



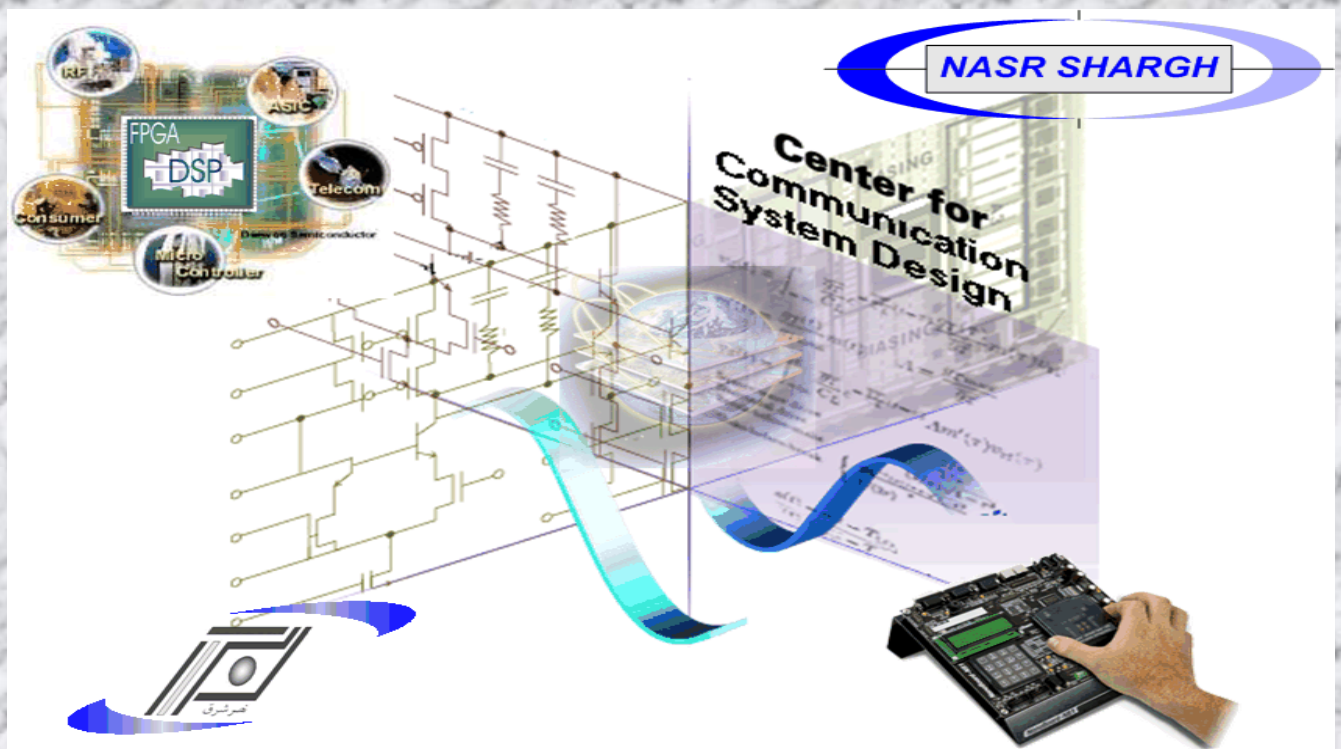
وزارت علوم تحقیقات و فناوری

دانشگاه صنعتی شجاد

دستور کار

آزمایشگاه مدار منطقی و معماری کامپیوتر (کارشناسی پیوسته)

آزمایشگاه معماری کامپیوتر (کارشناسی ناپیوسته)



تهیه کننده

شرکت نوآوران صنعت رایانه شرق

امیر باوفای طوسی

شعله هاشمی نمین

ژیلا عظیم زاده

ویرایش چهارم - زمستان ۱۳۹۵

پیشگفتار	۱
جلسه اول: آشنایی با قطعات و برد مورد و FPGA	۷
جلسه دوم: آزمایش مدار رأی گیر اکثریت بر روی برد مورد	۱۸
جلسه سوم: آزمایش مدار 7Segment و BCD بر روی برد مورد	۲۲
جلسه چهارم: آشنایی با نرم افزار Maxplus	۲۶
جلسه پنجم: آزمایش جمع کننده ها (ریپل کری و پیش بینی کننده کری)	۳۶
جلسه ششم: آزمایش دیاگرام حالت و مقسم فرکانس	۴۴
جلسه هفتم: آشنایی با زبان توصیف سخت افزاری Verilog و نرم افزار Quartus	۵۴
جلسه هشتم: آزمایش طراحی واحد محاسبه و منطق (ALU)	۷۷
جلسه نهم: آزمایش رجیسترها و ثبات ها و گذرگاه داده	۸۰
جلسه دهم: آشنایی با نرم افزار ISE Xilinx	۸۹
جلسه یازدهم: آزمایش طراحی حافظه	۱۰۰
جلسه دوازدهم: طراحی واحد کنترل به صورت سخت افزاری	۱۰۴
جلسه سیزدهم: آزمایش کامپیوتر پایه ای	۱۰۹
جلسه چهاردهم: آزمایش طراحی واحد کنترل به صورت نرم افزاری (میکرو پروگرام)	۱۱۰
جلسه پانزدهم: آزمایش توسعه دستورات	۱۱۳
جلسه شانزدهم: آزمایش حافظه خارجی و پورت سریال	۱۱۶
آشنایی با برد مبتنی بر تراشه شرکت Altera	۱۲۱
آشنایی با برد مبتنی بر تراشه شرکت Xilinx	۱۲۸

مراجع:

۱- دستور کار آزمایشگاه مدار منطقی شرکت نصر شرق

۲- دستور کار آزمایشگاه مدار منطقی دانشگاه سجاد - نوید ایزدخواستی

۳- مدارهای منطقی نلسون

تراشه‌های قابل برنامه‌ریزی با توجه به کارایی و توانایی بالا در سیستم‌های صنعتی و تحقیقاتی، باعث تحول فوق‌العاده‌ای در صنعت میکروالکترونیک شده‌اند. اما به دلیل فقدان آموزش و ابزار مورد نیاز در دانشگاه‌های کشورمان، همواره شکاف عمیقی میان محیط دانشگاهی و محیط صنعتی ملموس و محسوس بوده است.

آزمایشگاه مدار منطقی و معماری کامپیوتر با هدف ایجاد زیرساخت لازم جهت خودکارسازی طراحی مای الکترونیکی، آسان نمودن طراحی مای پیشرفته دیجیتال و ساخت نمونه مای آزمایشگاهی ارائه شده 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 سه بخش تقسیم شده است:

- کار با برد بورد و تراشه‌های منطقی (۳ جلسه، هفته اول الی سوم)،
- کار با بوردهای FPGA شرکت Altera (۶ جلسه، هفته چهارم الی نهم)،
- کار با بوردهای FPGA شرکت Xilinx (۷ جلسه، هفته دهم الی شانزدهم) و در این راستا مفاهیم ذیل به صورت عملی فراگرفته می‌شود:

- ✓ آشنایی با برد بورد و قطعات منطقی و انجام دو آزمایش از مدارهای ترکیبی و ترتیبی بر روی برد بورد
- ✓ آشنایی با ساختار تراشه‌های قابل برنامه‌ریزی FPGA
- ✓ روند طراحی با یک تراشه قابل برنامه‌ریزی FPGA
- ✓ آشنایی و برنامه‌نویسی با زبان توصیف سخت‌افزار Verilog
- ✓ آشنایی و طراحی با نرم‌افزار ISE, Quartus, Max+Plus II
- ✓ طراحی، پیاده‌سازی و آزمودن انواع مدارهای دیجیتال بر روی FPGA
- ✓ طراحی و پیاده‌سازی مدار ارتباط با نمایشگرهای مختلف از جمله مجموعه LED ها، نمایشگرهای هفت‌قسمتی، نمایشگرهای

LCD

- ✓ طراحی و پیاده‌سازی انواع پروتکل‌های سری و ارتباط سری با کامپیوتر از طریق FPGA
- شرکت نصر شرق نیز با در نظر گرفتن جایگاه مبحث طراحی‌های دیجیتال در دانشگاه‌های معتبر جهان و تصمیم بر ایجاد زیرساخت لازم به منظور توسعه این فناوری 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 قابل برنامه‌ریزی شرکت‌های XILINX و ALTERA و بردهای برنامه‌ریزی آن‌ها، دستور کار آزمایشگاه مدار منطقی و نرم‌افزارهای مورد نیاز است. شایان ذکر است

دانشگاه‌های معتبری همچون شریف و امیرکبیر از این مجموعه استفاده می‌نمایند. برای کسب اطلاعات بیشتر در زمینه‌های مختلف و مورد بحث در این آزمایشگاه می‌توانند به سایت شرکت نصر شرق^۱ مراجعه نمایند.

ساختار دستور کار آزمایشگاه

دستور کاری که در پیش روی شما قرار دارد کاری است مشترک توسط این شرکت و آقای مهندس باوفا و سر کارخانه مهندس هاشمی (اعضا هیئت علمی دانشگاه صنعتی سجاد) برگرفته از دستور کار آزمایشگاه‌های مدار منطقی دانشگاه‌های معتبر جهان می‌باشد و سعی شده است تا در تهیه آن، اهم مباحث مطرح شده درس مدار منطقی و معماری کامپیوتر تحت پوشش قرار گیرد. به طور کلی هر آزمایش در این دستور کار دارای بخش‌های زیر می‌باشد:

- | | |
|---|-----------------------|
| ✓ | اهداف آزمایش |
| ✓ | تئوری آزمایش |
| ✓ | 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 در زمینه^۲ 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 مورد نظر، مدار طراحی شده 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 پایان ترم | ۶ نمره |
| پروژه پایان ترم | ۴ نمره |

جلسه ۱

آشنایی با قطعات و برد بورد

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 قابل برنامه ریزی که در این آزمایشگاه استفاده خواهند نمود، بپردازیم. از آنجایی که در این آزمایشگاه هدف به کارگیری مطالبی است که درس مدار منطقی و معماری آموختید، بنابراین در بازگو نمودن جزئیات علمی خودداری و به توضیحی اجمالی بسنده خواهیم نمود.

۱) قطعات منطقی

همان طور که می دانید قطعات منطقی که گیت خوانده می شوند، اصلی ترین قسمت یک مدار منطقی هستند. این قطعات به منظور انجام اعمال پایه ای نظیر AND، OR، NOT، ... و یا اعمال پیچیده تری نظیر شمارنده ها، ثبات ها و ... ساخته و در بسته بندی های مختلف به بازار عرضه می شوند. اما این قطعات منطقی به روش های مختلفی ساخته می شوند که یکی از این روش ها استفاده از مدارهای مجتمع می باشد. هر مدار مجتمع یا آی سی دارای یک مشخصه عددی است که روی سطح بسته بندی آن و به منظور بیان مشخصات و پارامترهای آن چاپ می شود. لازم به ذکر است که تمام سازنده ها کتابچه راهنما یا کاتالوگ حاوی شرح دقیق و تمام اطلاعات لازم درباره آی سی های ساخت خود را چاپ می کند.

۱-۱) فناوری ساخت

با پیشرفت فناوری مدارهای ساخت مجتمع تعداد گیت هایی که می توانست در یک تراشه جای گیرد به میزان قابل توجهی افزایش یافت. مدارهای مجتمع با مقیاس کوچک (SSI) دارای چند گیت مستقل در یک بسته واحد هستند. تعداد این گیت ها معمولاً کمتر از ۱۰ و محدود به تعداد پایه ها در آی سی است. قطعات مجتمع با مقیاس متوسط (MSI) تقریباً دارای ۱۰ الی ۲۰۰ گیت در هر بسته می باشند. این قطعات معمولاً توابع دیجیتال ساده همچون دکترها، جمع کننده ها و ثبات ها را اجرا می نمایند. مدارهای مجتمع با مقیاس بزرگ (LSI) بین ۲۰۰ تا چند هزار گیت در هر بسته دارند. این بسته ها دستگاه های دیجیتالی همچون پردازنده ها، تراشه های حافظه و ماژول های قابل برنامه ریزی را شامل می شوند. قطعات مجتمع با مقیاس بسیار بزرگ (VLSI) حاوی هزاران گیت در یک بسته اند. VLSI ها به دلیل کوچکی و ارزانی انقلابی در فناوری ساخت سیستم ها کامپیوتری به وجود آورده و به طراحان امکان ساخت و ایجاد ساختارهایی را دادند که قبلاً اقتصادی نبودند.

مدارهای مجتمع نه تنها بر اساس عملکرد منطقی شان بلکه از نظر فناوری خاص مدارهایی که به آن تعلق دارند نیز طبقه بندی می شوند. فناوری به کاررفته در ساخت این مدارها را خانواده قطعات منطقی می خوانند. بسیاری از این خانواده ها به صورت مدارهای مجتمع در سطح تجاری عرضه شده اند. متداول ترین خانواده ها در زیر معرفی شده اند:

✓ TTL یا منطق ترانزیستور- ترانزیستور

✓ ECL یا منطق کوپل امیتر

✓ MOS یا منطق فلز اکسید نیمه هادی

✓ CMOS یا منطق فلز اکسید نیمه هادی مکمل

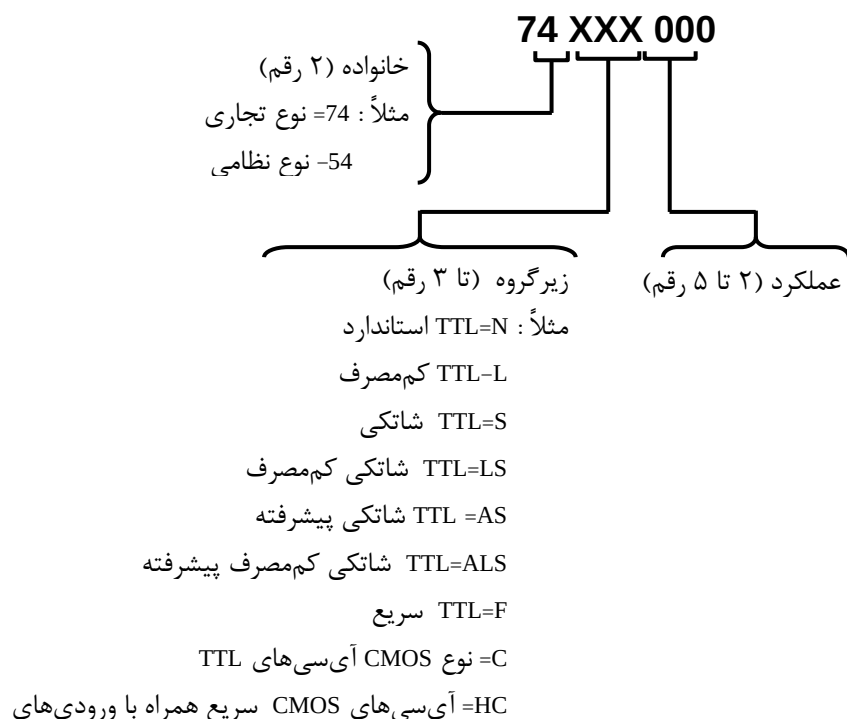
TTL یک خانواده متداول است که سال ها مورد استفاده بوده و به عنوان استاندارد تلقی می شود. ECL در سیستم هایی که به سرعت عمل بالا نیاز دارند ترجیح داده می شوند. MOS برای مدارهایی که نیاز به تراکم بالا دارند مناسب است و CMOS در سیستم های کم مصرف به کار می رود.

خانواده منطقی ترانزیستور- ترانزیستور گونه تکامل یافته فناوری قدیمی تری است که در آن از دیود و ترانزیستور برای ساخت گیت پایه NAND استفاده می شده است. بعدها برای بهبود عملکرد مدار به جای دیود از ترانزیستور استفاده شد و نام خانواده جدید ترانزیستور- ترانزیستور گذاشته شد. علاوه بر نوع استاندارد TTL انواع دیگری از این خانواده عبارت اند از TTL سرعت بالا، TTL توان پایین (یا کم مصرف)، TTL شاتکی، TTL شاتکی توان پایین و ... که در انتهای این بخش در مورد آن توضیحات بیشتری را ارائه خواهیم نمود.

منطق فلز اکسید نیمه هادی یک ترانزیستور تک قطبی است که به جریان یک نوع حامل الکتریکی وابسته است. این حامل ها ممکن است الکترون (در نوع کانال n) یا حفره باشند. MOS کانال p را PMOS و MOS کانال n را NMOS می نامند. در فناوری CMOS هر دو نوع ترانزیستور، که به شکل مکمل در تمام مدارها بسته شده اند به کاررفته است. بزرگ ترین مزیت CMOS نسبت به دوقطبی، تراکم بالای مدارها، ساده بودن تکنیک ساخت و عملکرد مقرون به صرفه آن به دلیل مصرف توان کم آن است.

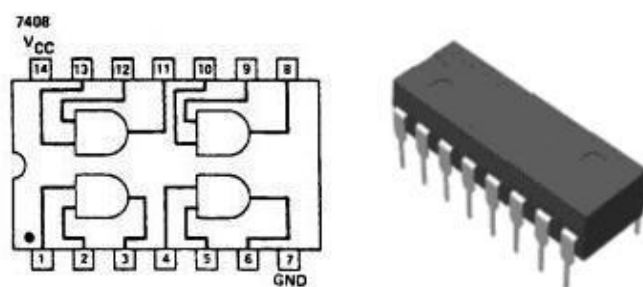
۱-۲) آی سی های دیجیتال سری 74

سال های متمادی است که ۸ زیرگروه آی سی های TTL و ۸ زیرگروه آی سی های CMOS در سری 74 قرار دارند. بیشتر این زیرگروه ها دیگر قدیمی بشمار آمده یا منسوخ شده اند. اما توجه داشته باشید که بیشتر زیرگروه با یکدیگر سازگار هستند. بنابراین به سادگی و با توجه به مشخصات قطعه مورد نظر که از روی شماره آن قابل استخراج است، می توان آن ها را با قطعه از خانواده جدیدتر جایگزین کرد. با این تفاسیر در ابتدا می بایست نحوه تعیین مشخصات قطعه از روی شماره آن را فراگیریم. در شکل زیر طرح های اساسی مورد استفاده برای مشخص کردن کدهای سری 74 را نشان می دهد.



شکل ۱-۱: روش پایه ای کدهای در آی سی های سری ۷۴

همان طور که در شکل نیز مشاهده می نمایید، در ساده ترین حالت کدهای حرفی- عددی آی سی ها از ۳ کد فرعی مسلسل تشکیل شده است. اولین کد فرعی عددی دورقمی است، که مشخص کننده حوزه کاربردی آی سی است و شامل کاربردهای تجاری (74)، نظامی (54) و قطعات رابط (75) می باشد. کد دوم حداکثر می تواند از ۳ حرف تشکیل شده باشد. این کد فناوری مورد استفاده در آی سی را مشخص می نماید.



شکل ۲-۱: بسته بندی فیزیکی قطعات

کد سوم که عدد ۲ تا ۵ رقمی می باشد، بیانگر عملکرد آی سی است. بنابراین کد به کاررفته در آی سی ها سری ۷۴ می توانند شامل 7414، 74HC03 و 74LS38 باشد. در انتها نیز در شکل فوق نمونه ای از بسته بندی این قطعات آر مشاهده می نمایید.

۲) تراشه های منطقی قابل برنامه نویسی^۱

تراشه منطقی قابل برنامه نویسی شامل مجموعه ای از المان های منطقی قابل برنامه نویسی و flip-flop می مرتبط باهم بعلاوه اتصال های قابل برنامه نویسی می باشد. کاربر می تواند با کمک سلول های حافظه^۲ وظیفه و برنامه هر قسمت از این مجموعه را مشخص و کنترل کند. اگرچه تراشه های متفاوت از ساختارهای^۳ مختلفی استفاده می کنند ولی اساس کار تمام آن ها یکی می باشد.

۲-۱) انواع تراشه های منطقی قابل برنامه نویسی^۴

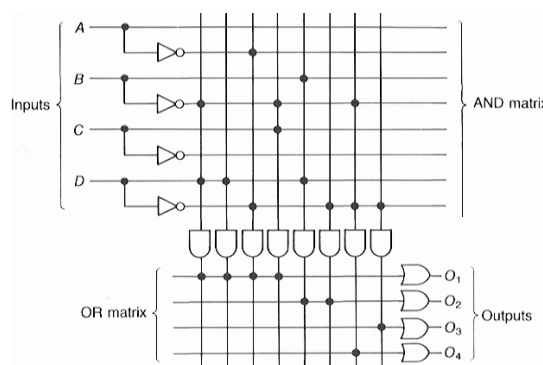
امروزه تعداد زیادی تراشه های قابل برنامه نویسی توسط شرکت های مختلف و با ساختارهای متفاوت در دسترس می باشد که می توان آن ها را در ۴ گروه عمده طبقه بندی نمود:

(SPLDs) Simple Programmable Logic Devices	طرح های منطق قابل برنامه نویسی ساده
(CPLDs) Complex Programmable Logic Devices	طرح های منطق قابل برنامه نویسی پیچیده
(FPGAs) Field Programmable Gate Arrays	آرایه های گیت قابل برنامه نویسی میدانی
(FPGAs) Field Programmable Inter Connect	اتصالات قابل برنامه نویسی میدانی

SPLD (۱-۱-۲)

SPLD ها، کوچک ترین و کم هزینه ترین شکل از تراشه های منطقی قابل برنامه نویسی می باشند. یک SPLD معمولاً شامل ۴ تا ۲۲ macrocell است که می تواند تعداد زیادی از TTL می سری ۷۴۰۰ را در خود جای دهد. هر macrocell به طور کامل با macrocell دیگر در ارتباط است. زیرمجموعه می این خانواده بدین شرح می باشند:

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



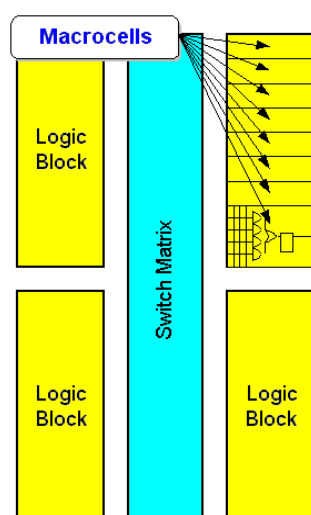
شکل ۱-۳: PLA

CPLD (۲-۱-۲)

CPLD

ها شبیه SPLD می‌باشند. با این تفاوت که دارای ظرفیت بسیار بالاتری هستند. یک CPLD مشخص هم‌ارز ۲ تا ۶۴ SPLD است و به‌طور معمول شامل ۱۰ تا ۱۰۰ macrocell می‌باشند. ۸ تا ۱۶ macrocell باهم تشکیل یک بلوک عملیات^۱ بزرگ‌تر می‌دهند. macrocell می‌تواند یک بلوک عملیاتی به‌طور کامل باهم ارتباط دارند. زیرمجموعه‌های 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)



در یک تعریف جامع، CPLD ها از چندین بلوک منطقی^۲ مانند PAL تشکیل شده‌اند که از طریق ماتریس سوئیچ^۳ 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.

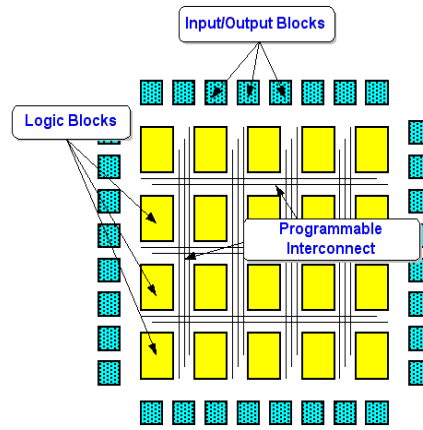
شکل ۴-۱: بلوک منطقی و ماتریس سوئیچ

FPGA (۳-۱-۲)

FPGA تراشه‌ای قابل برنامه‌ریزی است که از آرایه‌ای از بلوک‌های منطقی، بلوک‌های ورودی/خروجی^۴ و ماتریس‌های سوئیچ تشکیل شده است. شکل ۱-۲ 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 قابل برنامه‌ریزی انجام می‌گیرد. ساختار FPGA با ساختار خانواده‌های SPLD و CPLD متفاوت است و نسبت به آن‌ها بیشترین ظرفیت‌المان‌های 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:

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

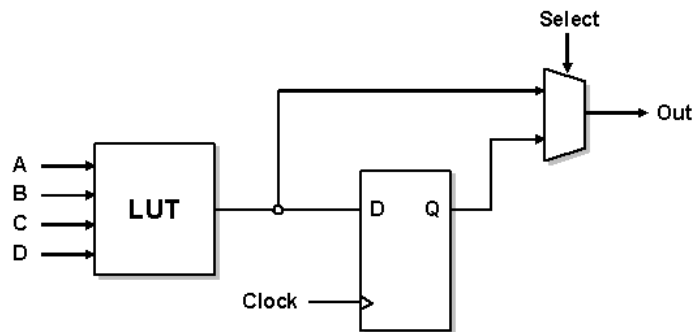
- **pASIC** (Altera)
- **FLEX, APEX** (Altera)
- **ACT** (Actel)
- **STRATIX** (Altera)
- **VIRTEX** (Xilinx)
- **SPARTAN** (Xilinx)



شکل ۵-۱: ساختار داخلی تراشه

۲-۱-۳) بلوکهای منطقی

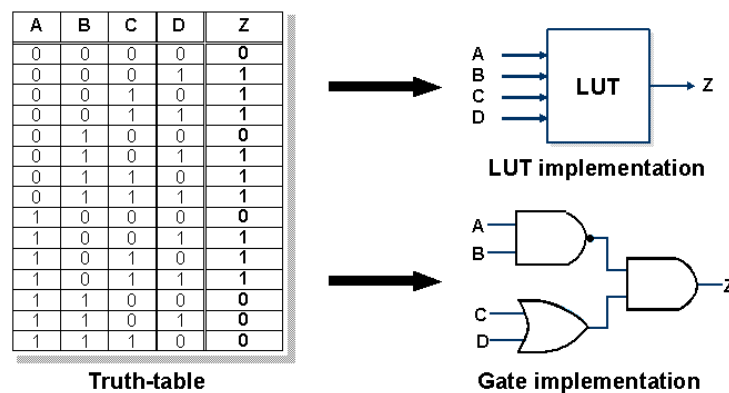
مداری که در FPGA پیاده‌سازی می‌شود، به قطعات کوچک‌تر تجزیه‌شده، و هر یک از این قطعات در یک سلول منطقی پیاده‌سازی می‌شود. FPGA به‌طور کلی از دو نوع سلول منطقی استفاده می‌نماید.



شکل ۶-۱: بلوکهای منطقی

الف) سلول‌های منطقی مبتنی بر LUT^۱

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 ۷-۱: سلول‌های منطقی مبتنی بر LUT

ب) سلول‌های منطقی مبتنی بر مالتی پلکسر^۱

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.

۲-۲) برنامه‌نویسی تراشه‌های منطقی

برنامه‌نویسی تراشه‌ها در ۴ مرحله انجام می‌گیرد.

۱) طراحی^۲

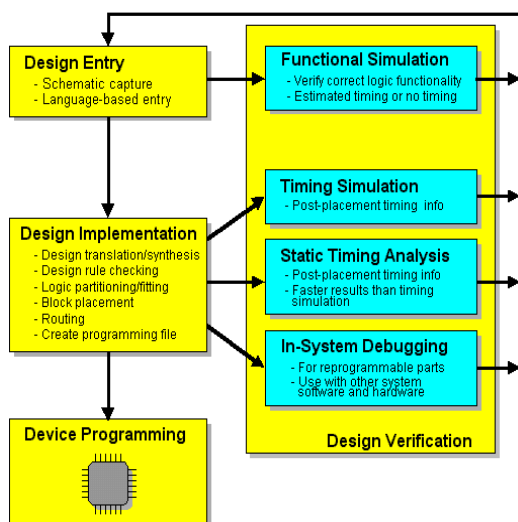
امروزه ابزارهای قابل استفاده متنوعی از شرکت‌های مختلف برای طراحی مدار در دسترس کاربران و برنامه‌نویسان می‌باشد. در تمام این

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 درحالی که برخی

دیگر ترجیح می‌دهند



شکل ۱-۸: 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^۳

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

سپس نرم‌افزار طرح را به بلوک‌های منطقی تقسیم^۴ کرده، طوری که برای پیاده‌سازی بر روی تراشه منطقی قابل برنامه‌نویسی مورد

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

نظر آماده باشد. قسمت‌بندی مهم‌ترین گام برای FPGA ها و CPLD ها می‌باشد. در نتیجه تقسیم‌بندی درست و مناسب در FPGA ها از سیم‌کشی طولانی جلوگیری شده و کارایی بالا می‌رود همچنین در CPLD کارایی و چگالی افزایش پیدا می‌کند. 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.

نرم‌افزار پیاده‌سازی هنگام جایگذاری کردن بلوک‌های منطقی طول سیم‌کشی و ازدحام مسیر سیم‌کشی را کنترل می‌کند. هنگامی که 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 فناوری بکار رفته در تراشه بستگی دارد. در بیشتر فناوری‌هایی که دارای PROM جهت بارگذاری FPGA بر اساس SRAM می‌باشند، به‌نوعی به پروگرامر نیاز است.

(۳-۲) 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

FPGAها اولین بار در سال ۱۹۸۵ توسط شرکت Xilinx معرفی شدند. از آن زمان به بعد FPGA های متفاوتی توسط شرکت‌های دیگر تولید گردید. از آنجایی که در این آزمایشگاه از تراشه‌های شرکت‌های Xilinx و Altera استفاده شده است، لذا لازم است تا کمی با محصولات این شرکت‌ها آشنا شویم.

۳- Placement

۴- Binary

۵- Verification

۱- Timing Simulation

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
- ✓ Spartan-II 2/5v
- ✓ Spartan-II E 1/8v

☛ Virtex FPGA

- ✓ Virtex-II
- ✓ 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

۲-۴) 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. از آنجایی که تنوع قطعات قابل‌برنامه‌ریزی زیاد می‌باشد بنابراین طیف وسیعی از

۲- Programmable logic devices

۱- System-on-a-programmable-chip

۲- High-Density

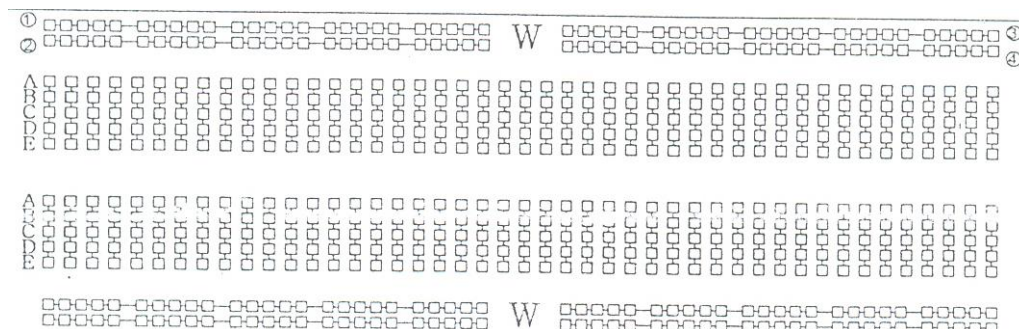
۳- High-Volume

۴- Mid-Density

نرم افزارهای مرتبط با این موضوع نیز در دسترس است. اما با توجه به این که در این آزمایشگاه از تراشه‌ها شرکت‌های Xilinx و Altera استفاده شده است، لذا در انتهای این دستور کار با دو برد از شرکت آل ترا (ص ۱۲۱) و شرکت زایلینکس (ص ۱۲۸) آشنا خواهید شد.

- برد برد (Bread board):

برد برد وسیله‌ای برای بستن مدارهای مختلف الکترونیکی جریان پایین است و از آنجاکه می‌توان به راحتی آی سی و قطعات مختلف الکترونیکی را روی آن سوار و ارتباط مای لازم را با سیم و بدون لحیم کاری برقرار کرد، کاربرد وسیعی در آزمایشگاه‌ها و مراکز تحقیقاتی دارد. در برد برد تعدادی از سوراخ‌ها از زیر به یکدیگر متصل می‌باشند. در شکل زیر اتصال برد برد آمده است:



شکل ۹- Bread board

قسمت مای ۱، ۲، ۳ و ۴ به صورت افقی به هم متصل بوده و اتصال عمودی ندارند و به وسیله حرف W نوشته شده روی برد برد قسمت ۱ و ۲ از قسمت ۳ و ۴ جدا می‌شوند. از قسمت افقی بیشتر برای تغذیه و زمین مدار استفاده می‌شود. حروف A, B, C, D, E به صورت عمودی به هم متصل بوده و اتصال افقی ندارند.

- پروب منطقی (Logic Prob)

از این وسیله برای بررسی ولتاژ نقطه‌ای از مدار استفاده می‌شود به گونه‌ای که با قراردادن 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 پیاده سازی می‌شوند با ولتاژ ثابت 5^V کار می‌کنند. به همین خاطر برای تأمین ولتاژ مورد نیاز مدارها از دستگاهی (منبع تغذیه دابل) استفاده می‌شود که ولتاژ ثابت 5^V را تولید می‌کند و دارای دو پایانه مثبت و منفی است که از پایانه مثبت برای V_{CC} یا ولتاژ بالا و از پایانه منفی به منظور زمین (GND) مدار استفاده می‌شود.

- سیگنال ژنراتور:

برای تولید کلاک و فرکانس مای مورد نیاز از این دستگاه استفاده می‌شود که انواع سیگنال‌های سینوسی، مثلثی و مربعی و خروجی TTL را تولید نموده و می‌توان فرکانس مورد نظر را در یک محدوده فرکانس از 0.1 هرتز تا یک مگاهرتز تنظیم نمود. در بخش طراحی و پیاده سازی 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 قابل تبدیل به جریان و ولتاژ. از اسیلوسکوپ برای اندازه‌گیری مای مختلف می‌مانند ولتاژ AC یا DC ، مقدار فرکانس، مقادیر پیک ولتاژ و غیره استفاده می‌شود.

نکات قابل توجه:

- 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 200Ω (اهم) به صورت سری استفاده کنید.
- ۴- سر مثبت منبع تغذیه ۵ ولت را به یکی از سوراخ مای بالاترین ردیف صفحه آزمایش متصل نمایید و سر منفی منبع تغذیه را به یکی از سوراخ‌های پایین‌ترین ردیف صفحه آزمایش متصل کنید از این پس ردیف بالای صفحه آزمایش ردیف ۵ ولت و ردیف پایین ردیف زمین «۰ ولت» نامیده خواهد شد. اکنون
- سر شماره ۷ تراشه ۷۴۰۰ را به وسیله سیم به ردیف زمین متصل نمایید. این کار با اتصال دادن یکی از سوراخ مای ستون زیر سر شماره ۷ به ردیف زمین انجام می‌شود. پایه شماره ۱۴ تراشه ۷۴۰۰ را به وسیله سیم به ردیف ۵ ولت متصل نمایید. اکنون تراشه به طور صحیح تغذیه شده است.
- ولتاژ ورودی صفر و ۵ ولت با اتصال دادن پایه شماره ۱ به ردیف 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 به منظور محدود کردن شدت جریان و جلوگیری از سوختن دیود و تراشه به کار رفته است و مقدار آن حدود ۱۰۰ تا ۳۰۰ اهم می‌باشد. ورودی مای مدار نبایست بازماند. با توجه به نوع تراشه به ۰ و یا یک منطقی متصل نمایید.

جلسه ۲

آزمایش مدار رأی گیری اکثریت بر روی برد مورد و شمارنده (مدارهای ترکیبی)

هدف

✓ آشنایی با آی-سی مای ۷۴۰۰ و ۷۴۱۰۷

✓ اهمیت ساده سازی توابع منطقی و پیاده سازی با تراشه های ۷۴۰۰ و ۷۴۱۰۷

تئوری آزمایش:

آزمایش ۱: یک مدار رأی گیری اکثریت برای یک شورا ۵ نفره (A,B,C,D,E) به گونه ای طراحی کنید. که شخص A دارای حق وتو باشد.

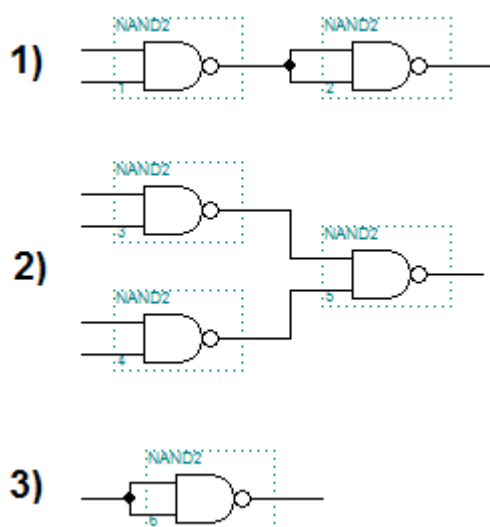
و هرگاه اکثریت آراء حاصل شد یک چراغ روشن شود (چراغ با یک منطق روشن و صفر منطق خاموش می شود)
جدول صحت مدار را به دست آورید.

تابع خروجی را رسم کنید.

با توجه به نکات ۴، ۵ و با استفاده از تراشه ۷۴۰۰ (به شکل ۱-۲ مراجعه شود) مدار رأی گیر را بر روی bread board ببندید

تکالیف پیش از آزمایش:

مدارات زیر با استفاده از گیت NAND طراحی شده اند ضمن پیاده سازی با آی-سی ۷۴۰۰ جدول درستی هریک را بنویسید و بیان کنید معادل چه گیت منطقی هستند.

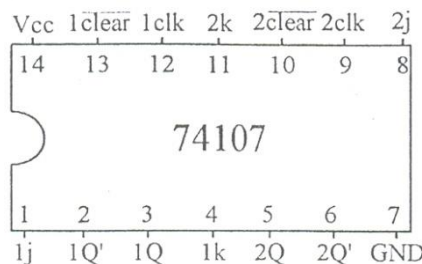


تمرین: تابع F را با حداقل گیت پیاده‌سازی نمایید و با آی‌سی ۷۴۰۰ بر روی برد بورد ببندید.

$$F=XY+X'Y'Z'+X'YZ'$$

آزمایش دوم:

آی‌سی ۷۴۱۰۷ را که حاوی ۲ فلیپ فلاپ JK با پایه پاک‌کننده (Clear) است بر روی صفحه آزمایش قرارداده و عملکرد فلیپ فلاپ JK را به کمک آن بررسی کنید. پایه Clear با صفر فعال می‌شود پس برای کار عادی آی‌سی باید این پایه را به یک وصل کنید.



شکل ۲-۲

پس از بررسی عملکرد تراشه ۷۴۱۰۷ با استفاده از آن یک شمارنده دوبیتی طراحی کنید.

جدول صحت مدار را به دست آورید.

تابع خروجی را رسم کنید و این مدار را بر روی breadboard ببندید.

برای مشاهده خروجی یک مدار به کمک LED (دیود نورانی). خروجی را توسط یک مقاومت به آند LED (معمولاً پایه‌ی بلند) وصل

و کاند آن را (معمولاً پایه کوتاه‌تر) را به زمین متصل می‌کنیم. این مقاومت به منظور محدود کردن شدت جریان و جلوگیری از

سوختن دیود و تراشه به کاررفته است و مدار آن حدود ۱۰۰ تا ۳۰۰ اهم می‌باشد.

MM74HC00

Quad 2-Input NAND Gate

General Description

The MM74HC00 NAND gates utilize advanced silicon-gate CMOS technology to achieve operating speeds similar to LS-TTL gates with the low power consumption of standard CMOS integrated circuits. All gates have buffered outputs. All devices have high noise immunity and the ability to drive 10 LS-TTL loads. The 74HC logic family is functionally as well as pin-out compatible with the standard 74LS logic family. All inputs are protected from damage due to static discharge by internal diode clamps to V_{CC} and ground.

Features

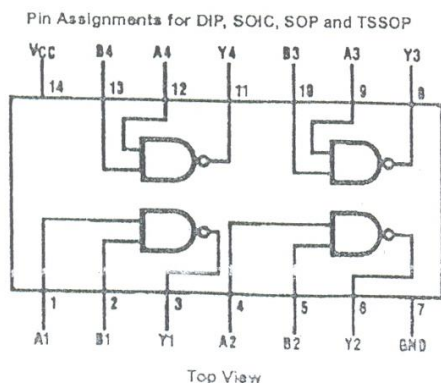
- Typical propagation delay: 8 ns
- Wide power supply range: 2–6V
- Low quiescent current: 20 μ A maximum (74HC Series)
- Low input current: 1 μ A maximum
- Fanout of 10 LS-TTL loads

Ordering Code:

Order Number	Package Number	Package Description
MM74HC00M	M14A	14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150" Narrow
MM74HC00MX_NL	M14A	Pb-Free 14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150" Narrow
MM74HC00SJ	M14D	Pb-Free 14-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
MM74HC00MTC	MTC14	14-Lead Thin Shrink Small Outline Package (TSSOP), JEDEC MO-153, 4.4mm Wide
MM74HC00MTCX_NL	MTC14	Pb-Free 14-Lead Thin Shrink Small Outline Package (TSSOP), JEDEC MO-153, 4.4mm Wide
MM74HC00N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
MM74HC00N_NL	N14A	Pb-Free 14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.
Pb-Free package per JEDEC J-STD-020B.

Connection Diagram



Logic Diagram



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

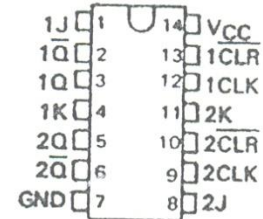
description

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 transition. For these devices the J and K inputs must be stable while the clock is high.

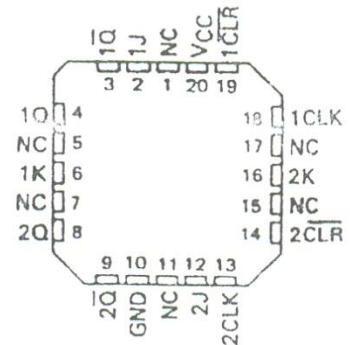
The 'LS107A contain two independent negative-edge-triggered flip-flops. The J and K inputs must be stable prior to the high-to-low clock transition for predictable operation. When the clear is low, it overrides the clock and data inputs forcing the Q output low and the $\bar{Q}$ output high.

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	
CLR	CLK	J	K	Q	$\bar{Q}$
L	X	X	X	L	H
H	$\downarrow$	L	L	Q_0	$\bar{Q}_0$
H	$\downarrow$	H	L	H	L
H	$\downarrow$	L	H	L	H
H	$\downarrow$	H	H	TOGGLE	TOGGLE

'LS107A
FUNCTION TABLE

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

شکل ۲-۲

آزمایش BCD to SevenSegment

هدف

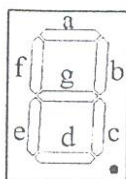
✓ آشنایی با تراشه ۷۴۴۸ و نمایشگر هفت‌قسمتی

تئوری آزمایش:

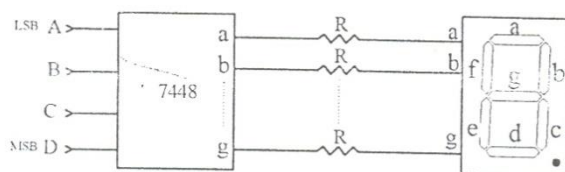
برای نشان دادن اعداد ذخیره‌شده در ثبات‌ها و یا شمارنده‌ها، می‌توان از نمایشگرهای هفت‌قسمتی استفاده کرد. نمایشگرهای هفت‌قسمتی از هفت دیود منتشرکننده نور (در بعضی موارد یک دیود هم برای نقطه اعشاری) ساخته شده‌اند. ترکیبات انتخابی LED ها، برای ایجاد ارقام عددی و دیگر سمبل‌ها روشن می‌شود. یک LED، هنگامی که ولتاژ در ورودی آن به اندازه کافی مثبت‌تر از ولتاژ پایین به کاتد باشد روشن می‌شود. در مدار دیجیتال، این ولتاژها با اعمال یک ولتاژ بالا به آن 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. در نتیجه اعمال یک ولتاژ منطق صفر موجب روشن شدن LED می‌گردد، در صورتی که 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 قرار داده و ارتباط بین پایه‌ها و دیود مای 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 با اتصال 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.



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.



