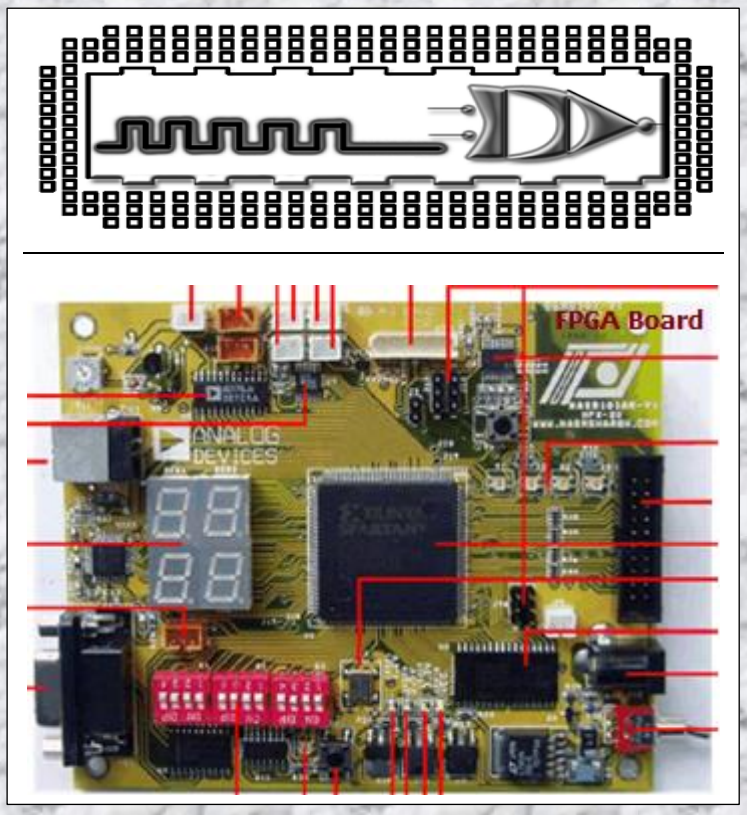
به نام هستی‌بخش

وزارت علوم تحقیقات وفناوری

دانشگاه صنعتی سجاد

دستور کار

آزمایشگاه مدار منطقی



نگارش:

امیر باوفای طوسی

شعله هاشمی نمین

ژیلا عظیم زاده

مینا عاشوری

شرکت نوآوران صنعت رایانه شرق

ویرایش اول-تابستان ۱۳۹۷

فهرست

مراجع:

۱- دستور کار آزمایشگاه مدار منطقی و سیستم دیجیتال دانشگاه‌های شریف، تهران، امیرکبیر، شهید رجایی، علم و صنعت، کاشان

۲- دستور کار آزمایشگاه مدار منطقی شرکت نصر شرق

۳- دستور کار آزمایشگاه مدار منطقی دانشگاه سجاد – نوید ایزدخواستی

۴- کتاب مدارهای منطقی نلسون

پیشگفتار

آزمایشگاه مدار منطقی با هدف ایجاد زیرساخت لازم جهت خودکارسازی طراحی مای الکترونیکی، آسان نمودن طراحی مای پیشرفته دیجیتال و ساخت نمونه مای آزمایشگاهی ارائه‌شده است. در این میان طراح نیز باید نسبت به مفاهیم پایه آگاهی و دانش کامل داشته باشد. به همین منظور در این آزمایشگاه سعی شده است تا این مفاهیم با دیدی آموزشی به بهترین نحو ممکن انتقال داده شود. این دستور کار به دو بخش تقسیم‌شده است :

- کار با برد بورد و تراشه‌های منطقی و نرم‌افزارهای طراحی PCB (6 جلسه)،
- کار با بوردهای FPGA شرکت Xilinx (۱۰ هفته)
- و در این راستا مفاهیم ذیل به‌صورت عملی فراگرفته می‌شود:

- ✓ آشنایی با برد بورد و قطعات منطقی و اصول طراحی بوردهای مدار چاپی و آشنایی با ساختار تراشه‌های قابل‌برنامه‌ریزی FPGA
- ✓ روند طراحی با یک تراشه قابل‌برنامه‌ریزی FPGA
- ✓ آشنایی و برنامه‌نویسی با زبان توصیف سخت‌افزار Verilog
- ✓ آشنایی و طراحی با نرم‌افزار ISE
- ✓ طراحی، پیاده‌سازی و آزمودن انواع مدارهای دیجیتال بر روی FPGA

شرکت نصر شرق نیز با در نظر گرفتن جایگاه مبحث طراحی‌های دیجیتال در دانشگاه‌های معتبر جهان و تصمیم بر ایجاد زیرساخت لازم به‌منظور توسعه این فنّاوری در سطح دانشگاه‌های داخل و ایجاد زمینه‌های شکوفایی و باروری دانشجویان ایرانی، مجموعه آزمایشگاهی مدار منطقی را ارائه نموده است. این مجموعه آزمایشگاهی شامل بردهایی مبتنی بر تراشه‌های قابل‌برنامه‌ریزی شرکت XILINX و بردهای برنامه‌ریزی آن‌ها، دستور کار آزمایشگاه مدار منطقی و نرم‌افزارهای موردنیاز است. شایان‌ذکر است دانشگاه‌های معتبری همچون شریف و امیرکبیر از این مجموعه استفاده می‌نمایند. *برای کسب اطلاعات بیشتر درزمینه‌های* مختلف و موردبحث در این آزمایشگاه می‌توانند به سایت شرکت نصر شرق[1] مراجعه نمایند.

ساختار دستور کار آزمایشگاه

به‌طورکلی هر آزمایش در این دستور کار دارای بخش‌های زیر می‌باشد:

- ✓ اهداف آزمایش
- ✓ تئوری آزمایش

1- *WWW.NASRSHARGH.COM*

- ✓ تکالیف پیش از آزمایش
- ✓ تکالیف داخل آزمایشگاه
- ✓ پروژه‌های پیشنهادی

در بخش اهداف آزمایش سعی شده است تا اهداف کلی از انجام هر آزمایش بیان شود تا دانشجویان در راستای آن اهداف سعی و تلاش نمایند. تئوری آزمایش حاوی اطلاعات کلی اما هدفمند درزمینهٔ آزمایش بوده و به دانشجویان در انجام آزمایش کمک می‌کند. تکالیف پیش از آزمایش شامل مطالب و کارهایی است که دانشجویان می‌بایست پیش از ورود به آزمایشگاه مطالعه و انجام دهند. نتایج حاصل از این بخش که می‌تواند متشکل از تشریح مدار موردنظر، مدار طراحی‌شده و نتایج تحلیل‌ها باشد می‌بایست در گزارش کار ثبت گردد. بخش تکالیف داخل آزمایشگاه بیانگر کلیه فعالیت‌هایی است که دانشجویان عزیز باید در طول مدت آزمایشگاه انجام دهند.

ارزیابی

کار کلاسی (انجام آزمایش‌های هفتگی و گزارش کار) ۱۰ نمره

امتحان پایان‌ترم ۱۰ نمره

جلسه ۱

آشنایی با تراشه‌های منطقی ، لوازم آزمایشگاهی و نرم‌افزار Altium Designer

در این بخش سعی داریم تا در ابتدا دانشجویان عزیز را با انواع قطعات منطقی، ساختار و مشخصات آن‌ها آشنا نماییم و سپس به بررسی مدارهای قابل‌برنامه‌ریزی که در این آزمایشگاه استفاده خواهند نمود، بپردازیم. ازآنجایی‌که در این آزمایشگاه هدف به‌کارگیری مطالبی است که درس مدار منطقی و معماری آموختید، بنابراین در بازگو نمودن جزئیات علمی خودداری و به توضیحی اجمالی بسنده خواهیم نمود.

۱) تراشه‌های منطقی

همان‌طور که می‌دانید قطعات منطقی که گیت /تراشه/دروازه خوانده می‌شوند، اصلی‌ترین قسمت یک مدار منطقی هستند. این قطعات به‌منظور انجام اعمال پایه‌ای نظیر AND، OR ، NOT، ... و یا اعمال پیچیده‌تری نظیر شمارنده‌ها، ثبات‌ها و ... ساخته و در بسته‌بندی-های مختلف به بازار عرضه می‌شوند. اما این قطعات منطقی به روش‌های مختلفی ساخته می‌شوند که یکی از این روش‌ها استفاده از مدارهای مجتمع می‌باشد. هر مدار مجتمع یا آی‌سی دارای یک مشخصه عددی است که روی سطح بسته‌بندی آن و به‌منظور بیان مشخصات و پارامترهای آن چاپ می‌شود. لازم به ذکر است که تمام سازنده‌ها کتابچه راهنما یا کاتالوگ حاوی شرح دقیق و تمام اطلاعات لازم درباره آی‌سی‌های ساخت خود را چاپ می‌کند.

۱-۱) فنّاوری ساخت

با پیشرفت فنّاوری مدارهای ساخت مجتمع تعداد گیت‌هایی که می‌توانست در یک تراشه جای گیرد به میزان قابل‌توجهی افزایش یافت. مدارهای مجتمع با مقیاس کوچک (SSI) دارای چند گیت مستقل در یک بسته واحد هستند. تعداد این گیت‌ها معمولاً کمتر از ۱۰ و محدود به تعداد پایه‌ها در آی‌سی است. قطعات مجتمع با مقیاس متوسط (MSI) تقریباً دارای ۱۰ الی ۲۰۰ گیت در هر بسته می‌باشند. این قطعات معمولاً توابع دیجیتال ساده همچون دکترها، جمع کننده‌ها و ثبات‌ها را اجرا می‌نمایند. مدارهای مجتمع با مقیاس بزرگ (LSI) بین ۲۰۰ تا چند هزار گیت در هر بسته دارند. این بسته‌ها دستگاه‌های دیجیتالی همچون پردازنده‌ها، تراشه‌های حافظه و ماژول‌های قابل‌برنامه‌ریزی را شامل می‌شوند. قطعات مجتمع با مقیاس بسیار بزرگ (VLSI) حاوی هزاران گیت در یک بسته-اند. VLSI ها به دلیل کوچکی و ارزانی انقلابی در فنّاوری ساخت سیستم‌ها کامپیوتری به وجود آورده و به طراحان امکان ساخت و ایجاد ساختارهایی را دادند که قبلاً اقتصادی نبودند.

مدارهای مجتمع نه‌تنها بر اساس عملکرد منطقی‌شان بلکه ازنظر فنّاوری خاص مدارهایی که به آن تعلق دارند نیز طبقه‌بندی می‌شوند. فنّاوری به‌کاررفته در ساخت این مدارها را خانواده قطعات منطقی می‌خوانند. بسیاری از این خانواده‌ها به‌صورت مدارهای مجتمع در سطح تجاری عرضه‌شده‌اند. متداول‌ترین خانواده‌ها در زیر معرفی‌شده‌اند:

- ✓ TTL یا منطق ترانزیستور- ترانزیستور

- ✓ ECL یا منطق کوپل امیتر
- ✓ MOS یا منطق فلز اکسید نیمه‌هادی
- ✓ CMOS یا منطق فلز اکسید نیمه‌هادی مکمل

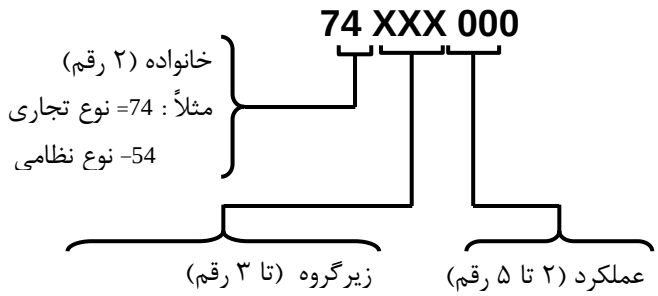
TTL یک خانواده متداول است که سال‌ها مورداستفاده بوده و به‌عنوان استاندارد تلقی می‌شود. ECL در سیستم‌هایی که به‌سرعت عمل بالا نیاز دارند ترجیح داده می‌شوند. MOS برای مدارهایی که نیاز به تراکم بالا دارند مناسب است و CMOS در سیستم‌های کم‌مصرف به کار می‌رود.

خانواده منطقی ترانزیستور- ترانزیستور گونه تکامل‌یافته فنّاوری قدیمی‌تری است که در آن از دیود و ترانزیستور برای ساخت گیت پایه NAND استفاده می‌شده است. بعدها برای بهبود عملکرد مدار به‌جای دیود از ترانزیستور استفاده شد و نام خانواده جدید ترانزیستور- ترانزیستور گذاشته شد. علاوه بر نوع استاندارد TTL انواع دیگری از این خانواده عبارت‌اند از TTL سرعت‌بالا، TTL توان پایین (یا کم‌مصرف)، TTL شاتکی، TTL شاتکی توان پایین و ... که در انتهای این بخش در مورد آن توضیحات بیشتری را ارائه خواهیم نمود.

منطق فلز اکسید نیمه‌هادی یک ترانزیستور تک‌قطبی است که به جریان یک نوع حامل الکتریکی وابسته است. این حامل‌ها ممکن است الکترون (در نوع کانال n) یا حفره باشند. MOS کانال p را PMOS و MOS کانال n را NMOS می‌نامند. در فنّاوری CMOS هر دو نوع ترانزیستور، که به شکل مکمل در تمام مدارها بسته‌شده‌اند به‌کاررفته است. بزرگ‌ترین مزیت CMOS نسبت به دوقطبی، تراکم بالای مدارها ، ساده بودن تکنیک ساخت و عملکرد مقرون‌به‌صرفه آن به دلیل مصرف توان کم آن است.

۱-۲) آی‌سی‌های دیجیتال سری 74

سال‌های متمادی است که ۸ زیرگروه آی‌سی‌های TTL و ۸ زیرگروه آی‌سی‌های CMOS در سری ۷۴ قرار دارند. بیشتر این زیرگروه‌ها دیگر قدیمی بشمار آمده یا منسوخ‌شده‌اند. اما توجه داشته باشید که بیشتر زیرگروه با یکدیگر سازگار هستند. بنابراین به‌سادگی و با توجه به مشخصات قطعه موردنظر که از روی شماره آن قابل‌استخراج است، می‌توان آن‌ها را با قطعه از خانواده جدیدتر جایگزین کرد. با این تفاسیر در ابتدا می‌بایست نحوه تعیین مشخصات قطعه از روی شماره آن را فرابگیریم. در شکل زیر طرح‌های اساسی مورداستفاده برای مشخص کردن کدهای سری ۷۴ را نشان می‌دهد.



مثلاً : TTL=N استاندارد

TTL-L کم‌مصرف

TTL=S شاتکی

TTL=LS شاتکی کم‌مصرف

TTL =AS شاتکی پیشرفته

TTL=ALS شاتکی کم‌مصرف پیشرفته

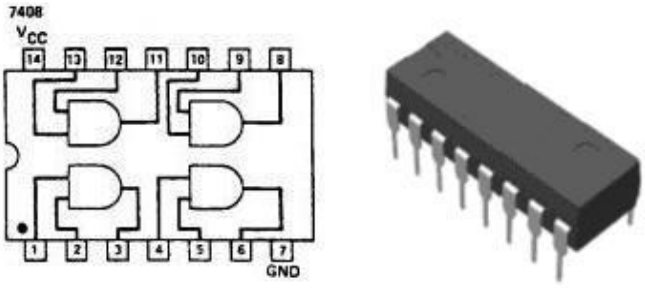
TTL=F سریع

C= نوع CMOS آی‌سی‌های TTL

HC= آی‌سی‌های CMOS سریع همراه با ورودی‌های

شکل۱-۱: روش پایه‌ای کددهی در آی‌سی‌های سری ۷۴

همان‌طور که در شکل نیز مشاهده می‌نمایید، در ساده‌ترین حالت کدهای حرفی- عددی آی‌سی‌ها از ۳ کد فرعی مسلسل تشکیل‌شده است. اولین کد فرعی عددی دورقمی است، که مشخص‌کننده حوزه کاربردی آی‌سی است و شامل کاربردهای تجاری (74)، نظامی (54) و قطعات رابط (75) می‌باشد. کد دوم حداکثر می‌تواند از ۳ حرف تشکیل‌شده باشد. این کد فنّاوری مورداستفاده در آی‌سی را مشخص می‌نماید.

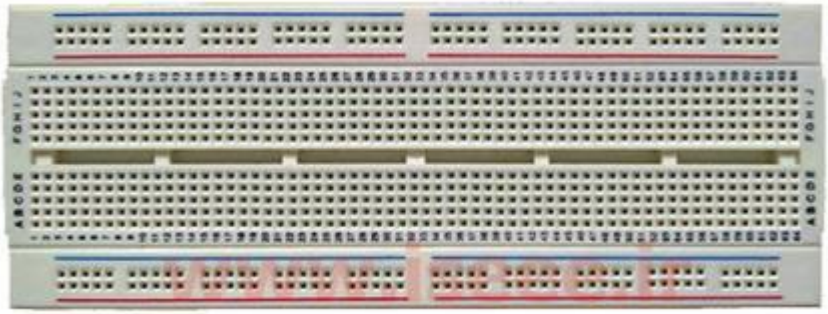


شکل ۱-۲: بسته‌بندی فیزیکی قطعات

کد سوم که عدد ۲ تا ۵ رقمی می‌باشد، بیانگر عملکرد آی‌سی است. بنابراین کد به‌کاررفته در آی‌سی‌ها سری ۷۴ می‌توانند شامل7414، 74HC03 و 74LS38 باشد. در انتها نیز در شکل فوق نمونه‌ای از بسته‌بندی این قطعات آر مشاهده می‌نمایید.

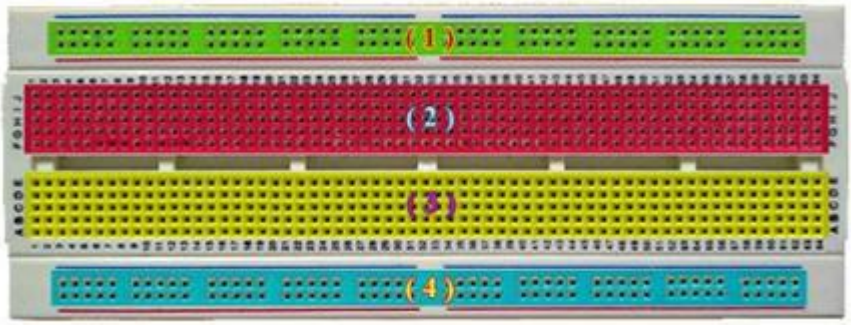
- برد بورد (Bread board):

به‌منظور اینکه مداری را به‌طور موقت بسته و مورد آزمایش قرار دهیم از برد بورد (Bread board) استفاده می‌کنیم. استفاده از برد بورد سرعت کار را افزایش داده و بستن مدار را بسیار آسان می‌کند. در شکل زیر تصویری از یک برد بورد نمایش داده‌شده است.



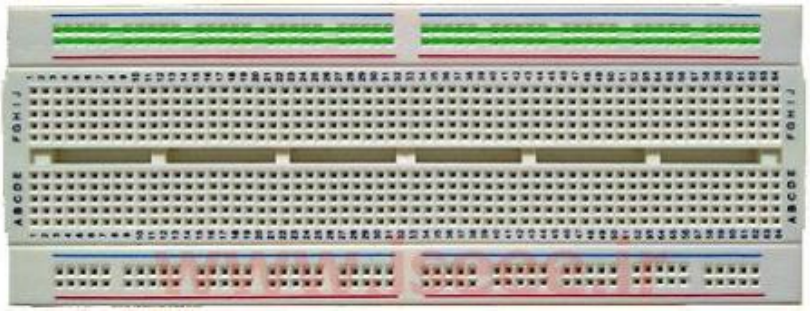
شکل ۱-۳: نمونه‌ای از یک برد بورد

همان‌طور که در این شکل مشاهده می‌کنید برد بورد دارای سوراخ‌های بسیار است که پایه‌های قطعات الکترونیکی داخل این سوراخ‌ها قرار می‌گیرد. سطح یک برد بورد را می‌توان به چهار قسمت تقسیم کرد. این چهار قسمت در شکل زیر با رنگ‌های مختلف و اعداد ۱ تا ۴ نمایش داده‌شده‌اند.



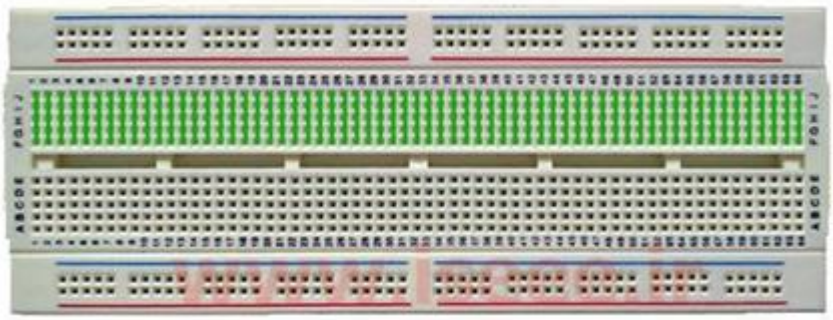
شکل ۱-۴: تقسیم‌بندی برد بورد به چهار قسمت

سوراخ‌های قسمت اول، خود به چهار قسمت تقسیم می‌شوند که سوراخ‌های هر قسمت در یک ردیف قرارگرفته و از داخل برد بورد به یکدیگر متصل شده‌اند. این سوراخ‌ها معمولاً جهت اتصال قطب‌های منبع تغذیه به مدار مورداستفاده قرار می‌گیرند. در شکل بعدی سوراخ‌هایی که در قسمت اول قرار دارند و از داخل به یکدیگر متصل هستند توسط خطوط سبزرنگی به هم وصل شده‌اند.



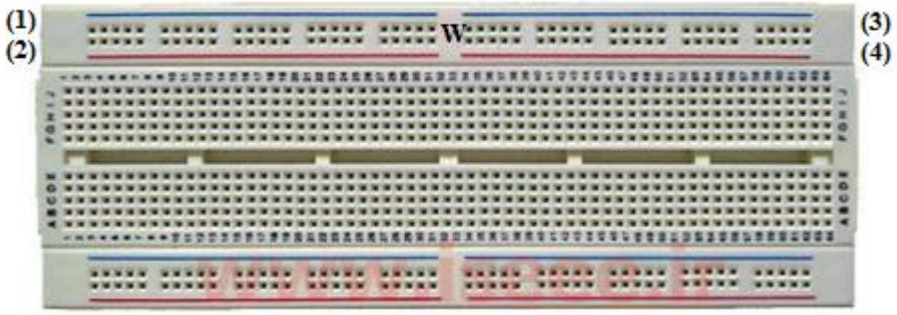
شکل ۱-۵: اتصال افقی سوراخ‌های برد بورد از داخل

در قسمت دوم، تمام سوراخ‌هایی که در یک ستون قرار دارند از داخل به یکدیگر متصل هستند. در این قسمت همان‌طور که مشاهده می‌کنید در هر ستون پنج سوراخ وجود دارد. تمامی این پنج سوراخ از داخل به یکدیگر متصل هستند. در شکل بعدی سوراخ‌هایی که در قسمت دوم قرار دارند و از داخل به یکدیگر متصل هستند توسط خطوط سبزرنگی به هم وصل شده‌اند.



شکل ۱-۶: اتصال عمودی سوراخ‌های برد بورد از داخل

قسمت سوم دقیقاً مشابه قسمت دوم می‌باشد و قسمت چهارم نیز دقیقاً مشابه قسمت اول می‌باشد. دقت داشته باشید که هرگز نباید هر دو پایه یک المان الکترونیکی را در سوراخ‌هایی که از داخل به هم متصل هستند قرارداد زیرا در این صورت آن المان عملاً از مدار حذف می‌شود.



شکل ۱-۷: اتصال عمودی سوراخ‌های برد بورد از داخل

به‌بیان‌دیگر در برد بورد فوق قسمت‌های ۱، ۲، ۳ و ۴ به‌صورت افقی به هم متصل بوده و اتصال عمودی ندارند و به‌وسیله حرف W نوشته‌شده روی برد بورد قسمت ۱و۲ از قسمت ۳ و۴ جدا می‌شوند. از قسمت افقی بیشتر برای تغذیه و زمین مدار استفاده می‌شود. حروف D,C,B,A وE به‌صورت عمودی به هم متصل بوده و اتصال افقی ندارند.

- پروب منطقی (Logic Prob)

از این وسیله برای بررسی ولتاژ نقطه‌ای از مدار استفاده می‌شود به‌گونه‌ای که با قرار دادن پروب منطقی در نقطه‌ای از مدار، در صورت یک بودن ولتاژ آن نقطه چراغ پروب منطقی روشن می‌ماند و در صورت صفر بودن چراغ آن خاموش می‌شود.

- منبع تغذیه:

مدارهایی که در این آزمایشگاه پیاده‌سازی می‌شوند با ولتاژ ثابت 5^{v} کار می‌کنند. به همین خاطر برای تأمین ولتاژ موردنیاز مدارها از دستگاهی (منبع تغذیه دوبل) استفاده می‌شود که ولتاژ ثابت 5^{v} را تولید می‌کند و دارای دو پایانه مثبت و منفی است که از پایانه مثبت برای Vcc یا ولتاژ بالا و از پایانه منفی به‌منظور زمین (GND) مدار استفاده می‌شود.

- سیگنال ژنراتور:

برای تولید کلاک و فرکانس‌های موردنیاز از این دستگاه استفاده می‌شود که انواع سیگنال‌های سینوسی، مثلثی و مربعی و خروجی TTL را تولید نموده و می‌توان فرکانس موردنظر را در یک محدوده فرکانس از 0.1 هرتز تا یک مگاهرتز تنظیم نمود. در بخش طراحی و پیاده‌سازی مدارهای ترتیبی برای تولید کلاک از سیگنالی با فرکانس کم استفاده می‌شود که توسط این دستگاه تولید می‌شود.

- اسیلوسکوپ:

دستگاهی است برای بررسی هر نوع پدیده متغیر قابل‌تبدیل به جریان و ولتاژ. از اسیلوسکوپ برای اندازه‌گیری‌های مختلفی مانند ولتاژ AC یا DC ، مقدار فرکانس، مقادیر پیک ولتاژ و غیره استفاده می‌شود.

نکات قابل‌توجه:

۱- خانواده TTL با منبع ولتاژ ۵ ولت و CMOS با ۳ تا ۱۵ ولت و معمولاً ۵ ولت کار می‌کند.

۲- خروجی تراشه را مستقیماً به زمین یا منبع ولتاژ متصل نکنید.

۳- LED را مستقیماً به خروجی TTL وصل نکنید و از یک مقاومت حدود ۲۰۰^{Ω} (اهم) به‌صورت سری استفاده کنید.

۴- سر مثبت منبع تغذیه ۵ ولت را به یکی از سوراخ‌های بالاترین ردیف صفحه آزمایش متصل نمایید و سر منفی منبع تغذیه را به یکی از سوراخ‌های پایین‌ترین ردیف صفحه آزمایش متصل کنید ازاین‌پس ردیف بالای صفحه آزمایش ردیف ۵ ولت و ردیف پایین ردیف زمین (۰ ولت) نامیده خواهد شد. اکنون سر شماره ۷ تراشه ۷۴۰۰ را به‌وسیله سیم به‌ردیف زمین متصل نمایید. این کار بااتصال دادن یکی از سوراخ‌های ستون زیر سر شماره ۷ به‌ردیف زمین انجام می‌شود. پایه شماره ۱۴ تراشه ۷۴۰۰ را به‌وسیله سیم به‌ردیف ۵ ولت متصل نمایید. اکنون تراشه به‌طور صحیح تغذیه‌شده است.

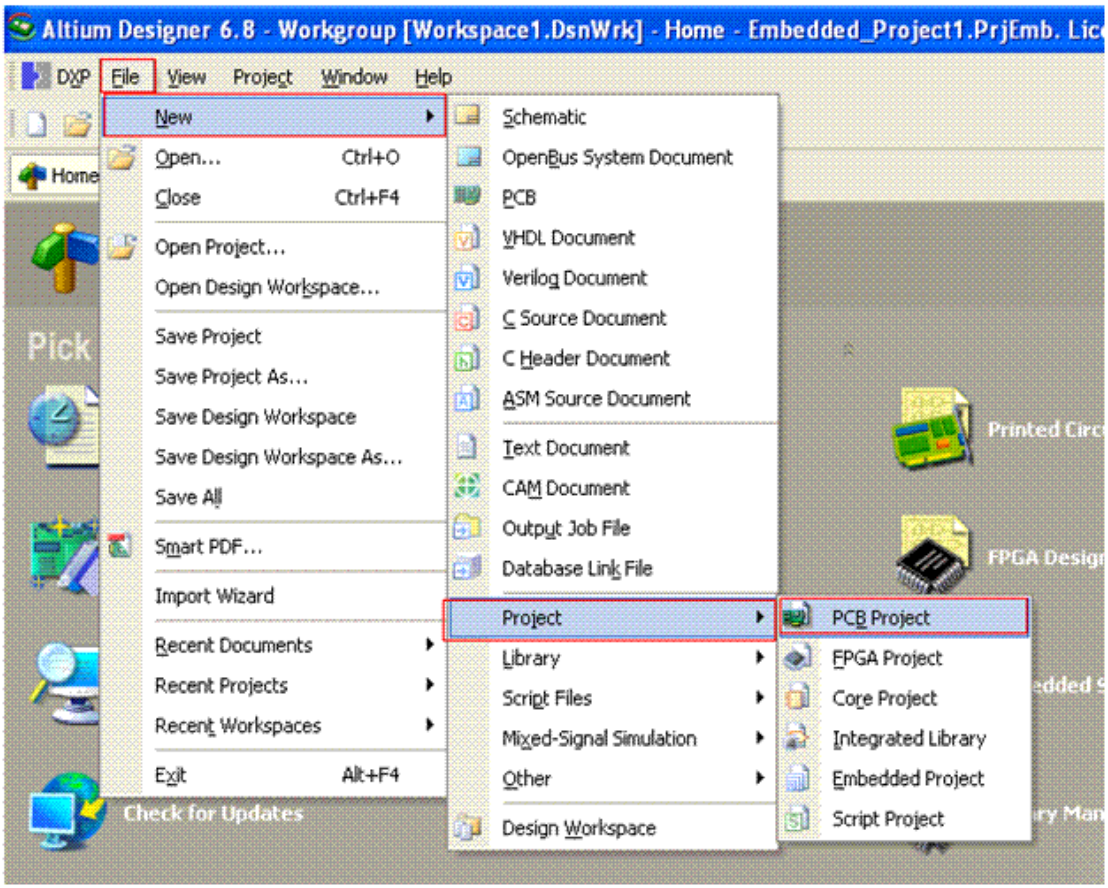
ولتاژ ورودی صفر و ۵ ولت بااتصال دادن پایه شماره ۱ به‌ردیف زمین یا ردیف ۵ ولت به دست می‌آید و ولتاژ خروجی (پایه شماره ۲) توسط پروب منطقی (Logic Prob) مشخص می‌شود.

۵- حالت خروجی یک مدار را می‌توان به کمک دیود نورانی «LED» مشاهده نمود. برای این کار خروجی مدار را توسط یک مقاومت به LED متصل کنید (برای این کار خروجی را توسط یک مقاومت به آند LED «معمولاً پایه‌ی بلندتر» وصل و کاتد آن «معمولاً پایه کوتاه‌تر» را به زمین متصل می‌کنیم). مقاومت به‌منظور محدود کردن شدت‌جریان و جلوگیری از سوختن دیود و تراشه به‌کاررفته

است و مقدار آن حدود ۱۰۰ تا ۳۰۰ اهم می‌باشد. ورودی‌های مدار نبایست بازبماند. با توجه به نوع تراشه به ۰ و یا یک منطقی متصل نمایید.

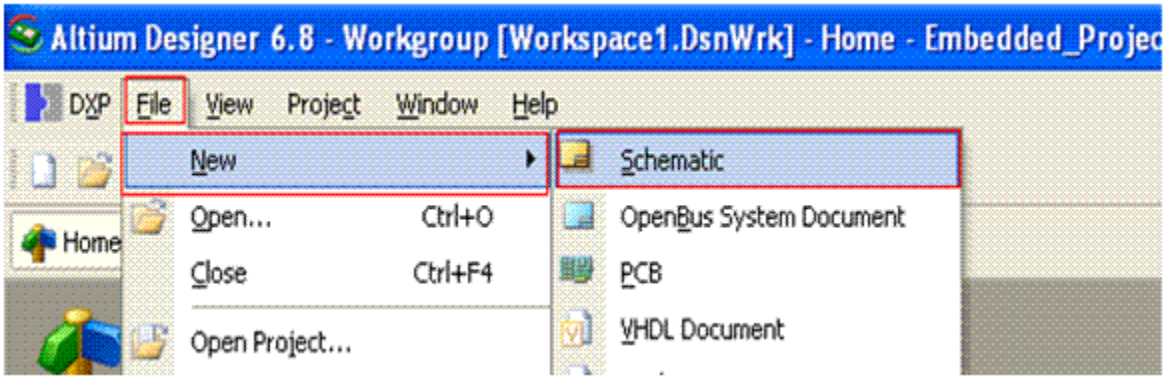
آشنایی با نرم‌افزار Altium Designer

Altium Designer یک نرم‌افزار پرقدرت برای طراحی مدار چاپی (PCB) می‌باشد. در این بخش شما با روش طراحی و ساخت یک مدار چاپی آشنا می‌شوید. اولین قدم برای طراحی یک PCB ایجاد یک پروژه می‌باشد ، برای ایجاد پروژه PCB مطابق شکل زیر از مسیر FILE > NEW > PROJECT گزینه NEW PCB PROJECT را انتخاب کنید:



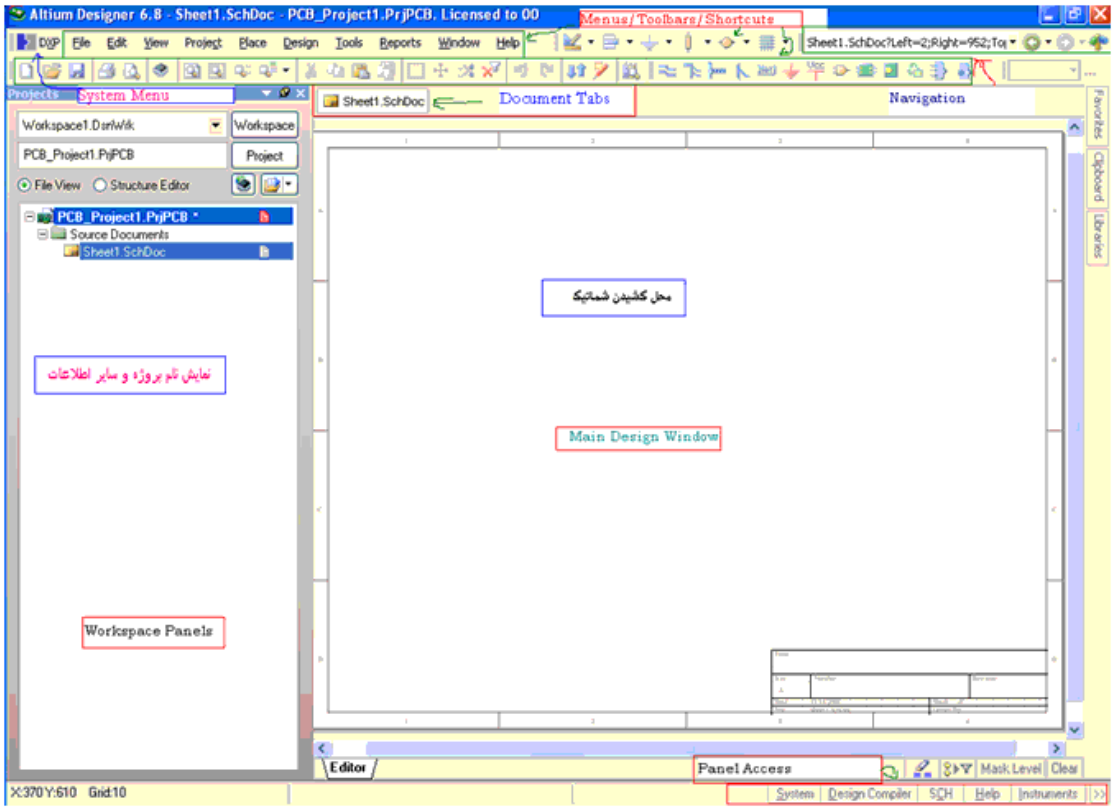
شکل ۱-۸: ایجاد پروژه PCB

در حال حاضر شما یک سند PCB ایجاد کرده‌اید ، اکنون مانند تصویر زیر ، از مسیر FILE > NEW گزینه schematic را انتخاب کنید تا یک سند شماتیک به پروژه اضافه شود.



شکل ۱-۹: افزودن سند شماتیک

بعد از انجام مراحل بالا ، همان‌طور که مشاهده می‌کنید ، دو پنجره باز می‌شود ، پنجره اول محل کشیدن شماتیک و پنجره دوم نام پروژه را نمایش می‌دهد.



شکل ۱-۱۰: پنجره‌های نام پروژه و شماتیک

منوهای موجود در این محیط را به 7 بخش تقسیم کرده‌ایم ، در زیر کار هر منو به‌صورت کلی گفته‌شده است:

System Menu : در این قسمت شما می‌توانید ابزار و منوها را مطابق میل خود بچینید.

Document Tabs : در این محل اسناد شماتیک PCB و ...که باز هستند نمایش داده می‌شود ، با کلیک راست‌روی هر یک از آن‌ها و انتخاب گزینه close می‌توانید آن‌ها را ببندید.

Menus/Toolbars/Shortcuts : شامل منوها ، میانبرها و نوارابزارهاست ، در نوارابزار، ابزاری که همیشه به آن نیاز دارید وجود دارد ، شما می‌توانید با رفتن به قسمت System Menu آن‌ها را ویرایش کنید.

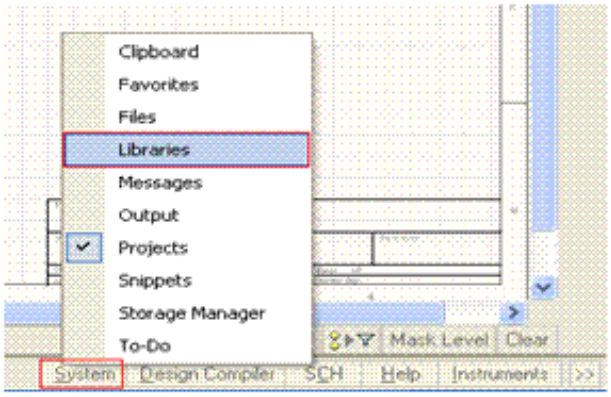
Navigation : برای دسترسی سریع به دیگر مکان‌های نرم‌افزار(home , محیط شماتیک, محیط PCB , محیط 3d و ...) می‌توانید از این ابزار استفاده کنید.

Panel Access: در این قسمت منوهای پرکاربرد نظیر کتابخانه (library) و ... برای دسترسی سریع‌تر قرار داده‌شده است.

Main Design Window: این قسمت محل کشیدن نقشه شماتیک می‌باشد ، شما باید قطعات را از کتابخانه برداشته و در اینجا نقشه خود را رسم کنید.

Workspace Panels: در این قسمت نام پرژه ها و دیگر اطلاعات نمایش داده می‌شود .

