is sent to the ps2interface on the tx_data output port and validated
-- by the write output signal. 'write' should be active for one clock
-- period when tx_data contains the command or data to be sent to the
-- mouse. ps2interface wraps the byte in a 11 bits packet that is sent
-- through the ps/2 port using the ps/2 protocol. Similarly, when the
-- mouse sends data, the ps2interface receives 11 bits for every byte,
-- extracts the byte from the ps/2 frame, puts it on rx_data and
```

```
-- activates read for one clock period. If an error occurs when sending
-- or receiving a frame from the mouse, the err input goes high for one
-- clock period. When this occurs, the controller enters reset state.

-- When in reset state, the controller resets the mouse and begins an
-- initialization procedure that consists of tring to put mouse in
-- scroll mode (enables wheel if the mouse has one), setting the
-- resolution of the mouse, the sample rate and finally enables
-- reporting. Implicitly the mouse, after a reset or imediately after a
-- reset, does not send data packets on its own. When reset(or power-up)
-- the mouse enters reset state, where it performs a test, called the
-- bat test (basic assurance test), when this test is done, it sends
-- the result: AAh for test ok, FCh for error. After this it sends its
-- ID which is 00h. When this is done, the mouse enters stream mode,
-- but with reporting disabled (movement data packets are not sent).
-- To enable reporting the enable data reporting command (F4h) must be
-- sent to the mouse. After this command is sent, the mouse will send
-- movement data packets when the mouse is moved or the status of the
-- button changes.

-- After sending a command or a byte following a command, the mouse
-- must respond with ack (FAh). For managing the intialization
-- procedure and receiving the movement data packets, a FSM is used.
-- When the fpga is powered up or the logic is reset using the global
-- reset, the FSM enters reset state. From this state, the FSM will
-- transition to a series of states used to initialize the mouse. When
-- initialization is complete, the FSM remains in state read_byte_1,
-- waiting for a movement data packet to be sent. This is the idle
-- state if the FSM. When a byte is received in this state, this is
-- the first byte of the 3 bytes sent in a movement data packet (4 bytes
-- if mouse in scrolling mode). After reading the last byte from the
-- packet, the FSM enters mark_new_event state and sets new_event high.
-- After that FSM enterss read_byte_1 state, resets new_event and waits
-- for a new packet.
-- After a packet is received, new_event is set high for one clock
-- period to "inform" the clients of this controller a new packet was
-- received and processed.

-- During the initialization procedure, the controller tries to put the
-- mouse in scroll mode (activates wheel, if mouse has one). This is
-- done by successively setting the sample rate to 200, then to 100, and
-- lastly to 80. After this is done, the mouse ID is requested by
-- sending get device ID command (F2h). If the received ID is 00h than
-- the mouse does not have a wheel. If the received ID is 03h than the
-- mouse is in scroll mode, and when sending movement data packets
-- (after enabling data reporting) it will include z movement data.
-- If the mouse is in normal, non-scroll mode, the movement data packet
-- consists of 3 successive bytes. This is their format:
--
--
--
-- bits      7     6     5     4     3     2     1     0
--         ---------------------------------------------------
-- byte 1 | YOVF| XOVF|YSIGN|XSIGN|  1  | MBTN| RBTN| LBTN|
--         ---------------------------------------------------
--         ---------------------------------------------------
-- byte 2 |                  X MOVEMENT                       |
--         ---------------------------------------------------
--         ---------------------------------------------------
-- byte 3 |                  Y MOVEMENT                       |
--         ---------------------------------------------------
-- OVF = overflow
-- BTN = button
-- M = middle
-- R = right
-- L = left
```

```
--
-- When scroll mode is enabled, the mouse send 4 byte movement packets.
-- bits      7     6     5     4     3     2     1     0
--         -------------------------------------------------
-- byte 1 | YOVF| XOVF|YSIGN|XSIGN|  1  | MBTN| RBTN| LBTN|
--         -------------------------------------------------
--         -------------------------------------------------
-- byte 2 |                  X MOVEMENT                     |
--         -------------------------------------------------
--         -------------------------------------------------
-- byte 3 |                  Y MOVEMENT                     |
--         -------------------------------------------------
--         -------------------------------------------------
-- byte 4 |                  Z MOVEMENT                     |
--         -------------------------------------------------
-- x and y movement counters are represented on 8 bits, 2's complement
-- encoding. The first bit (sign bit) of the counters are the xsign and
-- ysign bit from the first packet, the rest of the bits are the second
-- byte for the x movement and the third byte for y movement. For the
-- z movement the range is -8 -> +7 and only the 4 least significant
-- bits from z movement are valid, the rest are sign extensions.
-- The x and y movements are in range: -256 -> +255

-- The mouse uses as axes origin the lower-left corner. For the purpose
-- of displaying a mouse cursor on the screen, the controller inverts
-- the y axis to move the axes origin in the upper-left corner. This
-- is done by negating the y movement value (following the 2s complement
-- encoding). The movement data received from the mouse are delta
-- movements, the data represents the movement of the mouse relative
-- to the last position. The controller keeps track of the position of
-- the mouse relative to the upper-left corner. This is done by keeping
-- the mouse position in two registers x_pos and y_pos and adding the
-- delta movements to their value. The addition uses saturation. That
-- means the value of the mouse position will not exceed certain bounds
-- and will not rollover the a margin. For example, if the mouse is at
-- the left margin and is moved left, the x position remains at the left
-- margin(0). The lower bound is always 0 for both x and y movement.
-- The upper margin can be set using input pins: value, setmax_x,
-- setmax_y. To set the upper bound of the x movement counter, the new
-- value is placed on the value input pins and setmax_x is activated
-- for at least one clock period. Similarly for y movement counter, but
-- setmax_y is activated instead. Notice that value has 10 bits, and so
-- the maximum value for a bound is 1023.

-- The position of the mouse (x_pos and y_pos) can be set at any time,
-- by placing the x or y position on the value input pins and activating
-- the setx, or sety respectively, for at least one clock period. This
-- is useful for setting an original position of the mouse different
-- from (0,0).
----------------------------------------------------------------------------
--  Port definitions
----------------------------------------------------------------------------
-- clk            - global clock signal (100MHz)
-- rst            - global reset signal
-- read           - input pin, from ps2interface
--                - active one clock period when new data received
--                - and available on rx_data
-- err            - input pin, from ps2interface
--                - active one clock period when error occurred when
--                - receiving or sending data.
-- rx_data        - input pin, 8 bits, from ps2interface
--                - the byte received from the mouse.
-- xpos           - output pin, 10 bits
--                - the x position of the mouse relative to the upper
--                - left corner
-- ypos           - output pin, 10 bits
```

```
--                - the y position of the mouse relative to the upper
--                - left corner
-- zpos           - output pin, 4 bits
--                - last delta movement on z axis
-- left           - output pin, high if the left mouse button is pressed
-- middle         - output pin, high if the middle mouse button is
--                - pressed
-- right          - output pin, high if the right mouse button is
--                - pressed
-- new_event      - output pin, active one clock period after receiving
--                - and processing one movement data packet.
-- tx_data        - output pin, 8 bits, to ps2interface
--                - byte to be sent to the mouse
-- write          - output pin, to ps2interface
--                - active one clock period when sending a byte to the
--                - ps2interface.
----------------------------------------------------------------------------
-- Revision History:
-- 09/18/2006(UlrichZ): created
----------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- simulation library
library UNISIM;
use UNISIM.VComponents.all;

-- the mouse_controller entity declaration
-- read above for behavioral description and port definitions.
entity mouse_controller is
port(
   clk         : in std_logic;
   rst         : in std_logic;
   read        : in std_logic;
   err         : in std_logic;
   rx_data     : in std_logic_vector(7 downto 0);

   xpos        : out std_logic_vector(9 downto 0);
   ypos        : out std_logic_vector(9 downto 0);
   zpos        : out std_logic_vector(3 downto 0);
   left        : out std_logic;
   middle      : out std_logic;
   right       : out std_logic;
   new_event   : out std_logic;
   tx_data     : out std_logic_vector(7 downto 0);
   write       : out std_logic;

   value       : in std_logic_vector(9 downto 0);
   setx        : in std_logic;
   sety        : in std_logic;
   setmax_x    : in std_logic;
   setmax_y    : in std_logic
);
end mouse_controller;

architecture Behavioral of mouse_controller is

----------------------------------------------------------------------------
-- CONSTANTS
----------------------------------------------------------------------------

-- constants defining commands to send or received from the mouse
constant FA: std_logic_vector(7 downto 0) := "11111010"; -- 0xFA(ACK)
```

```
constant FF: std_logic_vector(7 downto 0) := "11111111"; -- 0xFF(RESET)
constant AA: std_logic_vector(7 downto 0) := "10101010"; -- 0xAA(BAT_OK)
constant OO: std_logic_vector(7 downto 0) := "00000000"; -- 0x00(ID)
                        -- (atention: name is 2 letters O not zero)

-- command to read id
constant READ_ID          : std_logic_vector(7 downto 0) := x"F2";
-- command to enable mouse reporting
-- after this command is sent, the mouse begins sending data packets
constant ENABLE_REPORTING : std_logic_vector(7 downto 0) := x"F4";
-- command to set the mouse resolution
constant SET_RESOLUTION   : std_logic_vector(7 downto 0) := x"E8";
-- the value of the resolution to send after sending SET_RESOLUTION
constant RESOLUTION       : std_logic_vector(7 downto 0) := x"03";
                                                    -- (8 counts/mm)
-- command to set the mouse sample rate
constant SET_SAMPLE_RATE  : std_logic_vector(7 downto 0) := x"F3";

-- the value of the sample rate to send after sending SET_SAMPLE_RATE
constant SAMPLE_RATE      : std_logic_vector(7 downto 0) := x"28";
                                                    -- (40 samples/s)

-- default maximum value for the horizontal mouse position
constant DEFAULT_MAX_X : std_logic_vector(9 downto 0) := "1111111111";
                                                      -- 639
-- default maximum value for the vertical mouse position
constant DEFAULT_MAX_Y : std_logic_vector(9 downto 0) := "0111111111";
                                                      -- 479

--------------------------------------------------------------------------
-- SIGNALS
--------------------------------------------------------------------------

-- after doing the enable scroll mouse procedure, if the ID returned by
-- the mouse is 03 (scroll mouse enabled) then this register will be set
-- If '1' then the mouse is in scroll mode, else mouse is in simple
-- mouse mode.
signal haswheel: std_logic := '0';

-- horizontal and veritcal mouse position
-- origin of axes is upper-left corner
-- the origin of axes the mouse uses is the lower-left corner
-- The y-axis is inverted, by making negative the y movement received
-- from the mouse (if it was positive it becomes negative
-- and vice versa)
signal x_pos,y_pos: std_logic_vector(10 downto 0) := (others => '0');

-- active when an overflow occurred on the x and y axis
-- bits 6 and 7 from the first byte received from the mouse
signal x_overflow,y_overflow: std_logic := '0';

-- active when the x,y movement is negative
-- bits 4 and 5 from the first byte received from the mouse
signal x_sign,y_sign: std_logic := '0';

-- 2's complement value for incrementing the x_pos,y_pos
-- y_inc is the negated value from the mouse in the third byte
signal x_inc,y_inc: std_logic_vector(7 downto 0) := (others => '0');

-- active for one clock period, indicates new delta movement received
-- on x,y axis
signal x_new,y_new: std_logic := '0';

-- maximum value for x and y position registers(x_pos,y_pos)
signal x_max,y_max: std_logic_vector(9 downto 0) := (others => '0');
```

```
-- active when left,middle,right mouse button is down
signal left_down,middle_down,right_down: std_logic := '0';