- حال پایه شماره ۳ تراشه (Lamp Test) 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 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 مقاومت مای 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.

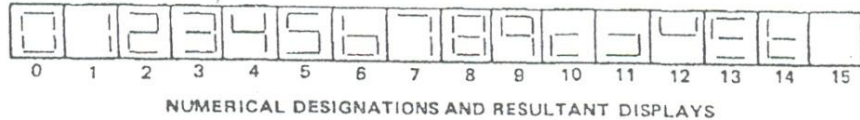
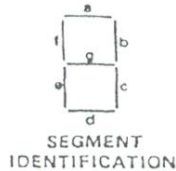
تمرین- ترکیب مدار شمارنده که با تراشه ۷۴۱۰۷ در جلسه ۲ انجام دادید را با تراشه‌های ۷۴۴۸ و نمایشگر هفت‌قسمتی بر روی breadboard ببندید.

description

The '46A, '47A, and 'LS47 feature active-low outputs designed for driving common-anode LEDs or incandescent indicators directly. The '48, 'LS48, and 'LS49 feature active-high outputs for driving lamp buffers or common-cathode LEDs. All of the circuits except 'LS49 have full ripple-blanking input/output controls and a lamp test input. The 'LS49 circuit incorporates a direct blanking input. Segment Identification and resultant displays are shown below. Display patterns for BCD input counts above 9 are unique symbols to authenticate input conditions.

The '46A, '47A, '48, 'LS47, and 'LS48 circuits incorporate automatic leading and/or trailing-edge zero-blanking control (RBI and RBO). Lamp test (LT) of these types may be performed at any time when the BI/RBO node is at a high level. All types (including the '49 and 'LS49) contain an overriding blanking input ($\overline{\text{BI}}$), which can be used to control the lamp intensity by pulsing or to inhibit the outputs. Inputs and outputs are entirely compatible for use with TTL logic outputs.

The SN54246/SN74246 and '247 and the SN54LS247/SN74LS247 and 'LS248 compose the $\overline{6}$ and the $\overline{9}$ with tails and were designed to offer the designer a choice between two indicator fonts.



'46A, '47A, 'LS47 FUNCTION TABLE (T1)

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

H = high level, L = low level, X = irrelevant

- NOTES:
1. The blanking input ($\overline{\text{BI}}$) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input ($\overline{\text{RBI}}$) must be open or high if blanking of a decimal zero is not desired.
 2. When a low logic level is applied directly to the blanking input ($\overline{\text{BI}}$), all segment outputs are off regardless of the level of any other input.
 3. When ripple-blanking input ($\overline{\text{RBI}}$) and inputs A, B, C, and D are at a low level with the lamp test input high, all segment outputs go off and the ripple-blanking output ($\overline{\text{RBO}}$) goes to a low level (response condition).
 4. When the blanking input/ripple blanking output ($\overline{\text{BI}}/\overline{\text{RBO}}$) is open or held high and a low is applied to the lamp-test input, all segment outputs are on.

$^\dagger \overline{\text{BI}}/\overline{\text{RBO}}$ is wire AND logic serving as blanking input ($\overline{\text{BI}}$) and/or ripple-blanking output ($\overline{\text{RBO}}$).

SN5446A, '47A, '48, SN54LS47, 'LS48, 'LS49 SN7446A, '47A, '48, SN74LS47, 'LS48, 'LS49 BCD-TO-SEVEN-SEGMENT DECODERS/DRIVERS

SDLS111 - MARCH 1974 - REVISED MARCH 1988

'46A, '47A, 'LS47 feature

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

'48, 'LS48 feature

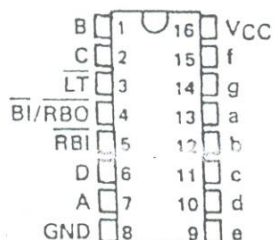
- Internal Pull-Ups Eliminate Need for External Resistors
- Lamp-Test Provision
- Leading/Trailing Zero Suppression

'LS49 feature

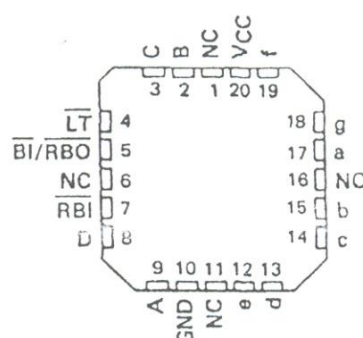
- Open-Collector Outputs
- Blanking Input

SN5448A, SN5447A, SN54LS47, SN5448,
SN54LS48 . . . J PACKAGE
SN7446A, SN7447A,
SN7448 . . . N PACKAGE
SN74LS47, SN74LS48 . . . D OR N PACKAGE

(TOP VIEW)

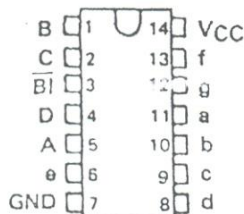


SN54LS47, SN54LS48 . . . FK PACKAGE
(TOP VIEW)

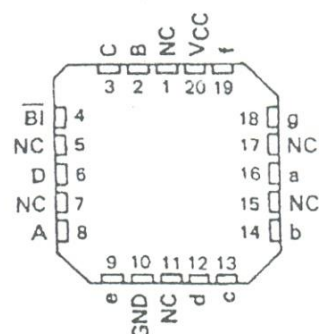


SN54LS49 . . . J OR W PACKAGE
SN74LS49 . . . D OR N PACKAGE

(TOP VIEW)



SN54LS49 . . . FK PACKAGE
(TOP VIEW)



NC - No Internal connection

شکل ۲-۳

جلسه ۴

آشنایی با نرم افزار MAX+Plus II

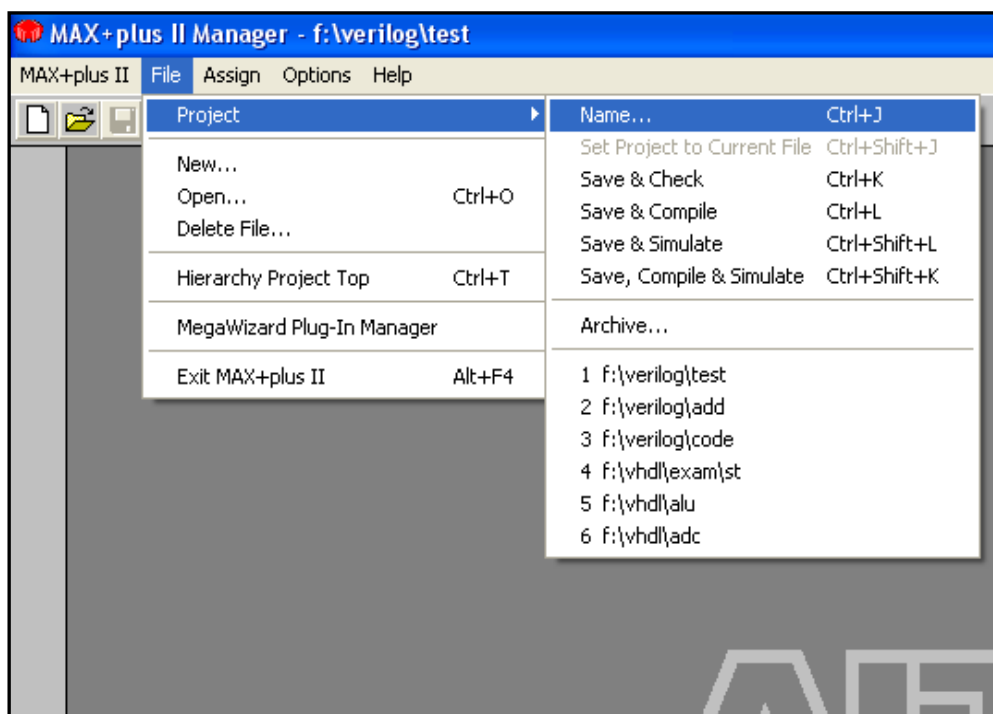
با آشنایی مختصری که در زمینهٔ FPGA حاصل شد و همچنین معرفی دو شرکت معروف در زمینهٔ طراحی‌های دیجیتال، این جلسه با نرم افزار شرکت ALTERA آشنا می‌شوید. شرکت ALTERA دارای دو نرم افزار معروف می‌باشد:

۱- MAX+ plus II

۲- QUARTUS

در این آزمایشگاه دانشجویان می‌بایست نحوه کار با این دو نرم افزار را فرا گرفته و با آن دو کار نمایند. لازم به ذکر است که در شرح این آزمایش و در کل دستور کار کلمه MAX به جای نرم افزار MAX + PLUS II مورد استفاده قرار گرفته است.

در این جلسه به معرفی نرم افزار MAX+PLUS II می‌پردازیم و در جلسه ۷ به معرفی نرم افزار QUARTUS خواهیم پرداخت. برای شروع یک طراحی دیجیتال می‌بایست محیط نرم افزار مورد نظر را باز نماییم و به این منظور با استفاده از فایل اجرایی MAX2WIN.EXE که در مسیر \MAXPLUS2\ قرار دارد یا با استفاده از منوی START نرم افزار MAX را اجرا نماییم. در کلیه نرم افزارهای جدید برای شروع یک طراحی باید پروژه‌ای را باز نمود. در این نرم افزار نیز با استفاده از منوی FILE و انتخاب PROJECT پروژه‌ای را با نام دلخواه در مسیر مورد نظر خود باز می‌کنید.



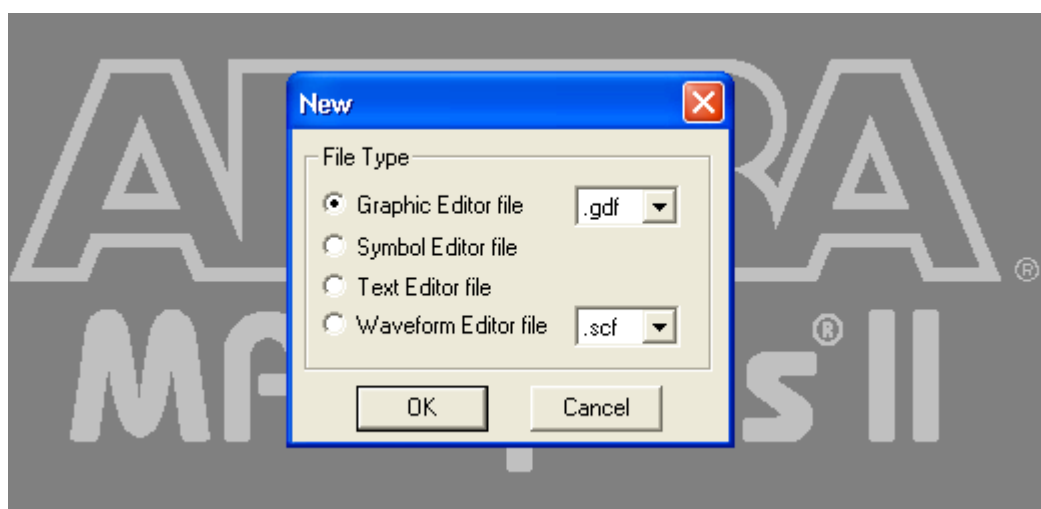
شکل ۴-۱

اکنون پروژه‌ای برای طراحی دیجیتال شما در نظر گرفته شده است. برای شروع باید فایل طراحی شما در نرم افزار وارد گردد. دو نوع فایل طراحی وجود دارد:

۱- متنی (TEXT)

۲- گرافیک (GRAPHIC)

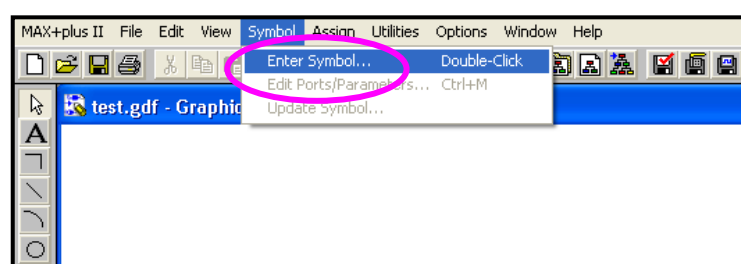
در فایل‌های متنی برای طراحی از زبانه‌ای HDL نظیر VERILOG و VHDL استفاده می‌گردد. پسوند فایل‌های متنی با زبانه‌ای مذکور به ترتیب *.V و *.VHD است. شرکت ALTERA زبانی مخصوص به خود تحت عنوان AHDL نیز دارد که پسوند فایل‌های آن *.AHD می‌باشد. در آزمایشگاه مدار منطقی به علت نداشتن اطلاعات کافی دانشجویان از طراحی‌های دیجیتال و به‌خصوص زبانه‌ای HDL، آزمایش‌ها از طریق صفحه گرافیکی این نرم‌افزار انجام خواهد شد. برای شروع طرح گرافیکی باید صفحه گرافیکی در نرم‌افزار باز گردد. برای این منظور گزینه NEW از منوی FILE را انتخاب و سپس پنجره شکل ۴-۲ باز می‌شود.



شکل ۴-۲

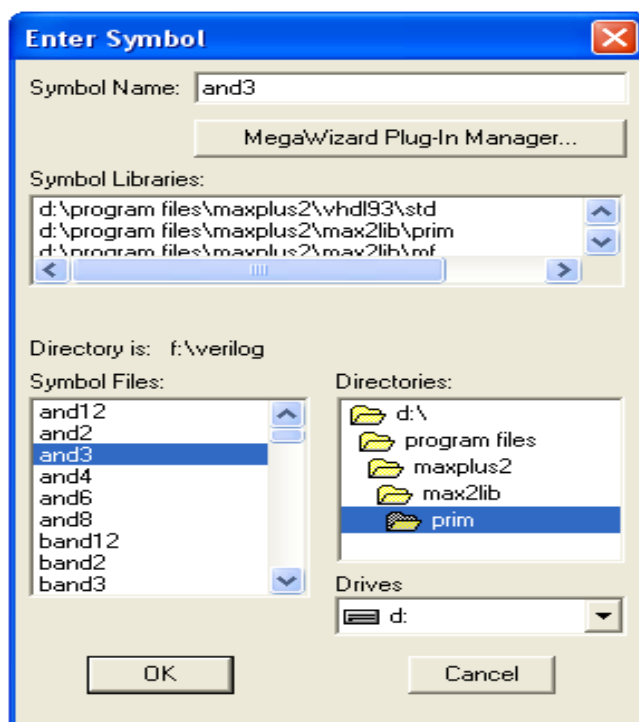
۴ گزینه موجود در این پنجره به تدریج در این جلسه معرفی خواهند شد. گزینه اول فایل گرافیکی و گزینه سوم فایل متنی را ایجاد می‌کند. اکنون با کلیک کردن روی فایل گرافیکی، فایلی برای طراحی دیجیتال در اختیار شما قرار می‌گیرد. با باز شدن صفحه گرافیکی طراحی نیز آغاز می‌شود. این صفحه مانند دیگر نرم‌افزارهای معمول به‌منظور ترسیم شماتیک نظیر نرم‌افزار ORCAD است. به‌منظور طراحی مدار مورد نظر ابتدا المان‌ها را انتخاب و سپس ارتباطات المان‌ها با یکدیگر برقرار نمایید. انتخاب المان‌ها به دو صورت انجام می‌گیرد:

۱- با استفاده از Symbol که در بالای صفحه قرار دارد (شکل ۴-۳).



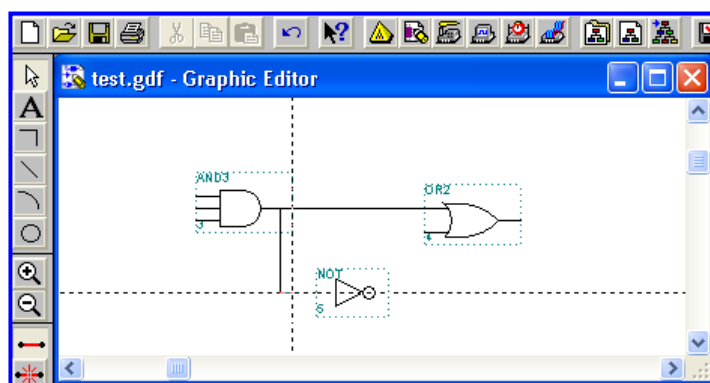
شکل ۴-۳

۲- با استفاده از دکمه سمت راست MOUSE گزینه ENETER SYMBOL را کلیک کرده، صفحه زیر باز خواهد شد:



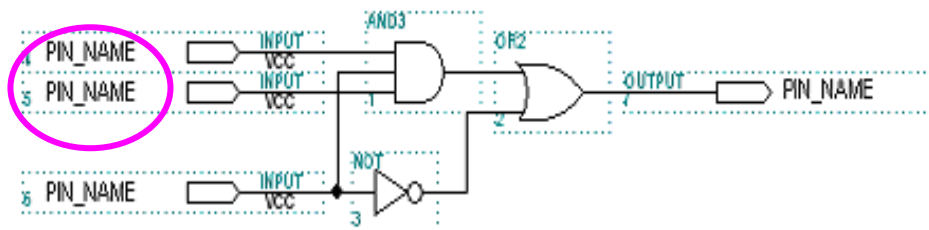
شکل ۴-۴

در این قسمت کتابخانه‌های مختلفی وجود دارد. ابتدا باید کتابخانه‌ای که المان موردنظر شما در آن قرار دارد مشخص شود. سپس المان موردنظر انتخاب می‌گردد. به‌طور مثال در شکل ۱۲ از کتابخانه PRIM المان AND3 انتخاب شده است. بعد از انتخاب المان‌های طراحی اکنون باید ارتباطات میان آن‌ها برقرار گردد. برای برقراری ارتباط یا به تعبیری WIRE، باید با استفاده از MOUSE روی پایه المان موردنظر رفته و در این حالت اشاره‌گر Mouse به علامت بعلاوه تبدیل شود. سپس کلید چپ را نگه‌داشته و MOUSE را به سمت پایه‌ای از المان دیگر حرکت دهید. (شکل ۴-۵)



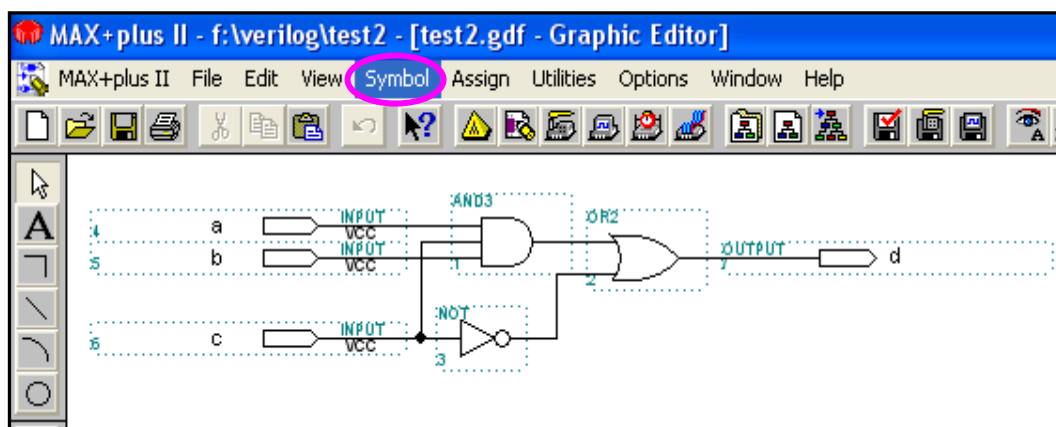
شکل ۴-۵

با کامل شدن مدار ، باید ورودی و خروجی‌های مدار تعیین گردد. برای این منظور با استفاده از OUTPUT, SYMBOL ENTER یا INPUT را در قسمت SYMBOL NAME وارد نموده و به این طریق سیگنال‌های ورودی یا خروجی مدار مشخص می‌شود (شکل ۴-۶)



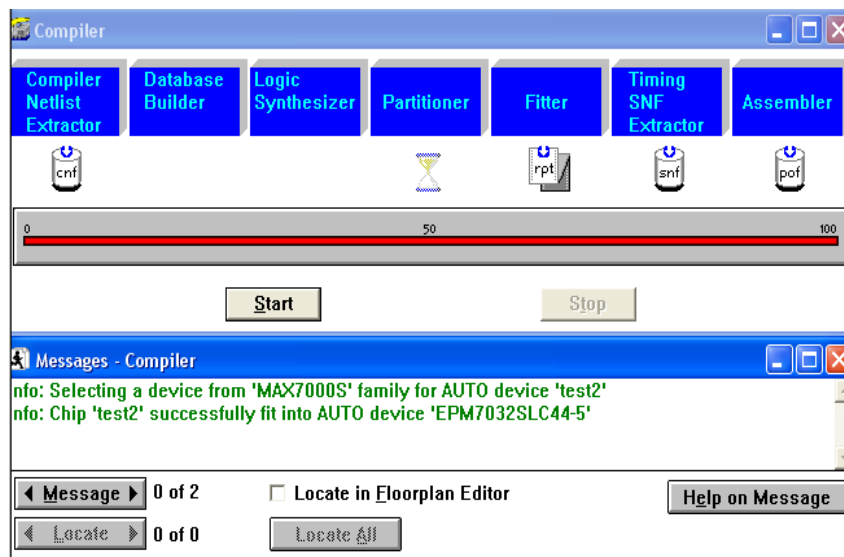
شکل ۴-۶

اکنون شماتیک مدار با قراردادن سیگنال‌های ورودی و خروجی کامل شده است . برای عملیاتی کردن مدار باید نام ورودی‌ها و خروجی‌ها مشخص گردد. بنابراین با کلیک کردن بر روی PIN NAME و قراردادن نام‌های موردنظر مدار شما نهایی خواهد شد. با تکمیل شماتیک، فایل موردنظر با همان نام پروژه ذخیره می‌گردد. شکل ۴-۷ به‌طور واضح ورودی‌ها و خروجی مدار و همچنین نام فایل که با نام پروژه یکسان است را نمایش می‌دهد.



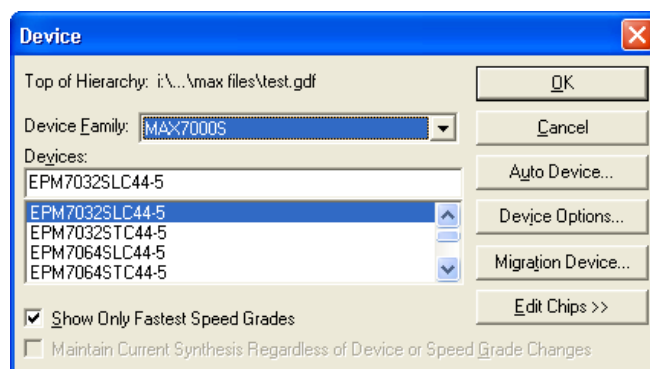
شکل ۴-۷

مرحله بعدی COMPILE نمودن فایل موردنظر است. در این مرحله طراحی از لحاظ ارتباطات، سنکرونیزاسیون (هم‌زمانی) و مناسب بودن با قطعه انتخابی آزمودن شده و نتیجه در یک فایل RPT وارد خواهد گردید. همه موارد فوق بر روی پنجره COMPILE موجود می‌باشد. شکل ۴-۸ قسمت‌های مختلف COMPILE را نمایش می‌دهد. پنجره COMPILE شامل قسمت‌هایی نظیر بررسی ارتباطات، بررسی ورودی‌ها و خروجی‌ها، بررسی سلول‌های منطقی، گزارش، زمان‌بندی و کلیدی برای برنامه‌ریزی IC است.



شکل ۴-۸

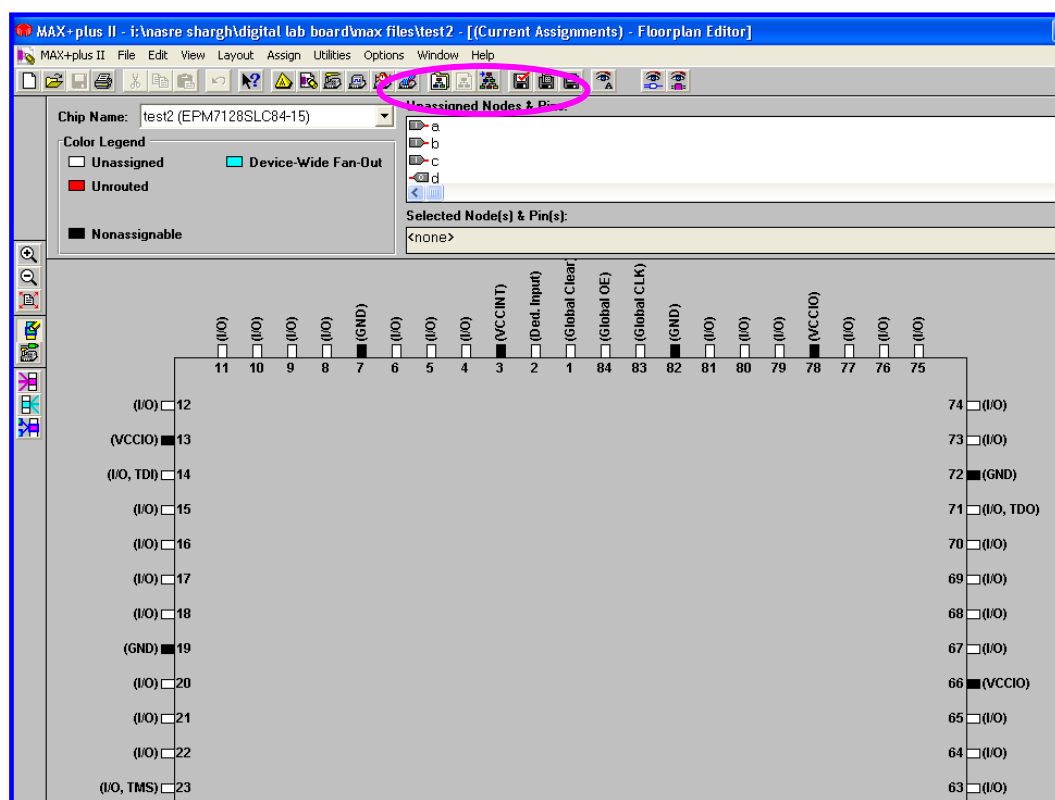
اگر کلیه مراحل COMPILER مدار بدون هیچ خطایی به پایان برسد آنگاه مدار شما از لحاظ منطقی بی نقص می باشد. اکنون باید قطعه ای که برای طراحی در نظر گرفته اید (یا به عبارتی همان قطعه ای که بر روی برد آزمایشگاه قرار دارد) را به عنوان قطعه FPGA مورد نظر، انتخاب کنید. این کار با استفاده از گزینه DEVICE در منوی ASSIGN در بالای صفحه صورت خواهد گرفت (شکل ۴-۹). همچنان که در فصل اول ذکر شد ALTERA قطعات مختلف و متنوعی دارد. برای انتخاب نوع قطعه باید شماره آن را به طور دقیق مشخص نمود که این مطلب در شکل ۴-۹ مشخص شده است.



شکل ۴-۹

با انتخاب نوع قطعه دوباره پروژه مورد نظر را COMPILER نمایید. این بار ارزیابی مدار شما بنا به قطعه مورد نظر شما انجام خواهد گرفت. اکنون باید ورودی و خروجی های مدار را به پایه ها نسبت داد. شکل ۴-۱۰ به طور واضح شکل IC، ورودی و خروجی ها و نوع قطعه را مشخص می نماید. با توجه به مدار چاپی برد آزمایشگاه هریک از پایه های ورودی و خروجی قطعه به یکی از قطعات موجود بر روی برد نظیر LED ها، سوئیچ های انتخاب^۱، IC مولد پالس ساعت و ... متصل می باشد. بنابراین با داشتن شماتیک برد می توانید پایه های قطعه را به ورودی و خروجی های مدار نسبت داد. همان طور که در شکل نیز مشاهده می نمایید در قسمت Unassigned Nodes & Pins ورودی و خروجی هایی که می بایست به پایه های قطعه نسبت داده شوند، مشخص می باشند.

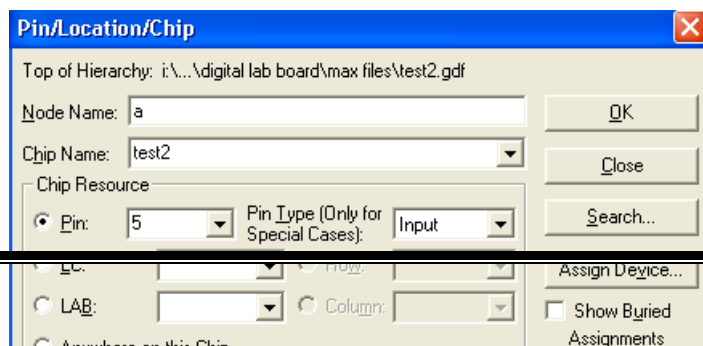
1- Switch selector



شکل ۴-۱۰

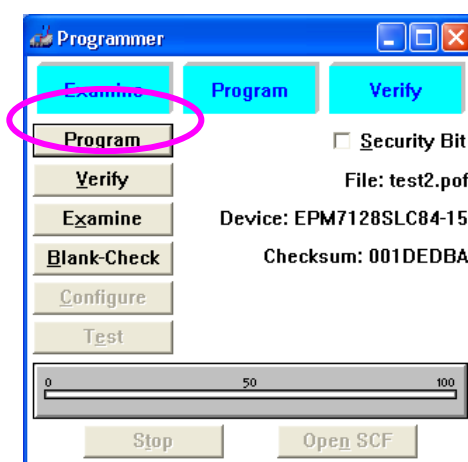
دو روش برای نسبت دادن ورودی و خروجی‌های مدار به پایه‌های قطعه وجود دارد. روش اول یک روش آموزشی و به تعبیری مخصوص تازه‌کاران است و روش دوم بسیار سریع و آسان می‌باشد. در روش اول باید وارد منوی PIN / LOCATION/ CHIP در قسمت ASSIGN شوید (شکل ۴-۱۱). در این منو در قسمت NODE NAME اسم ورودی و خروجی‌ها را وارد نمایید. در قسمت CHIP RESOURCE با وارد نمودن شماره پین و نوع ورودی و خروجی گزینه ADD روشن می‌شود. با کلیک روی گزینه ADD این ورودی به پایه موردنظر نسبت داده می‌شود.

اما در روش دوم اشاره‌گر MOUSE را بر روی هریک از ورودی یا خروجی‌های قطعه قرارداداده، کلید سمت چپ MOUSE را بر روی آن ورودی نگه‌داشته و به سمت پایه موردنظر بکشید به این طریق شما به راحتی یک ورودی یا خروجی را به یک پایه نسبت خواهید داد.



شکل ۴-۱۱

بعد از مشخص نمودن همه ورودی‌ها و خروجی‌ها دوباره مدار خود را COMPILE نمایید. اکنون مدار شما آماده برنامه‌ریزی بر روی IC موردنظر است. برای این امر نیاز به یک سخت‌افزار برنامه‌ریزی تحت عنوان PROGRAMMER JTAG دارید. این سخت‌افزار با استفاده از پورت موازی (LPT1) به برد موردنظر متصل می‌شود. انجام عمل برنامه‌ریزی با استفاده از پنجره ASSEMBLER در منوی COMPILE صورت خواهد گرفت. شکل ۴-۱۲ پنجره PROGRAMMER را نمایش می‌دهد. اگر در پروژه موردنظر خود پنجره را بازنمایید POF نشان داده شده مربوط به آن پروژه می‌باشد، در غیر این صورت باید آدرس POF خود را با استفاده از JTAG که در بالای صفحه می‌باشد مشخص نمایید.

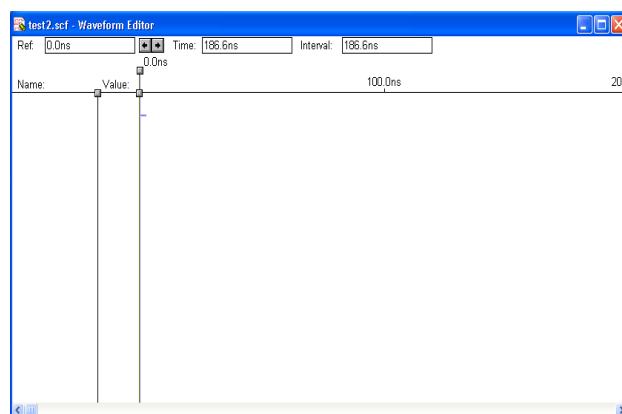


شکل ۴-۱۲

در قسمت PROGRAMMER ۴ گزینه مختلف وجود دارد که به ترتیب عبارت‌اند از:

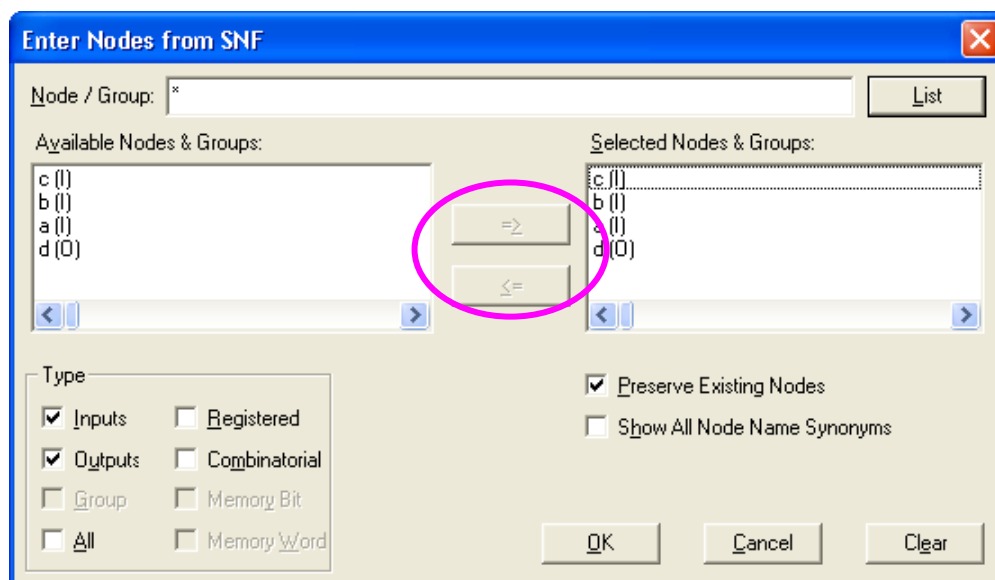
- ۱- PROGRAMMER : برای ارسال فایل POF به IC از طریق JTAG مورد استفاده قرار می‌گیرد. بعد از ارسال فایل بر روی FPGA، قطعه موردنظر برنامه‌ریزی خواهد شد.
- ۲- VERIFY: برای آزمودن و آزمایش صحت فایل ارسالی بر روی قطعه مورد استفاده قرار می‌گیرد.
- ۳- EXAMINE : برای خواندن فایل POF از روی قطعه مورد استفاده قرار می‌گیرد.
- ۴- BLANK CHECK : برای پاک کردن فایل از روی FPGA مورد استفاده قرار می‌گیرد.

این شرح مختصر به شما نحوه طراحی مدار، COMPILE نمودن آن، مشخص کردن IC، تخصیص پایه‌ها و برنامه‌ریزی FPGA را آموزش داده است. آخرین قابلیت این نرم‌افزار که می‌بایست به آن پردازیم، مبحث تحلیل کردن^۱ فایل قبل از برنامه‌ریزی می‌باشد. برای انجام عمل تحلیل نرم‌افزاری باید فایل SCF یا WDF (WAVE FORM EDITOR FILE) از طریق گزینه NEW در منوی FILE بازگردد. این کار در یک پروژه بعد از COMPILE صحیح صورت می‌گیرد. شکل ۴-۱۳ پنجره مربوط به تحلیل را به شما نمایش می‌دهد.

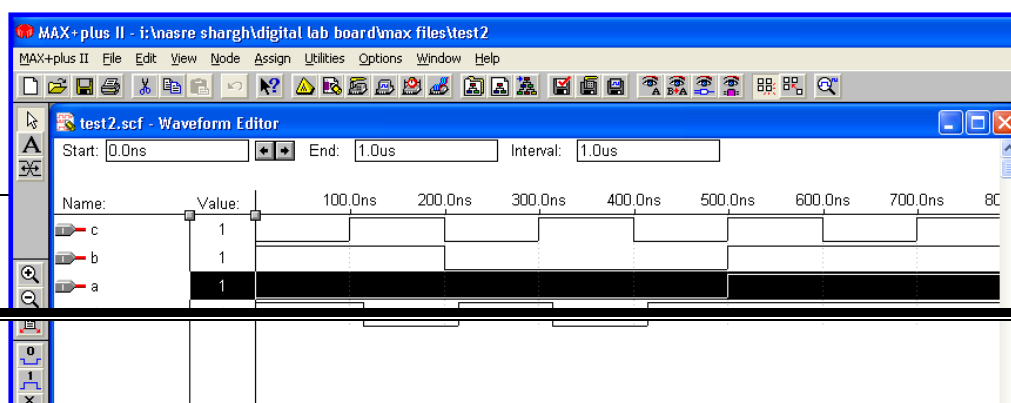


شکل ۴-۱۳

ابتدا با استفاده از گزینه NODE در بالای صفحه، پنجره ENTER NODE FROM SNF را بازنمایید. با فشردن گزینه LIST کلیه ورودی و خروجی‌ها موجود در مدار مشخص و در قسمت Available Nodes & Groups وارد می‌شوند. سپس شما می‌توانید ورودی و خروجی‌های موردنظر خود را انتخاب نمایید و به قسمت Selected Nodes & Groups منتقل نمایید (شکل ۴-۱۴). همچنان که در شکل ۴-۱۵ مشاهده می‌فرمایید پس از انجام این مرحله، ورودی و خروجی‌های موردنظر در سمت چپ پنجره تحلیل وارد خواهند شد.



شکل ۴-۱۴



شکل ۴-۱۵

تحلیل نرم‌افزاری کمک بسیار زیادی به تعیین میزان صحت عملکرد مدار قبل از پیاده‌سازی آن می‌نماید. برای مقدار دادن به ورودی‌ها با قراردادن اشاره‌گر MOUSE بر روی هر ورودی و کلیک کردن بر روی آن کلیدهای کنار صفحه روشن خواهد شد. این کلیدها عبارت‌اند از :

برای انتقال صفر روی ورودی	←	0
برای انتقال یک‌روی ورودی	←	1
برای انتقال X روی ورودی	←	X
برای انتقال Z روی ورودی	←	Z
برای معکوس کردن مقدار ورودی مورد استفاده قرار می‌گیرد	←	INV
برای ایجاد پالس‌های پریودیک مورد استفاده قرار می‌گیرد	←	X ₀
برای ارسال مقادیر به‌صورت پریودیک بر روی باس مورد استفاده قرار می‌گیرد	←	X _C
برای تخصیص خاص به یک قسمت از باس مورد استفاده قرار خواهد گرفت.	←	X _G
	←	X _S

لازم به ذکر است که در مدارهای منطقی علاوه بر منطق صفر و یک دو منطق دیگر نیز وجود دارد. منطق نامشخص با (X) و منطق امیدانس بالا با (Z) معرفی می‌شود.

جلسه ۵

آزمایش جمع کننده‌ها (ریپل کری و پیش‌بینی کننده کری)

در این جلسه با دو جمع کننده آشنا می‌شویم:

Ripple Carry-۱

هدف :

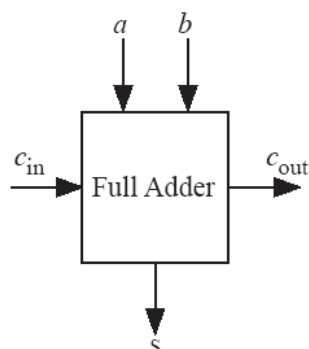
در این آزمایش اهداف زیر دنبال می‌شوند :

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

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 از آنجایی که 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 آزمون 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 طراحی شده (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 قابل مشاهده می‌باشد، لذا به کارگیری 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.



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

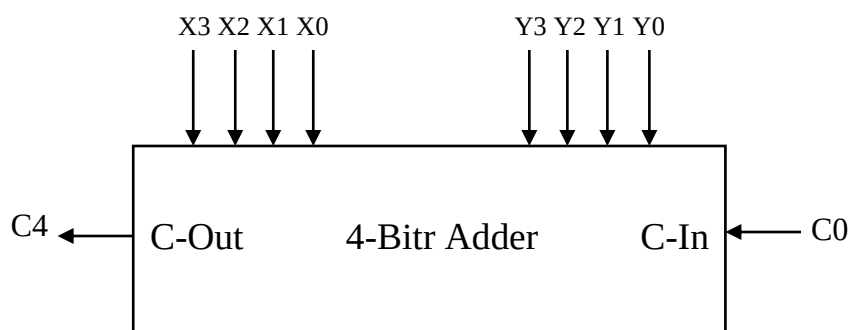
با هم جمع و حاصل را به همراه بیت نقلی تولید می‌نماید. مدار تمام-جمع کننده یا Full-Adder

تک‌بیتی نیز مداری است که دو عدد تک‌بیتی را با بیت نقلی ورودی جمع و مجدداً حاصل را

شکل ۵-۱: تمام-جمع کننده تک‌بیتی

به همراه بیت نقلی تولید می‌نماید (شکل ۵-۱).

تمام-جمع کننده را می توان به عنوان بلوک پایه در طراحی جمع کننده های n-بیتی مورد استفاده قرار داد. به عنوان مثال در طراحی جمع کننده ۴-بیتی از ۴ تمام-جمع کننده استفاده می شود. 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 می شود.



شکل ۵-۲: جمع کننده ۴-بیتی

تراشه یا آی سی هایی که دانشجویان عزیز در این آزمایش مورد استفاده قرار خواهند داد، عبارتند از:

۱. NOT: که با شماره ۷۴۰۴ یا نام NOT (NOT با دو ورودی) در کتابخانه نرم افزار قابل دسترسی می باشد.
۲. AND: که با شماره ۷۴۰۸ یا نام AND2 (AND با دو ورودی) در کتابخانه نرم افزار قابل دسترسی می باشد.
۳. OR: که با شماره ۷۴۳۲ یا نام OR2 (OR با دو ورودی) در کتابخانه نرم افزار قابل دسترسی می باشد.
۴. XOR: که با شماره ۷۴۸۶ یا نام XOR2 (XOR با دو ورودی) در کتابخانه نرم افزار قابل دسترسی می باشد.

تکالیف پیش از آزمایش :

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

(۱) مراحل پیاده سازی نرم افزاری و سخت افزاری ارائه شده در فصول ابتدایی را به دقت مطالعه و در صورت لزوم به مستندات نرم افزار مراجعه نمایید.

(۲) نیم-جمع کننده :

(۱-۲) مدار مورد نظر را به طور اجمالی تشریح و عملکرد آن را توضیح دهید.

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

(۳-۲) با استفاده از گیت های NOT، OR و AND مدار یک نیم-جمع کننده را طراحی نموده و توسط نرم افزار MAX، شماتیک آن را ترسیم و خروجی آن را پس از شبیه سازی با تئوری مقایسه نمایید. آیا جواب به دست آمده درست است ؟

(۴-۲) مدار ترسیم شده را با توجه به نمایش خروجی ها بر روی LED ها و به منظور پیاده سازی 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 درست‌نمایی 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؟

۳-۴) مدار ترسیم‌شده 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 درست‌نمایی 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 ترسیم شده مدار جمع کننده ۴ بیتی را توسط نرم افزار موردنظر اجرا و مقدمات پیاده سازی بر روی برد را مهیا نمایید.

۳-۲) مراحل پیاده سازی 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 آورده اید، مقایسه نمایید.

پروژه پیشنهادی :

با استفاده از مدار جمع کننده ۴ بیتی (به عنوان یک بلوک طراحی) یک جمع کننده ۱۶ بیتی طراحی نمایید. ورودی ها، خروجی ها، رقم نقلی خروجی و ارتباط داخلی مابین تمام جمع کننده ها را مشخص نمایید. مدار جمع کننده ۴ بیتی طراحی شده را به عنوان یک Symbole تعریف و با استفاده از این قابلیت شماتیک جمع کننده ۱۶ بیتی را توسط نرم افزار ترسیم و خروجی حاصل از شبیه سازی را با تئوری مقایسه نمایید. در نهایت شماتیک طراحی شده را بر روی برد پیاده سازی نمایید و نتایج حاصل را بر روی نمایشگر ۷-قسمتی مشاهده کنید.