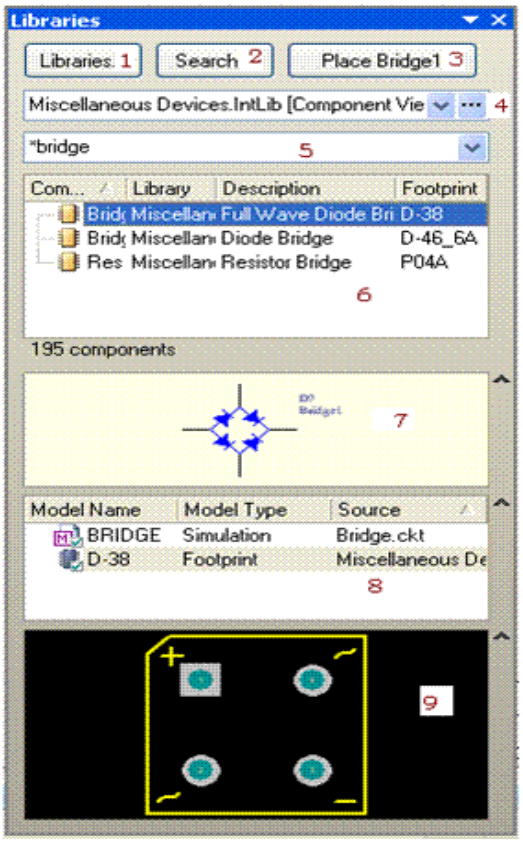
بعد از ایجاد پروژه PCB و سند شماتیک ، باید مدار را در صفحه شماتیک رسم کنید ، برای این کار از قسمت Panel Access منوی System گزینه‌ی Libraries را انتخاب کنید:



شکل ۱-۱۱: انتخاب کتابخانه نرم‌افزار

همان‌طور که مشاهده می‌کنید ، کتابخانه باز می‌شود ، برای آوردن قطعات باید در قسمت مشخص‌شده نام قطعه را وارد کنید و سپس روی Place کلیک کنید ، می‌بینید که قطعه به اشاره‌گر موس متصل می‌شود ، شما در هر قسمت از سند شماتیک کلیک کنید قطعه در همان‌جا گذاشته می‌شود ، اگر هنگامی‌که قطعه به اشاره‌گر متصل است ، کلید tab را بزنید می‌توانید مشخصات قطعه را ویرایش کنید. ما در اینجا پل دیودی را انتخاب کرده‌ایم (bridge).

برای درک بهتر کلیه اجزای پنجره کتابخانه شماره‌گذاری شده است و در ادامه قسمت‌های مختلف بیان‌شده است:



شکل ۱-۱۲: جست‌وجوی قطعات در کتابخانه

1– با کلیک روی این گزینه شما می‌توانید کتابخانه‌های دیگری را نصب‌کرده و از قطعات داخل آن‌ها استفاده کنید.

2– با کلیک روی این گزینه شما می‌توانید یک قطعه را در کتابخانه‌ها جستجو کنید.

3– با کلیک روی این گزینه قطعه انتخاب‌شده ، برای جایگذاری در سند شماتیک آماده می‌شود.

4– ممکن است شما برای طراحی یک مدار به قطعاتی نیاز داشته باشید که در یک کتابخانه موجود نباشد ، برای دسترسی راحت‌تر به کتابخانه‌ها می‌توانید آن‌ها را انتخاب کرده و در این قسمت قرار دهید.

5– نام قطعه موردنظر را باید در این قسمت بنویسید.

6- قطعاتی که مربوط به نام مورد جست‌وجوی شماست در این قسمت نمایش داده می‌شود ، با کلیک کردن روی نام هرکدام ، می‌توانید شکل ظاهری در شماتیک و PCB را در مکان‌های ۷ و ۹ ببینید.

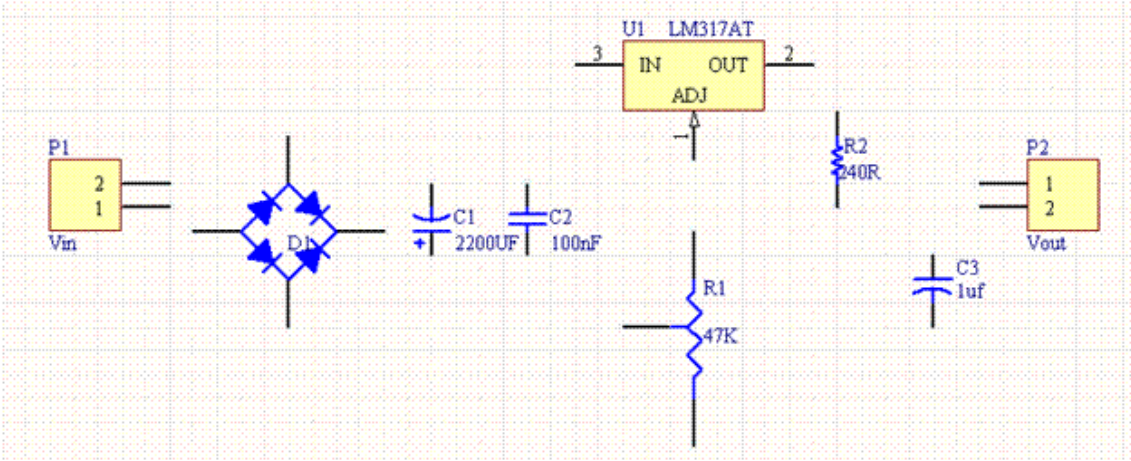
۷-شکل قطعه در سند شماتیک در این مکان نمایش داده می‌شود.

۸-شما با انتخاب این گزینه‌ها می‌توانید ، شکل قطعه را در سند PCB (Footprint) و در سند شبیه‌سازی (Simulation) (برای تعداد محدودی از قطعات) در مکان 9 ببینید.

۹- شکل قطعه را در سند PCB نشان می‌دهد.

اکنون قطعات را در مکان مناسب بچینید ، برای حرکت دادن قطعات روی آن‌ها کلیک چپ کنید و به هر جا که خواستید بکشید. برای بزرگنمایی و کوچک نمایی از کلیدهای page up و pagedown استفاده کنید .اگر خواستید قطعه را حذف کنید ، ابتدا روی آن کلیک چپ کنید (آن را انتخاب) و سپس کلید delet را بزنید. برای چرخاندن قطعه ، در حالتی که آن را با موس گرفته‌اید (روی آن کلیک چپ کنید و نگه‌دارید)کلید space را بزنید. برای برعکس کردن قطعه ، در حالتی که آن را گرفته‌اید از کلید x و کلید y استفاده کنید. برای ویرایش مشخصات قطعه روی آن دو بار کلیک چپ کنید.

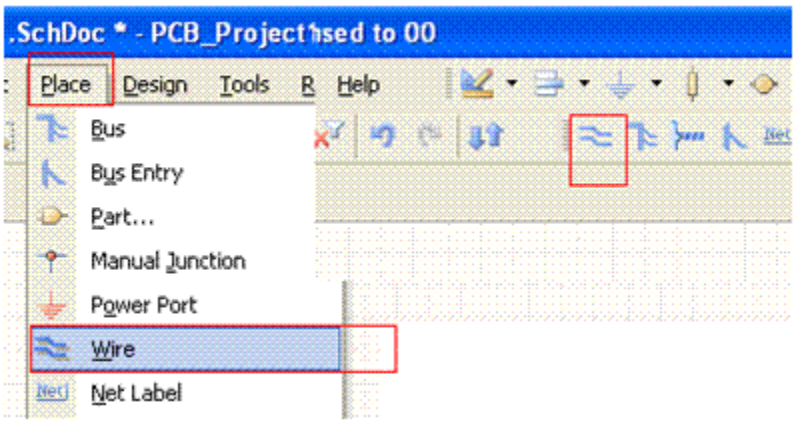
شماتیک مداری که قصد ساخت آن راداریم در زیر آورده شده است, تصویر زیر قطعات چیده شده را مشخص می‌کند:



شکل ۱-۱۳ : طرح شماتیک مدار

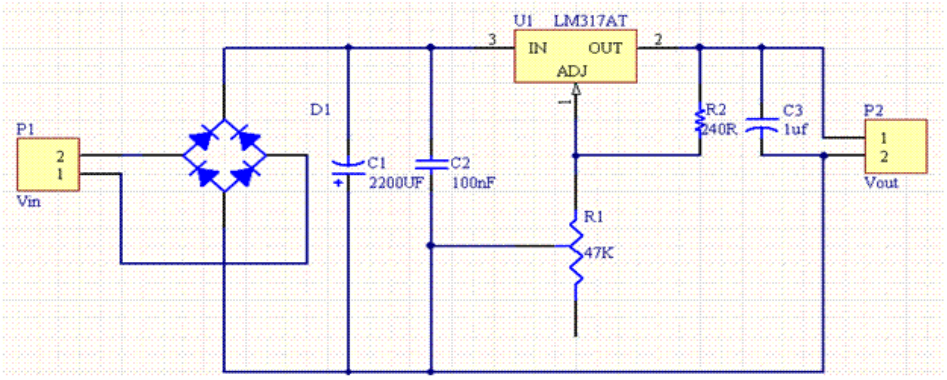
کشیدن مسیرهای رابط بین قطعات در سند شماتیک

برای کشیدن خطوط بین قطعات از منوی Place گزینه wire را انتخاب کنید ، این گزینه در قسمت toolbars نیز موجود است:



شکل ۱-۱۴: رسم خطوط بین قطعات

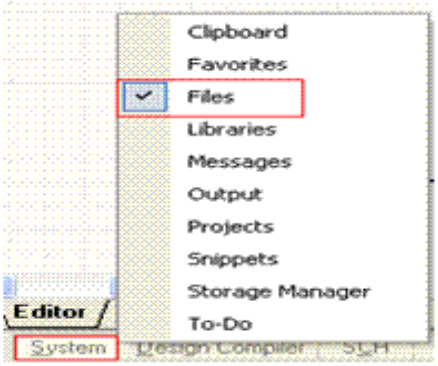
بر روی سر قطعه کلیک کنید ، تا هرکجا که خواستید ادامه دهید ، در مقصد دوباره کلیک کنید ، این کار را برای تمامی مسیرها انجام دهید. برای پاک کردن خط ابتدا آن را انتخاب کنید (روی آن کلیک چپ کنید) و سپس کلید delete را بزنید. برای دادن زاویه به خطوط از کلید SHIFT + SPACEاستفاده کنید. طرح نهایی شماتیک به‌صورت زیر است:



شکل ۱-۱۵: طرح نهایی شماتیک پس از اتصالات

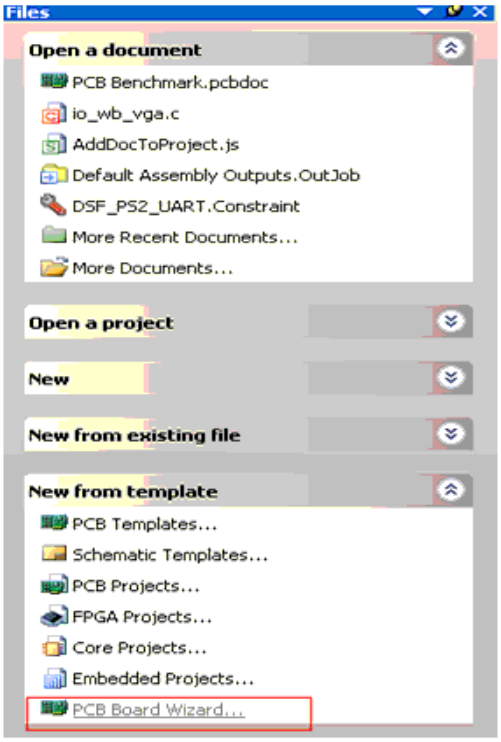
ایجاد سند PCB و تعیین اندازه برد

برای ایجاد سند PCB از قسمت Panel Access منوی System گزینه‌ی Files را انتخاب کنید .



شکل ۱-۱۶: انتخاب گزینه Files

در منوی فایل گزینه‌ی PCB Board Wizard را انتخاب کنید.

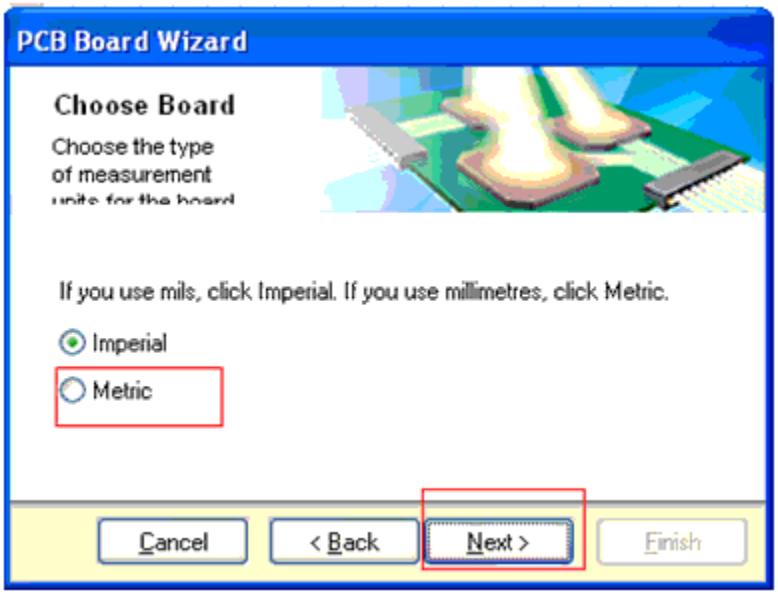


شکل ۱-۱۷: نمایی از منوی فایل

پنجره زیر باز می‌شود ، روی NEXT کلیک کنید:

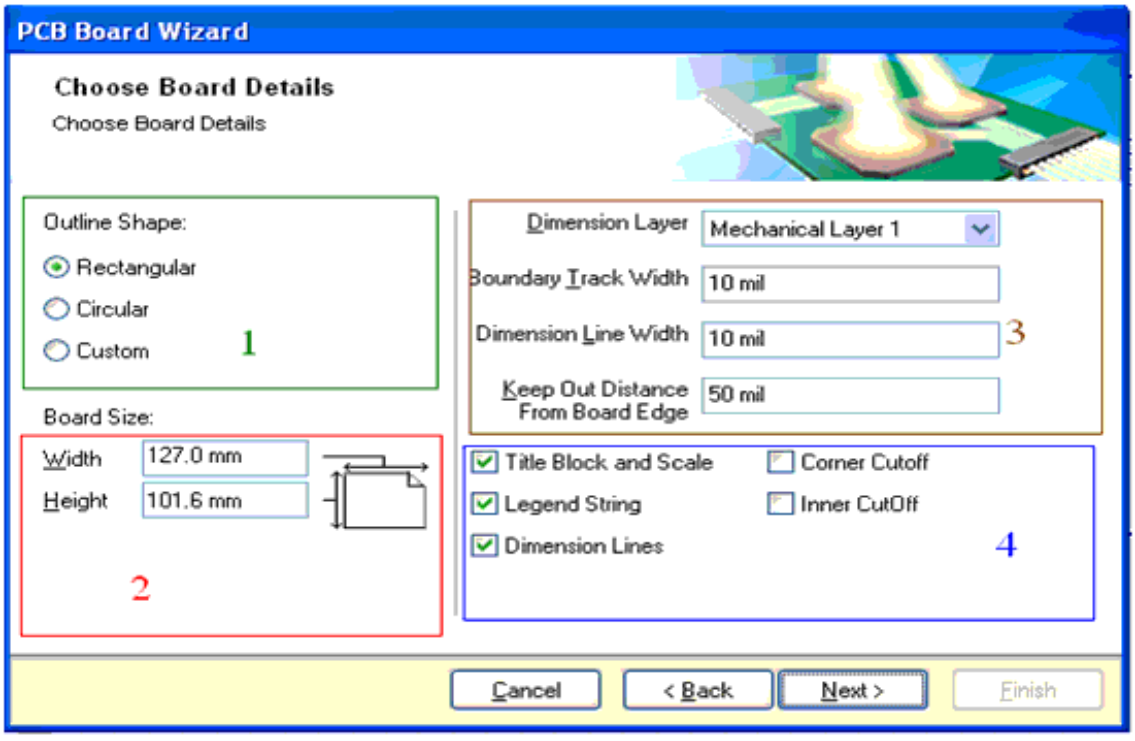
شکل ۱-۱۸: انتخاب گزینه Next

در پنجره زیر شما باید کمیت اندازه‌ها را مشخص کنید کمیت برحسب متر است یا اینچ. در این پنجره گزینه‌ی Metric را انتخاب کنید و بعد روی next کلیک نمایید.



شکل ۱-۱۹: تعیین کمیت اندازه‌ها

در پنجره بعدی گزینه Custom را انتخاب کنید و بعد روی next کلیک کنید (این گزینه به‌صورت پیش‌فرض انتخاب‌شده است). در پنجره بعدی که تصویر آن را در زیر مشاهده می‌کنید ، در قسمت 1 باید شکل برد را مشخص کنید ، گزینه اول بردی به شکل مستطیل یا مربع ، گزینه دوم بردی به شکل دایره و گزینه سوم بردی به شک دلخواه در اختیار شما می‌گذارد. ازآنجاکه برد ما مستطیل شکل است گزینه اول انتخاب می‌شود. در قسمت دوم ، طول و عرض برد برحسب میلی‌متر وارد می‌شود. بخش‌های سه و چهار دیگر تنظیمات مربوط به لایه‌ها می‌باشند که در مراحل بعدی توضیح داده می‌شود.

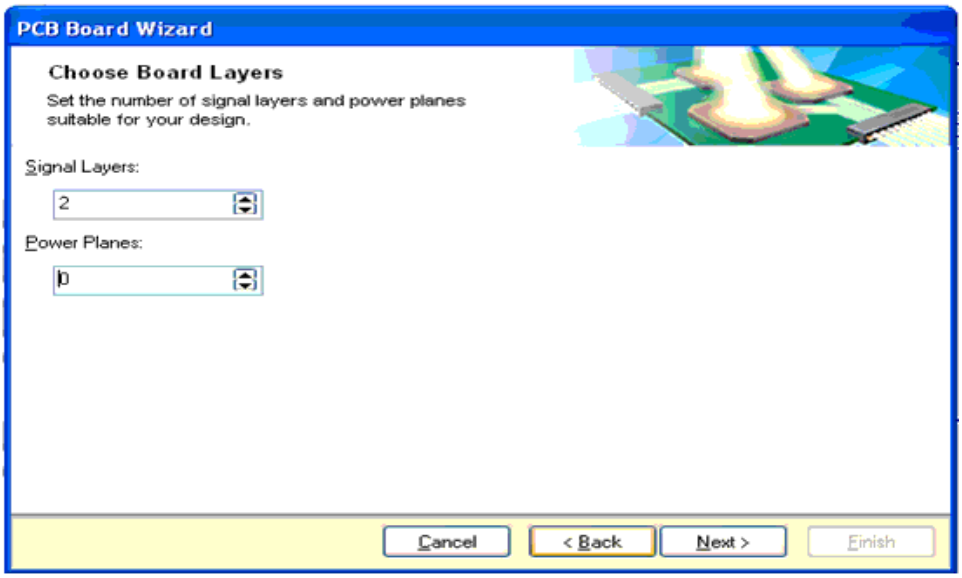


شکل ۱-۲۰: تعیین اندازه برد

در دو تا پنجره بعدی شما باید لبه‌های برد را مشخص کنید (این حاشیه‌ها در هنگام مونتاژ مدار لازم می‌باشند). اندازه دلخواه را وارد کرده و بر روی next کلیک کنید.

در پنجره بعدی که در تصویر زیر مشاهده می‌کنید ، تنظیمات را مانند شکل انجام دهید:

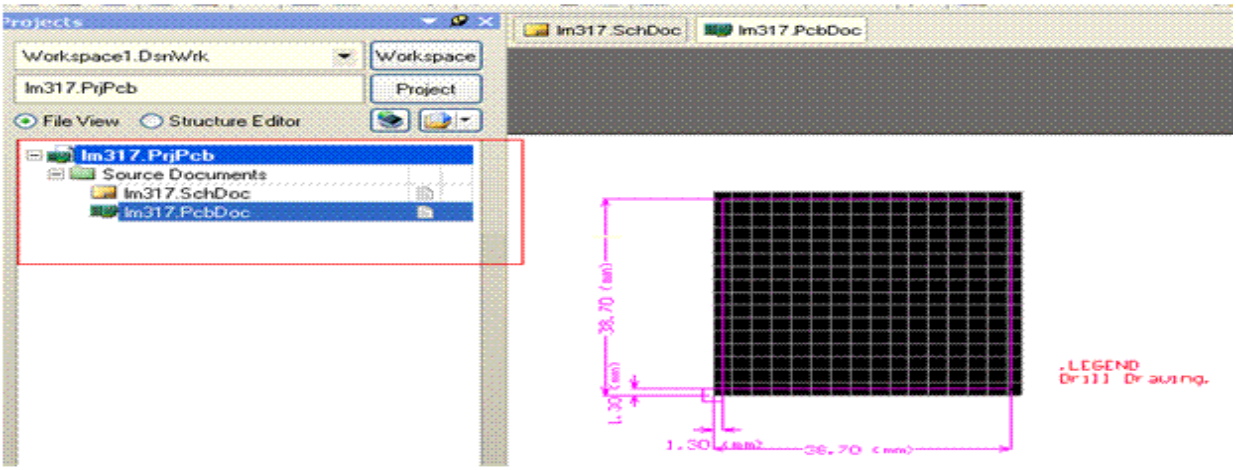
Signal Layers تعداد لایه‌های سیگنال را مشخص می‌کند و Power Planes تعداد سطوح نیرو (تعداد لایه‌های که جریان بالا از آن عبور می‌کند برای مورد اول مقدار 2 و برای مورد دوم مقدار 0 در نظر گرفته می‌شود).



شکل ۱- ۲۱: تنظیم تعداد لایه‌های برد

پنجره بعدی مربوط به ارتباط بین دو سطح در بردهای متالیزه می‌باشد ، در این پنجره روی next کلیک کنید. تنظیمات پنجره‌های بعدی را به‌صورت پیش‌فرض رها کرده و روی next کلیک کنید و درنهایت روی finish کلیک نمایید.

درصورتی‌که همه مراحل را درست انجام داده باشید در پنجره Projects نام فایل pcb آورده شده است، همچنین به Document Tabs پنجره جدیدی به نام pcb.pcbdoc اضافه‌شده است ، در پانل Projects فایل pcb.pcbdoc را در زیر pcb_ Project2 , drag کنید و سپس از منوی فایل گزینه all save را انتخاب کنید و سندها را به نام دلخواه و در مکان دلخواه ذخیره کنید.

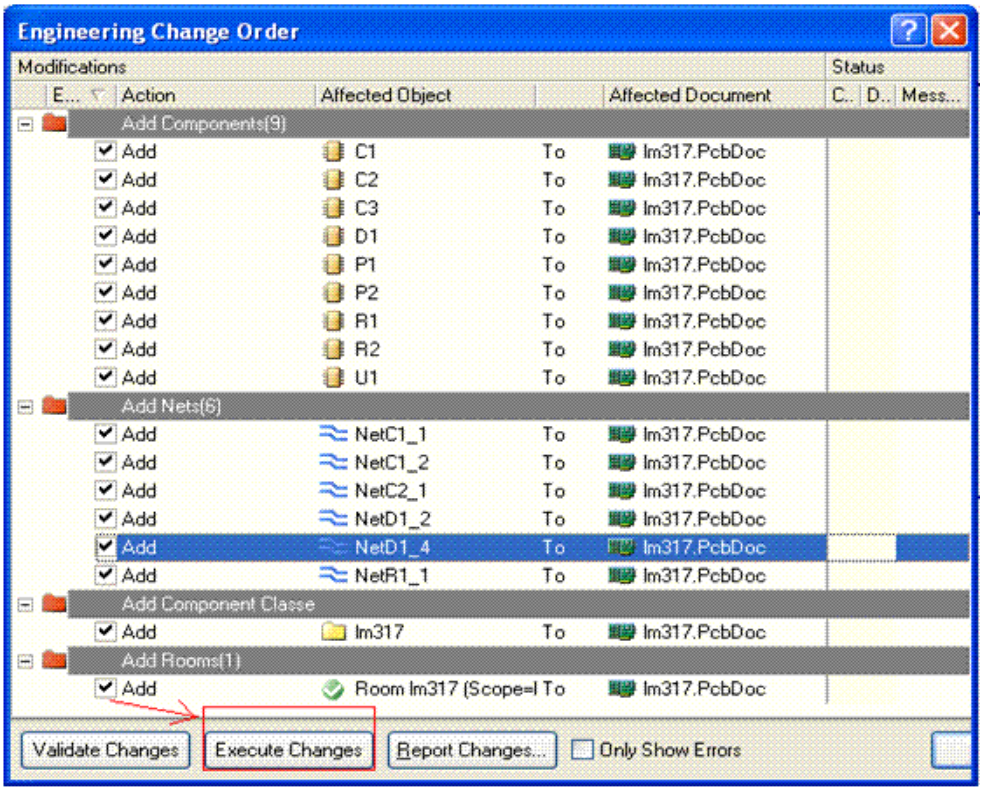


شکل ۱- ۲۲: نمایش نام پروژه در پنجره Projects

انتقال اطلاعات از سند شماتیک به سند PCB

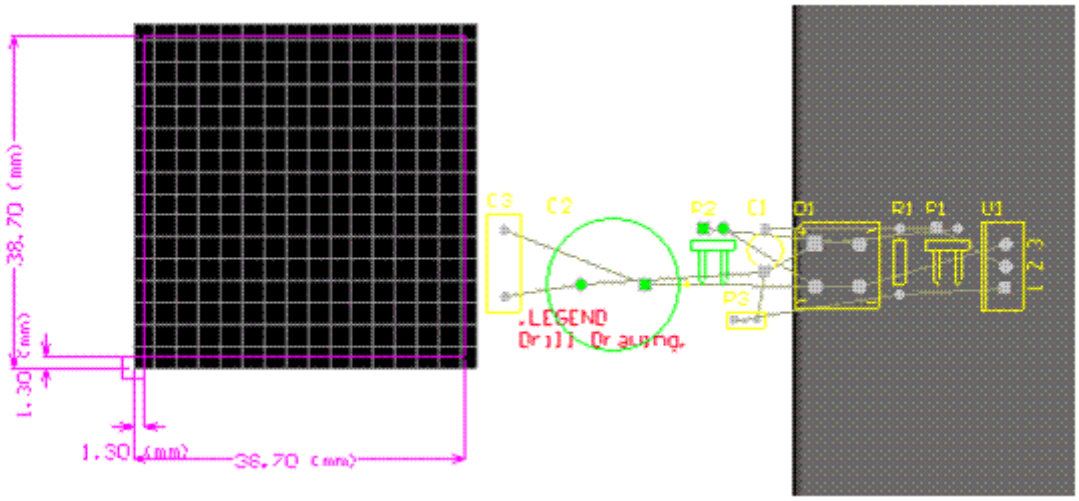
بعد از ذخیره‌ی سندها به سند شماتیک برگردید و در آنجا از منوی Design گزینه‌ی update pcb document pcb.pcbdoc را انتخاب کنید ،پنجره زیر باز می‌شود ، در این پنجره گزینه‌ی Execute Changes را بزنید ، قطعات از سند شماتیک به سند PCB منتقل می‌شود.

برای برداشتن صفحه سفید زیر سند PCB از منوی Design گزینه‌ی Board Options را انتخاب کنید و در پنجره بازشده تیک Display Sheet را بردارید.



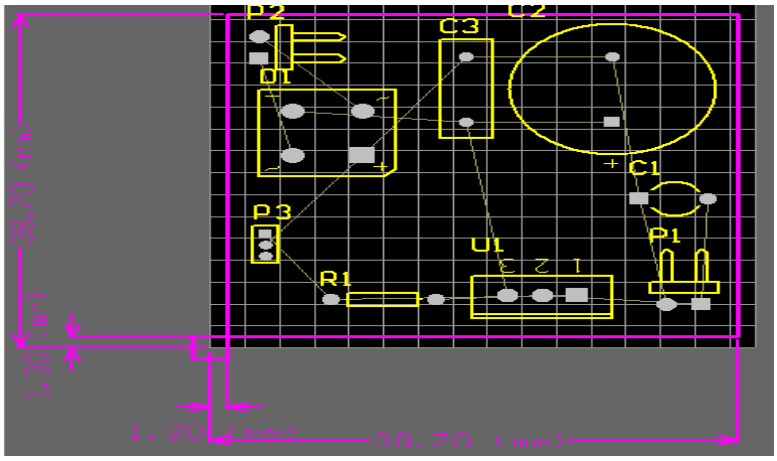
شکل ۱- ۲۳: مراحل انتقال اطلاعات از سند شماتیک به **PCB**

سند PCB و قطعات:



شکل ۱- ۲۴: تصویر قطعات بر روی سند PCB

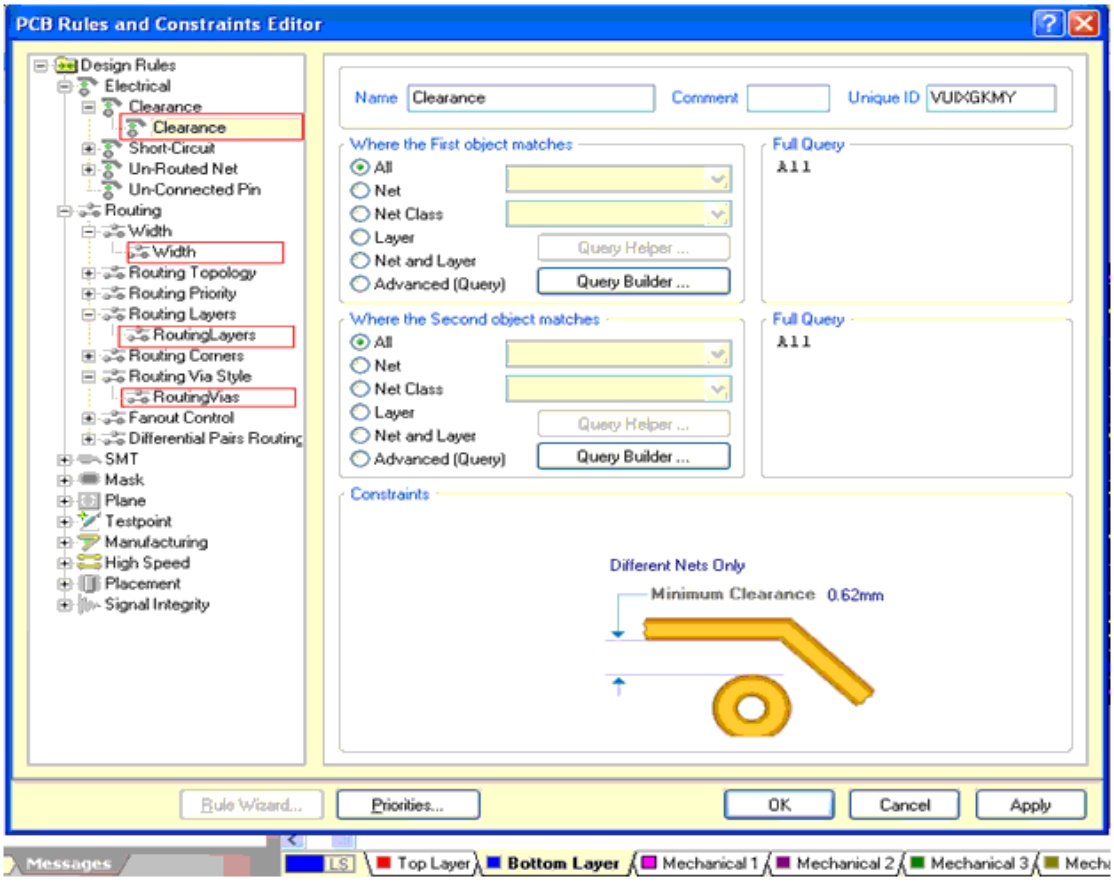
اکنون قطعات را در مکان دلخواه جایگذاری کنید ، انتقال قطعات و مانور روی آن‌ها مانند سند شماتیک می‌باشد ، در زیر سند PCB را پس از جابجایی قطعات مشاهده می‌کنید:



شکل ۱-۲۵: سند PCB پس از جابجایی قطعات

مسیر کشی بین قطعات در سند PCB

اکنون باید مسیرهای بین قطعات را رسم کنید ، برای این کار می‌توانید از سیم کشی دستی یا سیم‌کشی اتوماتیک استفاده کنید. در سند PCB از منوی Design گزینه‌ی Rules را انتخاب کنید , پنجره‌ی زیر باز می‌شود ، در منوی Electrical گزینه‌ی Clearance را انتخاب کنید ، این گزینه حداقل فاصله خطوط از یکدیگر و پایه قطعات را مشخص می‌کند ، آن‌ها به مقدار دلخواه تغییر دهید (0.6 میلی‌متر).

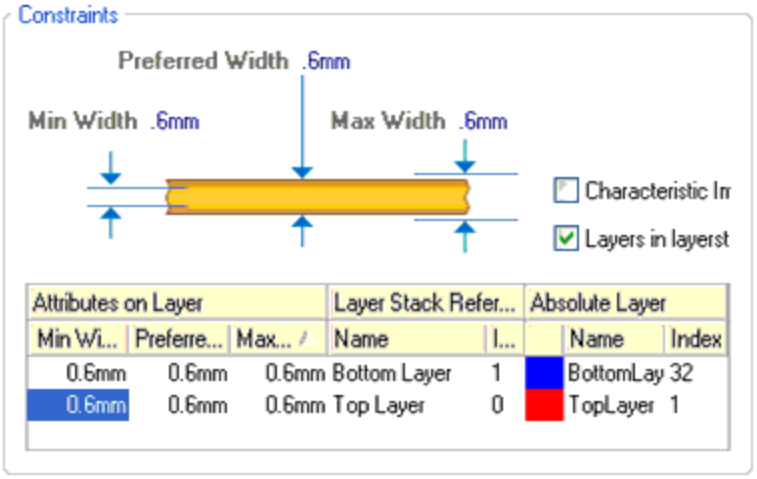


شکل ۱- ۲۶: انتخاب گزینه **Rules** از منوی **Design**

در همین صفحه بر روی گزینه‌ی Routing کلیک کنید ، در این قسمت تنظیمات سه منو به‌صورت زیر تغییر می‌کند.

گزینه اول Width:

با کلیک کردن روی این گزینه شما باید ضخامت خطوطی که قطعات را به هم متصل می‌کند مشخص کنید (پهنای لایه مس رابط بین قطعات) در این قسمت از شما سه اندازه min و Preferred و max خواسته‌شده است ، ما در اینجا هر سه اندازه را مساوی هم و برابر با 0. 6 میلی‌متر وارد کرده‌ایم:



شکل ۱- ۲۷: تعیین ضخامت خطوط اتصال

گزینه دوم Routing Layers :

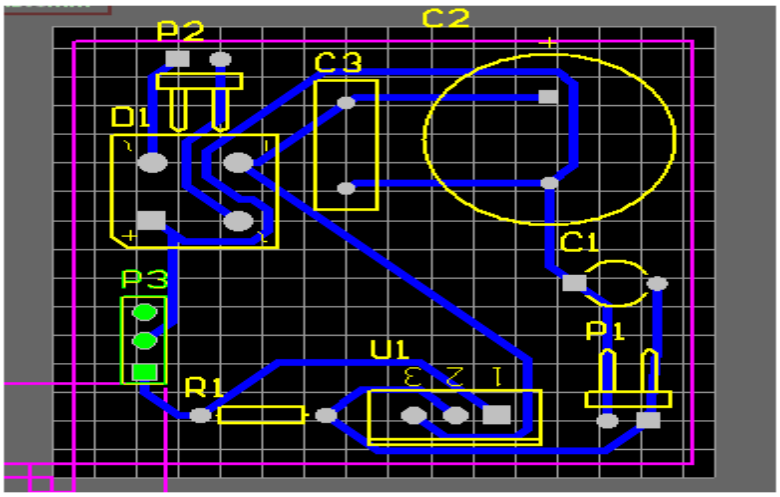
با کلیک کردن روی این گزینه شما می‌توانید تعداد لایه‌های فیبر مدار چاپی را تعیین کنید (فیبر در چندلایه طراحی می‌شود) (در مدارت پیچیده جهت کاهش فضای به‌کاررفته فیبر را چندلایه می‌سازند) . چون فیبر ما یک‌لایه است و به لایه زیرین نیاز داریم ، گزینه‌ی Bottom Layer را تیک می‌زنیم (تیک گزینه‌های دیگر را برمی‌داریم).

شکل ۱- ۲۸: تعیین تعداد لایه‌های فیبر مدار چاپی

گزینه سوم RoutingVias :

با کلیک کردن روی این گزینه ، می‌توانید در پنجره بازشده محدودیت‌های مربوط به سوراخ رابط بین لایه‌های مختلف برد را تعیین کنید (این گزینه فقط برای بردهای چندلایه کار برد دارد).

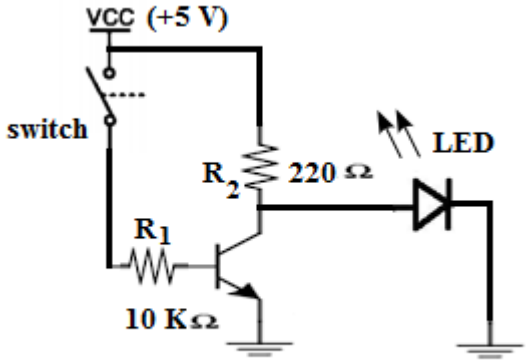
سند PCB را با آخرین تغیرت مشاهده می‌کنید:



شکل ۱- ۲۹: سند نهایی PCB

طراحی مدار به پایان رسید ، شما باید فایل‌هایی را که ذخیره کرده‌اید به شرکت‌هایی که درزمینهٔ ساخت PCB فعال هستند بدهید تا فیبر مدار چاپی آن را بسازند.

- مدار پول آپ ذیل را ابتدا بر روی برد بورد پیاده‌سازی و عملکرد آن را بررسی نمایید سپس با استفاده از نرم‌افزار آلتیوم, آن را شبیه‌سازی نموده و PCB آن را رسم نمایید.



شکل ۱- ۳۰: مدار پول آپ ترانزیستوری

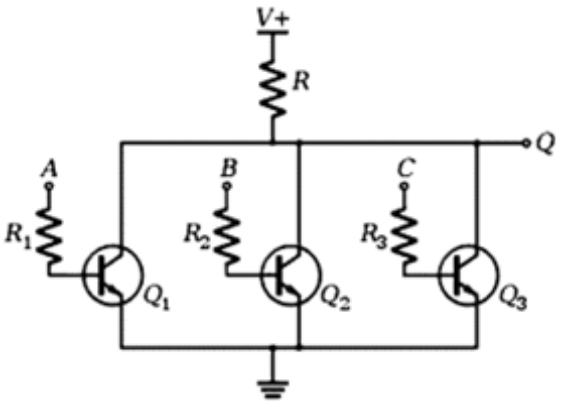
جلسه دوم آشنایی با فن‌آوری‌های RTL, RDL و DTL

هدف از این آزمایش آشنایی با فن‌آوری‌های RTL, RDL و DTL به‌وسیله پیاده‌سازی گیت‌های منطقی در این منطق‌ها می‌باشد. بعضاً این سه فن‌آوری را منطق نیز می‌نامند.

الف: مدارهای مقابل گیت‌های طراحی‌شده بر اساس منطق RDL را نشان می‌دهند. آن‌ها را بسته و جداول صحت مربوطه را استخراج نمایید. (ورودی‌ها ۰ و ۵+ ولت و مقاومت‌ها یک کیلو اهم و دیود 1N4001 انتخاب شوند).

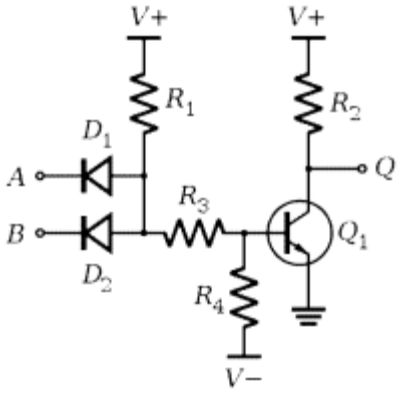
شکل ۲-۱: دو گیت طراحی‌شده با منطق **RDL**

ب: مدار مقابل گیت طراحی‌شده بر اساس منطق RTL را نشان می‌دهند. آن‌ها را بسته و جدول صحت مربوطه را استخراج نمایید. (ترانزیستورها BC107, ، مقاومت‌های ورودی ۴۵۰ اهم، مقاومت تغذیه ۶۴۰ اهم و تغذیه ۳/۶ ولت انتخاب شوند).