-- the FSM states
-- states that begin with "reset" are part of the reset procedure.
-- states that end in "_wait_ack" are states in which ack is waited
-- as response to sending a byte to the mouse.
-- read behavioral description above for details.
type fsm_state is
(
   reset,reset_wait_ack,reset_wait_bat_completion,reset_wait_id,
   reset_set_sample_rate_200,reset_set_sample_rate_200_wait_ack,
   reset_send_sample_rate_200,reset_send_sample_rate_200_wait_ack,
   reset_set_sample_rate_100,reset_set_sample_rate_100_wait_ack,
   reset_send_sample_rate_100,reset_send_sample_rate_100_wait_ack,
   reset_set_sample_rate_80,reset_set_sample_rate_80_wait_ack,
   reset_send_sample_rate_80,reset_send_sample_rate_80_wait_ack,
   reset_read_id,reset_read_id_wait_ack,reset_read_id_wait_id,
   reset_set_resolution,reset_set_resolution_wait_ack,
   reset_send_resolution,reset_send_resolution_wait_ack,
   reset_set_sample_rate_40,reset_set_sample_rate_40_wait_ack,
   reset_send_sample_rate_40,reset_send_sample_rate_40_wait_ack,
   reset_enable_reporting,reset_enable_reporting_wait_ack,
   read_byte_1,read_byte_2,read_byte_3,read_byte_4,mark_new_event
);
-- holds current state of the FSM
signal state: fsm_state := reset;

begin

   -- left output the state of the left_down register
   left <= left_down when rising_edge(clk);
   -- middle output the state of the middle_down register
   middle <= middle_down when rising_edge(clk);
   -- right output the state of the right_down register
   right <= right_down when rising_edge(clk);

   -- xpos output is the horizontal position of the mouse
   -- it has the range: 0-x_max
   xpos <= x_pos(9 downto 0) when rising_edge(clk);
   -- ypos output is the vertical position of the mouse
   -- it has the range: 0-y_max
   ypos <= y_pos(9 downto 0) when rising_edge(clk);

   -- sets the value of x_pos from another module when setx is active
   -- else, computes the new x_pos from the old position when new x
   -- movement detected by adding the delta movement in x_inc, or by
   -- adding 256 or -256 when overflow occurs.
   set_x: process(clk)
   variable x_inter: std_logic_vector(10 downto 0);
   variable inc: std_logic_vector(10 downto 0);
   begin
      if(rising_edge(clk)) then
         -- if setx active, set new x_pos value
         if(setx = '1') then
            x_pos <= '0' & value;
         -- if delta movement received from mouse
         elsif(x_new = '1') then
            -- if negative movement on x axis
            if(x_sign = '1') then
               -- if overflow occurred
               if(x_overflow = '1') then
                  -- inc is -256
                  inc := "11100000000";
               else
                  -- inc is sign extended x_inc
```

```
                inc := "111" & x_inc;
            end if;
            -- intermediary horizontal position
            x_inter := x_pos + inc;
            -- if first bit of x_inter is 1
            -- then negative overflow occurred and
            -- new x position is 0.
            -- Note: x_pos and x_inter have 11 bits,
            -- and because xpos has only 10, when
            -- first bit becomes 1, this is considered
            -- a negative number when moving left
            if(x_inter(10) = '1') then
               x_pos <= (others => '0');
            else
               x_pos <= x_inter;
            end if;
         -- if positive movement on x axis
         else
            -- if overflow occurred
            if(x_overflow = '1') then
               -- inc is 256
               inc := "00100000000";
            else
               -- inc is sign extended x_inc
               inc := "000" & x_inc;
            end if;
            -- intermediary horizontal position
            x_inter := x_pos + inc;
            -- if x_inter is greater than x_max
            -- then positive overflow occurred and
            -- new x position is x_max.
            if(x_inter > ('0' & x_max)) then
               x_pos <= '0' & x_max;
            else
               x_pos <= x_inter;
            end if;
         end if;
      end if;
   end if;
end process set_x;

-- sets the value of y_pos from another module when sety is active
-- else, computes the new y_pos from the old position when new y
-- movement detected by adding the delta movement in y_inc, or by
-- adding 256 or -256 when overflow occurs.
set_y: process(clk)
variable y_inter: std_logic_vector(10 downto 0);
variable inc: std_logic_vector(10 downto 0);
begin
   if(rising_edge(clk)) then
      -- if sety active, set new y_pos value
      if(sety = '1') then
         y_pos <= '0' & value;
      -- if delta movement received from mouse
      elsif(y_new = '1') then
         -- if negative movement on y axis
         -- Note: axes origin is upper-left corner
         if(y_sign = '1') then
            -- if overflow occurred
            if(y_overflow = '1') then
               -- inc is -256
               inc := "11100000000";
            else
               -- inc is sign extended y_inc
               inc := "111" & y_inc;
            end if;
```

```
            -- intermediary vertical position
            y_inter := y_pos + inc;
            -- if first bit of y_inter is 1
            -- then negative overflow occurred and
            -- new y position is 0.
            -- Note: y_pos and y_inter have 11 bits,
            -- and because ypos has only 10, when
            -- first bit becomes 1, this is considered
            -- a negative number when moving upward
            if(y_inter(10) = '1') then
               y_pos <= (others => '0');
            else
               y_pos <= y_inter;
            end if;
         -- if positive movement on y axis
         else
            -- if overflow occurred
            if(y_overflow = '1') then
               -- inc is 256
               inc := "00100000000";
            else
               -- inc is sign extended y_inc
               inc := "000" & y_inc;
            end if;
            -- intermediary vertical position
            y_inter := y_pos + inc;
            -- if y_inter is greater than y_max
            -- then positive overflow occurred and
            -- new y position is y_max.
            if(y_inter > ('0' & y_max)) then
               y_pos <= '0' & y_max;
            else
               y_pos <= y_inter;
            end if;
         end if;
      end if;
   end if;
end process set_y;

-- sets the maximum value of the x movement register, stored in x_max
-- when setmax_x is active, max value should be on value input pin
set_max_x: process(clk,rst)
begin
   if(rising_edge(clk)) then
      if(rst = '1') then
         x_max <= DEFAULT_MAX_X;
      elsif(setmax_x = '1') then
         x_max <= value;
      end if;
   end if;
end process set_max_x;

-- sets the maximum value of the y movement register, stored in y_max
-- when setmax_y is active, max value should be on value input pin
set_max_y: process(clk,rst)
begin
   if(rising_edge(clk)) then
      if(rst = '1') then
         y_max <= DEFAULT_MAX_Y;
      elsif(setmax_y = '1') then
         y_max <= value;
      end if;
   end if;
end process set_max_y;

-- Synchronous one process fsm to handle the communication
```

```
-- with the mouse.
-- When reset and at start-up it enters reset state
-- where it begins the procedure of initializing the mouse.
-- After initialization is complete, it waits packets from
-- the mouse.
-- Read at Behavioral decription for details.
manage_fsm: process(clk,rst)
begin
   -- when reset occurs, give signals default values.
   if(rst = '1') then
      state <= reset;
      haswheel <= '0';
      x_overflow <= '0';
      y_overflow <= '0';
      x_sign <= '0';
      y_sign <= '0';
      x_inc <= (others => '0');
      y_inc <= (others => '0');
      x_new <= '0';
      y_new <= '0';
      new_event <= '0';
      left_down <= '0';
      middle_down <= '0';
      right_down <= '0';

   elsif(rising_edge(clk)) then

      -- at every rising edge of the clock, this signals
      -- are reset, thus assuring that they are active
      -- for one clock period only if a state sets then
      -- because the fsm will transition from the state
      -- that set them on the next rising edge of clock.
      write <= '0';
      x_new <= '0';
      y_new <= '0';

      case state is

         -- if just powered-up, reset occurred or some error in
         -- transmision encountered, then fsm will transition to
         -- this state. Here the RESET command (FF) is sent to the
         -- mouse, and various signals receive their default values
         -- From here the FSM transitions to a series of states that
         -- perform the mouse initialization procedure. All this
         -- state are prefixed by "reset_". After sending a byte
         -- to the mouse, it respondes by sending ack (FA). All
         -- states that wait ack from the mouse are postfixed by
         -- "_wait_ack".
         -- Read at Behavioral decription for details.
         when reset =>
            haswheel <= '0';
            x_overflow <= '0';
            y_overflow <= '0';
            x_sign <= '0';
            y_sign <= '0';
            x_inc <= (others => '0');
            y_inc <= (others => '0');
            x_new <= '0';
            y_new <= '0';
            left_down <= '0';
            middle_down <= '0';
            right_down <= '0';
            tx_data <= FF;
            write <= '1';
            state <= reset_wait_ack;
```

```
-- wait ack for the reset command.
-- when received transition to reset_wait_bat_completion.
-- if error occurs go to reset state.
when reset_wait_ack =>
   if(read = '1') then
      -- if received ack
      if(rx_data = FA) then
         state <= reset_wait_bat_completion;
      else
         state <= reset;
      end if;
   elsif(err = '1') then
      state <= reset;
   else
      state <= reset_wait_ack;
   end if;

-- wait for bat completion test
-- mouse should send AA if test is successful
when reset_wait_bat_completion =>
   if(read = '1') then
      if(rx_data = AA) then
         state <= reset_wait_id;
      else
         state <= reset;
      end if;
   elsif(err = '1') then
      state <= reset;
   else
      state <= reset_wait_bat_completion;
   end if;

-- the mouse sends its id after performing bat test
-- the mouse id should be 00
when reset_wait_id =>
   if(read = '1') then
      if(rx_data = 00) then
         state <= reset_set_sample_rate_200;
      else
         state <= reset;
      end if;
   elsif(err = '1') then
      state <= reset;
   else
      state <= reset_wait_id;
   end if;

-- with this state begins the enable wheel mouse
-- procedure. The procedure consists of setting
-- the sample rate of the mouse first 200, then 100
-- then 80. After this is done, the mouse id is
-- requested and if the mouse id is 03, then
-- mouse is in wheel mode and will send 4 byte packets
-- when reporting is enabled.
-- If the id is 00, the mouse does not have a wheel
-- and will send 3 byte packets when reporting is enabled.
-- This state issues the set_sample_rate command to the
-- mouse.
when reset_set_sample_rate_200 =>
   tx_data <= SET_SAMPLE_RATE;
   write <= '1';
   state <= reset_set_sample_rate_200_wait_ack;

-- wait ack for set sample rate command
when reset_set_sample_rate_200_wait_ack =>
   if(read = '1') then
```

```
            if(rx_data = FA) then
               state <= reset_send_sample_rate_200;
            else
               state <= reset;
            end if;
         elsif(err = '1') then
            state <= reset;
         else
            state <= reset_set_sample_rate_200_wait_ack;
         end if;

      -- send the desired sample rate (200 = 0xC8)
      when reset_send_sample_rate_200 =>
         tx_data <= "11001000"; -- 0xC8
         write <= '1';
         state <= reset_send_sample_rate_200_wait_ack;

      -- wait ack for sending the sample rate
      when reset_send_sample_rate_200_wait_ack =>
         if(read = '1') then
            if(rx_data = FA) then
               state <= reset_set_sample_rate_100;
            else
               state <= reset;
            end if;
         elsif(err = '1') then
            state <= reset;
         else
            state <= reset_send_sample_rate_200_wait_ack;
         end if;

      -- send the sample rate command
      when reset_set_sample_rate_100 =>
         tx_data <= SET_SAMPLE_RATE;
         write <= '1';
         state <= reset_set_sample_rate_100_wait_ack;

      -- wait ack for sending the sample rate command
      when reset_set_sample_rate_100_wait_ack =>
         if(read = '1') then
            if(rx_data = FA) then
               state <= reset_send_sample_rate_100;
            else
               state <= reset;
            end if;
         elsif(err = '1') then
            state <= reset;
         else
            state <= reset_set_sample_rate_100_wait_ack;
         end if;

