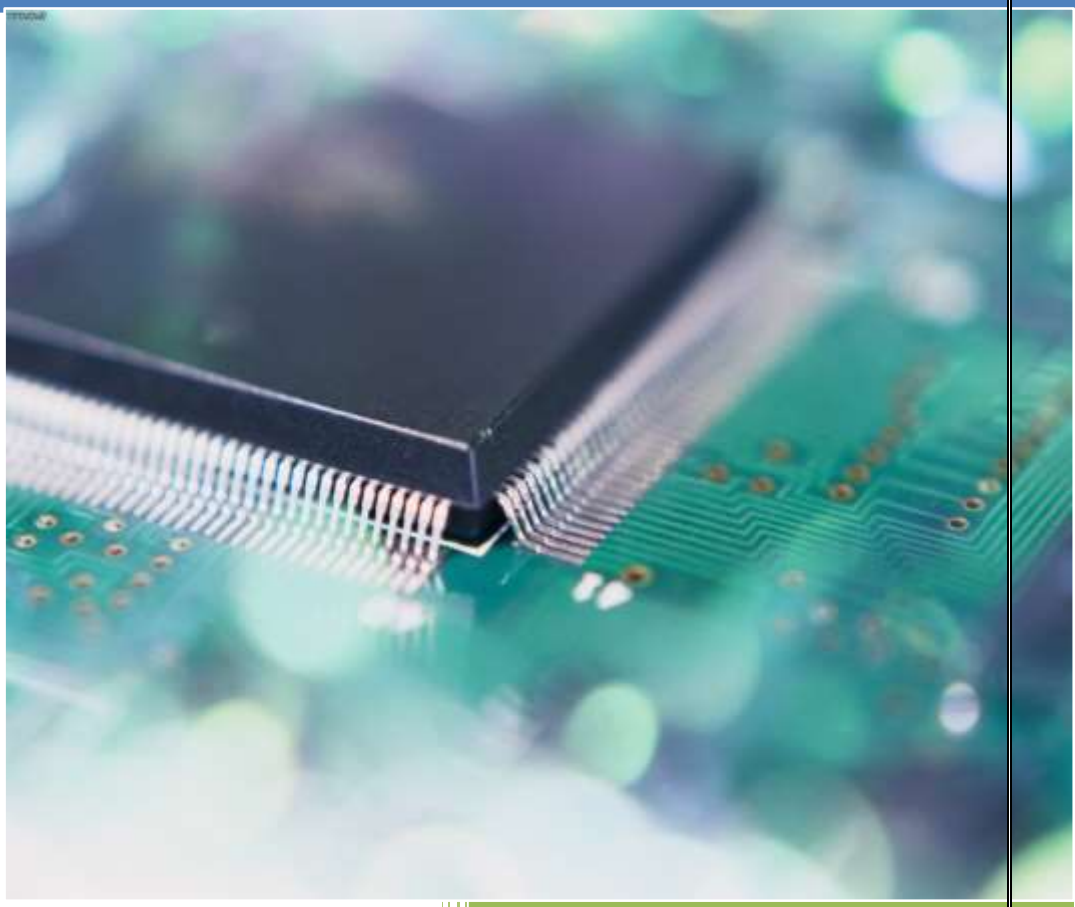
بسمه تعالی

دانشگاه کاشان

آزمایشگاه مدار منطقی و معماری کامپیوتر



دانشکدهٔ برق و کامپیوتر

آزمایشگاه دیجیتال

محمّدرضا فتّاح

بخش اول

آشنایی با تراشه های منطقی و چگونگی بکارگیری آنها

بسمه تعالی

هدف از ارائهٔ آزمایشگاه مدارهای منطقی ،در مرحلهٔ اول آشنایی دانشجویان با چگونگی کار با تراشه‌های دیجیتال و همچنین چگونگی طراحی وپیاده سازی مدارات ساده ترکیبی وترتیبی دیجیتال با روش معمول و دستی و در مرحلهٔ دوم طراحی و پیاده سازی با استفاده از زبانهای برنامه نویسی و توسط آی سی های برنامه پذیر است . قبل از شروع کار در آزمایشگاه لازم است نکاتی را در مورد چگونگی انجام آزمایشات و همچنین ارزیابی کار هر دانشجو متذکر شویم :

۱- دانشجو برای انجام هر آزمایش باید مقدمات آنرا قبل از ورود به آزمایشگاه آماده کند . که این موارد عبارتند از :

الف- مطالعه راهنمای آی سی های بکار رفته در آزمایش . راهنمای این آی سی ها در دستور کار آزمایش موجود است .