۲-Carry – Look – Ahead Adder

هدف :

در این آزمایش اهداف زیر دنبال می شوند:

- ✓ آشنایی با تأخیر در مدارات دیجیتال
- ✓ فراگیری استفاده از Timing Simulator.
- ✓ طراحی جمع کننده Carry Look Ahead ۴ بیتی و محاسبه تأخیر آن.
- ✓ پیاده سازی و آزمودن مدار طراحی شده

تئوری آزمایش :

در آزمایش گذشته به بررسی نحوه عملکرد مدارهای منطقی ترکیبی پرداختید و به منظور ارزیابی عملکرد مدار طراحی شده از نرم افزار تحلیل گر استفاده نمودید. همان گونه که می دانید در طراحی علاوه بر نحوه عملکرد مدار پارامترهای دیگری نیز در طراحی مدارات دیجیتال وجود دارند که می بایست به آنها نیز توجه کرد به عنوان مثال سرعت مدارهای منطقی در طراحی از درجه اهمیت بالایی برخوردار هستند.

اکنون ممکن است سؤالی در ذهن شما شکل گرفته باشد که عامل تأثیرگذار بر سرعت مدارهای منطقی چیست؟ جواب تأخیر قطعات می باشد. هر گیت NOT، OR، AND و ... مقدار تأخیری برابر τ خواهد داشت. یعنی با قراردادن متغیرهای ورودی، بعد از گذشت زمانی برابر τ ، خروجی مقدار موردنظر خود را خواهد داشت. البته هر یک از گیت های مذکور مقدار تأخیر متفاوتی خواهند داشت. در عمل بایستی مدار را به صورتی 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 می‌بیند (Fan-Out) و عامل دیگر نحوه طراحی مدار منطقی می‌باشد.

دو عامل اول به ساختار قطعات برمی‌گردد، اما عامل سوم مستقیماً^۱ 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. به‌عنوان مثال 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 قابل توجهی 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. به‌طور کلی سرعت مدار از طریق مقایسه سیگنال ورودی و سیگنال خروجی توسط اسیلوسکوپ قابل محاسبه می‌باشد. با این وجود در خلال طراحی و در زمانی که مدار هنوز ساخته نشده است، محاسبه تأخیر کاری مشکل و غیرممکن به نظر می‌رسد. در این حالت با استفاده از قابلیت Timing Simulation نرم‌افزارهای تحلیل گر می‌توان در خلال طراحی نیز تأخیر مدار را محاسبه نمود. البته این نکته نیز غیرقابل انکار نیست که تأخیر واقعی مدار کاملاً^۲ 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 راه کار 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 با ارزش‌ترین ۲ بیت، زمانی قابل دستیابی می‌باشد که رقم نقلی از کم‌ارزش‌ترین مرحله به با ارزش‌ترین مرحله^۱ منتقل شود. به‌سادگی مطلب فوق در مثال زیر قابل مشاهده می‌باشد.

$$\begin{array}{r}
 1 \quad 1 \quad 1 \quad 1 \\
 + \quad 1 \quad 1 \quad 1 \quad 1 \\
 \hline
 \quad 0 \quad 0 \quad 0 \quad 1 \\
 \hline
 1 \quad 0 \quad 0 \quad 0 \quad 0
 \end{array}$$

به‌سادگی می‌توانید محاسبه نمایید که در این روش رقم نقلی خروجی در جمع کننده N بیتی پس از $2N+2$ تأخیر گیت، قابل دستیابی می‌باشد و حاصل جمع با ارزش‌ترین ۲ بیت (S_{N-1}) نیز پس از $2N+2$ تأخیر گیت قابل دستیابی 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. به‌عنوان مثال 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^۱

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 می شود که عبارتند از: (۱) A_i, B_i هر دو یک باشند (۲) زمانی که یکی از ۲ بیت ورودی یک و رقم نقلی ورودی یک باشد. بنابراین می توان نوشت که:

$$C_{out} = C_{i+1} = A_i.B_i + (A_i \oplus B_i).C_i \quad (1)$$

در این عبارت " " بیانگر AND و $\oplus$ بیانگر XOR می باشد. این رابطه را می توان خلاصه تر نیز نوشت.

$$C_{out} = C_{i+1} = G_i + P_i.C_i \quad (2)$$

$$G_i = (A_i.B_i) \quad (3)$$

$$P_i = (A_i \oplus B_i) \quad (4)$$

که G_i بیانگر تولید رقم نقلی و P_i بیانگر انتشار رقم نقلی می باشد. همان طور که اشاره شد ساختار کلی CLA بر مبنای روابط فوق استوار می باشد. اما نکته قابل توجه در این جمع کننده میزان تأخیر آن است. حال به محاسبه این تأخیر می پردازیم. 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 می باشند و لذا به ترتیب پس از ۲ و ۱ واحد تأخیر قابل دستیابی هستند. لذا اگر از تعاریف فوق به منظور محاسبه رقم نقلی استفاده شود، دیگر نیازی نیست که برای تولید رقم نقلی منتظر انتقال رقم نقلی از مرحله های پایین تر باشیم. حال این تعریف را به جمع کننده ۴ بیتی تعمیم می دهیم:

$$C_1 = G_0 + P_0.C_0 \quad (5)$$

$$C_2 = G_1 + P_1.C_1 = G_1 + P_1.G_0 + P_1.P_0.C_0 \quad (6)$$

$$C_3 = G_2 + P_2.C_2 = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.C_0 \quad (7)$$

$$C_4 = G_3 + P_3.C_3 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0 + P_3.P_2.P_1.P_0.C_0 \quad (8)$$

مشاهده می نمایید 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 گیت های AND, OR) قابل دستیابی می باشد. حاصل جمع سیگنال نیز از فرمول زیر قابل محاسبه می باشد:

$$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i \quad (9)$$

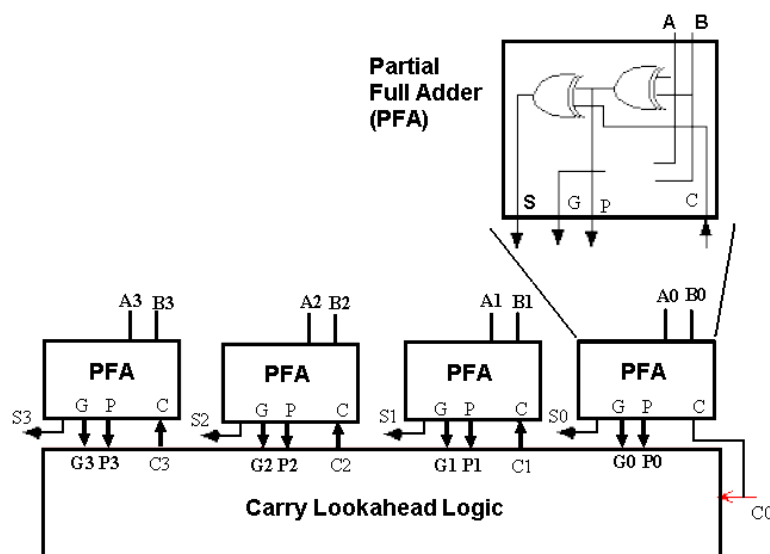
نتیجه خروجی نیز پس از ۲ واحد تأخیر دیگر یا به عبارت دیگر مجموعاً پس از ۶ واحد تأخیر پس از اعمال A_i, B_i به جمع کننده قابل دستیابی می باشد. از مزایای این روش این است که تأخیر حاصل مستقل از تعداد بیت های A و B می باشد.

با توجه به مطالبی که در مورد جمع کننده Carry Look Ahead بیان شد، می توان 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 فرمول های ۳، ۴ و ۹ تعریف شده اند را تولید می نماید،

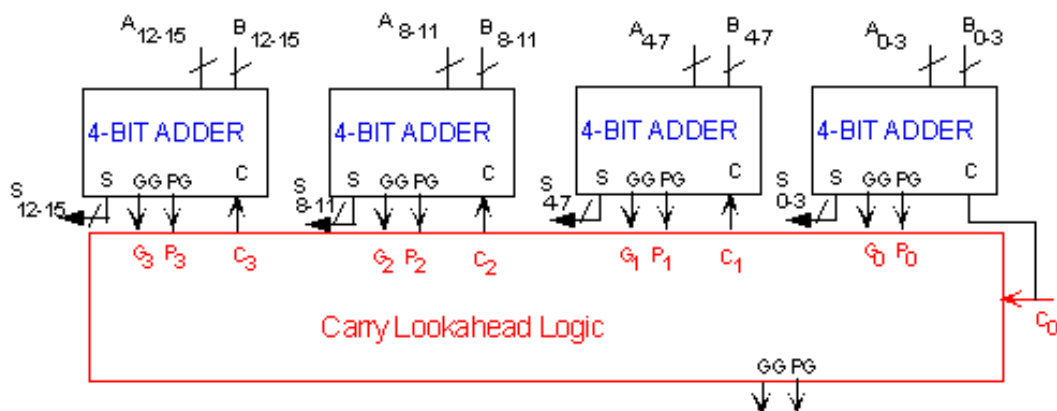
(۲) مدار منطقی Carry Look Ahead، که رقم نقلی خروجی را که مبتنی بر روابط ۵ تا ۸ می باشند، تولید می نماید.

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 ۵-۳: جمع کننده ۴ بیتی Carry Look Ahead

از معایب این جمع کننده می‌توان به پیچیدگی مدار منطقی بلوک Carry Look Ahead برای جمع کننده‌های بابت‌های زیاد (بیش از ۴ بیت) اشاره نموده البته این مشکل نیز با استفاده از طراحی‌های سلسله مراتبی قابل حل می‌باشد. در این روش بلوک جمع کننده ۴ بیتی CLA به‌عنوان بلوک اصلی بوده و طراحی بر اساس آن صورت می‌گیرد. (شکل ۵-۴)



شکل ۵-۴: 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 جمع‌کننده 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 جمع‌کننده 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 Simulation 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 جمع‌کننده 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 جمع‌کننده ۴ بیتی CLA را توسط نرم‌افزار موردنظر اجرا و مقدمات پیاده‌سازی بر روی برد را مهیا نمایید.

۲-۳) 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 جمع‌کننده ۴ بیتی CLA ای که طراحی نمودید (به‌عنوان یک بلوک طراحی) یک جمع‌کننده ۱۶ بیتی CLA طراحی نمایید. ورودی‌ها، خروجی‌ها، رقم نقلی خروجی و ارتباط داخلی مابین تمام جمع‌کننده‌ها را مشخص نمایید. مدار جمع‌کننده ۴ بیتی CLA طراحی‌شده را به‌عنوان یک Symbole تعریف و با استفاده از این قابلیت شماتیک جمع‌کننده 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 ۶

آزمایش دیاگرام حالت و مقسم فرکانس

۱- دیاگرام حالت

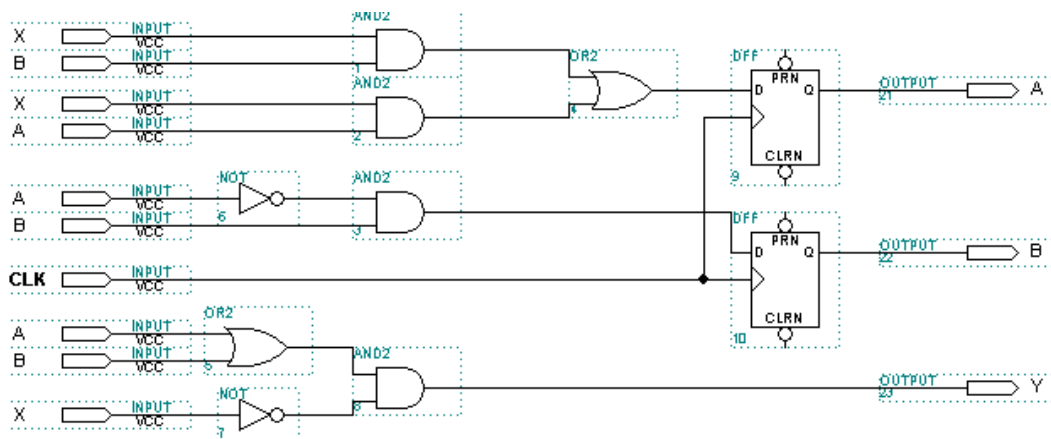
هدف :

در این آزمایش هدف زیر دنبال می‌شود:

✓ کسب مهارت بیشتر در زمینه طراحی مدارات ترتیبی

تئوری آزمایش :

ترتیب زمانی ورودی و خروجی‌ها و حالات فلیپ فلاپها را می‌توان در جدول حالات برشمرد. مدار شکل ۱-۶ که در کتاب مانو آمده است دارای جدول حالتی به صورت جدول ۱-۶ می‌باشد.



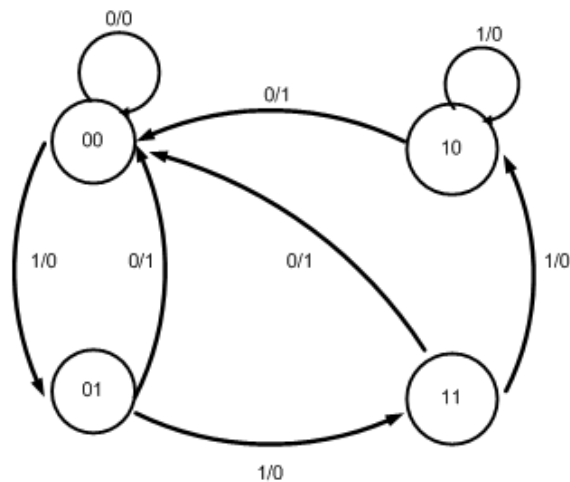
شکل ۱-۶

خروجی	حالت	ورودی	حالت
-------	------	-------	------

فعلی		بعدی	
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 ۱-۶: جدول حالات مدار شکل ۱-۶

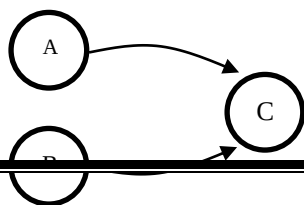
همچنان که از آزمایش شمارنده به یاد دارید، با داشتن جدول حالات یک مدار یعنی داشتن کلیه حالات آن می‌توان به آسانی مدار ترتیبی مربوطه را طراحی نمود. شکل ۱-۶ درواقع با استفاده از جدول ۱-۶ طراحی شده است. اطلاعات موجود در یک جدول حالت را می‌توان به صورت گرافیکی تحت عنوان دیاگرام حالت نمایش داد. شکل ۲-۶ دیاگرام حالت جدول ۱-۶ می‌باشد.



شکل ۲-۶: دیاگرام حالت جدول ۱-۶

با توجه به مقدماتی که بیان شد و مطالبی که از آزمایش شمارنده نیز به یاد دارید، دریافتیم که به آسانی 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 به صورت مسئله باید شما دیاگرام حالات را رسم نمایید.

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

۴- ساده سازی جدول حالت در صورت امکان

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

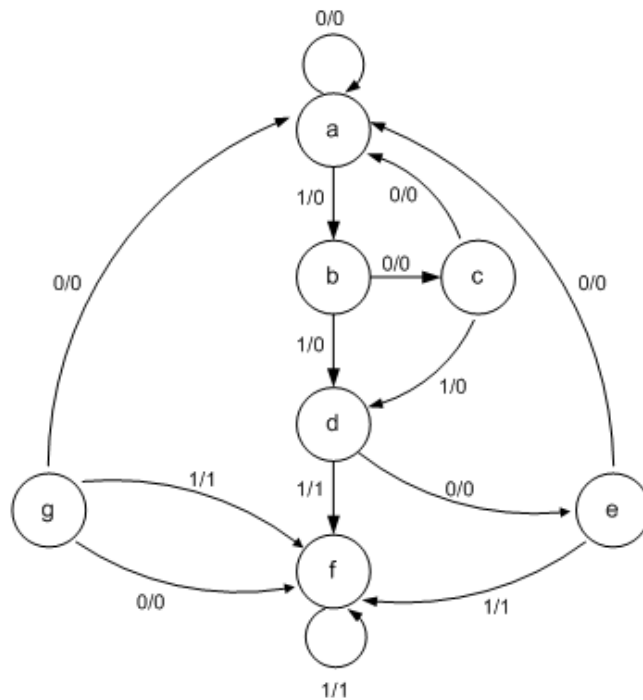
۵- با توجه به حالت های باقیمانده تعداد فلیپ فلاپها مشخص می شود.

۶- تخصیص حالات

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

۸- ساده کردن توابع حاصل و رسم مدار .

نکته مهم در طراحی مدارات ترتیبی (به غیر از مدارات شمارنده) ساده سازی جدول حالت است. اکنون مدار مربوط به دیاگرام حالتی که در شکل ۶-۳ آمده است را با به کارگیری مطالب فوق طراحی می نماییم.



شکل ۶-۳ : دیاگرام حالات

حالت بعدی

خروجی

	X=0	X=1	X=0	X=1
<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>
<i>b</i>	<i>c</i>	<i>d</i>	<i>c</i>	<i>d</i>
<i>c</i>	<i>a</i>	<i>d</i>	<i>a</i>	<i>d</i>
<i>d</i>	<i>e</i>	<i>f</i>	<i>e</i>	<i>f</i>
<i>e</i>	<i>a</i>	<i>f</i>	<i>a</i>	<i>f</i>
<i>f</i>	<i>g</i>	<i>f</i>	<i>g</i>	<i>f</i>
<i>g</i>	<i>a</i>	<i>f</i>	<i>a</i>	<i>f</i>

جدول ۲-۶ جدول حالات شکل ۳-۶

با توجه به جدول ۲-۶ حالت‌های *g* و *e* دارای خروجی‌ها و حالت‌های یکسان هستند بنابراین *g* حذف شده و بجای *g*، *e* قرار می‌دهیم.

	X=0	X=1	X=0	X=1
<i>a</i>	<i>a</i>	<i>b</i>	0	0
<i>b</i>	<i>c</i>	<i>d</i>	0	0
<i>c</i>	<i>a</i>	<i>d</i>	0	0
<i>d</i>	<i>e</i>	<i>f</i>	0	1
<i>e</i>	<i>a</i>	<i>f</i>	0	1
<i>f</i>	<i>e</i>	<i>f</i>	0	1

جدول ۳-۶ : جدول حالت با حذف حالت *g*

در جدول ۳-۶ نیز *d* و *f* معادل هستند. با جایگزین کردن *d* به جای *f*، یک حالت دیگر نیز از سیستم کم خواهد شد.

	X=0	X=1	X=0	X=1
<i>a</i>	<i>a</i>	<i>b</i>	0	0
<i>b</i>	<i>c</i>	<i>d</i>	0	0
<i>c</i>	<i>a</i>	<i>d</i>	0	0
<i>d</i>	<i>e</i>	<i>d</i>	0	1
<i>e</i>	<i>a</i>	<i>d</i>	0	1

جدول ۴-۶ : جدول حالت با حذف *f*

مرحله بعدی تخصیص حالات می‌باشد. این کار با استفاده از سه فلیپ فلاپ *A*، *B*، *C* 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 ۵-۶: تخصیص حالت

اکنون جدول حالت فلیپ فلاپ ها را با استفاده از جدول ۵-۶ طراحی نمایید.

تکالیف پیش از آزمایش:

۱- آزمایش اول

۱-۱- مدار شکل ۱-۶ در نرم افزار پیاده سازی نمایید.

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

۲- آزمایش دوم:

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

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

۳-۲- ورودی مدار را به یک سوئیچ، خروجی مدار را به یک LED تک رنگ و حالاتهای مدار را به سه LED دورنگ که یک پایه آن غیرفعال شده 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 به عنوان آزمایش اول توسط نرم افزار طراحی و آماده پیاده سازی نموده اید را بر روی FPGA پیاده نمایید.

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

۲- آزمایش دوم:

۱-۲- مداری را که به عنوان آزمایش دوم توسط نرم افزار طراحی و آماده پیاده سازی نموده اید را بر روی FPGA پیاده نمایید.

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

۲- مقسم فرکانس

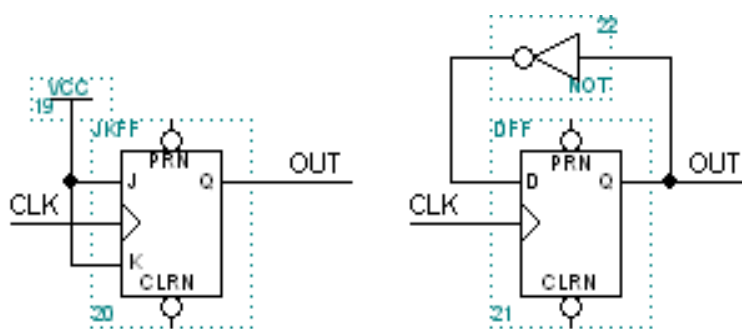
هدف:

در این آزمایش اهداف زیر دنبال می‌شوند:

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

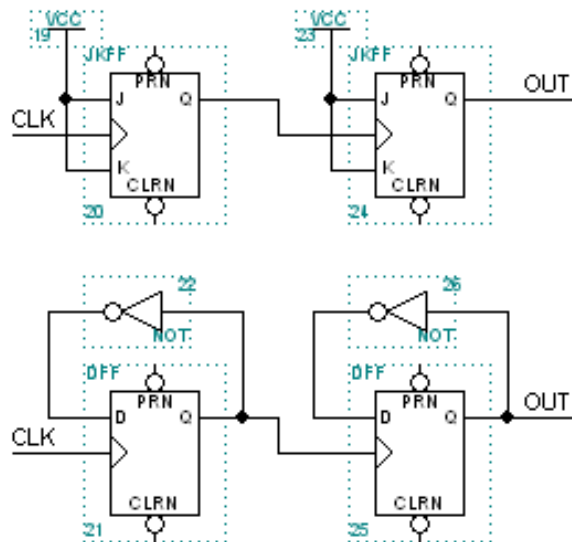
تئوری آزمایش:

مدارهای دیجیتالی که تاکنون در آزمایش‌های مختلف مورد بررسی قرار گرفتند، مدارهای ترکیبی بودند. اما بیشتر سیستم‌های عملی دارای عناصر حافظه‌ای هستند که مدارهای ترتیبی را به وجود می‌آورند. عناصر حافظه‌ای که به عنوان قطعات اصلی در مدارهای ترتیبی با پالس ساعت به کار می‌روند، فلیپ فلاپ نامیده می‌شوند. مدارهای ترتیبی مبتنی بر فلیپ فلاپ‌ها، کاربرد و عملکردهای مختلفی دارند. از جمله 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 به منظور 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 فلیپ فلاپهای نوع D, JK و T استفاده می‌نمایند. ساده‌ترین مقسم فرکانس عبارت است از مقسم دو که به سادگی با یک فلیپ فلاپ نوع D یا JK قابل پیاده‌سازی می‌باشد.



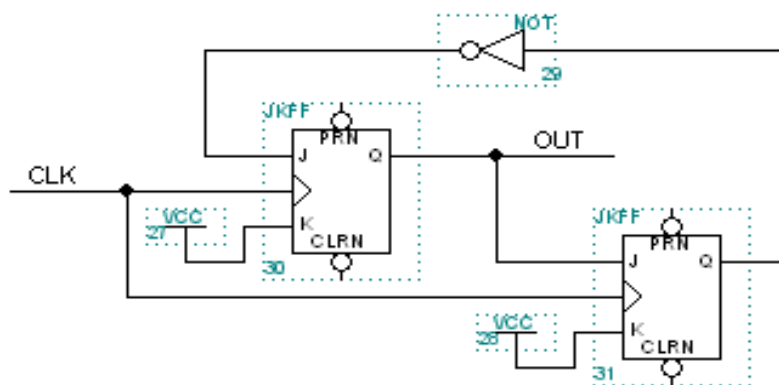
شکل ۴-۶: مقسم ۲ مبتنی بر فلیپ فلاپهای نوع D و JK

به سادگی می‌توان نتیجه گرفت که برای طراحی مقسم فرکانس‌های زوج $2n$ ، n مقسم فرکانس ۲ را باهم به صورت زنجیره‌ای متصل نموده و خروجی طبقه n ام حاصل تقسیم پالس ورودی بر $2n$ را نتیجه می‌دهد. به عنوان مثال شکل زیر مقسم فرکانس ۴ را با استفاده از دو نوع فلیپ - فلاپ نمایش می‌دهد.



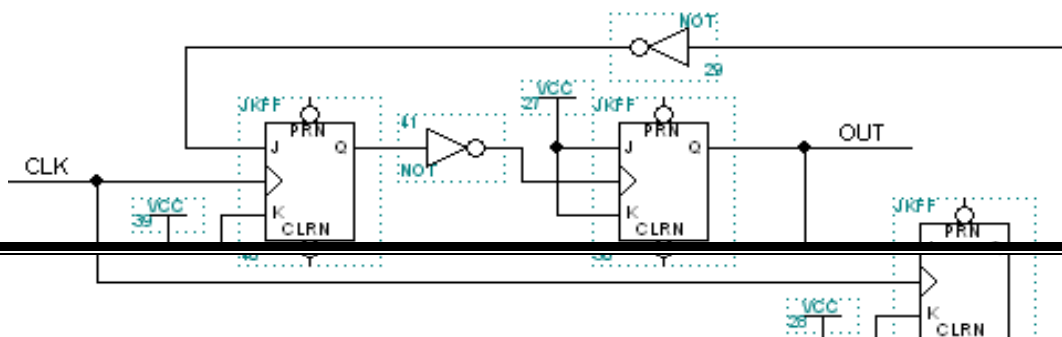
شکل ۵-۶: مقسم فرکانس ۴ مبتنی بر فلیپ های نوع D و JK

اما تقسیم فرکانس بر اعداد فرد کمی پیچیده تر می باشد. به منظور طراحی این مقسم فرکانس ابتدا عدد مورد نظر را تجزیه کرده و تعداد ۲هایی که در آن وجود دارد را به دست می آوریم. به عبارتی عدد مورد نظر را به فرم $2 \times X + 1$ تبدیل می نماییم. به عنوان مثال عدد ۳ عبارت است از $1 \times 2 + 1$ یا عدد ۵ برابر است با $2 \times 2 + 1$. سپس مقسم فرکانس مورد نظر را با استفاده از مدار مقسم فرکانس ۲ و مدارهای جانبی طراحی می نماییم. به طور مثال مقسم ۳ به صورت زیر قابل طراحی است:



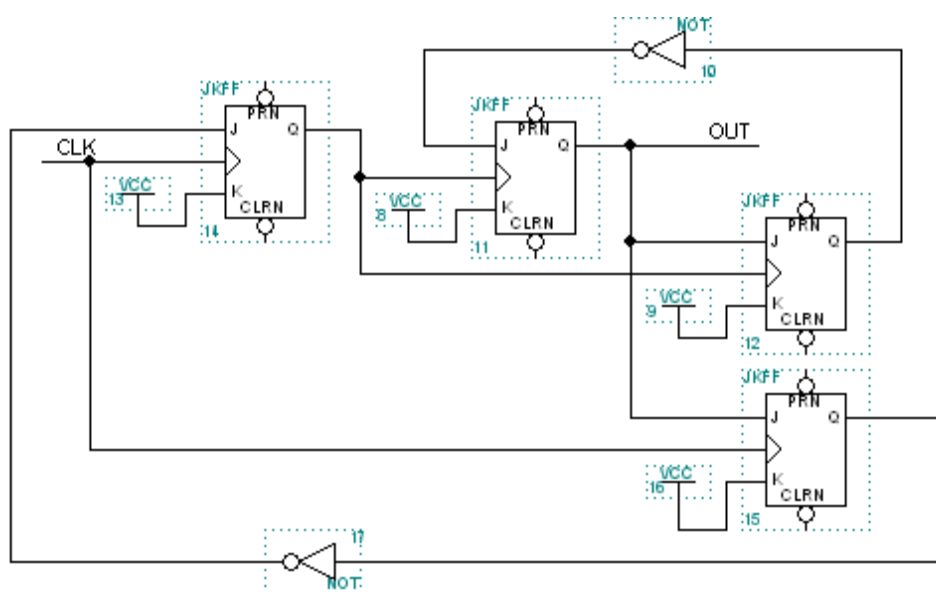
شکل ۶-۶: مقسم فرکانس ۳ مبتنی بر فلیپ فلاپ نوع JK

مثال دیگر عدد ۵ می باشد که با استفاده از دو مدار مقسم فرکانس ۲ و مدار جانبی طراحی مقسم فرکانس صورت می پذیرد:



شکل ۶-۷: مقسم فرکانس ۵ مبتنی بر فلیپ فلاپ نوع JK

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



شکل ۶-۸: مقسم فرکانس ۷ مبتنی بر فلیپ فلاپ نوع JK

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 مورد بررسی قرار می‌دهیم. قطعات ۷۴۷۳، ۷۴۷۶ و ۷۴۷۸ فلیپ-فلاپ‌های نوع JK و قطعه ۷۴۷۴ فلیپ-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 مورد نظر را طراحی و پیاده‌سازی نمایید.

تکالیف پیش از آزمایش :

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

(۱) اعداد ۲، ۳، ۴ و ۵ اعدادی هستند که تجزیه و تحلیل مقسم‌های آن‌ها از اهمیت خاصی برخوردار است و به سادگی 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 طراحی شده را توسط نرم‌افزار مورد نظر ترسیم و خروجی‌های آن را پس از تحلیل در گزارش خود ثبت نمایید. لازم به ذکر است که مدار را به گونه‌ای طراحی نمایید که نتایج خروجی توسط LEDها قابل مشاهده باشد.

(۲) سه مقسم فرکانس ۱۹، ۵۲ و ۱۲۰ را طراحی و توسط نرم‌افزار مورد نظر 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 به گونه‌ای طراحی نمایید که نتایج خروجی توسط نمایشگرهای ۷ قسمتی قابل مشاهده باشد.

(۳) مقسم فرکانس‌های طراحی شده را برای پیاده‌سازی آماده نمایید.

تکالیف داخل آزمایشگاه :

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

(۱) از مولد پالس به عنوان ورودی مدارها و سیگنال مرجع استفاده نمایید. برای قابل رؤیت شدن خروجی باید از فرکانس Clock را در حدود ۸KHz قرار دهید.

(۲) بر روی برد، چهار LED دورنگ وجود دارد از این LEDها به منظور نمایش پالس ورودی و پالس خروجی استفاده نمایید.

حال به پیاده‌سازی مدارهای طراحی شده بر روی برد می‌پردازیم :

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

(۱-۱) مدارهای طراحی شده را بر روی برد پیاده‌سازی نمایید.

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

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

(۲) مقسم فرکانس‌های ۱۳، ۱۹ و ۱۲۰

(۱-۲) 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 فلیپ فلاپ نوع T برای اعداد مضرب ۲ و به‌صورت 2^n ، مضرب ۲ غیر از 2^n و اعداد فرد.

آشنایی با زبان توصیف سخت‌افزاری Verilog و نرم‌افزار Quartus

حال به مرحله‌ای می‌رسیم که FPGA را می‌شناسیم و می‌خواهیم یک مدار Logic را روی آن پیاده نماییم. ابتدا باید تعیین کنیم مداری که می‌خواهد پیاده شود، چه کار می‌کند. برای این کار مثلاً می‌توانیم شماتیک مدار را به صورت مجموعه‌ای از گیت‌های منطقی که به هم وصل شده‌اند و یک مدار با ورودی و خروجی مشخص را ساخته‌اند، رسم کنیم. این ساده‌ترین کاری است که می‌توانیم انجام دهیم. رسم مدار برای یک مدار منطقی بزرگ با این روش به ساعت‌های بسیار زیاد نیاز است. پس بهتر است روش دیگری پیدا شود. می‌توانیم عملکرد مدار را به صورت یک سری جملات توصیفی پشت سر هم بنویسیم. مثلاً فرض کنید برای یک شمارنده که با لبه بالارونده خروجی‌اش یک واحد زیاد می‌شود می‌توان گفت: "در هر لبه بالارونده خروجی مساوی با خروجی بعلاوه ۱". حال بجای مدار شمارنده می‌توانیم این عبارت توصیفی را ذخیره کنیم بدیهی است که وقتی در آینده خودمان خواستیم از روی این عبارت مدار منطقی مربوط به آن را پیاده کنیم می‌دانیم که چه گیت‌هایی معادل این عبارت توصیفی بوده است. چراکه مدار شمارنده یک چیز بسیار متداول و شناخته شده است. نرم‌افزارهایی وجود دارند که عبارت توصیفی را که توصیف‌کننده یک مدار منطقی است و به صورت یک برنامه نوشته شده است به عنوان ورودی می‌گیرند و به ما آن آرایشی از گیت‌ها را که معادل عبارت‌های ما است می‌دهند. این برنامه‌ها بسیار هوشمند هستند و بهترین و پرسرعت‌ترین مدار منطقی ممکن را به ما می‌دهند. به این نرم‌افزارها Synthesizer می‌گویند.

زبان توصیف سخت‌افزار

زبان توصیف سخت‌افزار VHDL

VHDL نتیجه همکاری دو شرکت بزرگ Texas Instruments و IBM همراه با یک شرکت کوچک دیگر است. این زبان توصیف سخت‌افزار به سفارش وزارت دفاع ایالات متحده نوشته شد، به این منظور که وزارت دفاع بتواند طرح ICهای پرسرعت و بزرگ خود را به یک روش مطمئن و انعطاف‌پذیر بایگانی کند. VHDL مخفف عبارت زیر است:

Very high speed integrated circuit Hardware Description Language

بعداً استفاده از این زبان برای کارهای تجاری طراحی تراشه‌های منطقی متداول شد و برنامه‌های شبیه‌ساز و سنتز کننده مربوط به آن به بازار عرضه شدند. هم‌اکنون این نرم‌افزارها در بازار وجود دارند و معمولاً گران‌قیمت هستند.

زبان توصیف سخت‌افزار Verilog

Verilog زبان توصیف سخت‌افزاری است که به‌صورت موازی با VHDL به وجود آمد. شرکت خاصی تولیدکننده آن نیست اما شرکت Cadence نقش 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 هر ساله 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 و A، 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 هر کدام از آن‌ها می‌رسیم به خودمدار Logic ای که ارتباط بین ورودی و خروجی را به وجود می‌آورد:

در دستور assign اول گفته‌شده که $S = A \oplus B \oplus Cin$ 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 endmodule اعلام 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 schematic هنگامی 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 گفته شده که ترکیب {C,S} به عنوان عدد دوبیتی برابر است با حاصل جمع A,B و Cin.