      -- send the desired sample rate (100 = 0x64)
      when reset_send_sample_rate_100 =>
         tx_data <= "01100100"; -- 0x64
         write <= '1';
         state <= reset_send_sample_rate_100_wait_ack;

      -- wait ack for sending the sample rate
      when reset_send_sample_rate_100_wait_ack =>
         if(read = '1') then
            if(rx_data = FA) then
               state <= reset_set_sample_rate_80;
            else
               state <= reset;
            end if;
         elsif(err = '1') then
```

```
      state <= reset;
   else
      state <= reset_send_sample_rate_100_wait_ack;
   end if;

-- send set sample rate command
when reset_set_sample_rate_80 =>
   tx_data <= SET_SAMPLE_RATE;
   write <= '1';
   state <= reset_set_sample_rate_80_wait_ack;

-- wait ack for sending the sample rate command
when reset_set_sample_rate_80_wait_ack =>
   if(read = '1') then
      if(rx_data = FA) then
         state <= reset_send_sample_rate_80;
      else
         state <= reset;
      end if;
   elsif(err = '1') then
      state <= reset;
   else
      state <= reset_set_sample_rate_80_wait_ack;
   end if;

-- send desired sample rate (80 = 0x50)
when reset_send_sample_rate_80 =>
   tx_data <= "01010000"; -- 0x50
   write <= '1';
   state <= reset_send_sample_rate_80_wait_ack;

-- wait ack for sending the sample rate
when reset_send_sample_rate_80_wait_ack =>
   if(read = '1') then
      if(rx_data = FA) then
         state <= reset_read_id;
      else
         state <= reset;
      end if;
   elsif(err = '1') then
      state <= reset;
   else
      state <= reset_send_sample_rate_80_wait_ack;
   end if;

-- now the procedure for enabling wheel mode is done
-- the mouse id is read to determine is mouse is in
-- wheel mode.
-- Read ID command is sent to the mouse.
when reset_read_id =>
   tx_data <= READ_ID;
   write <= '1';
   state <= reset_read_id_wait_ack;

-- wait ack for sending the read id command
when reset_read_id_wait_ack =>
   if(read = '1') then
      if(rx_data = FA) then
         state <= reset_read_id_wait_id;
      else
         state <= reset;
      end if;
   elsif(err = '1') then
      state <= reset;
   else
      state <= reset_read_id_wait_ack;
```

```
      end if;

   -- received the mouse id
   -- if the id is 00, then the mouse does not have
   -- a wheel and haswheel is reset
   -- if the id is 03, then the mouse is in scroll mode
   -- and haswheel is set.
   -- if anything else is received or an error occurred
   -- then the FSM transitions to reset state.
   when reset_read_id_wait_id =>
      if(read = '1') then
         if(rx_data = "000000000") then
            -- the mouse does not have a wheel
            haswheel <= '0';
            state <= reset_set_resolution;
         elsif(rx_data = "00000011") then  -- 0x03
            -- the mouse is in scroll mode
            haswheel <= '1';
            state <= reset_set_resolution;
         else
            state <= reset;
         end if;
      elsif(err = '1') then
         state <= reset;
      else
         state <= reset_read_id_wait_id;
      end if;

   -- send the set resolution command to the mouse
   when reset_set_resolution =>
      tx_data <= SET_RESOLUTION;
      write <= '1';
      state <= reset_set_resolution_wait_ack;

   -- wait ack for sending the set resolution command
   when reset_set_resolution_wait_ack =>
      if(read = '1') then
         if(rx_data = FA) then
            state <= reset_send_resolution;
         else
            state <= reset;
         end if;
      elsif(err = '1') then
         state <= reset;
      else
         state <= reset_set_resolution_wait_ack;
      end if;

   -- send the desired resolution (0x03 = 8 counts/mm)
   when reset_send_resolution =>
      tx_data <= RESOLUTION;
      write <= '1';
      state <= reset_send_resolution_wait_ack;


   -- wait ack for sending the resolution
   when reset_send_resolution_wait_ack =>
      if(read = '1') then
         if(rx_data = FA) then
            state <= reset_set_sample_rate_40;
         else
            state <= reset;
         end if;
      elsif(err = '1') then
         state <= reset;
      else
```

```
      state <= reset_send_resolution_wait_ack;
   end if;

-- send the set sample rate command
when reset_set_sample_rate_40 =>
   tx_data <= SET_SAMPLE_RATE;
   write <= '1';
   state <= reset_set_sample_rate_40_wait_ack;

-- wait ack for sending the set sample rate command
when reset_set_sample_rate_40_wait_ack =>
   if(read = '1') then
      if(rx_data = FA) then
         state <= reset_send_sample_rate_40;
      else
         state <= reset;
      end if;
   elsif(err = '1') then
      state <= reset;
   else
      state <= reset_set_sample_rate_40_wait_ack;
   end if;

-- send the desired sampele rate.
-- 40 samples per second is sent.
when reset_send_sample_rate_40 =>
   tx_data <= SAMPLE_RATE;
   write <= '1';
   state <= reset_send_sample_rate_40_wait_ack;

-- wait ack for sending the sample rate
when reset_send_sample_rate_40_wait_ack =>
   if(read = '1') then
      if(rx_data = FA) then
         state <= reset_enable_reporting;
      else
         state <= reset;
      end if;
   elsif(err = '1') then
      state <= reset;
   else
      state <= reset_send_sample_rate_40_wait_ack;
   end if;

-- in this state enable reporting command is sent
-- to the mouse. Before this point, the mouse
-- does not send packets. Only after issuing this
-- command, the mouse begins sending data packets,
-- 3 byte packets if it doesn't have a wheel and
-- 4 byte packets if it is in scroll mode.
when reset_enable_reporting =>
   tx_data <= ENABLE_REPORTING;
   write <= '1';
   state <= reset_enable_reporting_wait_ack;

-- wait ack for sending the enable reporting command
when reset_enable_reporting_wait_ack =>
   if(read = '1') then
      if(rx_data = FA) then
         state <= read_byte_1;
      else
         state <= reset;
      end if;
   elsif(err = '1') then
      state <= reset;
   else
```

```
            state <= reset_enable_reporting_wait_ack;
         end if;

      -- this is idle state of the FSM after the
      -- initialization is complete.
      -- Here the first byte of a packet is waited.
      -- The first byte contains the state of the
      -- buttons, the sign of the x and y movement
      -- and overflow information about these movements
      -- First byte looks like this:
      --    7       6        5        4      3   2   1   0
      -----------------------------------------------------
      -- | Y OVF | X OVF | Y SIGN | X SIGN | 1 | M | R | L |
      -----------------------------------------------------
      when read_byte_1 =>
         -- reset new_event when back in idle state.
         new_event <= '0';
         -- reset last z delta movement
         zpos <= (others => '0');
         if(read = '1') then
            -- mouse button states
            left_down <= rx_data(0);
            middle_down <= rx_data(2);
            right_down <= rx_data(1);
            -- sign of the movement data
            x_sign <= rx_data(4);
            -- y sign is changed to invert the y axis
            -- because the mouse uses the lower-left corner
            -- as axes origin and it is placed in the upper-left
            -- corner by this inversion (suitable for displaying
            -- a mouse cursor on the screen).
            -- y movement data from the third packet must be
            -- also negated.
            y_sign <= not rx_data(5);

            -- overflow status of the x and y movement
            x_overflow <= rx_data(6);
            y_overflow <= rx_data(7);

            -- transition to state read_byte_2
            state <= read_byte_2;
         else
            -- no byte received yet.
            state <= read_byte_1;
         end if;

      -- wait the second byte of the packet
      -- this byte contains the x movement counter.
      when read_byte_2 =>
         if(read = '1') then
            -- put the delta movement in x_inc
            x_inc <= rx_data;
            -- signal the arrival of new x movement data.
            x_new <= '1';
            -- go to state read_byte_3.
            state <= read_byte_3;
         elsif(err = '1') then
            state <= reset;
         else
            -- byte not received yet.
            state <= read_byte_2;
         end if;

      -- wait the third byte of the data, that
      -- contains the y data movement counter.
      -- negate its value, for the axis to be
```

```
            -- inverted.
            -- If mouse is in scroll mode, transition
            -- to read_byte_4, else go to mark_new_event
            when read_byte_3 =>
               if(read = '1') then
                  -- when y movement is 0, then ignore
                  if(rx_data /= "00000000") then
                     -- 2's complement positive numbers
                     -- become negative and vice versa
                     y_inc <= (not rx_data) + "00000001";
                     y_new <= '1';
                  end if;
                  -- if the mouse has a wheel then transition
                  -- to read_byte_4, else go to mark_new_event
                  if(haswheel = '1') then
                     state <= read_byte_4;
                  else
                     state <= mark_new_event;
                  end if;
               elsif(err = '1') then
                  state <= reset;
               else
                     state <= read_byte_3;
               end if;

            -- only reached when mouse is in scroll mode
            -- wait for the fourth byte to arrive
            -- fourth byte contains the z movement counter
            -- only least significant 4 bits are relevant
            -- the rest are sign extension.
            when read_byte_4 =>
               if(read = '1') then
                  -- zpos is the delta movement on z
                  zpos <= rx_data(3 downto 0);
                  -- packet completly received,
                  -- go to mark_new_event
                  state <= mark_new_event;
               elsif(err = '1') then
                  state <= reset;
               else
                  state <= read_byte_4;
               end if;

            -- set new_event high
            -- it will be reset in next state
            -- informs client new packet received and processed
            when mark_new_event =>
               new_event <= '1';
               state <= read_byte_1;

            -- if invalid transition occurred, reset
            when others =>
               state <= reset;

         end case;
      end if;
   end process manage_fsm;


end Behavioral;

----------------------------------------------------------------------
-- ps2interface.vhd
----------------------------------------------------------------------
-- Author : Ulrich Zolt?n
--          Copyright 2006 Digilent, Inc.
```

```
----------------------------------------------------------------------
-- Software version : Xilinx ISE 7.1.04i
--                    WebPack
-- Device           : 3s200ft256-4
----------------------------------------------------------------------
-- This file contains the implementation of a generic bidirectional
-- ps/2 interface.
----------------------------------------------------------------------
--  Behavioral description
----------------------------------------------------------------------
-- Please read the following article on the web for understanding how
-- the ps/2 protocol works.
-- http://www.computer-engineering.org/ps2protocol/

-- This module implements a generic bidirectional ps/2 interface. It can
-- be used with any ps/2 compatible device. It offers its clients a
-- convenient way to exchange data with the device. The interface
-- transparently wraps the byte to be sent into a ps/2 frame, generates
-- parity for byte and sends the frame one bit at a time to the device.
-- Similarly, when receiving data from the ps2 device, the interface
-- receives the frame, checks for parity, and extract the usefull data
-- and forwards it to the client. If an error occurs during receiving
-- or sending a byte, the client is informed by settings the err output
-- line high. This way, the client can resend the data or can issue
-- a resend command to the device.

-- The physical ps/2 interface uses 4 lines
-- For the 6-pin connector pins are assigned as follows:
-- 1 - Data
-- 2 - Not Implemented
-- 3 - Ground
-- 4 - Vcc (+5V)
-- 5 - Clock
-- 6 - Not Implemented

-- The clock line carries the device generated clock which has a
-- frequency in range 10 - 16.7 kHz (30 to 50us). When line is idle
-- it is placed in high impedance. The clock is only generated when
-- device is sending or receiving data.
-- The Data and Clock lines are both open-collector with pullup
-- resistors to Vcc. An "open-collector" interface has two possible
-- states: low('0') or high impedance('Z').