ب- تهیه پیش گزارش : بسیاری از آزمایشاتی که قرار است انجام شود لازم است مدار آن قبلا طراحی شود. پس دانشجو باید قبل از ورود به آزمایشگاه و بعنوان پیش گزارش طراحی مدار را انجام داده و تمام مراحل طراحی را در پیش گزارش بیاورد . این موارد شامل جداول درستی ، جداول ساده سازی ، عبارات و همچنین شکل نهایی مدار است .

۲- حضور در تمام جلسات آزمایشگاه الزامی است و در صورت غیبت ، دانشجو باید در همان هفته و در یکی دیگر از گروهای آزمایشگاه و البته با هماهنگی مربی آزمایشگاه ، آزمایش مربوطه را انجام دهد . لازم به ذکر است غیبت بیش از دو جلسه مجاز نیست .

۳- هر آزمایش دارای چند قسمت می باشد که تمامی آنها در یک جلسه دو ساعته انجام می شود . بعد از اتمام هر جلسه آزمایشگاه دانشجو باید یک گزارش کار از آزمایشات انجام شده به مربی آزمایشگاه تحویل دهد . البته گزارش به صورت گروهی و حداکثر یک هفته بعد از انجام آزمایش تحویل می شود .

گزارش آزمایشگاه شامل پاسخ سؤالات در هر آزمایش و در نهایت برداشت نهایی و تجربیات کسب شده دانشجو در آن آزمایش می باشد .

۴- نمره نهایی از مجموع نمرات زیر بدست خواهد آمد :

پیش گزارش کامل و درست(۲۵ درصد) – کار در آزمایشگاه و جواب درست آزمایشها(۳۵ درصد) – گزارش کار (۱۵ درصد)– امتحان عملی یا کتبی (۲۵ درصد)

لازم بذکر است حضور دانشجو در امتحان آزمایشگاه الزامیست . در غیر این صورت بقیه درصد نمرات نیز لحاظ نخواهد شد .