شکل ۲- ۲: نمونه‌ای از منطق RTL

ج: مدار مقابل گیت طراحی‌شده بر اساس منطق DTL را نشان می‌دهد، آن را بسته و جدول صحت مربوطه را استخراج نمایید. (ترانزیستور BC107 ، R_1=5K ، R_2=2K ، R_3=1K ، R_4=5K ، دیودها 1N4001، ترانزیستور BC107 و تغذیه ۵ ولت انتخاب شوند).



شکل ۲- ۳: طرحی از منطق DTL

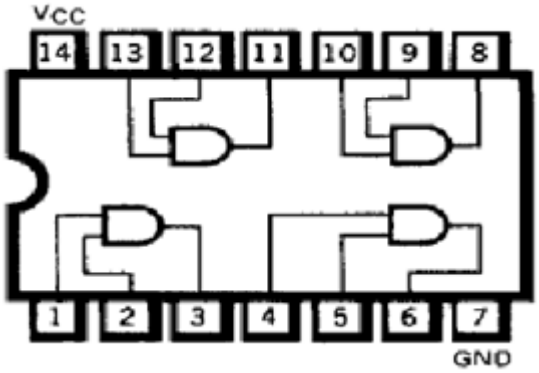
پس از پیاده‌سازی مدارهای الف، ب و ج بر روی برد بورد برای هر مدار یک PCB طراحی نمایید.

جلسه سوم آشنایی با فناوری‌های TTL, CMOS و چگونگی استخراج جدول درستی یک تراشه

هدف از این آزمایش آشنایی با آی‌سی‌های فناوری TTL و چگونگی استخراج جدول درستی یک آی سی با اعمال ورودی‌های مختلف به یک آی سی می‌باشد.

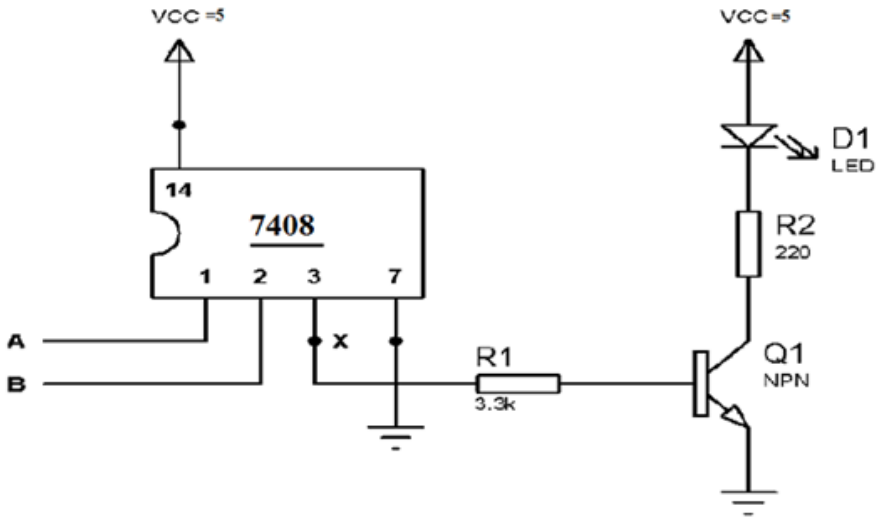
الف- در این بخش برای بررسی عملکرد آی سی‌های دیجیتال، آی سی ۷۴۰۸ را که شامل ۴ گیت AND می‌باشد بکار گرفته و با استفاده از المان‌های موردنیاز دیگر برای تست آن، بر روی برد بورد پیاده‌سازی می‌کنیم.

مدار داخلی و شماره پایه‌های گیت AND به‌صورت زیر می‌باشد:



شکل ۳-۱: تصویر یک گیت **AND**

ما در این آزمایش از یکی از گیت‌های AND داخل این آی سی استفاده خواهیم کرد. برای استفاده از این آی سی باید ولتاژهای تغذیه را به آن اعمال کنیم. برای این منظور پایه ۷ آی سی را به زمین و پایه ۱۴ آن را به ولتاژ ۵ ولت وصل کنید. بر روی برد بورد مداری مطابق شکل زیر ببندید. تغذیه آی سی و نیز تغذیه کلکتور ترانزیستور را وصل کنید. برای استفاده از اولین گیت AND درون آی سی، ورودی‌های مدار را به پایه‌های ۱ و ۲ آی سی وصل کنید.



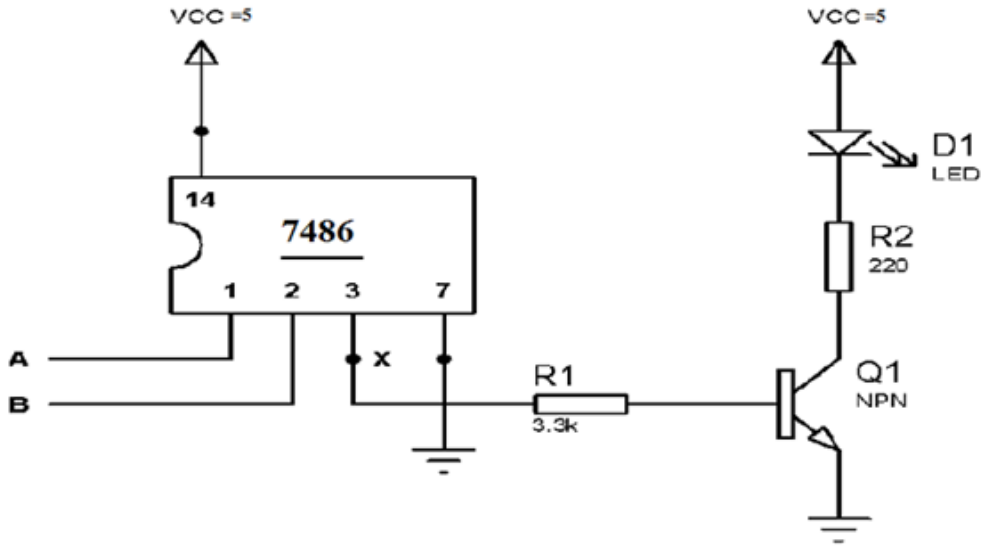
شکل ۳- ۲: مدار طراحی‌شده با استفاده از گیت **AND**

با اعمال ۴ حالت مختلف برای ورودی‌ها، خروجی (ولتاژ نقطه x) را اندازه‌گیری کرده و جدول درستی زیر را تکمیل کنید. همچنین در هر حالت وضعیت روشن یا خاموش بودن LED را هم ثبت کنید.

A	B	ولتاژ خروجی	منطق خروجی	وضعیت LED
0	0			
0	1			
1	0			
1	1			

جدول ۳-۱: جدول صحت مدار ترانزیستوری با گیت **AND**

ب- مانند بخش قبل مدار شکل (۳-۳) را بر روی برد بورد ببندید و نتیجه را گزارش کنید.



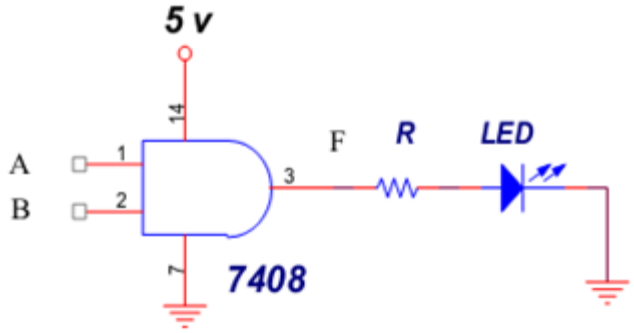
شکل ۳-۳: مدار ترانزیستوری با گیت **7486**

A	B	ولتاژ خروجی	منطق خروجی	وضعیت LED
0	0			
0	1			
1	0			
1	1			

جدول ۳-۲: جدول صحت مدار ترانزیستوری با گیت **7486**

با توجه به جدول درستی به‌دست‌آمده مشخص کنید که آی سی ۷۴۸۶ چه نوع گیتی می‌باشد؟

حالت خروجی یک مدار (High یا Low) را می‌توان با استفاده از دیود نورانی (LED) مشاهده نمود.



$$R = \frac{VDD - VDon}{I\ Dmax}$$

شکل ۳-۴: تعیین سطح خروجی یک گیت به کمک **LED**

$V_{DD=}5\ V$

$I_{Dmax=16\ mA}$ بیشترین جریان قابل‌تحمل LED:

$V_{D=}1.6\ V$ افت ولتاژ دو سر دیود نورانی در هنگام روشن بودن

مقاومت R به‌منظور محدود کردن شدت‌جریان و جلوگیری از سوختن دیود و تراشه به‌کاررفته است.

جدول درستی ذیل را با توجه به مدار فوق پر نمایید.

A	B	F
0	0	
0	5	
5	0	
5	5	
0	باز	
باز	0	
5	باز	
باز	باز	

جدول ۳-۳: جدول صحت مدار شکل ۳-۴

- اگر یکی از ورودی‌های یک آی سی در فن‌آوری TTL بازبماند، این ورودی معادل چه منطقی عمل می‌کند؟

آزمایش قبل را در مورد تراشه‌ای CMOS گیت NAND یعنی ۴۰۱۱ تکرار نمایید.

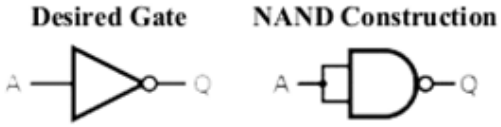
اگر یکی از ورودی‌های یک آی سی در فن‌آوری CMOS بازبماند، این ورودی معادل چه منطقی عمل می‌کند.

جلسه چهارم معادل‌سازی انواع روابط و گیت‌های منطقی

هدف: معادل‌سازی انواع روابط و گیت‌های منطقی با استفاده از گیت‌های پایه NAND و NOR

- روش پیاده‌سازی روابط و گیت‌های منطقی با استفاده از گیت پایه NAND در زیر نشان داده‌شده است. علاوه بر پیاده‌سازی این مدارها تمامی روابط را با استفاده از گیت پایه NOR پیاده نمایید. برای این منظور از گیت‌های NANAD موجود در آی سی ۷۴۰۰ و گیت‌های NOR موجود در آی سی ۷۴۰۲ با تکنولوژی TTL استفاده نمایید.

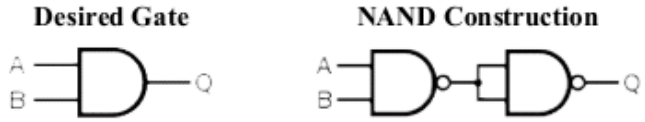
۱)



Truth Table	
Input A	Output Q
0	1
1	0

شکل ۴-۱: مدار معادل یک گیت NOT

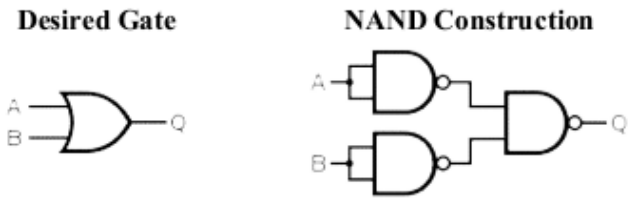
۲)



Truth Table		
Input A	Input B	Output Q
0	0	0
0	1	0
1	0	0
1	1	1

شکل ۴-۲: مدار معادل یک گیت AND

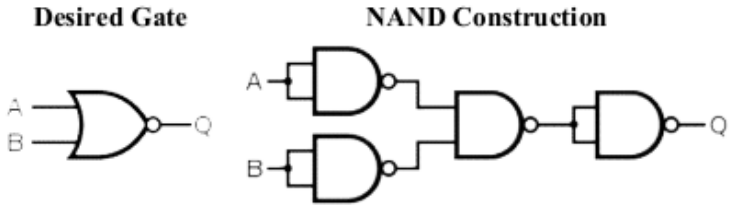
۳)



Truth Table		
Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0

شکل ۴-۳: مدار معادل یک گیت OR

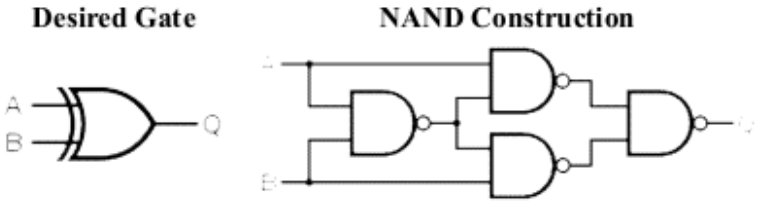
٤)



Truth Table		
Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	0

شکل ۴-۴: مدار معادل یک گیت NOR

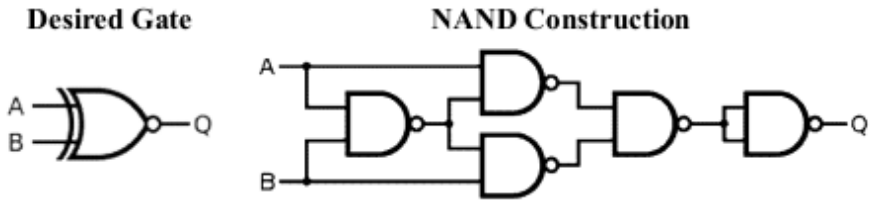
٥)



Truth Table		
Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	0

شکل ۴-۵: مدار معادل یک گیت XOR

٦)



Truth Table		
Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	1

شکل ۴-۶: مدار معادل یک گیت XNOR

۷) تابع منطقی $f = AB + CD$

جلسه پنجم

ساده‌سازی توابع منطقی

هدف: تشکیل جدول صحت از روی صورت‌مسئله و ساده‌سازی توابع منطقی

الف: برای کنترل چراغ راهنمایی یک چهارراه، از یک مرکز فرماندهی دستور تغییر رنگ چراغ‌ها داده می‌شود. این دستور توسط چند رشته سیم به چهارراه می‌رسد. هدف طراحی مداری است که توسط آن چراغ‌ها را به ترتیب زیر روشن یا خاموش نمایند.

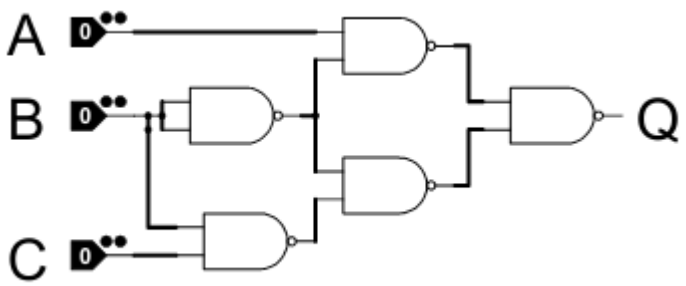
- چراغ A زمانی سبز یا زرد است که چراغ B قرمز باشد.
- چراغ B زمانی سبز یا زرد است که چراغ A قرمز باشد.

جدول صحت مدار فوق را طراحی کرده و مدار را با کمک گیت‌های منطقی NAND پیاده نمایید. آیا می‌توان با استفاده از گیت‌های منطقی دیگر تعداد گیت‌های مورداستفاده را کاهش داد؟ بررسی نمایید آیا می‌توان مدار فوق را با مدار دیگر جایگزین نمود. آیا لزوماً این مسئله یک جواب دارد؟ اگر جواب منفی است مدار دومی پیشنهاد نمایید.

ب: می‌خواهیم در یک سیستم صنعتی هرگاه یک فاز و یا دو فاز از سه فاز ورودی قطع شدند، سیستم اخطار دهد. جدول صحتی برای پیاده‌سازی مدار فوق طراحی کرده و پس از ساده‌سازی آن به کمک جدول کارنو ، مدار را پیاده نمایید.

ج: مداری طراحی نمایید که دارای سه ورودی B, A و C باشد. می‌خواهیم هرگاه C=0 است در خروجی A ظاهر شود و هرگاه C=1 است B در خروجی ظاهر شود. ابتدا جدول صحت و رابطه‌ی خروجی را به دست آورید, سپس مدار را با استفاده از گیت‌های منطقی لازم ساخته و نتیجه را تحقیق نمایید.

-مدار زیر را با استفاده از گیت‌های NAND ساخته و جدول صحت و رابطه خروجی را به دست آورید و مدار را با مدار قبل مقایسه نمایید. آیا تفاوتی وجود دارد؟



شکل ۵-۱: مدار پیاده‌سازی شده فقط با گیت NAND

پس از پیاده‌سازی مدارهای الف، ب و ج بر روی برد بورد برای هر مدار یک PCB طراحی نمایید.

جلسه ششم

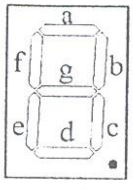
آزمایش BCD to SevenSegment

هدف: آشنایی با تراشه ۷۴۴۸ و نمایشگر هفت‌قسمتی

تئوری آزمایش: برای نشان دادن اعداد ذخیره‌شده در ثبات‌ها و یا شمارنده‌ها، می‌توان از نمایشگرهای هفت‌قسمتی استفاده کرد. نمایشگرهای هفت‌قسمتی از هفت دیود منتشرکننده نور (در بعضی موارد یک دیود هم برای نقطه اعشاری) ساخته‌شده‌اند. ترکیبات انتخابیLED ها، برای ایجاد ارقام عددی و دیگر سمبل‌ها روشن می‌شود. یک LED، هنگامی‌که ولتاژ در ورودی آند به‌اندازه کافی مثبت‌تر از ولتاژ پایین به کاتد باشد روشن می‌شود. در مدار دیجیتال، این ولتاژها با اعمال یک ولتاژ بالا به آند و یک ولتاژ پایین در کاتد، ایجاد می‌شود. برای حداقل کردن تعداد سیگنال مای کنترل، آند مای LED ها معمولاً در یک نقطه مشترک به هم وصل شده‌اند و لذا آن را آند مشترک، می‌نامند. در پیکربندی آند مشترک، آند معمولاً به ولتاژ بالا و کاتد به‌طور جداگانه کنترل می‌شوند. درنتیجه اعمال یک ولتاژ منطق صفر موجب روشن شدن LED می‌گردد، درصورتی‌که منطق ۱، LED را غیرفعال می‌سازد. وضعیت مخالفی برای پیکربندی کاتد مشترک برقرار است. با توجه به خاموش و روشن شدن LED ها می‌توان ارقام و اشکال مختلفی را بر روی نمایشگر هفت‌قسمتی مشاهده کرد.

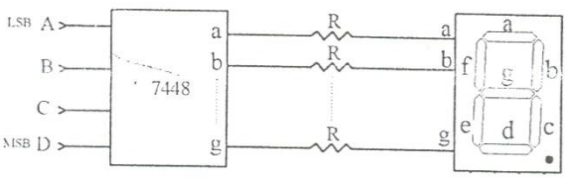
کار در داخل آزمایشگاه

۱- یک نمایشگر هفت‌قسمتی را بر صفحه آزمایش قرارداده و ارتباط بین پایه‌ها و دیودهای نورانی a تا g را به دست آورید و برای این کار در نمایشگر هفت‌قسمتی آند مشترک (کاتد مشترک) سر مشترک را توسط یک مقاومت (حداقل ۱۰۰ اهم) به ولتاژ بالا (ولتاژ پایین برای کاتد مشترک) متصل می‌کنیم سپس بااتصال ولتاژ پایین (ولتاژ بالا) به هر یک از پایه‌ها و روشن شدن هر یک از دیودهای نورانی a تا g ارتباط بین پایه‌ها و دیودهای نورانی را به دست می‌آوریم و نام هر دیود نورانی را بر روی پایه مربوط به آن می‌نویسیم.



شکل ۶-۱: ساختار یک SevenSegment

۲- تراشه ۷۴۴۸ را مطابق شکل زیر به یک نمایشگر هفت‌قسمتی کاتد مشترک متصل کرده و به ازای کلیه حالات ورودی (0000 تا 1111) علائم مشاهده‌شده بر روی نمایشگر را یادداشت کنید. توجه داشته باشید که مقاومت مای نشان داده‌شده در شکل جهت جلوگیری از سوختن نمایشگر و تراشه ضروری می‌باشد.



شکل ۶-۲: اتصال تراشه ۷۴۴۸ به SevenSegment

- حال پایه شماره ۳تراشه (Lamp Test) را به ولتاژ Low وصل نموده و به ازای حالت‌های مختلف ورودی، علائمی که بر روی نمایشگر ظاهر می‌شود را یادداشت کنید و کار پایه LT را نتیجه‌گیری کنید.

- مدار را به حالت اولیه برگردانید و حال پایه شماره ۴ تراشه را به ولتاژ Low وصل نموده و به ازای حالت مای مختلف ورودی، علائمی که بر روی نمایشگر ظاهر می‌شود را یادداشت کنید و کار پایه ۴ را نتیجه بگیرید.

- مدار را به حالت اولیه برگردانید سپس پایه شماره ۵ تراشه را به ولتاژ Low وصل نموده و به ازای حالت‌های مختلف ورودی (0000 تا 1111) اشکال نشان داده‌شده توسط نمایشگر را یادداشت کنید همچنین در هر حالت، ولتاژ پایه شماره ۴ را با یک LED یا پروب منطقی تشخیص دهید.

۵- تراشه ۴۵۱۱ را که از خانواده CMOS و یک نمایشگر هفت‌قسمتی می‌باشد را توسط مقاومت‌های مناسب (بین ۱۰۰ تا ۳۰۰ اهم) به یک نمایشگر هفت‌قسمتی کاتد مشترک متصل نموده سپس پایه LT و BL را به ولتاژ بالا و LE را به ولتاژ پایین متصل کرده حال به ازای کلیه حالات ورودی (0000 تا 1111)، علائمی را که بر روی نمایشگر ظاهر می‌شود را یادداشت کنید. حال پایه شماره ۳ تراشه (LT) را به ولتاژ Low وصل نموده و به ازای حالت مای مختلف ورودی، علائم نشان داده‌شده را یادداشت کنید و کار این پایه را نتیجه بگیرید.(به شکل ۵ مراجعه فرمایید)

- مدار را به حالت اولیه برگردانید سپس پایه شماره ۴ تراشه (BL) را به ولتاژ Low وصل کرده و به ازای ترکیبات مختلف ورودی، نتایج را مشاهده و یادداشت کنید.

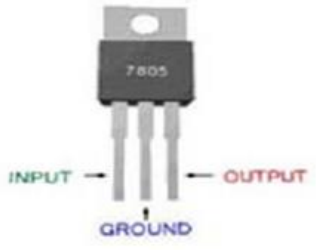
- مدار را به حالت اولیه برگردانید سپس یک ورودی بین ۱ تا ۹ را انتخاب کرده و به تراشه اعمال کنید. سپس پایه شماره ۵ تراشه را به ولتاژ بالا وصل کرده و ورودی را چندین بار تغییر دهید. با مشاهده خروجی در هر حالت عملکرد این پایه را نتیجه بگیرید.

در ادامه می‌خواهیم یک شمارنده BCD را همراه با تمام مدارها و عناصر جانبی‌اش بر روی برد بورد ببندیم و نتیجه شمارش را بر روی 7Segment ببینیم.

هدف, آشنایی با آی سی‌های TTL مربوط به شمارنده‌های BCD ، آشنایی با آی سی‌های مبدل BCD به 7Segment و راه‌اندازی نمایشگرهای ۷ قطعه‌ای با استفاده از آی سی‌های TTL بر روی برد بورد است.

نکته: این آزمایش را در دو حالت برای اعمال پالس انجام خواهیم داد. در ابتدا کلاک را به‌صورت دستی و سپس از طریق سیگنال ژنراتور اعمال می‌کنیم.

نکته: دقت کنید که برای جلوگیری از آسیب رسیدن به آی سی ها می‌توانید از رگولاتور ۷۸۰۵ برای تولید ولتاژ ۵ ولت تنظیم‌شده استفاده کنید. پایه‌های این رگولاتور به‌صورت زیر می‌باشد:



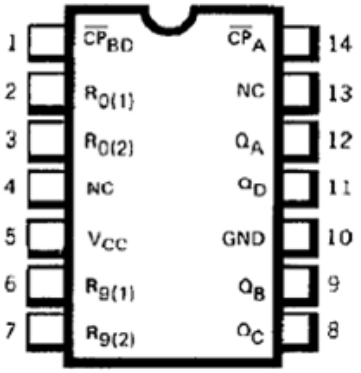
شکل ۶-۳: رگولاتور ۷۸۰۵

الف- پیاده‌سازی مدار با اعمال کلاک دستی:

نکته: آی سی ۷۴۹۰ یک شمارنده BCD می‌باشد. پایه‌های ۱ و ۱۴ کلاک ورودی آی سی هستند و پایه‌های $R_O(1)$ و $R_O(2)$ و $R_g(1)$ و $R_g(2)$ برای ریست تراشه یا انتخاب مد شمارش طبق جدول زیر می‌باشند:

RESET/COUNT (see Note 2)

RESET INPUTS				OUTPUT
$R_{O(1)}$	$R_{O(2)}$	$R_{g(1)}$	$R_{g(2)}$	Q_D Q_C Q_B Q_A
H	H	L	X	L L L L
H	H	X	L	L L L L
X	X	H	H	H L L H
X	L	X	L	COUNT
L	X	L	X	COUNT
L	X	X	L	COUNT
X	L	L	X	COUNT



شکل۶-۴: ساختار آی سی ۷۴۹۰ و جدول عملکرد پایه‌های آن

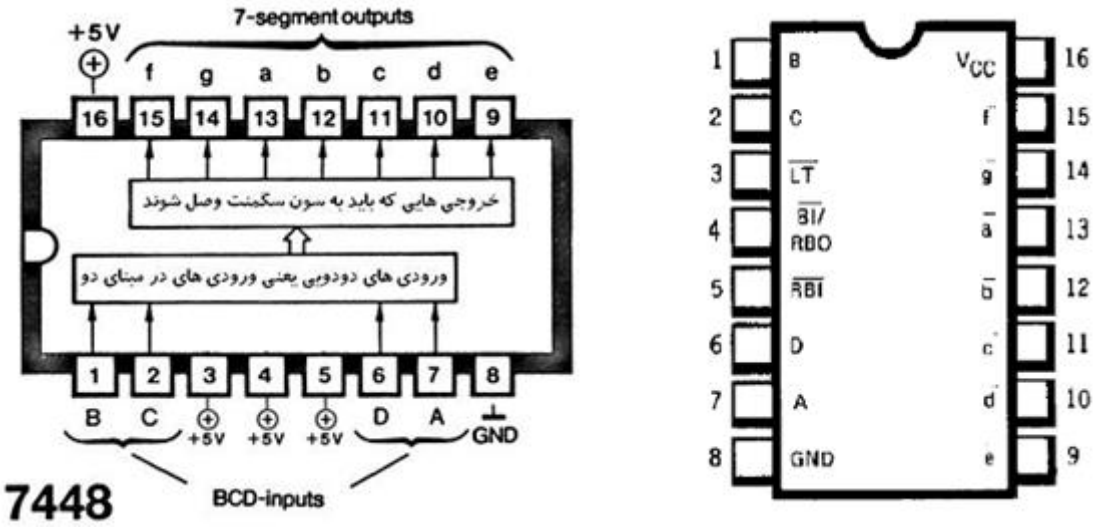
توصیف پایه‌های این آی سی در جدول زیر آمده است:

Pin No	Function	Name
1	Clock input 2	Input2
2	Reset1	R1
3	Reset2	R2
4	Not connected	NC
5	Supply voltage; 5V (4.75V – 5.25V)	Vcc
6	Reset3	R3
7	Reset4	R4
8	Output 3, BCD Output bit 2	Q_C
9	Output 2, BCD Output bit 1	Q_B
10	Ground (0V)	Ground
11	Output 4, BCD Output bit 3	Q_D
12	Output 1, BCD Output bit 0	Q_A
13	Not connected	NC
14	Clock input 1	Input1

جدول۶-۱: توصیف پایه‌های آی سی ۷۴۹۰

نکته: آی سی ۷۴۴۷ یک مبدل BCD به 7Segment می‌باشد. این آی سی اعداد صفرتا ۹ (خروجی حاصل از شمارنده BCD) را به کدهای قابل‌فهم برای نمایشگر هفت قطعه‌ای جهت نمایش اعداد تبدیل می‌کند.

همان‌طور که بیان شد ورودی‌های آی سی ۷۴۴۷ می‌توانند خروجی‌های یک آی سی ۷۴۹۰ (شمارنده BCD) باشند. از طرف دیگر خروجی‌های آی سی ۷۴۴۷ به 7Segment وصل می‌شوند. در شکل زیر این آی سی به همراه جدول درستی آن نشان داده‌شده است:



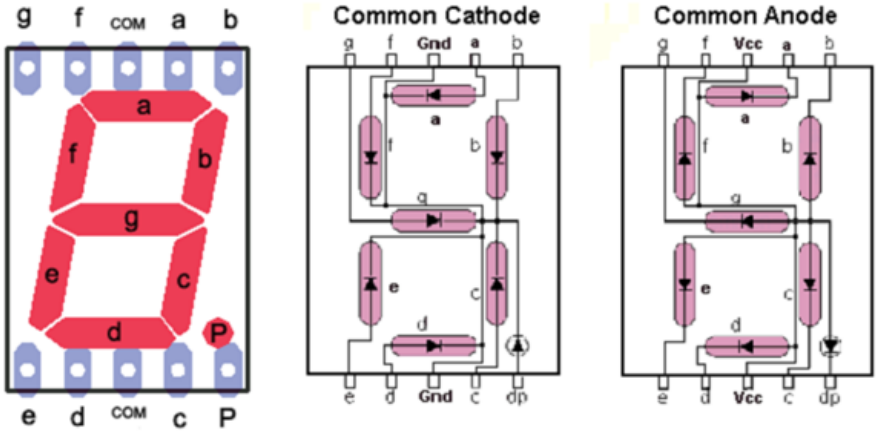
شکل۶-۵: نمایی از پایه‌های آی سی ۷۴۴۷

	INPUTS						OUTPUTS							
DECIMAL OR FUNCTION	$\overline{LT}$	$\overline{RBI}$	D	C	B	A	$\overline{BI/RBO}$	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L
5	H	X	L	H	L	H	H	L	H	L	L	H	L	L
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H
$\overline{BI}$	X	X	X	X	X	X	L	H	H	H	H	H	H	H
$\overline{RBI}$	H	L	L	L	L	L	L	H	H	H	H	H	H	H
$\overline{LT}$	L	X	X	X	X	X	H	L	L	L	L	L	L	L

جدول۶-۲: عملکرد پایه‌های ۷۴۴۷

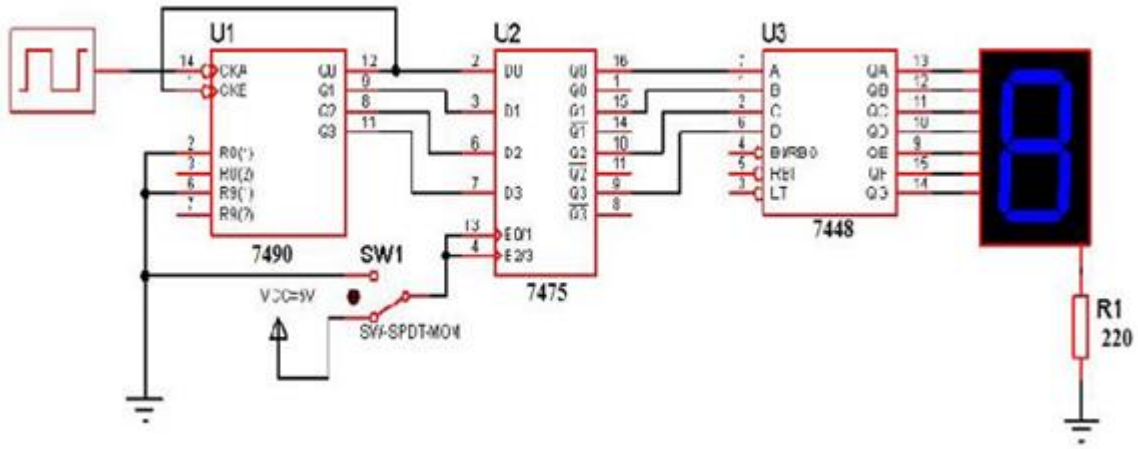
نکته: اگر نمایشگر شما کاتد مشترک است, باید از ۷۴۴۸ استفاده کنید و اگر نمایشگر شما آند مشترک است, باید از ۷۴۴۷ استفاده کنید.

نکته: پایه‌های سون سگمنت را هم مانند شکل زیر وصل کنید (البته شکل زیر یک سون سگمنت کاتد مشترک است, به همین جهت است که پایه‌های مشترک را به ولتاژ منفی وصل کرده‌ایم و با اعمال ورودی مثبت به هرکدام از پایه‌های ورودی‌اش, LED مربوط به آن روشن می‌شود. اگر سون سگمنت شما آند مشترک باشد باید پایه‌های مشترک را به ولتاژ مثبت وصل کنید.):



شکل۶-۶: تصویر سون سگمنت های کاتد مشترک و آند مشترک

حال مداری مطابق با شکل زیر را ازنظر عملکرد و کارکرد بخش‌های مختلف آن توصیف و تشریح کرده, سپس آن را بر روی برد بورد ببندید و نتیجه را گزارش کنید.



شکل ۶-۷: مدار مورد آزمایش

توجه:برای اعمال کلاک به شمارنده از کلیدهای فشاری ببندید.

نکته: یک خازن ۱۰۰ nf را به‌صورت موازی با کلیدهای فشاری ببندید. در هنگام اعمال پالس به‌صورت دستی, این خازن برای حذف نویز به کار می‌رود.

توجه: می‌توانید مدار را مرحله‌به‌مرحله ببندید و تست کنید, آی سی بعدی را به مدار اضافه کنید. زیرا عیب‌یابی مدارهای بزرگ‌تر به‌مراتب سخت‌تر است. مثلاً می‌توانید ابتدا فقط شمارنده را به همراه المان‌های موردنیاز و LED هایی برای تست خروجی آن, بر روی برد بورد وصل کرده و آن را تست کنید. سپس مثلاً آی سی ۷۴۴۷ را همراه یک 7Segment به‌صورت مجزا ببندید و تست کنید و در پایان مدارها را بر روی یک بورد ببندید.

نکته: در این مدار از یک آی سی ۷۴۷۵ نیز استفاده‌شده است که یک latch ۴بیتی برای قفل مقدار شمارش‌شده می‌باشد. علت به‌کارگیری آن در مدار را توضیح دهید.

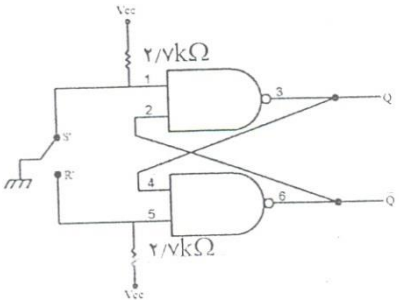
ب- پیاده‌سازی مدار با اعمال کلاک دستی:

مانند بخش قبل مدار را بر روی برد بورد ببندید. در این بخش با استفاده از سیگنال ژنراتور موجود در آزمایشگاه, یک سیگنال پالس 1HZ به ورودی کلاک شمارنده اعمال کرده و نتیجه را بر روی 7segment ببندید.

تولید کلاک دستی (بدون نویز کلید): هنگامی‌که با یک سوئیچ و با یک قطعه سیم (بااتصال به GND یا VCC) بخواهیم کلاک دستی تولید کنیم بایستی اغتشاش یا نویز حاصل هنگام قطع یا وصل کلید را برطرف کنیم . اغتشاش کلید به علت ارتعاشات نامحسوسی (زمانی حدود 10^{ms}) می‌باشد که هنگام زدن کلید به وجود می‌آید، مانند شکل زیر:

شکل۶-۸: اغتشاش کلید

اغتشاش کلید می‌تواند باعث نتایج اشتباه در مدارات ترتیبی گردد. برای مثال در یک شمارنده باعث شمارش نامنظم می‌گردد. بنا براین بایستی آن را از کلاک دستی یا سوئیچ حذف نمود. برای این منظور می‌توان از مدار زیر استفاده نمود.



شکل۶-۹: فلیپ فلاپ پایه RS

مدار فوق یک فلیپ فلاپ پایه است که حالت بی‌اثر آن به ازای ۱ و ۱ روی پایه‌های 's و 'R می‌باشد. با توجه به‌اضافه شدن مقاومت بین VCC و هر یک از این پایه‌ها می‌توان مطمئن بود که هنگام آزاد بودن این پایه‌ها (وصل نبودن به سوئیچ) سطح ۱ به ورودی دروازه NAND مربوط به آن پایه برسد. بنابراین اگر سوئیچ فوق به 'S وصل شود خروجی Q سطح یک می‌شود و هرگونه ارتعاش کلید هنگام وصل به 'S بی‌اثر روی نتیجه اولیه خواهد بود. تنها موقعی خروجی Q تغییر می‌کند که کلید به پایه 'R وصل گردد و در این صورت نیز هرگونه ارتعاش سوئیچ نسبت به این پایه، بی‌اثر روی اولین نتیجه (سطح صفر Q) می‌باشد.

جلسه هفتم

آشنایی با PLDs و نرم‌افزار Xilinx ISE

تراشه‌های منطقی قابل برنامه‌نویسی[1]

تراشه منطقی قابل برنامه‌نویسی شامل مجموعه‌ای از المان‌های منطقی قابل برنامه‌نویسی و flip-flop مای مرتبط باهم بعلاوه اتصال‌های قابل برنامه‌نویسی می‌باشد. کاربر می‌تواند با کمک سلول‌های حافظه[2] وظیفه و برنامه هر قسمت از این مجموعه را مشخص و کنترل کند. اگرچه تراشه‌های متفاوت از ساختارهای[3] مختلفی استفاده می‌کنند ولی اساس کار تمام آن‌ها یکی می‌باشد.

۷-۱) انواع تراشه‌های منطقی قابل برنامه‌نویسی[4]

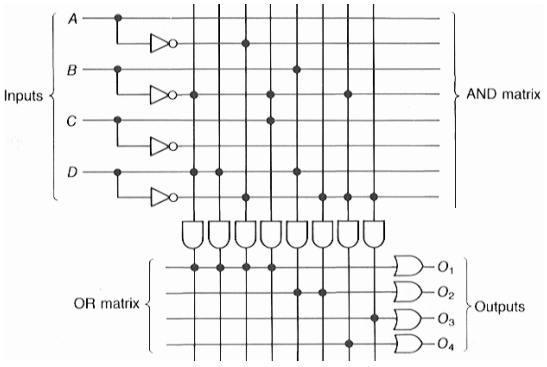
امروزه تعداد زیادی تراشه‌های قابل برنامه‌نویسی توسط شرکت‌های مختلف و با ساختارهای متفاوت در دسترس می‌باشد که می‌توان آن‌ها را در ۴ گروه عمده طبقه‌بندی نمود:

طرح‌های منطق قابل برنامه‌نویسی ساده	**(SPLDs)** **S**imple **P**rogrammable **L**ogic **D**evices
طرح‌های منطق قابل برنامه‌نویسی پیچیده	**(CPLDs)** **C**omplex **P**rogrammable **L**ogic **D**evices
آرایه مای گیت قابل برنامه‌نویسی میدانی	**(FPGAs)** **F**ield **P**rogrammable **G**ate **A**rrays
اتصالات قابل برنامه‌نویسی میدانی	**(FPICs)** **F**ield **P**rogrammable **I**nter **C**onnect

SPLD (۷-۱-۱

SPLD ها، کوچک‌ترین و کم‌هزینه‌ترین شکل از تراشه‌های منطقی قابل برنامه‌نویسی می‌باشند. یک SPLD معمولاً شامل ۴ تا ۲۲ macrocellاست که می‌تواند تعداد زیادی از TTL مای سری ۷۴۰۰ را در خود جای دهد. هر macrocell به‌طور کامل با macrocell دیگر در ارتباط است. زیرمجموعه مای این خانواده بدین شرح می‌باشند:

- **PAL** (Programmable Array Logic)
- **GAL** (Generic Array Logic)
- **PLA** (Programmable Logic Array)
- **PLD** (Programmable Logic Device)



شکل ۷-۱: PLA

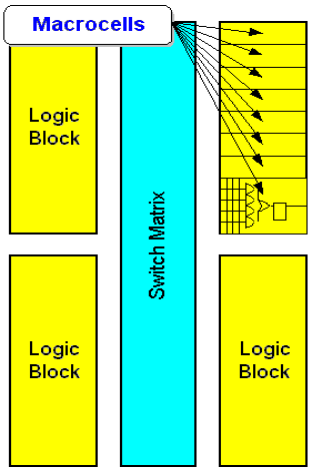
1-Programmable Logic Devices
2- Memory Cells
3- Architecture

۴- بی کسب اطلاعات بیشتر می‌توانید به سایت شرکت مراجعه نمایید.