-- When device wants to send a byte, it pulls the clock line low and the
-- host(i.e. this interfaces) recognizes that the device is sending data
-- When the host wants to send data, it maeks a request to send. This
-- is done by holding the clock line low for at least 100us, then with
-- the clock line low, the data line is brought low. Next the clock line
-- is released (placed in high impedance). The devices begins generating
-- clock signal on clock line.
-- When receiving data, bits are read from the data line (ps2_data) on
-- the falling edge of the clock (ps2_clk). When sending data, the
-- device reads the bits from the data line on the rising edge of the
-- clock.
-- A frame for sending a byte is comprised of 11 bits as shown bellow:
-- bits     10     9    8    7    6    5    4    3    2    1      0
--         -----------------------------------------------------------
--         | STOP| PAR | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | START |
--         -----------------------------------------------------------
-- STOP  - stop  bit, always '1'
-- PAR   - parity bit, odd parity for the 8 data bits.
--       - select in such way that the number of bits of '1' in the data
--       - bits together with parity bit is odd.
-- D0-7  - data bits.
-- START - start bit, always '0'
--
```

```
-- Frame is sent bit by bit starting with the least significant bit
-- (starting bit) and is received the same way. This is done, when
-- receiving, by shifting the frame register to the left when a bit
-- is available and placing the bit on data line on the most significant
-- bit. This way the first bit sent will reach the least significant bit
-- of the frame when all the bits have been received. When sending data
-- the least significant bit of the frame is placed on the data line
-- and the frame is shifted to the right when another bit needs to be
-- sent. During the request to send, when releasing the clock line,
-- the device reads the data line and interprets the data on it as the
-- first bit of the frame. Data line is low at that time, at this is the
-- way the start bit('0') is sent. Because of this, when sending, only
-- 10 shifts of the frame will be made.
-- While the interface is sending or receiving data, the busy output
-- signal goes high. When interface is idle, busy is low.
-- After sending all the bits in the frame, the device must acknowledge
-- the data sent. This is done by the host releasing and data line
-- (clock line is already released) after the last bit is sent. The
-- devices brings the data line and the clock line low, in this order,
-- to acknowledge the data. If data line is high when clock line goes
-- low after last bit, the device did not acknowledge the data and
-- err output is set.
-- A FSM is used to manage the transitions the set all the command
-- signals. States that begin with "rx_" are used to receive data
-- from device and states begining with "tx_" are used to send data
-- to the device.
-- For the parity bit, a ROM holds the parity bit for all possible
-- data (256 possible values, since 8 bits of data). The ROM has
-- dimensions 256x1bit. For obtaining the parity bit of a value,
-- the bit at the data value address is read. Ex: to find the parity
-- bit of 174, the bit at address 174 is read.
-- For generating the necessary delay, counters are used. For example,
-- to generate the 100us delay a 14 bit counter is used that has the
-- upper limit for counting 10000. The interface is designed to run
-- at 100MHz. Thus, 10000x10ns = 100us.

----------------------------------------------------------------------
-- If using the interface at different frequency than 100MHz, adjusting
-- the delay counters is necessary!!!
----------------------------------------------------------------------

-- Clock line(ps2_clk) and data line(ps2_data) are passed through a
-- debouncer for the transitions of the clock and data to be clean.
-- Also, ps2_clk_s and ps2_data_s hold the debounced and synchronized
-- value of the clock and data line to the system clock(clk).
----------------------------------------------------------------------
--  Port definitions
----------------------------------------------------------------------
-- ps2_clk        - inout pin, clock line of the ps/2 interface
-- ps2_data       - inout pin, clock line of the ps/2 interface
-- clk            - input pin, system clock signal
-- rst            - input pin, system reset signal
-- tx_data        - input pin, 8 bits, from client
--                - data to be sent to the device
-- write          - input pin, from client
--                - should be active for one clock period when then
--                - client wants to send data to the device and
--                - data to be sent is valid on tx_data
-- rx_data        - output pin, 8 bits, to client
--                - data received from device
-- read           - output pin, to client
--                - active for one clock period when new data is
--                - available from device
-- busy           - output pin, to client
--                - active while sending or receiving data.
-- err            - output pin, to client
```

```
--                - active for one clock period when an error occurred
--                - during sending or receiving.
------------------------------------------------------------------------
-- Revision History:
-- 09/18/2006(UlrichZ): created
------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- simulation library
library UNISIM;
use UNISIM.VComponents.all;

-- the ps2interface entity declaration
-- read above for behavioral description and port definitions.
entity ps2interface is
port(
   ps2_clk  : inout std_logic;
   ps2_data : inout std_logic;

   clk      : in std_logic;
   rst      : in std_logic;

   tx_data  : in std_logic_vector(7 downto 0);
   write    : in std_logic;

   rx_data  : out std_logic_vector(7 downto 0);
   read     : out std_logic;
   busy     : out std_logic;
   err      : out std_logic
);

-- forces the extraction of distributed ram for
-- the parityrom memory.
-- please remove if block ram is preffered.
attribute rom_extract : string;
attribute rom_extract of ps2interface: entity is "yes";
attribute rom_style : string;
attribute rom_style of ps2interface: entity is "distributed";

end ps2interface;

architecture Behavioral of ps2interface is

------------------------------------------------------------------------
-- CONSTANTS
------------------------------------------------------------------------

-- Values are valid for a 100MHz clk. Please adjust for other
-- frequencies if necessary!

-- upper limit for 100us delay counter.
-- 10000 * 10ns = 100us
constant DELAY_100US : std_logic_vector(13 downto 0):= "10011100010000";
                                                    -- 10000 clock periods
-- upper limit for 20us delay counter.
-- 2000 * 10ns = 20us
constant DELAY_20US  : std_logic_vector(10 downto 0) := "11111010000";
                                                     -- 2000 clock periods
-- upper limit for 63clk delay counter.
constant DELAY_63CLK : std_logic_vector(5 downto 0)  := "111111";
                                                    -- 63 clock periods
-- delay from debouncing ps2_clk and ps2_data signals
```

```
constant DEBOUNCE_DELAY : std_logic_vector(3 downto 0)  := "1111";

-- number of bits in a frame
constant NUMBITS: std_logic_vector(3 downto 0) := "1011"; -- 11

-- parity bit position in frame
constant PARITY_BIT: positive := 9;

-- (odd) parity bit ROM
-- Used instead of logic because this way speed is far greater
-- 256x1bit rom
-- If the odd parity bit for a 8 bits number, x, is needed
-- the bit at address x is the parity bit.
type ROM is array(0 to 255) of std_logic;
constant parityrom : ROM := (
'1','0','0','1','0','1','1','0',
'0','1','1','0','1','0','0','1',
'0','1','1','0','1','0','0','1',
'1','0','0','1','0','1','1','0',
'0','1','1','0','1','0','0','1',
'1','0','0','1','0','1','1','0',
'1','0','0','1','0','1','1','0',
'0','1','1','0','1','0','0','1',
'0','1','1','0','1','0','0','1',
'1','0','0','1','0','1','1','0',
'1','0','0','1','0','1','1','0',
'0','1','1','0','1','0','0','1',
'1','0','0','1','0','1','1','0',
'0','1','1','0','1','0','0','1',
'0','1','1','0','1','0','0','1',
'1','0','0','1','0','1','1','0',
'0','1','1','0','1','0','0','1',
'1','0','0','1','0','1','1','0',
'1','0','0','1','0','1','1','0',
'0','1','1','0','1','0','0','1',
'1','0','0','1','0','1','1','0',
'0','1','1','0','1','0','0','1',
'0','1','1','0','1','0','0','1',
'1','0','0','1','0','1','1','0',
'1','0','0','1','0','1','1','0',
'0','1','1','0','1','0','0','1',
'0','1','1','0','1','0','0','1',
'1','0','0','1','0','1','1','0',
'0','1','1','0','1','0','0','1',
'1','0','0','1','0','1','1','0',
'1','0','0','1','0','1','1','0',
'0','1','1','0','1','0','0','1'
);

--------------------------------------------------------------------------------
-- SIGNALS
--------------------------------------------------------------------------------

-- 14 bits counter
-- max value DELAY_100US
-- used to wait 100us
signal delay_100us_count: std_logic_vector(13 downto 0) :=
                                           (others => '0');

-- 11 bits counter
-- max value DELAY_20US
-- used to wait 20us
signal delay_20us_count: std_logic_vector(10 downto 0) :=
                                         (others => '0');
-- 11 bits counter
-- max value DELAY_63CLK
```

```
-- used to wait 63 clock periods
signal delay_63clk_count: std_logic_vector(5 downto 0) :=
                                          (others => '0');

-- done signal for the couters above
-- when a counter reaches max value,the corresponding done signal is set
signal delay_100us_done, delay_20us_done, delay_63clk_done: std_logic;

-- enable signal for 100us delay counter
signal delay_100us_counter_enable: std_logic := '0';
-- enable signal for 20us delay counter
signal delay_20us_counter_enable : std_logic := '0';
-- enable signal for 63clk delay counter
signal delay_63clk_counter_enable: std_logic := '0';

-- synchronzed input for ps2_clk and ps2_data
signal ps2_clk_s,ps2_data_s: std_logic := '1';

-- control the output of ps2_clk and ps2_data
-- if 1 then corresponding signal (ps2_clk or ps2_data) is
-- put in high impedance ('Z').
signal ps2_clk_h,ps2_data_h: std_logic := '1';

-- states of the FSM for controlling the communcation with the device
-- states that begin with "rx_" are used when receiving data
-- states that begin with "tx_" are used when transmiting data
type fsm_state is
(
   idle,rx_clk_h,rx_clk_l,rx_down_edge,rx_error_parity,rx_data_ready,
   tx_force_clk_l,tx_bring_data_down,tx_release_clk,
   tx_first_wait_down_edge,tx_clk_l,tx_wait_up_edge,tx_clk_h,
   tx_wait_up_edge_before_ack,tx_wait_ack,tx_received_ack,
   tx_error_no_ack
);

-- the signal that holds the current state of the FSM
-- implicitly state is idle.
signal state: fsm_state := idle;

-- register that holds the frame received or the one to be sent.
-- Its contents are shifted in from the bus one bit at a time
-- from left to right when receiving data and are shifted on the
-- bus (ps2_data) one bit at a time to the right when sending data
signal frame: std_logic_vector(10 downto 0) := (others => '0');

-- how many bits have been sent or received.
signal bit_count: std_logic_vector(3 downto 0) := (others => '0');

-- when active the bit counter is reset.
signal reset_bit_count: std_logic := '0';

-- when active the contents of the frame is shifted to the right
-- and the most significant bit of frame is loaded with ps2_data.
signal shift_frame: std_logic := '0';

-- parity of the byte that was received from the device.
-- must match the parity bit received, else error occurred.
signal rx_parity: std_logic := '0';
-- parity bit that is sent with the frame, representing the
-- odd parity of the byte currently being sent
signal tx_parity: std_logic := '0';

-- when active, frame is loaded with the start bit, data on
-- tx_data, parity bit (tx_parity) and stop bit
-- this frame will be sent to the device.
signal load_tx_data: std_logic := '0';
```

```
-- when active bits 8 downto 1 from frame are loaded into
-- rx_data register. This is the byte received from the device.
signal load_rx_data: std_logic := '0';

-- intermediary signals used to debounce the inputs ps2_clk and ps2_data
signal ps2_clk_clean,ps2_data_clean: std_logic := '1';
-- debounce counter for the ps2_clk input and the ps2_data input.
signal clk_count,data_count: std_logic_vector(3 downto 0);
-- last value on ps2_clk and ps2_data.
signal clk_inter,data_inter: std_logic := '1';

begin

   ----------------------------------------------------------------------
   -- FLAGS and PS2 CLOCK AND DATA LINES
   ----------------------------------------------------------------------