با آرزوی توفیق برای شما

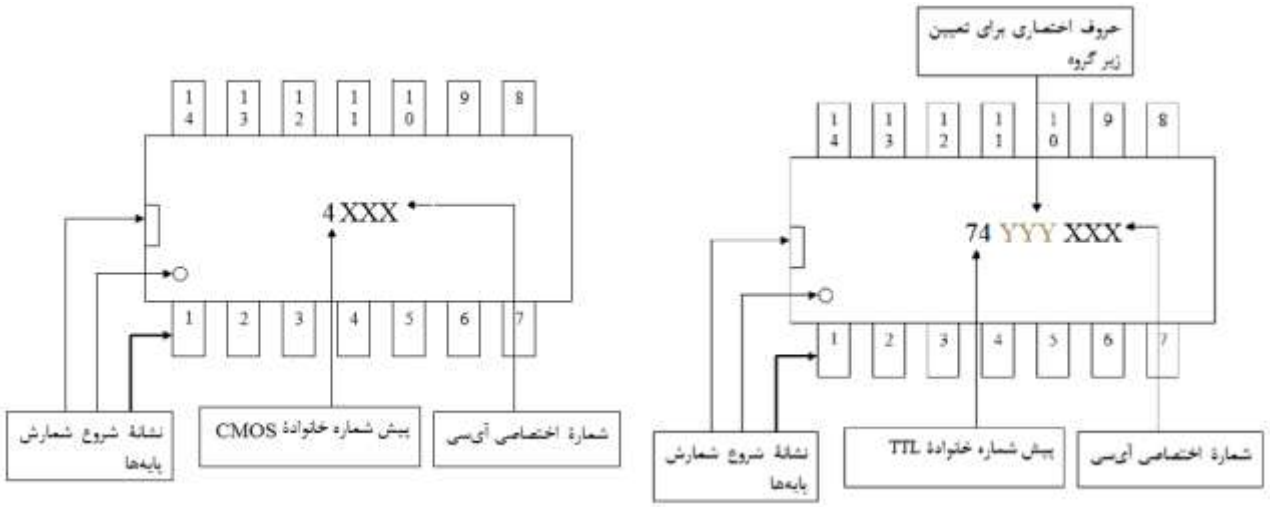
مربی آزمایشگاه : محمدرضا فتاح

مقدمه

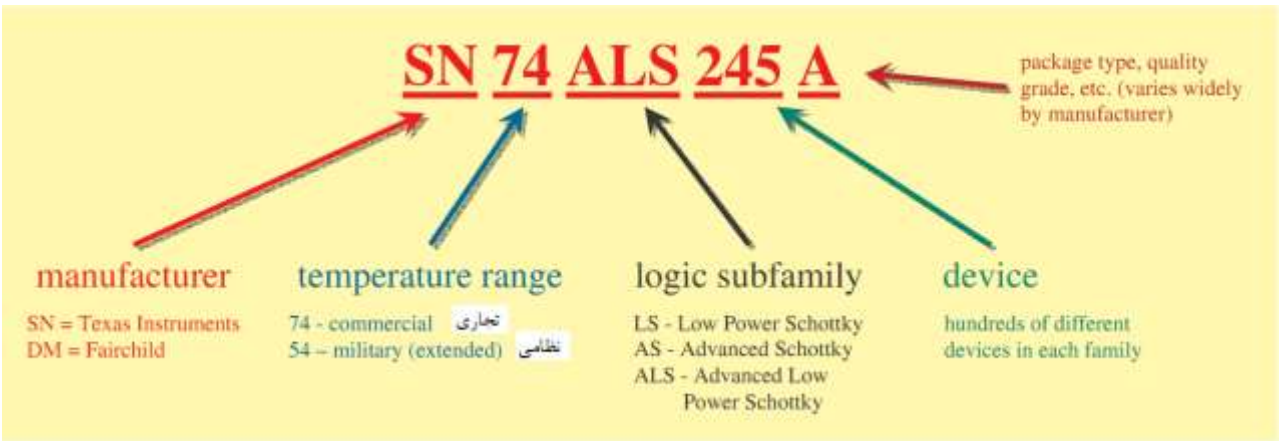
قبل از شروع کار لازم است آشنایی اولیه در مورد انواع آی سی‌های منطقی داشته باشیم و همچنین نحوهٔ کار با آنها را یاد بگیریم .

متداولترین خانواده مدارهای منطقی که حاوی مدارهای پایه هستند عبارتند از خانوادهٔ CMOS و خانوادهٔ TTL.

برای شناخت یک تراشهٔ دیجیتال از نظر نوع تکنولوژی ساخت و عملکرد آن ، می‌توان از شمارهٔ مخصوصی که روی هر آی‌سی نوشته شده است استفاده کرد . با توجه به این شماره و مراجعه به کتابچه‌های CMOS یا TTL می توان با عملکرد آن تراشه بصورت کامل آشنا شد . متداولترین آی‌سی‌های TTL با پیشوند 74 و آی‌سی‌های CMOS با پیشوند 4 از هم متمایز می‌گردند . در شکل زیرطریقهٔ شناخت تراشه و ترتیب قرار گرفتن پایه‌های آن آورده شده است .