این برنامه وقتی به Synthesizer داده 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 Modeling که در آن با عبارت‌هایی عملکرد مدار را توصیف می‌کنیم. لازم به ذکر است که در روش دوم در مورد اینکه چه گیت‌هایی لازم است تا این عملکرد ایجاد شود حرفی نمی‌زنیم و آن را به عهده Synthesizer می‌گذاریم. از اینجا نقش و اهمیت یک Synthesizer آشکار می‌شود. (قیمت این برنامه‌ها معمولاً گران و بین 15 تا 500 هزار دلار است). علاوه بر Behavioral و Gate Level ، Structural Modeling از Verilog 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 نوشته شده است) استفاده شده است. Full adder اول 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 به وسیله 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 به صورت 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 فلیپ فلاپ D ساده را نمایش می‌دهد. دقت می‌کنیم که خروجی Q علاوه بر خروجی، به صورت reg هم تعریف شده است. این به ابزار سنتز می‌گوید که برای Q یک فلیپ فلاپ در نظر بگیرد. دو خط اصلی برنامه با یک دستور always شروع می‌شود. این دو خط می‌گوید: همواره، در هر لبه بالارونده مقدار D را به Q منتقل نماید. حالا فرض کنید می‌خواستیم اولاً، این کار در لبه‌های پایین‌رونده انجام شود، ثانیاً مقدار not شده D به Q منتقل 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 به جای posedge از negedge استفاده می‌کنیم. ثانیاً یک علامت not یعنی "~" جلوی D اضافه شده است. به این مفهوم که مقدار not شده D را دریافت خواهد کرد. حالا فرض کنید می‌خواستیم یک ورودی reset هم برای فلیپ فلاپ بگذاریم که در هر لبه بالارونده‌ای که مقدار آن 1 شد، خروجی فلیپ 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 همانطور که بوده مهم‌اند).

```
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 اجرا شدن (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 case در ورودی mux_control نباشد (مثلاً mux_control برابر با 2'bzz امپدانس بالا باشد). آن وقت 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'hfff )
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 reset دوباره از آدرس صفر شروع می‌شود، وقتی که 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 انتخاب شده 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 انجام شده است. هر زمان (read not write) rnw برابر با ۱ باشد، یعنی قرار باشد از ram بخوانیم، داده روی پورت data 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 هنگامی که قرار است از این پورت داده به بیرون برود، مقدار اختصاص داده شده است که این مقدار برابر با صفر است. دقت کنید که هنگامی که data در حالت خواندن است دستور assign مقدار 16'bz را که درواقع امپدانس بالا و مدار قطع است روی data قرار می‌دهد. به عبارت دیگر 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، هنگامی که در سطح رفتاری برنامه‌نویسی می‌کنیم، ممکن است مداری طراحی نماییم که معادل سخت‌افزاری ندارد. این گونه طرح‌های سخت‌افزاری فقط در شبیه‌سازهای HDL قابل اجرا 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 هر کدام 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 هر کدام 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 آزمون تولید می‌کند. به این ترتیب عملکرد کلی این است که با استفاده از یکی از نرم‌افزارهای شبیه‌سازی Verilog مجموعه ماژول اصلی و ماژول آزمون آن، را که به هم وصل شده‌اند شبیه‌سازی می‌کنیم و می‌بینیم که آیا ماژول اصلی درست کار 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. به این ترتیب 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 عناصر را سر جای مناسب روی FPGA قرار می‌دهد (Placement) و بین آن‌ها را سیم‌کشی می‌کند (Routing) در نهایت یک فایل با پسوند BIT تولید می‌شود که ما می‌توانیم از آن برای Program کردن Rom ای 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 نرم‌افزار Quartus استفاده می‌کنیم.

آشنایی با نرم افزار Quartus

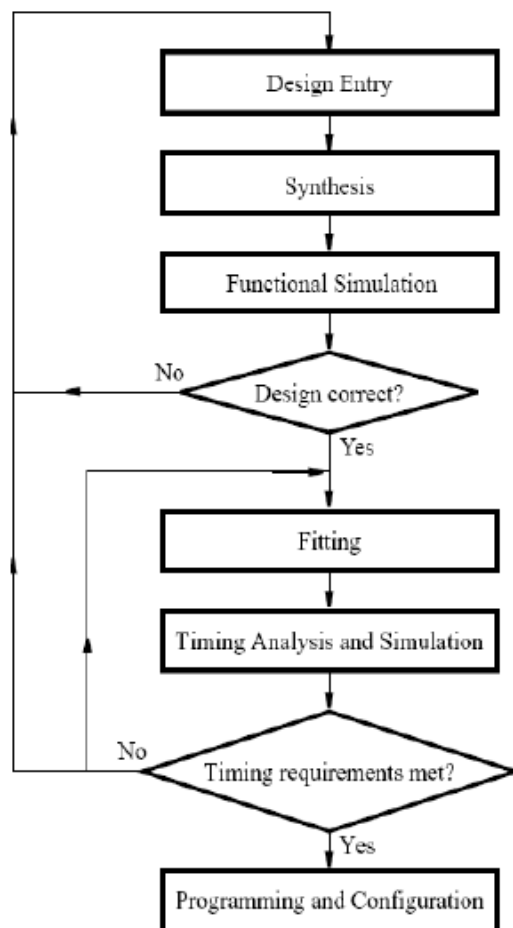
در این بخش چرخه طراحی برای مدارهایی که روی FPGA پیاده سازی می شوند را بررسی نموده و نشان می دهیم که چطور یک چرخه طراحی FPGA با استفاده از نرم افزار QUARTUSII قابل تحقق می باشد. با توجه به اینکه این نرم افزار از همه روش های ایجاد طرح پشتیبانی نموده و ابزارهای مختلفی برای هر یک از آنها دارد، در اینجا از روش برنامه نویسی با زبان Verilog استفاده می کنیم.

توصیف یک چرخه طراحی

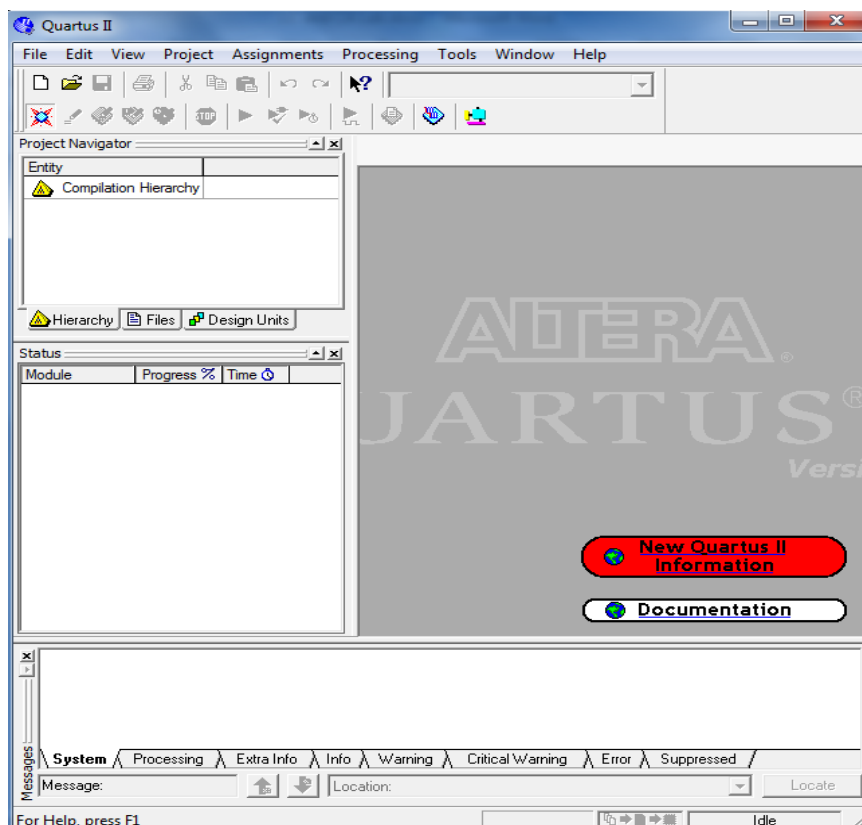
به طور کلی طراحی و پیاده سازی مدارهای منطقی بر روی تراشه های قابل برنامه ریزی با استفاده از نرم افزارهای CAD بسیار راحت و آسان تر می شود. در شکل ۷-۱ چرخه طراحی FPGA با استفاده از نرم افزارهای CAD نشان داده شده است.

به طور کلی چرخه طراحی با استفاده از نرم افزارهای CAD شامل موارد زیر است - که برای هر قسمت ابزاری (نرم افزاری) وجود دارد:

- Design Entry: در این مرحله مدار یا طرح دلخواه شما با استفاده از دیگرام حالت و یا با استفاده از زبانه ای توصیف سخت افزار مانند Verilog مشخص می گردد.
- Synthesis: در این مرحله طرح ایجاد شده به مجموعه ای Logic Element ها تبدیل (سنتز) می شوند.
- Functional Simulation: در این مرحله توسط شبیه ساز، مدار سنتز شده بدون در نظر گرفتن موارد زمانی مانند تأخیر انتشار و . . . شبیه سازی می گردد تا مشخص شود که عملکرد آن درست است یا خیر.
- Fitting: در این مرحله ابزار Fitter که یکی از ابزارهای داخل نرم افزار Quartus II است، مشخص می کند که هر LE ای که در Netlist تعریف شده، دقیقاً کدام LE داخل FPGA می باشد. همچنین نحوه سیم بندی بین این LE های درون FPGA انتخاب می نماید.
- Timing Analysis: در این مرحله تأخیرهای انتشار در مسیرهای مختلف سیم بندی موجود در مدار Fit شده، تحلیل و بررسی گردیده و Performance مدار مشخص می شود.
- Timing Simulation: در این مرحله مدار Fit شده شبیه سازی گردیده تا مشخص شود که عملکرد آن درست است یا خیر.
- Programming and Configuration: در این مرحله مدار طراحی شده روی FPGA برنامه ریزی می شود. برنامه ریزی FPGA، نحوه پیکربندی سوئیچ های داخلی FPGA و همچنین نحوه سیم بندی بین LE ها را مشخص می نماید.



شکل ۷-۱: چرخه طراحی CAD

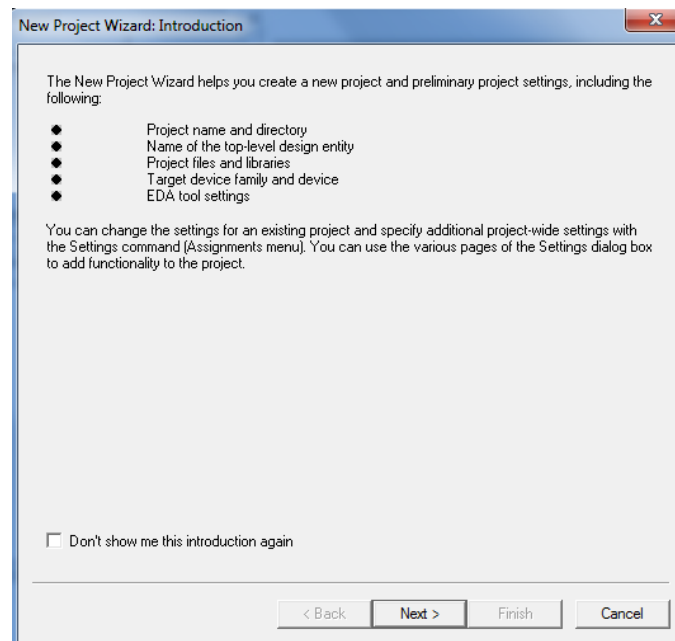


طراحی با نرم افزار Quartus II

برای طراحی با استفاده از این نرم افزار ابتدا آن را اجرا نمایید تا وارد محیط اصلی شوید.

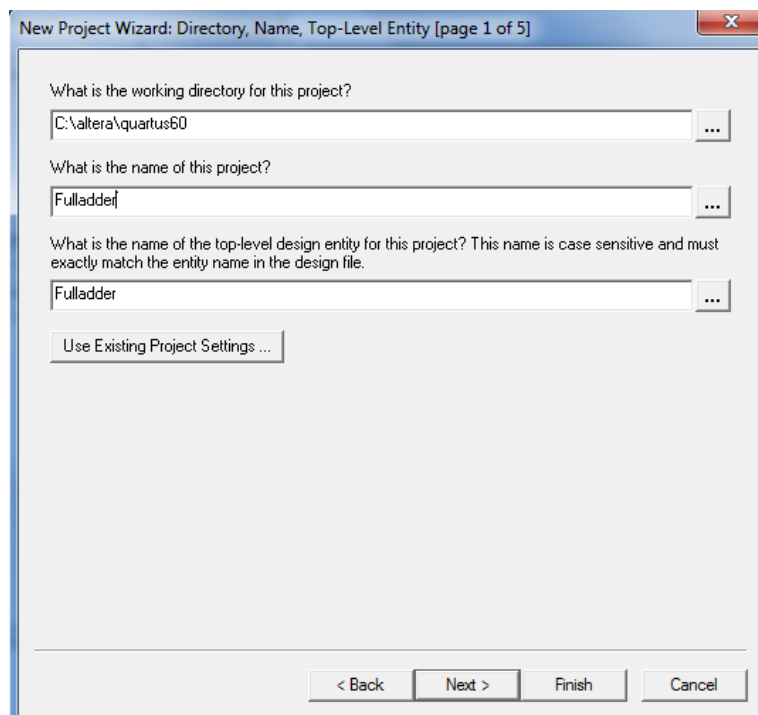
شکل ۷-۲

ایجاد یک پروژه جدید در این نرم افزار توسط یک Wizard صورت می گیرد. برای این منظور File→New Project Wizard را انتخاب کنید تا پنجره زیرنمایان شود.



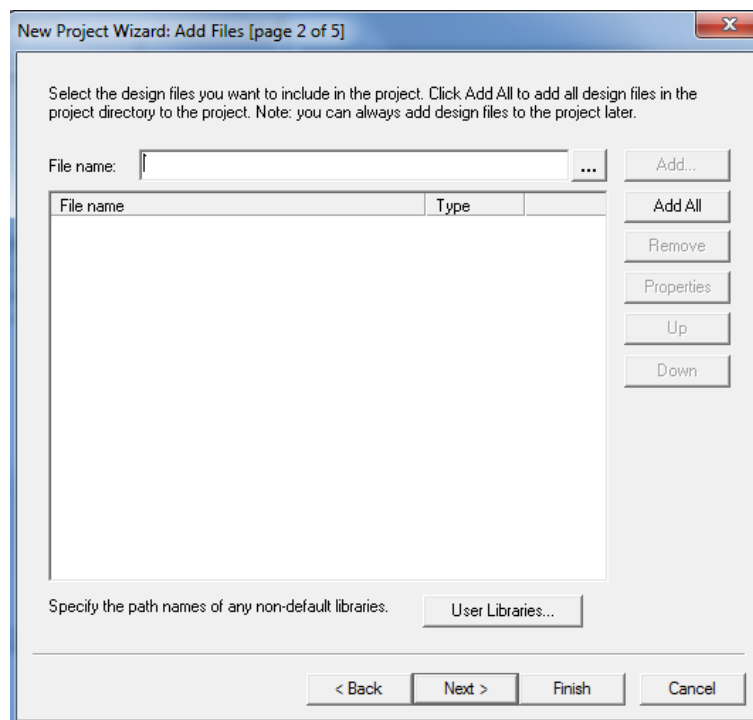
شکل ۳-۷

گزینه Next را کلیک کنید تا پنجره زیرنمایان شود.



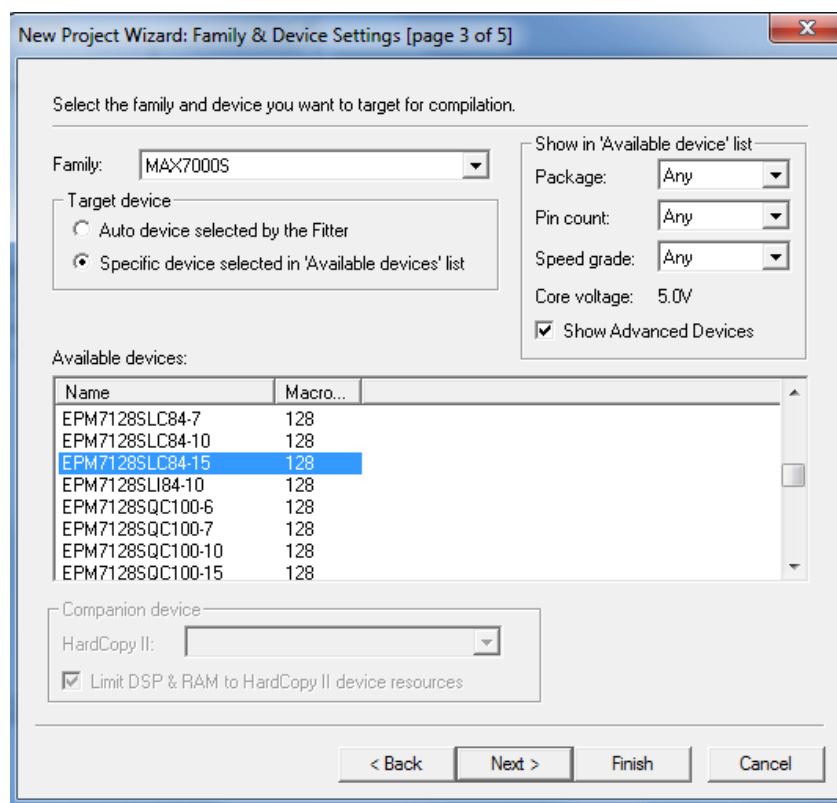
شکل ۴-۷

نام و مسیری برای ذخیره اطلاعات مربوط به پروژه خود در نظر بگیرید. (معمولاً نام پروژه و نام ماژول Top Level یکسان در نظر گرفته می شود.)



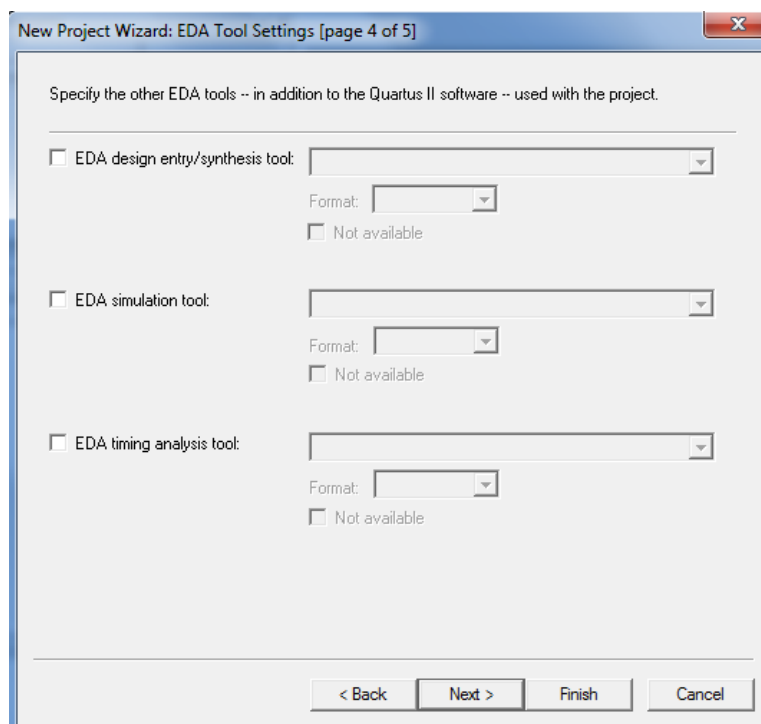
شکل ۵-۷

اگر فایلی برای اضافه کردن به پروژه وجود داشته باشد، Wizard در این قسمت این عمل را به راحتی انجام می دهد، در غیر این صورت فرض کنید فایلی موجود برای اضافه کردن به پروژه نداریم. Next را کلیک کرده تا پنجره بعدی ظاهر گردد. نوع تراشه ای که قرار است طرح شما روی آن پیاده شود را از داخل لیست انتخاب و سپس Next را کلیک کنید. (Device:Emp7128SLC84-15 و Family:Max7000s)

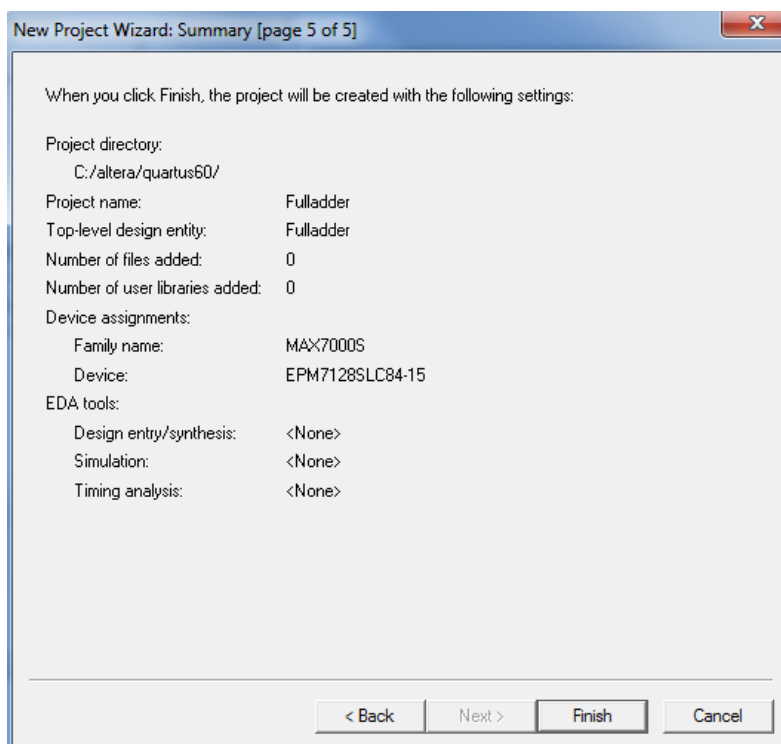


شکل ۶-۷

در این قسمت از Wizard، اگر می‌خواهید علاوه بر Quartus II از نرم‌افزارهای Third Party استفاده کنید، انتخاب نمایید. اما ما برای تمام قسمت‌های چرخه طراحی از ابزارهای داخل Quartus II استفاده می‌کنیم، سپس Next را کلیک کنید. سپس خلاصه‌ای از تنظیمات انجام‌شده برای پروژه، در یک پنجره ظاهر می‌شود Finish را کلیک کنید.

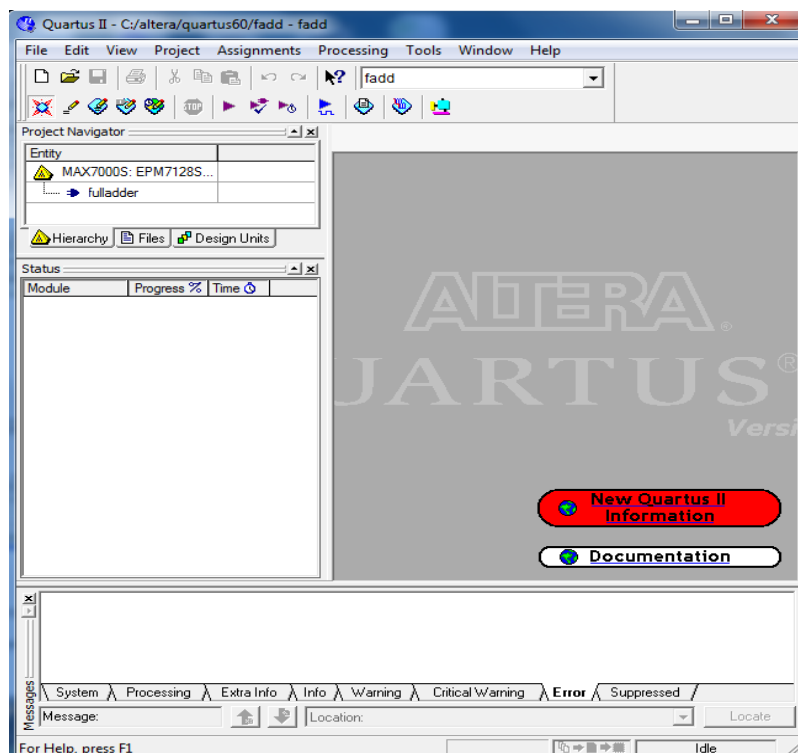


شکل ۷-۷



شکل ۷-۸

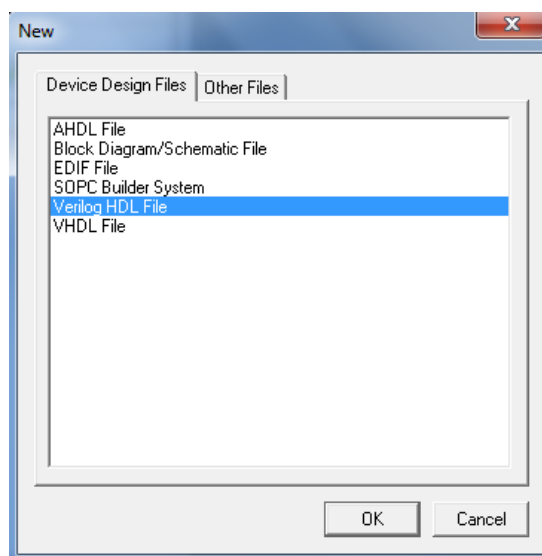
اکنون پروژه شما ایجاد شده و محیط اصلی به شکل زیر خواهد بود.



شکل ۷-۹

Verilog Design Entry

برای نوشتن کد Verilog، نیاز به محیط ویرایشگر نرم‌افزار Quartus II داریم. برای این منظور **File → New** و سپس **Verilog HDL file** را انتخاب و **OK** کنید تا محیط متنی ویرایشگر باز شود.



شکل ۷-۱۰

برای ذخیره کردن فایل **File → Save As** و در **Save Type As** گزینه **Verilog** را انتخاب نمایید.

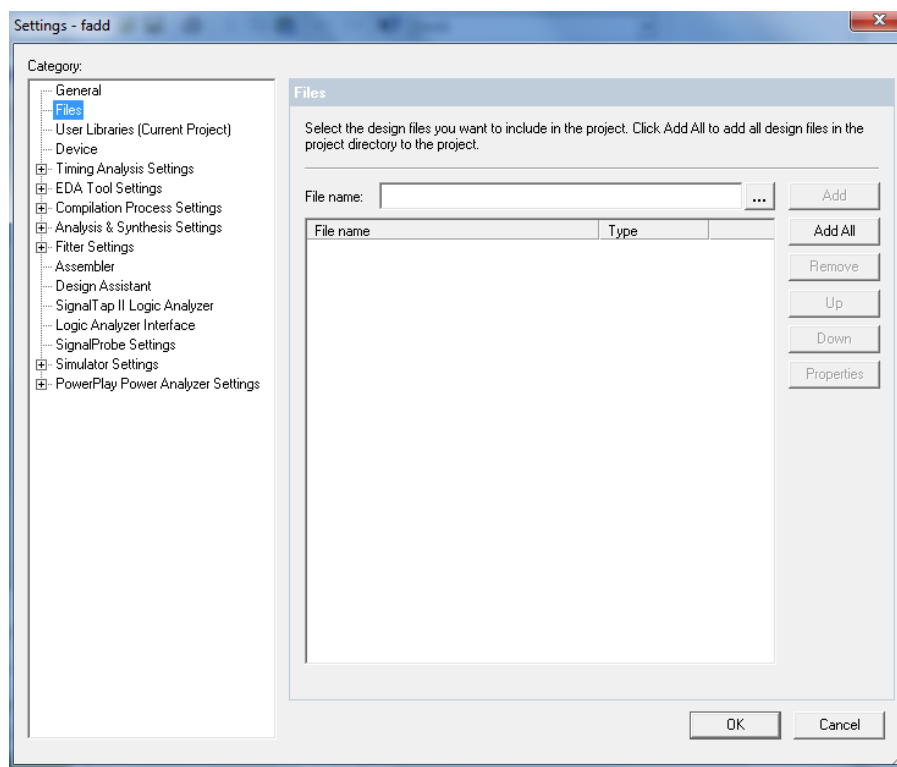
نام فایل را در قسمت File Name بنویسید و گزینه Add File To Current Project را انتخاب کنید.

Save را کلیک کنید. حالا کد موردنظر خود را داخل محیط ویرایشگر بنویسید و ذخیره کنید.

برخی اوقات به خاطر سپردن Syntax دستورات زبان Verilog سخت به نظر می‌رسد به همین دلیل الگوهای مختلفی از دستورات Verilog آماده درون نرم‌افزار وجود دارد. برای استفاده از این الگوها Verilog HDL → Insert Template → Edit را انتخاب کنید.

اضافه کردن فایل‌ها به یک پروژه

گزینه Assignment → Setting را انتخاب کنید تا پنجره زیر ظاهر شود. از ستون سمت چپ File را کلیک کنید. و یا اینکه می‌توانید Project → Add/Remove Files in Project را انتخاب کنید.

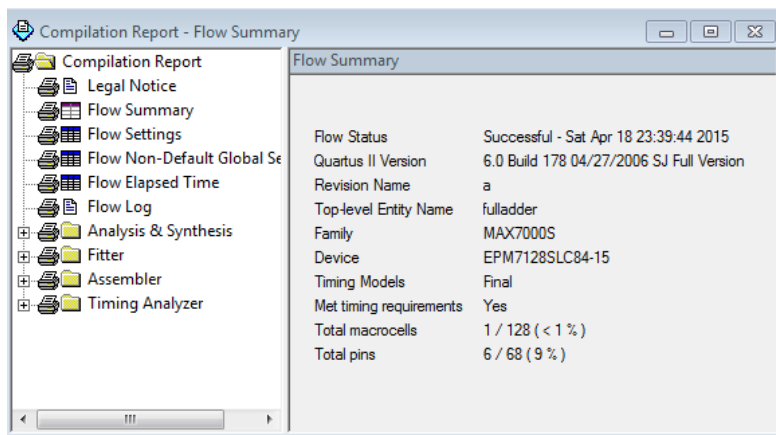


شکل ۷-۱۱

کد Verilog نوشته‌شده توسط ابزارهای مختلف موجود در نرم‌افزار آنالیز، سنتز و قابل پیاده‌سازی روی تراشه موردنظرمی‌گردد. تمام ابزارها به‌وسیله یک برنامه کاربردی به نام کامپایلر کنترل می‌شوند.

برای اجرای کامپایلر گزینه Processing → Start Compilation را انتخاب نمایید و یا اینکه روی آیکون  کلیک کنید. کامپایلر گزارش کامل پیشرفت مراحل را در حین اجرای هر مرحله در پنجره سمت چپ محیط اصلی خواهد داد.

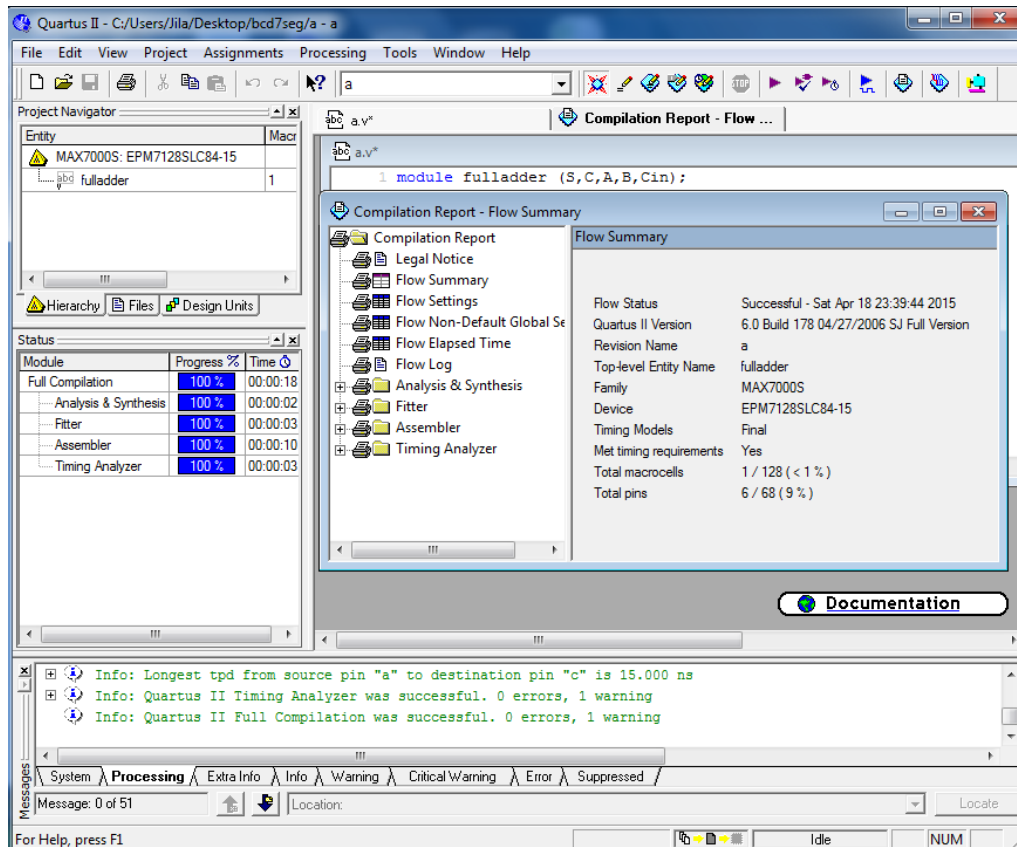
در پنجره پائینی محیط اصلی نیز پیغام‌های مختلفی نشان داده می‌شود. بنابراین وقتی خطا یا اشکالی وجود دارد، این پیغام‌ها، دلیل وجود خطا را بیان می‌نمایند و می‌توانید با دوار-کلیک روی هر پیغام خطا، خط برنامه مورد اشکال مشخص می‌شود. برای اطلاعات بیشتر هر خطا می‌توانید پیغام را انتخاب نموده و F1 را کلیک نمایید.



شکل ۷-۱۲

همچنین بعد از اتمام عمل کامپایل نتیجه گزارش داده می‌شود. و پنجره نشان‌دهنده این گزارش به صورت اتوماتیک به وجود می‌آید. البته

این گزارش با انتخاب **Compilation Report** → **Processin** و یا کلیک روی آیکن  نیز باز می‌شود.



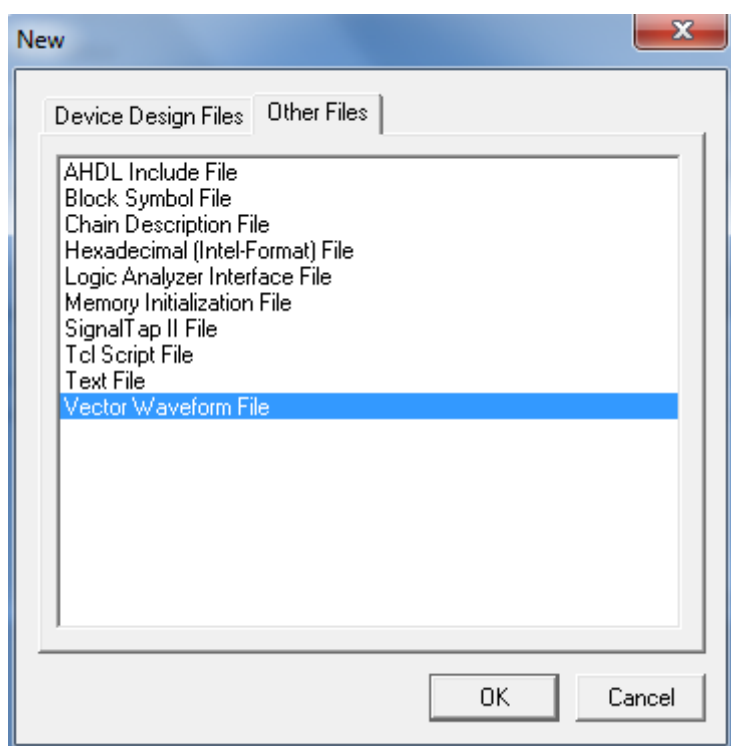
شکل ۷-۱۳

این گزارش شامل چندین بخش می‌باشد که در ستون سمت چپ لیست شده‌اند. مثلاً در این گزارش تعداد پین و LE می‌مورد استفاده برای پیاده‌سازی طرح، مشخص شده است و یا اینکه معادلات منطقی که توسط کامپایلر بعد از عمل سنتز تولید می‌شوند با انتخاب Analysis & Synthesis→Equation نشان داده می‌شود.

شبیه‌سازی طرح

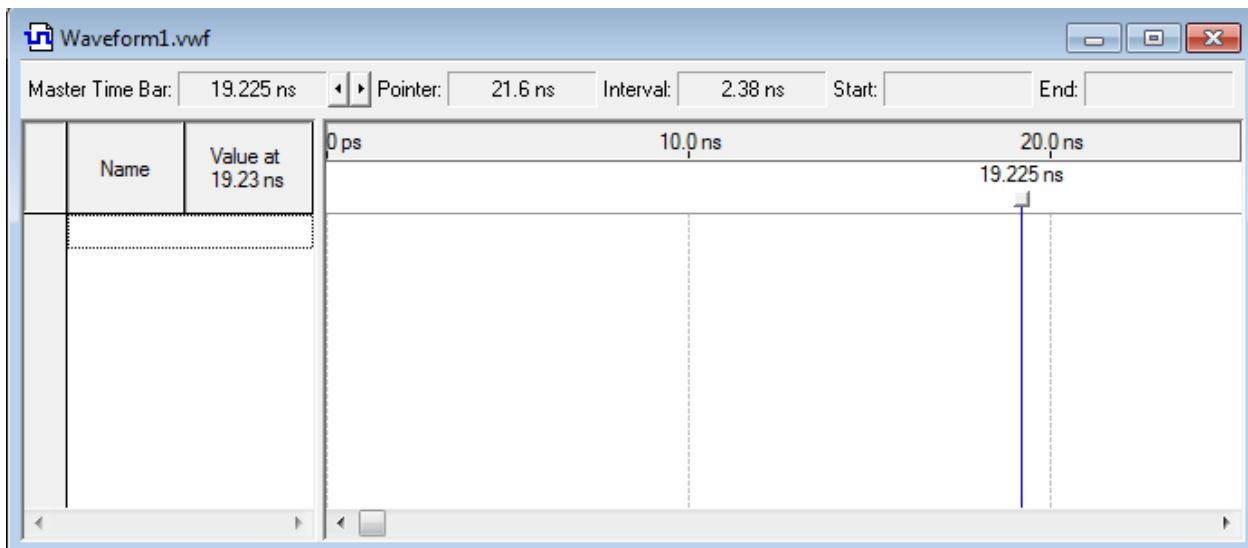
قبل از پیاده‌سازی طرح روی FPGA، باید عمل شبیه‌سازی انجام شود. Quartus II ابزار شبیه‌ساز برای شبیه‌سازی مدار طراحی شده دارد. فقط قبل از انجام شبیه‌سازی، لازم است شکل موج‌هایی با نام Test Vector ایجاد شده به ورودی مای مدار داده شود. همچنین خروجی‌ها و نقاط آزمودن داخلی که لازم است دیده شود، مشخص گردد. سپس شبیه‌ساز، Text vector را به مدل پیاده‌سازی شده طرح می‌دهد و خروجی را تعیین می‌نماید.

برای ایجاد Text Vector در Quartus II از Waveform Editor استفاده می‌نماییم. بدین منظور ابتدا File→New و بعد Other File را انتخاب کنید. با انتخاب Vector Waveform File، Ok را کلیک کنید.



شکل ۷-۱۶

و محیط Waveform Editor باز می‌شود.

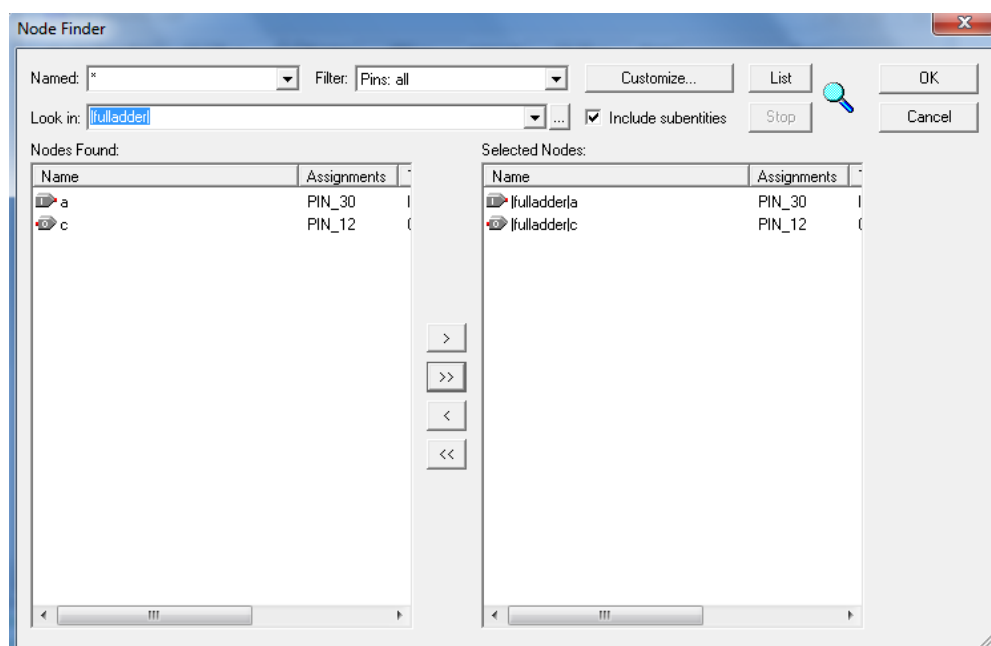


شکل ۷-۱۷

می‌خواهیم ورودی و خروجی مای مدار شبیه‌سازی شوند. Edit → Insert Node or Bus را انتخاب کنید تا پنجره زیر باز شود.

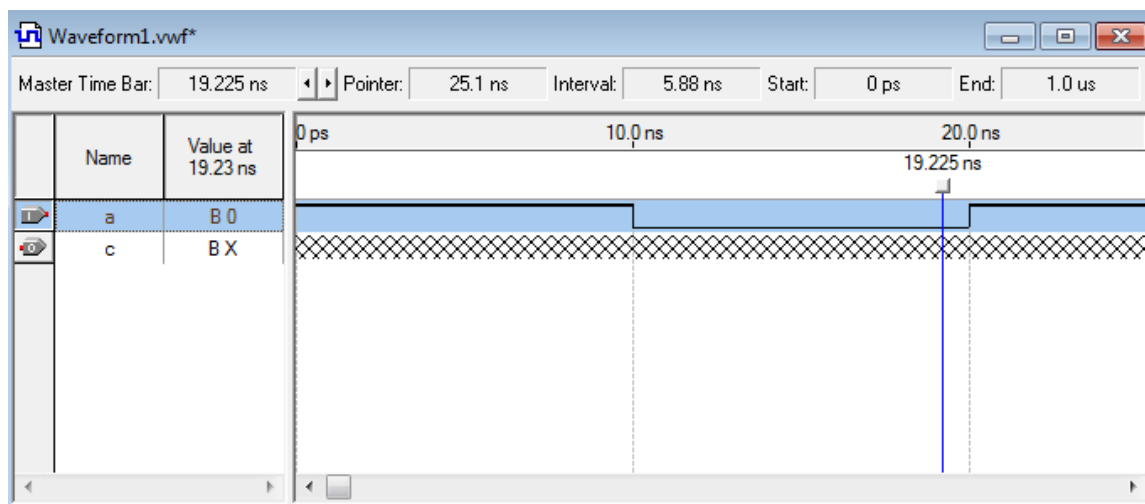
شکل ۷-۱۸

در این قسمت می‌توانیم نام پین را در بخش Name بنویسیم اما بهتر است از Node Finder استفاده نماییم. باید Pind:all و بعد List را برای پیدا کردن Node ها کلیک کنید. سیگنال موردنظر را انتخاب و روی علامت > کلیک کنید تا به Node مای انتخاب‌شده اضافه شود. همین عمل را برای بقیه سیگنال‌ها تکرار کنید و بر روی OK کلیک کنید. حال به سیگنال مای ورودی مشابه نرم‌افزار MaxPlus از منوی سمت چپ که فعال می‌شود مقدار دهید. (و یا Edit → Value)



شکل ۷-۱۹

پس از ساخت شکل موج‌های ورودی، شکل موج خروجی هنوز به صورت هاشور خورده باقی می‌ماند و مقدار آن در حین شبیه‌سازی مشخص می‌شود. فایل را ذخیره نمایید.



شکل ۷-۲۰

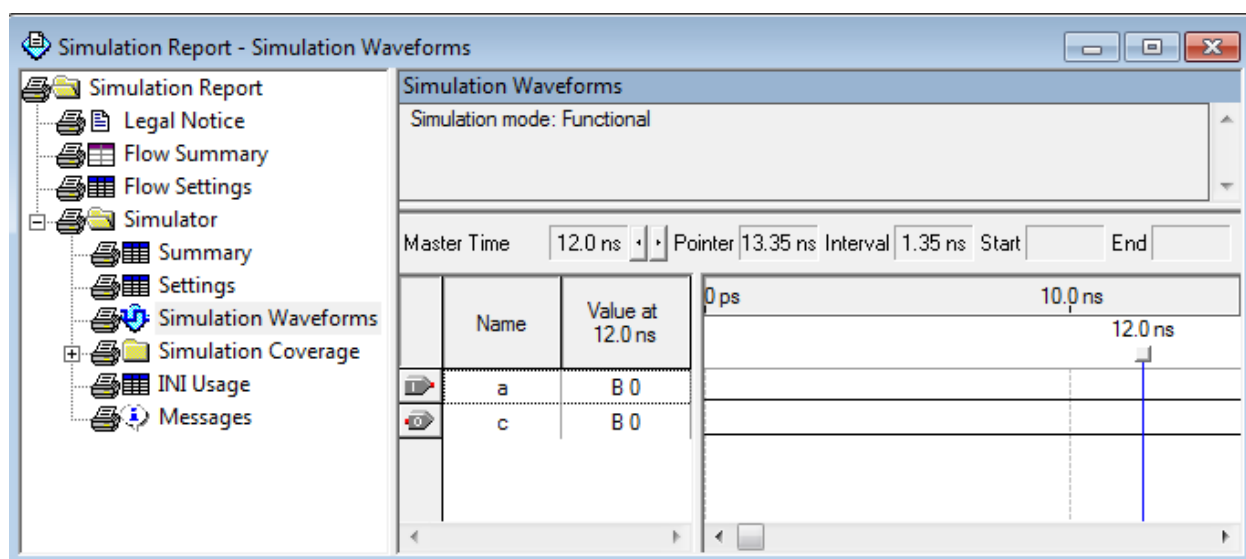
اجرای شبیه‌سازی

مدار طراحی شده می‌تواند به دو روش شبیه‌سازی شود. در روش ساده‌تر LE ها و سیم‌های ارتباطی ایده‌آل و بدون تأخیر انتشار در نظر گرفته شده که به آن شبیه‌سازی عملکردی می‌گویند. در روش دوم که پیچیده‌تر باشد، تمام تأخیر انتشارها محاسبه شده که به آن شبیه‌سازی زمانی گفته می‌شود.

شبیه‌سازی عملکردی

برای اجرای این شبیه‌سازی Simulator Tool → Processing سپس برای Simulation Mode گزینه Functional را انتخاب و Generate Functional Simulation را انتخاب کنید. سپس بر روی Open کلیک کنید و به سیگنال‌های ورودی مقدار دهید. Save کنید و سپس مسیر Simulator Tool → Processing را انتخاب کنید و بر روی گزینه Start کلیک کنید و پس از اتمام با موفقیت و بدون خطا می‌توانید برای دیدن خروجی بر روی گزینه Report کلیک کنید.

شبیه‌سازی با استفاده از آیکون  آغاز می‌شود و در انتها موفقیت شبیه‌سازی گزارش و پنجره زیر ظاهر می‌گردد.



شکل ۷-۲۱

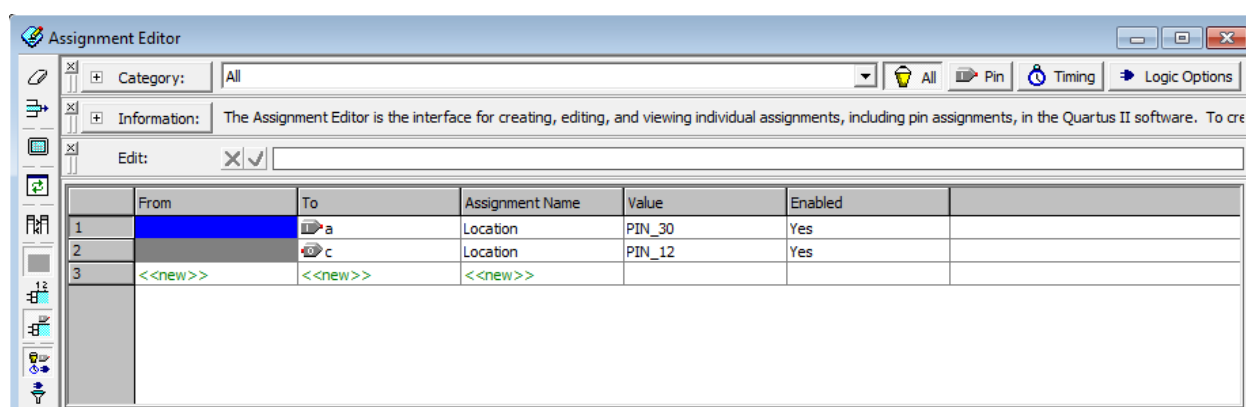
شبیه‌سازی زمانی

بعد از اطمینان از درست بودن عملکرد مدار طراحی‌شده، شبیه‌سازی زمانی را انجام می‌دهیم تا ببینیم رفتار واقعی مدار بعد از پیاده‌سازی روی FPGA چیست. بنابراین بعد از انتخاب Simulator → Setting → Assignment گزینه Timing را برای مد شبیه‌سازی در نظر می‌گیریم و OK می‌نماییم. بعد از اجرای شبیه‌سازی، تأخیر و سیگنال‌های Flitch کاملاً در شکل موج خروجی نمایان هستند.

پیاده‌سازی بر روی برد FPGA

انتساب پین‌ها

اگر کامپایل به‌صورت فوق صورت پذیرد، Quartus II هر پینی از FPGA را که بخواهد برای ورودی و خروجی مای ماژول در نظر می‌گیرد.



شکل ۷-۲۲

برای انتساب پین‌ها از ابزار Assignment Editor استفاده می‌شود. بدین منظور Assignment → Pins را انتخاب و در لیست ارائه‌شده Pin را کلیک نموده و بعد <<NEW>> را که پررنگ شده، دوبار کلیک کنید. یک پنجره پایین‌رو ظاهر می‌شود. روی هر ورودی یا خروجی که می‌خواهید کلیک کنید تا در جدول قرار گیرد. سپس با دوبار کلیک آن پنجره دیگری باز می‌گردد.

Location	I/O Bank	I/O Standard	General Function	Special Fun
PIN_N26		LVTTL		
PIN_M24	I/O Bank 5	Row I/O	L/D5125p	
PIN_M25	I/O Bank 5	Row I/O	L/D5125n	