CPLD (۲-۱-۷

CPLD ها شبیه SPLDها می‌باشند. با این تفاوت که دارای ظرفیت بسیار بالاتری هستند. یک CPLD مشخص هم‌ارز ۲ تا ۶۴ SPLD است و به‌طورمعمول شامل ۱۰ تا ۱۰۰ macrocell می‌باشند.۸ تا ۱۶ macrocell باهم تشکیل یک بلوک عملیات[1] بزرگ‌تر می‌دهند. macrocell های داخل یک بلوک عملیاتی به‌طور کامل باهم ارتباط دارند.زیرمجموعه های این خانواده بدین شرح می‌باشند:

- **EPLD** (Erasable Programmable Logic Device)
- **PEEL** (Programmable Electrically Erasable Logic)
- **EEPLD** (Electrically-Erasable Programmable Logic Device)
- **MAX** (Multiple Array Matrix)



شکل ۷-۲: بلوک منطقی و ماتریس سوییچ

در یک تعریف جامع ، CPLD ها از چندین بلوک منطقی[2] مانند PAL تشکیل‌شده‌اند که از طریق ماتریس سوئیچ[3] قابل برنامه‌نویسی، با یکدیگر در ارتباط می‌باشند. هر بلوک منطقی شامل ۴ تا ۱۶ macrocell است، که این تعداد به نوع ساختار آن‌ها بستگی دارد. شکل ۴ بلوک‌های منطقی و ماتریس سوئیچ یک CPLD را نشان می‌دهد.

FPGA (۳-۱-۷

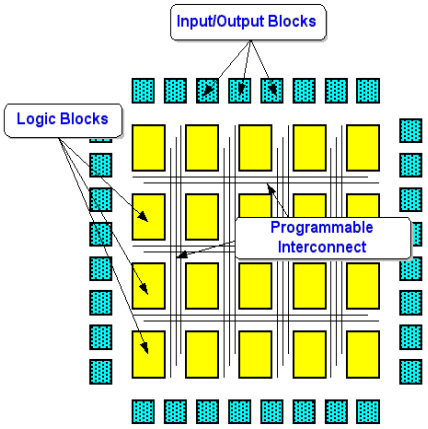
FPGA تراشه‌ای قابل‌برنامه‌ریزی است که از آرایه‌ای از بلوک‌های منطقی، بلوک‌های ورودی/خروجی[4] و ماتریس‌های سوئیچ تشکیل‌شده است. شکل ۲-۱ ساختار داخلی تراشه FPGA را نشان می‌دهد. محتوای بلوک‌های منطقی به نوع FPGA بستگی دارد و می‌تواند از ساختارهای (معماری‌های) متنوعی تبعیت نماید، برنامه‌نویسی FPGA با استفاده از سوئیچ‌های قابل‌برنامه‌ریزی انجام می‌گیرد.

ساختار FPGA با ساختار خانواده‌های SPLD و CPLD متفاوت است و نسبت به آن‌ها بیشترین ظرفیت المان‌های منطقی را در اختیار کاربران قرار می‌دهد. موفقیت این تراشه‌ها در ظرفیت بالا و کارایی خوب ، به دلیل ساختار بلوک‌های منطقی و ساختار سیم‌کشی بین بلوک‌هاست. یک FPGA شامل ۶۴ تا ده‌ها هزار بلوک‌های منطقی و تعداد بیشتری flip-flop می‌باشد. برخی از تراشه‌های این خانواده بدین شرح می‌باشند:

- **pASIC** (Altera)
- **FLEX, APEX** (Altera)

1-Function Block
2- Logic Block
3- Switch Matrix
4- Input/Output Blocks

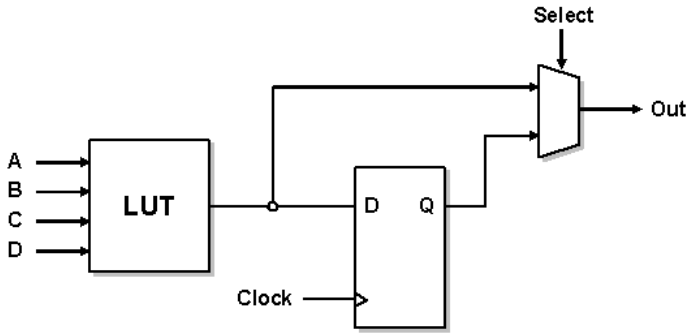
- **ACT** (Actel)
- **STRATIX** (Altera)
- **VIRTEX** (Xilinx)
- **SPARTAN** (Xilinx)



شکل ۷-۳: ساختار داخلی تراشه

۷-۱-۳-۱) بلوک‌های منطقی

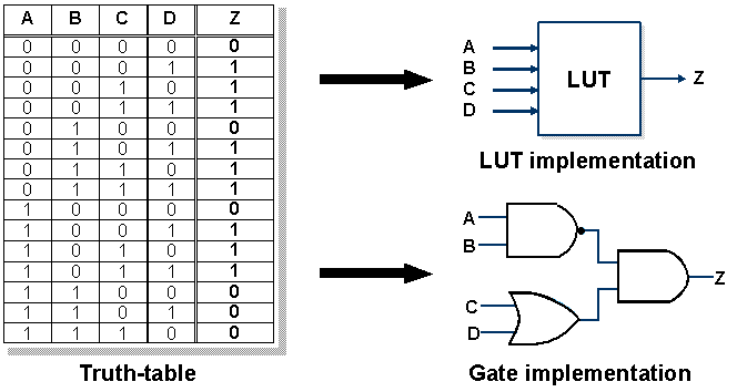
مداری که در FPGA پیاده‌سازی می‌شود، به قطعات کوچک‌تر تجزیه‌شده، و هر یک از این قطعات در یک سلول منطقی پیاده‌سازی می‌شود. FPGA به‌طورکلی از دو نوع سلول منطقی استفاده می‌نماید.



شکل ۷-۴: بلوک‌های منطقی

الف) سلول‌های منطقی مبتنی بر LUT [1]

این سلول‌ها شامل بلوک‌های منطقی بزرگی هستند که دارای ۲ یا تعداد بیشتری LUT و flip-flop می‌باشند. LUT حافظه کوچکی است که برای پیاده‌سازی یک جدول صحت از آن استفاده می‌شود.



شکل ۷-۵: سلول‌های منطقی مبتنی بر LUT

1- Look-up Table

ب) سلول‌های منطقی مبتنی بر مالتی پلکسر[1]

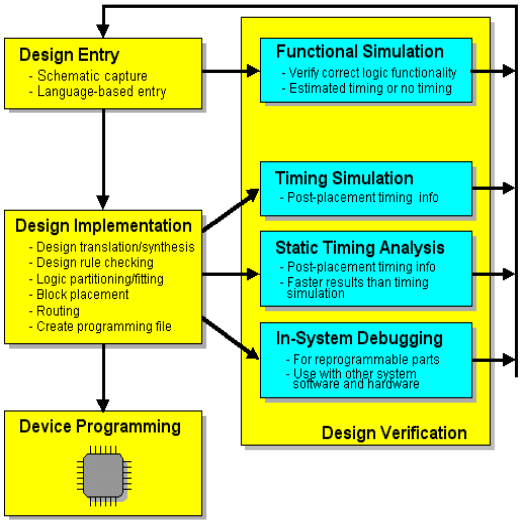
در این سلول‌ها تعداد زیادی بلوک‌های منطقی نسبتاً ساده‌ای وجود دارد، این بلوک‌ها شامل یک تابع منطقی ۲ ورودی یا مالتی پلکسر ۴ به ۱ و یک flip-flop می‌باشد. که برای طرح‌های پیچیده و فشرده مناسب می‌باشند.

۷-۲) برنامه‌نویسی تراشه‌های منطقی

برنامه‌نویسی تراشه‌ها در ۴ مرحله انجام می‌گیرد.

۱) طراحی[2]

امروزه ابزارهای قابل‌استفاده متنوعی از شرکت‌های مختلف برای طراحی مدار در دسترس کاربران و برنامه نویسان می‌باشد. در تمام این ابزارها می‌توان از دو طریق استفاده از شماتیک و یا زبان سخت‌افزاری، مدار موردنظر را شبیه‌سازی نمود. برخی از طراحان ترجیح می‌دهند از قسمت شماتیک که بیشتر موردعلاقه‌شان می‌باشد استفاده نمایند درحالی‌که برخی دیگر ترجیح می‌دهند



شکل ۷-۶: نحوه برنامه‌نویسی تراشه

طرحشان را با زبان سخت‌افزاری مانند Verilog، VHDL یا ABEL توصیف نمایند. برخی نیز از ترکیب شماتیک و زبان سخت‌افزاری استفاده می‌کنند.

۲) پیاده‌سازی طرح[3]

بعدازاینکه طراحی مدار از طریق شماتیک یا سنتز کردن انجام شد، آماده پیاده‌سازی بر روی تراشه موردنظرمی‌شود. اولین گام، شامل برگرداندن طرح مدار به فرمتی است که در داخل ابزارها تعریف‌شده‌اند. اکثر ابزارهای ساده‌سازی، فرمت‌های استاندارد Netlist را می‌خوانند و ترجمه طرح معمولاً به‌صورت خودکار انجام می‌شود.

سپس نرم‌افزار طرح را به بلوک‌های منطقی تقسیم[4] کرده، طوری که برای پیاده‌سازی بر روی تراشه منطقی قابل برنامه‌نویسی مورد نظر آماده باشد. قسمت‌بندی مهم‌ترین گام برای FPGA ها و CPLD ها می‌باشد. درنتیجه تقسیم‌بندی درست و مناسب در FPGA ها از سیم‌کشی طولانی جلوگیری شده و کارایی بالا می‌رود همچنین در CPLD کارایی و چگالی افزایش پیدا می‌کنند.

بعد از تقسیم‌بندی طرح موردنظر به بلوک‌های منطقی، نرم‌افزار پیاده‌سازی به جستجوی مناسب‌ترین و بهترین مکان برای بلوک‌های منطقی می‌گردد. هدف اساسی، کاهش تعداد و طول سیم‌کشی موردنیاز برای افزایش کارایی سیستم می‌باشد. این عملکرد محاسباتی دقیق برای تمام FPGA ها و CPLD های بزرگ لازم است.

2-Multiplexer
3- Design Entry
۱-Design Implementation
۲-Partitioning

نرم‌افزار پیاده‌سازی هنگام جایگذاری کردن بلوک‌های منطقی طول سیم‌کشی و ازدحام مسیر سیم‌کشی را کنترل می‌کند. هنگامی‌که سیم‌کشی و مکان‌یابی[۱] به اتمام رسید، نرم‌افزار فایل برنامه را که برای پیکربندی طرح به شکل باینری[۲] نوشته می‌شود ایجاد می‌کند.

۳) اثبات یا تأیید طرح[۳]

بررسی و تأیید طرح در چندین مرحله و در حین طراحی انجام می‌گیرد. اول، شبیه‌سازی که به‌طور عملی و متقارن با شروع طراحی، قبل از فرآیند سیم‌کشی و جایگذاری بلوک‌ها، برای تصحیح و تأیید روابط منطقی صورت می‌پذیرد . شبیه‌سازی زمانی[۴] کامل بعد از سیم‌کشی و جایگذاری انجام می‌گیرد، در این حالت نرم‌افزار تأخیرهای ناشی از سیم‌کشی و المان‌ها را در Netlist برای شبیه‌سازی منظور می‌کند.

۴) برنامه‌ریزی نهایی طرح

بعد از ایجاد یک فایل برنامه‌نویسی باینری، تراشه منطقی قابل برنامه‌نویسی پیکره‌بندی شده و آماده کار می‌باشد. روش برنامه‌ریزی نهایی به فنّاوری بکار رفته در تراشه بستگی دارد. در بیشتر فنّاوری‌هایی که دارای PROM جهت بارگذاری FPGA بر اساس SRAMمی‌باشند، به‌نوعی به پروگرامر نیاز است.

۳-۷) شرکت‌های تولیدکننده تراشه منطقی قابل برنامه‌نویسی

عمده‌ترین تولیدکنندگان تراشه‌ها با ظرفیت بالا عبارت‌اند از:

1-Altera
2-Xilinx
3-Vantis
4-Actel
5-Atmel
6-Lattice Semiconductor
7-Lucent Technologies
8-Cypress Semiconductor
9-QuickLogic

FPGA ها اولین بار در سال ۱۹۸۵ توسط شرکت Xilinx معرفی شدند. از آن زمان به بعد FPGA های متفاوتی توسط شرکت‌های دیگر تولید گردید. ازآنجایی‌که در این آزمایشگاه از تراشه‌ها شرکت‌های Xilinx و Altera استفاده‌شده است، لذا لازم است تاکمی با محصولات این شرکت‌ها آشنا شویم.

تراشه‌های شرکت Xilinx برای طراحی‌هایی که خیلی پیچیده نمی‌باشند ولی به ظرفیت بالایی نیاز دارند بسیار مفید و مؤثر خواهد بود. این شرکت، تولیدات خود را به دو گروه عمده زیر دسته‌بندی کرده است:

☞ **Spartan FPGA**
✓**Spartan-3**

☞ **Virtex FPGA**
✓**Virtex-II**

۳- Placement
۴- Binary
۵- Verification
۱- Timing Simulation

✓**Spartan-II 2/5v**

✓**Spartan-II E 1/8v**

✓**Virtex-II Pro X**

✓**Virtex-E 1/8v**

✓**Virtex-4**

شرکت Altera یکی از پیشگامان جهانی تولیدکننده طرح‌های منطقی قابل‌برنامه‌ریزی (PLDs)[۱] و پیاده‌سازی طرح بر روی تراشه‌های قابل‌برنامه‌ریزی (SOPC)[۲] می‌باشد . این تراشه‌ها دارای عملکردی بالا ،با ابزارهای پیشرفته نرم‌افزاری، پردازشگر، حافظه و دیگر المان‌های منطقی پیچیده می‌باشند.

این شرکت تراشه‌های FPGA خود را به سه قسمت عمده تقسیم می‌کند:

- **FPGAهای پرظرفیت[۳]:**
  - **Stratix**
  - **Stratix II**
  - **Stratix GX**
  - **APEXTM_ II**
  - **APEX 20K**
  - **Mercury**
- **FPGAهای کم‌هزینه و با تولید انبوه[۴]:**
  - Cyclone
  - **Cyclone II**
  - **ACEX 1K**
  - **FLEX 6000**
- **FPGAهای با ظرفیت متوسط[۵]:**
  - **FLEX 10K**

۷-۴)ابزارهای طراحی ارزان

همان‌طور که در بخش قبل نیز بیان شد، برای استفاده از قطعات قابل‌برنامه‌ریزی می‌بایست از نرم‌افزارهای مناسب که دارای خصوصیات و بخش‌های ذکرشده هستند، استفاده نمود. ازآنجایی‌که تنوع قطعات قابل‌برنامه‌ریزی زیاد می‌باشد بنابراین طیف وسیعی از نرم‌افزارهای مرتبط با این موضوع نیز در دسترس است. اما با توجه به این‌که در این آزمایشگاه از تراشه‌ها شرکت‌های Xilinx و Altera استفاده‌شده است، لذا در انتهای این دستور کار با دو بورد از شرکت آل ترا (ص ۱۲۱) و شرکت زایلینکس (ص۱۲۸) آشنا خواهید شد.

۲- Programmable logic devices

۱- System-on-a-programmable-chip

۲- High-Density

۳- High-Volume

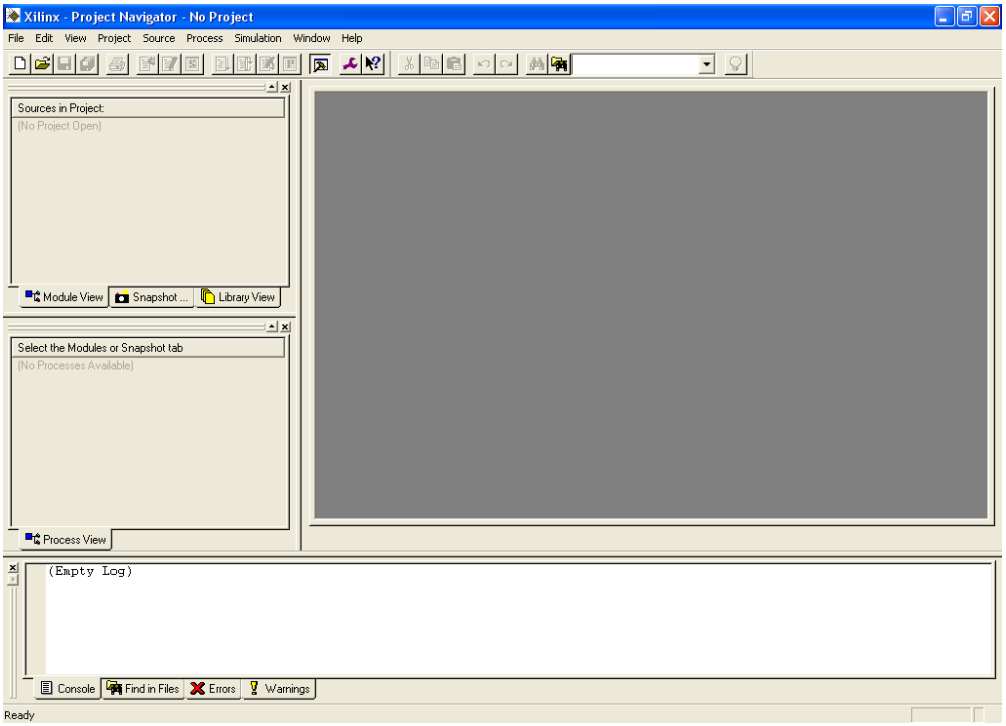
۴- Mid-Density

معرفی ISE

مقدمه

شما در این دوره چگونگی طراحی توسط نرم‌افزار Xilinx ISE را به‌طور دقیق , با جزئیات کامل می‌آموزید و یاد می‌گیرید چگونه با انواع روش‌های طراحی با تراشه‌های شرکت Xilinx کارکنید . روش‌های طراحی (design entry method) ذکرشده در اینجا شامل طراحی به‌وسیله نوشتن کد با زبان Verilog/VHDL ، طراحی شماتیکی با محیط ECS و طراحی به‌وسیله دیاگرام حالت با محیط StateCAD می‌باشد. پس مراحل زیر را با آرامش خاطر دنبال کنید . . .

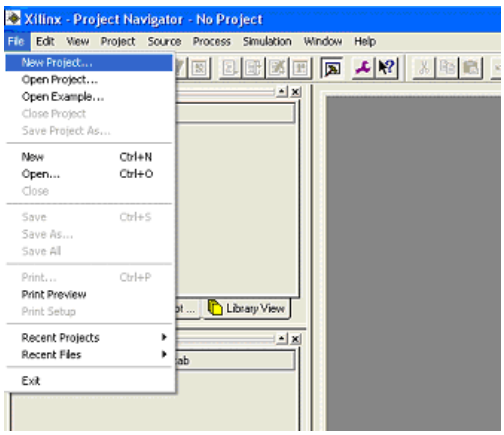
شکل زیر محیط کلی این نرم‌افزار را نشان می‌دهد .



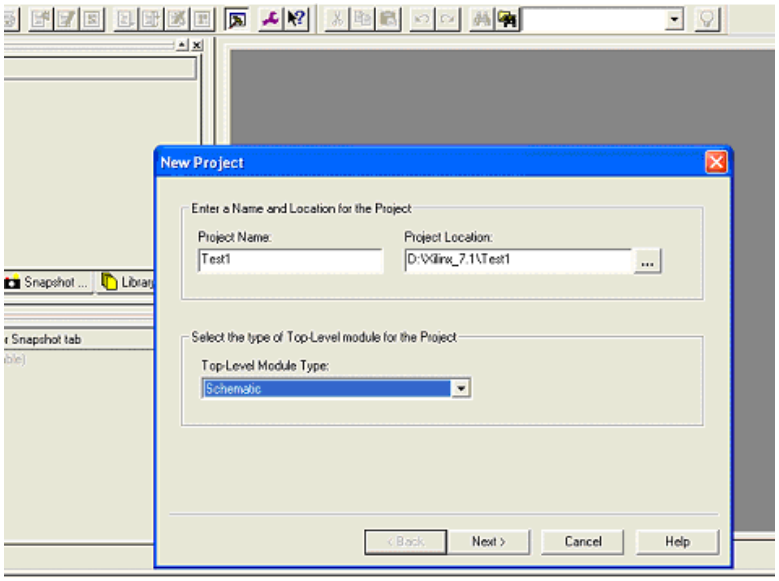
شکل ۷-۷: محیط نرم‌افزار ISE

ایجاد یک پروژه جدید

گزینه File>New Project را انتخاب نموده و اسم پروژه خود را در جای مربوطه وارد کنید.

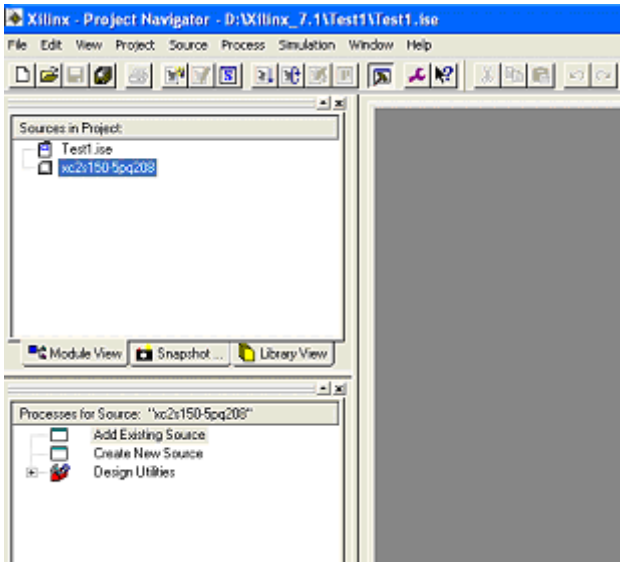


شکل ۷-۸: ایجاد پروژه



شکل ۷-۹: ادامه ایجاد پروژه

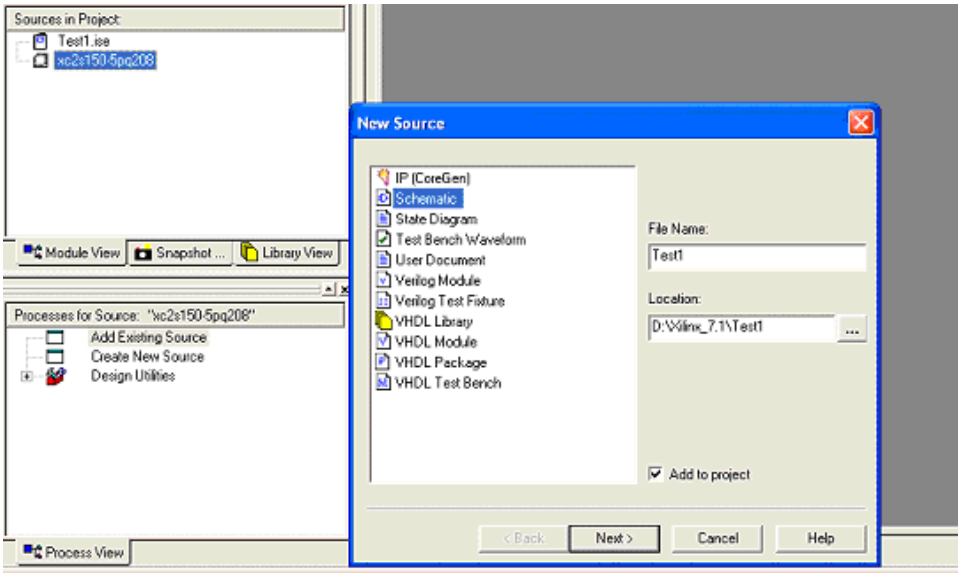
گزینه Next را کلیک کرده , نوع و شماره IC خود را به همراه دیگر مشخصات آن و نیز زبان برنامه‌نویسی را انتخاب نمایید. بقیه پنجره‌ها را به‌صورت پیش‌فرض بپذیرید و در انتها Finish را کلیک نمایید. درنهایت پنجره نرم‌افزار شما به‌صورت زیر درخواهد آمد.



شکل ۷-۱۰: پنجره نرم‌افزار پس از ایجاد پروژه

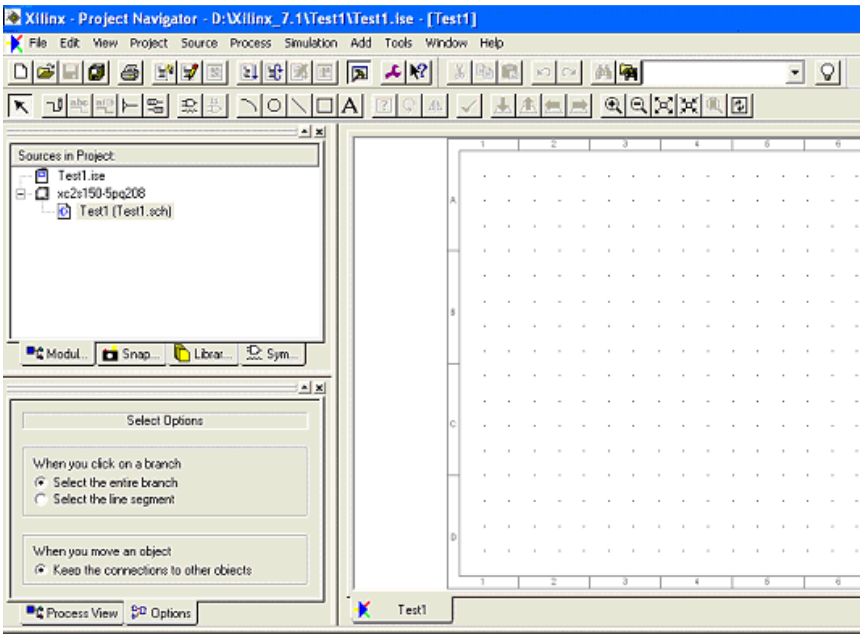
اضافه کردن فایل شماتیک جدید به پروژه

در Project Navigator گزینه Project>New Source را انتخاب کنید .



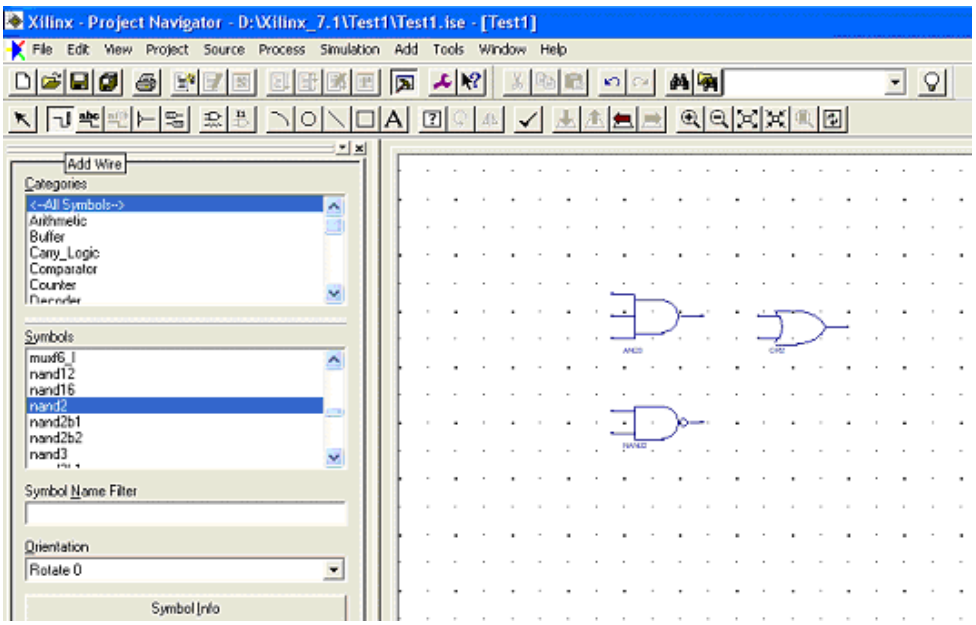
شکل ۷-۱۱: افزودن فایل شماتیک

گزینه Next و سپس Finish را بزنید . محیط طراحی شماتیک بانام فایل خودتان ایجاد می‌گردد. درصورتی‌که بخواهید محیط کد نویسی وریلاگ برای شما باز شود باید از این قسمت گزینه Verilog Module را انتخاب کنید و باقی مراحل شبیه‌سازی و پیاده‌سازی مشابه است.



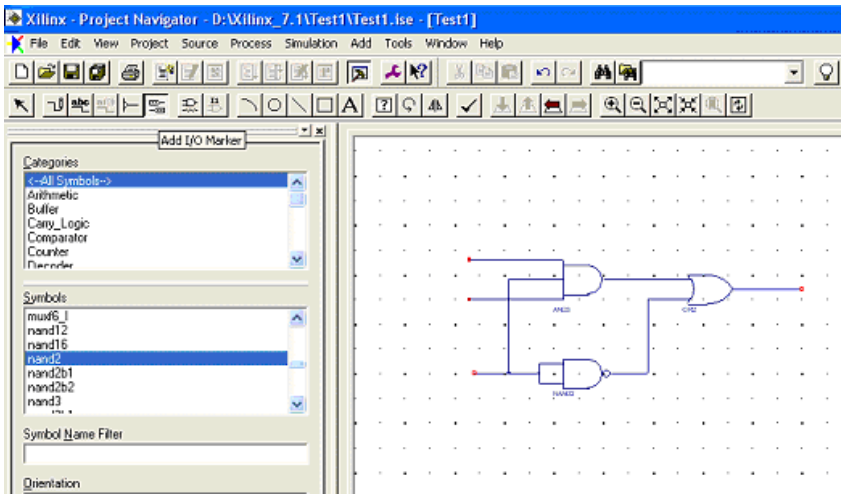
شکل ۷-۱۲: محیط طراحی شماتیک

برای انتخاب المان از دکمه Add Symbol  استفاده نمایید. اکنون پنجره شما به ترتیب زیر خواهد بود. سمبل and3 که همان گیت and با سه ورودی است انتخاب نموده و سپس روی صفحه شماتیک کلیک کنید. سعی کنید شکل زیر را بسازید .



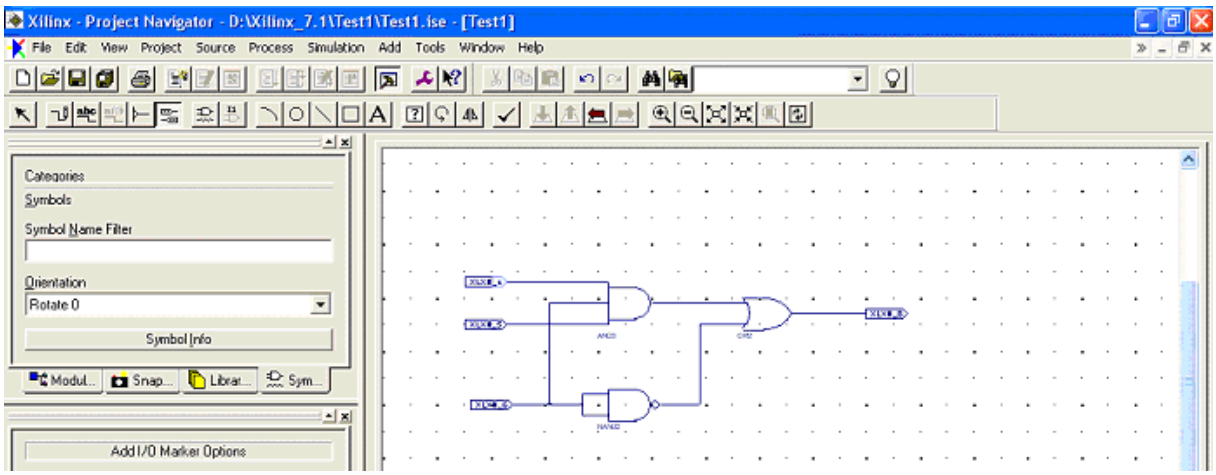
شکل ۷-۱۳: جایگذاری قطعات

حالا باید Wiring (سیم بندی) مدار را انجام دهید. پس دکمه Add Wire را انتخاب و با کلیک کردن روی هر نقطه شماتیک مشغول Wiring شوید. حالا Wiring شماتیک شما به‌صورت زیر درآمده‌است .



شکل ۷-۱۴: سیم بندی

حالا باید سیگنال‌های ورودی و خروجی را به‌صورت یک پورت تعریف نمایید . پس دکمه Add I/O Marker را انتخاب و در قسمت Add I/O Marker Option گزینه Add an Input Marker را کلیک کنید. حالا روی سیگنال‌های ورودی کلیک کنید . برای تعریف پورت خروجی نیز گزینه Add an Output Marker را کلیک کنید و روی سیگنال خروجی کلیک کنید . درنهایت پورت‌ها به‌صورت شکل زیر به شماتیک اضافه می‌شوند .

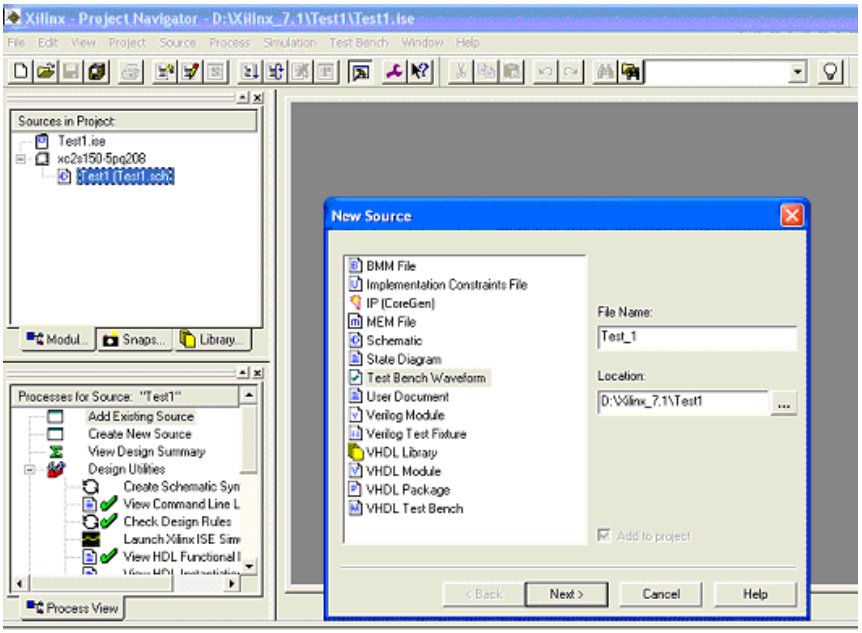


شکل ۷-۱۵: افزودن سیگنال های ورودی و خروجی

برای نام‌گذاری پورت‌های ورودی و خروجی دکمه Add Net Name را انتخاب ، اسم پورت‌ها را وارد و روی پورت موردنظر کلیک کنید.

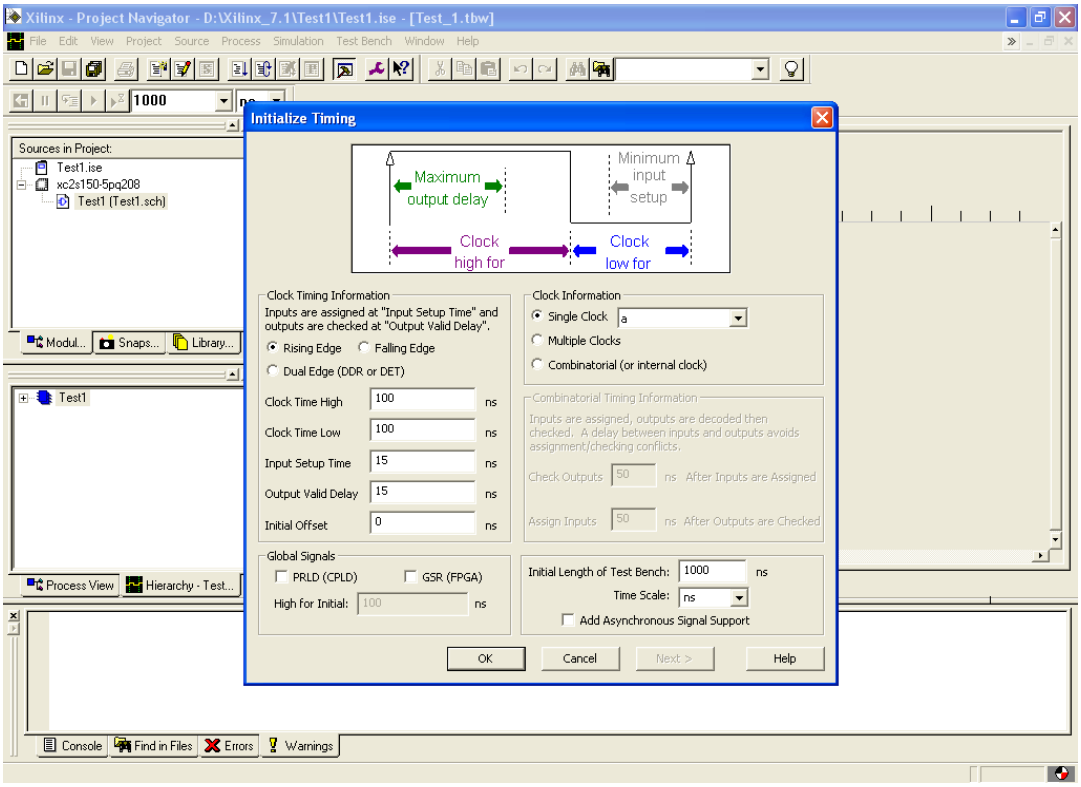
حالا باید شماتیک خود را ذخیره کرده و سپس توسط دکمه  چک نمایید تا دارای اشکال نباشد. پس‌ازاینکه از شماتیک خود مطمئن شدید، از محیط شماتیک خارج‌شده و به محیط اصلی نرم‌افزار بازگردید.

برای شبیه‌سازی مدار خود باید فایل دیگری بانام Test Bench Waveform را اضافه کنید. این فایل جدید برای شبیه‌سازی فایل اصلی استفاده می‌گردد .



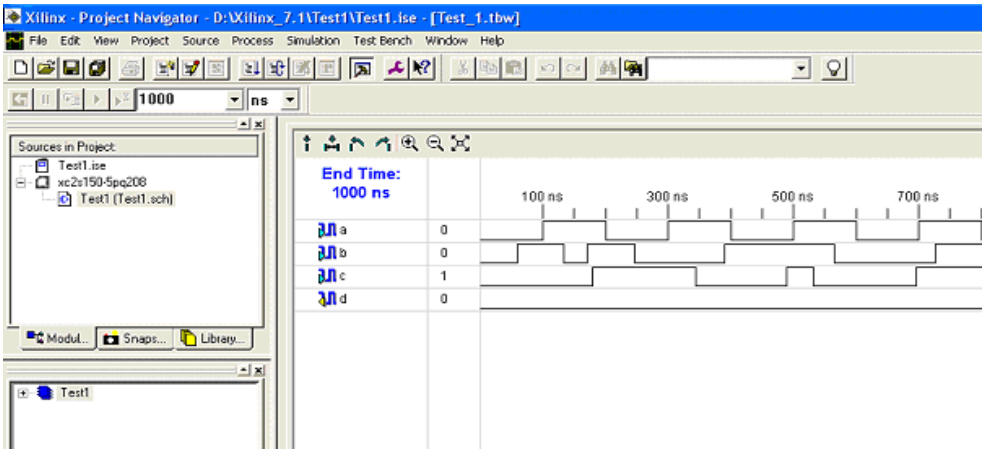
شکل ۷-۱۶: افزودن فایل جدید

سپس محیط تولید شکل موج برای پورت‌های ورودی ظاهر می‌شود .