   -- clean ps2_clk signal (debounce)
   -- note that this introduces a delay in ps2_clk of
   -- DEBOUNCE_DELAY clocks
   process(clk)
   begin
      if(rising_edge(clk)) then
         -- if the current bit on ps2_clk is different
         -- from the last value, then reset counter
         -- and retain value
         if(ps2_clk /= clk_inter) then
            clk_inter <= ps2_clk;
            clk_count <= (others => '0');
         -- if counter reached upper limit, then
         -- the signal is clean
         elsif(clk_count = DEBOUNCE_DELAY) then
            ps2_clk_clean <= clk_inter;
         -- ps2_clk did not change, but counter did not
         -- reach limit. Increment counter
         else
            clk_count <= clk_count + 1;
         end if;
      end if;
   end process;

   -- clean ps2_data signal (debounce)
   -- note that this introduces a delay in ps2_data of
   -- DEBOUNCE_DELAY clocks
   process(clk)
   begin
      if(rising_edge(clk)) then
         -- if the current bit on ps2_data is different
         -- from the last value, then reset counter
         -- and retain value
         if(ps2_data /= data_inter) then
            data_inter <= ps2_data;
            data_count <= (others => '0');
         -- if counter reached upper limit, then
         -- the signal is clean
         elsif(data_count = DEBOUNCE_DELAY) then
            ps2_data_clean <= data_inter;
         -- ps2_data did not change, but counter did not
         -- reach limit. Increment counter
         else
            data_count <= data_count + 1;
         end if;
      end if;
   end process;
```

```
-- Synchronize ps2 entries
ps2_clk_s <= ps2_clk_clean when rising_edge(clk);
ps2_data_s <= ps2_data_clean when rising_edge(clk);

-- Assign parity from frame bits 8 downto 1, this is the parity
-- that should be received inside the frame on PARITY_BIT position
rx_parity <= parityrom(conv_integer(frame(8 downto 1)))
             when rising_edge(clk);
-- The parity for the data to be sent
tx_parity <= parityrom(conv_integer(tx_data)) when rising_edge(clk);

-- Force ps2_clk to '0' if ps2_clk_h = '0', else release the line
-- ('Z' = +5Vcc because of pull-ups)
ps2_clk <= 'Z' when ps2_clk_h = '1' else '0';

-- Force ps2_data to '0' if ps2_data_h = '0', else release the line
-- ('Z' = +5Vcc because of pull-ups)
ps2_data <= 'Z' when ps2_data_h = '1' else '0';

-- Control busy flag. Interface is not busy while in idle state.
busy <= '0' when state = idle else '1';

-- reset the bit counter when in idle state.
reset_bit_count <= '1' when state = idle else '0';

-- Control shifting of the frame
-- When receiving from device, data is read
-- on the falling edge of ps2_clk
-- When sending to device, data is read by device
-- on the rising edge of ps2_clk
shift_frame <= '1' when state = rx_down_edge or
                        state = tx_clk_l else
               '0';

-------------------------------------------------------------------------
-- FINITE STATE MACHINE
-------------------------------------------------------------------------

-- For the current state establish next state
-- and give necessary commands
manage_fsm: process(clk,rst,state,ps2_clk_s,ps2_data_s,write,tx_data,
                    bit_count,rx_parity,frame,delay_100us_done,
                    delay_20us_done,delay_63clk_done)
begin
   -- if reset occurs, go to idle state.
   if(rst = '1') then
      state <= idle;
   elsif(rising_edge(clk)) then

      -- default values for these signals
      -- ensures signals are reset to default value
      -- when coditions for their activation are no
      -- longer applied (transition to other state,
      -- where signal should not be active)
      -- Idle value for ps2_clk and ps2_data is 'Z'
      ps2_clk_h <= '1';
      ps2_data_h <= '1';
      load_tx_data <= '0';
      load_rx_data <= '0';
      read <= '0';
      err <= '0';

      case state is

         -- wait for the device to begin a transmission
         -- by pulling the clock line low and go to state
```

```
-- rx_down_edge or, if write is high, the
-- client of this interface wants to send a byte
-- to the device and a transition is made to state
-- tx_force_clk_l
when idle =>
   if(ps2_clk_s = '0') then
      state <= rx_down_edge;
   elsif(write = '1') then
      state <= tx_force_clk_l;
   else
      state <= idle;
   end if;

-- ps2_clk is high, check if all the bits have been read
-- if, last bit read, check parity, and if parity ok
-- load received data into rx_data.
-- else if more bits left, then wait for the ps2_clk to
-- go low
when rx_clk_h =>
   if(bit_count = NUMBITS) then
      if(not (rx_parity = frame(PARITY_BIT))) then
         state <= rx_error_parity;
      else
         load_rx_data <= '1';
         state <= rx_data_ready;
      end if;
   elsif(ps2_clk_s = '0') then
      state <= rx_down_edge;
   else
      state <= rx_clk_h;
   end if;

-- data must be read into frame in this state
-- the ps2_clk just transitioned from high to low
when rx_down_edge =>
   state <= rx_clk_l;

-- ps2_clk line is low, wait for it to go high
when rx_clk_l =>
   if(ps2_clk_s = '1') then
      state <= rx_clk_h;
   else
      state <= rx_clk_l;
   end if;

-- parity bit received is invalid
-- signal error and go back to idle.
when rx_error_parity =>
   err <= '1';
   state <= idle;

-- parity bit received was good
-- set read signal for the client to know
-- a new byte was received and is available on rx_data
when rx_data_ready =>
   read <= '1';
   state <= idle;

-- the client wishes to transmit a byte to the device
-- this is done by holding ps2_clk down for at least 100us
-- bringing down ps2_data, wait 20us and then releasing
-- the ps2_clk.
-- This constitutes a request to send command.
-- In this state, the ps2_clk line is held down and
-- the counter for waiting 100us is eanbled.
-- when the counter reached upper limit, transition
```

```
-- to tx_bring_data_down
when tx_force_clk_l =>
   load_tx_data <= '1';
   ps2_clk_h <= '0';
   if(delay_100us_done = '1') then
      state <= tx_bring_data_down;
   else
      state <= tx_force_clk_l;
   end if;

-- with the ps2_clk line low bring ps2_data low
-- wait for 20us and then go to tx_release_clk
when tx_bring_data_down =>
   -- keep clock line low
   ps2_clk_h <= '0';
   -- set data line low
   -- when clock is released in the next state
   -- the device will read bit 0 on data line
   -- and this bit represents the start bit.
   ps2_data_h <= '0';   -- start bit = '0'
   if(delay_20us_done = '1') then
      state <= tx_release_clk;
   else
      state <= tx_bring_data_down;
   end if;

-- release the ps2_clk line
-- keep holding data line low
when tx_release_clk =>
   ps2_clk_h <= '1';
   -- must maintain data low,
   -- otherwise will be released by default value
   ps2_data_h <= '0';
   state <= tx_first_wait_down_edge;

-- state is necessary because the clock signal
-- is not released instantaneously and, because of debounce,
-- delay is even greater.
-- Wait 63 clock periods for the clock line to release
-- then if clock is low then go to tx_clk_l
-- else wait until ps2_clk goes low.
when tx_first_wait_down_edge =>
   ps2_data_h <= '0';
   if(delay_63clk_done = '1') then
      if(ps2_clk_s = '0') then
         state <= tx_clk_l;
      else
         state <= tx_first_wait_down_edge;
      end if;
   else
      state <= tx_first_wait_down_edge;
   end if;

-- place the least significant bit from frame
-- on the data line
-- During this state the frame is shifted one
-- bit to the right
when tx_clk_l =>
   ps2_data_h <= frame(0);
   state <= tx_wait_up_edge;

-- wait for the clock to go high
-- this is the edge on which the device reads the data
-- on ps2_data.
-- keep holding ps2_data on frame(0) because else
-- will be released by default value.
```

```
-- Check if sent the last bit and if so, release data line
-- and go to state that wait for acknowledge
when tx_wait_up_edge =>
   ps2_data_h <= frame(0);
   -- NUMBITS - 1 because first (start bit = 0) bit was read
   -- when the clock line was released in the request to
   -- send command (see tx_bring_data_down state).
   if(bit_count = NUMBITS-1) then
      ps2_data_h <= '1';
      state <= tx_wait_up_edge_before_ack;
   -- if more bits to send, wait for the up edge
   -- of ps2_clk
   elsif(ps2_clk_s = '1') then
      state <= tx_clk_h;
   else
      state <= tx_wait_up_edge;
   end if;

-- ps2_clk is released, wait for down edge
-- and go to tx_clk_l when arrived
when tx_clk_h =>
   ps2_data_h <= frame(0);
   if(ps2_clk_s = '0') then
      state <= tx_clk_l;
   else
      state <= tx_clk_h;
   end if;

-- release ps2_data and wait for rising edge of ps2_clk
-- once this occurs, transition to tx_wait_ack
when tx_wait_up_edge_before_ack =>
   ps2_data_h <= '1';
   if(ps2_clk_s = '1') then
      state <= tx_wait_ack;
   else
      state <= tx_wait_up_edge_before_ack;
   end if;

-- wait for the falling edge of the clock line
-- if data line is low when this occurs, the
-- ack is received
-- else if data line is high, the device did not
-- acknowledge the transimission
when tx_wait_ack =>
   if(ps2_clk_s = '0') then
      if(ps2_data_s = '0') then
         -- acknowledge received
         state <= tx_received_ack;
      else
         -- acknowledge not received
         state <= tx_error_no_ack;
      end if;
   else
      state <= tx_wait_ack;
   end if;

-- wait for ps2_clk to be released together with ps2_data
-- (bus to be idle) and go back to idle state
when tx_received_ack =>
   if(ps2_clk_s = '1' and ps2_data_s = '1') then
      state  <= idle;
   else
      state <= tx_received_ack;
   end if;

-- wait for ps2_clk to be released together with ps2_data
```

```
            -- (bus to be idle) and go back to idle state
            -- signal error for not receiving ack
            when tx_error_no_ack =>
               if(ps2_clk_s = '1' and ps2_data_s = '1') then
                  err <= '1';
                  state  <= idle;
               else
                  state <= tx_error_no_ack;
               end if;

            -- if invalid transition occurred, signal error and
            -- go back to idle state
            when others =>
               err <= '1';
               state  <= idle;

         end case;
      end if;
   end process manage_fsm;

   ----------------------------------------------------------------------
   -- DELAY COUNTERS
   ----------------------------------------------------------------------

   -- Enable the 100us counter only when state is tx_force_clk_l
   delay_100us_counter_enable <= '1' when state = tx_force_clk_l else '0';

   -- Counter for a 100us delay
   -- after done counting, done signal remains active until
   -- enable counter is reset.
   delay_100us_counter: process(clk)
   begin
      if(rising_edge(clk)) then
         if(delay_100us_counter_enable = '1') then
            if(delay_100us_count = (DELAY_100US)) then
               delay_100us_count <= delay_100us_count;
               delay_100us_done <= '1';
            else
               delay_100us_count <= delay_100us_count + 1;
               delay_100us_done <= '0';
            end if;
         else
            delay_100us_count <= (others => '0');
            delay_100us_done <= '0';
         end if;
      end if;
   end process delay_100us_counter;

   -- Enable the 20us counter only when state is tx_bring_data_down
   delay_20us_counter_enable <= '1' when state = tx_bring_data_down else '0';

   -- Counter for a 20us delay
   -- after done counting, done signal remains active until
   -- enable counter is reset.
   delay_20us_counter: process(clk)
   begin
      if(rising_edge(clk)) then
         if(delay_20us_counter_enable = '1') then
            if(delay_20us_count = (DELAY_20US)) then
               delay_20us_count <= delay_20us_count;
               delay_20us_done <= '1';
            else
               delay_20us_count <= delay_20us_count + 1;
               delay_20us_done <= '0';
            end if;
         else
```