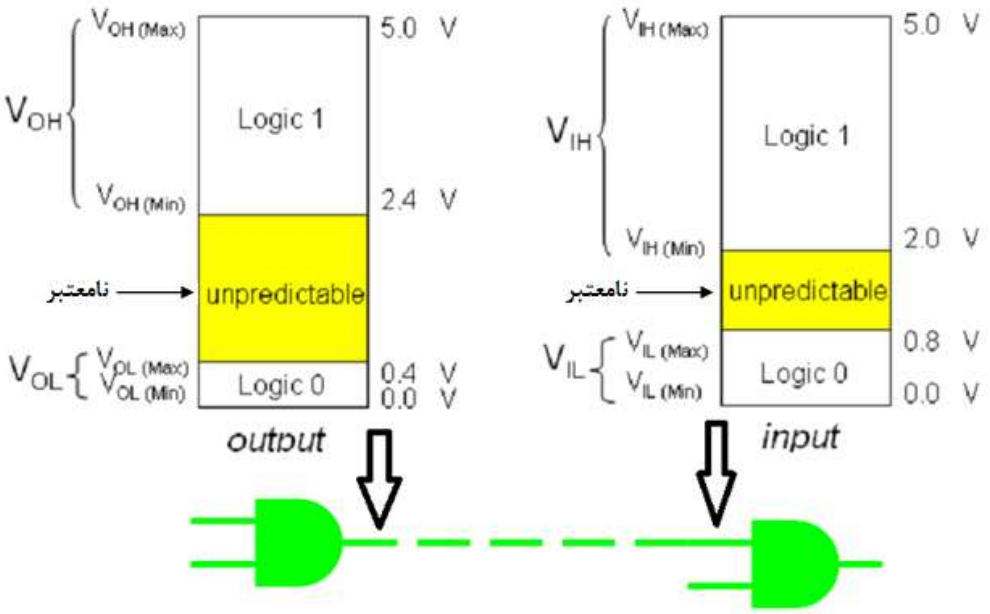
برای مثال شکل زیر چگونگی بدست آوردن اطلاعات اولیه از روی شمارهٔ تراشه را نشان داده است .



بعضی از زیر گروههای مربوط به خانوادهٔ TTL در جدول زیر آمده است :

حرف اختصاری	مفهوم آن
C	نمونهٔ CMOS آی‌سی TTL آن
F	نمونهٔ سریع
H	نمونهٔ سریع و پر قدرت
S	نمونهٔ شاتکی
HC	نمونهٔسریع CMOS آی‌سی TTL که با CMOS سازگار است
HCT	نمونهٔسریع CMOS آی‌سی TTL که با TTL سازگار است
L	کم مصرف
LS	کم مصرف با ورودی شاتکی
ALS	نمونهٔ پیشرفته کم مصرف با ورودی شاتکی

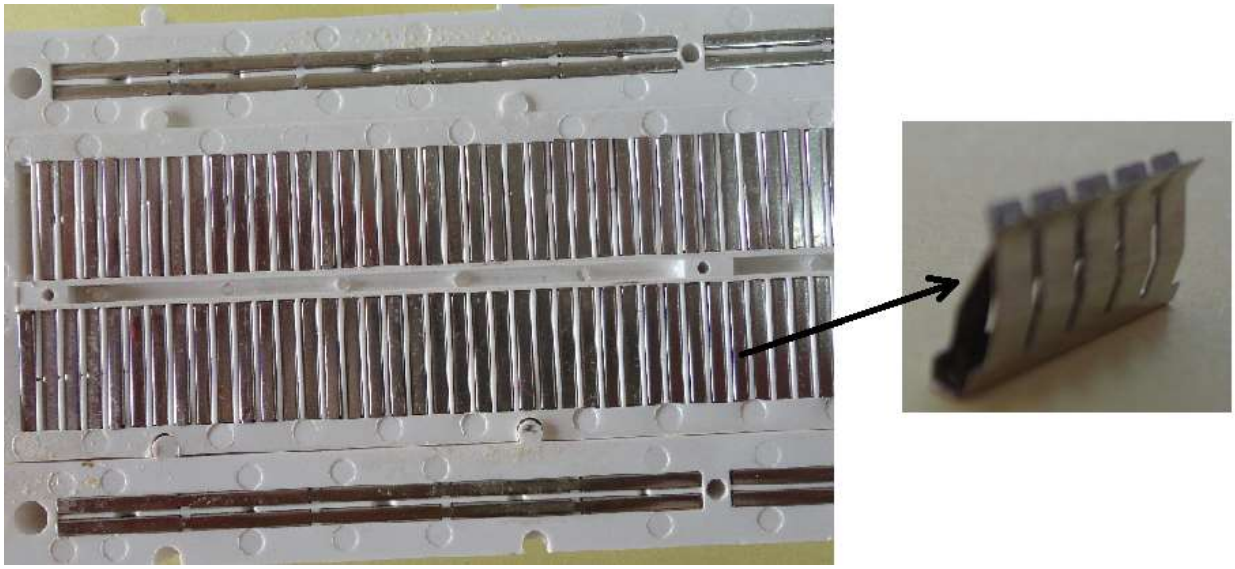
سطح تغذیه در خانواده TTL ولتاژ نامی ۵+ ولت و در خانوادهٔ CMOS از ۳+ تا ۱۵ ولت می تواند انتخاب شود . همچنین سطوح منطقی معتبر صفر و یک برای ورودی و خروجیهای خانواده TTLدر شکل زیر آمده است .