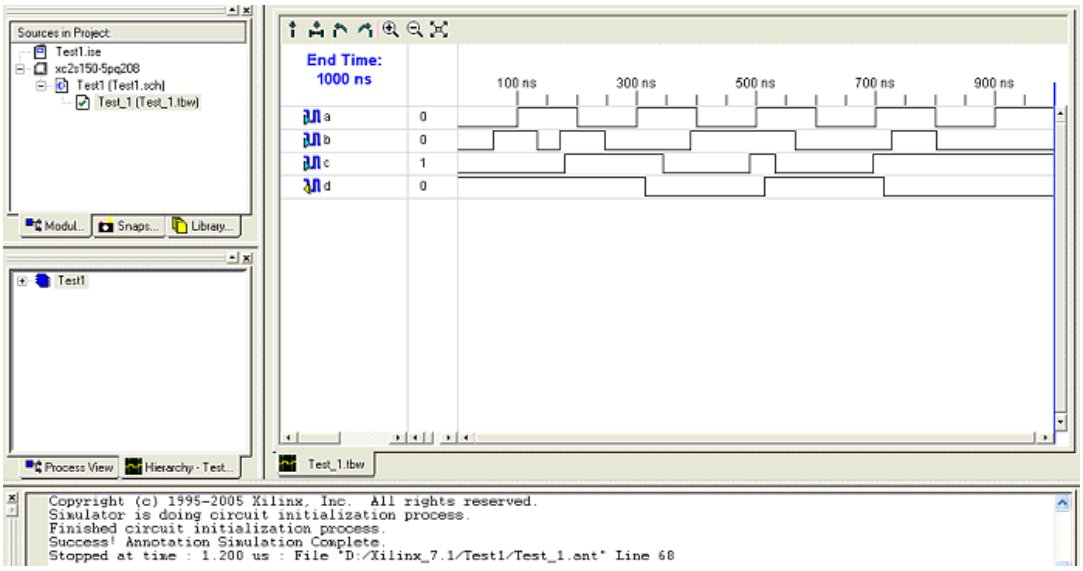
شکل ۷-۱۷: محیط تولید شکل موج

گزینه ADD Asynchronus Signal Support را انتخاب نموده و گزینه Next را کلیک نمایید. حالا یک سیگنال را به‌عنوان پالس ساعت تعریف کنید. گزینه Asynchronous Signal را انتخاب نموده تا بتوانید هرجایی از شکل موج را که می‌خواهید , تغییر دهید و به دنبال آن سیگنال‌های باقیمانده را جهت مقداردهی Add کنید. پنجره زیر ظاهر می‌شود حالا می‌توانید با کلیک ماوس هرجایی از سیگنال را که بخواهید مقدار یک و یا صفر بدهید. مثلاً می‌توانید شکل موج زیر را در نظر بگیرید .



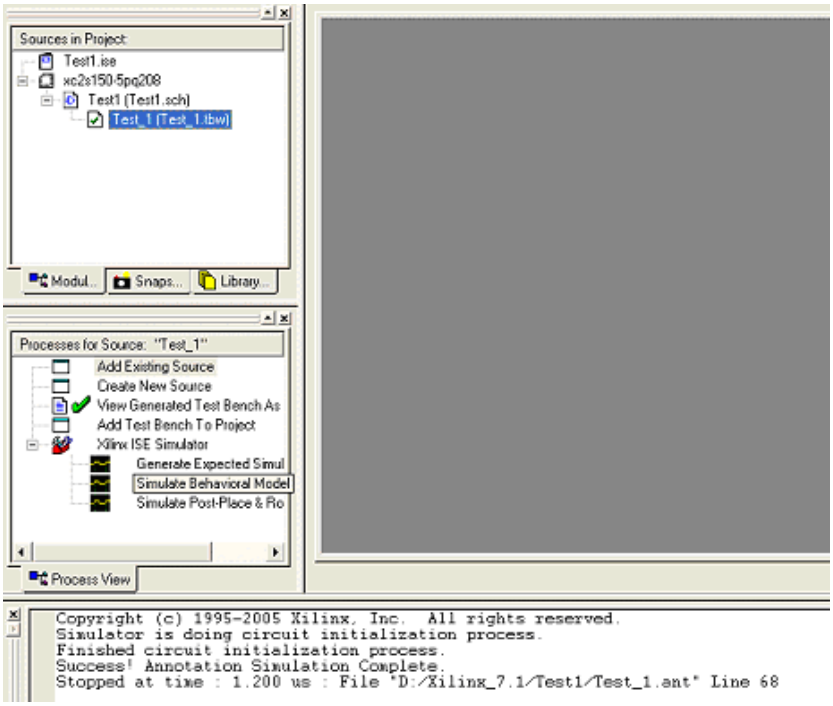
شکل ۷-۱۸:پالس ساعت

این فایل را ذخیره نموده و از این محیط خارج‌شده و به محیط اصلی بازگردید. در محیط اصلی روی فایل Test Bench Waveform کلیک کرده تا HighLight شود و سپس از پنجره Process view گزینه Generate Expected simulation result را دو بار کلیک کنید. پنجره محیط قبلی مجدد باز می‌شود با این تفاوت که سیگنال خروجی بر اساس شماتیک مقدار یافته است .



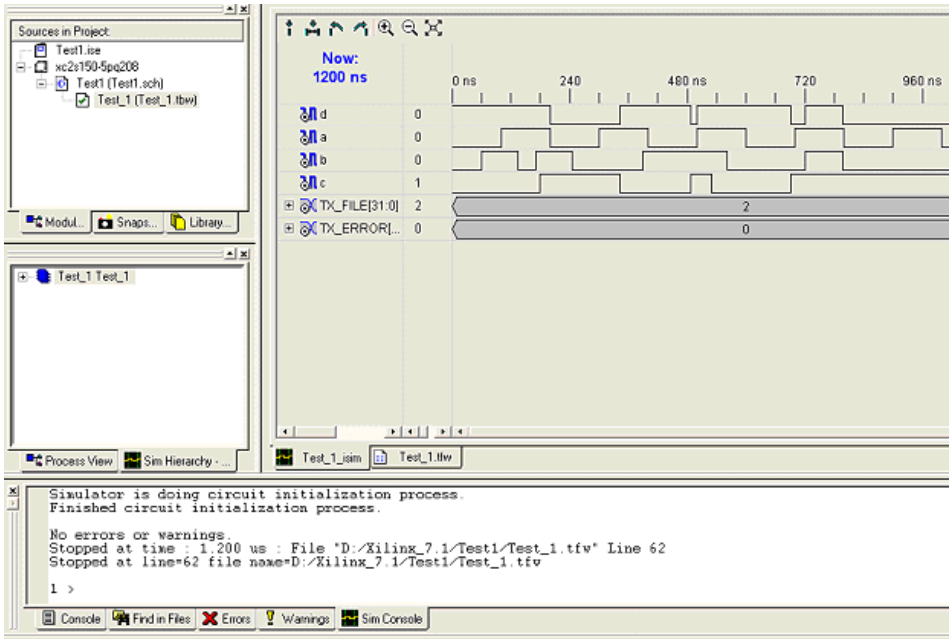
شکل ۷-۱۹: مقدار دهی سیگنال

فایل را ذخیره نموده و خارج‌شده و به محیط اصلی بازگردید . در این حالت دوباره فایل Test Bench Waveform را High Light نموده و از پنجره Process view گزینه Simulate Behavioral Model را دو بار کلیک کنید .



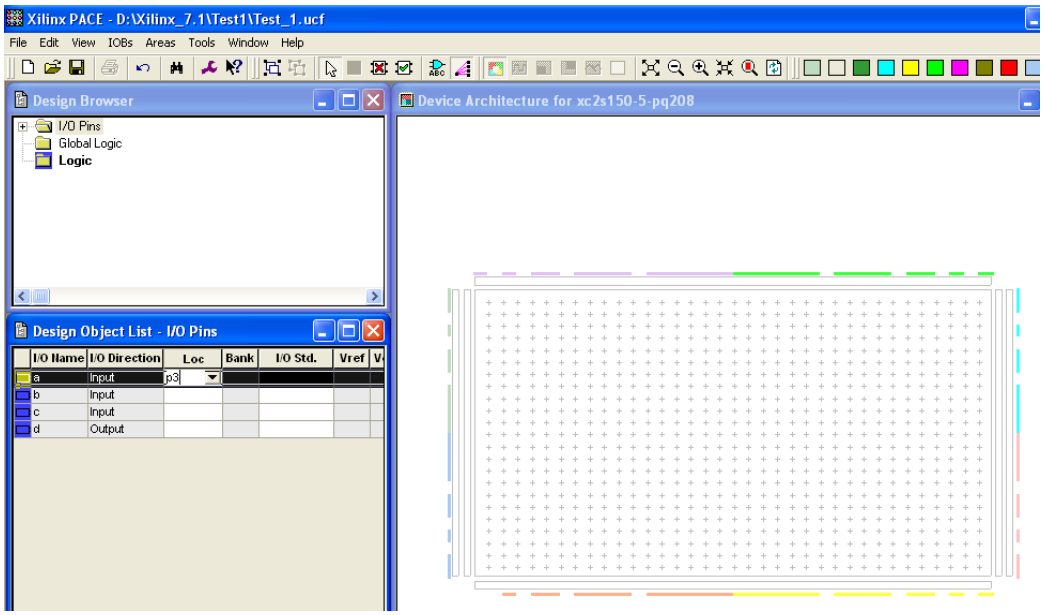
شکل ۷-۲۰: ذخیره فایل

پنجره زیر که نتایج شبیه‌سازی را نشان می‌دهد، ظاهر می‌شود.



شکل ۷-۲۱: نتایج شبیه‌سازی

حالا می‌خواهیم مدار خود را برای پیاده‌سازی روی FPGA آماده کنیم , پس ابتدا باید برای پورت‌ها روی FPGA پایه‌ای در نظر بگیریم . به این کار Pin Assignment می‌گویند . برای همین منظور فایل جدیدی را بانام Implementation Constraint File باید به پروژه اضافه نماییم. این فایل با پسوند *.ucf ساخته می‌شود. این فایل را High Light نموده و از پنجره Process view گزینه Assign Package Pins را انتخاب نمایید. پنجره زیر ظاهر می‌شود که در حقیقت FPGA را با پایه‌های آن نشان می‌دهد.



شکل ۷-۲۲: نمایش FPGA با پایه‌ها

پورت‌های مدار شما نیز در پنجره Design Object List – I/O Pin ظاهرشده‌اند .برای در نظر گرفتن یک‌پایه برای هر پورت ورودی یا خروجی در این پنجره و زیرستون Loc عبارت P را به معنای پین و بعد عدد پایه را مشخص کنید . مثلاً بنویسید : P3

نکته بسیار مهم:

هنگام استفاده از بردهای آموزشی FPGA باید به این نکته بسیار مهم دقت نمود:

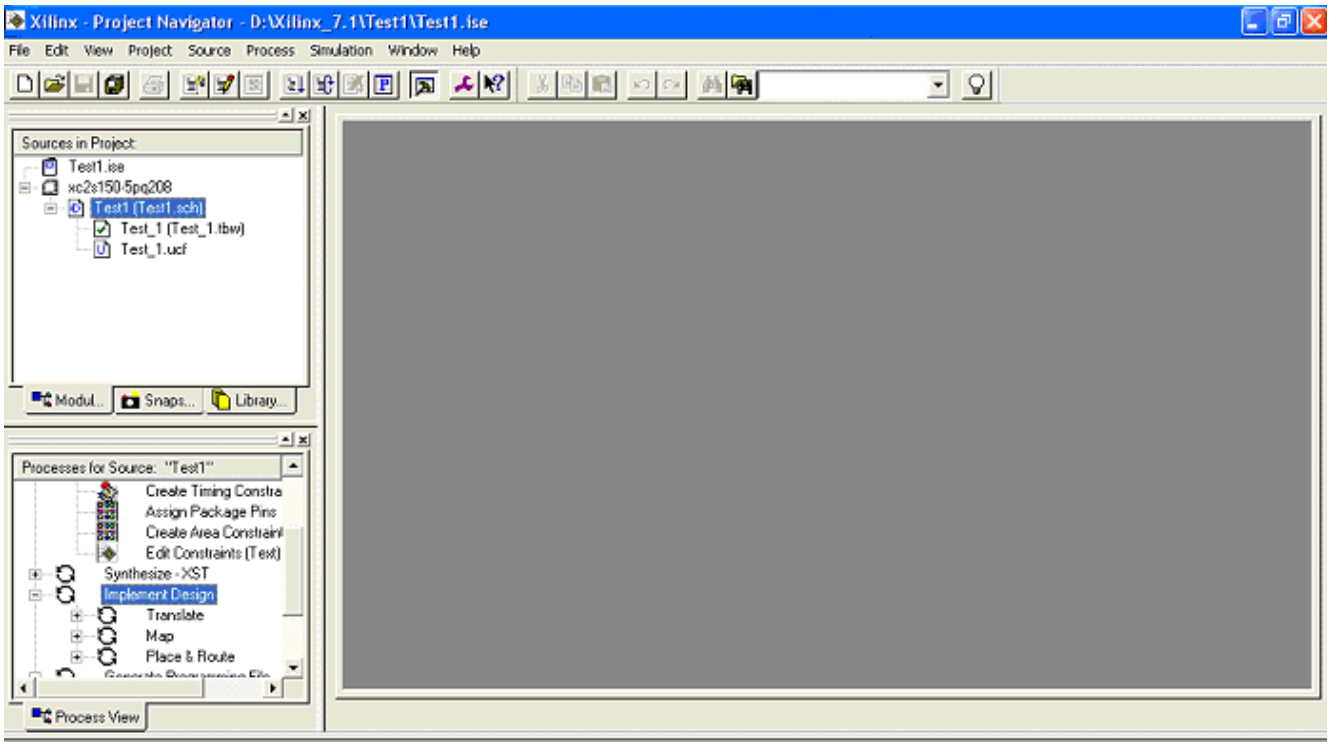
پایه چهارم از سوئیچ سوم (S3) به دو پین مختلف FPGA متصل شده است (p-18, p-204). بنابراین برای عملکرد صحیح این پایه، باید پین 204 از FPGA به‌صورت جعلی مقداردهی شود تا از پین دیگر یعنی p-18 بتوان به‌عنوان ورودی استفاده کرد. برای این کار می‌توان یک سیگنال ورودی (تقلبی) تعریف کرد و به پین204 متصل نمود. این سیگنال ورودی در هیچ جای پروژه استفاده نخواهد شد و هدف از تعریف و اتصال آن به پین 204 این است که پایه چهارم سوئیچ سوم که به پین 18 متصل خواهد شد به‌درستی کار کند.

مثال:

```
Module test(
    …
    input logic Fake,
    …
    );
```

حال این سیگنال ورودی به نام Fake را به پین 204 از FPGA متصل می‌نماییم.

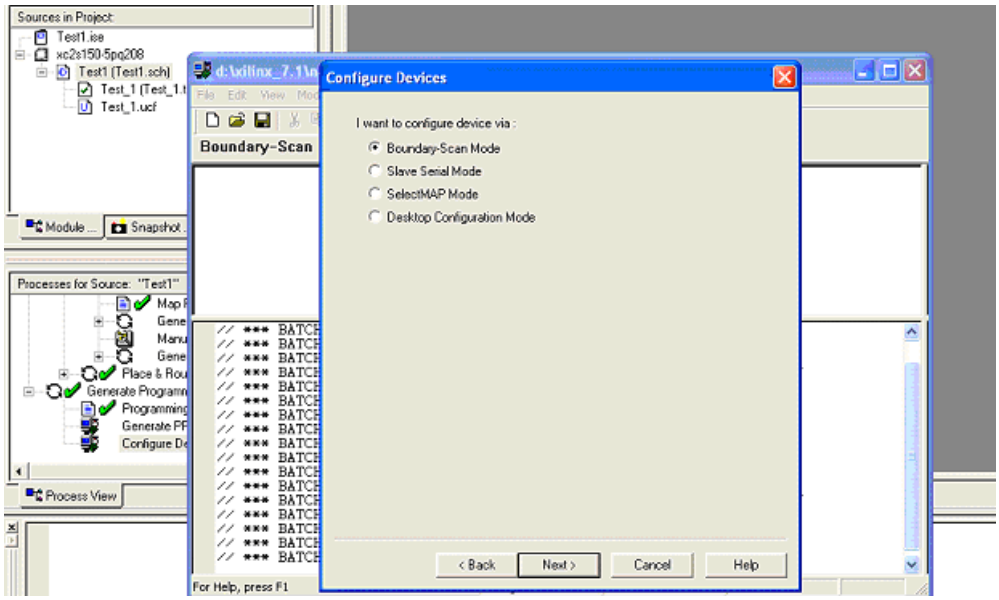
هنگامی‌که برای تمامی پورت‌های ورودی و خروجی پایه‌ای در نظر گرفتید، این فایل را ذخیره نموده و از این محیط خارج شوید. در محیط اصلی نرم‌افزار فایل شماتیک اصلی خودتان را High Light نموده و از پنجره Process view گزینه Implement Design را دو بار کلیک نمایید .



شکل ۷-۲۳: انتخاب گزینه Implement Design

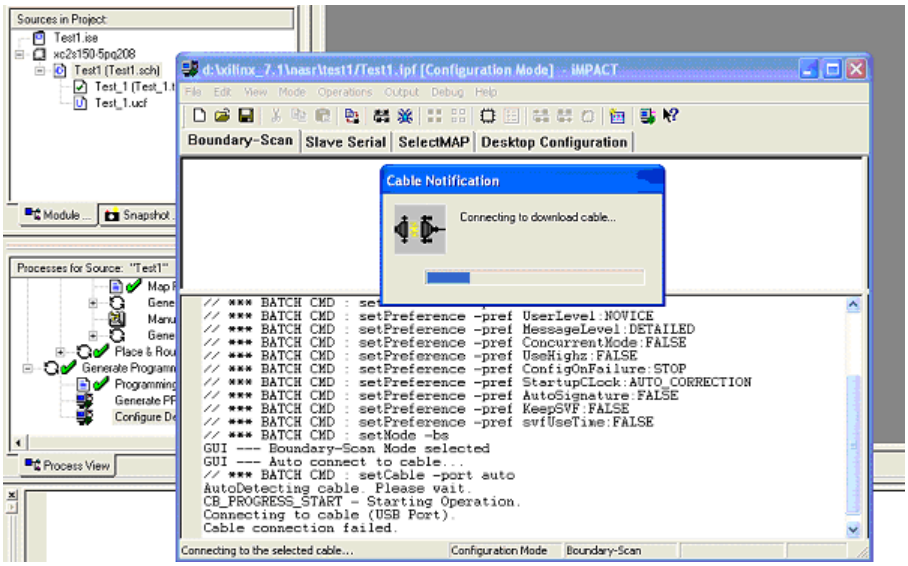
سپس نرم‌افزار مشغول سنتز و پیاده‌سازی فایل شماتیک شما روی FPGA خواهد شد. بعدازآن برای پروگرام نمودن FPGA روی برد , نرم‌افزار باید فایلی را بانام bit.* تولید نماید . این کار را با دو بار کلیک کردن روی گزینه Generate Programming File در پنجره Process view انجام دهید. با کمی دقت، چند جامپر روی برد آموزشی می‌بینید که برای برنامه‌ریزی FPGA باید در وضعیت مای زیر قرار بگیرند: جامپر J17 در وضعیت 1-2 , جامپر J8 در وضعیت 1-2 , جامپر J6 در وضعیت 3-4 و 5-6

حالا نرم‌افزار باید از طریق پروگرامری که به پورت پرینتر کامپیوتر خود وصل کرده‌اید , FPGA روی برد را برنامه‌ریزی نماید . به همین منظور گزینه Configure Device (iMPACT) را دو بار کلیک کنید. گزینه‌ها را به‌صورت پیش‌فرض پذیرفته و Next را کلیک نمایید.

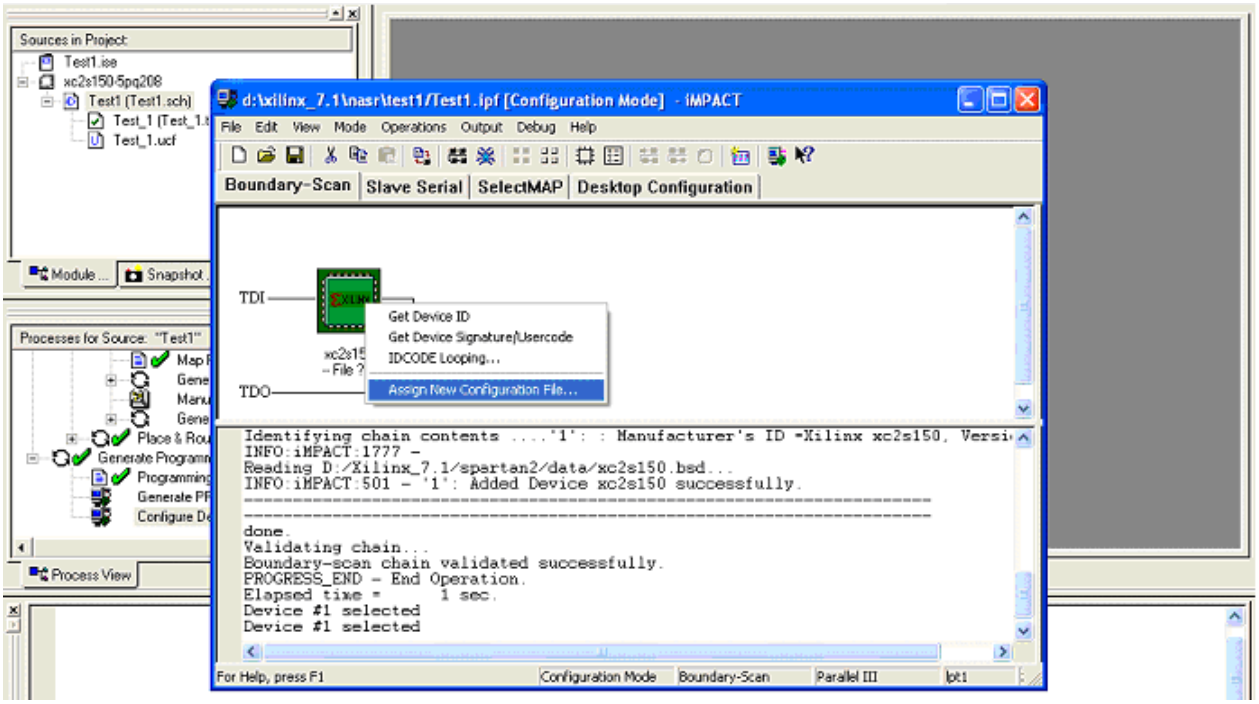


شکل ۷-۲۴: مراحل برنامه ریزی

درنهایت نرم‌افزار به‌طور اتوماتیک کابل برنامه‌ریزی شما را تشخیص و FPGA روی برد را خواهد شناخت. سپس روی FPGA، راست-کلیک نمایید و فایل bit را که در مرحله قبل تولید شده را بازکنید .

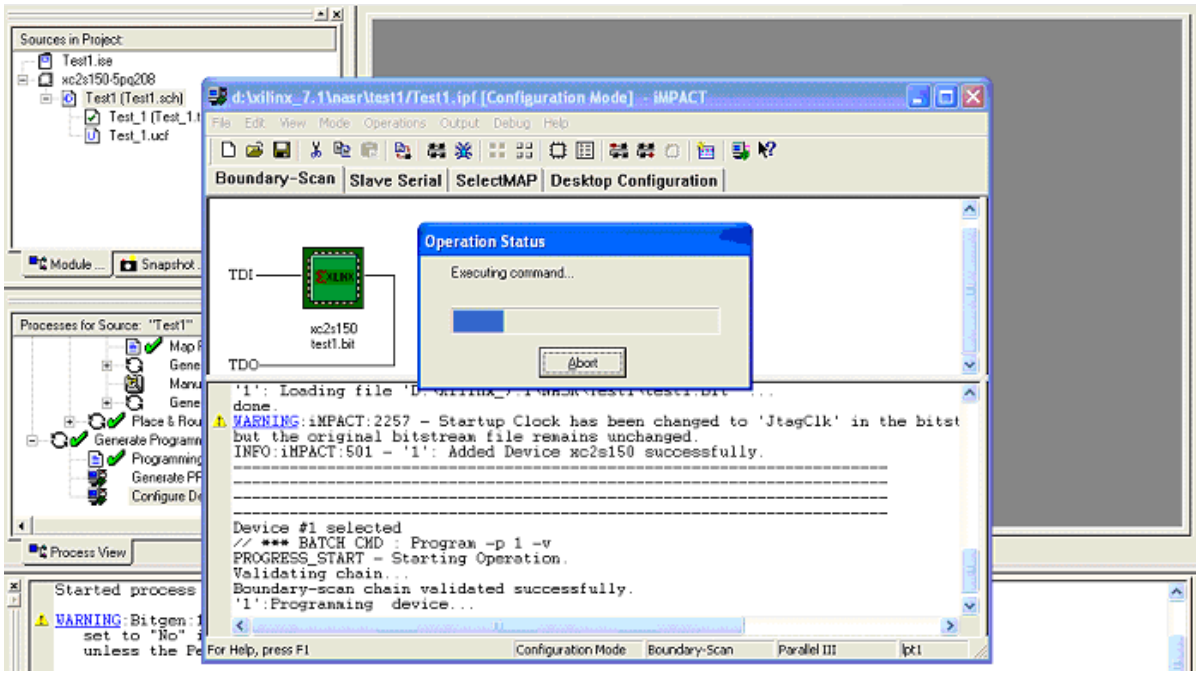


شکل ۷-۲۵: فایل بیت



شکل ۷-۲۶: برنامه ریزی

در این حالت فایل bit شما زیر FPGA شما در نظر گرفته می‌شود . روی FPGA راست – کلیک نموده و گزینه Program را کلیک نمایید. نرم‌افزار شروع به برنامه‌ریزی FPGA از طریق کابل پروگرامر شما خواهد کرد .

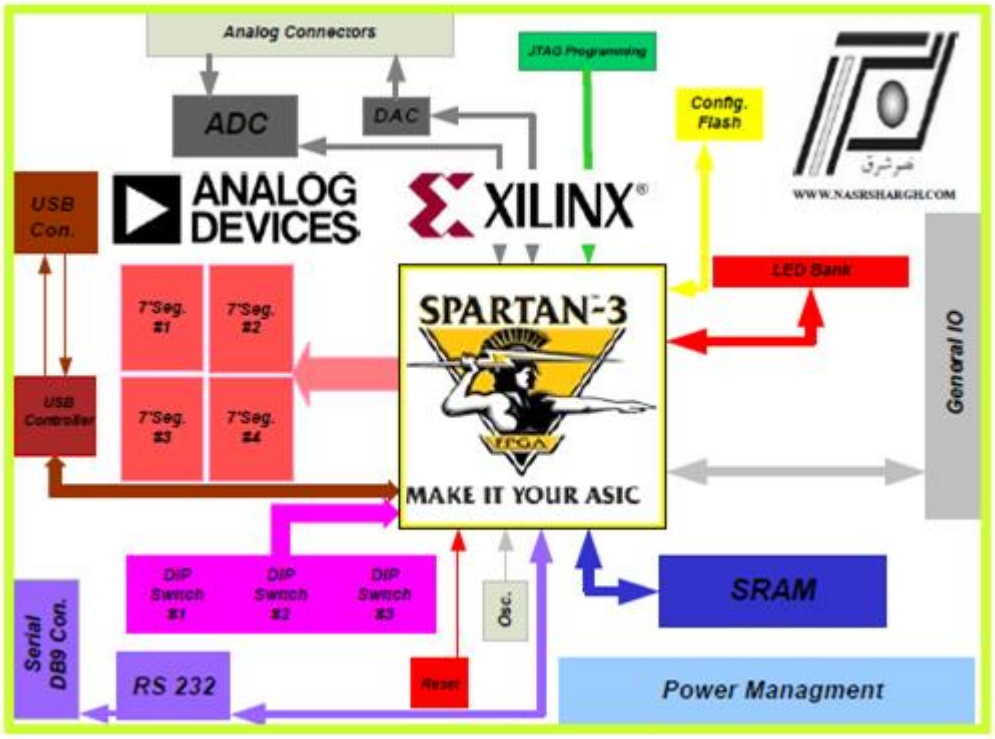


شکل ۷-۲۷: آخرین مرحله برنامه ریزی

حالا شما طرح خود را با موفقیت روی FPGA پیاده‌سازی و برنامه‌ریزی نمودید.

آشنایی با بورد مبتنی بر تراشه شرکت Xilinx

مجموعه آزمایشگاه FPGA از بخش‌های مختلفی تشکیل شده است که این بخش ها شامل برد اصلی, پروگرامر شرکت XILINX و مستندات مربوطه است. در این گزارش فنی شما با ساختار برد اصلی آزمایشگاه و بخش‌های مخلف تشکیل دهنده آن آشنا خواهید شد. شکل ۱ بلوک دیاگرام برد آزمایشگاه را نمایش می‌دهد. به‌طور کلی این برد شامل یک FPGA از شرکت XILINX , منابع تغذیه, قسمت‌های مختلف آنالوگ, دیجیتال و کانکتورها است. در شکل ۲ نیز نحوه و جایگاه قرار گرفتن این بخش ها را روی برد آزمایشگاه مشاهده می نمایید. در ادامه به توضیح قسمت‌های مختلف این برد خواهیم پرداخت.



شکل۷-۲۸: بلوک دیاگرام برد FPGA

تراشه FPGA:

تراشه اصلی یا همان FPGA که روی این برد استفاده‌شده است از سری XC3S400 از شرکت XILINX می‌باشد. همان‌طور که می دانید شرکت XILINX به عنوان یکی از اصلی ترین تولیدکنندگان تراشه‌های الکترونیکی و بخصوص FPGA است. این شرکت تاکنون خانواده های مختلفی از FPGA ها را تولید و به بازار عرضه نموده است. یکی از این سری تراشه ها خانواده Spartan-3 است که در جدول ۱ انواع آن‌ها ازنظر تعداد گیت‌ها, بیت های حافظه تعداد I/O ها و نوع بسته بندی نمایش داده‌شده است. تراشه ای که روی این برد استفاده‌شده است با بسته بندی PQFP و دارای ۲۰۸ پایه است و بیشترین پایه ای که در اختیار کاربر قرار می‌دهد ۱۴۱ عدد است.

Spartan-3 FPGA Family								
Spartan-3	XC3S50	XC3S200	XC3S400	XC3S1000	XC3S1500	XC3S2000	XC3S4000	XC3S5000
Spartan-3L	—	—	—	XC3S1000L	XC3S1500L	—	XC3S4000L	—
Spartan-3 EasyPath	—	—	—	—	XCE3S1500	XCE3S2000	XCE3S4000	XCE3S5000
System Gates	50K	200K	400K	1000K	1500K	2000K	4000K	5000K
Logic Cells	1,728	4,320	8,064	17,280	29,952	46,080	62,208	74,880
Block RAM Bits	72K	216K	288K	432K	576K	720K	1,728K	1,872K
Distributed RAM Bits	12K	30K	50K	120K	208K	320K	432K	520K
DCMs	2	4	4	4	4	4	4	4
Multipliers	4	12	16	24	32	40	96	104
I/O Standards	24	24	24	24	24	24	24	24
Max Single-Ended I/O**	124	173	264	391	487	565	712	784
Package and I/O Offerings								
	XC3S50	XC3S200	XC3S400	XC3S1000	XC3S1500	XC3S2000	XC3S4000	XC3S5000
VQ100 14 x 14 mm	63	63						
TQ144 20 x 20 mm	97	97	97					
PQ208 28 x 28 mm	124	141	141					
FT256 17 x 17 mm		173	173	173*				
FG320 23 x 23 mm			221	221*	221*			
FG456 23 x 23 mm			264	333*	333*			
FG676 27 x 27 mm				391	487*	489		
FG900 31 x 31 mm						565	633*	633
FG1156 35 x 35 mm							712	784

جدول ۷-۱: خانواده های مختلف FPGA

برای برنامه ریزی FPGA روی برد می‌توان از روش JTAG استفاده نمود. پروگرامر XILINX از یک طرف به پورت موازی و از سمت دیگر به کانکتور J7A در روی برد متصل و با استفاده از نرم‌افزار ISE برنامه ریزی یا پیکربندی می‌شود. بر روی برد یک تراشه PROM از سری XCF02S نیز در نظر گرفته شده است که امکان برنامه ریزی FPGA از طریق PROM را نیز مهیا می نماید. به این منظور جامپرهای J5A و J6A را می بایست به ترتیب زیر قرارداد.

J5A: پین ۲ به ۳..... برنامه ریزی از طریق پروگرامر

J6A: پین ۲ به ۳..... برنامه ریزی از طریق پروگرامر

J5A: پین ۲ به ۳..... برنامه ریزی از طریق پروگرامر

J5A: پین ۲ به ۱..... برنامه ریزی PROM توسط پروگرامر

J5A: پین ۲ به ۱..... برنامه ریزی PROM توسط پروگرامر

همچنین جامپر J9A به‌منظور تعیین حالت برنامه ریزی FPGA در نظر گرفته شده است که چگونگی آن را در جدول زیر مشاهده می نمایید.

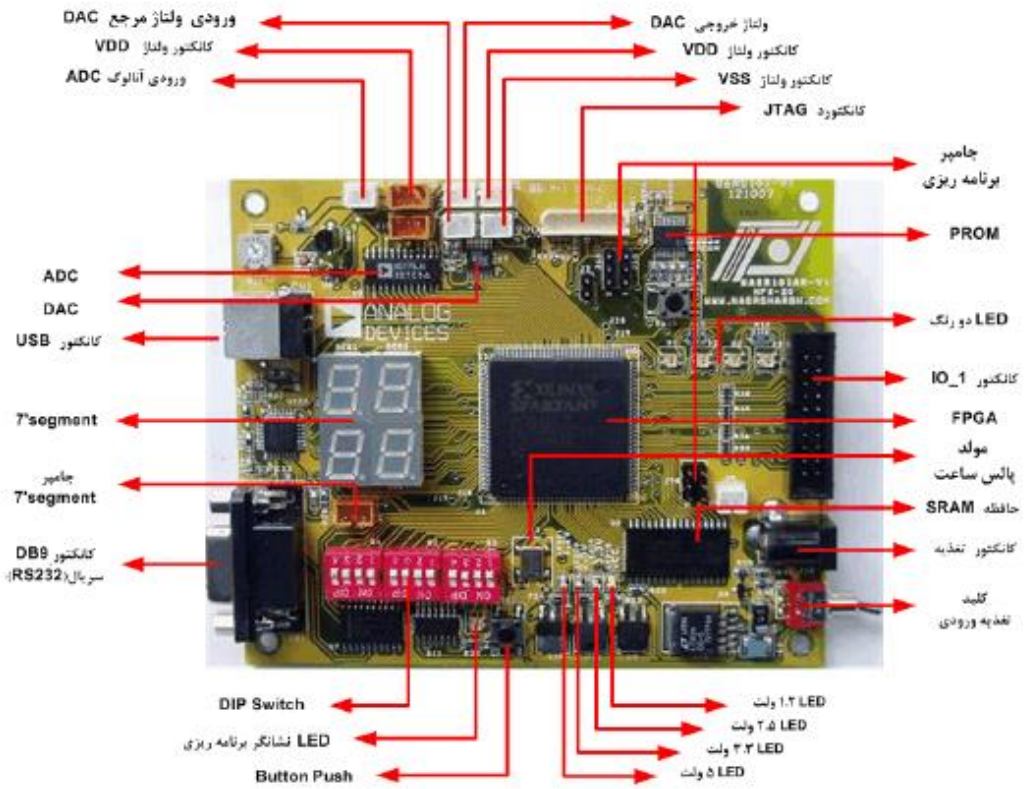
Configuration Mode(1)	M0	M1	M2	Synchronizing Clock
Master Serial	0	0	0	CCLK Output
Slave Serial	1	1	1	CCLK Input
Master Parallel	1	1	0	CCLK Output
Slave Parallel	0	1	1	CCLK Input
JTAG	1	0	1	TCK Input

Notes:
1. The voltage levels on the M0, M1, and M2 pins select the configuration mode.
2. The daisy chain is possible only in the Serial modes when DOUT is used.
3. M0=1: J9A Pin 1 and 2 is connected
   M1=1: J9A Pin 3 and 4 is connected
   M2=1: J9A Pin 5 and 6 is connected

جدول ۷-۲: چگونگی تعیین حالت برنامه ریزی

تغذیه:

تغذیه ورودی برد از طریق کانکتور J12 (کانکتور تغذیه) و کلید تغذیه ورودی تامین و از طریق سه رگولاتور, ولتاژهای ۵ ولت, ۳/۳ ولت و ۲/۵ ولت ساخته می‌شود. LED های نشانگر ولتاژهای ۱/۲, ۲/۵, ۳/۳ و ۵ ولت در قسمت بالای رگولاتورها با شماره های LED 21,20,19,18 برای نشان دادن وضعیت ولتازهای سیستم به کار می روند. محدوده ولتاژ تغذیه ورودی از ۷ تا ۱۲ ولت بوده که بهتر است خروجی یک منبع تغذیه پایدار باشد.



شکل ۷-۲۹: برد اصلی آزمایشگاه

مولد پالس ساعت

یک اسیلاتور با فرکانس 20MHZ روی برد قرارگرفته است که به پایه شماره ۲۲ FPGA متصل می‌باشد و می‌تواند به عنوان clock اصلی برد مورداستفاده قرار گیرد.

حافظه

یک حافظه ۱۲۸ کیلو بیت شرکت SAMSUNG به شماره K6R1008V1C روی برد قرار دارد و باسهای آدرس, داده و سیگنال های کنترلی آن به FPGA متصل شده است (در جدولی که در انتهای این مستند آورده شده است پایه‌های استفاده‌شده مشخص‌شده است).

مبدل آنالوگ به دیجیتال

برای پردازش سیگنال توسط FPGA یک ADC شرکت Analog Device به شماره AD775 روی برد قرارگرفته است. این ADC هشت

بیتی با ولتاژ ۵ ولت تغذیه شده و می‌تواند تا 20MSPS کار نماید. باس داده و سیگنال های کنترلی ADC به پایه‌های FPGA متصل شده است (در جدولی که در انتهای این مستند آورده شده است پایه‌های استفاده‌شده مشخص‌شده است).

مبدل دیجیتال به آنالوگ

برای ارتباط با دنیای آنالوگ و ارائه نتیجه پردازش سیگنال آنالوگ به‌صورت خروجی, از یک DAC شرکت Analog Device استفاده‌شده است. این DAC هشت بیتی با تغذیه ۲/۵ تا ۵ ولت کار می‌کند و باس داده و سیگنال های کنترلی آن به پایه‌های FPGA متصل شده است (در جدولی که در انتهای این مستند آورده شده است پایه‌های استفاده‌شده مشخص‌شده است).

کانکتورهای I/O (ورودی / خروجی)

برای ارتباط با خارج برد یک کانکتور ورودی-خروجی ۲۰ پایه در نظر گرفته شده است و می‌توان از آن برای ارتباط با اجزای جانبی مانند LCD و یا بردهای دیگر استفاده نمود. این کانکتور J14 است.

کانکتور USB

برای ارتباط با پورت USB کامپیوتر کانکتور CN1 در نظر گرفته شده است که در قسمت سمت چپ برد واقع شده است.