```
            delay_20us_count <= (others => '0');
            delay_20us_done <= '0';
         end if;
      end if;
   end process delay_20us_counter;

   -- Enable the 63clk counter only when state is tx_first_wait_down_edge
   delay_63clk_counter_enable <= '1' when state = tx_first_wait_down_edge else
'0';

   -- Counter for a 63 clock periods delay
   -- after done counting, done signal remains active until
   -- enable counter is reset.
   delay_63clk_counter: process(clk)
   begin
      if(rising_edge(clk)) then
         if(delay_63clk_counter_enable = '1') then
            if(delay_63clk_count = (DELAY_63CLK)) then
               delay_63clk_count <= delay_63clk_count;
               delay_63clk_done <= '1';
            else
               delay_63clk_count <= delay_63clk_count + 1;
               delay_63clk_done <= '0';
            end if;
         else
            delay_63clk_count <= (others => '0');
            delay_63clk_done <= '0';
         end if;
      end if;
   end process delay_63clk_counter;

   ------------------------------------------------------------------------
   -- BIT COUNTER AND FRAME SHIFTING LOGIC
   ------------------------------------------------------------------------

   -- counts the number of bits shifted into the frame
   -- or out of the frame.
   bit_counter: process(clk)
   begin
      if(rising_edge(clk)) then
         if(reset_bit_count = '1') then
            bit_count <= (others => '0');
         elsif(shift_frame = '1') then
            bit_count <= bit_count + 1;
         end if;
      end if;
   end process bit_counter;

   -- shifts frame with one bit to right when shift_frame is acitve
   -- and loads data into frame from tx_data then load_tx_data is high
   load_tx_data_into_frame: process(clk)
   begin
      if(rising_edge(clk)) then
         if(load_tx_data = '1') then
            frame(8 downto 1) <= tx_data;       -- byte to send
            frame(0) <= '0';                    -- start bit
            frame(10) <= '1';                   -- stop bit
            frame(9) <= tx_parity;              -- parity bit
         elsif(shift_frame = '1') then
            -- shift right 1 bit
            frame(9 downto 0) <= frame(10 downto 1);
            -- shift in from the ps2_data line
            frame(10) <= ps2_data_s;
         end if;
      end if;
   end process load_tx_data_into_frame;
```

```
   -- Loads data from frame into rx_data output when data is ready
   do_load_rx_data: process(clk)
   begin
      if(rising_edge(clk)) then
         if(load_rx_data = '1') then
            rx_data <= frame(8 downto 1);
         end if;
      end if;
   end process do_load_rx_data;

end Behavioral;

--------------------------------------------------------------------------------
-- resolution_mouse_informer.vhd
--------------------------------------------------------------------------------
-- Author : Ulrich Zolt?n
--          Copyright 2006 Digilent, Inc.
--------------------------------------------------------------------------------
-- Software version : Xilinx ISE 7.1.04i
--                    WebPack
-- Device           : 3s200ft256-4
--------------------------------------------------------------------------------
-- This file contains the logic that send the mouse_controller new
-- position of the mouse and new maximum values for the position
-- when resolution changes, so that the mouse will be centered on the
-- screen and the bounds for the new resolution are properly set.
--------------------------------------------------------------------------------
--  Behavioral description
--------------------------------------------------------------------------------
-- This module implements the logic that sets the position of the mouse
-- when the fpga is powered-up and when the resolution changes. It
-- also sets the bounds of the mouse corresponding to the currently used
-- resolution.
-- The mouse is centered for the currently selected resolution and the
-- bounds are set appropriately. This way the mouse will first appear
-- in the center in the screen at start-up and when resolution is
-- changed and cannot leave the screen.
-- The position (and similarly the bounds) is set by placing and number
-- representing the middle of the screen dimension on the value output
-- and activation the corresponding set signal (setx for horizontal
-- position, sety for vertical position, setmax_x for horizontal
-- maximum value, setmax_y for the veritcal maximum value).
--------------------------------------------------------------------------------
--  Port definitions
--------------------------------------------------------------------------------
-- clk         - global clock signal
-- rst         - reset signal
-- resolution  - input pin, from resolution_switcher
--             - 0 for 640x480 selected resolution
--             - 1 for 800x600 selected resolution
-- switch      - input pin, from resolution_switcher
--             - active for one clock period when resolution changes
-- value       - output pin, 10 bits, to mouse_controller
--             - position on x or y, max value for x or y
--             - that is sent to the mouse_controller
-- setx        - output pin, to mouse_controller
--             - active for one clock period when the horizontal
--             - position of the mouse cursor is valid on value output
-- sety        - output pin, to mouse_controller
--             - active for one clock period when the vertical
--             - position of the mouse cursor is valid on value output
-- setmax_x    - output pin, to mouse_controller
--             - active for one clock period when the horizontal
--             - maximum position of the mouse cursor is valid on
--             - value output
```

```
-- setmax_y    - output pin, to mouse_controller
--             - active for one clock period when the vertical
--             - maximum position of the mouse cursor is valid on
--             - value output
----------------------------------------------------------------------------
-- Revision History:
-- 09/18/2006(UlrichZ): created
----------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- simulation library
library UNISIM;
use UNISIM.VComponents.all;

-- the resolution_mouse_informer entity declaration
-- read above for behavioral description and port definitions.
entity resolution_mouse_informer is
port (
   clk         : in std_logic;
   rst         : in std_logic;

   resolution  : in std_logic;
   switch      : in std_logic;

   value       : out std_logic_vector(9 downto 0);
   setx        : out std_logic;
   sety        : out std_logic;
   setmax_x    : out std_logic;
   setmax_y    : out std_logic
);
end resolution_mouse_informer;

architecture Behavioral of resolution_mouse_informer is

----------------------------------------------------------------------------
-- CONSTANTS
----------------------------------------------------------------------------

-- center horizontal position of the mouse for 640x480 and 800x600
constant POS_X_128: std_logic_vector(9 downto 0) := "1000000000"; -- 64
--constant POS_X_640: std_logic_vector(9 downto 0) := "0101000000"; -- 320
constant POS_X_800: std_logic_vector(9 downto 0) := "0110010000"; -- 400

-- center vertical position of the mouse for 640x480 and 800x600
constant POS_Y_128: std_logic_vector(9 downto 0) := "1000000000"; -- 32
--constant POS_Y_640: std_logic_vector(9 downto 0) := "0011110000"; -- 240
constant POS_Y_800: std_logic_vector(9 downto 0) := "0100101100"; -- 300

-- maximum horizontal position of the mouse for 640x480 and 800x600
constant MAX_X_128: std_logic_vector(9 downto 0) := "1111111111"; -- 127 = 6 msb
--constant MAX_X_640: std_logic_vector(9 downto 0) := "1001111111"; -- 639
constant MAX_X_800: std_logic_vector(9 downto 0) := "1100011111"; -- 799

-- maximum vertical position of the mouse for 640x480 and 800x600
constant MAX_Y_128: std_logic_vector(9 downto 0) := "1111111111"; -- 63 = 5 msb
--constant MAX_Y_640: std_logic_vector(9 downto 0) := "0111011111"; -- 479
constant MAX_Y_800: std_logic_vector(9 downto 0) := "1001010111"; -- 599

constant RES_128  : std_logic := '1';
--constant RES_640  : std_logic := '0';
constant RES_800  : std_logic := '0';
```

```
--------------------------------------------------------------------------
-- SIGNALS
--------------------------------------------------------------------------

type fsm_state is (sReset,sIdle,sSetX,sSetY,sSetMaxX,sSetMaxY);
-- signal that holds the current state of the FSM
signal state: fsm_state := sIdle;

begin

   -- value receives the horizontal position of the mouse, the vertical
   -- position, the maximum horizontal value and maximum vertical
   -- value for the active resolution when in the apropriate state
   value <= POS_X_128 when state = sSetX and resolution = RES_128 else
             POS_X_800 when state = sSetX and resolution = RES_800 else
             POS_Y_128 when state = sSetY and resolution = RES_128 else
             POS_Y_800 when state = sSetY and resolution = RES_800 else
            MAX_X_128 when state = sSetMaxX and resolution = RES_128 else
            MAX_X_800 when state = sSetMaxX and resolution = RES_800 else
            MAX_Y_128 when state = sSetMaxY and resolution = RES_128 else
            MAX_Y_800 when state = sSetMaxY and resolution = RES_800 else
            (others => '0');

   -- when in state sSetX, set the horizontal value for the mouse
   setx <= '1' when state = sSetX else '0';
   -- when in state sSetY, set the vertical value for the mouse
   sety <= '1' when state = sSetY else '0';
   -- when in state sSetMaxX, set the horizontal max value for the mouse
   setmax_x <= '1' when state = sSetMaxX else '0';
   -- when in state sSetMaxX, set the vertical max value for the mouse
   setmax_y <= '1' when state = sSetMaxY else '0';

   -- when a resolution switch occurs (even to the same resolution)
   -- leave the idle state
   -- if just powered up or reset occures go to reset state and
   -- from there set the position and bounds for the mouse
   manage_fsm: process(clk,rst)
   begin
      if(rst = '1') then
         state <= sReset;

      elsif(rising_edge(clk)) then

         case state is

            -- when reset occurs (or power-up) set the position
            -- and bounds for the mouse.
            when sReset =>
               state <= sSetX;

            -- remain in idle while switch is not active.
            when sIdle =>
               if(switch = '1') then
                  state <= sSetX;
               else
                  state <= sIdle;
               end if;

            when sSetX =>
               state <= sSetY;

            when sSetY =>
               state <= sSetMaxX;

            when sSetMaxX =>
               state <= sSetMaxY;
```

```
            when sSetMaxY =>
               state <= sIdle;