PIN_N1	I/O Bank 2	Dedicated Clock	CLK1, LVDSCLK0n, Input	
PIN_N2	I/O Bank 2	Dedicated Clock	CLK0, LVDSCLK0p, Input	
PIN_N9	I/O Bank 2	Row I/O	L/D531p	
PIN_N18	I/O Bank 5	Row I/O	L/D5110p	
PIN_N20	I/O Bank 5	Row I/O	L/D5124p	
PIN_N23	I/O Bank 5	Row I/O	L/D5126p, DPCLK7/DQS0R/CQ1R	
PIN_N24	I/O Bank 5	Row I/O	L/D5126n	
PIN_N25	I/O Bank 5	Dedicated Clock	CLK4, LVDSCLK2p, Input	
PIN_N26	I/O Bank 5	Dedicated Clock	CLK5, LVDSCLK2n, Input	
PIN_P1	I/O Bank 1	Dedicated Clock	CLK3, LVDSCLK1n, Input	
PIN_P2	I/O Bank 1	Dedicated Clock	CLK2, LVDSCLK1p, Input	
PIN_P3	I/O Bank 1	Row I/O	L/D526p, DPCLK7/DQS1L/CQ1L#	
PIN_P4	I/O Bank 1	Row I/O	L/D526n	
PIN_P6	I/O Bank 1	Row I/O	L/D522n	
PIN_P7	I/O Bank 1	Row I/O	L/D522p	
PIN_P9	I/O Bank 2	Row I/O	L/D531n	
PIN_P17	I/O Bank 6	Row I/O	L/D5130n	
PIN_P18	I/O Bank 5	Row I/O	L/D5110n	

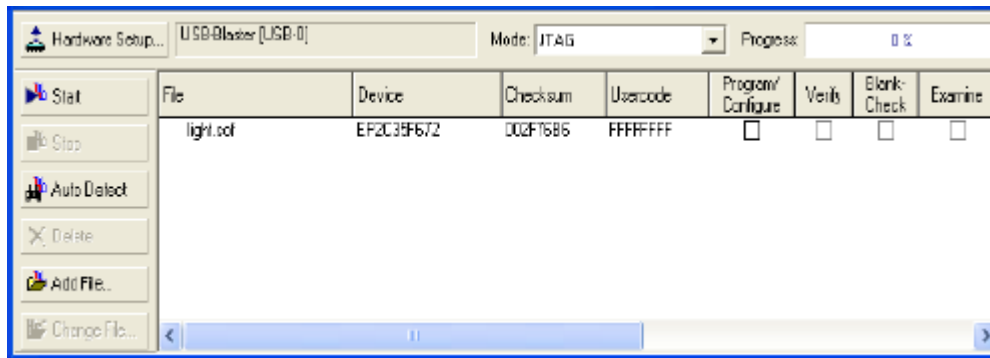
شکل ۷-۲۳

حالا هر پین واقعی روی FPGA را که می‌خواهید انتخاب نمایید تا انتساب صورت گیرد به همین ترتیب بقیه ورودی خروجی‌ها را به پین مای FPGA تخصیص دهید. اکنون با استفاده از File → Save این فایل تخصیص پین‌ها را ذخیره نمایید. بار دیگر طرح خود را که پین مای واقعی و موردنظر شما را برای ورودی و خروجی‌ها در نظر گرفته‌شده است کامپایل نمایید.

برنامه‌ریزی FPGA

فایلی که برای برنامه‌ریزی احتیاج داریم توسط Assembler نرم‌افزار Quartus II تولید می‌شود. و یک فایل باینری بوده که حاوی اطلاعاتی برای پیکربندی FPGA می‌باشد. این فایل SOF. می‌باشد که مخفف SRAM Object File است. در اینجا ما از فایلی با پسوند POF. که مخفف Program Object File می‌باشد که این اجازه را می‌دهد که با یک پیکربندی موازی و از طریق واسطه JTAG برنامه‌ریزی FPGA را انجام دهیم.

چندین روش برای برنامه‌ریزی FPGA وجود دارد که JTAG یکی از روش مای موجود می‌باشد. در این روش اطلاعات پیکربندی مستقیماً بر روی FPGA بار می‌شوند. Programmer → Tools را انتخاب نمایید تا پنجره زیر ظاهر شود.



شکل ۷-۲۴

در اینجا لازم است پروگرامر سختافزاری و مدی که استفاده خواهد شد را مشخص کنید. JTAG را برای مد در نظر بگیرید و از قسمت Hardware Setup گزینه ByteBlaster را انتخاب کنید. گزینه Auto Detect را کلیک کنید تا FPGA متصل به سیستم شناسایی شود سپس در صورتی که فایل برنامه‌ریزی شما در لیست قرار نگرفته با استفاده از گزینه Add file فایل را باز کنید و سپس بر روی Start کلیک کنید تا FPGA برنامه‌ریزی شود.

جلسه ۸

آزمایش طراحی واحد محاسبه و منطق^{۲۷}

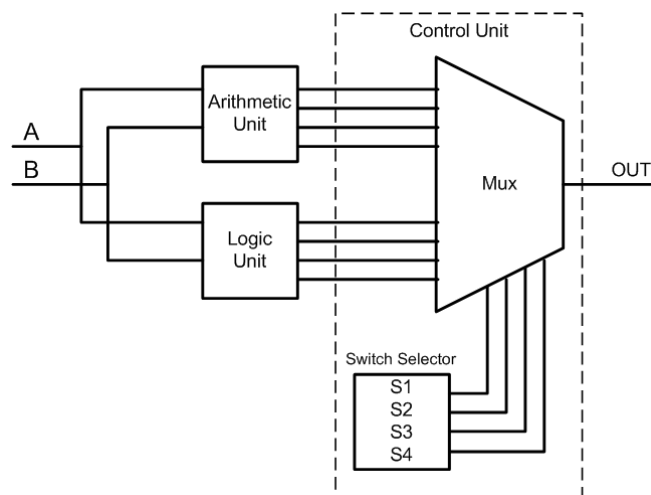
هدف

در این آزمایش اهداف زیر دنبال می‌شوند :

- ✓ 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. به‌طور کلی ALU از سه قسمت محاسبات، منطق و کنترل تشکیل شده 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 AND, OR, XOR, NOT صورت می‌پذیرد و در نهایت 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 ۸-۱ - بلوک دیاگرام 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 تا کمی بیشتر در مورد بلوک‌های این سیستم به بحث بپردازیم. در بخش محاسباتی دانشجویان عزیز می‌بایست از بلوک جمع‌کننده‌ای که در آزمایش مای اخیر طراحی شد به‌عنوان جمع‌کننده ۴ - بیتی استفاده نمایند. با استفاده از همان مدار نیز می‌توانند افزایش به‌اندازه یک واحد را نیز پیاده‌سازی نمایند. اما مدار تفریق‌کننده و کاهش به‌اندازه یک واحد را که در آزمایش - های گذشته به آن‌ها نپرداخته‌اند می‌بایست از ابتدا طراحی نمایند. البته طراحی تفریق‌کننده ۴ - بیتی نیز بر پایه تمام جمع‌کننده امکان‌پذیر است و از این‌رو با جمع‌کننده 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 ۸-۲

تکالیف پیش از آزمایش

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

تکالیف داخل آزمایشگاه

نکات : دانشجویان عزیز برای مشاهده عملکرد مدارهای طراحی شده 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) آزمون‌های لازم را انجام و نتایج را در گزارش خود ثبت نمایید.

جلسه ۹

آزمایش ثبات‌ها و گذرگاه داده

ثبات‌ها

از مهم‌ترین بخش‌های تشکیل‌دهنده سیستم‌های دیجیتال ثبات‌ها می‌باشند. در این سیستم‌ها بسیاری از اعمال داخلی توسط ثبات‌ها یا داده‌هایی که در آن‌ها ذخیره می‌شوند انجام می‌گیرند.

هدف اصلی این بخش طراحی مهم‌ترین المان یک کامپیوتر پایه است. ثبات‌ها به صورت‌های مختلفی در داخل سیستم‌های دیجیتال یا کامپیوتر مورد استفاده قرار می‌گیرند. کاربردهای مختلفی دروس مدار منطقی و معماری کامپیوتر برای شما ارائه گردیده است که در این آزمایش‌ها مشاهده می‌کنید. این بخش از سه آزمایش تشکیل شده است.

در آزمایش اول شما با طراحی یک ثبات که به صورت سری و موازی قابل خواندن و نوشتن می‌باشد آشنا می‌گردید. همچنین نحوه طراحی و پیاده‌سازی ثبات‌های انتقالی^{۲۸} چرخشی و ریاضی به شما آموزش داده خواهد شد.

در آزمایش دوم ثبات‌هایی که دارای قابلیت‌های مختلفی نظیر متمم یک، متمم دو افزایشی یک و کاهش یک است را طراحی می‌نمایید. آزمایش سوم تکمیل‌کننده دو آزمایش قبل است. در این آزمایش شما باید یک ثبات که دارای همه قابلیت‌هایی که در آزمایش‌های اول و دوم انجام داده بودید به صورت مجتمع طراحی نمایید.

۱-۱- ثبات ۱

هدف

در این آزمایش اهداف زیر دنبال می‌شوند :

- ✓ طراحی و پیاده‌سازی یک ثبات انتقالی
- ✓ طراحی و پیاده‌سازی یک ثبات ۱۶ بیتی که نحوه ورودی و خروجی آن به شکل زیر باشد :
- قابلیت ثبت موازی و سری و خواندن موازی و سری^{۲۹}
- ثبات انتقالی چرخشی به چپ و راست
- ثبات انتقالی ریاضی به چپ و راست

^{۲۸} Shift Register
^{۲۹}

SISO, SIPO, PIPO PISO

تئوری آزمایش

ثبات‌ها یکی از قسمت‌های مهم ساختار یک سیستم پردازشگر دیجیتال می‌باشند. ثبات‌ها اغلب برای ذخیره موقت اطلاعات باینری استفاده می‌شوند و شامل مجموعه‌ای از فلیپ فلاپ‌ها هستند. یک ثبات n بیتی شامل n فلیپ فلاپ است. ثبات‌ها را می‌توان از لحاظ ورودی و خروجی به چهار دسته تقسیم نمود. این تقسیم‌بندی عبارت است از :

✓ ورودی موازی - خروجی موازی (PIPO)

✓ ورودی سری - خروجی موازی (SIPO)

✓ ورودی موازی - خروجی سری (PISO)

✓ ورودی سری - خروجی سری (SISO)

یک ثبات انتقالی از مجموعه‌ای از فلیپ فلاپ‌ها تشکیل شده است که به صورت زنجیره‌ای به هم متصل هستند و ورودی هر فلیپ فلاپ به خروجی فلیپ فلاپ قبلی متصل است. از ثبات‌های بسیار مهم می‌توان به ثبات انتقالی اشاره نمود. ثبات انتقال بنا به نوع سیگنال کنترلی آن ممکن است اطلاعات را به سمت راست یا به سمت چپ انتقال دهد. از دیدگاه زمان‌بندی ثبات‌ها را می‌توان به دودسته سنکرون و آسنکرون تقسیم‌بندی نمود. ثبات سنکرون ثباتی است که دارای سیگنال تحریک یکسان برای هر فلیپ فلاپ می‌باشد. در ثبات آسنکرون این عمل وجود ندارد. در کامپیوتر پایه بنا به نوع کاربردی که در نظر گرفته شده است دو نوع شیفت ریاضی و چرخشی اضافه گردیده است. بنابراین در کنار هر ثبات انتقالی یک فلیپ فلاپ قرار خواهد گرفت که به وسیله آن ثبات انتقالی چرخشی و ریاضی با ثبات انتقالی مرسوم تکمیل می‌گردد.

تکالیف پیش از آزمایش

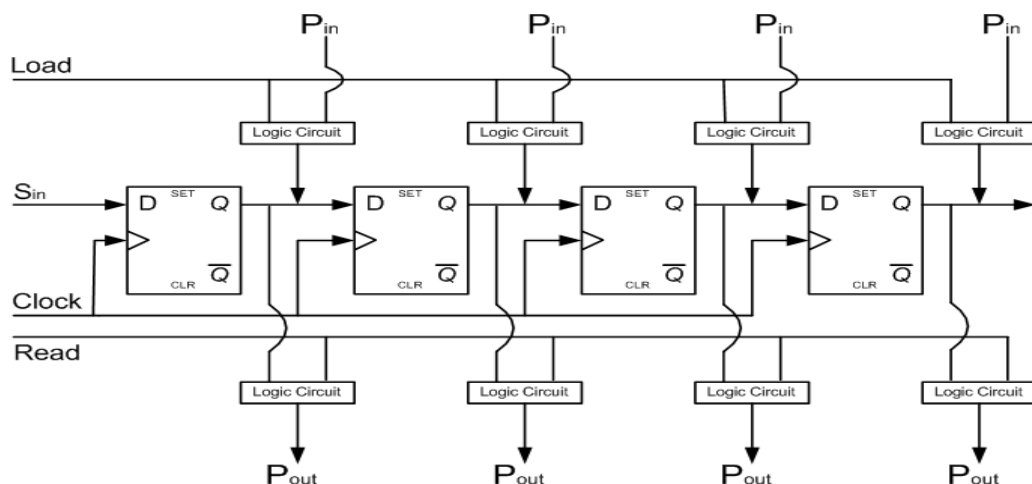
(۱) در این آزمایش ابتدا یک ثبات انتقالی طراحی نمایید که دارای یک سیگنال کنترلی است. این سیگنال کنترلی مشخص‌کننده حالت انتقال به چپ یا به راست می‌باشد. لازم به ذکر است که در کلیه طراحی‌ها از فلیپ فلاپ D استفاده نمایید. برنامه ثبات موردنظر را نوشته و خروجی آن را تحلیل نمایید.

(۲) ثبات انتقالی که به صورت موازی و سری ثبت و به صورت موازی و سری خوانده می‌شود را طراحی نمایید.

(۳) اکنون که طراحی شماتیک کامل گردید کد Verilog ثبات انتقالی را که دارای قابلیت‌های ذکر شده در بند ۲ می‌باشد را تولید نمایید.

(۴) با دانستن نحوه طراحی یک ثبات انتقالی با DFF، اکنون مراحل ۲ و ۳ را برای ثبات انتقالی چرخشی و ریاضی تکمیل کنید. در این مرحله شما می‌توانید تمامی قابلیت‌های فوق را بر روی یک ثبات پیاده‌سازی نمایید.

راهنمایی : هر یک از این ثبات‌ها نیازمند سیگنال‌های کنترل نظر CLK, Read و Load و ... می‌باشند که بنا به نوع تبادل باید در نظر گرفته شوند. شکل زیر به طور خلاصه راهنمایی‌های لازم را برای شما خواهد داشت. لازم به ذکر است که این بلوک دیاگرام بسیار کلی است و کلیه حالات فوق را در بردارد. بنابراین در هر کدام از مدارات فوق بخشی از این شکل بکار می‌رود.



شکل ۹-۱- بلوک دیاگرام کلی مدارها

تکالیف داخل آزمایشگاه

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

- ✓ برای قابل رؤیت شدن باید پریود Clock را در حدود ۵۰۰ms قرارداد.
- ✓ بر روی برد، چهار LED یکرنگ وجود دارد از این LED ها به عنوان بیت‌های خروجی ثابت‌ها استفاده نمایید. از سوئیچ‌های موجود بر روی برد به عنوان سیگنال‌های کنترلی و ورودی استفاده نمایید.

۱) مدار ثابت انتقال SISO و SIPO، PISO، PIPO را بر روی برد پیاده‌سازی و آزمودن نمایید. نتایج به دست آمده را در گزارش خود ثبت نمایید.

۲) مدار ثابت انتقال چرخشی به چپ و راست را بر روی برد پیاده‌سازی و آزمودن نمایید. نتایج به دست آمده را در گزارش خود ثبت نمایید.

۳) مدار ثابت انتقال ریاضی به چپ و راست را بر روی برد پیاده‌سازی و آزمودن نمایید. نتایج به دست آمده را در گزارش خود ثبت نمایید.

۴) مدار ثابت انتقالی که همه قابلیت‌های فوق را دارا است بر روی برد پیاده‌سازی و آزمودن نمایید. نتایج به دست آمده را در گزارش خود ثبت نمایید.

۲-۱- ثبات ۲

هدف

در این آزمایش اهداف زیر دنبال می‌شوند :

طراحی و پیاده‌سازی یک ثبات ۱۶ بیتی که به شکل زیر باشد :

- ✓ با قابلیت متمم یک
- ✓ با قابلیت متمم دو
- ✓ با قابلیت افزایشی یک

تئوری آزمایش

در این آزمایش شما با کاربردهای دیگری از ثبات‌ها آشنا خواهید شد. این کاربردها، که در بالا به آن‌ها اشاره شده است، در قسمت‌های مختلف یک سیستم دیجیتال مورد استفاده قرار می‌گیرند. به طور مثال در سیستم کامپیوتر پایه برای انجام عملیات افزایش یک یا کاهش یک به راحتی در یک سیکل با سیگنال‌های INC یا DEC انجام خواهند شد. طراحی چنین ثباتی که دارای قابلیت‌های فوق است، در سیکل‌های اجرائی بسیاری از دستورالعمل‌ها صرفه جویی خواهد نمود.



شکل ۹-۲

تکالیف پیش از آزمایش

(۱) در این آزمایش ابتدا یک ثبات با قابلیت افزایشی یک با استفاده از فلیپ فلاپ D طراحی نمایید. سپس کد Verilog مرتبط با این مدار را نوشته و آزمودن کنید.

(۲) مرحله ۱ را به طور کامل برای سه کاربرد دیگر، طراحی و آزمودن می‌نمایید.

یک ثبات طراحی و آزمودن نمایید که دارای هر چهار قابلیت فوق یعنی متمم یک، متمم ۲، افزایشی یک و کاهشی یک باشد. سپس کد Verilog مرتبط با این مدار را نوشته و آزمودن نمایید.

تکالیف داخل آزمایشگاه

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

✓ برای قابل رؤیت شدن باید پریود Clock را در حدود ۵۰۰ms قرارداد.

✓ بر روی برد، چهار LED یک رنگ وجود دارد از این LED ها به عنوان بیت‌های خروجی ثبات‌ها استفاده نمایید.

✓ از سوئیچ‌های موجود بر روی برد به عنوان سیگنال‌های کنترلی و ورودی استفاده نمایید.

برنامه ثبات با قابلیت‌های فوق را بر روی برد پیاده‌سازی و آزمودن نمایید. نتایج به دست آمده را در گزارش خود ثبت نمایید.

۳-۱- ثبات ۳

هدف

در این آزمایش اهداف زیر دنبال می‌شوند :

طراحی و پیاده‌سازی یک ثابت ۴ بیتی که به شکل زیر باشد :

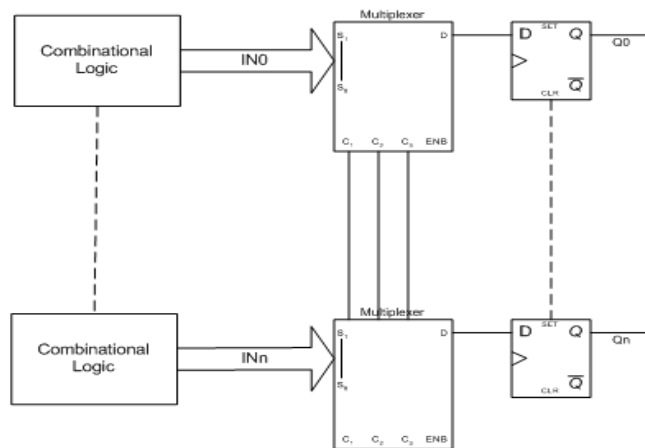
- ✓ SIPO, SISO, PIPO, PISO
- ✓ Circulate Right shift, Circulate Left shift
- ✓ Arithmetic Right shift, Arithmetic Left shift
- ✓ One's complement, Two's complement
- ✓ INC, DEC
- ✓ Clear

تئوری آزمایش

این آزمایش تکمیل‌کننده دو آزمایش قبل است. شما در آزمایش‌های گذشته با طراحی و پیاده‌سازی هر یک از عملگرهای فوق به صورت مجزا آشنا شدید. در این آزمایش شما با دانش قبلی که در زمینه طراحی تک‌تک آن‌ها به دست آورده‌اید. اکنون به راحتی می‌توانید یک ثابت با قابلیت‌های فوق طراحی نمایید. همچنان که در آزمایش قبل ذکر شد، داشتن چنین ثابتی باعث کاهش سیکل‌های اجرایی یک دستورالعمل در کامپیوتر می‌گردد. به عبارت دیگر ریز دستورالعمل‌های مورد نیاز برای یک دستورالعمل کاهش پیدا می‌کند.

تکالیف پیش از آزمایش

در این آزمایش با استفاده از Multiplexer ورودی‌های فلیپ فلاپ D مشخص می‌گردد. شکل ۳-۹-۱ بلوک دیاگرام کلی مدارات مورد نظر را نمایش می‌دهد. مدار مورد نظر را ترسیم ، برنامه آن را نوشته و خروجی را تحلیل نمایید.



شکل ۳-۹-۱ بلوک دیاگرام کلی مدارات مورد نظر

تکالیف داخل آزمایشگاه

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

✓ برای قابل‌رؤیت شدن باید پریود Clock را در حدود ۵۰۰ms قرارداد.

✓ بر روی برد، چهار LED یک‌رنگ وجود دارد از این LED ها به‌عنوان بیت‌های خروجی ثبات‌ها استفاده کرده و ثبات‌ها را ۴ بیتی طراحی کنید. از سوئیچ‌های موجود بر روی برد به‌عنوان سیگنال‌های کنترلی Multiplexer استفاده نمایید.

برنامه ثبات باقابلیت‌های فوق را بر روی برد پیاده‌سازی و آزمودن نموده و نتایج به‌دست‌آمده را در گزارش خود ثبت کنید .

۲- گذرگاه داده^{۲۰}

هدف

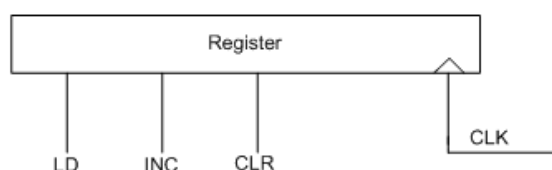
در این آزمایش اهداف زیر دنبال می‌شوند :

✓ آشنایی با BUS و ساختار آن

✓ طراحی و پیاده‌سازی مدارهای محاسباتی ، منطقی و ثباتی یک CPU

تئوری آزمایش

BUS یکی از واسطه‌های بسیار مهم در سیستم‌های دیجیتالی است. در این سیستم‌ها برای ارسال و دریافت داده از BUS به‌عنوان محیط انتقال استفاده می‌گردد. همان‌طور که در معماری کامپیوتر با آن آشنا شده‌اید به دو طریق BUS ها قابل پیاده‌سازی هستند. آن دو راه به‌کارگیری Multiplexer و استفاده از بافر tri-state می‌باشد. در کامپیوترها به علت دوطرفه بودن BUS ، امکان استفاده از Multiplexer برای یک خط که خواندن و نوشتن از روی آن صورت می‌گیرد امکان‌پذیر نمی‌باشد. بنابراین این عمل از طریق tri-state انجام خواهد شد. در فصل‌های قبل با نحوه طراحی ثبات‌ها آشنا شدید. در سیستم کامپیوتر پایه، یک ثبات ممکن است دارای قابلیت‌های مختلفی باشد. اما در کامپیوتر پایه‌ای که برای این آزمایشگاه طراحی می‌شود یک ثبات باید قابلیت مای Increment ، LD ، Clear و Read را دارا باشد (شکل ۹-۴) .



لازم به ذکر است که ثبات‌های مذکور ثبات‌های داخلی کامپیوتر و مورد استفاده در قسمت کنترلی و BUS خواهند بود. نکته بسیار مهم نحوه ارتباط این ثبات‌ها با BUS سیستم می‌باشد. همان‌طور که می‌دانید به علت دوطرفه بودن BUS داده، خواندن و نوشتن از یک مسیر صورت خواهد گرفت. بنابراین هر ثبات باید در زمان مورد نیاز BUS را در اختیار گیرد و سپس بعد از اتمام ارسال یا دریافت آن را رها نماید. با توجه به مطالب فوق T در قسمت ورودی هر ثبات با سیگنال کلاک و LD مشکل وارد شدن داده جدید به ثبات حل می‌شود. اما در قسمت خروجی باید در مسیر هر بیت خروجی یک tri-state قرار گیرد که با سیگنال Read هر ثبات فعال شده و بعد از خواندن، خروجی ثبات به صورت امپدانس بالا درآید. با طراحی چنین ثباتی به راحتی می‌توان تعداد زیاد ثبات را که از یک BUS واحد استفاده می‌نمایند، به یکدیگر متصل نمود.