ارتباط سری RS-232

دو پایه از FPGA به عنوان خط ارسال و خط دریافت برای ارتباط با پورت COM کامپیوتر که از پروتکل RS-232 استفاده می‌کند در نظر گرفته شده است. این دو پایه از طریق تراشه MAX233 شرکت MAXIM به ولتاژهای RS-232 تبدیل شده و به کانکتور DB9 سری وصل می گردند.

نمایشگر LED و هفت قسمتی

۸ LED در ۴ پکیج بسته بندی شده‌اند, برای نمایش سیگنال های خروجی و نیز چهار عدد نمایشگر هفت قسمتی برای نمایش اعداد و ... روی برد موجود است که به یک سری از پایه‌های FPGA متصل هستند. همچنین یک jamper برای نمایشگر های هفت قسمتی در نظر گرفته شده است تا در حالت آند مشترک و یا کاتد مشترک (بسته به اینکه نمایشگرها چطور کار می کنند) قرار گیرد.

سوئیچ های ورودی وRESET

یک Push Button برای سیگنال های ورودی لحظه ای مانند Reset و ... و سه عدد DIP Switch چهارتایی برای سیگنال های ورودی setting و یا برای ارسال اعداد BCD , باینری و ... روی برد در نظر گرفته شده که به یک سری از پایه‌های FPGA متصل هستند.

جلسه هشتم

طراحی آزمایش ششم و مقایسه آن با مدار رأی گیر اکثریت با استفاده از طراحی شماتیک با ابزار Xilinx ISE بر روی بورد FPGA

جلسه نهم

طراحی مبدل BCD به 7Seg با استفاده از طراحی شماتیک با ابزار Xilinx ISE

جلسه دهم

زبان توصیف سخت‌افزارVerilog

Verilog زبان توصیف سخت‌افزاری است که به‌صورت موازی با VHDL به وجود آمد. شرکت خاصی تولیدکننده آن نیست اما شرکت Cadenceنقش بسیار عمده‌ای در توسعه آن داشته است. یک سازمان به اسم Open Verilog International وجود دارد که وظیفه توسعه و تعریف استانداردها بر عهده اوست. IEEE، Verilog را به‌عنوان زبان توصیف سخت‌افزار استاندارد قبول دارد و هرساله استانداردهای مربوط به آن را انتشار می‌دهد. هنوز هیچ نظر قطعی مبنی بر اینکه کدام‌یک از زبانه‌ای توصیف سخت‌افزار Verilog و یا VHDL کامل‌تر و بهتر هستند وجود ندارد. هر دو بسیار انعطاف‌پذیرند و امکانات زیادی را در اختیار طراح قرار می‌دهند و برای هردو، نرم‌افزارهای فراوان و کتابخانه مای مختلفی توسعه‌یافته است. آموزش زبان Verilog و روش‌های برنامه‌نویسی مربوط به آن خود نیازمند به یک کتاب مجزا است ولی ما در این قسمت با دیدگاهی کاملاً آموزشی طی چند مثال، که روند آسان به سخت دارد، این زبان را بررسی می‌کنیم:

مثال ۱ :

```
module full_adder (S, C, A, B, Cin);
output S, C;
input A, B, Cin;
assign S = A ^ B ^ Cin;
assign C = (A & B ) | (A & Cin) | (B & Cin);
endmodule
```

توضیح: نکته مهم اول اینکه زبان Verilog به بزرگ و کوچک بودن حرف‌ها حساس است. برنامه کوچک بالا یک Full Adder است. در Verilog همه‌چیز به‌صورت ماژول تعریف می‌شود، یعنی اینکه تمام مدارهای منطقی به‌صورت یک جعبه در نظر گرفته می‌شوند که چند ورودی و چند خروجی دارد. این جعبه یک اسم دارد، در برنامه بالا اسم این جعبه یا module را full adder گذاشته‌ایم و سپس تعریف کرده‌ایم که این ماژول چه ورودی‌ها و چه خروجی‌هایی (چه درگاه‌ها یا پورت‌هایی) دارد. در مثال بالا C و S خروجی‌هایی ماژول وA ، B و Cin ورودی‌های ماژول می‌باشند. تمام این پورت‌ها تک‌بیتی هستند، یعنی عرض آن‌ها یک بیت است.

پس از تعیین نام ورودی و خروجی‌ها و جهت هرکدام از آن‌ها می‌رسیم به خودمدار Logic ای که ارتباط بین ورودی و خروجی را به وجود می‌آورد:

در دستور assign اول گفته‌شده که S = A^B^Cin یعنی خروجی S برابر است با xor شده A، B و Cin.

در دستور assign دوم مقدار Cin که درواقع رقم نقلی خروجی است تعیین‌شده. علامت & بیانگر عمل and و علامت | بیانگر عمل or می‌باشد. نهایتاً با دستور endmoduleاعلام می‌کنیم که توصیف مدار داخل ماژول به‌طور کامل انجام‌شده و دیگر چیزی برای نوشتن نداریم. اگر مدار منطقی یک Full adder را در نظر بیاوریم می‌بینیم که عبارت‌های بالا دقیقاً معادل همان مدار هستند. به‌جای کشیدن

شکل گیت‌ها می‌توان عبارات بالا را نوشت. برتری زبان توصیف سخت‌افزار بر روش کشیدن schematicهنگامی معلوم می‌شود که بدانیم برنامه فوق را به‌صورت زیر هم می‌توان نوشت:

مثال ۲:

```
module full_adder (S, C, A, B, Cin);
output S, C;
input A, B, Cin;
assign {C, S} = A + B + Cin;
endmodule
```

در این برنامه مستقیماً گفته‌شده که ترکیب {C,S} به‌عنوان عدد دوبیتی برابر است با حاصل جمع B،A و Cin.

این برنامه وقتی به Synthesizerداده شود، تبدیل به گیت‌های مناسب که عمل جمع را انجام می‌دهند می‌شود. پس یک مدار جمع کننده را به دو روش می‌توان مدل کرد: Gate Level Modeling که در آن خود گیت‌هایی که مدار را می‌سازند مستقیماً بیان می‌کنیم و دوم Behavioral Modelingکه در آن با عبارت‌هایی عملکرد مدار را توصیف می‌کنیم. لازم به ذکر است که درروش دوم در مورد اینکه چه گیت‌هایی لازم است تا این عملکرد ایجاد شود حرفی نمی‌زنیم و آن را به عهده Synthesizerمی‌گذاریم. ازاینجا نقش و اهمیت یک Synthesizer آشکار می‌شود. (قیمت این برنامه‌ها معمولاً گران و بین 15 تا 500 هزار دلار است). علاوه بر Behavioralو Gate Level ، Verilog از Structural Modeling هم حمایت می‌کند. یک نمونه از این مدل‌سازی نیز در مثال زیر می‌آید.

مثال ۳:

```
module two_bit_adder (S, A, B);
output [2:0] S;
input [1:0] A, B;
wire C_b;
full_adder I0 ( S[0], C_b, A[0], B[0], 1'b0 );
full_adder I1 ( S[1], S[2], A[1], B[1], C_b );
endmodule
```

در مدار فوق در داخل ماژول two_bit_adder دو بار یک ماژول دیگر (full_adder که در بالا برنامه‌اش نوشته‌شده است.) استفاده‌شده است Full adderاول بیت‌های اول از پورت‌های دو بیت A و B را دریافت می‌کند (A[0] و B[0]) حاصل جمع را درون S[0] که بیت اول پورت خروجی S است می‌ریزد و بیت انتقالی موجود را به‌وسیله سیمی به نام C_b به Full adder دوم انتقال می‌دهد. می‌بینیم که ورودی پورت C برای Full adder دوم سیم C_b است. از طرفی دو ورودی دیگر A[1] و B[1] هستند که بیت‌های بالایی پورت‌های ورودی A و B می‌باشند. حاصل جمع در Full adder دوم به S[1] و بیت نقلی نهایی در S[2] ریخته می‌شود. در این مثال مدار را به‌صورت Structural مدل کردیم.

مثال :

```
module d_flip_flop (Q, D, clk);
```

```
output Q;
input D;
input clk;
reg Q;
always @(posedge clk)
Q <= D;
endmodule
```

این برنامه یک فیلیپ فلاپ D ساده را نمایش می‌دهد. دقت می‌کنیم که خروجی Qعلاوه بر خروجی، به‌صورت regهم تعریف‌شده است. این به ابزار سنتز می‌گوید که برایQ یک فیلیپ فلاپ در نظر بگیرد. دو خط اصلی برنامه با یک دستور alwaysشروع می‌شود. این دو خط می‌گوید: همواره، در هر لبه بالارونده مقدارD را به Qمنتقل نماید. حالا فرض کنید می‌خواستیم اولاً، این کار در لبه‌های پایین‌رونده انجام شود، ثانیاً مقدار not شده D به Qمنتقل شود، کد Verilog به این صورت تغییر خواهد کرد:

```
module d_flip_flop (Q, D, clk);
output Q;
input D;
input clk;
reg Q;
always @(negedge clk)
Q <= ~D;
endmodule
```

به تغییراتی که نسبت به برنامه بالایی به وجود آمد دقت کنید: اولاً به‌جای posedge از negedge استفاده می‌کنیم. ثانیاً یک علامت not یعنی"~" جلویD اضافه‌شده است. به این مفهوم که مقدار not شده D را دریافت خواهد کرد. حالا فرض کنید می‌خواستیم یک ورودی resetهم برای فیلیپ فلاپ بگذاریم که در هر لبه بالارونده‌ای که مقدار آن 1 شد، خروجی فیلیپ فلاپ برابر با صفر شود:

```
module d_flip_flop (Q, D, clk);
output Q;
input D;
input clk;
reg Q;
always @(posedge clk)
if ( reset )
Q <= 0;
else
Q <= D;
endmodule
```

حالا فرض کنید بخواهیم این reset آسنکرون باشد، یعنی اینکه نیازی به لبه بالارونده ساعت برای صفر کردن خروجی نداشته باشد و هر زمان خودش یک شد، خروجی را صفر کند، بلوک اصلی برنامه به این صورت می‌شود: (بقیه برنامه همانطورکه بوده مِهم‌اند.)

```
always @(posedge clk or posedge reset)
if ( reset )
Q <= 0;
else
Q <= D;
```

همواره در هر لبه بالارونده ساعت و یا هر لبه بالارونده reset هرکدام که روی داد، کد داخل بلوک always اجرا می‌شود : اگر reset بالا باشد در خروجی صفر و در غیر این صورت مقدار D روی Q می‌رود.

```
module four_bit_mux (mux_out, mux_in, mux_control, clk);
output mux_out;
input [3:0] mux_in;
input [1:0] mux_control;
input clk;
reg mux_out;
wire mux_out_w;
assign mux_out_w = (mux_control == 2'b00) ? mux_in[0] :
      (mux_control == 2'b01) ? mux_in[1] :
      (mux_control == 2'b10) ? mux_in[2] :
       mux_in[3];
always @(posedge clk)
mux_out <= mux_out_w;
endmodule
```

در برنامه بالا یک مالتی پلکسر که ۴ ورودی، یک خروجی و یک ورودی کنترل دارد را نشان می‌دهد. خروجی تک‌بیتی است ولی ورودی‌ها به‌صورت بردار (Bus) انتقال داده تعریف‌شده‌اند و هرکدام بیشتر از یک بیت عرض دارند. یک سیم به اسم mux_out_w تعریف‌شده که خروجی مدار mux است. خودمدار mux با استفاده از یک دستور assign که مقداردهی را به‌طور شرطی انجام می‌دهد، پیاده شده است. این دستور ۴ سطر برنامه را تشکیل می‌دهد.

شرط‌ها در پرانتزهایی هستند که قبل از علامت ? قرار دارند. علامت : به معنای else است. این روش شرطی کردن، دقیقاً مطابق با آن چیزی است که در زبان برنامه‌نویسی C وجود دارد. درنهایت با استفاده از یک بلوک always خروجی مدار mux با لبه‌های بالارونده ساعت روی پورت mux_out قرار می‌گیرد. درواقع آن بخشی از مدار که با استفاده از دستور assign پیاده می‌شود، معادل با یک مدار ترکیبی است و آن بخشی که در آن به register ها مقدار داده می‌شود، معادل با یک مدار ترتیبی است. باید دقت نمود که برنامه‌نویسی Verilog با بقیه زبانه‌ای برنامه‌نویسی تفاوت‌های محسوس دارد. در Verilog امکان دارد تمام خط‌های یک برنامه باهم و به‌صورت موازی در حال اجراشدن (پیاده شدن) باشند.

برنامه بالا را با دستور case هم می‌توان پیاده کرد:

```
module four_bit_mux (mux_out, mux_in, mux_control, clk);
output mux_out;
input [3:0] mux_in;
input [1:0] mux_control;
input clk;
reg mux_out;
always @(posedge clk)
case ( mux_control )
2'b00 : mux_out <= mux_in[0];
2'b01 : mux_out <= mux_in[1];
2'b10 : mux_out <= mux_in[2];
2'b11 : mux_out <= mux_in[3];
default : mux_out <= mux_out;
endcase
endmodule
```

برنامه بالا با استفاده از دستور case، در هر لبه بالارونده وضعیت ورودی mux_control را چک می‌کند و برای هر مقدار از mux_control ورودی مناسب را روی خروجی می‌گذارد. اگر هیچ‌کدام از چهار مقدار ذکرشده در دستور caseدر ورودی mux_control نباشد (مثلاً mux_control برابر با 2'bzz امپدانس بالا باشد.) آن‌وقت کد نوشته‌شده در بخش default اجرا خواهد شد. یعنی اینکه mux_out مقدار قبلی خود را حفظ خواهد کرد و مقدار آن تغییر نخواهد کرد.

مثال :

```
module ram_interface (address_out, data, rnw, reset, clk);
output [15:0] address_out;
inout [15:0] data;
output rnw;
input reset, clk;
reg [15:0] address_out;
reg rnw;
reg [15:0] access_address;
reg read_write_turn;
reg [15:0] data_in_register;
assign data = (! rnw ) ? 0 : 16'bz;
always @(posedge clk)
if ( reset ) begin
address_out <= 0;
rnw <= 1;
end
else begin
if ( rnw ) begin
data_in_register <= data;
```

```
access_address <= access_address + 1;
end
if ( data_in_register == 16'hff0f )
rnw <= ~ rnw;
if (! rnw )
rnw <= ~ rnw;
end
endmodule
```

کار این برنامه این است که محتوای خانه مای یک ram را از آدرس صفر به ترتیب می‌خواند، و چک می‌کند که آیا مقدار آن خانه برابر با عدد 0xff0f هست یا نه. اگر جواب مثبت باشد، مقدار آن خانه از ram را صفر می‌کند و ادامه می‌دهد، اگر جواب منفی باشد، ادامه می‌دهد. با هر resetکار دوباره از آدرس صفر شروع می‌شود، وقتی‌که رسیدیم به آخر حافظه دوباره به بالا برمی‌گردیم. اینجا فرض شده عرض گذرگاه داده برای حافظه 16 بیت است و حافظه دارای 64 K خانه است (برای همین گذرگاه داده هم 16 بیتی انتخاب‌شده است). در آینده خواهیم دید که این برنامه را می‌شود روی یک FPGA پیاده کرد، و FPGA را به ram وصل کرد تا عمل بالا را برای ما روی ram انجام دهد. نکته جدیدی که در برنامه فوق وجود دارد اضافه شدن یک نوع پورت جدید است inout پورتی است که از طریق آن داده هم می‌تواند داخل شود و هم به خارج برود. دقت کنید که چگونه مدیریت داده مربوط به این پورت انجام‌شده است. هر زمانrnw (read not write) برابر با ۱ باشد، یعنی قرار باشد از ram بخوانیم، داده روی پورتdata که ورودی/خروجی است، داخل یک ثبات به اسم data_in_register منتقل می‌شود. از طرفی در بالای برنامه، توسط یک دستور assign به پورت data هنگامی‌که قرار است از این پورت داده به بیرون برود، مقدار اختصاص داده‌شده است که این مقدار برابر با صفر است. دقت کنید که هنگامی‌که data در حالت خواندن است دستور assign مقدار 16'bz را که درواقع امپدانس بالا و مدار قطع است روی data قرار می‌دهد. به‌عبارت‌دیگر در این حالت پورت data را درایو نمی‌کند و هر کس دیگری می‌تواند روی آن داده قرار دهد. نکته دیگری که باید به آن توجه کرد، وجود دستورات begin و end است که بلوک مای کد، که در هنگام درست بودن یک شرط باید اجرا شوند را مشخص می‌کند.

کدهای غیرقابل سنتز

در زبان Verilog، هنگامی‌که در سطح رفتاری برنامه‌نویسی می‌کنیم، ممکن است مداری طراحی نماییم که معادل سخت‌افزاری ندارد. این‌گونه طرح‌های سخت‌افزاری فقط در شبیه‌سازهای HDL قابل‌اجرا می‌باشند و فقط ارزش شبیه‌سازی دارند. برای مثال کد زیر را در نظر بگیرید.

```
module d_flipflop_2clks(clk1, clk2, d, q);
input clk1, clk2, d;
output q;
reg q;
always @(posedge clk1 or posedge clk2)
begin
 q <= d;
```

```
end
endmodule
```

این مدار یک flipflop را که دو clock دارد، نشان می‌دهد، اما هیچ معادل سخت‌افزاری نداشته و هیچ سنتز کننده‌ای نمی‌تواند آن را پردازش نماید و فقط قابل شبیه‌سازی است.

ابزارهای انجام پروژه برای FPGA

کلاً برای هر طرحی که قرار است روی FPGA پیاده شود، باید یک سری کارهای مشخص انجام شود. ابتدا باید مدار اصلی طراحی گردد. در طراحی مدار اصلی معمولاً این‌طور عمل می‌کنند که ابتدا تعداد ماژول‌ها، عملکرد هرکدام و ارتباط آن‌ها باهم را مشخص می‌کنند و سپس به طراحی تک‌تک ماژول‌ها می‌پردازند. به این روش طراحی، روش top to down design گفته می‌شود. یعنی شما ابتدا عملکرد کلی را تعیین می‌کنید و بعد هرکدام از اجزا را به‌دقت توصیف می‌کنید تا نتیجه مطلوب حاصل شود. کلاً کارهایی که تا مرحله آماده شدن کد Verilog برای انجام شبیه‌سازی انجام می‌شود را Design Entry می‌گویند. پس بعد از مرحله Design Entry کدهای Verilog ما آماده هستند. حال به مرحله Functional Simulation می‌رسیم. در این مرحله عملکرد مدار را در حالت ایده‌ال (تأخیر صفر) شبیه‌سازی می‌کنیم تا مطمئن شویم طراحی را درست انجام داده‌ایم. یعنی به ورودی‌های مدار هر دفعه سیگنال‌های متفاوتی می‌دهیم و سپس خروجی را بررسی می‌نمایم تا از صحت خروجی مطمئن شویم. طبیعی است که اگر جواب برای هر یک از مراحل آزمودن درست نباشد دوباره باید به مرحله Design Entry بازگشت و طرح را اصلاح کرد. معمولاً همراه هر ماژول Verilog که می‌نویسیم یک برنامه دیگر (یک ماژول دیگر) به اسم Verilog Test Fixture هم می‌نویسیم. این برنامه کارش این است که سیگنال‌های مناسب را برای ورودی ماژول تحت آزمودن تولید می‌کند. به‌این‌ترتیب عملکرد کلی این است که با استفاده از یکی از نرم‌افزارهای شبیه‌سازی Verilog مجموعه ماژول اصلی و ماژول آزمودن آن، را که به هم وصل شده‌اند شبیه‌سازی می‌کنیم و می‌بینیم که آیا ماژول اصلی درست‌کار می‌کند یا خیر. پس از اطمینان از عملکرد صحیح مدار به مرحله Synthesis می‌رویم. در این مرحله با استفاده از یکی از نرم‌افزارهای Synthesizer مدار را تبدیل به مجموعه‌ای از گیت‌های منطقی می‌کنیم. باز این مجموعه گیت‌ها را می‌شود تبدیل به یک برنامه Verilog و حاصل را شبیه‌سازی نمود. به‌این‌ترتیب مطمئن می‌شویم خروجی ابزار سنتز همان چیزی است که ما می‌خواهیم. تا این زمان به‌عنوان ابزارهای شبیه‌سازی و سنتز از محصولات هر شرکتی که بخواهیم می‌توانیم استفاده کنیم. مرحله آخر آن است که خروجی Synthesizer را به ابزار Implement بدهیم. ابزار Implement فایلی که خروجی Synthesizer است را می‌گیرد و آن را به ترکیبی از المان‌های موجود روی FPGA تبدیل می‌کند. سپس این عنصرها را سر جای مناسب روی FPGA قرار می‌دهد (Placement) و بین آن‌ها را سیم‌کشی می‌کند (Routing) درنهایت یک فایل با پسوند BIT تولید می‌شود که ما می‌توانیم از آن برای Program کردن Romای که قرار است به FPGA وصل شود استفاده کنیم. ابزار Implement هر شرکت مخصوص خودش است، مثلاً برای FPGAهای Xilinx حتماً باید از ابزار Implement خود Xilinx استفاده کرد. در این آزمایشگاه از نرم‌افزارXILINX استفاده می‌کنیم.

جلسه یازدهم

آزمایش طراحی و پیاده‌سازی جمع کننده‌ی ریپل کری

در این جلسه با دو جمع کننده آشنا می‌شویم:

۱-Ripple Carry

هدف :

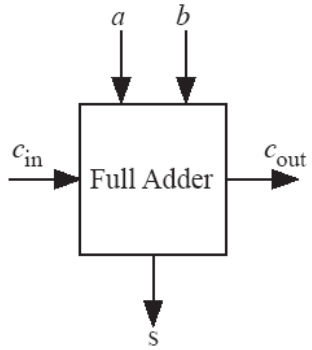
در این آزمایش اهداف زیر دنبال می‌شوند :

- ✓ طراحی و شبیه‌سازی نیم- جمع کننده، تمام جمع کننده و جمع کننده ۴ بیتی و مقایسه آن با تئوری.
- ✓ پیاده‌سازی مدارهای طراحی‌شده بر روی FPGA.
- ✓ آزمودن مدار پیاده‌سازی شده و بررسی خروجی‌های به‌دست‌آمده.

تئوری آزمایش :

همان‌طور که از اهداف آزمایش نیز مشخص است، هدف نهایی در این آزمایش طراحی و پیاده‌سازی جمع کننده ۴ بیتی و ۱۶ بیتی می‌باشد. اما ازآنجایی‌که ساده‌ترین و پایه‌ای‌ترین مدار در طبقه‌بندی مدارهای جمع کننده، مدار نیم- جمع کننده یا Half-Adder است، لذا در ابتدا به طراحی مدار نیم- جمع کننده می‌پردازیم. پس از شبیه‌سازی و پیاده‌سازی این مدار و بررسی خروجی آن به طراحی مدار تمام- جمع کننده خواهیم پرداخت. در این دو بخش اهداف اصلی آشنایی و آزمودن عملکرد این مدارها و همچنین تمرین بیشتر مراحل پیاده‌سازی یک طرح بر روی برد و مشاهده نتایج آن است.

در بخش بعدی با بهره‌گیری مدار تمام- جمع کننده طراحی‌شده (به‌عنوان یک بلوک با ورودی‌های و خروجی‌های مشخص) به طراحی مدار جمع کننده ۴ بیتی خواهیم پرداخت. لازم به ذکر است که در این بخش نیز با توجه به اینکه خروجی نهایی مدار بر روی نمایشگر ۷-قسمتی قابل‌مشاهده می‌باشد، لذا به‌کارگیری و استفاده از بلوک دیکد کننده دودویی به ۷-قسمتی که در آزمایش ۴ طراحی نمودید را نیز به‌عنوان یک بلوک طراحی تجربه خواهید نمود.

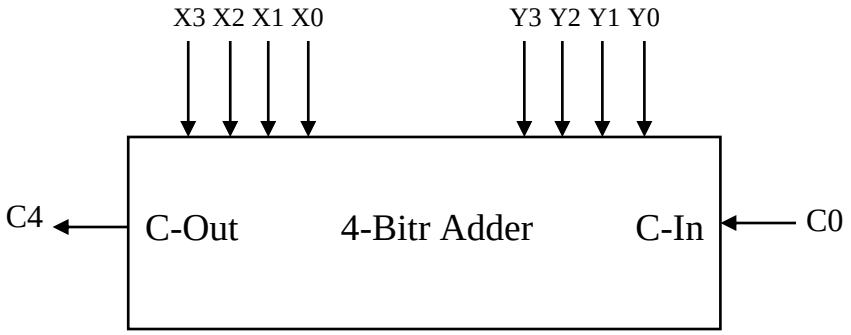


شکل ۱۱-۱: تمام- جمع کننده تک‌بیتی

حال پیش از انجام آزمایش مختصری به تئوری مربوط به مدارهای جمع کننده می‌پردازیم.

همان‌طور که می‌دانید مدار نیم- جمع کننده یا Half-Adder مداری است که دو عدد تک‌بیتی را باهم جمع و حاصل را به همراه بیت نقلی تولید می‌نماید. مدار تمام-جمع کننده یا Full-Adder تک‌بیتی نیز مداری است که دو عدد تک‌بیتی را بابیت نقلی ورودی جمع و مجدداً حاصل را به همراه بیت نقلی تولید می‌نماید (شکل ۱۱-۱).

تمام-جمع کننده را می‌توان به‌عنوان بلوک پایه در طراحی جمع کننده‌های n-بیتی مورداستفاده قرارداد. به‌عنوان‌مثال در طراحی جمع کننده ۴-بیتی از ۴ تمام-جمع کننده استفاده می‌شود. در این جمع کننده دو عدد ۴-بیتی با احتساب بیت نقلی ورودی باهم جمع و حاصل که یک عدد ۴-بیتی و بیت نقلی خروجی است نتیجه می‌شود.



شکل ۱۱-۲: جمع کننده ۴-بیتی

تراشه یا آی‌سی‌هایی که دانشجویان عزیز در این آزمایش مورداستفاده قرار خواهند داد، عبارت‌اند از:

۱. NOT: که با شماره ۷۴۰۴ یا نام NOT (NOT با دو ورودی) در کتابخانه نرم‌افزار قابل‌دسترسی می‌باشد.

۲. AND: که با شماره ۷۴۰۸ یا نام AND2 (ADN با دو ورودی) در کتابخانه نرم‌افزار قابل‌دسترسی می‌باشد.

۳. OR: که با شماره ۷۴۳۲ یا نام OR2 (OR با دو ورودی) در کتابخانه نرم‌افزار قابل‌دسترسی می‌باشد.

۴. XOR: که با شماره ۷۴۸۶ یا نام XOR2 (XOR با دو ورودی) در کتابخانه نرم‌افزار قابل‌دسترسی می‌باشد.

تکالیف پیش از آزمایش :

پیش از ورود به آزمایشگاه و شروع آزمایش، مطالب این بخش را مطالعه نموده، موارد خواسته‌شده را انجام داده و به سؤالات مطرح‌شده پاسخ دهید و نتایج به‌دست‌آمده را در گزارش خود ثبت نمایید.

۱) مراحل پیاده‌سازی نرم‌افزاری و سخت‌افزاری ارائه‌شده در فصول ابتدایی را به‌دقت مطالعه و در صورت لزوم به مستندات نرم‌افزار مراجعه نمایید.

۲) نیم- جمع کننده :

۲-۱) مدار موردنظر را به‌طور اجمالی تشریح و عملکرد آن را توضیح دهید.

۲-۲) جدول درست نمایی مربوط به نیم- جمع کننده را تکمیل نمایید.

۲-۳) با استفاده از گیت‌های NOT، OR و AND مدار یک نیم- جمع کننده را طراحی نموده و توسط نرم‌افزار MAX، شماتیک آن را ترسیم و خروجی آن را پس از شبیه‌سازی با تئوری مقایسه نمایید. آیا جواب به‌دست‌آمده درست است ؟

۲-۴) مدار ترسیم‌شده را با توجه به نمایش خروجی‌ها بر روی LEDها و به‌منظور پیاده‌سازی بر روی برد تکمیل و آماده نمایید.

۳) تمام - جمع کننده :

۳-۱) مدار موردنظر را به‌طور اجمالی تشریح و عملکرد آن را توضیح دهید.

۳-۲) جدول درست نمایی مربوط به مدار تمام- جمع کننده را تکمیل نمایید.

۳-۳) با استفاده از گیت‌های منطقی، مدار یک تمام- جمع کننده را طراحی نموده و توسط نرم‌افزار MAX شماتیک آن را ترسیم و خروجی حاصل از شبیه‌سازی را با تئوری مقایسه نمایید، آیا جواب به‌دست‌آمده درست است؟

۲-۴) مدار ترسیم شده را با توجه به نمایش خروجی‌ها بر روی LEDها و به‌منظور پیاده‌سازی بر روی برد تکمیل و آماده نمایید.

۴) جمع کننده ۴ بیتی :

۴-۱) مدار موردنظر را به‌صورت اجمالی تشریح و عملکرد آن را توضیح دهید.

۴-۲) جدول درست نمایی مربوط به جمع کننده ۴ بیتی را تکمیل نمایید.

۴-۲) با استفاده از مدار تمام-جمع کننده (به‌عنوان یک بلوک طراحی) یک جمع کننده ۴ بیتی طراحی نمایید. ورودی‌های A_0 تا A_3 ، B_0 تا B_3، خروجی‌های S_0 تا S_3، رقم نقلی خروجی و ارتباط داخلی مابین تمام جمع کننده‌ها را مشخص نمایید.

۴-۳) مدار تمام-جمع کننده طراحی‌شده را به‌عنوان یک Symbole تعریف و با استفاده از این قابلیت شماتیک جمع کننده ۴ بیتی را ترسیم و خروجی حاصل از شبیه‌سازی را با تئوری مقایسه نمایید.

۴-۵) به‌منظور نمایش خروجی جمع کننده ۴ بیتی بر روی نمایشگر ۷-قسمتی، از بلوک دیکد کننده‌ای که در آزمایش ۱ طراحی نمودید استفاده و شماتیک نهایی مدار موردنظر را ترسیم نمایید. شماتیک نهایی را به‌منظور پیاده‌سازی بر روی برد آماده نمایید.

تکالیف داخل آزمایشگاه :

۱) نیم جمع کننده :

۱-۱) شماتیک ترسیم‌شده مدار نیم- جمع کننده را توسط نرم‌افزار موردنظر اجرا و مقدمات پیاده‌سازی بر روی برد را مهیا نمایید.

۱-۲) مراحل پیاده‌سازی طرح موردنظر بر روی FPGA را اجرا و نتایج حاصل را بر روی LEDها مشاهده نمایید.

۱-۳) نتایج حاصل از اجرای برنامه بر روی برد را با آنچه قبلاً به دست آورده‌اید، مقایسه نمایید.

۲) تمام - جمع کننده :

۲-۱) شماتیک ترسیم‌شده مدار تمام– جمع کننده را توسط نرم‌افزار موردنظر اجرا و مقدمات پیاده‌سازی بر روی برد را مهیا نمایید.

۲-۲) مراحل پیاده‌سازی طرح موردنظر بر روی FPGA را اجرا و نتایج حاصل را بر روی LEDها مشاهده نمایید.

۲-۳) نتایج حاصل از اجرای برنامه بر روی برد را با آنچه قبلاً به دست آورده‌اید، مقایسه نمایید.

۳) جمع کننده ۴ بیتی :

۳-۱) شماتیک ترسیم‌شده مدار جمع کننده ۴بیتی را توسط نرم‌افزار موردنظر اجرا و مقدمات پیاده‌سازی بر روی برد را مهیا نمایید.

۳-۲) مراحل پیاده‌سازی طرح موردنظر را اجرا و نتایج حاصل را بر روی نمایشگر ۷-قسمتی مشاهده نمایید.

۳-۳) نتایج حاصل از اجرای برنامه بر روی برد را با آنچه قبلاً" به دست آورده‌اید، مقایسه نمایید.

پروژه پیشنهادی :

با استفاده از مدار جمع کننده ۴ بیتی (به‌عنوان یک بلوک طراحی) یک جمع کننده ۱۶ بیتی طراحی نمایید. ورودی‌ها، خروجی‌ها، رقم نقلی خروجی و ارتباط داخلی مابین تمام جمع کننده‌ها را مشخص نمایید. مدار جمع کننده ۴ بیتی طراحی‌شده را به‌عنوان یک Symbole تعریف و با استفاده از این قابلیت شماتیک جمع کننده ۱۶ بیتی را توسط نرم‌افزار ترسیم و خروجی حاصل از شبیه‌سازی را با تئوری مقایسه نمایید. درنهایت شماتیک طراحی‌شده را بر روی برد پیاده‌سازی نمایید و نتایج حاصل را بر روی نمایشگر ۷-قسمتی مشاهده کنید.

جلسه دوازدهم

Carry – Look – Ahead Adder

هدف : در این آزمایش اهداف زیر دنبال می‌شوند:

- ✓ آشنایی با تأخیر در مدارات دیجیتال
- ✓ فراگیری استفاده از Timing Simulator.
- ✓ طراحی جمع کننده Carry Look Ahead ۴ بیتی و محاسبه تاخیرآن.
- ✓ پیاده‌سازی و آزمودن مدار طراحی‌شده

تئوری آزمایش :

در آزمایش گذشته به بررسی نحوه عملکرد مدارهای منطقی ترکیبی پرداختید و به‌منظور ارزیابی عملکرد مدار طراحی‌شده از نرم‌افزار تحلیل‌گر استفاده نمودید. همان‌گونه که می‌دانید در طراحی علاوه بر نحوه عملکرد مدار پارامترهای دیگری نیز در طراحی مدارات دیجیتال وجود دارند که می‌بایست به آن‌ها نیز توجه کرد به‌عنوان‌مثال سرعت مدارهای منطقی در طراحی از درجه اهمیت بالایی برخوردار هستند.

اکنون ممکن است سؤالی در ذهن شما شکل‌گرفته باشد که عامل تأثیرگذار بر سرعت مدارهای منطقی چیست؟ جواب تأخیر قطعات می‌باشد. هر گیت NOT، OR، AND و ... مقدار تأخیری برابر τ خواهد داشت. یعنی با قرار دادن متغیرهای ورودی، بعد از گذشت زمانی برابر τ، خروجی مقدار موردنظر خود را خواهد داشت. البته هر یک از گیت‌های مذکور مقدار تأخیر متفاوتی خواهند داشت. در عمل بایستی مدار را به‌صورتی طراحی نمود که حداقل تأخیر را داشته باشد تا سرعت مدار بالا رود.

عوامل بسیار زیادی در تولید و تأخیر انتشار گیت دخیل می‌باشند. یکی از آن‌ها تأخیر انتشار است که با توجه به ساختار داخلی گیت تعیین می‌شود و عامل دیگر باری است که خروجی گیت در مقابل خود می‌بینند (Fan-Out) و عامل دیگر نحوه طراحی مدار منطقی می‌باشد.

دو عامل اول به ساختار قطعات برمی‌گردد، اما عامل سوم مستقیماً" با نحوه طراحی و پیگیری طراح در ارتباط می‌باشد. در عمل دو مدار که دارای عملکرد یکسان اما با طراحی‌های مختلف می‌باشند، ممکن است دارای سرعت‌های متفاوتی باشند. به‌عنوان‌مثال می‌توان به مدار جمع کننده اشاره داشت. مدار جمع کننده ۴ بیتی که در جلسه پیش طراحی نمودید Ripple–Adder نام داشت و به مقدار قابل‌توجهی کندتر از مداری است که در این جلسه طراحی خواهید نمود. پیش از پرداختن به مقایسه جمع کننده‌های فوق لازم است تا مختصری در مورد اندازه‌گیری تأخیر مدار بدانیم. به‌طورکلی سرعت مدار از طریق مقایسه سیگنال ورودی و سیگنال خروجی توسط اسیلوسکوپ قابل‌محاسبه می‌باشد. بااین‌وجود در خلال طراحی و درزمانی که مدار هنوز ساخته نشده است، محاسبه تأخیر کاری مشکل و غیر ممکن به نظر می‌رسد. در این حالت با استفاده از قابلیت Timing Simulation نرم‌افزارهای تحلیل‌گر می‌توان در خلال

طراحی نیز تأخیر مدار را محاسبه نمود. البته این نکته نیز غیر قابل انکار نیست که تأخیر واقعی مدار کاملاً" به نحوه پیاده‌سازی گیت‌ها در FPGA وابسته می‌باشد.

در آخرین بخش این قسمت لازم می‌دانیم تا علت کند بودن جمع کننده ۴ بیتی که در جلسه پیش طراحی نمودید را تشریح و راه‌کار مناسب را ارائه نماییم. همان‌طور که بیان شد ، جمع کننده ۴ بیتی که در جلسه پیش طراحی و پیاده‌سازی نمودید به جمع کننده Ripple – Carry معروف می‌باشد. چراکه، نتیجه حاصل از جمع ۲ بیت ، به رقم نقلی حاصل از مجموع ۲ بیت قبلی وابسته است. بنابراین حاصل جمع باارزش‌ترین ۲ بیت ، زمانی قابل‌دستیابی می‌باشد که رقم نقلی از کم‌ارزش‌ترین مرحله به باارزش‌ترین مرحله[1] منتقل شود. به‌سادگی مطلب فوق در مثال زیر قابل‌مشاهده می‌باشد.

$$\begin{array}{cccccc} 1 & 1 & 1 & 1 & & \\ & 1 & 1 & 1 & 1 & \\ + & 0 & 0 & 0 & 1 & \\ \hline 1 & 0 & 0 & 0 & 0 & \end{array}$$

به‌سادگی می‌توانید محاسبه نمایید که در این روش رقم نقلی خروجی در جمع کننده N بیتی پس از 2N+2 تأخیر گیت، قابل‌دستیابی می‌باشد و حاصل جمع باارزش‌ترین ۲ بیت (S_{N-1}) نیز پس از 2N+2 تأخیر گیت قابل‌دستیابی خواهد بود. با توجه به این اعداد و ارقام می‌توان ملاحظه نمود که این جمع کننده برای محاسبه حاصل جمع تعداد بیت‌های زیاد بسیار کند می‌باشد. به‌عنوان‌مثال در یک جمع کننده ۳۲ بیتی که هر گیت منطقی دارای تأخیر 1ns می‌باشد. تأخیر کلی مدار چیزی در حدود 66 ns خواهد بود که خود نشان‌دهنده این موضوع است که مدار فوق‌الذکر حداکثر با فرکانس 10Mhz کار خواهد کرد[1]

جمع کننده Carry – Look Ahead این مشکل را حل نموده است. طرح و ایده اصلی در این جمع کننده نحوه تولید رقم نقلی است. در این مدار رقم نقلی به دو روش تولید می‌شود که عبارت‌اند از : (۱) B_i , A_i هر دو یک باشند (۲) زمانی که یکی از ۲ بیت ورودی یک و رقم نقلی ورودی یک باشد. بنابراین می‌توان نوشت که :

$$Cout = C_{i+1} = A_i.B_i + (A_i \oplus B_i).C_i \qquad (۱)$$

در این عبارت "." بیانگر AND و $\oplus$بیانگر XOR می‌باشد. این رابطه را می‌توان خلاصه‌تر نیز نوشت.

$$Cout = C_{i+1} = G_i + P_i.C_i \qquad (۲)$$

$$G_i = (A_i.B_i) \qquad (۳)$$

$$P_i = (A_i \oplus B_i) \qquad (۴)$$

که G_i بیانگر تولید رقم نقلی و P_i بیانگر انتشار رقم نقلی می‌باشد. همان‌طور که اشاره شد ساختار کلی CLA برمبنای روابط فوق استوار می‌باشد. اما نکته قابل‌توجه در این جمع کننده میزان تأخیر آن است. حال به محاسبه این تأخیر می‌پردازیم.

1- Stage

فرض کنید که هر گیت AND دارای یک واحد تأخیر و هر گیت XOR دارای ۲ واحد تأخیر باشد. توجه نمایید که G_i , P_i تنها به ورودی وابسته می‌باشند و لذا به ترتیب پس از ۲ و ۱ واحد تأخیر قابل‌دستیابی هستند. لذا اگر از تعاریف فوق به‌منظور محاسبه رقم نقلی استفاده شود، دیگر نیازی نیست که برای تولید رقم نقلی منتظر انتقال رقم نقلی از مرحله‌های پایین‌تر باشیم.