            when others =>
               state <= sIdle;
         end case;
      end if;
   end process;
end Behavioral;
```

۴۵- ارتباط سریال با کامپیوتر برای بورد آموزشی

پورت ۹ پینی سریال (COM) کامپیوتر را با یک کابل سریال کراس (کابلی که پایه‌ی ۲ را به ۳ و پایه‌ی ۳ را به ۲ به صورت ضربدری متصل می‌کند) به پورت سریال بورد متصل کنید (دقت کنید که بعضی از جک-های سریال کمی بزرگ هستند و ممکن است در حین اتصال به بورد فشار وارد کنند؛ در این موارد کمی با احتیاط عمل کنید). برنامه‌ی hyperterminal را که در ویندوز وجود دارد اجرا کنید و تنظیمات آن را روی باود ریت 1200bps با ۸ بیت داده و ۱ بیت stop و بدون parity ست کنید. :Flow control هم را در هر یک از حالت‌هایش قرار دهید (فرقی ندارد).

البته می‌توانید به جای hyperterminal از برنامه‌های دیگری هم استفاده کنید یا برنامه‌ی خودتان را با زبان‌هایی مثل VB و غیره بنویسید.

به محض اجرای برنامه، عبارت AmOL electronics (با یک enter قبل و یکی هم بعدش) به کامپیوتر فرستاده می‌شود (به صورت حرف حرف) و همزمان در سطر اول LCD، جلوی عبارت «T:» هم به صورت حرف حرف نشان داده می‌شود (کدهای مربوط به Enter شامل LF(Line Feed) و CR(Cariage Return) در LCD قابل نمایش نیستند و به جای آنها اشکال نامشخصی در LCD نشان داده می‌شود). پس از آن می‌توانید حروف دلخواه خود را با تنظیم کلیدهای on/off با کد اسکی کاراکتر مورد نظر و سپس فشردن کلید فشاری متصل به پایه‌ی 8 به کامپیوتر بفرستید. همزمان حرف ارسالی در LCD نشان داده می‌شود. (باز هم توجه کنید که بعضی از کدها در LCD قابل نمایش نیستند و برای آنها اشکال نامشخصی در LCD می‌بینید).

در همین حال حروف فشرده شده در کی‌بورد کامپیوتر هم به بورد فرستاده می‌شود و در سطر دوم LCD، جلوی عبارت «R:» نشان داده می‌شود.

همزمان در هشت LED سمت راست بورد، کد اسکی حرف دریافتی نشان داده می‌شود و در هشت LED سمت چپ هم کد حرف ارسالی نشان داده می‌شود. با هر تغییر در دریافت و ارسال هم LED های متناظر چشمک می‌زنند (یک بار خاموش و روشن می‌شوند).

کلید فشاری متصل به پایه‌ی 41، برای ریست می‌باشد که لزومی ندارد که در ابتدای کار حتماً فشرده شود. برای پالس ساعت هم از اسیلاتور 1^{MHz} روی بورد استفاده شده است. بازر بورد هم غیر فعال شده تا نویزهای موجود باعث تولید احتمالی صداهای ریز مزاحم نشوند.

برنامه:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity serial is
   Port (
                    --lcd--
                    data : out std_logic_vector(7 downto 0);
                    rs : out std_logic;
                    rw : out std_logic;
                    e : out std_logic;
                    bl : out std_logic;

                    --serial--
                    txd :   out std_logic;
                    rxd:   in std_logic;
                    leds :  out std_logic_vector(15 downto 0);
                    port_in :  in std_logic_vector(7 downto 0);
                    enter :  in std_logic;

                    --general--
                    clk_1M :      in std_logic;
                    reset:       in std_logic;
                    buzzer : out std_logic
      );
end serial;

architecture Behavioral of serial is
   constant delay_4000 : integer := 16384;
   constant data_show : integer := 8;    --number of data to be showed in LCD
   signal e1 : std_logic;

      constant baud : integer := 1200;          --baud
      constant crystal : integer := 1000000;   --crystal
      constant bit_time : integer := (crystal/baud)-1;
   signal lcd_show_receive : std_logic;
   signal receive_reg : std_logic_vector (9 downto 0);
   signal send_reg : std_logic_vector (10 downto 0);
   signal mask_send, mask_receive : std_logic_vector (7 downto 0);

      type state is (start, write, delay, stop);
   signal current, return_state : state;

      type state_receive is (start, receive_data, delay_receive);
   signal receive : state_receive;
```

```
      type state_send is (start, send_data, delay_send);
   signal send, return_send : state_send;

   type array_show is array (0 to data_show - 1)
                           of std_logic_vector (8 downto 0);
   constant table : array_show := (
               "000111000", "000001100", "000000001",
                           "101010011", "100111010", --S:
                           "011000000",
                           "101010010", "100111010" --R:
                       );
   type array_send is array (0 to 19)
                           of std_logic_vector (7 downto 0);
   constant send_table : array_send := (
                           "00001010", "00001101",
               "01000001", "01101101", "01001111", "01001100", --AmOL
               "00100000", "01100101", "01101100", "01100101", -- ele
                           "01100011", "01110100", "01110010", "01101111", --ctro

               "01101110", "01101001", "01100011", "01110011", --nics
                           "00001010", "00001101"
            );
begin

   process (clk_1M, reset)
            variable i, count1, count, count_receive, count2,
                            count_send, count3, count4 : integer;
            variable round_receive, lcd_show_receive,
                                    round_send, lcd_show_send, first_send :
std_logic;
   begin
            if reset = '1' then
            current <= start;
                rs <= '0';
                    rw <= '0';
                    e1 <= '1';
                    bl <= '1';
                    i := 0;
                    count1 := 0;
                    count2 := 0;
                    round_receive := '0';
                    lcd_show_receive := '0';
                    count_receive := 0;
                    receive <= start;
                    count3 := 0;
                    round_send := '0';
                    lcd_show_send := '0';
                    count_send := 0;
                    send <= start;
                    first_send := '0';
                    count4 := 0;
                    mask_send <= "11111111";
                    mask_receive <= "11111111";
                    buzzer <= '1';
          elsif clk_1M = '1' and clk_1M'event then
                case current is
                    when start =>
                   data <= table(i)(7 downto 0);
                           rs <= table(i)(8);
                           if i < data_show  then
                                   current <= write;
                                   count1 := delay_4000;
                                   i := i + 1;
                           else
                                   if lcd_show_receive = '1' then
```

```
                        if round_receive = '0' then
                        data <= "11000011";
                               rs <= '0';
                               current <= write;
                               count1 := delay_4000;
                               round_receive := '1';
                        else
                        data <= receive_reg(8 downto 1);
                               rs <= '1';
                               current <= write;
                               count1 := delay_4000;
                               round_receive := '0';
                               lcd_show_receive := '0';
                               mask_receive <= "11111111";
                        end if;
                  end if;
                  if lcd_show_send = '1' then
                        if round_send = '0' then
                        data <= "10000011";
                               rs <= '0';
                               current <= write;
                               count1 := delay_4000;
                               round_send := '1';
                        else
                        data <= send_reg(8 downto 1);
                               rs <= '1';
                               current <= write;
                               count1 := delay_4000;
                               round_send := '0';
                               lcd_show_send := '0';
                        end if;
                  end if;
end if;
 when write =>
       e1 <= not e1;
       count := count1;
           if e1 = '1' then
                  return_state <= write;
           else
                  return_state <= start;
           end if;
       current <= delay;
 when delay =>
           count := count - 1;
           if count = 0 then
           current <= return_state;
end if;
 when others =>
 end case;

    case  receive  is
    when start =>
           if(rxd='0')then
                  mask_receive <= "00000000";
                  receive <= delay_receive;
                  count2 := bit_time / 2;
           end if;
    when receive_data =>
           if (count_receive < 9) then
                  receive_reg (count_receive) <= rxd;  --receiving
                  count_receive := count_receive + 1;
                  receive <= delay_receive;
                  count2 := bit_time;
           else
                  count_receive := 0;
```

```
                                lcd_show_receive := '1';          --send char to
lcd
                                receive <= start;
                        end if;
                when delay_receive =>
                        if count2 = 0 then
                                receive <= receive_data;
                        else
                                count2 := count2 - 1;
                        end if;
                end case;

                case send is
                when start =>
                        mask_send <= "11111111";
                        if first_send = '0' then
                                if count4 <= 19 then
                                        send_reg <= "11" & send_table (count4) &
'0';
                                        send <= send_data;
                                        count4 := count4 + 1;
                                else
                                        first_send := '1';
                                end if;
                        elsif (enter='1')then
                                mask_send <= "00000000";
                                send_reg <= "11" & port_in & '0';
                                send <= send_data;
                        end if;
                when send_data =>
                        txd <= send_reg (count_send);  --sending
                        if (count_send = 10) then
                                count_send := 0;
                                count3 := 300000;
                                lcd_show_send := '1';          --send char to lcd
                                return_send <= start;
                                send <= delay_send;
                        else
                                count_send := count_send + 1;
                                count3 := bit_time;
                                return_send <= send_data;
                                send <= delay_send;
                        end if;
                when delay_send =>
                        if count3 = 0 then
                                send <= return_send;
                        else
                                count3 := count3 - 1;
                        end if;
                end case;

        end if;
    end process;
        e <= e1;
        leds(7 downto 0) <= receive_reg(8 downto 1) and mask_receive;
        leds(15 downto 8) <= send_reg(8 downto 1) and mask_send;

end Behavioral;
```

۴۶- اتصال تلویزیون برای بورد آموزشی

فیش زرد رنگ RCA تلویزیون (فیش Video in) را به پورت TV روی بورد وصل کنید. بلافاصله پس از اتصال در وسط تلویزیون عبارت AmOL *electronics* را روی زمینه‌ی مشکی می‌توانید مشاهده کنید (در این بورد تصاویر به صورت آنچه در عامه به سیاه و سفید مشهور است (به بیان دقیق‌تر Grayscale) نشان داده می‌شوند). در صورتی که کلید متصل به پایه‌ی 7، روی '0' باشد، با شش کلید on/off دیگر می‌توانید سطح خاکستری نوشته‌ی وسط صفحه را از مشکی کامل تا سفید کامل تغییر دهید. کلیدهای سمت چپ دارای ارزش بالاتر می‌باشند و سطح را بیشتر به سفید نزدیک می‌کنند. در صورتی که یک کلید فعال ('1') شود، همه‌ی کلیدهای سمت راست آن بی اثر می‌شوند. در نتیجه کلاً دارای هشت سطح خاکستری (منجمله مشکی و سفید) می‌باشیم.

در صورتی که کلید متصل به پایه‌ی 7، روی '1' باشد، عمل تغییر سطح خاکستری نوشته، بصورت اتوماتیک انجام می‌شود. بدین صورت که سطح خاکستری نوشته از سفید شروع شده (اگر بیش از یک کلید در حالت قبل از '1' کردن کلید پایه‌ی 7 فعال باشند یا هیچ کلیدی فعال نشده باشد و در غیر اینصورت از همان سطحی که بود شروع می‌شود) و به تدریج کمرنگ شده تا به نزدیکترین حد به مشکی برسد و سپس درجا به حالت سفید بر می‌گردد و این عمل را تکرار می‌کند.

LED آخر بورد (سمت چپ‌ترین) هم وضعیت کلید پایه‌ی 7 (اینکه تغییر سطح به صورت دستی '0' یا اتوماتیک '1' می‌باشد) را نشان می‌دهد و هفت LED قبل آن در هر لحظه سیگنال‌های دیجیتال ارسالی به تلویزیون را نشان می‌دهد.

کلید فشاری متصل به پایه‌ی 41، برای ریست می‌باشد که لزومی ندارد که در ابتدای کار حتماً فشرده شود.

برای پالس ساعت هم از اسیلاتور 50^{MHz} روی بورد استفاده شده است. بازر بورد هم غیر فعال شده تا نویزهای موجود باعث تولید احتمالی صداهای ریز مزاحم نشوند.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity tv is
   Port ( clk_50M : in std_logic;
          reset : in std_logic;
          level : out std_logic_vector(6 downto 0);
                    led : out std_logic_vector( 7 downto 0);
                    key_Auto:in std_logic;
                    key: in std_logic_vector(6 downto 0);
          level_0_3v : out std_logic;
                    buzzer : out std_logic);
end tv;

architecture Behavioral of tv is

      type tv_state is (start_page1, hsync_1, level_data1, end_page1, vsync_1,
                                          start_page2, page2_47us, hsync_2,
level_data2, vsync_2);
      signal type_level:std_logic_vector(6 downto 0);

      type pic_rom is array (0 to 9999) of std_logic;
      constant pic_matrix : pic_rom :=
('0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','1','1','1
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','1','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','1','1','0','1','1','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','1','1','1','0','0','0','0','0','0','1','1','1','0','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
```

,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','1','1','1','0','1','1','1','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','1','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','1','1','0','0','0','1','1','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','1','1','1','0','0','0','1','1','1','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','1','1','0','0','0','0','0','1','1','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','1','1','0','0','0','0','0','1','1','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','1','1','1','0','0','0','0','0','1','1','1','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',