اکنون با داشتن یک ثبات که قابلیت‌های اساسی سیستم کامپیوتر پایه را دارا می‌باشد، به معرفی ثبات‌های این پردازشگر پایه می‌پردازیم. ثبات‌های مهم این سیستم عبارت‌اند از:

- (۱) ثبات ^{۳۱}PC: آدرس شروع برنامه و خط بعدی برنامه را دارا می‌باشد.
- (۲) ثبات ^{۳۲}AR: آدرس دستورالعمل یا داده را برای حافظه تأمین می‌نماید.
- (۳) ثبات ^{۳۳}IR: کد باینری دستورالعمل از طریق حافظه در این ثبات قرار می‌گیرد.
- (۴) ثبات ^{۳۴}DR: ثباتی که محتویات داده هر عملگر که بر اساس حافظه است در آن قرار می‌گیرد. به تعبیر دیگر، هر یک از اعمال ریاضی و منطقی که با دو داده انجام می‌شود یک‌طرف آن در DR خواهد بود.
- (۵) AC: ثبات پردازنده که مهم‌ترین ثبات پردازشگر و تمام اعمال ریاضی، منطقی ثباتی، ورودی و خروجی با این ثبات انجام می‌شود.

(۶) ^{۳۵}TR: ثبات موقتی که در حین انجام بعضی از دستورالعمل‌ها، داده به‌طور موقت در آن ذخیره خواهد شد.

(۷) ^{۳۶}INPR: ثباتی که برای ورود داده به داخل کامپیوتر مورد استفاده قرار می‌گیرد.

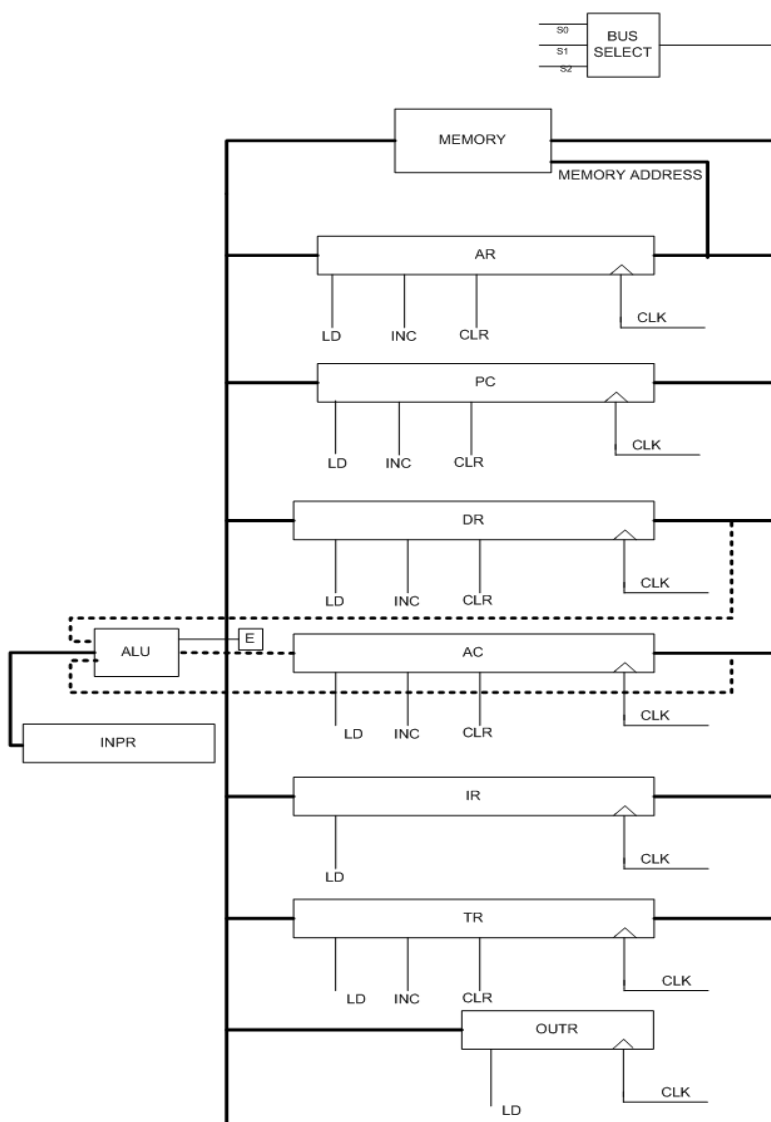
(۸) ^{۳۷}OUTR: برای ارسال داده به خارج از کامپیوتر از این ثبات استفاده می‌شود.

شکل ۹-۵ نحوه ارتباطات بین این ثبات‌ها و سیگنال‌های کنترل مورد نیاز آن‌ها را نمایش می‌دهد.

تکالیف پیش از آزمایش

^{۳۱} Program Counter
^{۳۲} Address Register
^{۳۳} Instruction Register
^{۳۴} Data Register
^{۳۵} Temporary Register
^{۳۶} Input Register

- ۱- ثابتی ۱۶ بیتی مطابق با شکل ۹-۴ طراحی نمایید. در این ثابت از فلیپ فلاپ‌های D استفاده می‌شود و در قسمت خروجی باید در مسیر هر بیت خروجی یک tri-state قرار گیرد که با سیگنال Read فعال شده و بعد از خواندن، خروجی ثابت به صورت امپدانس بالا درآید.
- ۲- شکل ۹-۵ را با استفاده از ثابتی که در بند اول طراحی نموده‌اید، پیاده‌سازی نمایید. در این قسمت به جای حافظه یک ثابت و به جای ALU یک بافر قرار دهید. سپس کد Verilog مرتبط با این مدار را نوشته و آزمودن نمایید.



شکل ۹-۵- نحوه ارتباطات بین ثابت‌ها در BUS

تکالیف داخل آزمایشگاه

- ۱- در این آزمایش شما باید در Simulator ، یک محیط آزمودن برای برنامه BUS بند ۲ ، نوشته و به طور کامل ارتباطات بین ثابت‌های مختلف را آزمودن نمایید.

نکته: این BUS به عنوان یکی از المان‌های مهم کامپیوتر پایه می‌باشد. بنابراین باید هر گروه تا پایان ترم تمامی فایل‌های مربوط به آن را نگهداری نماید.

اکنون باید BUS و ثبات‌های آن را به ALU متصل نمایید و دوباره قسمت‌های ارتباطی را به همراه ALU آزمودن نمایید.

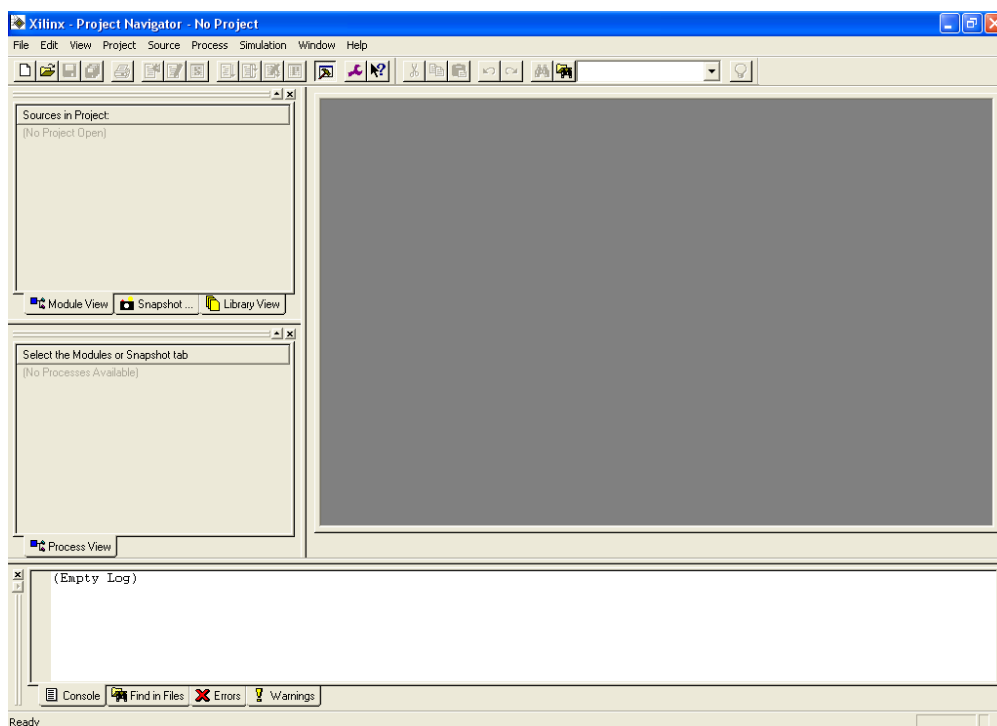
جلسه ۱۰

آشنایی با نرم افزار Xilinx ISE

مقدمه

شما در این دوره چگونگی طراحی توسط نرم افزار Xilinx ISE را به طور دقیق ، با جزئیات کامل می آموزید و یاد می گیرید چگونه با انواع روش های طراحی با تراشه های شرکت Xilinx کار کنید . روش های طراحی (design entry method) ذکر شده در اینجا شامل طراحی به وسیله نوشتن کد با زبان Verilog/VHDL ، طراحی شماتیکی با محیط ECS و طراحی به وسیله دیاگرام حالت با محیط StateCAD می باشد. پس مراحل زیر را با آرامش خاطر دنبال کنید . . .

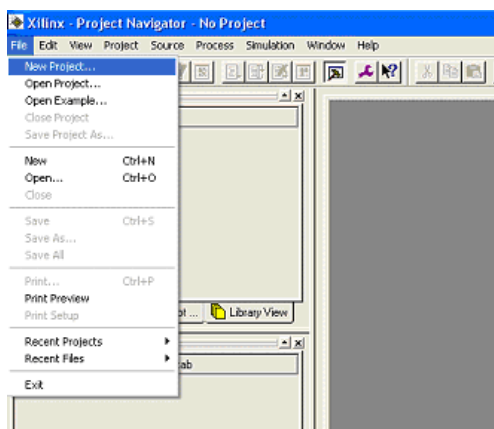
شکل زیر محیط کلی این نرم افزار را نشان می دهد .



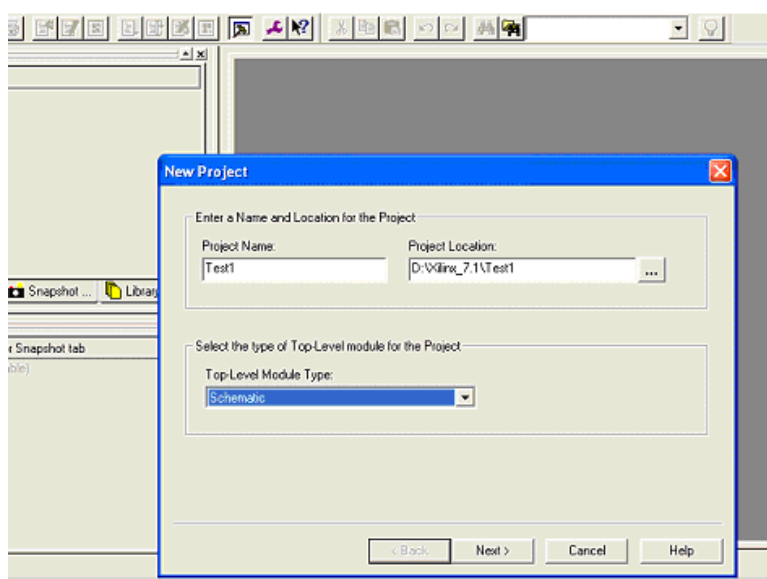
شکل ۱-۱۰

ایجاد یک پروژه جدید

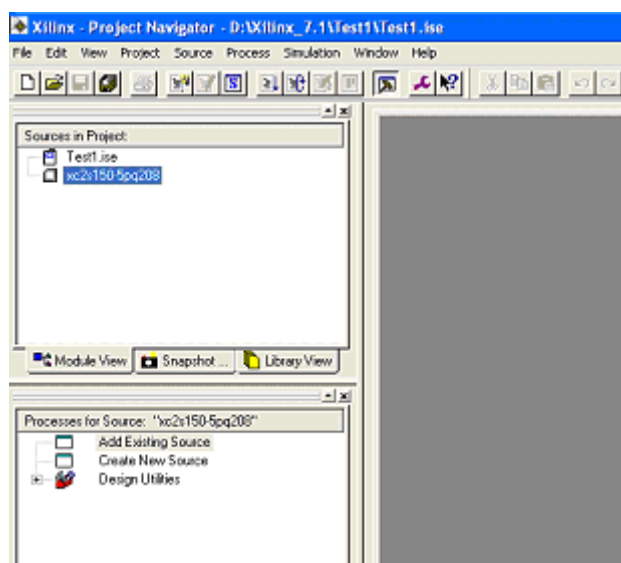
گزینه File>New Project را انتخاب نموده و اسم پروژه خود را در جای مربوطه وارد کنید.



شکل ۲-۱۰



شکل ۳-۱۰

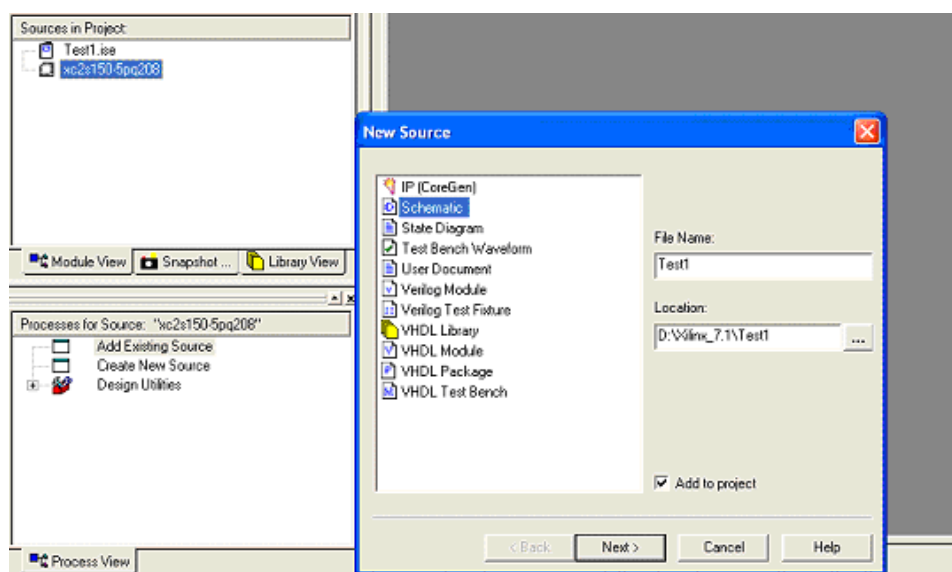


گزینه Next را کلیک کرده ، نوع و شماره IC خود را به همراه دیگر مشخصات آن و نیز زبان برنامه‌نویسی را انتخاب نمایید. بقیه پنجره‌ها را به‌صورت پیش‌فرض بپذیرید و در انتها Finish را کلیک نمایید. در نهایت پنجره نرم‌افزار شما به‌صورت زیر درخواهد آمد.

شکل ۴-۱۰

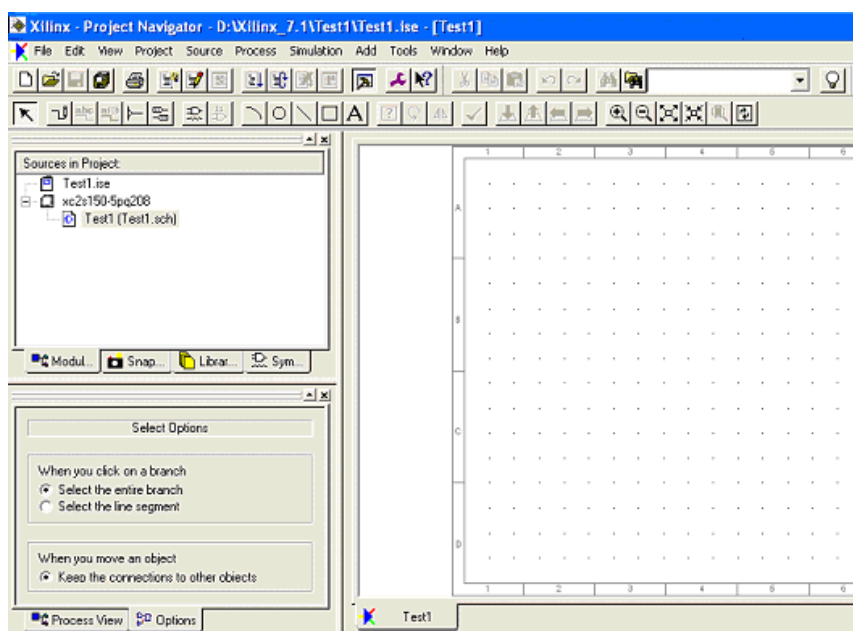
اضافه کردن فایل شماتیک جدید به پروژه

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



شکل ۵-۱۰

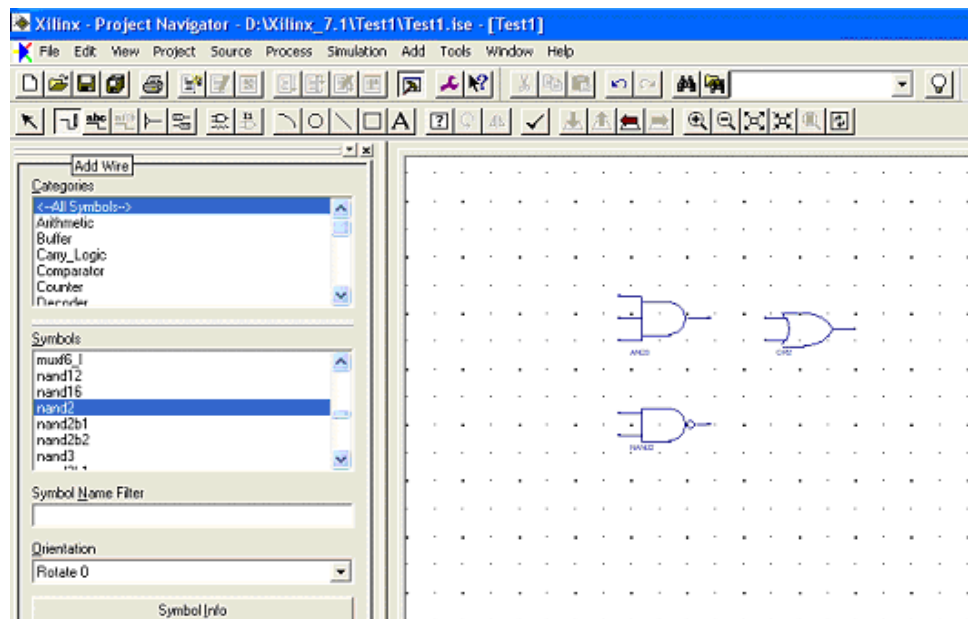
گزینه Next و سپس Finish را بزنید . محیط طراحی شماتیک با نام فایل خودتان ایجاد می گردد. در صورتی که بخواهید محیط کد نویسی وریلاگ برای شما باز شود باید از این قسمت گزینه Verilog Module را انتخاب کنید و باقی مراحل شبیه سازی و پیاده سازی مشابه است.



شکل ۶-۱۰

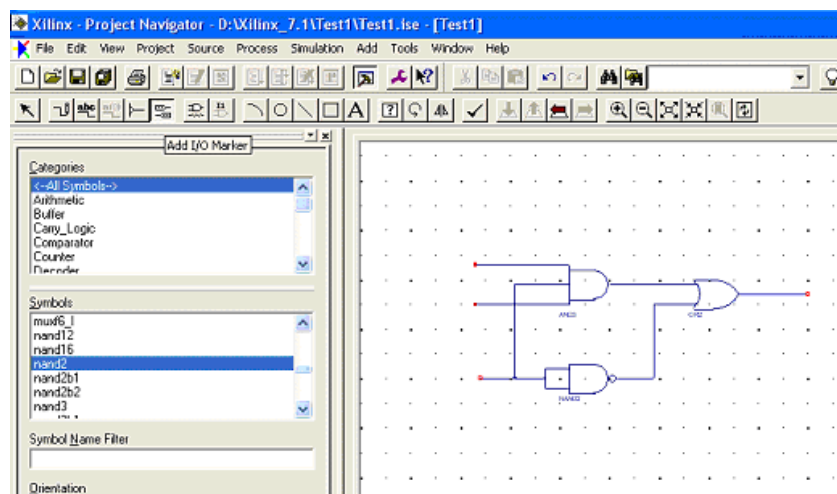
برای انتخاب المان از دکمه Add Symbol  استفاده نمایید. اکنون پنجره شما به ترتیب زیر خواهد بود. سمبل and3 که همان گیت

and با سه ورودی است انتخاب نموده و سپس روی صفحه شماتیک کلیک کنید. سعی کنید شکل زیر را بسازید .



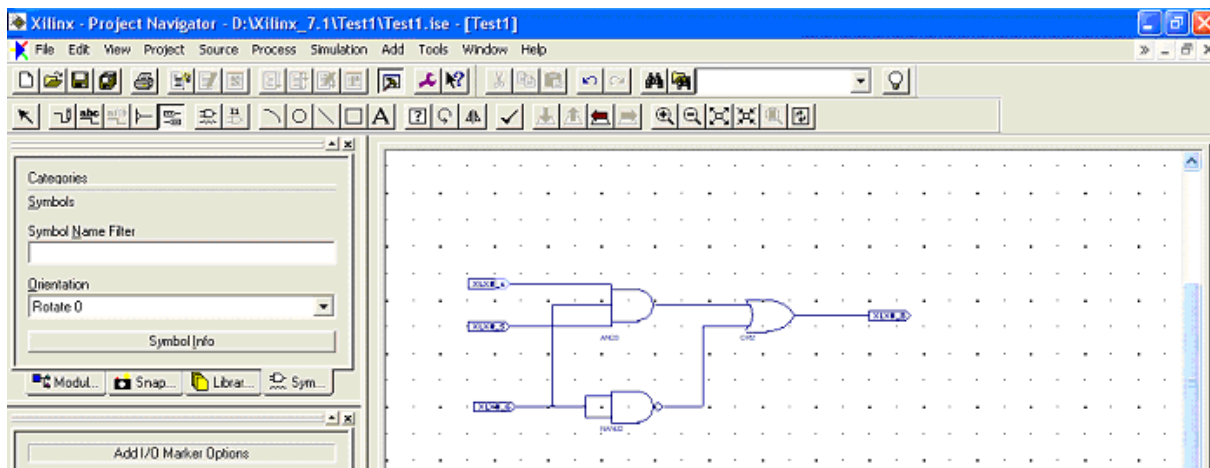
شکل ۷-۱۰

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 به صورت زیر درآمد هاست .



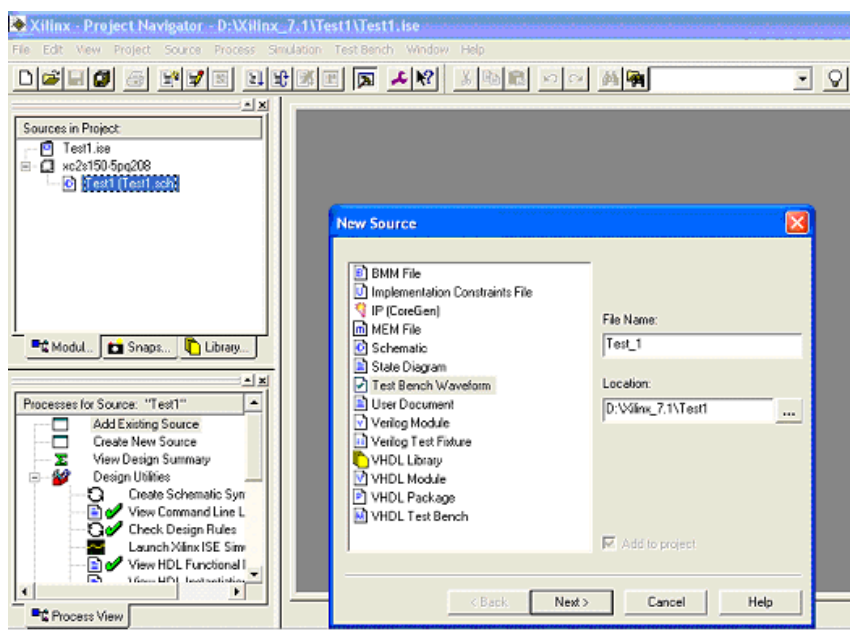
شکل ۸-۱۰

حالا باید سیگنال های ورودی و خروجی را به صورت 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 سیگنال های ورودی کلیک کنید . برای تعریف پورت خروجی نیز گزینه Add an Output Marker را کلیک کنید و روی سیگنال خروجی کلیک کنید . در نهایت پورت ها به صورت شکل زیر به شماتیک اضافه می شوند .



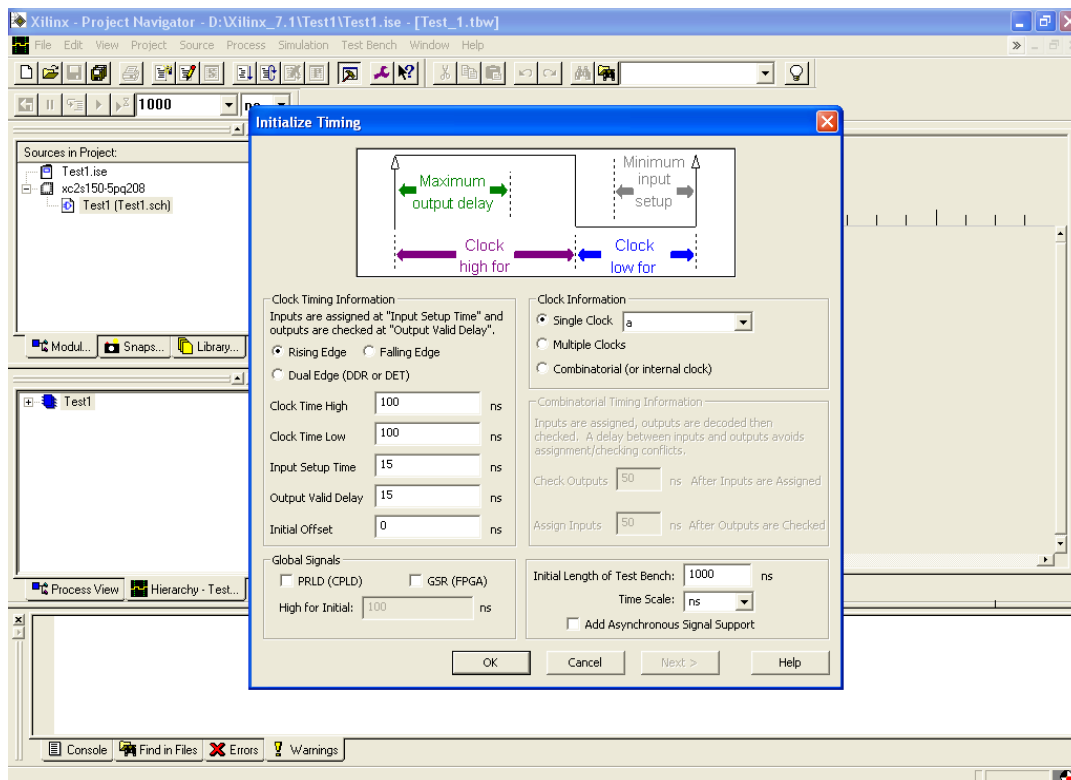
شکل ۱۰-۹

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 با نام Test Bench Waveform را اضافه کنید. این فایل جدید برای شبیه‌سازی فایل اصلی استفاده می‌گردد .



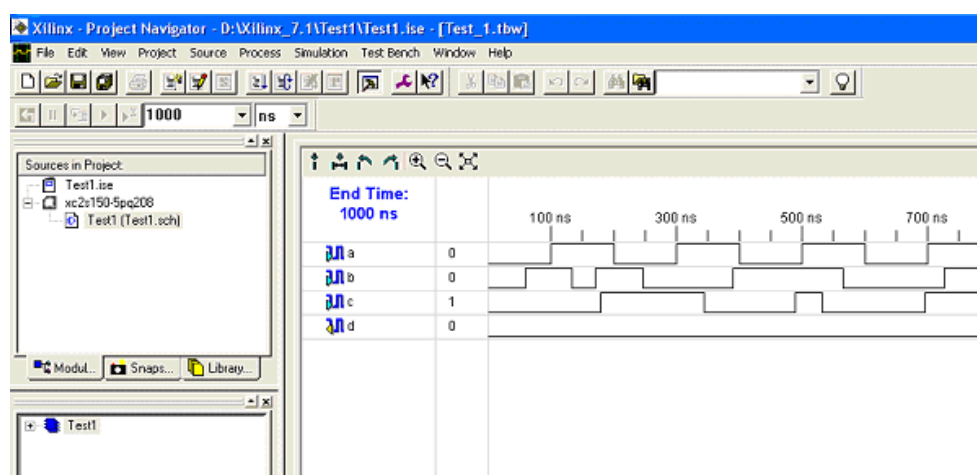
شکل ۱۰-۱۱

سپس محیط تولید شکل موج برای پورت‌های ورودی ظاهر می‌شود .



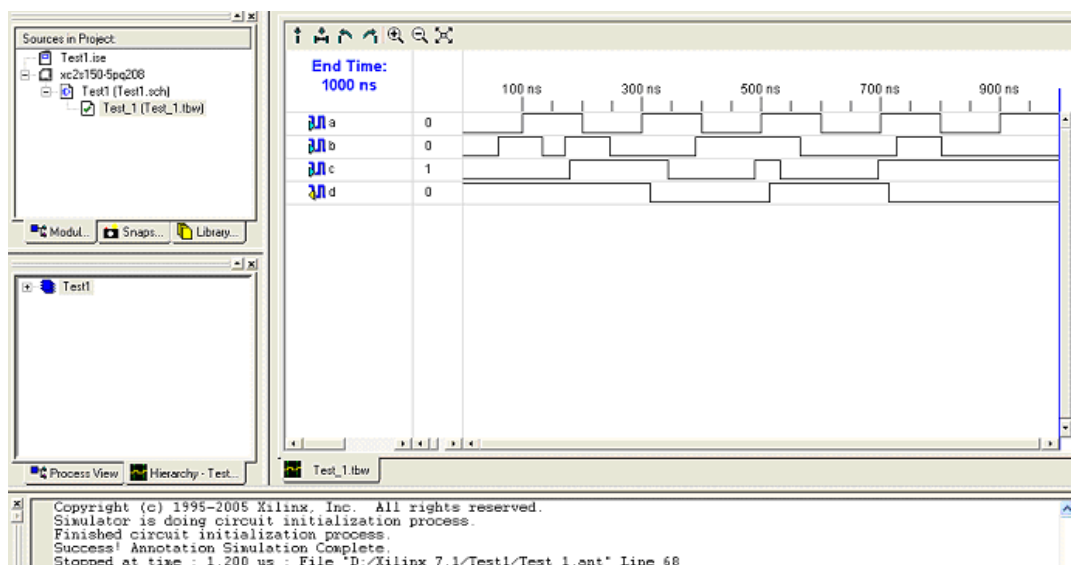
شکل ۱۰-۱۲

گزینه ADD Asynchronous Signal Support را انتخاب نموده و گزینه Next را کلیک نمایید. حالا یک سیگنال را به عنوان 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.



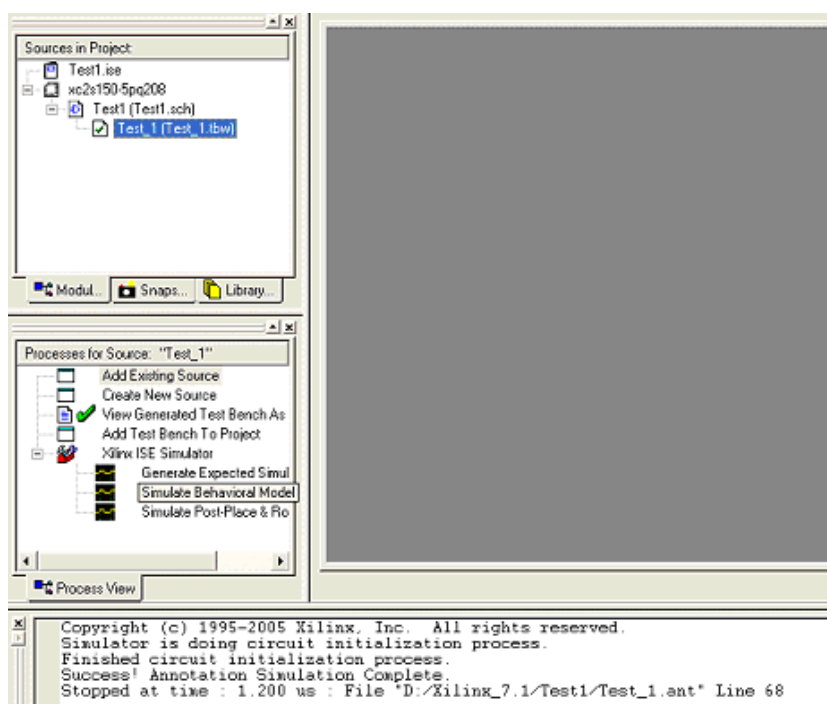
شکل ۱۰-۱۳

این فایل را ذخیره نموده و از این محیط خارج شده 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 دوبار کلیک کنید. پنجره محیط قبلی مجدد باز می شود با این تفاوت که سیگنال خروجی بر اساس شماتیک مقدار یافته است .



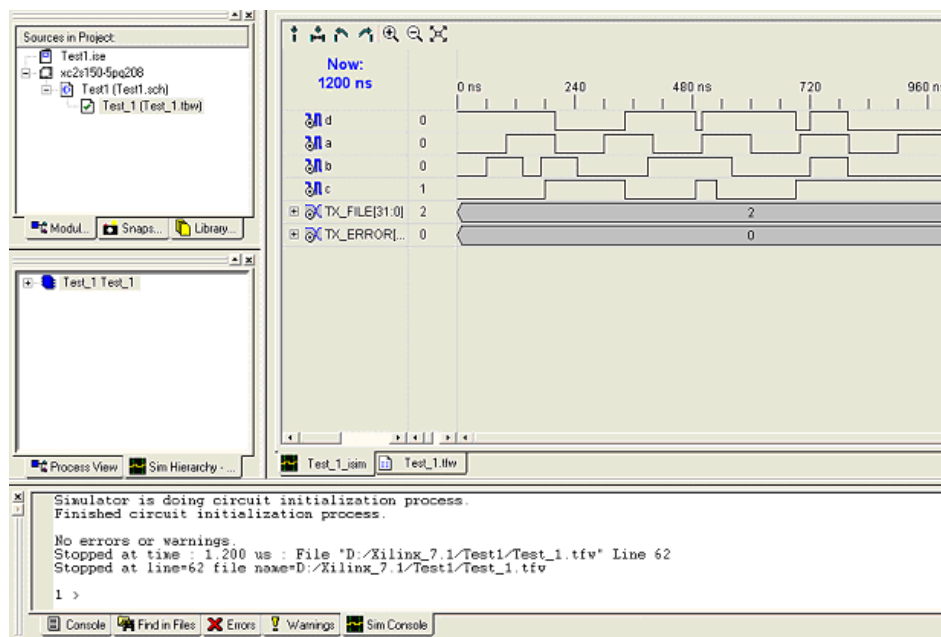
شکل ۱۰-۱۴

فایل را ذخیره نموده و خارج شده 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 دوبار کلیک کنید .



شکل ۱۰-۱۵

پنجره زیر که نتایج شبیه سازی را نشان می دهد، ظاهر می شود.



شکل ۱۰-۱۶

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 با نام 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

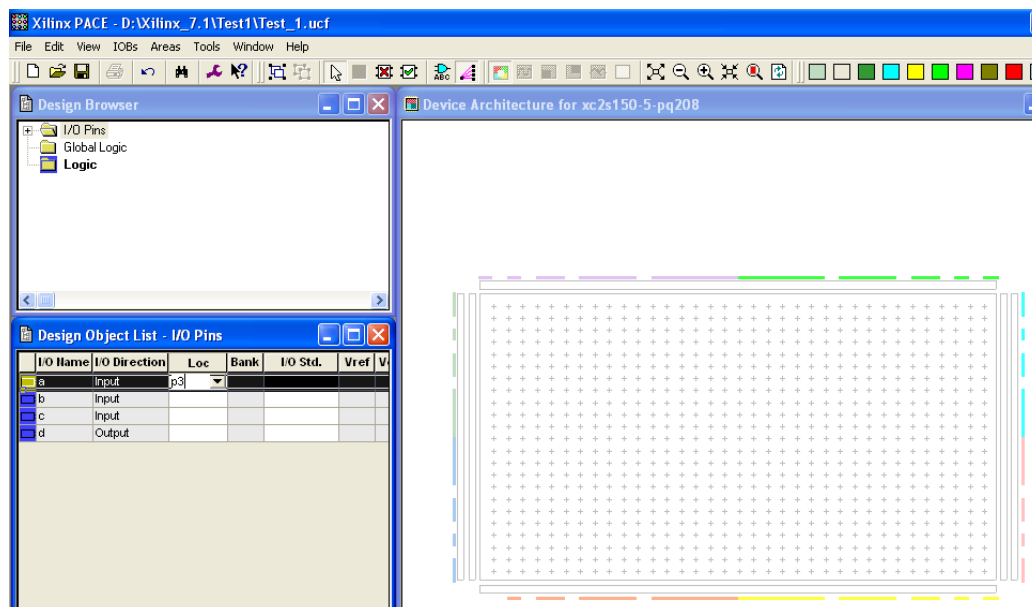
را انتخاب نمایید. پنجره

زیر ظاهر می‌شود که در

حقیقت FPGA را با

پایه‌های آن نشان

می‌دهد.



شکل ۱۰-۱۷

پورت‌های مدار شما نیز در پنجره Design Object List – I/O Pin ظاهر شده‌اند. برای در نظر گرفتن یک پایه 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 به صورت جعلی مقداره‌ی شود تا از پین دیگر یعنی p-18 بتوان به عنوان ورودی استفاده کرد. برای این کار می‌توان یک سیگنال ورودی (تقلبی) تعریف کرد و به پین 204 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 به درستی کار کند.
مثال:

Module test (

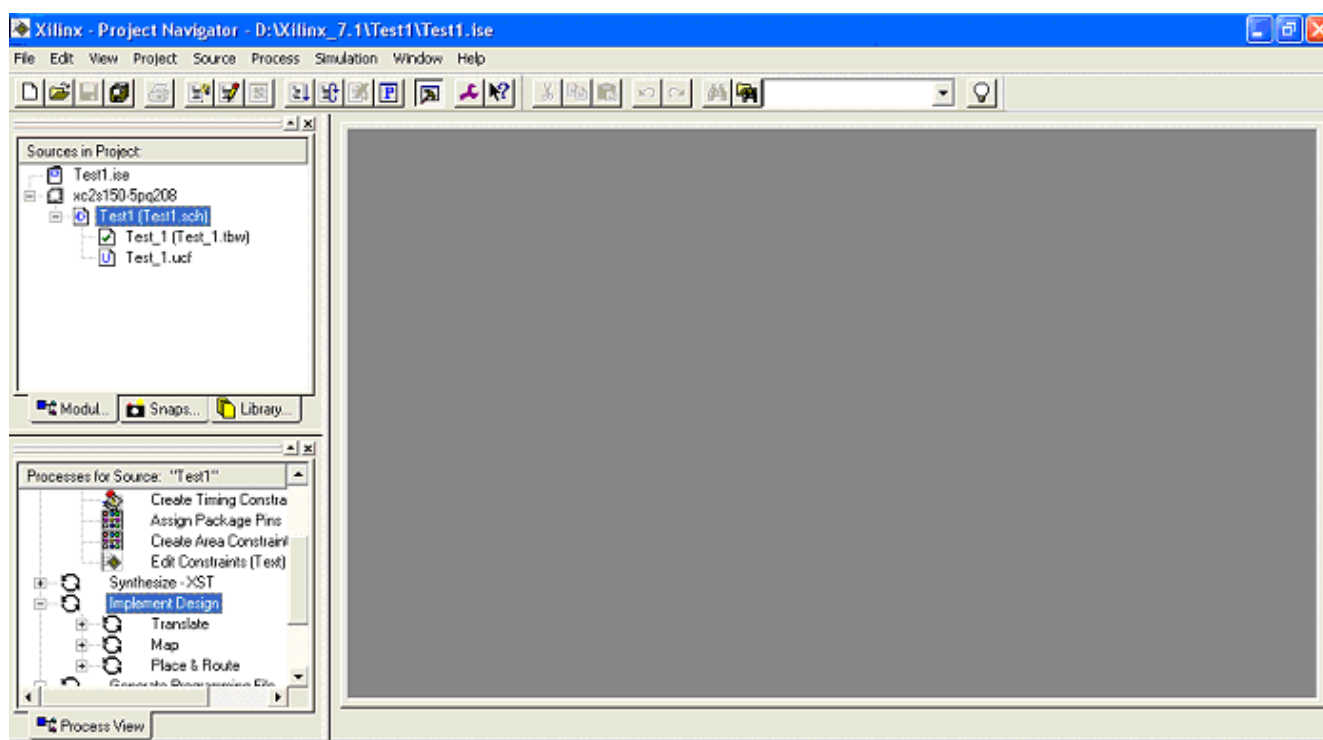
```
...  
input logic Fake,  
...  
);
```

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

هنگامی که 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 ۱۰-۱۸

سپس نرم‌افزار مشغول سنتز و پیاده‌سازی فایل شماتیک شما روی FPGA خواهد شد. بعد از آن برای پروگرام نمودن FPGA روی برد ،

نرم‌افزار باید فایل را با نام *.bit تولید نماید . این کار را با دو بار کلیک کردن روی گزینه Generate Programming File در پنجره

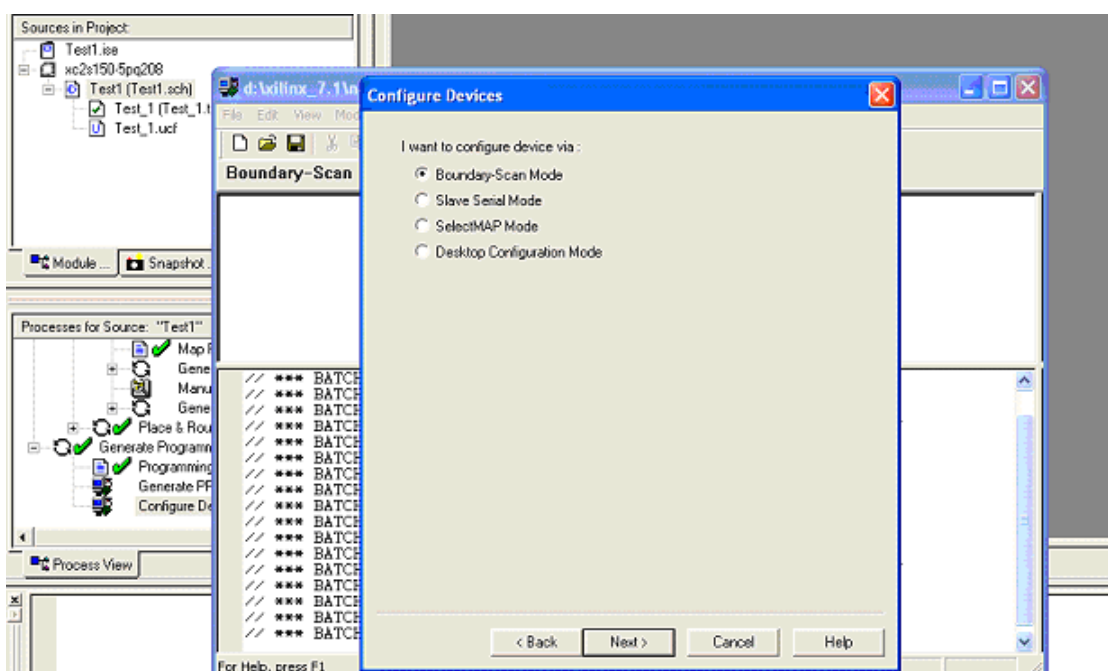
Process view انجام دهید. با کمی دقت، چند جامپر روی برد آموزشی می‌بینید که برای برنامه‌ریزی FPGA باید در وضعیت مای زیر قرار بگیرند:

جامپر J17 در وضعیت 1-2

جامپر J8 در وضعیت 1-2

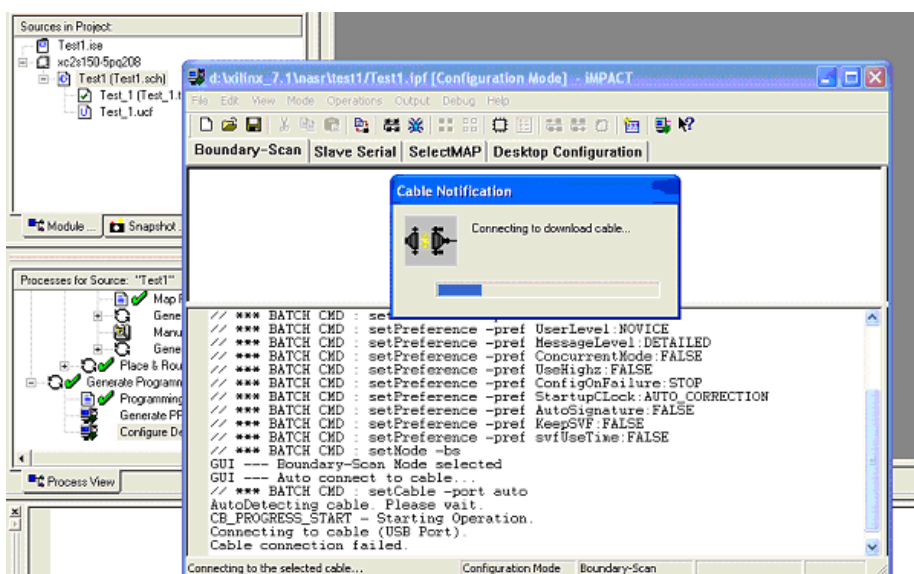
جامپر J6 در وضعیت 3-4 و 5-6

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 دوبار- کلیک کنید. گزینه‌ها را به‌صورت پیش‌فرض پذیرفته و Next را کلیک نمایید.

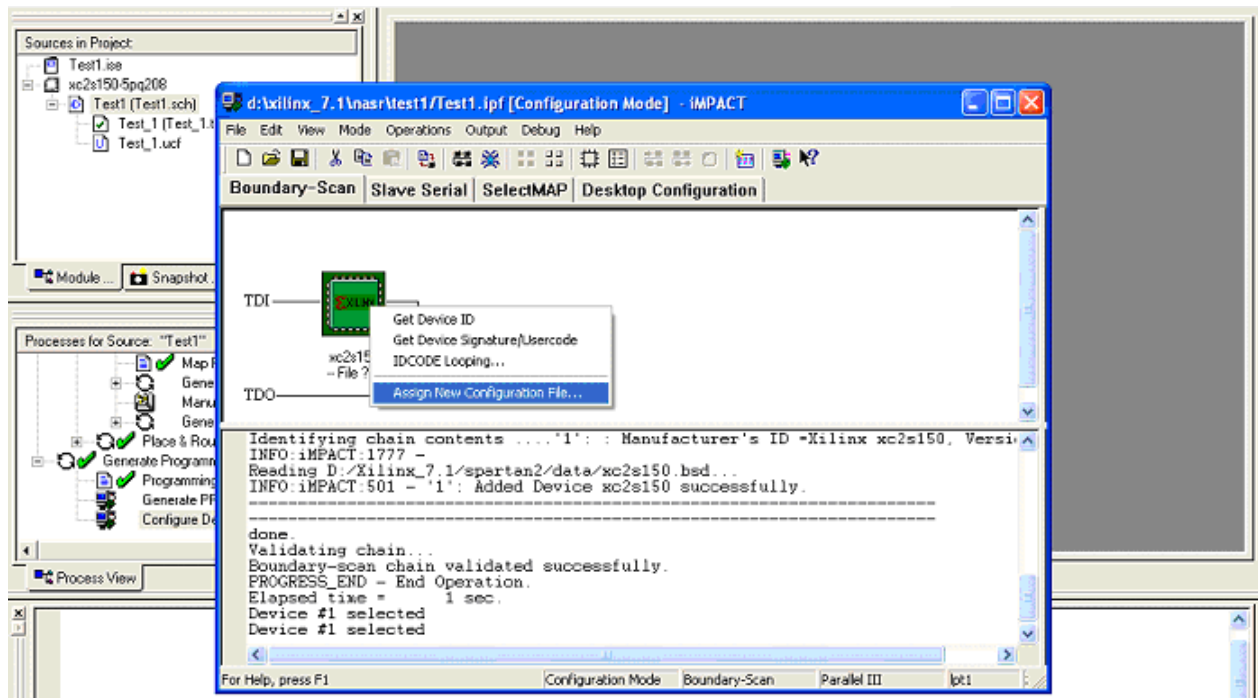


شکل ۱۰-۱۹

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 تولیدشده را باز کنید.

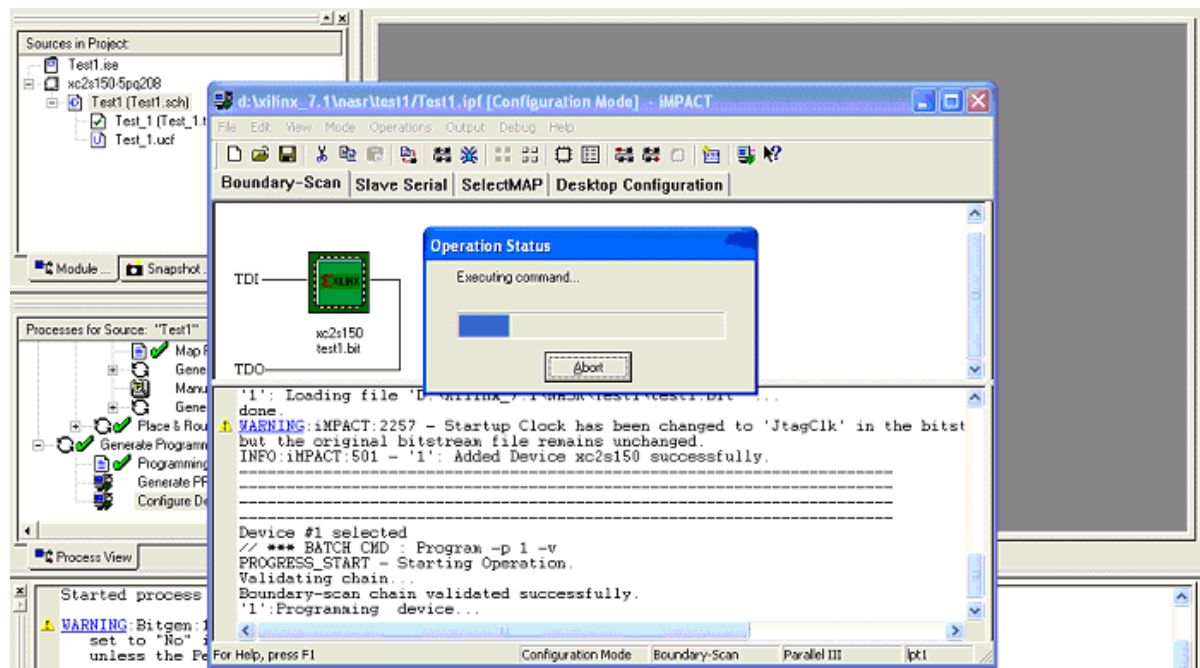


شکل ۲۰-۱۰



شکل ۲۱-۱۰

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 ۲۲-۱۰

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

تکلیف داخل آزمایشگاه: آزمایش گذرگاه داده جلسه ۹ را مجدداً با استفاده از این نرم‌افزار طراحی و بر روی FPGA سری Spartan پیاده‌سازی کنید.

جلسه ۱۱

طراحی حافظه

هدف

در این آزمایش اهداف زیر دنبال می‌شوند:

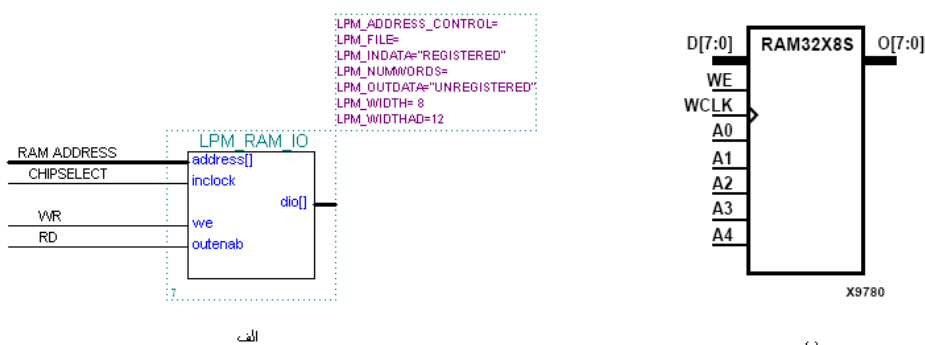
✓ طراحی و پیاده‌سازی حافظه با استفاده و بدون استفاده از IP Core بر روی FPGA های سری اسپارتان

در این جلسه می‌خواهیم حافظه را که متن کلیه برنامه‌ها و داده‌ها داخل آن قرار دارد را به دو روش پیاده‌سازی کنیم

۱- با استفاده از IP Core موجود

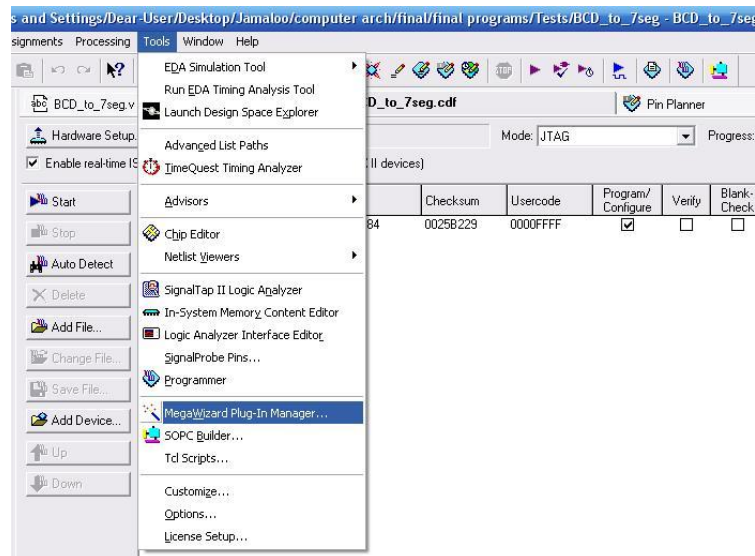
تئوری آزمایش

با استفاده از IP core های موجود در نرم‌افزارهای ISE یا MAX این عمل انجام می‌شود. شرکت ALTERA با معرفی یکسری قطعات عمومی تحت عنوان LPM^{۳۷} امکان توسعه سخت‌افزار را تا حدود بسیار زیادی به کاربر می‌دهد. از جمله در زمینه تعریف حافظه‌ها LPM_RAM_IO است که به راحتی در داخل فایل گرافیکی قابل استفاده می‌باشد. در نرم‌افزار ISE شرکت XILINX نیز نظیر چنین قطعه‌ای به عنوان RAM32X8S در قسمت Memory Categories موجود می‌باشد. شکل ۱۱-۱ نمونه‌هایی از بلوک‌های حافظه مربوط به هر دو شرکت را به نمایش گذاشته است.



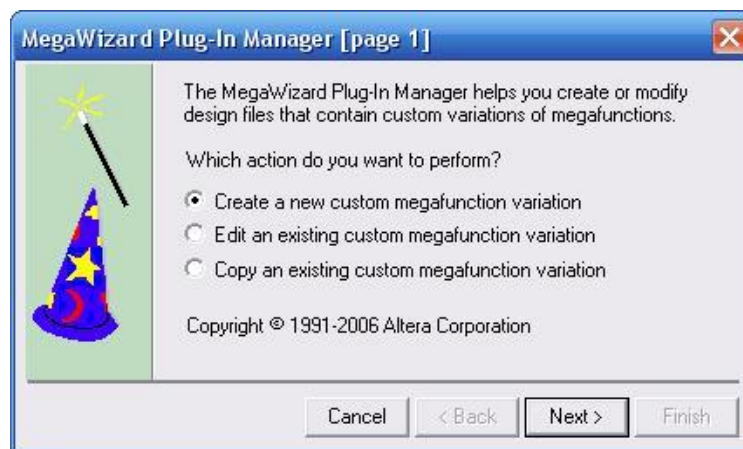
شکل ۱۱-۱- بلوک تعریف حافظه‌ها در نرم‌افزارهای (الف) MUX (ب) ISE

برای طراحی RAM از منوی Tools گزینه MegaWizard Plugin Manager را انتخاب می‌کنیم (شکل ۲-۱۱).



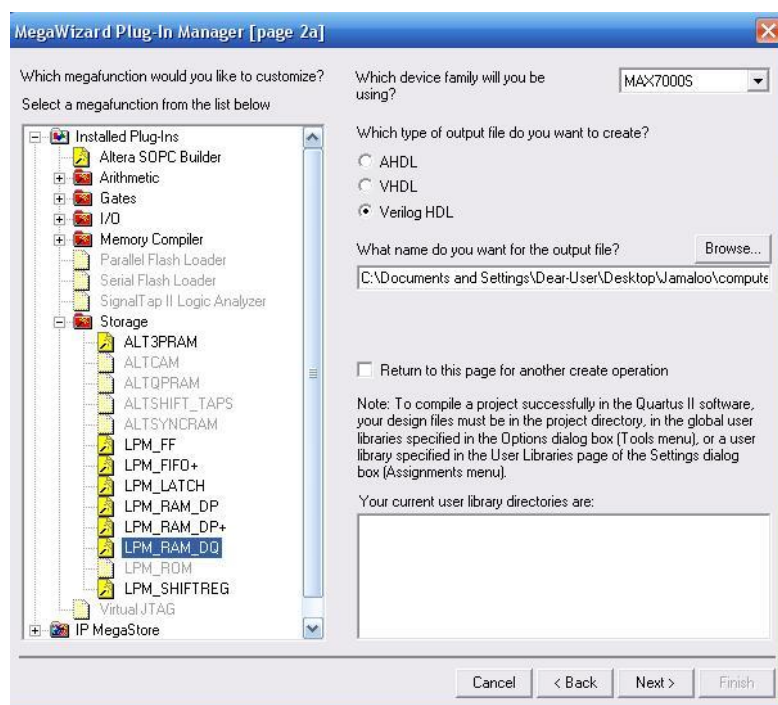
شکل ۲-۱۱

در پنجره شکل ۳-۱۱ گزینه اول را انتخاب می‌کنیم.



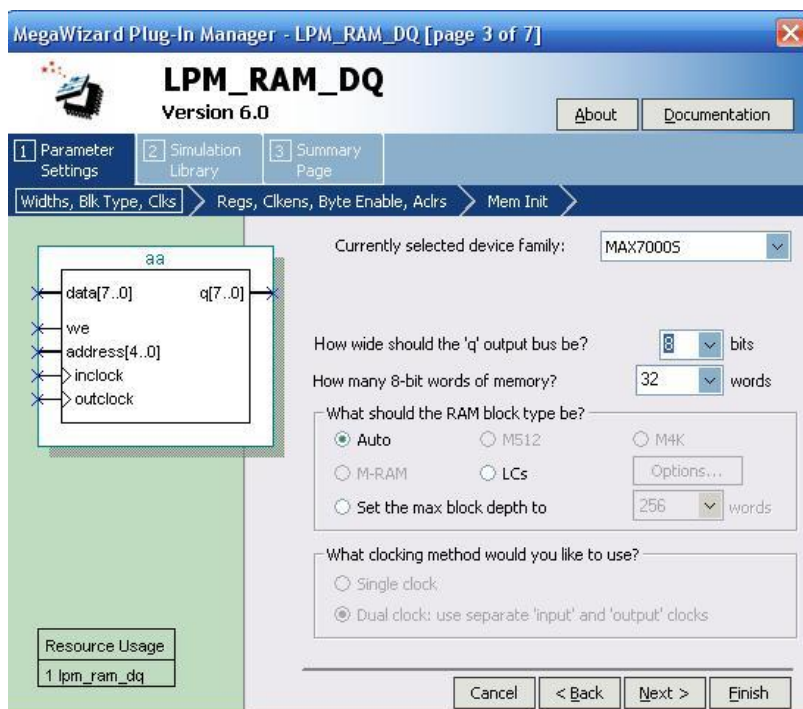
شکل ۳-۱۱

در پنجره شکل ۴-۱۱ خانواده CPLD یا FPGA را تعیین می‌کنیم. همچنین نامی را که می‌خواهیم برنامه با آن ذخیره شود و مسیر آن را تعیین می‌کنیم. از منوی Storage، گزینه LPM_RAM_DQ را انتخاب می‌کنیم.



شکل ۴-۱۱

در مرحله بعد، تعداد بیت‌های آدرس و داده را تعیین و گزینه Finish را انتخاب می‌کنیم.



شکل ۵-۱۱

پس از طراحی ، RAM ابتدا باید به وسیله ارتباط سری در اختیار کار برقرار گیرد تا برنامه موردنظرش را روی آن برنامه ریزی نماید و سپس برای اجرای برنامه باید در اختیار CPU قرار گیرد. برای انجام این عمل، RAM نیاز به یک بیت selector دارد. عمل را بدین ترتیب که، به عنوان مثال ابتدا selector را صفر می کنیم تا از طریق ارتباط سری کاربر برنامه اش را برنامه ریزی نماید و سپس آن را یک می نماییم تا برای اجرای برنامه در اختیار CPU قرار گیرد.

تکالیف داخل آزمایشگاه

- ۱- برنامه ای بنویسید که از Switch select های خارج FPGA داده ای را بخواند و بعد از جمع کردن با نقطه ۱۰۰ حافظه آن را روی نمایشگر هفت قسمتی نمایش دهد.
- ۲- در این آزمایش Switch select به عنوان ورودی و نمایشگر هفت قسمتی به عنوان خروجی می باشد.
- ۳- باید مبدل باینری به نمایشگر هفت قسمتی را به سیستم کامپیوتر پایه اضافه نمایید.

۲- بدون استفاده از IP Core

تئوری آزمایش

در آزمایش قبل، با استفاده از IPcore های موجود در نرم افزار با طراحی حافظه آشنا شده اید. هدف کلی این آزمایش، طراحی یک حافظه بدون استفاده از IPCore ها با امکانات بسیار اولیه و ساده می باشد که بتوان بر روی FPGA پیاده سازی کرد. به وسیله زبان توصیف سخت افزار Verilog مشابه جلسه قبل برنامه ای بنویسید که دو عدد را از حافظه می خواند و سپس آن دو را باهم جمع و در نقطه ای دیگر از حافظه ذخیره می کند بعد از نوشتن برنامه FPGA را برنامه ریزی کنید. بعد از راه اندازی FPGA دوباره حافظه را بخوانید و ببینید که آیا نتایج حاصله درست بوده است یا خیر؟ به همین صورت تست های دیگر را می توانید در روی حافظه ای که طراحی نموده اید انجام دهید.

تکالیف داخل آزمایشگاه

- ۱- بعد از انجام آزمون اولیه کد این جلسه را با کد از پیش آماده آزمایش قبل جایگزین کنید سپس با استفاده از ورودی و خروجی، برنامه ای بنویسید که از Switch select های خارج FPGA داده ای را بخواند و بعد از جمع کردن با نقطه ۱۰۰ حافظه آن را روی نمایشگر هفت قسمتی نمایش دهد.
- ۲- در این آزمایش Switch select به عنوان ورودی و نمایشگر هفت قسمتی به عنوان خروجی می باشد.
- ۳- بدون استفاده از IPcore های موجود در نرم افزار برنامه ای برای حافظه بنویسید، در این حالت دوباره برنامه را اجرا نمایید.

جلسه ۱۲

طراحی واحد کنترل به صورت سخت افزاری

هدف

در این آزمایش اهداف زیر دنبال می شوند :

✓ آشنایی با طراحی واحد کنترل کننده و ساختار آن

✓ طراحی و پیاده سازی واحد کنترل کننده یک CPU

تئوری آزمایش

در آزمایش های قبل با بعضی از قسمت های مهم کامپیوتر پایه آشنا شدید. در این آزمایش، بعد از آشنایی مختصری که با سازمان کامپیوتر پیدا می کنید، به طراحی واحد کنترل کننده خواهیم پرداخت. یک سازمان کامپیوتر شامل ۴ قسمت اصلی می باشد:

(۱) واحد محاسباتی منطقی

(۲) واحد کنترل کننده

(۳) واسطه های ارتباطی

(۴) ثبات ها

قسمت مهم از کامپیوتر پایه در گذشته معرفی و پیاده سازی شده است. در این فصل به واحد کنترل کننده که شامل ساختار سیگنال های کنترلی و زمان بندی های موجود در سیستم است، پرداخته می شود. به منظور طراحی قسمت کنترلی دانستن چند مطلب مورد نیاز است:

(۱) یکی از ثبات های کارا در سازمان کامپیوتر، ثبات دستورالعمل است. همچنان که از معماری کامپیوتر به خاطر دارید یک دستورالعمل در چهار مرحله متوالی انجام می شود. آن مراحل عبارت اند از:

- 1- Fetch
- 2- Decode
- 3- Indirect
- 4- Execute

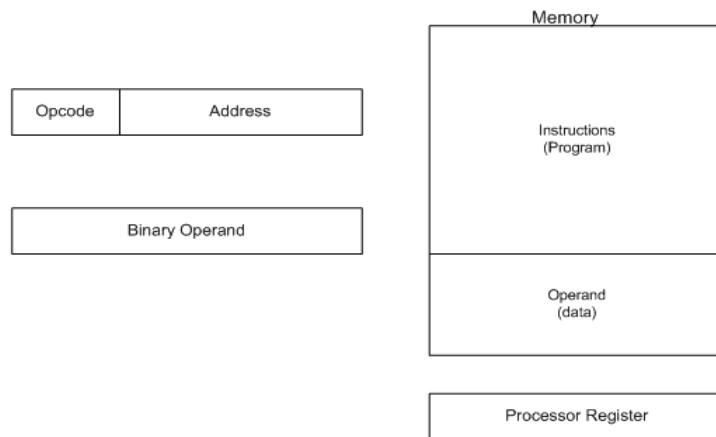
به طور خلاصه، کد باینری هر دستورالعمل، در مرحله Fetch، از حافظه خوانده و وارد ثبات دستورالعمل می شود. در مرحله Decode، این کد توسط واحد کنترل کننده ارزیابی می شود و سیگنال های کنترلی لازم برای انجام دستورالعمل ساخته خواهند شد. مرحله سوم در بعضی دستورالعمل های خاص که شامل آدرس دهی غیرمستقیم هستند، در یک سیکل زمانی اجرا می شود. در نهایت مرحله چهارم عمل منطقی یا ریاضی، ورودی و خروجی، انتقالی یا ثباتی مورد نظر دستورالعمل را انجام می دهد:

کدهای زیر به طور خلاصه دستورالعمل ADD در کامپیوتر پایه را نشان می دهند:

$T_0 : AR \leftarrow PC$	}	Fetch
$T_1 : IR \leftarrow M[AR], PC \leftarrow PC+1$		
$T_2 : D_0 \dots D_7 \leftarrow \text{Decode}[IR(12-14)], AR \leftarrow IR(0-11)$	}	Decode
$T_3 : AR \leftarrow M[AR]$		
$T_4 : DR \leftarrow M[AR]$	}	Execute