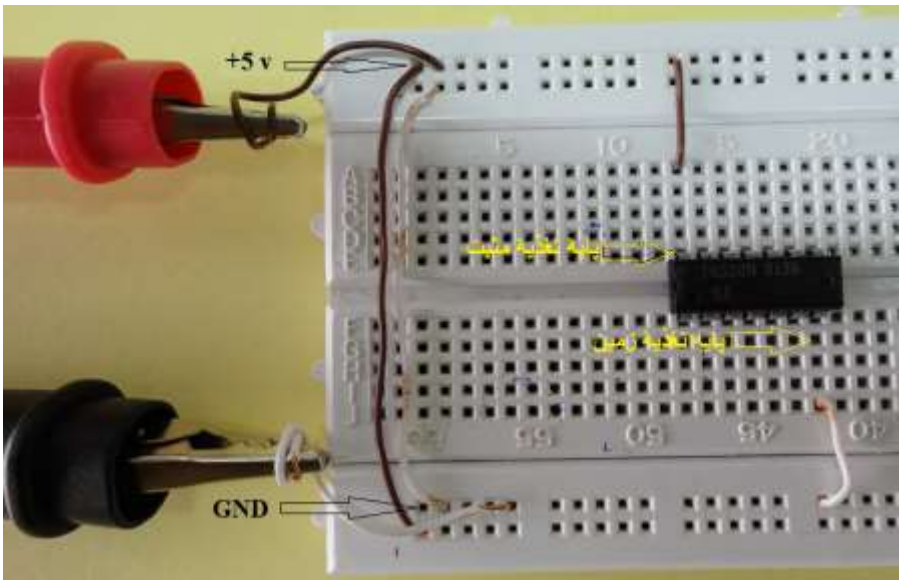
شکل سمت چپ ولتاژ معتبر برای خروجی یک گیت در سطوح منطقی ’0‘ و ’1‘ را نشان می دهد . برای سطح منطقی ’1‘ ولتاژ خروجی گیت در بازهٔ 2.4V تا 5V و برای سطح ’0‘ در بازهٔ 0V تا 0.4V باید قرار گیرد . حال اگر ولتاژ اندازه گیری شده در خروجی گیت در بازهٔ 0.4V تا 2.4V باشد سطح منطق خروجی نامعتبر خواهد بود . حالت نامعتبر معمولا ناشی از خرابی گیت ، ولتاژ تغذیهٔ ناکافی و یا ناشی از بار اضافی در خروجی است.
شکل سمت راست نیز سطوح ولتاژ معتبر برای ورودی گیت را نشان می دهد .

چگونگی استفاده از برد بورد در آزمایشگاه مدارهای منطقی:

به نمای ظاهری بردبورد در دو شکل زیر توجه کنید . این دو شکل اتصالات داخلی بردبورد را نشان می دهد . سوراخهای موجود در ردیفهای کناری در بالا و پایین برد بورد بصورت افقی تا وسط برد بورد بهم متصل هستند . ردیفهای قرار گرفته در دو طرف شیار وسط بردبورد بصورت عمودی دارای اتصال هستند . خطوط قرمز رنگ روی شکل نشان دهندهٔ چگونگی این اتصالات است .