```
       '0','0','0','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','1','1','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','1','1','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0'
,'1','1','1','0','0','1','1','1','0','0','0','0','1','1','0','0','0','0','0','0',
'0','1','1','1','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','0','1','1','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0'
,'1','1','1','0','1','1','1','1','1','0','0','1','1','1','1','1','0','0','0','0',
'0','0','1','1','1','0','0','0','0','0','0','1','1','1','0','0','0','0','0','1','
1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','1','1','1','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0'
,'1','1','1','1','1','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0',
'0','0','1','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','1','
1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','1','1','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','1','0','0','0','1','1','1','0','0','0',
'0','0','0','0','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','1','
1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
       '0','1','1','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','1','
1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
```

'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',

'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','
1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'1','1','1','0','0','0','0','1','1','1','0','0','0','0','1','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1'
,'1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1'
,'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','1','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','1'
,'1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','1','1'
,'1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0

','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','1','1','0','
0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0
','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','1','1','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','0',
'0','0','0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','0'
,'0','0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','1','1','0','0','0','1','1','1','0','0','0','0','0','0','0','1','1','1','0','
0','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','1','1','1','0
','0','0','1','1','0','0','0','0','0','1','1','0','0','0','0','0','1','1','1','0','0'
,'0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','1','1','1','1','1','0',
'0','0','1','1','1','0','0','0','0','1','1','1','1','0','0','0','0','0','1','1','1','
1','0','0','0','0','1','1','1','1','1','1','0','0','0','0','0','0','0','0','1','1
','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','1','1'
,'0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','0','1','1','1','0','0','0','0','0','1','1','1','0','
0','0','0','0','0','0','0','1','1','0','0','0','0','0','1','1','1','0','0','0','0
','0','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0'
,'0','0','0','0','0','0','0','1','1','1','0','0','0','0','1','1','0','0','0','0','0',
'0','1','1','1','0','0','0','0','0','0','1','1','1','1','1','1','1','1','1','1','
1','0','0','0','1','1','1','0','0','1','1','0','0','0','0','0','0','0','0','1','1
','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','0'
,'0','0','0','0','0','1','1','1','0','0','0','0','0','0','0','0','0','1','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','1','1','0','0','0','0','0','0','1','1','0','0','
0','0','0','0','0','0','1','1','1','0','0','0','0','1','1','0','0','0','0','0
','1','1','1','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0'
,'0','0','0','0','0','0','0','0','1','1','1','0','0','1','1','0','0','0','0','0',
'1','1','1','0','0','0','0','0','0','0','0','0','1','1','0','1','1','1','1','1','1','
1','0','0','1','1','1','0','0','0','1','1','0','0','0','0','0','0','0','1','1
','1','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0'
,'0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','0','0','0','0','0','1','1','1','1','0','0','0','0','0','1','1','1','1','0','0','
0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0','0
','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','1','
1','0','0','1','1','0','0','0','0','1','1','0','0','0','0','0','0','0','1','1
','1','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0'
,'0','0','0','0','1','0','1','1','1','0','0','0','0','0','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','0','0','0','0','1','1','0','0','0','0','0','0','1','1','1','1','0','0','
0','0','0','0','0','1','1','1','0','0','0','0','1','1','0','0','0','0','0','0
','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','1','
1','0','0','1','1','0','0','0','0','1','1','0','0','0','0','0','0','0','1','1
','1','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0'
,'0','0','0','0','0','1','0','1','1','1','0','0','0','0','0','1','1','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','0','0','0','0','1','1','0','0','0','0','0','0','1','1','1','1','0','0','
0','0','0','0','0','1','1','1','0','0','0','0','1','1','0','0','0','0','0','0
','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','0','0','0','0','1','1','0','0','0','0','0','0','1','1','1','1','0','0','
0','0','0','0','0','1','1','1','0','0','0','0','1','1','0','0','0','0','0','0','1

```
','1','1','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','0','0'
,'0','0','0','0','0','0','0','0','1','1','0','0','1','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0','0','1','
1','0','1','1','0','0','0','0','0','1','1','1','0','0','0','0','0','0','1','1','1
','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0','0','0','0'
,'0','0','0','1','0','0','0','1','1','0','0','0','0','1','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1',
'1','1','1','1','1','1','1','1','0','0','0','0','0','0','1','1','1','1','1','0','0','
0','0','0','0','1','1','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','1','1
','1','1','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','0','0'
,'0','0','0','0','0','0','0','0','1','1','0','1','0','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','1','1','
1','1','1','0','0','0','0','0','0','0','1','1','1','1','1','1','1','1','1','1','1
','0','0','0','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0'
,'0','0','1','1','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','0','0','0','0','0','1','1','0','1','1','1','0','0','
0','0','0','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','0','1','1','1
','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','
0','0','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','1','1','0','0','0
','1','1','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0','1','1','1','0','0'
,'0','0','0','1','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','0','1','1','
1','1','0','0','0','0','0','0','0','1','1','1','0','0','0','1','1','1','1','1','1','1
','0','0','0','0','0','0','1','1','0','0','1','1','0','0','0','0','0','0','0','0','0'
,'1','1','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1',
'1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','1','1','1','
0','0','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0
','1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','1','1','1','0','0'
,'0','0','0','1','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','1','1','
1','1','0','0','0','0','0','0','0','0','1','1','1','1','1','1','1','1','1','1','1
','0','0','0','0','0','0','0','0','1','1','0','0','1','1','0','0','0','0','0','0','0'
,'1','1','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1',
'1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','1','1','1','
0','0','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0
','1','1','0','0','0','0','0','0','0','0','0','0','0','1','1','0','0','1','1','1','0','0'
,'0','0','0','1','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0','1',
'1','1','0','0','0','0','0','0','0','0','0','0','1','1','0','0','0','0','0','0','1','1','
1','1','0','0','0','0','0','0','0','1','1','1','1','1','1','1','1','1','1','1','1
','0','0','0','0','0','0','1','1','0','0','1','1','1','0','0','0','0','0','0','0','1'
,'1','1','0','0','0','0','1','1','1','0','1','1','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','1','0','0','0','0','0','1','1','1','1','0','0','0','0','1','1','1','1','1','
1','1','1','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','1','1','1','0','0
','1','1','1','1','1','1','0','0','0','0','1','1','1','1','0','0','0','0','1','1','1','1','1'
,'1','1','1','1','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0',
'1','1','1','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0','0','0','0','1','1','
1','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','1','1','1','1','0','0','1','1','1
','0','0','0','1','1','1','0','0','0','0','1','1','1','1','1','1','0','0','0','0','1','1','1'
,'0','0','0','0','0','1','1','1','0','0','0','1','1','1','0','0','0','0','0','0','0','0','0',
       '0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','1','1','1','1','1','
1','1','0','0','0','0','0','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0
','0','1','1','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','1','1','1','1'
,'1','1','0','0','0','0','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','1','1','0','0','0','0','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1','
0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','1','1','1
','1','1','1','1','1','0','0','0','0','0','0','1','1','1','1','1','1','1','1','1','0','0'
,'0','0','0','1','1','0','0','0','0','1','1','1','0','0','0','0','0','0','0','0',
```

'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1','
0','0','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0','0','0','0','0
','0','0','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','1','1','1'
,'0','0','0','0','0','0','1','1','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','1','1','1','1','1','0','0','0','0','0','0','0','0','0','0','0','1','1','
0','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','0','0','0','0','1
','1','1','0','0','0','0','0','0','0','0','0','1','1','1','1','1','0','0','0','0'
,'0','0','0','1','1','0','0','0','1','1','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','1','1','1','1','1','1','1','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',

```
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','
0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'
,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0');

begin

	process(clk_50M,reset)
		variable state: tv_state;
		variable line_counter, count, pixel_counter,
				counter2, x, secend: integer RANGE 0 TO 20000;
		variable c: integer ;
	begin
		if reset = '1' then
			state := start_page1;
			count := 0;
			buzzer <= '1';
		elsif  (clk_50M='1' and clk_50M'event)then
			count := count + 1;
			case state is
			when start_page1 =>   --4.7us
				if count < 235 then
					state := start_page1;
					level_0_3v <= '0';
					level <= "0000000";
				elsif count = 235 then
					state := hsync_1;
				end if;
			when hsync_1 =>   --5.8us
				if count > 235 and count < 525 then
					state := hsync_1;
					level_0_3v <= '1';
					level <= "0000000";
				elsif count = 525 then
					state := level_data1;
					count := 0;
				end if;
			when level_data1 =>   --53.5us
				if count > 0 and count < 2675 then
					state := level_data2;
					level_0_3v <= '1';
				elsif count = 3200 then
					line_counter := line_counter + 1;
					level  <= "0000000";
					level_0_3v <= '1';
					if line_counter < 314 then
						state  :=     start_page1;
						count := 0;
					elsif line_counter = 314 then
						state  :=     end_page1;
						count  :=     0;
						line_counter := 0;
					end if;
				end if;
			when end_page1 =>
				if count < 1600 then
					state  :=     end_page1;
					level_0_3v <= '1';
					level  <=     "0000000";
				 elsif count = 1600 then
					state := vsync_1;
					level_0_3v <= '0';
					level <= "0000000";
					count  :=     0;
				 end if;
			when vsync_1 =>
```

```
		if count < 8000 then
			state := vsync_1;
			level_0_3v <= '0';
			level <= "0000000";
		elsif count = 8000 then
			state :=	start_page2;
			level_0_3v <= '1';
			level <= "0000000";
			count := 0;
		end if;
	when start_page2 =>
		if count < 1600 then
			state  :=	start_page2;
			level_0_3v	<=	'1';
			level  <=	"0000000";
		elsif count = 1600 then
			state  :=	page2_47us;
			level_0_3v <= '0';
			level  <=	"0000000";
			count := 0;
		end if;
	when page2_47us =>
		if count < 235 then
			state  :=	page2_47us;
			level_0_3v <= '0';
			level  <=	"0000000";
		elsif count = 235 then
			state  :=	hsync_2;
			level_0_3v <= '1';
			level  <=	"0000000";
			count  :=	0;
		end if;
	when hsync_2 =>
		if count < 290 then
			state  :=	hsync_2;
			level_0_3v <= '1';
			level  <=	"0000000";
		elsif count = 290 then
			state  :=	level_data2;
			level_0_3v <= '1';
			level  <=	"0000000";
			count  :=	0;
		end if;
	when level_data2 =>
		if count > 0 and count < 2675 then
			state := level_data2;
			level_0_3v <= '1';
			counter2 := counter2 + 1;
		   if counter2 = 4 then
			counter2 := 0;
				pixel_counter := pixel_counter + 1;
				if line_counter > 128 and line_counter <
179 and
				pixel_counter > 210 and pixel_counter <
411 then
					if x < 9999 then
						x := x + 1;
					else
						x := 0;
					end if;
					if pic_matrix(x) = '1' then
						level <= type_level;
					else
						level <= "0000000";
					end if;
				end if;
```

```
                                end if;
                            elsif count = 2675 then
                                pixel_counter := 0;
                                line_counter := line_counter + 1;
                                level  <= "0000000";
                                level_0_3v <= '1';
                                if line_counter < 314 then
                                    state  :=    page2_47us;
                                    count := 0;
                                elsif  line_counter = 314 then
                                    state := vsync_2;
                                    count := 0;
                                    line_counter := 0;
                                end if;
                            end if;
                        when vsync_2 =>
                            if count < 8000 then
                                state := vsync_2;
                                level_0_3v <= '0';
                                level <= "0000000";
                            elsif count = 8000 then
                                state :=     start_page1;
                                level_0_3v <= '1';
                                level <= "0000000";
                                count := 0;
                            end if;
                        end case;
                        c :=c+1;
        if c = 50000000 then
                        c := 0;
                        if key_Auto ='1' then
                                if type_level = "1000000" or type_level =
"0100000"
                                        or type_level = "0010000" or
type_level = "0001000"
                                        or type_level = "0000100" or
type_level = "0000010"
                                        or type_level = "0000001"  then
                                    type_level <= type_level(0) &
type_level(6 downto 1);
                                else
                                    type_level <= "1000000";
                                end if;
                            else
                                type_level <= key;
                            end if;
        end if;
                        led(6 downto 0)<=   type_level;
                        led(7)<= key_Auto;
                end if;
        end process;

end Behavioral;
```