$T_5 : AC \leftarrow AC+DR, E \leftarrow \text{Cout}, SC \leftarrow 0$		

(۲) کامپیوتر پایه ای که باید در این ترم طراحی شود، مطابق با کتاب معماری کامپیوتر مانو می باشد.

شکل زیر به طور خلاصه سازمان ذخیره برنامه در کامپیوتر را نمایش می دهد.



شکل ۱-۱۲ سازمان ذخیره برنامه در کامپیوتر

همچنان که از معماری کامپیوتر به یاد دارید، ساختار دستورالعمل در یک ثبات، شامل Address و کد دستورالعمل می باشد. کد باینری دستورالعمل در فضائی از حافظه ذخیره شده است.

برای انجام هر دستورالعمل کد مربوطه از حافظه خوانده و به ثبات دستورالعمل وارد می شود. سیستم بعد از مشخص نمودن نوع Operation در صورت نیاز به عملوند^{۳۸}، آن را از حافظه، بنا به آدرسی که در ساختار دستورالعمل موجود می باشد، می خواند و با ثبات های پردازنده اجرا می نماید.

(۳) به طور کلی سه نوع فرمت برای کامپیوتر پایه در نظر گرفته ایم:

- (a) دستورالعمل هایی که بر اساس حافظه می باشند.
- (b) دستورالعمل هایی که بر اساس ثبات می باشند.
- (c) دستورالعمل هایی که بر اساس ورودی و خروجی می باشند.

I	Opcode	Address
---	--------	---------

الف

0111	Register Operation
------	--------------------

ب

1111	Register Operation
------	--------------------

ج

شکل ۱۲-۲ الف) دستورالعمل‌های بر اساس حافظه ب) بر اساس ثبات ج) بر اساس ورودی و خروجی

اکنون ما با معرفی تعدادی دستورالعمل که در جدول ۱۲-۱ آمده است، به طراحی واحد کنترل می‌پردازیم:

Symbol	I=0	I=1	Description
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	ADD memory word to AC
LDA	2xxx	Axxx	LOAD memory word to AC
STA	3xxx	Bxxx	Store AC in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
CLA	7800		Clear AC
CLE	7400		Clear E bit
CMA	7200		Complement AC
CME	7100		Complement E bit
CIR	7080		Circulate right AC and E bit
CIL	7040		Circulate Left AC and E bit
INC	7020		Increment AC
SPA	7010		Skip next instruction if AC is positive
SNA	7008		Skip next instruction if AC is negative
SZA	7004		Skip next instruction if AC is zero
SZE	7002		Skip next instruction if E bit is zero
HLT	7001		Halt Computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Intrupt on
IOF	F040		Intrupt off

جدول ۱۲-۱ دستورالعمل‌های کامپیوتر پایه

با بیان مقدمه‌های فوق اکنون به معرفی ساختار کنترل کننده می‌پردازیم.

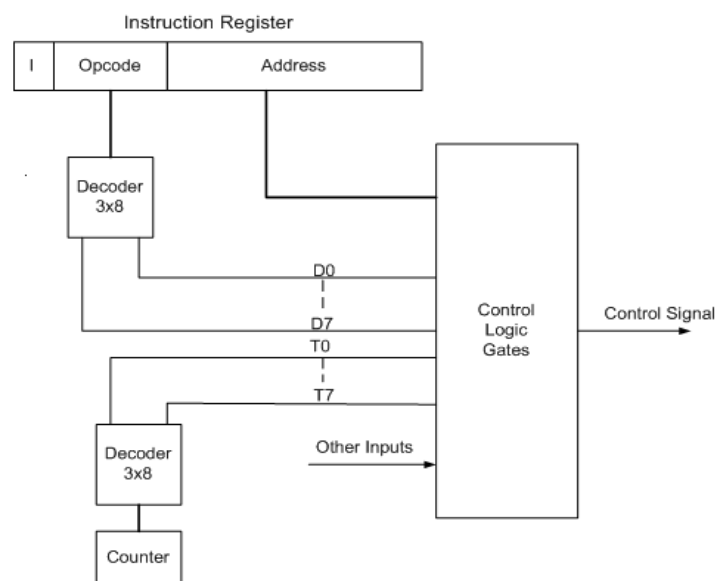
شکل ۱۲-۲ واحد کنترل کننده کامپیوتر پایه را نشان می‌دهد. این واحد از دو قسمت بسیار مهم تشکیل شده است:

۱- واحد زمان بندی

۲- مدارات ترکیبی

کامپیوتر پایه در هر سیکل زمانی تعدادی از ریزدستورالعمل‌ها را بنا به ساختار ALU، BUS و ثبات‌ها انجام می‌دهد. در نتیجه باید یک واحد زمان بندی برای کنترل سیکل‌های اجرایی یک دستورالعمل از ابتدا تا انتها وجود داشته باشد. در کامپیوتری که باید در آزمایشگاه طراحی شود، ماکزیمم سیکل‌های هر دستورالعمل ۸ پریود می‌باشد.

مدارات ترکیبی نیز با استفاده از بیت‌های ثبات دستورالعمل ساخته خواهند شد.

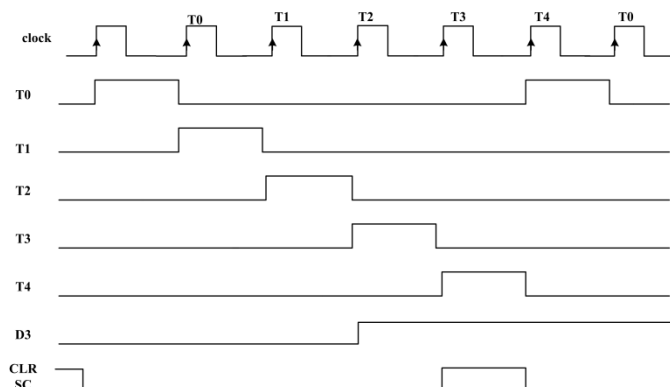


شکل ۱۲-۲ واحد کنترل کننده کامپیوتر پایه

تکالیف پیش از آزمایش

(۱) ابتدا کلیه ریز دستورالعمل‌های یک دستورالعمل، به طور کامل، بنا به سیکل‌های اجرایی آن باید نوشته شوند.

(۲) با استفاده از یک شمارنده و یک دیکدر باید سیگنال‌های T_0 ، T_1 و ... مطابق با شکل زیر ساخته شود:



شکل ۱۲-۳ سیگنال‌های زمان بندی واحد کنترل

۳) بعد از نوشتن تمامی ریز دستورالعمل‌ها، طراحی شمارنده T_0 تا T_7 و دیگر مشخص‌کننده نوع عملگرها با استفاده از بیت‌های ثابت دستورالعمل، اکنون باید کلیه سیگنال‌های کنترلی ثابت‌ها به صورت مدارات ترکیبی پیاده‌سازی گردند.

۴) با طراحی کلیه سیگنال‌های کنترل، اکنون باید واحد کنترل‌کننده را به صورت مجزا آزمودن نمود. این کار با استفاده از ورودی دادن به ثابت دستورالعمل صورت می‌گیرد.

نکته مهم: واحد زمان‌بندی در ساختن بسیاری از سیگنال‌های کنترلی نقش بسیار مهمی را ایفا می‌نماید. به طور مثال در مرحله Fetch، T_0 و T_1 از جمله سیگنال‌های کنترلی برای ساختن سیستم‌های مشخصه BUS می‌باشند.

۵) مرحله آخر ترکیب نمودن BUS و ALU با واحد کنترلی می‌باشد. در این قسمت نیز با استفاده از تحلیل‌گر ALTERA یا XILINX به راحتی تست‌های مختلفی نظیر:

$T_0: AR \leftarrow PC$

را انجام خواهیم داد. هدف از این تست‌ها، آزمایش سیگنال‌های کنترل، انتقال داده از طریق BUS و انجام توابع ریاضی و منطقی به صورت غیرهمزمان می‌باشد. بنابراین شما به راحتی می‌توانید به جای حافظه فقط یک ثابت تعریف نمایید.

تکالیف داخل آزمایشگاه

در این آزمایش شما باید در Simulator، یک محیط آزمودن برای برنامه‌نمایی حاصل از ترکیب BUS و ALU با واحد کنترلی با کد verilog ایجاد و آن را روی برد پیاده‌سازی نمایید. با استفاده از سویچ مای موجود بر روی برد و دستورالعمل‌های جدول ۱۲-۱ ورودی ثابت دستورالعمل را تعیین و نتایج حاصل را روی نمایشگرهای هفت‌قسمتی مشاهده نمایید.

طراحی کامپیوتر پایه

هدف

در این آزمایش اهداف زیر دنبال می‌شوند :

- ✓ طراحی و پیاده‌سازی کامپیوتر پایه
- ✓ آزمون کامپیوتر پایه با زبان ماشین
- ✓ طراحی گذرگاه داده بر روی سری اسپارتمان

تئوری آزمایش

در آزمایش‌های گذشته، با طراحی کلیه بلوک‌های کامپیوتر پایه به‌طور مجزا آشنا شده‌اید. هدف کلی این آزمایشگاه و به‌طور مخصوص این آزمایش، طراحی یک CPU با امکانات بسیار اولیه و ساده می‌باشد. همچنان که در آزمایش‌های گذشته ذکر شد؛ یک سیستم کامپیوتر پایه از ۴ قسمت ALU, BUS, Control unit و ثبات‌ها تشکیل شده است. تمامی بلوک‌های مهم در آزمایش‌های قبل پیاده‌سازی گردیده‌اند. در این آزمایش باید همه بلوک‌ها و قسمت حافظه به‌صورت یک مدار مجتمع بر روی FPGA پیاده‌سازی گردند. برای انجام این عمل سه مقدمه موردنیاز است:

(۱) تعیین نحوه ارتباط بلوک‌های مختلف در FPGA های XILINX یا ALTERA.

(۲) برقراری ارتباط بلوک‌های کامپیوتر پایه با استفاده از زبان توصیف سخت‌افزار Verilog

تکالیف داخل آزمایشگاه

- ۱- بعد از یکپارچه کردن کامپیوتر پایه و انجام آزمون اولیه، اکنون با استفاده از ورودی و خروجی، برنامه‌ای بنویسید که از Switch select های خارج FPGA داده‌ای را بخواند و بعد از جمع‌کردن با نقطه ۱۰۰ حافظه آن را روی نمایشگر هفت‌قسمتی نمایش دهد.
- ۲- در این آزمایش Switch select به‌عنوان ورودی و نمایشگر هفت‌قسمتی به‌عنوان خروجی می‌باشد.
- ۳- باید مبدل باینری به نمایشگر هفت‌قسمتی را به سیستم کامپیوتر پایه اضافه نمایید.

طراحی واحد کنترل به صورت نرم‌افزاری (کنترل ریزبرنامه‌ریزی^{۳۹})هدف

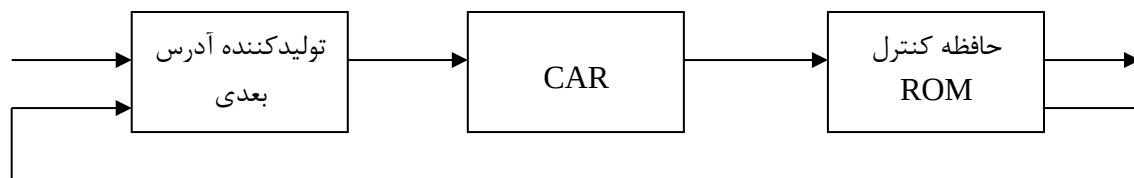
در این آزمایش اهداف زیر دنبال می‌شوند :

✓ آشنایی با طراحی واحد کنترل ریزبرنامه‌ریزی و ساختار آن

✓ طراحی و پیاده‌سازی میکروپروگرام

تئوری آزمایش

در آزمایش‌های قبل با طراحی قسمت‌های مختلف کامپیوتر از جمله واحد کنترل آشنا شدید. واحد کنترل که در آزمایش‌های قبل طراحی نمودید ، کنترل سخت‌افزاری بود. راه دیگر برای طراحی واحد کنترل استفاده از کنترل ریزبرنامه‌ریزی می‌باشد. ریزبرنامه‌ریزی یک روش تقریباً نرم‌افزاری برای کنترل و اجرای ریز عملیات در کامپیوتر است. یکی از محاسن کنترل ریزبرنامه‌ریزی این است که اگر نیاز به تغییر ریز دستورات باشد، دیگر نیازی به تغییر سخت‌افزاری نیست ، بلکه کافی است دستورات جزئی در حافظه کنترل را تغییر دهیم. ساختار کلی یک واحد کنترل ریزبرنامه‌ریزی در شکل زیر نشان داده شده است.



شکل ۱۴-۱

تولیدکننده آدرس بعدی همان‌طور که از نام آن مشخص است وظیفه تعیین آدرس بعدی را در کنترل ریزبرنامه‌ریزی بر عهده دارد. به‌طور

خلاصه روش‌های تولید آدرس در این بخش عبارت‌اند از :

۱- افزایش ثبات آدرس کنترل به میزان یک واحد.

۲- اجرای انشعاب شرطی یا غیرشرطی.

۳- نگاشت بیت‌های دستور کامپیوتر به آدرس روتین نظیر در حافظه کنترل.

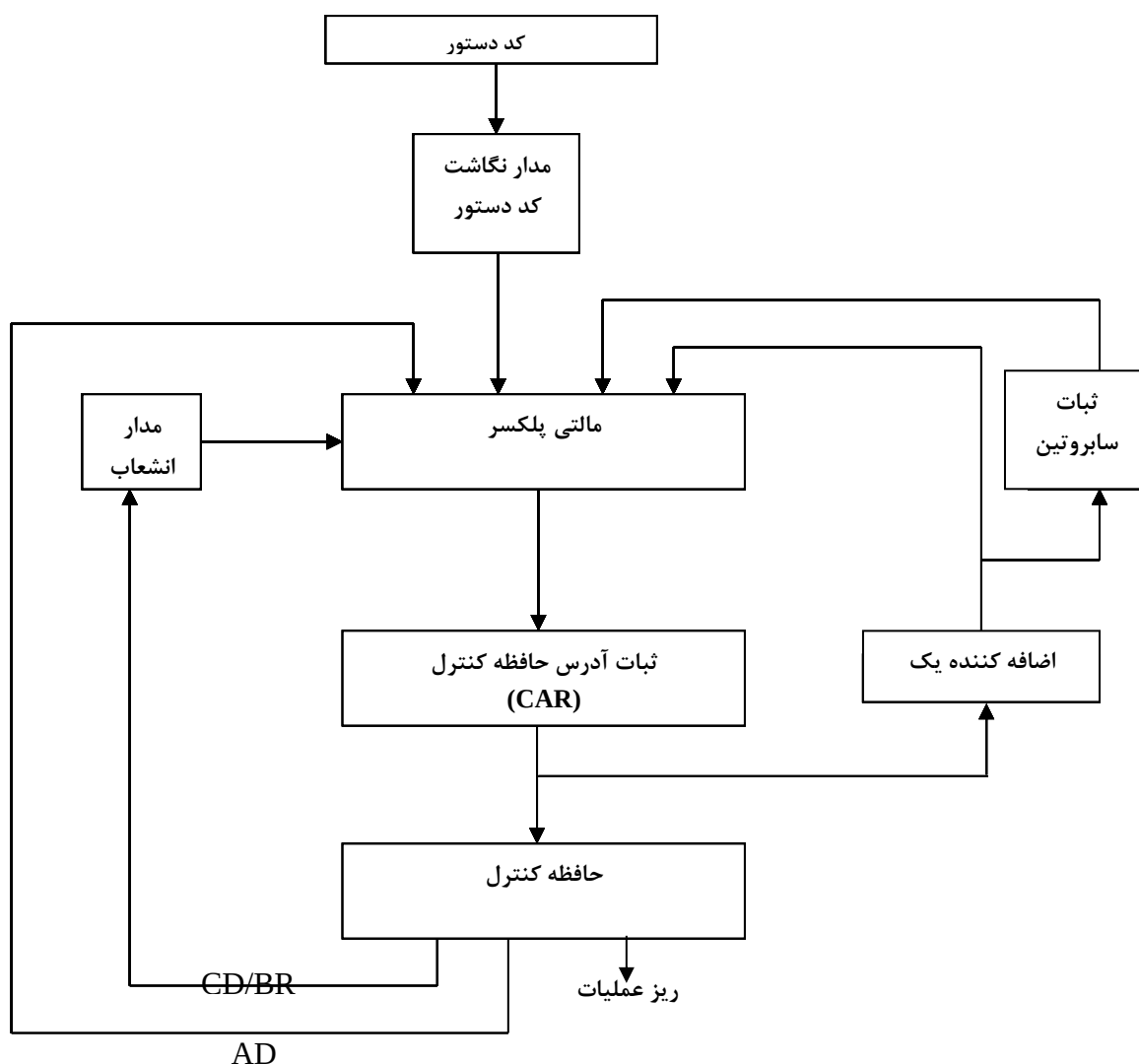
۴- تسهیلاتی برای اجرای سابروتین و برگشت از سابروتین.

ثبات $CAR^{۴۰}$ آدرسی را که از بخش تولیدکننده آدرس بعدی می‌آید را به حافظه کنترل انتقال می‌دهد. میکروپروگرام از تعدادی ریز

دستورات تشکیل می‌شود که در حافظه کنترل ذخیره می‌گردند. هر دستور کامپیوتر دارای روتین میکروپروگرام خود در حافظه کنترل

می‌باشد ، که عملیات جزئی برای اجرای دستور مربوطه را تولید می‌کند. بنابراین برای اجرای هر دستور یا به عبارت دیگر برای هر کد دستور ، کامپیوتر می‌بایست روتین میکروپروگرام نظیر را در واحد کنترل اجرا نماید. تبدیل بیت‌های کد اجرای دستور کامپیوتر به آدرس شروع میکروپروگرام نظیر آن در حافظه کنترل، نگاشت کردن نام دارد.

در یکی دیگر از روش‌های تولید آدرس می‌بایست امکان اجرای سابروتین و برگشت از آن فراهم باشد. سابروتین برنامه‌ای است که برای کار خاصی نوشته می‌شود و از هرکجای برنامه اصلی میکروپروگرام می‌تواند فراخوانی شود. بلوک دیاگرام انتخاب آدرس برای حافظه کنترل در شکل زیر نشان داده شده است.



شکل ۱۴-۲ انتخاب آدرس برای حافظه کنترل فرمت ریز دستورات حافظه کنترل به سه قسمت اساسی تقسیم می‌شود: قسمت ریز عملیات ، ریز عملیات کامپیوتر را مشخص می‌نماید. AD حافظه کنترل را آدرس‌دهی می‌کند و بالاخره CD/BR شرایط بیت‌های پرچم و نوع انشعاب را تعیین می‌کند.

برای طراحی واحد کنترل میکروپروگرام ابتدا باید یک ROM برای حافظه کنترل تعریف نمایید. بدین منظور از منوی Tools گزینه Megawizard Plugin Manager را انتخاب نمایید و همان‌طور که در بخش‌های قبل توضیح داده شد تنظیمات ROM را انجام دهید. تنها تفاوت موجود این است که در آزمایش قبل یک RAM طراحی کردید، ولی این بار می‌بایست یک ROM تعریف نمایید. برای تعریف ROM می‌بایست محتویات آن را از قبل در یک فایل با پسوند .mif و یا .hex آماده نمایید تا در موقع تنظیمات، آدرس آن فایل را به‌عنوان محتویات ROM بدهید.

تکالیف داخل آزمایشگاه

- (۱) ابتدا کلیه ریز دستورالعمل‌های یک دستورالعمل به‌طور کامل بنا به سیکل‌های اجرایی آن باید نوشته شود.
- (۲) بعد از نوشتن ریز دستورالعمل‌ها حافظه ROM با آن‌ها باید پر شود.
- (۳) سپس ریز عملیات خروجی از حافظه کنترل باید دیکد شود و گیت‌های منطقی برای ساختن سیگنال‌های کنترلی طراحی شوند. با توجه به مراحل گفته‌شده در بالا یک واحد کنترل ریز برنامه‌نویسی طراحی نمایید.

آزمایش توسعه دستورات

هدف

در این آزمایش اهداف زیر دنبال می‌شوند :

✓ آشنایی با پیاده‌سازی دستورات دستیابی حافظه به آدرس‌دهی غیرمستقیم

✓ آشنایی با پیاده‌سازی دستورات وقفه

✓

۱- مدهای آدرس‌دهی

تئوری آزمایش

برای شناخت انواع روش مای آدرس‌دهی بهتر است ابتدا بدانیم که یک دستورالعمل در چهار مرحله متوالی انجام می‌شود. آن مراحل عبارت‌اند از:

- 5- Fetch
- 6- Decode
- 7- Indirect
- 8- Execute

به‌طور خلاصه، کد باینری هر دستورالعمل، در مرحله Fetch، از حافظه خوانده و وارد ثبات دستورالعمل می‌شود. در مرحله Decode، این کد توسط واحد کنترل‌کننده ارزیابی می‌شود و سیگنال‌های کنترلی لازم برای انجام دستورالعمل ساخته خواهند شد. مرحله سوم در بعضی دستورالعمل‌های خاص که شامل آدرس‌دهی غیرمستقیم هستند، در یک سیکل زمانی اجرا می‌شود. در نهایت مرحله چهارم عمل منطقی یا ریاضی، ورودی و خروجی، انتقالی یا ثباتی موردنظر دستورالعمل را انجام می‌دهد:

کدهای زیر به‌طور خلاصه دستورالعمل ADD در کامپیوتر پایه را نشان می‌دهند:

$T_0 : AR \leftarrow PC$	}	Fetch
$T_1 : IR \leftarrow M[AR], PC \leftarrow PC+1$		
$T_2 : D_0 \dots D_7 \leftarrow \text{Decode}[IR(12-14)], AR \leftarrow IR(0-11)$	}	Decode
$T_3 : AR \leftarrow M[AR]$		
$T_4 : DR \leftarrow M[AR]$	}	Indirect
$T_5 : AC \leftarrow AC+DR, E \leftarrow \text{Cout}, SC \leftarrow 0$		
	}	Execute

انواع مدهای آدرس دهی عبارت اند از:

- روش ضمنی
- روش بلا فصل
- روش ثباتی
- روش غیرمستقیم ثباتی
- روش خود افزایشی یا خودکاهشی
- روش مستقیم
- روش غیرمستقیم
- روش نسبی
- روش شاخص دار
- روش آدرس دهی با ثبات پایه

در شکل زیر تفاوت بین آدرس دهی ها را نشان می دهیم. دستور دو کلمه ای واقع در آدرس های ۲۰۰ و ۲۰۱ یک دستور Load AC با آدرس ۵۰۰ می باشد. اولین کلمه دستور العمل، کد عمل و روش آدرس دهی و دومین کلمه آدرس می باشد. مقدار PC عدد ۲۰۰ برای دریافت این دستور است. محتوای ثبات پردازشگر R1 برابر ۴۰۰ و محتوای ثبات اندیس XR نیز ۱۰۰ است.

شکل ۱-۱۵ مثال عددی برای شیوه های آدرس دهی

PC=200	حافظه		آدرس	آدرس دهی	آدرس مؤثر	محتوای AC	
R1=400	۲۰۰	Load AC	Mode	مستقیم	۵۰۰	۸۰۰	
XR=100	۲۰۱	آدرس=۵۰۰			۲۰۱	۵۰۰	
AC	۲۰۲	دستور العمل بعدی			۸۰۰	۳۰۰	
					نسبی	۷۰۲	۳۲۵
	۳۹۹	۴۵۰			شاخص دار	۶۰۰	۹۰۰
	۴۰۰	۷۰۰			ثباتی	-	۴۰۰
	۵۰۰	۸۰۰			ثباتی	۴۰۰	۷۰۰
	۶۰۰	۹۰۰			غیرمستقیم	۴۰۰	۷۰۰
	۷۰۲	۳۲۵			خودافزایشی	۴۰۰	۷۰۰
	۸۰۰	۳۰۰			خودکاهش	۳۹۹	۴۵۰

در این جلسه قصد شبیه سازی مدهای مختلف آدرس دهی و پیاده سازی مد آدرس دهی غیرمستقیم بر روی FPGA را داریم. همان طور که از معماری کامپیوتر می دانید و در مثال بالا نشان دادیم آدرس مؤثر در آدرس دهی غیرمستقیم: مجموعه محتوای ثبات در CPU و بخش آدرس دستور است.

تکالیف داخل آزمایشگاه

- ۱) برای دستور جمع تمام مدهای آدرس‌دهی را در قالب کد نویسی Verilog پیاده‌سازی کنید.
- ۲) قابلیت آدرس‌دهی غیرمستقیم را به کامپیوتر پایه که در جلسات گذشته طراحی کرده‌اید بی‌افزایید.

۲- وقفه

تئوری آزمایش

ایده وقفه برخورد با مسائلی است که دنباله عادی برنامه را برهم می‌زند. وقفه به انتقال کنترل برنامه از برنامه در حال اجرای جاری به یک برنامه دیگر گفته می‌شود. پس از اجرای برنامه سرویس وقفه، کنترل به برنامه عادی بازمی‌گردد. پروسه وقفه تقریباً مشابه صدا کردن زیرروال است ولی سه تفاوت دارد: ۱- یک وقفه معمولاً توسط سیگنال داخلی یا خارجی رخ می‌دهد نه با اجرای یک دستور (به جز وقفه مای نرم‌افزاری) ۲- آدرس برنامه سرویس‌دهی به وقفه، سخت‌افزاری مشخص می‌شود نه از روی فیلد آدرس و دستور ۳- یک پروسه وقفه معمولاً تمام اطلاعات لازم برای تعیین حالت CPU (مثل بیت مای وضعیت) را ذخیره می‌کند ولی صدا کردن زیر روال فقط PC را ذخیره می‌کند.

اگر وقفه در حین اجرای دستور رخ دهد اجرای دستور خاتمه می‌یابد و پس از اجرا، پردازنده چک می‌کند آیا وقفه رخ داده است یا خیر که در صورت وقوع وقفه، روتین پاسخ‌دهی به وقفه اجرا می‌شود.

انواع وقفه:

- ۱- وقفه مای خارجی: از وسایل ورودی/خروجی، از یک وسیله زمان‌بندی یا از هر منبع خارجی دیگری می‌آیند. (آسنکرون)
- ۲- وقفه مای داخلی: به علت استفاده ناصحیح از دستورات یا داده‌ها رخ می‌دهد مانند تقسیم بر صفر (سنکرون با برنامه)
- ۳- وقفه مای نرم‌افزاری: توسط اجرای یک دستور ایجاد می‌شود. یک وقفه نرم‌افزاری یک صدا کردن بخصوص زیرروال است که شبیه یک وقفه عمل می‌کند.

تکالیف داخل آزمایشگاه

- ۱) به طور مجزا برای هر سه نوع وقفه برنامه‌ای به زبان Verilog بنویسید و روتین وقفه به صورتی باشد که کاربر متوجه قطع عملیات و اجرای روتین وقفه بشود.