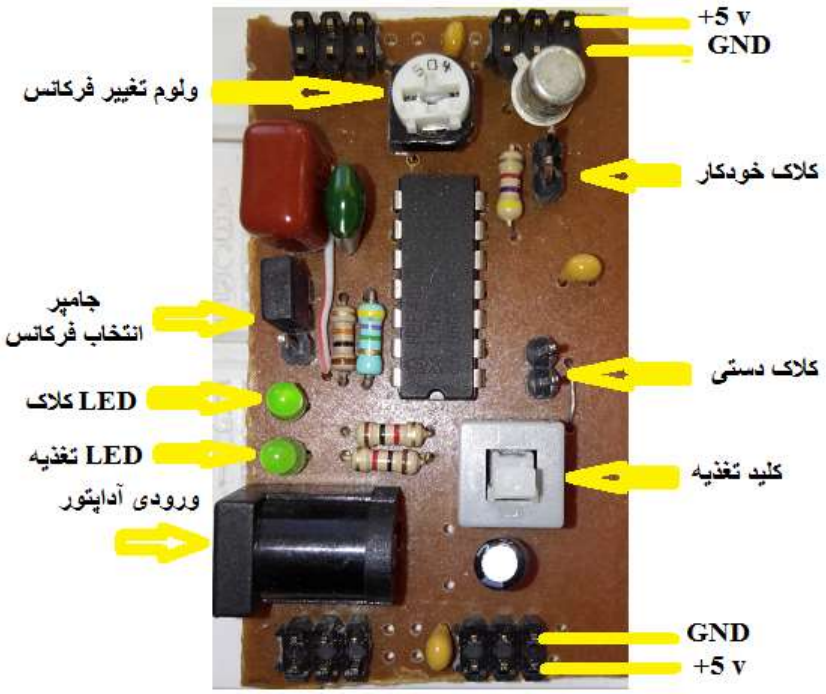
معمولاً ردیفهای دو طرف بردبورد در بالا و پایین بعنوان ردیفهای تغذیه مورد استفاده قرار میگیرد . همانطور که در شکل بعدی می بینید یکی از ردیفهای بالا برای تغذیهٔ مثبت و ردیف دیگر بعنوان ردیف زمین و یا صفر در نظر گرفته شده سپس با استفاده از دو سیم دو ردیف دیگر بردبورد نیز به تغذیه مثبت و زمین متصل گردیده است.در این صورت شما می توانید از دو طرف آی سی و از نزدیکترین نقطه به ولتاژ مثبت و زمین دسترسی داشته باشید. سعی کنید در تمام آزمایشات از بردبورد به همین صورت استفاده کنید.



برای قرار دادن آی سی روی بردبورد همانند شکل بالا حتماً باید از سوراخهای دو طرف شیار وسط برد بورد استفاده کرد تا پایه های دو طرف آی سی اتصال کوتاه نشوند. توجه داشته باشید برای بستن مدار و استفاده از آی سی ابتدا پایه های تغذیه مثبت و زمین را به ردیفهای مربوطه در بالا و یا پایین برد بورد متصل نمایید . برای تحریک ورودیهای مدار برای منطق صفر و یا یک ، میتوان بترتیب از اتصال پایه ها به ردیف زمین و یا مثبت استفاده نمود.

استفاده از بورد تغذیه

برای راحتی کار و در صورت عدم دسترسی به منبع تغذیه می توان از بورد تغذیه که برای همین منظور طراحی شده است استفاده نمود .این بورد مستقیماً روی بردبورد قرار می گیرد و توانایی تامین تغذیه ۵ ولت همراه با پالس مربعی TTL را دارا می باشد.



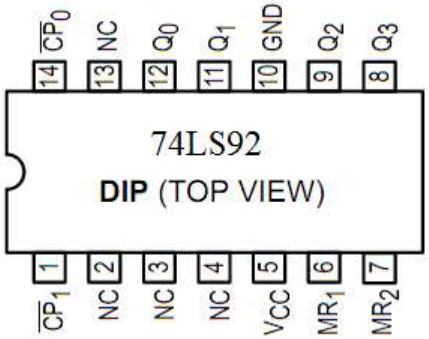
هنگام استفاده از این بورد باید به LED تغذیه و پالس توجه داشته باشید در هنگام روشن بودن بورد باید LED تغذیه در حالت روشن باشد . خاموش بودن این LED ، نشانگر وجود اتصال کوتاه در مدار خواهد بود در این صورت کلید منبع را در حالت خاموش قرار داده و نسبت به رفع عیب مدار اقدام کنید .

خواندن راهنمای آی سی(Datasheet)

برای انجام آزمایش با یک تراشه ابتدا باید با کارکرد آن و چیدمان پایه های آن آشنا شد . برای این منظور سازندگان آی سی راهنمای (DataSheet) آنرا در اختیار کاربران قرار می دهند . راحتترین راه برای دسترسی به این راهنما استفاده از اینترنت است . در آزمایشگاه راهنمای تراشه هایی که در آزمایشات مختلف با آنها سر و کار داریم در قالب یک نرم افزار در اختیار دانشجویان قرار می گیرد . همچنین در داخل دستور کار نیز ، این راهنما در انتهای هر دستور آزمایش قرار گرفته است. اکنون چند اصطلاح مهم که در راهنمای تراشه ها وجو دارند را توضیح میدهیم.