حال این تعریف را به جمع کننده ۴ بیتی تعمیم می‌دهیم :

$$C_1 = G_0 + P_0.C_0 \quad (۵)$$

$$C_2 = G_1 + P_1.C_1 = G_1 + P_1.G_0 + P_1.P_0.C_0 \quad (۶)$$

$$C_3 = G_2 + P_2.C_2 = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.C_0 \quad (۷)$$

$$C_4 = G_3 + P_3.C_3 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0 + P_3.P_2.P_1.P_0.C_0 \quad (۸)$$

مشاهده می‌نمایید که بیت رقم نقلی خروجی، C_{i+1}، مرحله آخر پس از ۴ واحد تأخیر (۲ واحد تأخیر برای محاسبه سیگنال انتشار و ۲ واحد تأخیر برای گیت‌های AND،OR) قابل‌دستیابی می‌باشد. حاصل جمع سیگنال نیز از فرمول زیر قابل‌محاسبه می‌باشد :

$$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i \quad (۹)$$

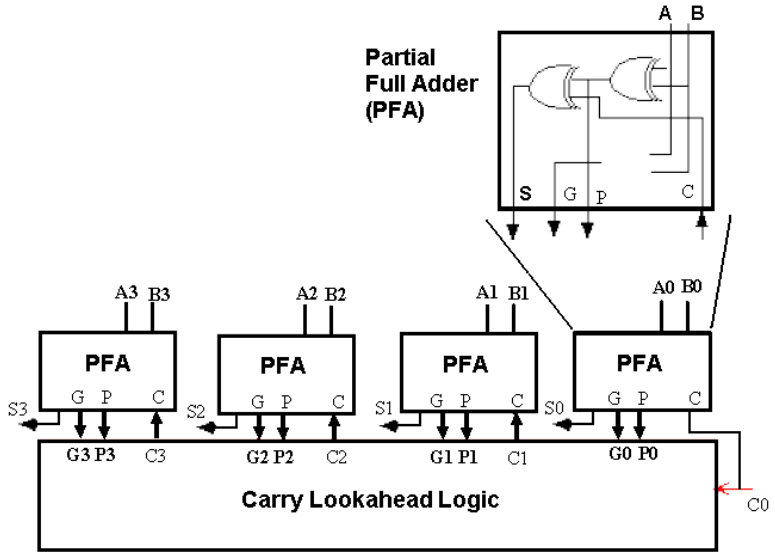
نتیجه خروجی نیز پس از ۲ واحد تأخیر دیگر یا به‌عبارت‌دیگر مجموعاً پس از ۶ واحد تأخیر پس از اعمال B_i , A_i به جمع کننده قابل‌دستیابی می‌باشد. از مزایای این روش این است که تأخیر حاصل مستقل از تعداد بیت‌های A و B می‌باشد.

با توجه به مطالبی که در مورد جمع کننده Carry Look Ahead بیان شد، می‌توان ساختار آن را به ۲ بخش کلی تقسیم نمود:

۱) تمام جمع کننده جزئی (Partial Full – Adder) که G_i , P_i , S_i را که در فرمول‌های ۳، ۴ و ۹ تعریف‌شده‌اند را تولید می‌نماید،

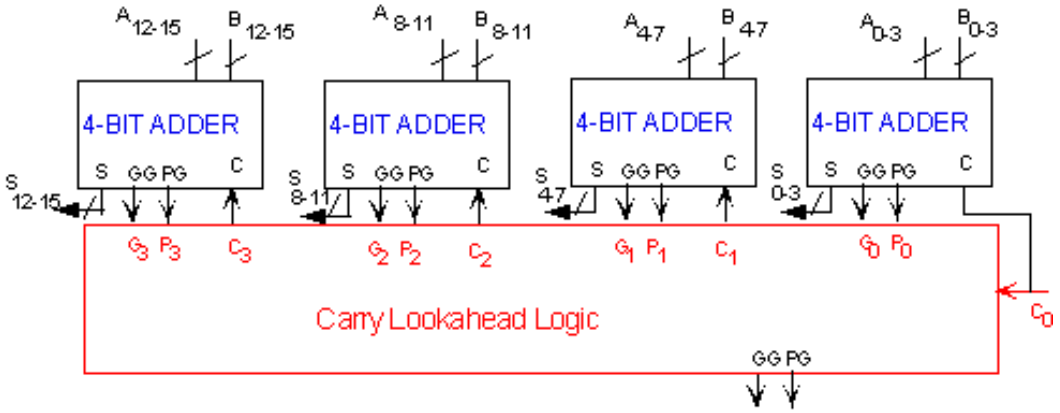
(۲) مدار منطقی Carry Look Ahead، که رقم نقلی خروجی را که مبتنی بر روابط ۵ تا ۸ می‌باشند، تولید می‌نماید.

لذا می‌توان مدار جمع کننده ۴ بیتی Carry Look Ahead را با استفاده از چهار بلوک PFA و یک بلوک منطقی Carry Look Ahead پیاده‌سازی نمود. (شکل ۵-۳)



شکل ۱۲-۱ : جمع کننده ۴ بیتی Carry Look Ahead

از معایب این جمع کننده می‌توان به پیچیدگی مدار منطقی بلوک Carry Look Ahead برای جمع کننده‌های بابیت‌های زیاد (بیش از ۴ بیت) اشاره نموده البته این مشکل نیز با استفاده از طراحی‌های سلسله مراتبی قابل‌حل می‌باشد. در این روش بلوک جمع کننده ۴ بیتی CLA به‌عنوان بلوک اصلی بوده و طراحی بر اساس آن صورت می‌گیرد. (شکل ۱۲-۲)



شکل ۱۲-۲ : جمع کننده ۱۶ بیتی Carry Look Ahead

تکالیف پیش از آزمایش :

پیش از ورود به آزمایشگاه و شروع آزمایش مطالب این بخش را مطالعه، موارد خواسته‌شده را انجام و به سؤالات مطرح‌شده پاسخ دهید و نتایج به‌دست‌آمده را در گزارش خود ثبت نمایید.

۱) همان‌طور که در تئوری آزمایش نیز بیان شد، طولانی‌ترین تأخیر در یک جمع کننده n بیتی Rippled – Carry برابر با تأخیر 2n +2 گیت می‌باشد، این مطلب را تحقیق و اثبات نمایید.

۲) نکات مربوط به Timing Simulation را مطالعه و در صورت نیاز به مستندات نرم‌افزار مراجعه نمایید.

۳) مدار PFA و Carry Look Ahead را طراحی و به‌عنوان یک بلوک تعریف نمایید.

۴) مدار یک جمع کننده ۴ بیتی CLA را با استفاده از بلوک‌های از پیش طراحی‌شده PFA و Carry Look Ahead Logic طراحی نمایید.

۵) مدارهای طراحی‌شده را توسط نرم‌افزار موردنظر ترسیم و تحلیل‌های موردنیاز را انجام و نتایج را ثبت نمایید. این تحلیل‌ها عبارت‌اند از Functional Simualtion که صحت عملکرد مدار را تضمین می‌نماید و Timing Simulation که میزان تأخیر مدار را مشخص می‌نماید.

تکالیف داخل آزمایشگاه :

نکات : دانشجویان عزیز برای مشاهده عملکرد مدارهای طراحی‌شده باید مقدمات زیر را فراهم نمایند.

- ✓ بر روی برد چهار عدد نمایشگر ۷- قسمتی وجود دارند. از این LED ها و نمایشگرها به‌منظور نمایش خروجی جمع کننده استفاده نمایید.

- ✓ از سوئیچ‌های موجود بر روی برد به‌عنوان ورودی‌ها و همچنین رقم نقلی ورودی استفاده نمایید.

حال به پیاده‌سازی مدارات طراحی‌شده می‌پردازیم :

۱) شماتیک ترسیم‌شده مدار جمع کننده ۴بیتی CLA را توسط نرم‌افزار موردنظر اجرا و مقدمات پیاده‌سازی بر روی برد را مهیا نمایید.

۳-۲) مراحل پیاده‌سازی طرح موردنظر را اجرا و نتایج حاصل را بر روی نمایشگر ۷-قسمتی مشاهده نمایید.

۳-۳) نتایج حاصل از اجرای برنامه بر روی برد را با آنچه قبلاً" به دست آورده‌اید، مقایسه نمایید.

تاخیر انتشار در مدار جمع کننده ریپل کری(آزمایش قبل) را با پیش بینی کننده کری(این آزمایش) مقایسه نمایید.

پروژه پیشنهادی :

با استفاده از مدار جمع کننده ۴ بیتی CLA ای که طراحی نمودید (به‌عنوان یک بلوک طراحی) یک جمع کننده ۱۶ بیتی CLA طراحی نمایید. ورودی‌ها، خروجی‌ها، رقم نقلی خروجی و ارتباط داخلی مابین تمام جمع کننده‌ها را مشخص نمایید. مدار جمع کننده ۴ بیتی CLA طراحی‌شده را به‌عنوان یک Symbole تعریف و با استفاده از این قابلیت شماتیک جمع کننده ۱۶ بیتی CLA را توسط نرم‌افزار ترسیم و خروجی حاصل از شبیه‌سازی را با تئوری مقایسه نمایید. درنهایت شماتیک طراحی‌شده را بر روی برد پیاده‌سازی نمایید و نتایج حاصل را بر روی نمایشگر ۷-قسمتی مشاهده کنید.

جلسه سیزدهم

آزمایش دیاگرام حالت

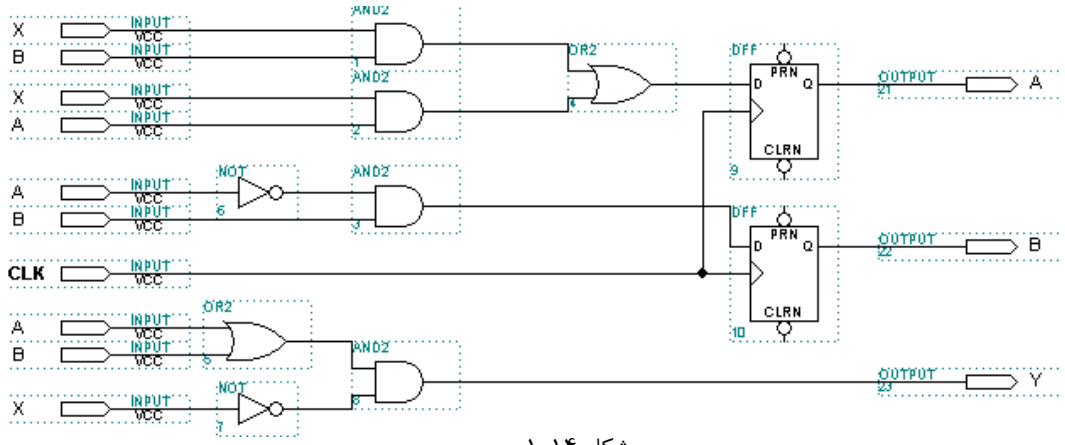
هدف :

در این آزمایش هدف زیر دنبال می‌شود:

- ✓ کسب مهارت بیشتر درزمینه ی طراحی مدارات ترتیبی

تئوری آزمایش :

ترتیب زمانی ورودی و خروجی‌ها و حالات فیلیپ فلاپها را می‌توان در جدول حالات برشمرد. مدار شکل ۱۴-۱ که در کتاب مانو آمده است دارای جدول حالتی به‌صورت جدول ۱۴-۱ می‌باشد.

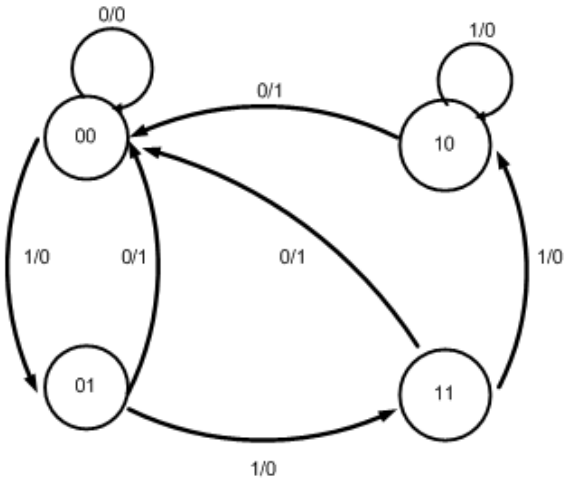


شکل ۱۴-۱

حالت فعلی	ورودی	حالت بعدی	خروجی
A B	X	A B	Y
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	0 0	1
0 1	1	1 1	0
1 0	0	0 0	1
1 0	1	1 0	0
1 1	0	0 0	1
1 1	1	1 0	0

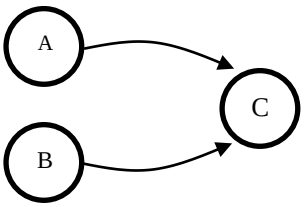
جدول ۱۴-۱: جدول حالات مدار شکل ۱۴-۱

همچنان که از آزمایش شمارنده به یاد دارید، با داشتن جدول حالات یک مدار یعنی داشتن کلیه حالات آن می‌توان به‌آسانی مدار ترتیبی مربوطه را طراحی نمود. شکل ۱۴-۱ درواقع با استفاده از جدول ۱۴-۱ طراحی‌شده است. اطلاعات موجود در یک جدول حالت را می‌توان به‌صورت گرافیکی تحت عنوان دیاگرام حالت نمایش داد. شکل ۱۴-۲ دیاگرام حالت جدول ۱۴-۱ می‌باشد.



شکل ۱۴-۲: دیاگرام حالت جدول ۱۴-۱

با توجه به مقدماتی که بیان شد و مطالبی که از آزمایش شمارنده نیز به یاد دارید، دریافتیم که به‌آسانی می‌توان با داشتن جدول حالت یا دیاگرام حالت، مدار ترتیبی موردنظر را طراحی نمود. همان‌طور که در آزمایش شمارنده نیز بیان شد طراحی مدارات ترتیبی دارای مراحل مشخصی می‌باشد، اما در طراحی مدارات ترتیبی که دارای نظم و روال منظمی نبوده و از مجموعه حالات منظمی پیروی نمی‌کند، علاوه بر نکات ذکرشده در آزمایش شمارنده بیان چند نکته دیگر نیز مهم به نظر می‌رسد. در ادامه علاوه بر مرور نکات ذکرشده به نکات ضروری جدید اشاره خواهیم نمود:



۱- بیان منطقی مسئله با توجه به بیان لفظی آن.

۲- با توجه به‌صورت‌مسئله باید شما دیاگرام حالات را رسم نمایید.

۳- دستیابی به جدول حالت سیستم با استفاده از دیاگرام حالت .

۴- ساده‌سازی جدول حالت در صورت امکان

❋ حالت معادل حالتی است که از آن دو حالت سیستم به حالت مشخصی رود که خروجی مشخصی دارد.

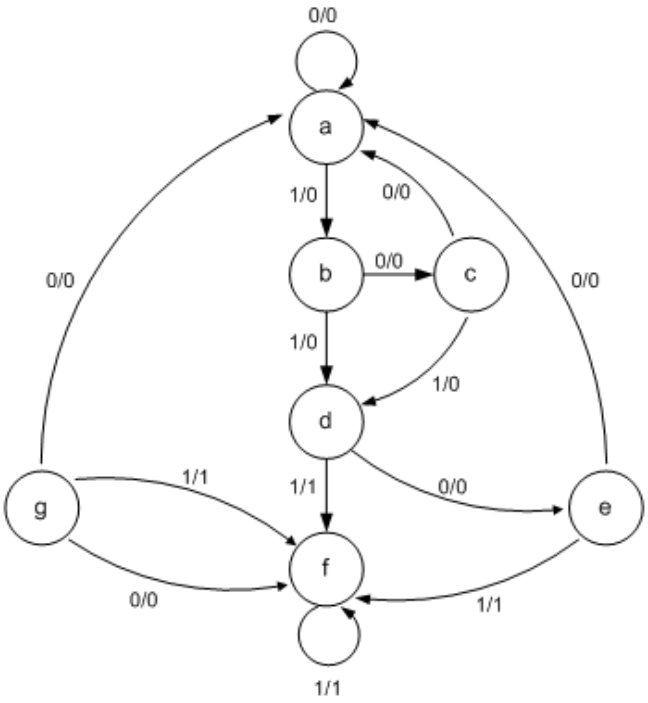
۵- با توجه به حالت‌های باقیمانده تعداد فیلیپ فلاپها مشخص می‌شود.

۶- تخصیص حالات

۷- به دست آوردن توابع ورودی فیلیپ فلاپها به کمک جدول حالت نهائی.

۸- ساده کردن توابع حاصل و رسم مدار .

نکته مهم در طراحی مدارات ترتیبی (به‌غیراز مدارات شمارنده) ساده‌سازی جدول حالت است. اکنون مدار مربوط به دیاگرام حالتی که در شکل ۱۴-۳ آمده است را با به‌کارگیری مطالب فوق طراحی می‌نماییم.



شکل ۱۴-۳ : دیاگرام حالات

	حالت بعدی		خروجی	
	X=0	**X=1**	**X=0**	**X=1**
a	*a*	*b*	*a*	*b*
b	*c*	*d*	*c*	*d*
c	*a*	*d*	*a*	*d*
d	*e*	*f*	*e*	*f*
e	*a*	*f*	*a*	*f*
f	*g*	*f*	*g*	*f*
g	*a*	*f*	*a*	*f*

جدول ۱۴-۲ جدول حالات شکل ۱۴-۳

با توجه به جدول ۱۴-۲ حالت‌های g و e دارای خروجی‌ها و حالت‌های یکسان هستند بنابراین g حذف‌شده و بجای g، e قرار می‌دهیم.

	X=0	X=1	X=0	X=1
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*f*	0	1
e	*a*	*f*	0	1
f	*e*	*f*	0	1

جدول ۱۴-۳ : جدول حالت با حذف حالت g

در جدول ۱۴-۳ نیز d و f معادل هستند. با جایگزین کردن d به‌جای f , یک حالت دیگر نیز از سیستم کم خواهد شد.

	X=0	X=1	X=0	X=1
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*d*	0	1
e	*a*	*d*	0	1

جدول ۱۴-۴ : جدول حالت با حذف f

مرحله بعدی تخصیص حالات می‌باشد. این کار با استفاده از سه فیلیپ فلاپ C , B , A صورت خواهد گرفت.

	A	B	C
a	0	0	0
b	0	0	1
c	0	1	0
d	0	1	1
e	1	0	0

جدول ۱۴-۵ : تخصیص حالت

اکنون جدول حالت فیلیپ فلاپ ها را با استفاده از جدول ۱۴-۵ طراحی نمایید.

تکالیف پیش از آزمایش :

۱- آزمایش اول

۱-۱- مدار شکل ۶-۱ در نرم‌افزار پیاده‌سازی نمایید.

۱-۲- ورودی را به یک سوئیچ، خروجی مدار را به یک LED تک‌رنگ و حالات مدار را به دو LED تک‌رنگ متصل نمایید.

۲- آزمایش دوم :

۲-۱- جدول حالت فیلیپ فلاپها را با استفاده از جدول ۶-۵ ترسیم نمایید.

۲-۲- با فیلیپ فلاپ JK مدار ترتیبی جدول ۶-۵ را رسم نمایید.

۲-۳- ورودی مدار را به یک سوئیچ، خروجی مدار را به یک LED تک‌رنگ و حالت‌های مدار را به سه LED دورنگ که یک‌پایه آن غیرفعال‌شده است , متصل نمایید.

تکالیف داخل آزمایشگاه :

نکته: این دو آزمایش با کلاکی در حدود 500ms انجام خواهد گرفت .

۱- آزمایش اول

۱-۱- مداری را که به‌عنوان آزمایش اول توسط نرم‌افزار طراحی و آماده پیاده‌سازی نموده‌اید را بر روی FPGA پیاده نمایید.

۱-۲- با تغییر دادن ورودی نتیجه خروجی و حالات مختلف مدار را یادداشت نمایید.

۲- آزمایش دوم :

۲-۱- مداری را که به‌عنوان آزمایش دوم توسط نرم‌افزار طراحی و آماده پیاده‌سازی نموده‌اید را بر روی FPGA پیاده نمایید.

۲-۲- با تغییر دادن ورودی نتیجه خروجی و حالات مختلف مدار را یادداشت نمایید.

جلسه چهاردهم

شمارنده‌ها

هدف: در این آزمایش اهداف زیر دنبال می‌شوند :

- ✓ آشنایی با مفاهیم طراحی مدارهای ترتیبی
- ✓ طراحی و پیاده سازی شمارنده‌ها

تئوری آزمایش: شمارنده‌ها ، در طراحی سیستم های دیجیتال به‌طور عام و بویژه در پردازشگرها بکار می روند. این مدارها برای تولید مقادیر زمانی به‌منظور تنظیم و کنترل عملیات در یک دستگاه دیجیتال مفید هستند. یک شمارنده ضرورتاً یک ثبات است که به ازای پالسهای ورودی به آن، دنباله‌ای از حالتهای از پیش تعیین شده را ایجاد می‌کند. شمارنده‌ها را می توان به دو دسته سنکرون و آسنکرون (موج گونه) طبقه بندی نمود.

اما نکته بسیار مهم در مدارهای ترتیبی، نحوه طراحی آن‌ها است. در طراحی مدارهای ترتیبی برخلاف مدار ترکیبی که با ترسیم یک جدول درستی طراحی مدار کاملاً مشخص می‌شود، به جدول و دیاگرام حالت نیز نیاز است. طراحی مدار ترتیبی شامل تعیین فلیپ فلاپ و سپس یافتن ساختار مدار ترکیبی است که همراه با فلیپ فلاپ ها نیازهای تعیین شده را برآورده می سازد. روش طراحی مطرح شده شامل مراحل زیر است:

۱- توصیف عملکرد مدار توسط دیاگرام حالت یا دیاگرام زمانی.

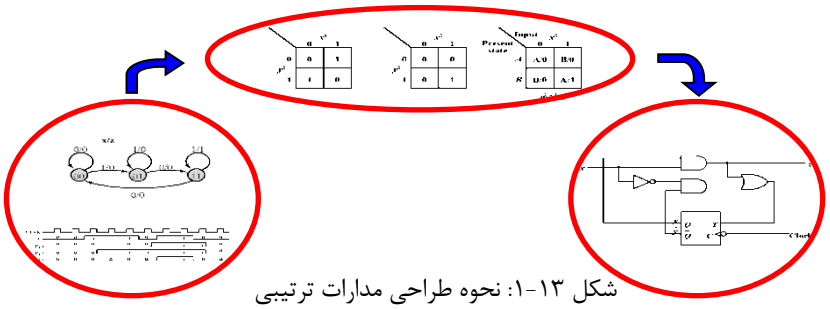
۲- تهیه جدول حالات با توجه به اطلاعات مفروض.

۳- ساده‌سازی جدول حالات، در صورت امکان.

۴- تعیین تعداد و نوع فلیپ فلاپ‌های مورد نیاز و اختصاص یک سمبل به هرکدام.

۵- به دست آوردن توابع خروجی مدار و توابع ورودی فلیپ فلاپ ها، با بکارگیری روش نقشه یا هر روش ساده سازی دیگر.

۶- ترسیم دیاگرام منطقی مدار.



شکل ۱۳-۱: نحوه طراحی مدارات ترتیبی

تکالیف پیش از آزمایش

پیش از ورود به آزمایشگاه و شروع آزمایش مطالب این بخش را مطالعه، موارد خواسته‌شده را انجام و به سؤالات مطرح شده پاسخ دهید و نتایج به دست آمده را در گزارش خود ثبت نمایید.

۱) طراحی شمارنده ۴ - بیتی سنکرون (۳- افزا)

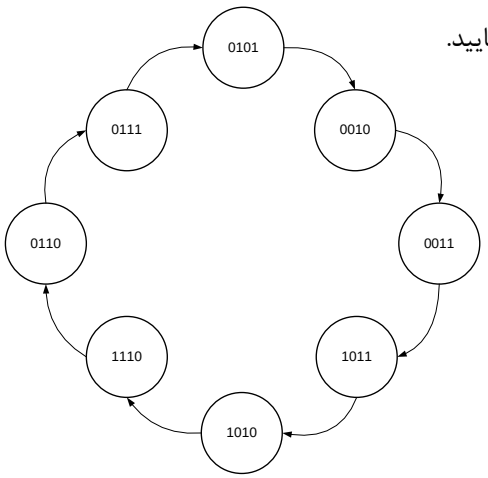
۱-۱) با استفاده از فلیپ فلاپ های نوع JK شمارنده ۴ - بیتی سنکرونی طراحی نمایید که دارای یک ورودی X باشد. این ورودی جهت شمارش رو به بالا (X=1) و رو به پایین (X=0) را تعیین می‌نماید. شمارنده موردنظر ۳- افزا است . بدین معنی که باید اعداد ۰ تا ۱۵ را به‌طوری که اختلاف دو عدد متوالی همواره ۳ باشد، بشمارد.

۱-۲) برنامه شمارنده موردنظر را نوشته و تست نمایید.

۱-۳) برنامه را به‌منظور نمایش خروجی برروی نمایشگر هفت قسمتی تکمیل نمایید.

۲) طراحی شمارنده اختصاصی (شبیه کد گری)

۲-۱) مدار شمارنده‌ای را طراحی نمایید که دیاگرام حالت آن به‌صورت زیر باشد. در طراحی خود از فلیپ فلاپ های نوع D استفاده کنید. مراحل طراحی خود را در گزارش ثبت نمایید.



شکل ۱۳-۲: دیاگرام حالت

۲-۲) برنامه مدار طراحی شده را نوشته و خروجی مدار را توسط شبیه سازی مشاهده و تست نمایید.

۲-۳) برنامه را به‌منظور نمایش خروجی برروی نمایشگر هفت قسمتی تکمیل نمایید.

راهنمایی: مدار موردنظر یک شمارنده دو دویی می‌باشد که تنها اعداد ۵ , ۷ , ۶ , ۱۴ , ۱۰ , ۱۱ , ۳ , ۲ را می شمارد .این مدار نوعی کد کننده گری[1] است.

نکات : دانشجویان عزیز برای مشاهده عملکرد مدارهای طراحی شده باید مقدمات زیر را فراهم نمایند:

[1] Gray Coder

۱) برای قابل رویت شدن نتایج باید پریود پالس ساعت را در حدود ۵۰۰ ms قـرارداد. بـدین منظـور مـی‌توانیـد از مـدارات مقسم فرکانس و پایه پنج F0o*=ΛKHz U0 استفاده نمایید.

۲) برروی برد چهار LED و ۴ عدد نمایشگر هفت قسمتی وجود دارنـد. از ایـن LED هـا و نمایشـگرها بـه‌منظـور نمـایش خروجی شمارنده‌ها استفاده نمایید.

۳) از سوئیچ‌های موجود بر روی برد به‌منظور فرمان دادن یا بار نمودن شمارنده‌ها استفاده نمایید.

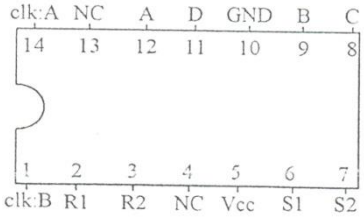
تکالیف داخل آزمایشگاه:

۱- شمارنده ۳- افزا طراحی شده را برروی برد پیاده سازی نموده و با اسـتفاده از سـوئیچ هـای موجـود بـرروی بـرد نحـوه شمارش (بالا شمار - پایین شمار) را مشخص و نتایج حاصل را برروی نمایشگرهای هفت قسمتی مشاهده نمایید.

۲- شمارنده اختصاصی کد گری طراحی شده را برروی برد پیاده سازی نموده و نتایج حاصل را برروی نمایشـگرهای هفـت قسمتی مشاهده نمایید.

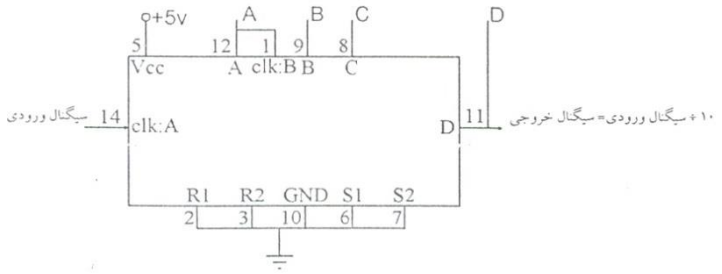
به کمک آزمایش شش یک شمارنده BCD طراحی و خروجی‌های شمارنده را به ورودی 7seg (نمایشگر هفت قسمتی) متصل نمایید. (در واقع به گونه‌ای بجای تراشه ۷۴۴۸ یک مدار معادل را جایگزین می نمایید)

راهنمایی:۱- آی سی ۷۴۹۰ نوعی شمارنده /مقسم فرکانس قابل برنامه ریزی می‌باشد که از دو شمارنده مستقل تشکیل شده است . نسبت تقسیم یکی از این شمارنده‌ها ۲ و دیگری ۵ می‌باشد ، که با لبه رو به پایین پالس ساعت تحریک می‌شوند . می‌توان این دو شمارنده را با یکدیگر مورداستفاده قرارداده ، به این ترتیب نوعی شمارنده مبنای ۱۰ با کد دهی BCD ، یا بدون آن ، ایجاد می‌شوند . همچنین می‌توان آن را طوری پیکربندی کرد که نسبت تقسیم شمارنده عددی بین ۹ – ۲ باشد . شکل زیر نقشه عملی نحوه قرار گرفتن پایه‌های آی سی را نشان می‌دهد.



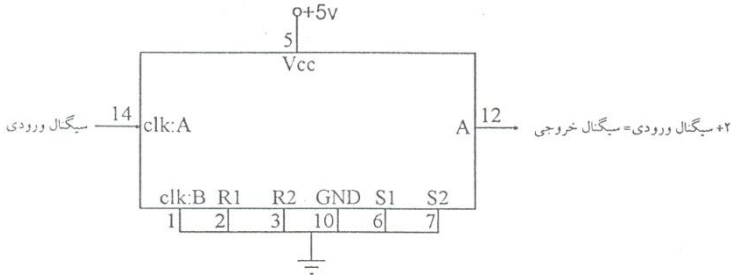
شکل۱۳-۳: نحوه قرار گرفتن پایه‌های آی سی ۷۴۹۰

راهنمایی: ۲- شکل زیر روش استفاده از آی سی ۷۴۹۰ در مبنای ۱۰ و با خروجی BCD را نشان می‌دهد . در اینجا هر دو شمارنده داخلی آی سی مورداستفاده قرارگرفته اند. توجه داشته باشید که خروجی نا متقارن می‌باشد.



شکل۱۳-۴: شمارنده مبنای ۱۰

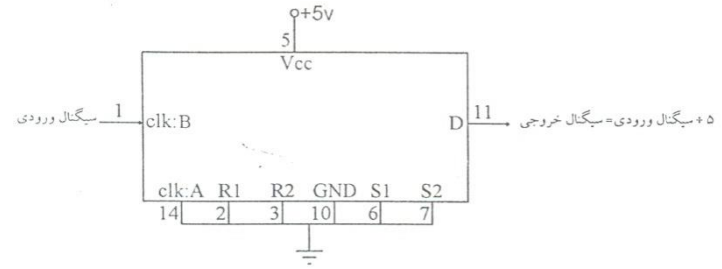
۳- در شکل زیر فقط شمارنده نصف کننده فرکانس داخلی مورداستفاده قرارگرفته است . در این حالت آی سی به‌صورت شمارنده مبنای ۲ عمل می‌کند.



شکل۱۳-۵: شمارنده مبنای ۲

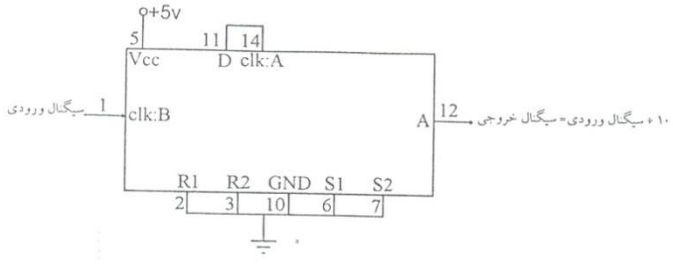
٤- در شکل زیر فقط شمارنده / مقسم فرکانسی مورداستفاده قرارگرفته که نسبت تقسیم آن ۵ می‌باشد.

در اینجا بخش نصف کننده فرکانس و گیت‌های کنترل کننده Reset ، Set غیر فعال شده‌اند بنابراین آی سی مزبور فقط به‌صورت شمارنده / مقسم فرکانس یا نسبت تقسیم ۵ عمل می‌کند.



شکل۱۳-۶: مقسم فرکانس بر ۵

۵- شکل زیر روش استفاده از آی سی مزبور به‌صورت شمارنده مبنای ۱۰ و خروجی متقارن را نشان می‌دهد.



شکل۱۳-۷: شمارنده مبنای ۱۰ و خروجی متقارن

'46A, '47A, 'LS47 feature

- Open-Collector Outputs Drive Indicators Directly
- Lamp-Test Provision
- Leading/Trailing Zero Suppression

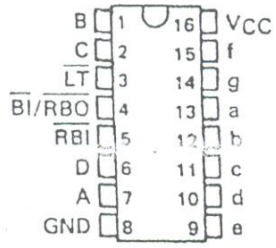
'48, 'LS48 feature

- Internal Pull-Ups Eliminate Need for External Resistors
- Lamp-Test Provision
- Leading/Trailing Zero Suppression

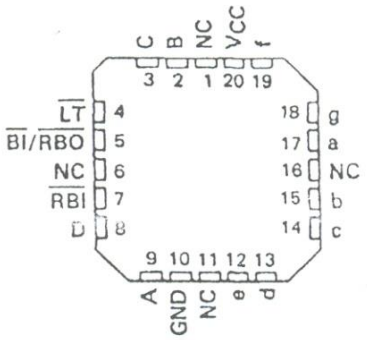
'LS49 feature

- Open-Collector Outputs
- Blanking Input

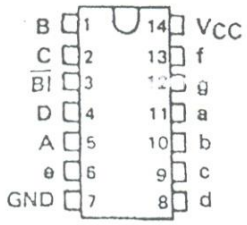
SN5446A, SN5447A, SN54LS47, SN5448, SN54LS48 . . . J PACKAGE
SN7446A, SN7447A, SN7448 . . . N PACKAGE
SN74LS47, SN74LS48 . . . D OR N PACKAGE
(TOP VIEW)



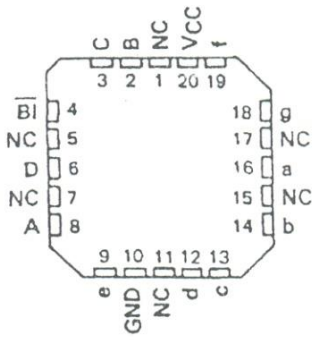
SN54LS47, SN54LS48 . . . FK PACKAGE
(TOP VIEW)



SN54LS49 . . . J OR W PACKAGE
SN74LS49 . . . D OR N PACKAGE
(TOP VIEW)



SN54LS49 . . . FK PACKAGE
(TOP VIEW)



NC — No internal connection

شکل ۱۳-۵

- **Package Options Include Plastic "Small Outline" Packages, Ceramic Chip Carriers, and Plastic and Ceramic DIPs**
- **Dependable Texas Instruments Quality and Reliability**

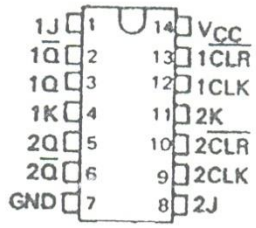
description

The '107 contain two independent J-K flip-flops with individual J-K, clock, and direct clear inputs. The '107 is a positive pulse-triggered flip-flop. The J-K input data is loaded into the master while the clock is high and transferred to the slave and the outputs on the high-to-low clock transistion. For these devices the J and K inputs must be stable while the clock is high.

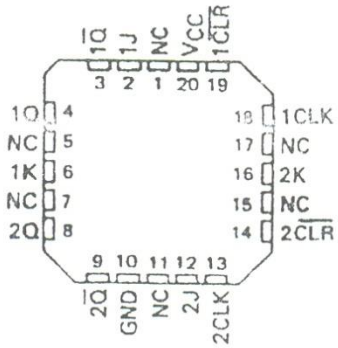
The 'LS107A contain two independent negative-edge-triggered flip-flops. The J and K inputs must be stable prior to the high-to-low clock transition for predictable operation. When the clear is low, it overrides the clock and data inputs forcing the Q output low and the $\overline{Q}$ output high.

The SN54107 and the SN54LS107A are characterized for operation over the full military temperature range of −55°C to 125°C. The SN74107 and the SN74LS107A are characterized for operation from 0°C to 70°C.

SN54107, SN54LS107A . . . J PACKAGE
SN74107 . . . N PACKAGE
SN74LS107A . . . D OR N PACKAGE
(TOP VIEW)



SN54LS107A . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection

'107
FUNCTION TABLE

INPUTS				OUTPUTS	
$\overline{CLR}$	CLK	J	K	Q	$\overline{Q}$
L	X	X	X	L	H
H	⊓	L	L	Q_0	$\overline{Q}_0$
H	⊓	H	L	H	L
H	⊓	L	H	L	H
H	⊓	H	H	TOGGLE	

'LS107A
FUNCTION TABLE

INPUTS				OUTPUTS	
$\overline{CLR}$	CLK	J	K	Q	$\overline{Q}$
L	X	X	X	L	H
H	↓	L	L	Q_0	$\overline{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	TOGGLE	
H	H	X	X	Q_0	$\overline{Q}_0$

شکل ۱۳-۶

جلسه پانزدهم

آزمایش طراحی مقسم فرکانس

۱- مقسم فرکانس

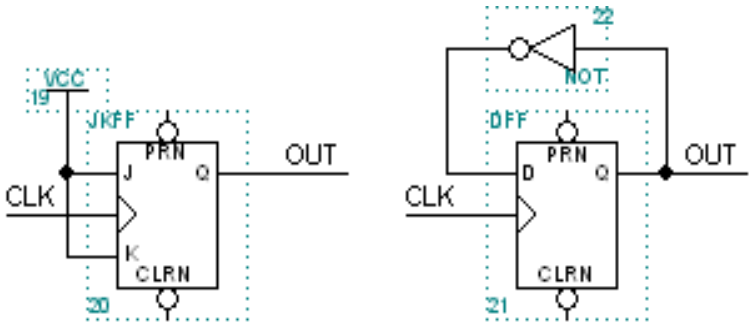
هدف :

در این آزمایش اهداف زیر دنبال می‌شوند :

- ✓ طراحی و شبیه‌سازی مقسم فرکانس‌های زوج و فرد.
- ✓ پیاده‌سازی آن‌ها بر روی FPGA.
- ✓ آزمودن مدارهای پیاده‌سازی شده.