- ۲) برنامه‌ای بنویسید که شامل انواع وقفه باشد و پس از شبیه‌سازی بر روی FPGA سری اسپارitan پیاده‌سازی کنید.
- ۳) سیکل وقفه را به کامپیوتر پایه که در جلسات گذشته طراحی کرده‌اید بی‌افزایید.

آزمایش حافظه خارجی و پورت سریال

دانشجویان سخت‌افزار و برق این آزمایش را انجام دهند.

۱- حافظه خارجی

هدف

در این آزمایش اهداف زیر دنبال می‌شوند :

✓ آشنایی با حافظه استاتیک

✓ بررسی نحوه ارتباط FPGA و حافظه خارجی

تئوری آزمایش

در برخی موارد، نیاز به استفاده از یک حافظه بزرگ می‌باشد، به‌طوری‌که RAMهای داخلی FPGA نیز کافی نخواهند بود. استفاده از یک حافظه خارجی، تنها راه‌حل ممکن می‌باشد. در RAM خارجی، مشابه انواع داخلی آن، ورودی مای کنترلی write و enable، هر دو عمل خواندن و نوشتن را انجام می‌دهند: وقتی $we=0$ باشد، مقدار دیتا باس در قسمت آدرس داده‌شده حافظه نوشته می‌شود. و هنگامی که $we=1$ و $oe=0$ باشد، محتویات آن قسمت حافظه، روی دیتا باس قرار خواهد گرفت. خط کنترلی Chip-Select نیز جهت فعال کردن خواندن از RAM یا نوشتن بر RAM به کار گرفته می‌شود. این باعث می‌شود که مقادیر در RAM بار شود و سپس FPGA را، بدون نگرانی از invalid شدن RAM، برنامه‌ریزی کنید.

تکالیف داخل آزمایشگاه

برنامه کامپیوتر پایه‌ای را که در آزمایش ۱۱ با استفاده از RAM داخلی نوشته‌اید، برای حافظه خارجی بازنویسی کنید و آن را آزمودن نمایید.

جدول ۱-۱۶ اتصالات پین مای FPGA و RAM استاتیک 128KB را نشان می‌دهد.

Pin Function	K6R1008V1C Pin
A0	24
A1	25
A2	26
A3	27
A4	38
A5	39
A6	40
A7	41
A8	63
A9	62
A10	61
A11	60
A12	58
A13	47
A14	46
A15	45
A16	44
I/O1	29
I/O2	30
I/O3	31
I/O4	36
I/O5	57
I/O6	56
I/O7	55
I/O8	54
/CS	28
/OE	53
/WE	37

جدول ١٦-١

۲- پورت سریال

هدف

در این آزمایش اهداف زیر دنبال می‌شوند :

✓ آشنایی با پورت سریال

✓ بررسی نحوه ارتباط سریال FPGA از طریق پورت RS232 و تبادل اطلاعات بین این دو

تئوری آزمایش

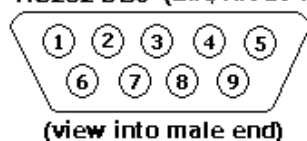
ارسال و دریافت اطلاعات باینری به صورت بیت به بیت را انتقال سریال و پورت مورد استفاده برای این عمل را پورت سریال گویند. RS-232 یکی از استانداردهای پر کاربرد در کامپیوترهای شخصی و کاربردهای صنعتی است. این استاندارد هم ارتباط سریال سنکرون و هم آسنکرون را پشتیبانی کرده و به صورت Full Duplex عمل می‌نماید. کامپیوترهای شخصی تنها ارتباط آسنکرون را پشتیبانی می‌کنند و از طریق چیپ UART موجود در برد اصلی، اطلاعات را از حالت موازی به سریال و یا بالعکس تبدیل کرده و با تنظیمات زمانی آن را از طریق پورت سریال ارسال یا دریافت می‌کند.

پورت سریال دارای یک کانکتور ۹ پین می‌باشد و از آنجایی که این استاندارد در ابتدا برای طراحی با مودم طراحی شده بود، دارای پین مای Handshaking و وضعیت می‌باشد. اما نوع خاصی از ارتباط با RS-232 به نام Null-Modem که تنها شامل پین- مای ارسال و دریافت است، برای ارتباط با غیر از مودم استفاده می‌شود. بنابراین تنها دو پین Rx و Tx (و البته زمین) مورد نیاز است. در انتقال سریال قبل از ارسال هر کاراکتر یک بیت صفر (start bit) به معنی شروع و آمادگی و سپس ۸ بیت اطلاعات و در آخر ۱ یا ۲ بیت (stop bit) به عنوان توقف یا پایان یک کاراکتر ارسال می‌شود.

در استاندارد RS-232 سطح ولتاژ +۳ تا +۱۲ ولت نمایانگر وضعیت Space یا صفر منطقی و بازه ۳- تا ۱۲- ولت نمایشگر وضعیت Mark یا یک منطقی می‌باشد. اگرچه تجهیزات استاندارد TTL با سطوح منطقی ۰ و ۵ ولت کار می‌کنند اما قالب اطلاعات ارسالی تفاوتی ندارد و با یک مدار تغییر سطح ولتاژ، PC می‌تواند با ادوات TTL ارتباط برقرار نماید. یکی از مبدل مای RS-232 به TTL مدار مجتمع MAX232 و یا HIN232 می‌باشد.

* تذکر:

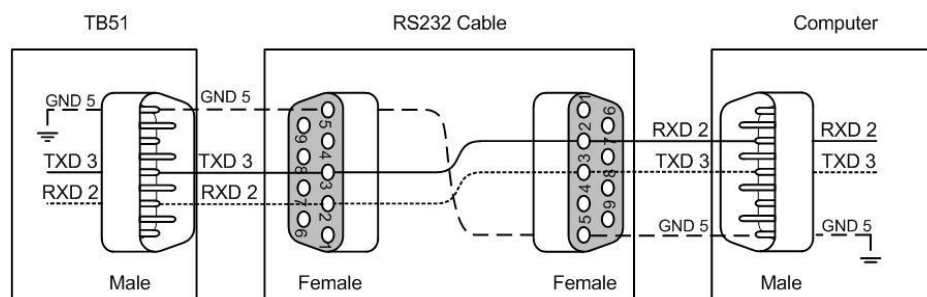
RS232 DB9 (EIA/TIA 574)



شکل ۱۶-۱ کانکتور پورت سریال (D9)

PIN	Description
۱	data carrier detect
*۲	received data(RxD)
*۳	transmitted data(TxD)
۴	data terminal ready
*۵	signal ground(GND)
۶	data set ready
۷	request to send
۸	clear to send
۹	ring indicator

جدول ۱۶-۲ عملکرد پین مای D9



شکل ۱۶-۳ Null modem connection

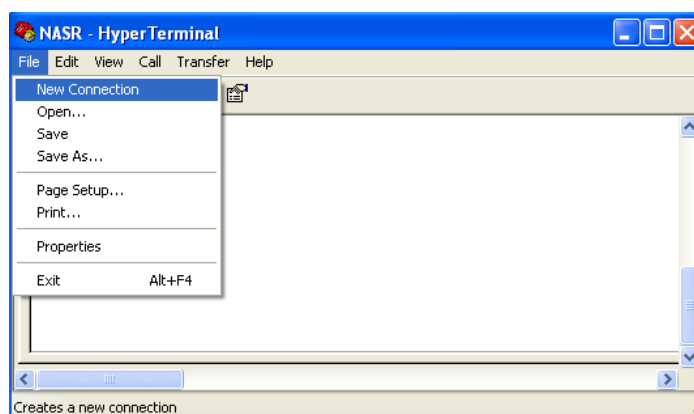
تکالیف داخل آزمایشگاه

(۱) Hyperterminal و ارتباط سریال از طریق PC

Hyperterminal نرم‌افزاری همراه همه نسخه‌های سیستم‌عامل Windows است که می‌تواند به‌عنوان ترمینال ارتباط سریال استفاده

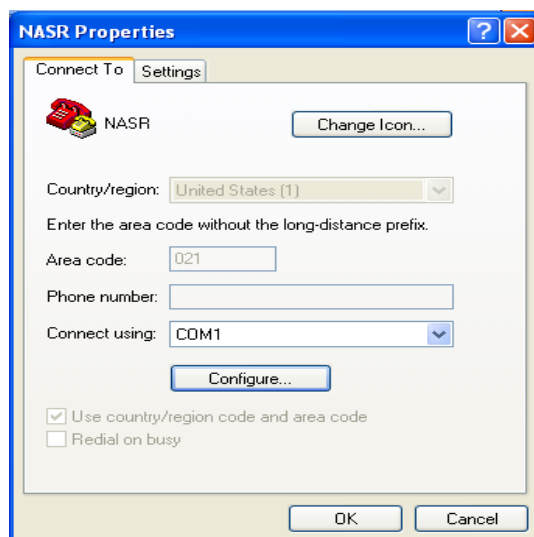
شود. به این منظور ابتدا باید تنظیمات زیر انجام شوند:

ابتدا از پنجره File گزینه New Connection را انتخاب کنید.



پس از انتخاب یک اسم، پنجره Connect to باز می شود.

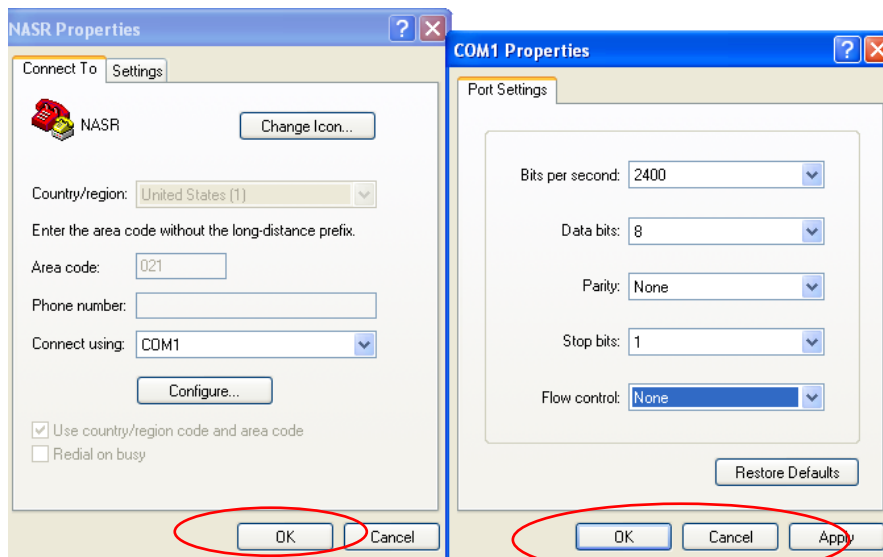
برای تنظیمات New Connection ، وارد قسمت File\ Properties شوید.



در قسمت Connect using گزینه Com 1 را انتخاب کنید. در پنجره COM Properties تنظیمات ارتباط سریال زیر را انجام دهید:

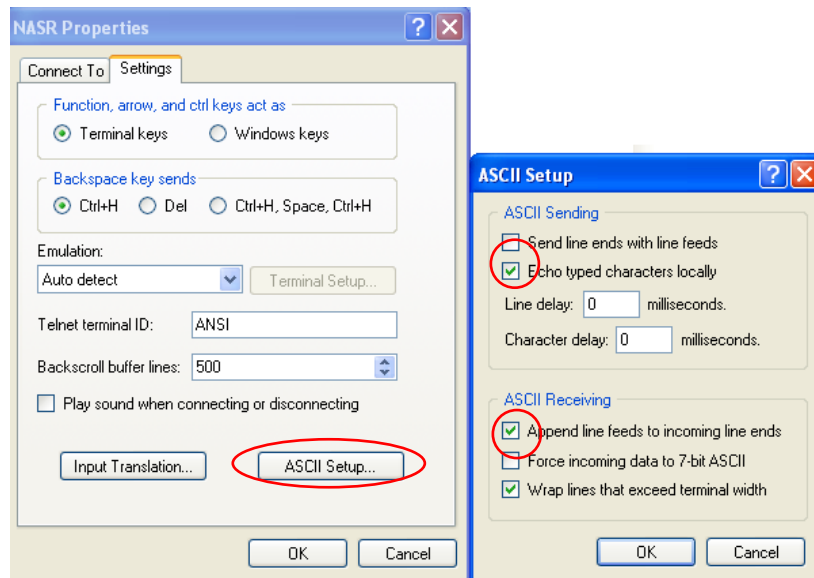
(ارتباط سریال با مشخصات ۸ بیت بدون Parity Bit و یک Stop Bit و Buad Rate برابر ۲۴۰۰ bps)

Bits per sec = 2400, Data bits = 8, Parity = none, Stop bits = 1, Flow control = none



برای مشاهده کاراکترهایی که تایپ می کنید، ذیل پنجره File گزینه Properties و سپس Setting tab را انتخاب کنید. به قسمت

ASCII Setup... بروید و گزینه Echo typed characters locally را فعال کنید.

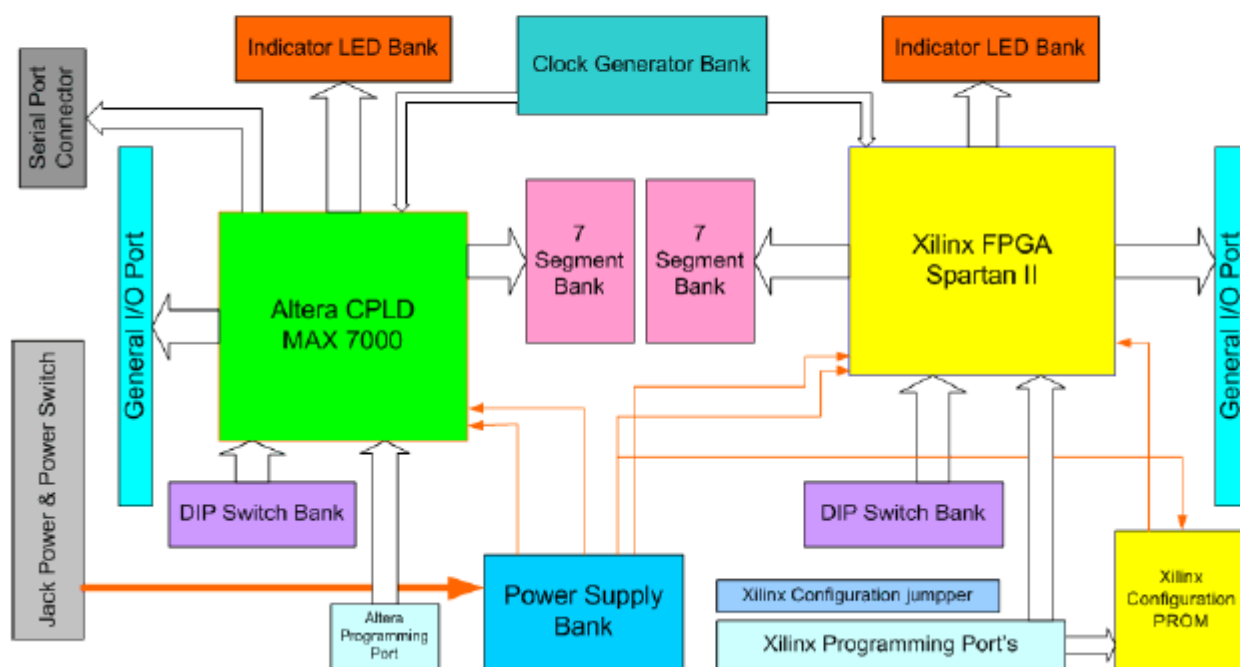


(۲) یک کاراکتر اسکی را به‌طور دائم از پورت سریال TxD ارسال کنید و آن را روی اسیلوسکوپ مشاهده کنید. BaudRate را 2400bps قرار دهید.

(۳) کد verilog مربوط به ارسال و دریافت دیتا را از طریق پورت سریال نوشته سپس از طریق آن یک کاراکتر اسکی را به‌طور مداوم به FPGA ارسال کنید. همچنین عملیات عکس آن یعنی ارسال از FPGA را نیز آزمون نمایید.

آشنایی با برد مبتنی بر تراشه شرکت ALTERA

Logic Lab NASRSHARGH



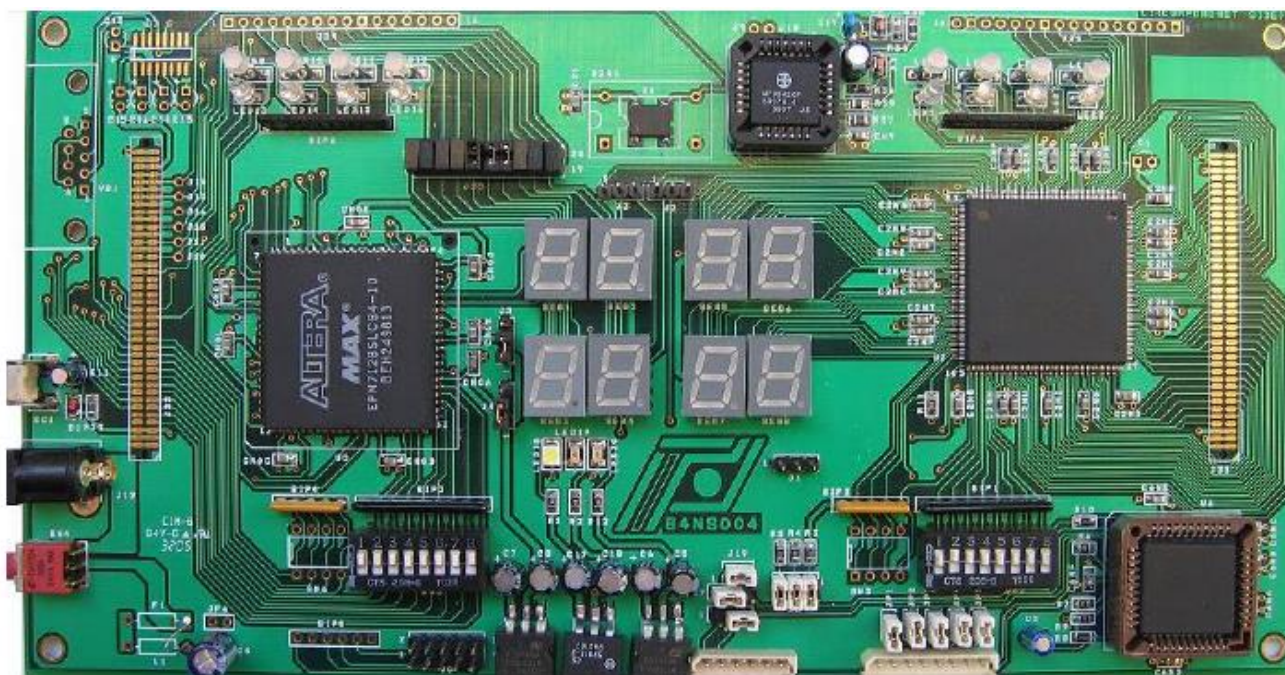
مولد پالس ساعت

مولد کلاک برای تولید کلاک‌های ۲ MHz، ۴ MHz، ۸ MHz و ۱۶ MHz و دو عدد سیگنال کلاک 8KHz با دقت بسیار و بالا از شرکت ZARLINK برای آزمایش‌هایی نظیر مقسم فرکانس یا کلاک CPU در نظر گرفته شده است.

نمایشگر LED و هفت قسمتی

در هر قسمت دوازده عدد switch به عنوان ورودی‌های صفر و یک FPGA یا CPLD می‌باشد. چهار LED تک رنگ و چهار LED دو رنگ برای نمایش اعداد چهاربیتی باینری در نظر گرفته شده است. چهار عدد Seven Segment برای نمایش اعداد به صورت نمادین در قسمت وسط برد موجود می‌باشد.

همان طور که در برد نیز مشاهده می‌نمایید کانکتورهای J25، J2 و J3 به علت کمبود پایه‌های MAX 7000 جهت برقراری ارتباط با نمایشگرهای ۷ قسمتی Seg3 و Seg4 مورد استفاده قرار گرفته‌اند. بدین ترتیب که پایه‌های aaa1 تا aaa10 از طریق کانکتور J25 به کلیه پایه‌های Seg3 و پایه‌های ۴۱ و ۴۲ Seg4 متصل شده‌اند و پایه‌های ۴۷ و ۴۸ Seg4 نیز از طریق کانکتورهای J2 و J3 به FPGA وصل شده‌اند.



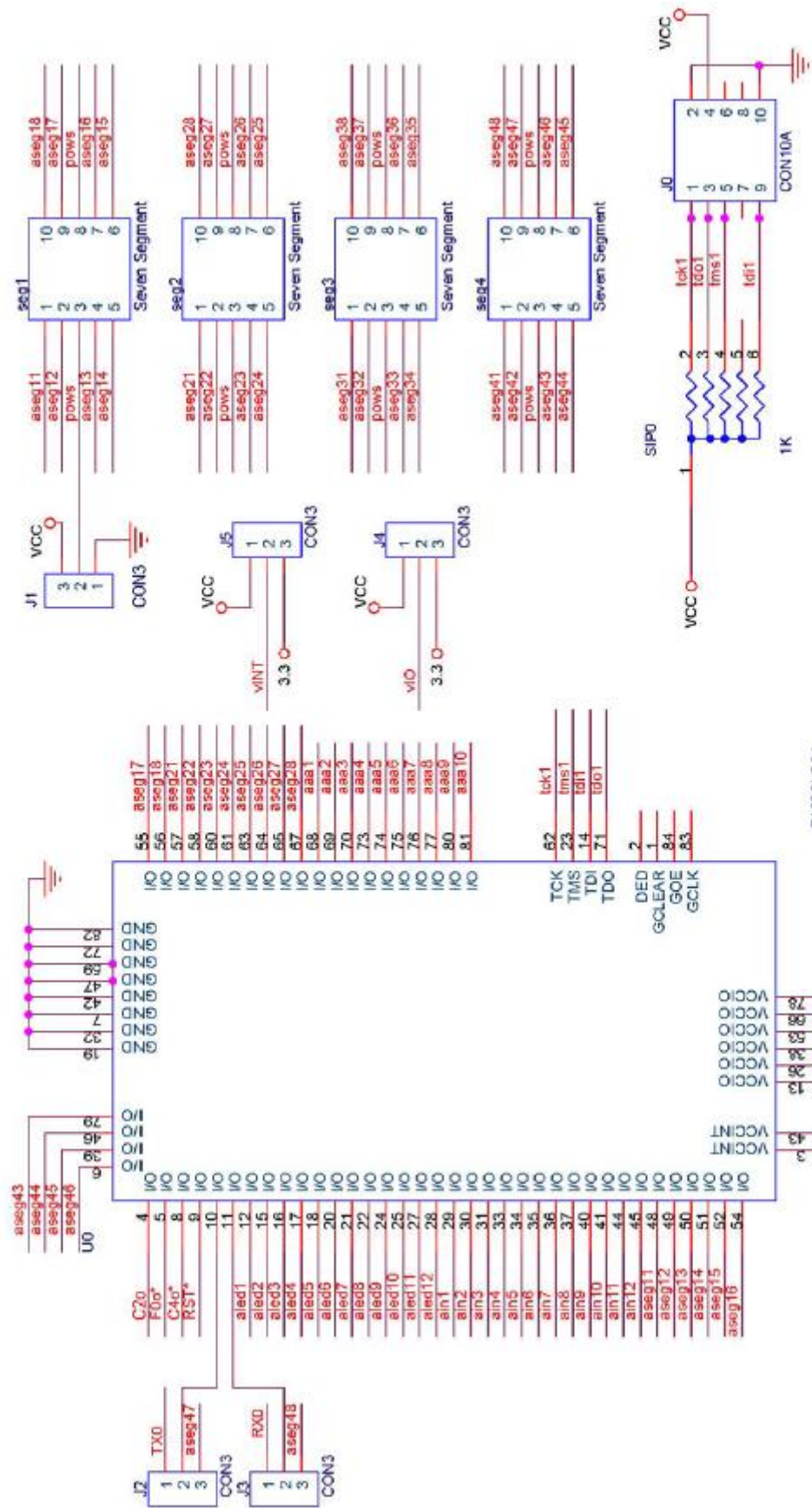
کانکتورهای I/O (ورودی/خروجی) و Test Point ها

برای انجام پروژه‌های دانشجویی پایه‌های مختلفی (PIN HEADER) را در اطراف برد قرار داده‌ایم تا بدین طریق بتوان سیگنال‌های مورد نیاز خود را از بیرون وارد سیستم نمایند و بعد از انجام آزمایش به راحتی به خارج برد انتقال دهند. در قسمت CPLD کانکتورهای J22 و J24 و در قسمت FPGA کانکتور J23 و J21 بدین منظور در نظر گرفته شده‌اند. در قسمت CPLD کانکتور J25 در نظر گرفته شده تا چنانچه دانشجو قصد نمایش مقادیر پایه‌های کانکتور J24 را روی 7-SEG داشته باشد، این عمل امکان پذیر باشد. همچنین در سمت چپ برد شش عدد TEST POINT J10, J11, J13, J14, J15, J16 قرار داده شده است. در هر قسمت یک کانکتور که به پورت LPT1 متصل می‌شود برای برنامه‌ریزی FPGA یا CPLD مورد استفاده قرار می‌گیرد. برای CPLD کانکتور J0 و برای FPGA کانکتور J20 است.

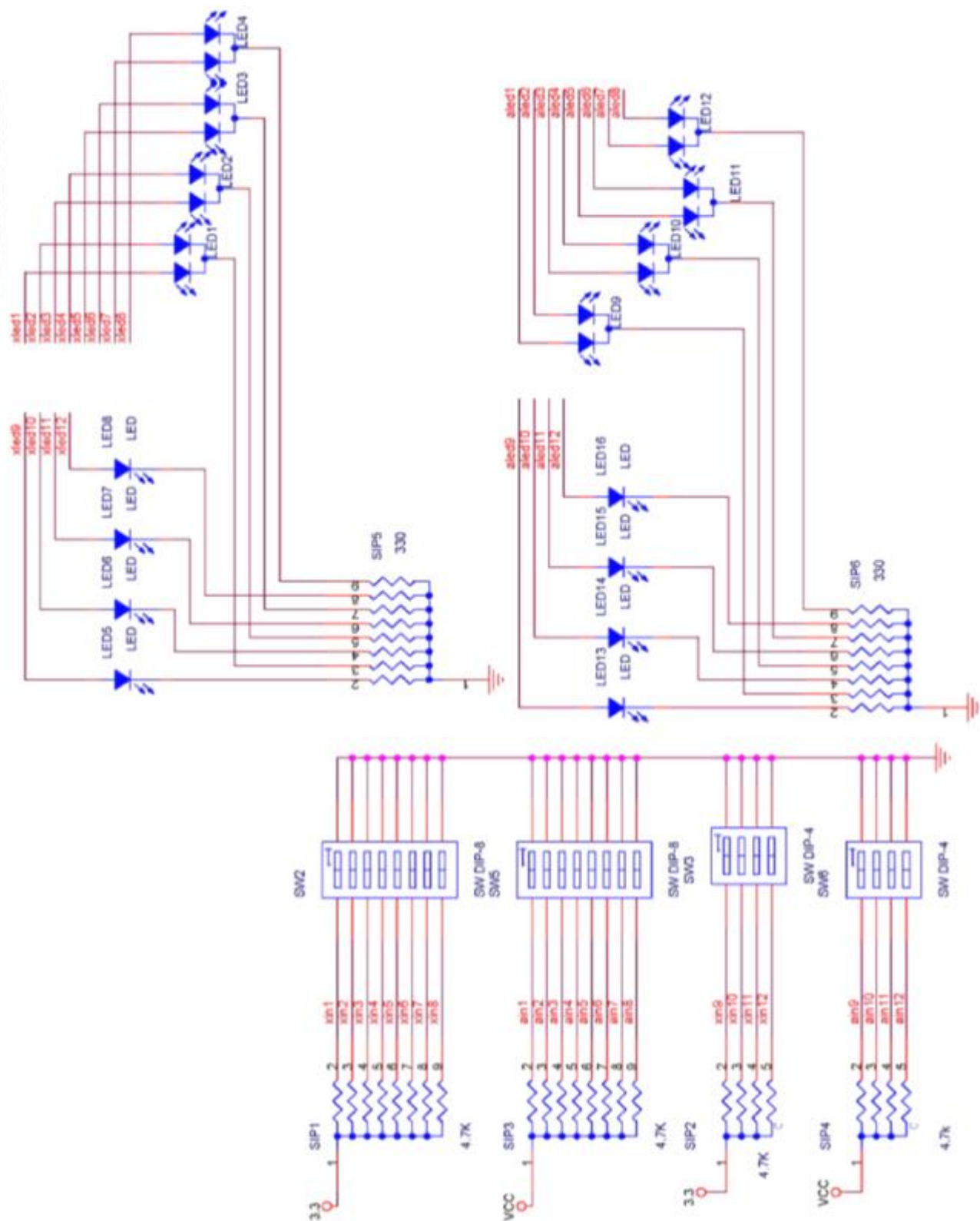
تغذیه

در سمت چپ برد یک Reset Switch برای صفر کردن یا دادن کلاک به صورت دستی در نظر گرفته شده است. همچنین تعدادی Regulator برای تولید ولتاژهای 2.5، 3.3 و 5 وجود دارند. ورودی این سیستم ۷/۵ تا ۱۲ ولت می‌باشد که از طریق کانکتور سیاه رنگ به داخل برد وارد می‌شود.

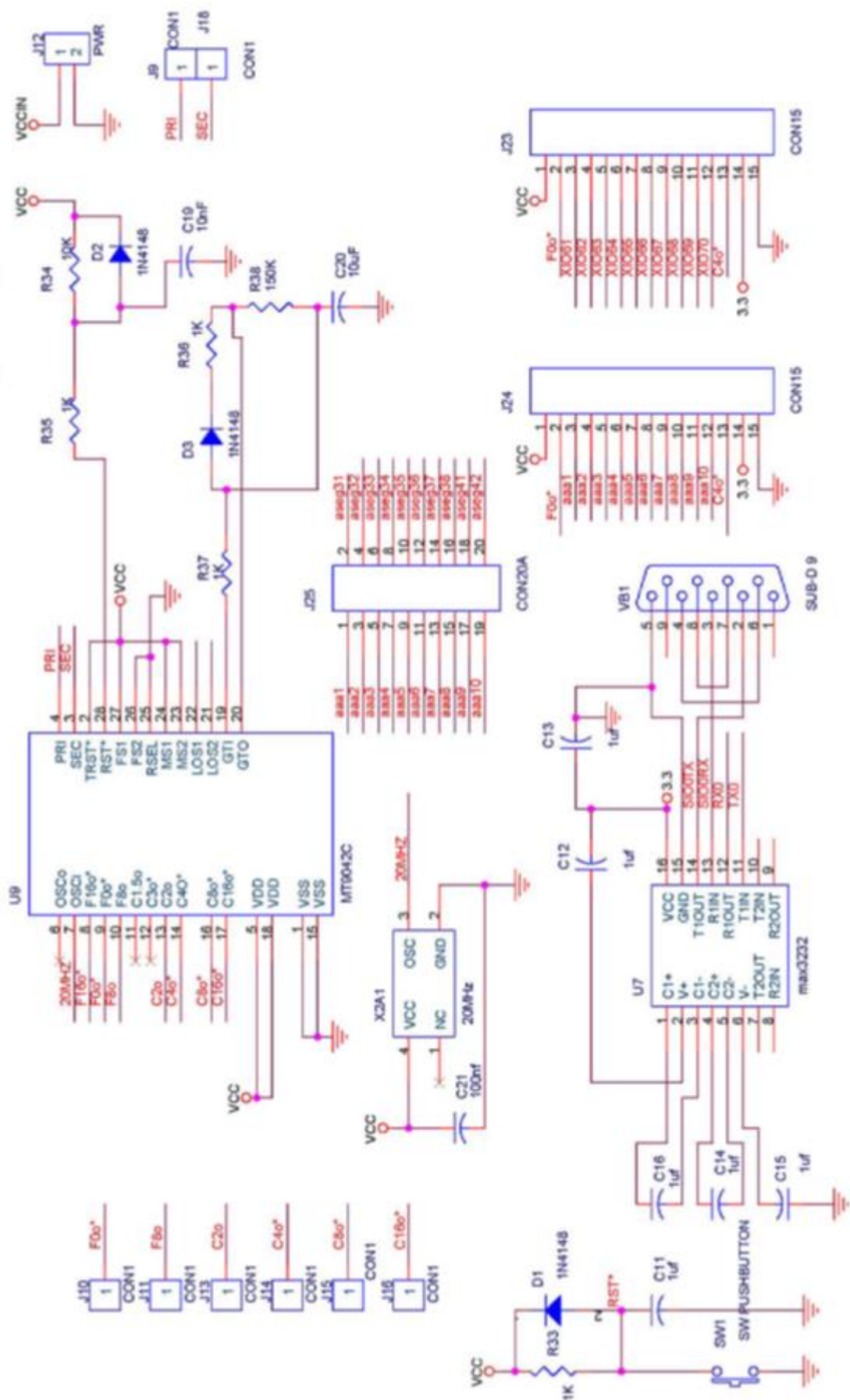
ALTERA FPGA



Switches & LEDs



RS-232 & mt9042

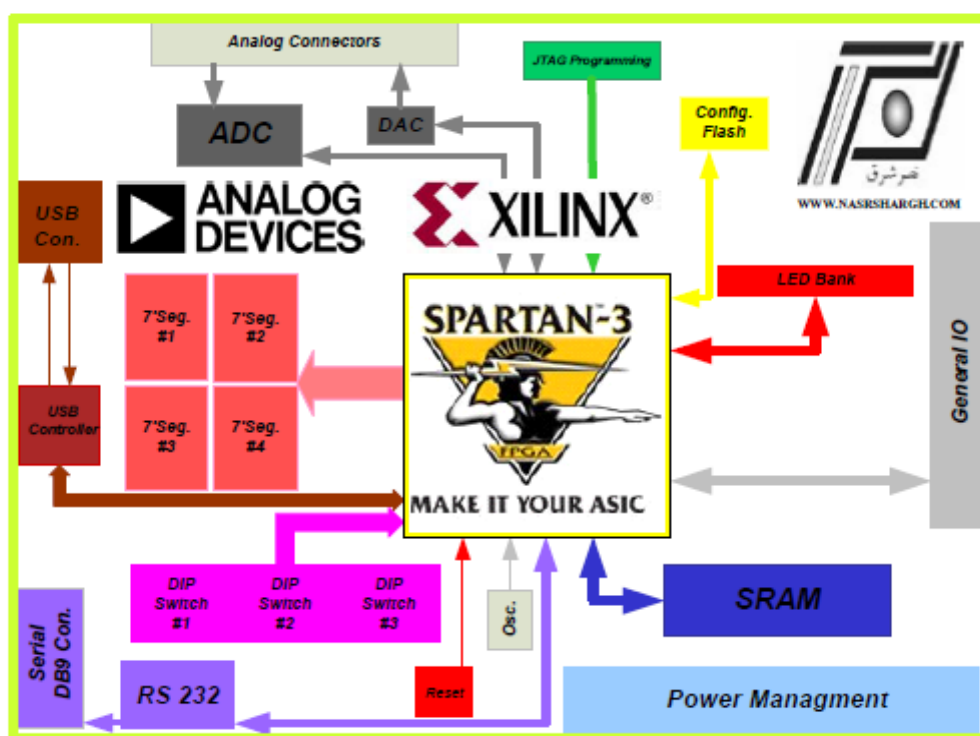


اتصالات Pin های ALTERA

<i>ALTERA PIN NO.</i>	<i>CONNECTED TO</i>	<i>ALTERA PIN NO.</i>	<i>CONNECTED TO</i>
4	J22-PIN34	44	SW6-NO. 3
5	J22-PIN26	45	SW6-NO. 4
6	SSEG4-C	46	SSEG4-D
8	J22-PIN36	48	SSEG1-G
9	SW1	49	SSEG1-F
10	J2-PIN2	50	SSEG1-E
11	J3-PIN2	51	SSEG1-D
12	LED9-1	52	SSEG1-DP
15	LED9-3	54	SSEG1-C
16	LED10-1	55	SSEG1B
17	LED10-3	56	SSEG1-A
18	LED11-1	57	SSEG2-G
20	LED11-3	58	SSEG2-F
21	LED12-1	60	SSEG2-E
22	LED12-3	61	SSEG2-D
24	LED13	63	SSEG2-DP
25	LED14	64	SSEG2-C
27	LED15	65	SSEG2-B
28	LED16	67	SSEG2-A
29	SW5-NO. 1	68	J25-PIN1
30	SW5-NO. 2	69	J25-PIN3
31	SW5-NO. 3	70	J25-PIN5
33	SW5-NO. 4	73	J25-PIN7
34	SW5-NO. 5	74	J25-PIN9
35	SW5-NO. 6	75	J25-PIN11
36	SW5-NO. 7	76	J25-PIN13
37	SW5-NO. 8	77	J25-PIN15
39	SSEG4-DP	79	SSEG4-E
40	SW6-NO. 1	80	J25-PIN17
41	SW6-NO. 2	81	J25-PIN19

آشنایی با برد مبتنی بر تراشه شرکت Xilinx

مجموعه آزمایشگاه FPGA از بخش‌های مختلفی تشکیل شده است که این بخش‌ها شامل برد اصلی، پروگرامر شرکت XILINX و مستندات مربوطه است. در این گزارش فنی شما با ساختار برد اصلی آزمایشگاه و بخش‌های مختلف تشکیل دهنده آن آشنا خواهید شد. شکل ۱ بلوک دیاگرام برد آزمایشگاه را نمایش می‌دهد. به طور 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 بخش‌ها را روی برد آزمایشگاه مشاهده می‌نمایید. همان طور که در این شکل مشخص است. در ادامه به توضیح قسمت‌های مختلف این برد خواهیم پرداخت.



شکل ۲: بلوک دیاگرام برد FPGA

تراشه FPGA

تراشه اصلی یا همان FPGA که روی این برد استفاده شده است از سری XC3S400 از شرکت XILINX می‌باشد. همان طور که می‌دانید شرکت XILINX به عنوان یکی از اصلی‌ترین تولید کنندگان تراشه‌های الکترونیکی و بخصوص FPGA است. این شرکت تا کنون خانواده‌های مختلفی از FPGA ها را تولید و به بازار عرضه نموده است. یکی از این سری تراشه‌ها خانواده Spartan-3 است که در جدول ۱ انواع آنها از نظر تعداد گیت‌ها، بیت‌های حافظه تعداد I/O ها و نوع بسته‌بندی نمایش داده شده است. تراشه‌ای که روی این برد استفاده شده است با بسته‌بندی PQFP و دارای ۲۰۸ پایه است و بیشترین پایه‌ای که در اختیار کاربر قرار می‌دهد ۱۴۱ عدد است.

جدول ۳

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	62,108	74,880
Block RAM Bits	72K	216K	288K	432K	576K	720K	1,728K	1,872K
Distributed RAM Bits	12K	30K	56K	120K	208K	320K	432K	520K
DCMs	2	4	4	4	4	4	4	4
Multpliers	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

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 نرم افزار ISE برنامه‌ریزی 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 قرار داد.

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

⇐ J6A : 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 به منظور تعیین حالت برنامه‌ریزی FPGA در نظر گرفته شده است که چگونگی آن را در جدول زیر مشاهده می نمایید.

جدول ۴

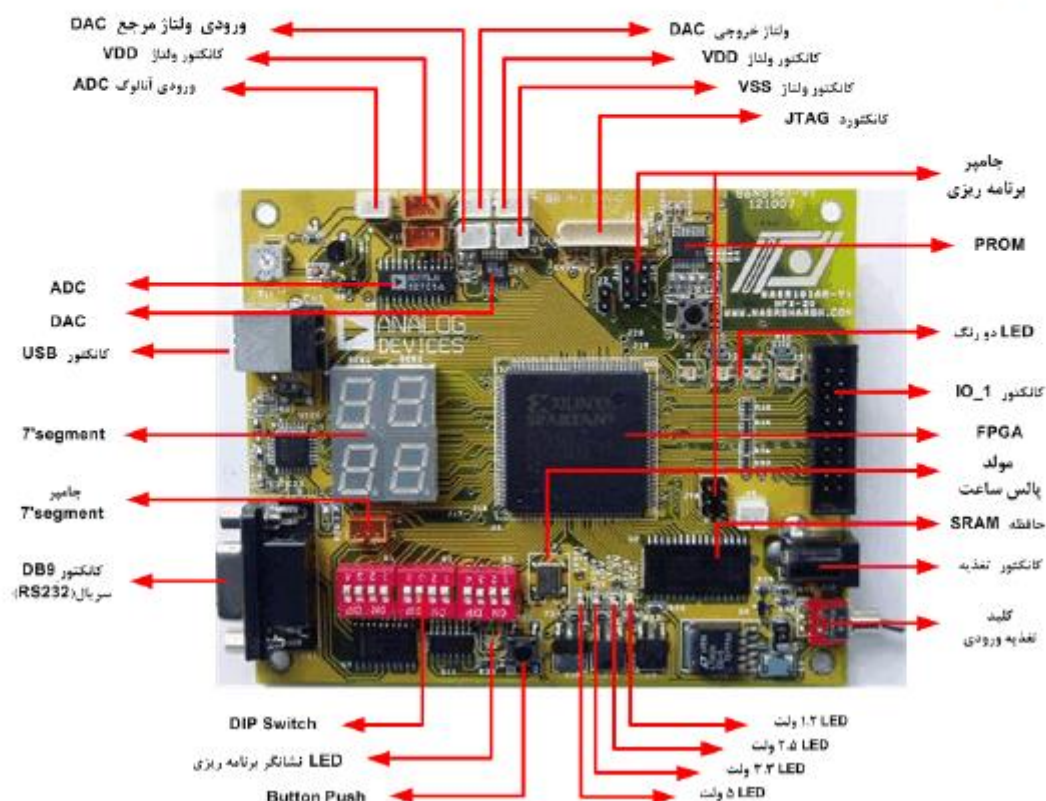
Configuration Mode ⁽¹⁾	M0	M1	M2	Synchronizing Clock
Master Serial	0	0	0	CCLK Output
Slave Serial	1	1	1	CCLK Input
Master Parallel	1	1	0	CCLK Output
Slave Parallel	0	1	1	CCLK Input
JTAG	1	0	1	TCK Input

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

تغذیه

تغذیه ورودی برد از طریق کانکتور J12 (کانکتور تغذیه) و کلید تغذیه ورودی تأمین و از طریق سه رگولاتور، ولتاژهای ۵ ولت، ۳/۳ ولت و ۲/۵ ولت ساخته می‌شود. LEDهای نشانگر ولتاژهای ۱/۲، ۲/۵، ۳/۳ و ۵ ولت در قسمت بالای رگولاتورها با شماره‌های LED18,19,20,21 برای نشان دادن وضعیت ولتاژهای سیستم به کار می‌روند. محدوده ولتاژ تغذیه ورودی از ۷ تا ۱۲ ولت بوده که بهتر است خروجی یک منبع تغذیه پایدار باشد.



شکل ۳: برد اصلی آزمایشگاه

مولد پالس ساعت

یک اسیلاتور با فرکانس 20MHz روی برد قرار گرفته است که به پایه شماره ۲۲ FPGA متصل می‌باشد و می‌تواند به عنوان clock اصلی برد مورد استفاده قرار گیرد.

حافظه

یک حافظه ۱۲۸ کیلو بیت شرکت SAMSUNG به شماره K6R1008V1C روی برد قرار دارد و باسهای آدرس، داده و سیگنالهای کنترلی آن به FPGA متصل شده است (در جدولی که در انتهای این مستند آورده شده است پایه‌های استفاده شده مشخص شده است).

مبدل آنالوگ به دیجیتال

برای پردازش سیگنال آنالوگ توسط FPGA یک ADC شرکت Analog Device به شماره AD775 روی برد قرار گرفته است. این ADC هشت بیتی با ولتاژ ۵ ولت تغذیه شده و می تواند تا 20MSPS کار نماید. باس داده و سیگنالهای کنترلی ADC به پایه های FPGA متصل شده است (در جدولی که در انتهای این مستند آورده شده است پایه های استفاده شده مشخص شده است).

مبدل دیجیتال به آنالوگ

برای ارتباط با دنیای آنالوگ و ارائه نتیجه پردازش سیگنال آنالوگ به صورت خروجی، از یک DAC شرکت Analog Device استفاده شده است. این DAC هشت بیتی با تغذیه ۲/۵ تا ۵ ولت کار می کند و باس داده و سیگنالهای کنترلی آن به پایه های FPGA متصل شده است (در جدولی که در انتهای این مستند آورده شده است پایه های استفاده شده مشخص شده است).

کانکتورهای I/O (ورودی/خروجی)

برای ارتباط با خارج برد یک کانکتور ورودی-خروجی ۲۰ پایه در نظر گرفته شده است و می توان از آن برای ارتباط با اجزای جانبی مانند LCD و یا بردهای دیگر مورد 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 می کند در نظر گرفته شده است. این دو پایه از طریق تراشه MAX232 شرکت MAXIM به ولتاژهای RS-232 تبدیل شده و به کانکتور DB9 سری وصل می گردند.

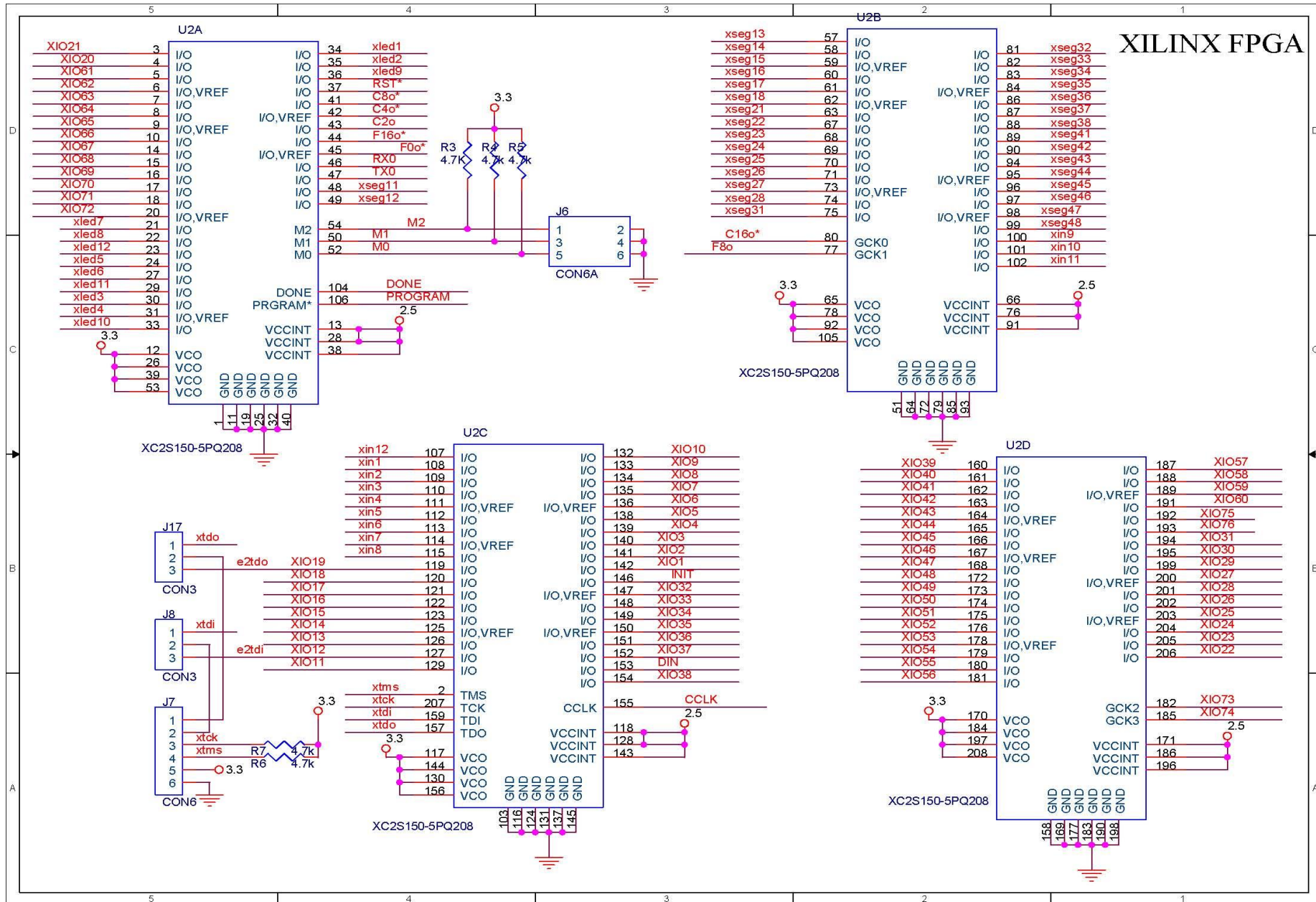
نمایشگر LED و هفت قسمتی

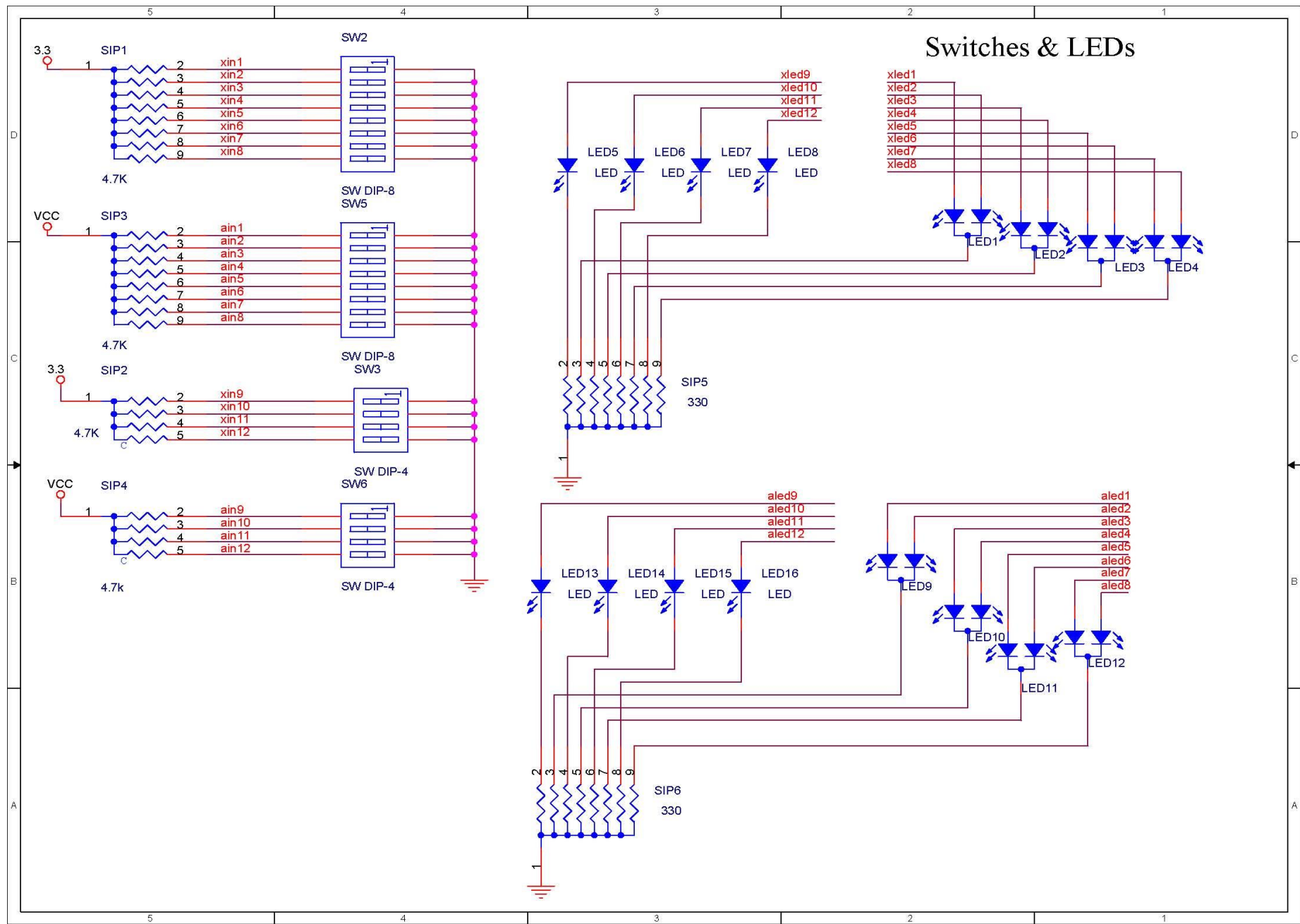
۸ LED در ۴ پکیج بسته بندی شده اند، برای نمایش سیگنالهای خروجی و نیز چهار عدد نمایشگر هفت قسمتی برای نمایش اعداد و ... روی برد موجود است که به یک سری از پایه های FPGA متصل هستند. همچنین یک jamper برای نمایشگرهای هفت قسمتی در نظر گرفته شده است تا در حالت آند مشترک و یا کاتد مشترک (بسته به اینکه نمایشگرها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

سوئیچ های ورودی و RESET

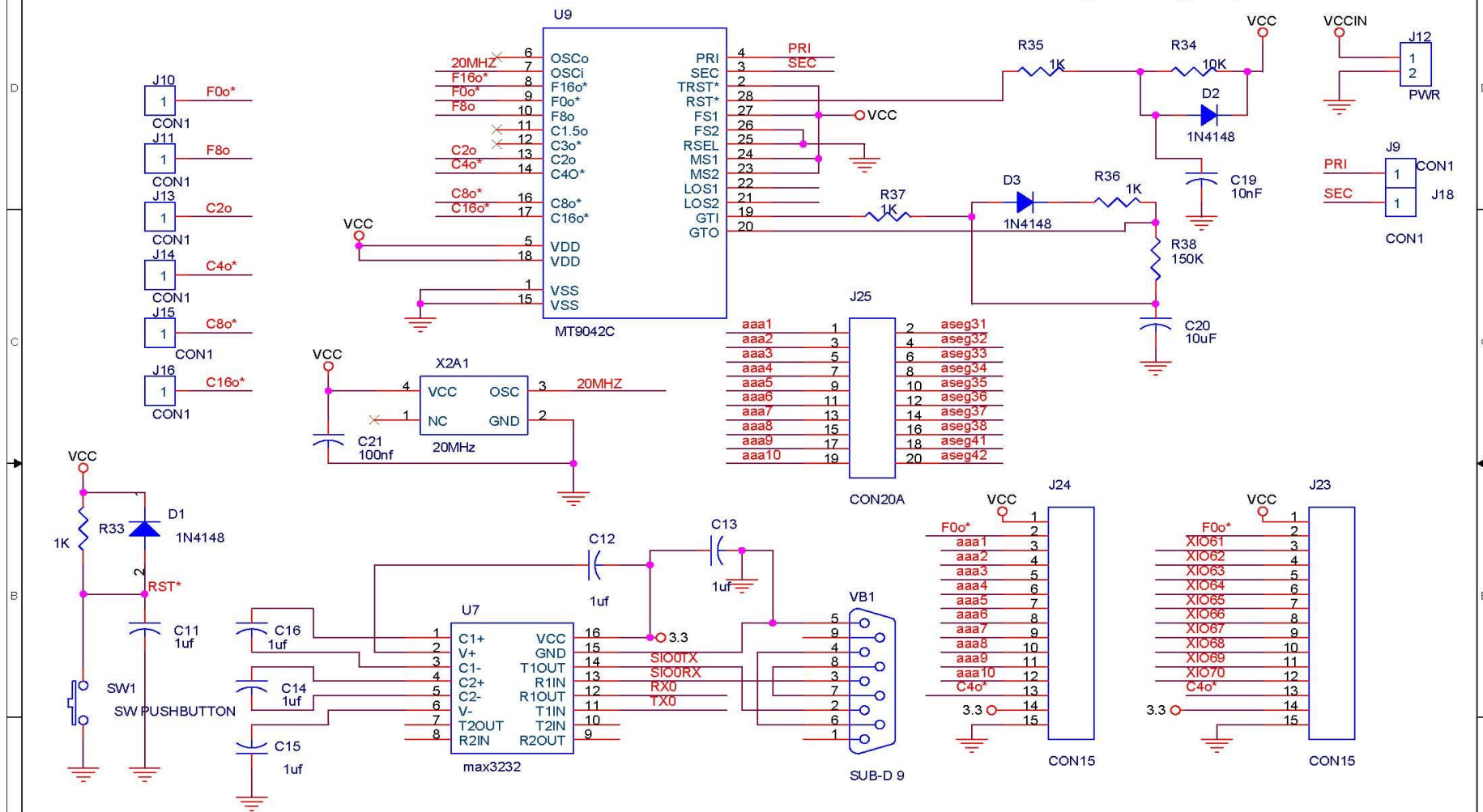
یک Push Button برای سیگنالهای ورودی لحظه ای مانند Reset و ... و سه عدد DIP Switch چهار تایی برای سیگنالهای ورودی setting و یا برای ارسال اعداد BCD، باینری و ... روی برد در نظر گرفته شده که به یک سری از پایه های FPGA متصل هستند.

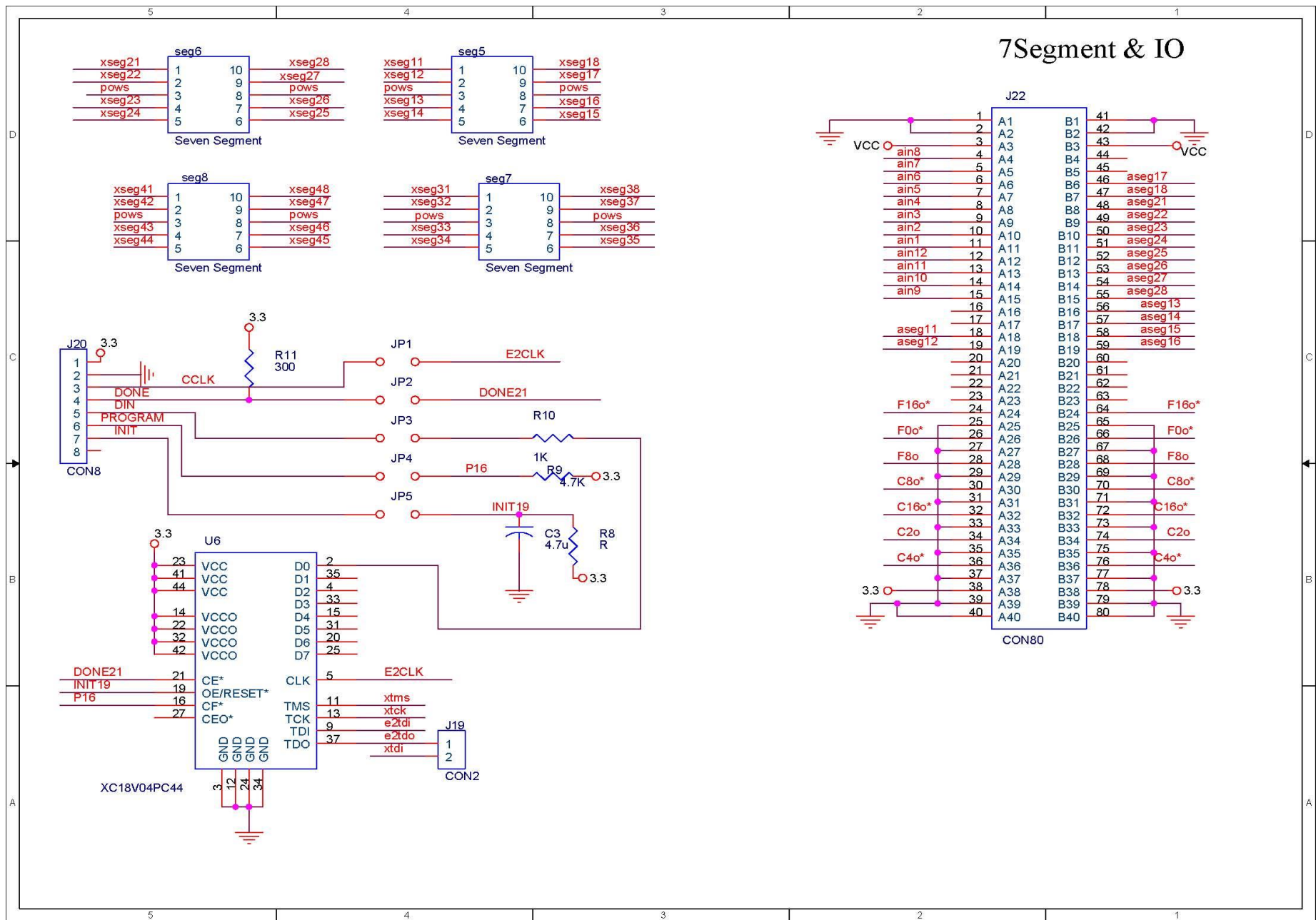
XILINX FPGA





RS-232 & mt9042





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