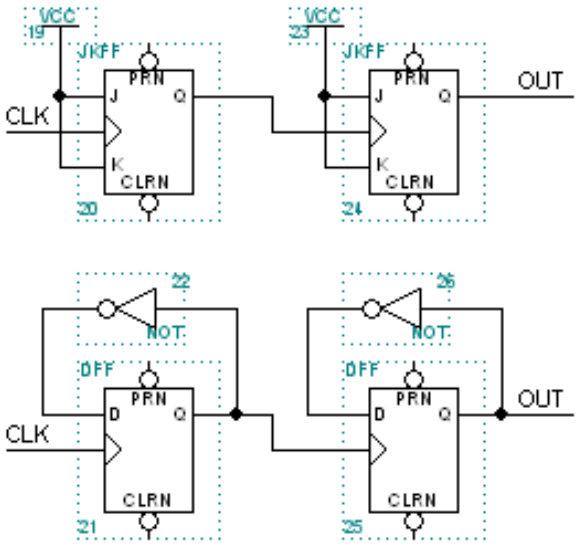
تئوری آزمایش :

مدارهای دیجیتالی که تاکنون در آزمایش‌های مختلف موردبررسی قرار گرفتند، مدارهای ترکیبی بودند. اما بیشتر سیستم‌های عملی دارای عناصر حافظه‌ای هستند که مدارهای ترتیبی را به وجود می‌آورند. عناصر حافظه‌ای که به‌عنوان قطعات اصلی در مدارهای ترتیبی با پالس ساعت به کار می‌روند، فیلیپ فلاپ نامیده می‌شوند. مدارهای ترتیبی مبتنی بر فیلیپ فلاپ‌ها، کاربرد و عملکردهای مختلفی دارند. ازجمله این کاربردها می‌توان به مقسم فرکانس اشاره داشت که نوع ساده‌ای از شمارنده‌ها می‌باشند (در آزمایش‌های بعد با شمارنده‌ها بیشتر آشنا خواهیم شد). مدارهای مقسم فرکانس به‌منظور تولید پالس‌هایی با سرعتی کمتر از سرعت پالس مرجع به کار می‌روند. هدف نهایی در این آزمایش طراحی و پیاده‌سازی انواع مدارهای مقسم فرکانسی است که از فیلیپ فلاپهای نوع D ,JK و T استفاده می‌نمایند. ساده‌ترین مقسم فرکانس عبارت است از مقسم دو که به‌سادگی با یک فیلیپ فلاپ نوع D یا JK قابل پیاده‌سازی می‌باشد.



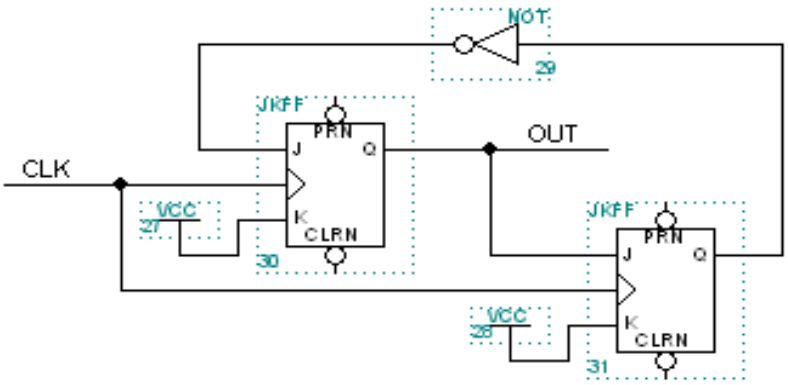
شکل ۱۵-۱: مقسم ۲ مبتنی بر فیلیپ فلاپ‌های نوع D و JK

به‌سادگی می‌توان نتیجه گرفت که برای طراحی مقسم فرکانس‌های زوج ۲n، n مقسم فرکانس ۲ را باهم به‌صورت زنجیره‌ای متصل نموده و خروجی طبقه n ام حاصل تقسیم پالس ورودی بر ۲n را نتیجه می‌دهد. به‌عنوان‌مثال شکل زیر مقسم فرکانس ۴ را با استفاده از دو نوع فیلیپ - فلاپ نمایش می‌دهد.



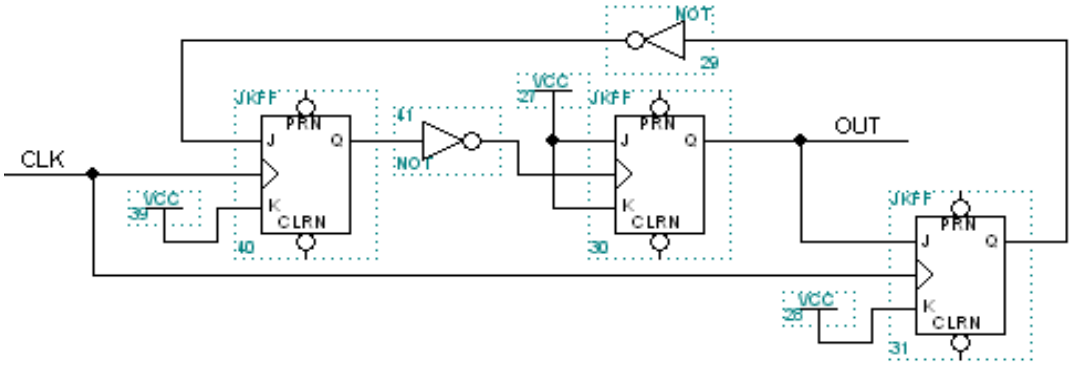
شکل ۱۵-۲: مقسم فرکانس ۴ مبتنی بر فیلیپ فلاپ‌های نوع D و JK

اما تقسیم فرکانس بر اعداد فرد کمی پیچیده‌تر می‌باشد. به‌منظور طراحی این مقسم فرکانس ابتدا عدد موردنظر را تجزیه کرده و تعداد ۲هایی که در آن وجود دارد را به دست می‌آوریم. به عبارتی عدد موردنظر را به فرم 2×X+1 تبدیل می‌نماییم. به‌عنوان‌مثال عدد ۳ عبارت است از ۱×۲+۱ یا عدد ۵ برابر است با ۲×۲+۱. سپس مقسم فرکانس موردنظر را با استفاده از مدار مقسم فرکانس ۲ و مدارهای جانبی طراحی می‌نماییم. به‌طور مثال مقسم ۳ به‌صورت زیر قابل طراحی است:



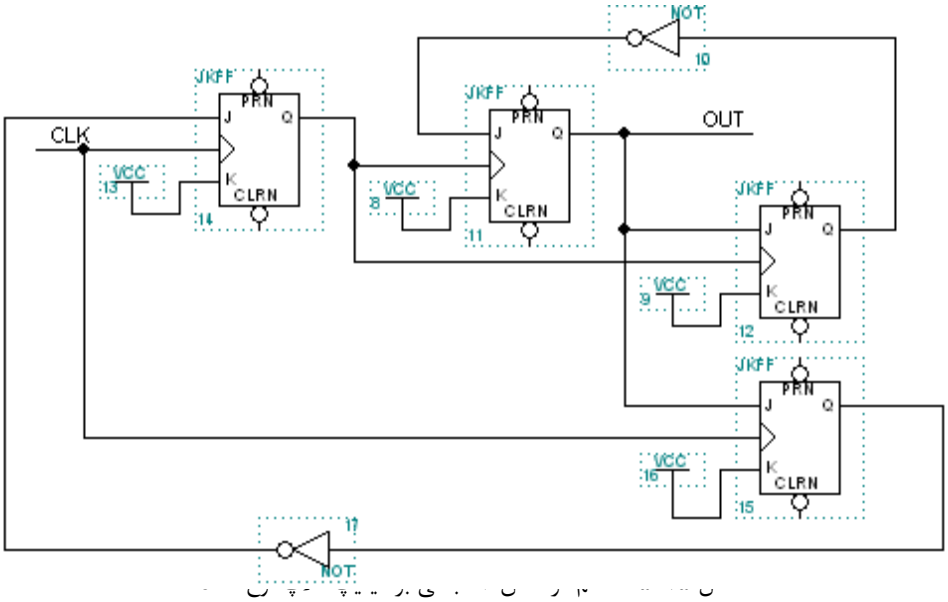
شکل ۱۵-۳: مقسم فرکانس ۳ مبتنی بر فیلیپ فلاپ نوع JK

مثال دیگر عدد ۵ می‌باشد که با استفاده از دو مدار مقسم فرکانس ۲ و مدار جانبی طراحی مقسم فرکانس صورت می‌پذیرد:



شکل ۱۵-۴: مقسم فرکانس ۵ مبتنی بر فیلیپ فلاپ نوع JK

مقسم فرکانس ۷ نیز به همین ترتیب طراحی می‌گردد و عبارت است از :



بدین ترتیب می‌توان با ترکیب این مقسم فرکانس‌ها به مدارهای جدید دست پیدا کرد.

در انتها کمی در مورد قطعاتی که دانشجویان عزیز در خلال این آزمایش می‌بایست از آن‌ها استفاده نمایند را موردبررسی قرار می‌دهیم. قطعات ۷۴۷۳، ۷۴۷۶ و ۷۴۷۸ فیلیپ-فلاپ‌های نوع JK و قطعه ۷۴۷۴ فیلیپ-فلاپ‌های نوع D هستند. با استفاده از قطعات JKFF، DFF و TFF و یا با استفاده از ICهای متداول با شماره‌های ۷۴۷۳، ۷۴۷۶، ۷۴۷۸ و ۷۴۷۴ مدارهای موردنظر را طراحی و پیاده‌سازی نمایید.

تکالیف پیش از آزمایش :

پیش از ورود به آزمایشگاه و شروع آزمایش مطالب این بخش را مطالعه، موارد خواسته‌شده را انجام و به سؤالات مطرح‌شده پاسخ دهید و نتایج به‌دست‌آمده را در گزارش خود ثبت نمایید.

۱) اعداد ۲, ۳, ۴ و ۵ اعدادی هستند که تجزیه‌وتحلیل مقسم‌های آن‌ها از اهمیت خاصی برخوردار است و به‌سادگی از ترکیب آن‌ها می‌توان به اعداد و مقسم فرکانس‌های دیگر دست پیدا کرد. علت اهمیت این امر در آشنایی شما با نحوه طراحی و پیاده‌سازی مقسم فرکانس‌های $2^n + 1$ می‌باشد. مدارهای طراحی‌شده را توسط نرم‌افزار موردنظر ترسیم و خروجی‌های آن را پس از تحلیل در گزارش خود ثبت نمایید. لازم به ذکر است که مدار را به‌گونه‌ای طراحی نمایید که نتایج خروجی توسط LEDها قابل‌مشاهده باشد.

۲) سه مقسم فرکانس ۱۹, ۵۲ و ۱۲۰ را طراحی و توسط نرم‌افزار موردنظر شماتیک آن را ترسیم و شبیه‌سازی‌های لازم را صورت دهید و نتایج تحلیل را در گزارش خود ثبت نماید. لازم به ذکر است که مدار را به‌گونه‌ای طراحی نمایید که نتایج خروجی توسط نمایشگرهای ۷ قسمتی قابل‌مشاهده باشد.

۳) مقسم فرکانس‌های طراحی‌شده را برای پیاده‌سازی آماده نمایید.

تکالیف داخل آزمایشگاه :

نکات : دانشجویان عزیز برای مشاهده عملکرد مدارهای طراحی‌شده باید مقدمات زیر را فراهم نمایند.

۱) از مولد پالس به‌عنوان ورودی مدارها و سیگنال مرجع استفاده نمایید. برای قابل‌رؤیت شدن خروجی باید از فرکانس Clock را در حدود ۸KHz قرار دهید.

۲) بر روی برد، چهار LED دورنگ وجود دارد از این LED ها به‌منظور نمایش پالس ورودی و پالس خروجی استفاده نمایید.

حال به پیاده‌سازی مدارهای طراحی‌شده بر روی برد می‌پردازیم :

۱) مقسم فرکانس‌های اعداد پایه نظیر ۲, ۳, ۴ و ۵:

۱-۱) مدارهای طراحی‌شده را بر روی برد پیاده‌سازی نمایید.

۱-۲) آزمایش را انجام داده و نتایج را توسط LEDها مشاهده نمایید.

۱-۳) به‌منظور درک بهتر عملکرد مقسم‌ها با استفاده از اسکوپ پالس خروجی مدارها را با ورودی مقایسه نمایید.

۲) مقسم فرکانس‌های ۱۳, ۱۹ و ۱۲۰

۲-۱) مقسم فرکانس‌های ۱۹, ۵۲ و ۱۲۰ را که طراحی نموده‌اید , بر روی برد پیاده نمایید.

۲-۲) آزمایش را انجام داده و نتایج را توسط نمایشگرهای ۷ قسمتی مشاهده نمایید.

پروژه پیشنهادی :

۱) طراحی و پیاده‌سازی مقسم فرکانس‌های فرد یا زوجی که به‌صورت 2^n نباشند.

پیشنهاد و طراحی مقسم فرکانس با استفاده از فیلیپ فلاپ نوع T برای اعداد مضرب ۲ و به‌صورت 2^n، مضرب ۲ غیر از 2^n و اعداد فرد.

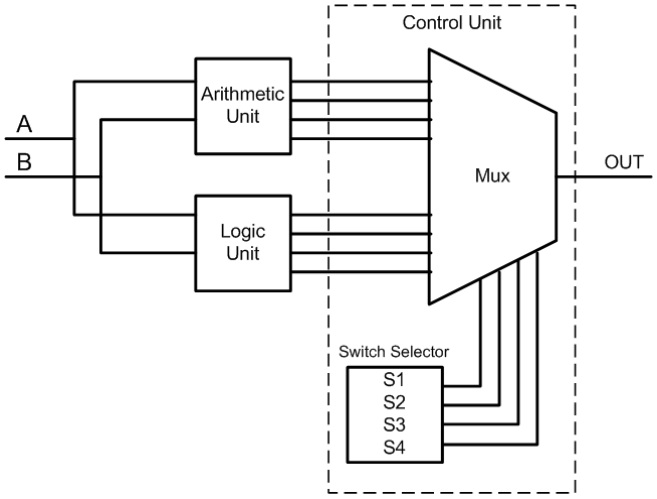
جلسه شانزدهم

آزمایش طراحی واحد محاسبه و منطق[1]

هدف : آشنایی با ALU و ساختار آن

طراحی و پیاده‌سازی مدارهای محاسباتی ، منطقی و ثباتی یک CPU

تئوری آزمایش: بخش محاسبات و منطق یکی از مهم‌ترین قسمت‌های یک CPU می‌باشد. به‌طورکلی ALU از سه قسمت محاسبات، منطق و کنترل تشکیل‌شده است. هر بخش با توجه به وظایفی که دارد یک سری عملکردهایی را انجام می‌دهد. قسمت محاسباتی اعمالی نظیر جمع، تفریق، ضرب و تقسیم را بر عهده دارد. در قسمت منطقی عملیاتی نظیر NOT ,XOR ,OR AND, صورت می‌پذیرد و درنهایت قسمت کنترل نیز وظیفه تعیین واحد عملیاتی و عملیات موردنظر را به عهده دارد. در این آزمایش هدف پیاده‌سازی یک ALU بسیار ساده می‌باشد. شکل ۸-۱ بلوک دیاگرام سیستم موردنظر را نمایش می‌دهد.



شکل ۱۶-۱- بلوک دیاگرام ALU

[1] ALU

جدول زیر کدهای عملیاتی را که توسط بخش محاسبات و منطق انجام می‌شود، نمایش می‌دهد. دانشجویان با استفاده از سوئیچ‌های موجود بر روی برد، عملکرد موردنظر را مشخص و داده ورودی را به ALU اعمال می‌نمایند. لازم به ذکر است که تمام عملیات موردنظر حداکثر ۴ بیتی می‌باشند.

پیش از آغاز آزمایش لازم است تاکمی بیشتر در مورد بلوک‌های این سیستم به بحث بپردازیم. در بخش محاسباتی دانشجویان عزیز می‌بایست از بلوک جمع کننده‌ای که در آزمایش مای اخیر طراحی شد به‌عنوان جمع کننده ۴ - بیتی استفاده نمایند. با استفاده از همان مدار نیز می‌توانند افزایش به‌اندازه یک واحد را نیز پیاده‌سازی نمایند. اما مدار تفریق کننده و کاهش به‌اندازه یک واحد را که در آزمایش‌های گذشته به آن‌ها نپرداخته‌اند می‌بایست از ابتدا طراحی نمایند. البته طراحی تفریق کننده ۴ - بیتی نیز بر پایه تمام جمع کننده امکان‌پذیر است و ازاین‌رو با جمع کننده ۴ - بیتی اختلاف بسیار کوچکی دارد.

به‌منظور سادگی سیستم، بخش کنترلی تنها به‌عنوان مالتی‌پلکسر عمل می‌نماید. ورودی این بلوک حاصل کلیه اعمال موردنظر (محاسباتی: جمع، تفریق، ... و منطقی: AND، OR، ...) است و با توجه به سیگنال کنترلی، ورودی موردنظر به خروجی منتقل می‌شود.

	S3	S2	S1	S0
ADD	0	0	0	0
SUB	0	0	0	1
INC1	0	0	1	0
DEC1	0	0	1	1
AND	1	0	0	0
OR	1	0	0	1
XOR	1	0	1	0
NOT	1	0	1	1

جدول ۱۶-۱

تکالیف پیش از آزمایش

تاکنون کلیه مدارهای موردنیاز در این بخش در آزمایش‌های قبل پیاده‌سازی شده‌اند، به‌جز تفریق کننده ۴ - بیتی که می‌بایست برنامه آن را نوشته و پیاده‌سازی نمایید.

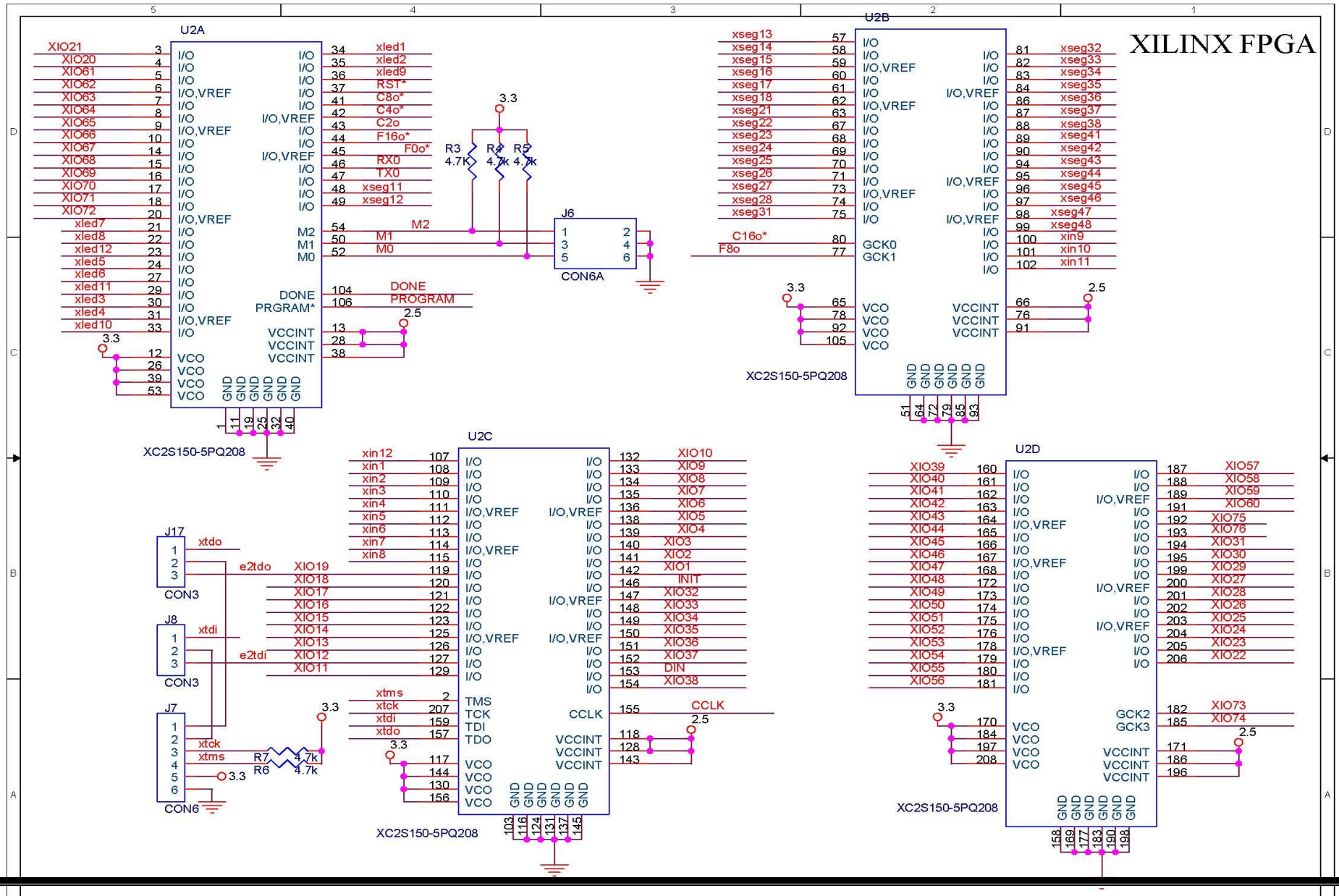
تکالیف داخل آزمایشگاه

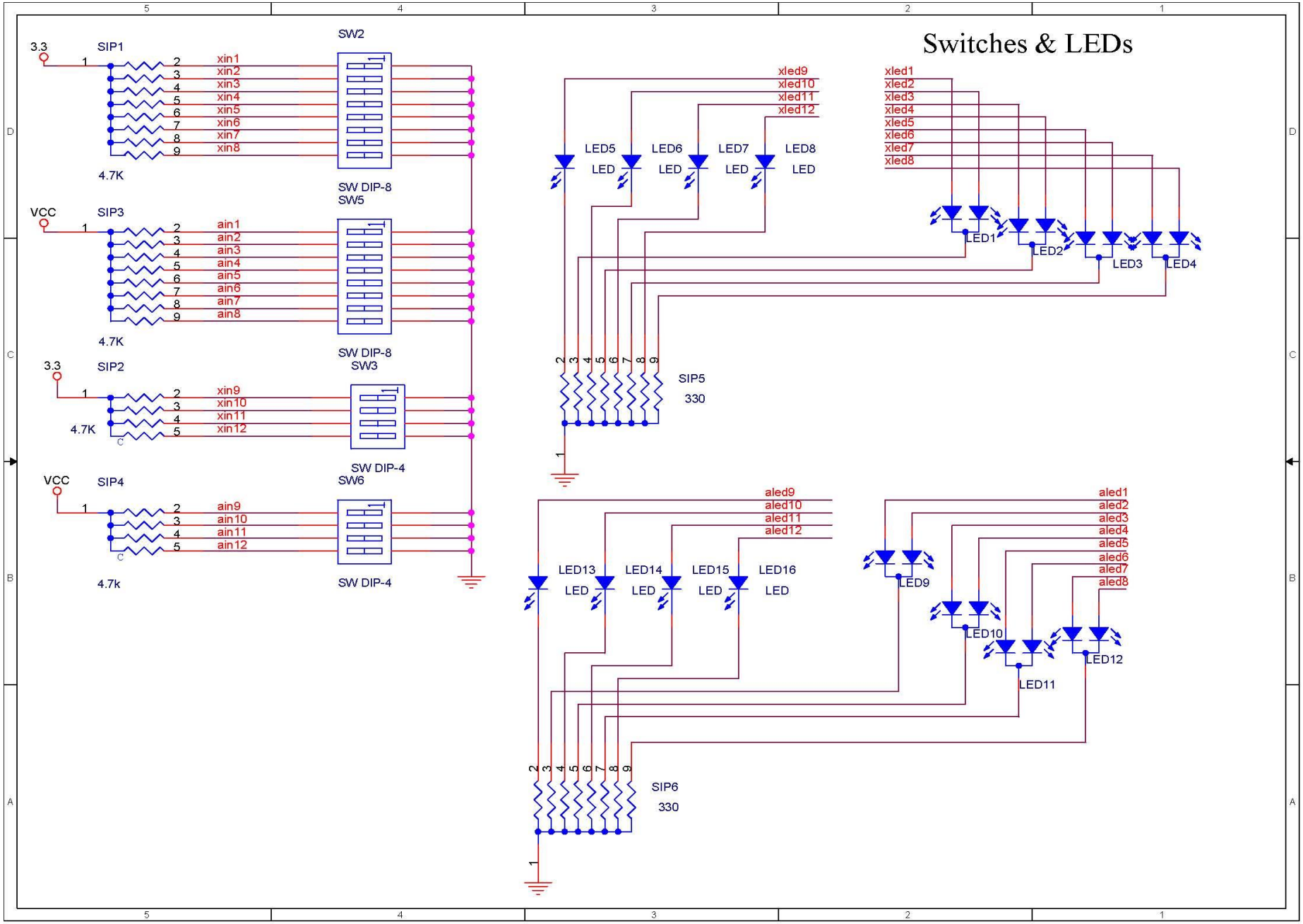
نکات : دانشجویان عزیز برای مشاهده عملکرد مدارهای طراحی‌شده باید مقدمات زیر را فراهم نمایند.

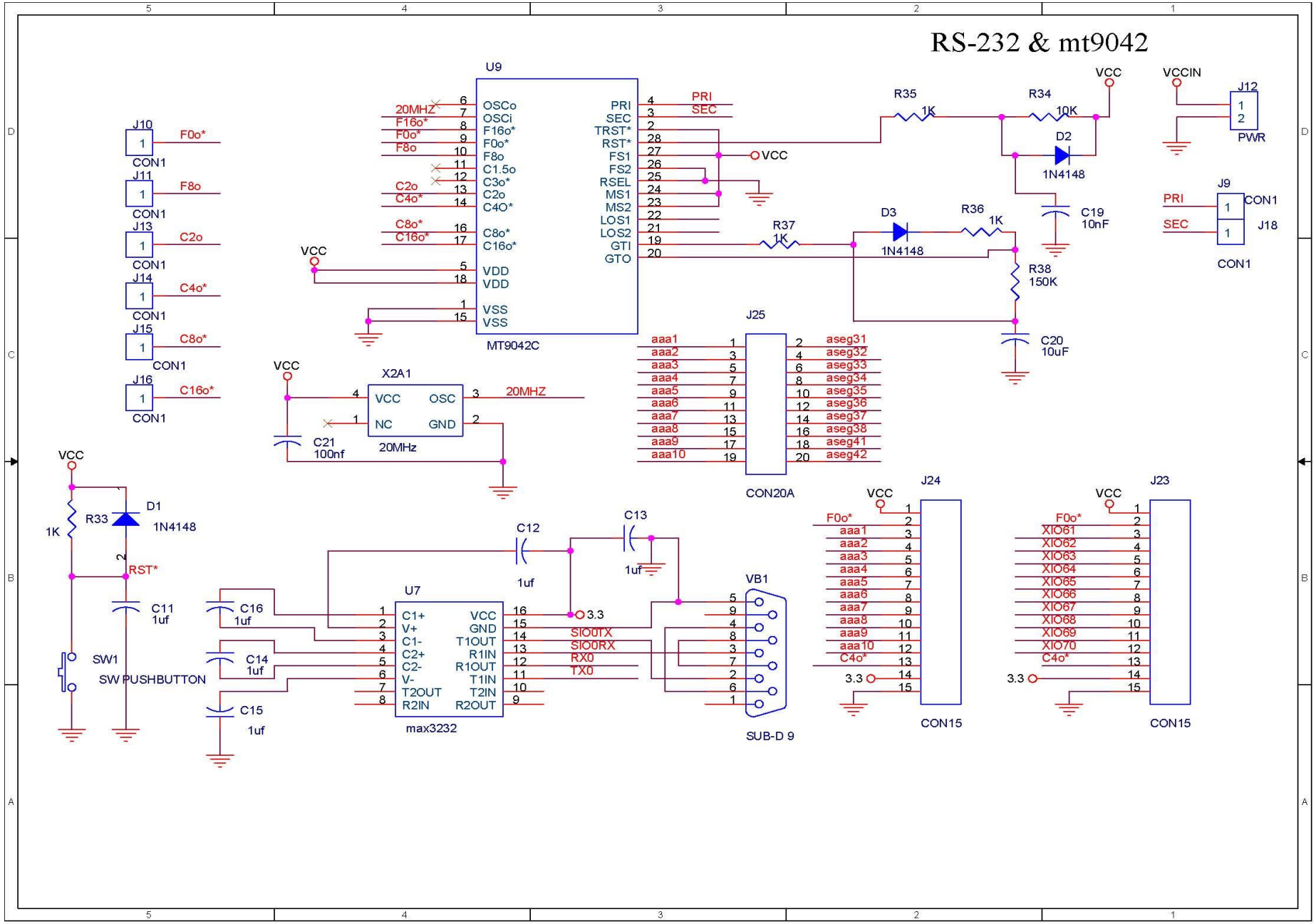
- ✓ بر روی برد چهار عدد LED و نمایشگر هفت‌قسمتی وجود دارد. از این LED ها و نمایشگرها به‌منظور نمایش خروجی ALU استفاده نمایید.
- ✓ از سوئیچ‌های موجود بر روی برد به‌عنوان ورودی‌ها، سیگنال‌های کنترلی و رقم نقلی ورودی استفاده نمایید.

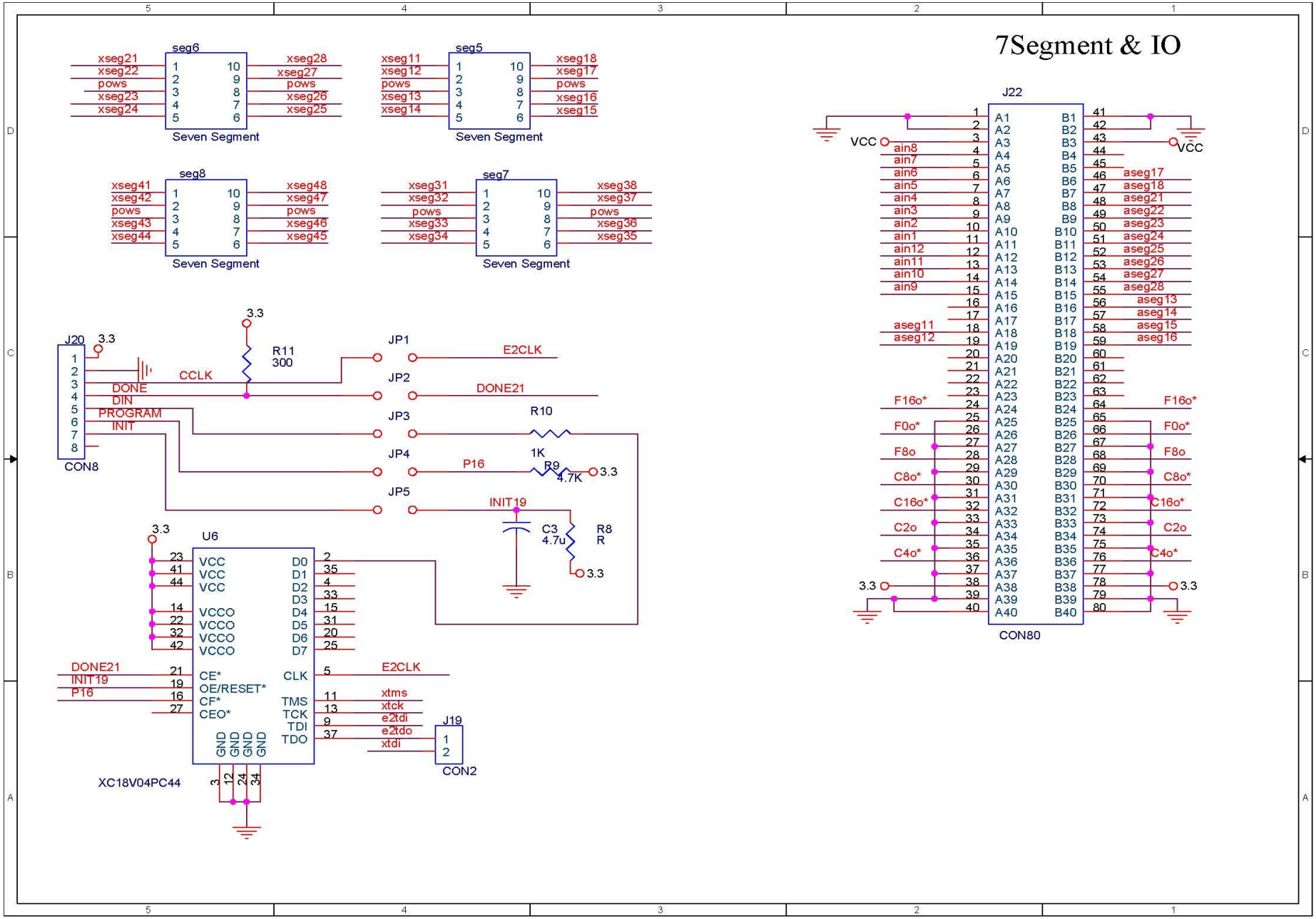
۱) برنامه مدار ترسیم‌شده را نوشته و بر روی FPGA برنامه‌ریزی نمایید.

۲) آزمودن‌های لازم را انجام و نتایج را در گزارش خود ثبت نمایید.


XILINX FPGA
U2A
U2B
U2C
U2D
XC2S150-5PQ208
CON3
CON6
CON6A
J6
J7
J8
J17
R3 4.7K
R4 4.7k
R5 4.7k
R6 4.7k
R7 4.7k
3.3
2.5
DONE
PROGRAM
M2
M1
M0
CCLK
INIT
DIN
xtdo
xtdi
xtck
xtms
e2tdo
e2tdi
TMS
TCK
TDI
TDO
GCK0
GCK1
GCK2
GCK3
VCO
VCCINT
GND


Switches & LEDs
3.3
SIP1
SW2
xin1
xin2
xin3
xin4
xin5
xin6
xin7
xin8
4.7K
SW DIP-8
SW5
VCC
SIP3
ain1
ain2
ain3
ain4
ain5
ain6
ain7
ain8
4.7K
SW DIP-8
SW3
3.3
SIP2
xin9
xin10
xin11
xin12
4.7K
SW DIP-4
SW6
VCC
SIP4
ain9
ain10
ain11
ain12
4.7k
SW DIP-4
xled9
xled10
xled11
xled12
LED5
LED6
LED7
LED8
SIP5
330
xled1
xled2
xled3
xled4
xled5
xled6
xled7
xled8
LED1
LED2
LED3
LED4
aled9
aled10
aled11
aled12
LED13
LED14
LED15
LED16
SIP6
330
aled1
aled2
aled3
aled4
aled5
aled6
aled7
aled8
LED9
LED10
LED11
LED12


RS-232 & mt9042
U9
MT9042C
OSCo
OSCi
F16o*
F0o*
F8o
C1.5o
C3o*
C2o
C4O*
C8o*
C16o*
VDD
VDD
VSS
VSS
PRI
SEC
TRST*
RST*
FS1
FS2
RSEL
MS1
MS2
LOS1
LOS2
GTI
GTO
20MHZ
F16o*
F0o*
F8o
C2o
C4o*
C8o*
C16o*
J10
J11
J13
J14
J15
J16
CON1
F0o*
F8o
C2o
C4o*
C8o*
C16o*
VCC
VCCIN
R35
1K
R34
10K
D2
1N4148
C19
10nF
J12
PWR
J9
PRI
SEC
CON1
J18
D3
1N4148
R36
1K
R37
1K
R38
150K
C20
10uF
X2A1
VCC
OSC
NC
GND
20MHZ
C21
100nf
20MHz
J25
CON20A
aaa1
aaa2
aaa3
aaa4
aaa5
aaa6
aaa7
aaa8
aaa9
aaa10
aseg31
aseg32
aseg33
aseg34
aseg35
aseg36
aseg37
aseg38
aseg41
aseg42
R33
1K
D1
1N4148
RST*
C11
1uf
SW1
SW PUSHBUTTON
C12
1uf
C13
1uf
U7
max3232
C1+
V+
C1-
C2+
C2-
V-
T2OUT
R2IN
VCC
GND
T1OUT
R1IN
R1OUT
T1IN
T2IN
R2OUT
3.3
SIO0TX
SIO0RX
RX0
TX0
C16
1uf
C14
1uf
C15
1uf
VB1
SUB-D 9
J24
CON15
J23
CON15
XIO61
XIO62
XIO63
XIO64
XIO65
XIO66
XIO67
XIO68
XIO69
XIO70


7Segment & IO
seg6
xseg21
xseg22
pows
xseg23
xseg24
xseg28
xseg27
pows
xseg26
xseg25
Seven Segment
seg5
xseg11
xseg12
pows
xseg13
xseg14
xseg18
xseg17
pows
xseg16
xseg15
Seven Segment
seg8
xseg41
xseg42
pows
xseg43
xseg44
xseg48
xseg47
pows
xseg46
xseg45
Seven Segment
seg7
xseg31
xseg32
pows
xseg33
xseg34
xseg38
xseg37
pows
xseg36
xseg35
Seven Segment
J20
CON8
DONE
DIN
PROGRAM
INIT
CCLK
R11
300
JP1
E2CLK
JP2
DONE21
JP3
R10
JP4
P16
1K
R9
4.7K
JP5
INIT19
C3
4.7u
R8
R
U6
XC18V04PC44
VCC
GND
D0
D1
D2
D3
D4
D5
D6
D7
CE*
OE/RESET*
CF*
CEO*
CLK
TMS
TCK
TDI
TDO
xtms
xtck
e2tdi
e2tdo
xtdi
J19
CON2
J22
CON80
ain8
ain7
ain6
ain5
ain4
ain3
ain2
ain1
ain12
ain11
ain10
ain9
aseg11
aseg12
aseg17
aseg18
aseg21
aseg22
aseg23
aseg24
aseg25
aseg26
aseg27
aseg28
aseg13
aseg14
aseg15
aseg16
F16o*
F0o*
F8o
C8o*
C16o*
C2o
C4o*

پیکربندی پایه‌های FPGA (Spartan3)

FPGA Pin Number	Application	FPGA Pin Number	Application	FPGA Pin Number	Application
2	7SEG3-d	68	IO16	138	ADC-D0
3	TX0	71	IO7	139	ADC-OE
4	RX0	72	IO15	140	usbD0
5	SW1-ain4	74	IO6	141	usbD3
7	SW1-ain3	76	IO14	143	usbRD
9	SW1-ain2	77	IO5	144	usbD4
10	SW1-ain1	78	IO13	146	usbD5
11	RST*	79	CLKIN	147	7SEG1-g
12	SW2-ain8	80	IO12	148	usbD6
13	SW2-ain7	81	IO3	149	7SEG1-f
15	SW2-ain6	85	IO11	150	usbD7
16	SW2-ain5	86	IO2	152	usbTXE
18	SW3-ain12	87	IO10	154	7SEG1-a
19	SW3-ain11	90	IO1	155	7SEG2-g
20	SW3-ain10	93	IO9	156	7SEG1-b
21	SW3-ain9	94	D4-aled7	161	7SEG2-a
22	20MHZ	95	usbRXF	162	7SEG2-f
24	R1A0	96	D4-aled8	165	7SEG2-b
26	R1A1	97	D3-aled5	166	7SEG1-c
27	R1A2	100	7SEG1-e	167	usbPWREN
28	R1A3	101	D3-aled6	168	7SEG2-e
29	R1CS0	102	D2-aled3	169	7SEG2-c
31	R1D0	106	D2-aled4	171	7SEG2-d
33	R1D1	107	D4-aled1	172	usbWR
34	R1D2	108	D1-aled2	175	7SEG2-DP
35	R1D3	109	IO4	176	7SEG1-d
36	R1WE	111	usbRSTn	178	7SEG1-DP
37	R1A4	113	usbSIWU	180	7SEG3-a
39	R1A5	114	DAC_D2	181	7SEG4-a
40	R1A6	115	DAC_D1	182	7SEG4-g
42	R1A7	116	DAC_D0	183	7SEG4-b
43	R1A16	117	DAC_CS	184	7SEG4-f
44	R1A15	119	DAC_RW	185	usbD2
45	R1A14	120	DAC_D3	187	7SEG4-e
46	R1A13	122	DAC_D4	189	7SEG4-c
48	R1OE	123	DAC_D5	190	usbD1
50	R1D7	124	DAC_D6	191	7SEG4-d
51	R1D6	125	DAC_D7	194	7SEG4-DP
57	R1D5	126	ADC-CLK	196	7SEG3-DP
58	R1D4	128	ADC-D7	197	7SEG3-c
61	R1A12	130	ADC-D6	198	7SEG3-b
62	R1A11	131	ADC-D5	199	7SEG3-g
63	R1A10	132	ADC-D4	200	7SEG3-f
64	R1A9	133	ADC-D3	203	7SEG3-e
65	R1A8	135	ADC-D2	204	SW3-ain12
67	IO8	137	ADC-D1	205	usbXTIN