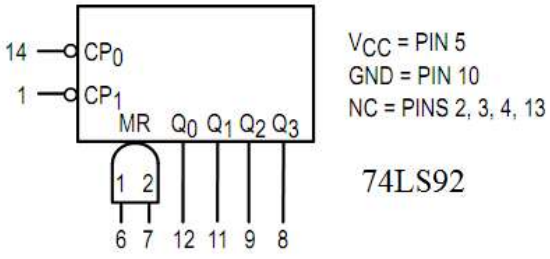
۱- شکل ظاهری تراشه (CONNECTION DIAGRAM)

شکل ظاهری تراشه با ذکر شماره و اسم هر پایه در CONNECTION DIAGRAM تراشه نمایش داده می شود . اگر از قبل با عملکرد این آی سی آشنا باشید این شکل برای بستن مدار و استفاده از این آی سی کافی خواهد بود و گر نه به اطلاعات بیشتری نیاز خواهید داشت . بعنوان مثال در شکل زیر شکل ظاهری تراشهٔ ۷۴۹۲ نشان داده شده است .



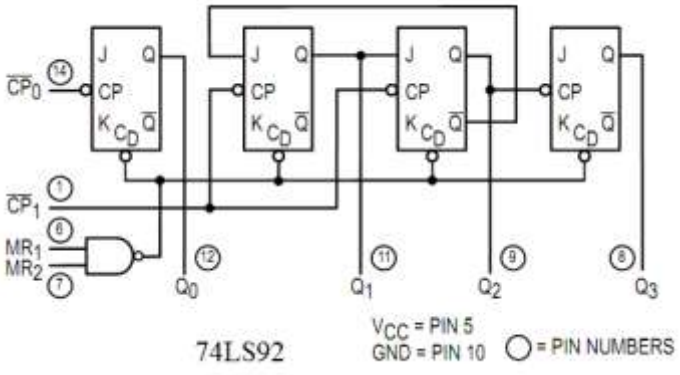
۲- نماد منطقی (LOGIC SYMBOL)

معمولاً این شکل از نمایش آی سی ، مناسب ترسیم مدارات منطقی روی کاغذ و یا نرم افزارهای الکترونیک است. در این شکل هر پایه با ذکر شمارهٔ پایه و نام آن آورده می شود . البته محل قرار گیری پایه با شکل واقعی آن تفاوت دارد . شکل زیر نماد منطقی تراشهٔ ۷۴۹۲ است . معمولا در این شکل ، پایه های تغذیه آورده نمی شود. اسم و شمارهٔ هر پایه در این شکل را با شکل قبلی مقایسه کنید.



۳- LOGIC DIAGRAM

در این شکل ، جزئیات مدار در سطح گیت و فلیپ فلاپ نشان داده می شود . شکل زیر مدار داخلی تراشهٔ ۷۴۹۲ را نشان می دهد .



۴- TRUTH TABLE (جدول درستی)

در این جدول ، عملکرد خروجیهای آی سی با توجه به حالات مختلف ورودی نشان داده می شود. شکلهای بالا و جدول درستی ، تقریباً تمامی اطلاعاتی که برای استفاده از یک آی سی لازم است را در اختیار کاربر قرار می دهد. جدول زیر نیز مربوط به تراشهٔ ۷۴۹۰ است با کنارهم گذاشتن شکلهای مربوط به این تراشه و جدول اخیر می توان از کاربرد این آی سی و همچنین وظایف هر یک از پایه های آن مطلع شد .

LS92
TRUTH TABLE

COUNT	OUTPUT			
	Q_0	Q_1	Q_2	Q_3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	L	L	H
7	H	L	L	H
8	L	H	L	H
9	H	H	L	H
10	L	L	H	H
11	H	L	H	H

NOTE: Output Q_0 is connected to Input CP_1.

آزمایش اول

آشنایی با تراشه‌های منطقی

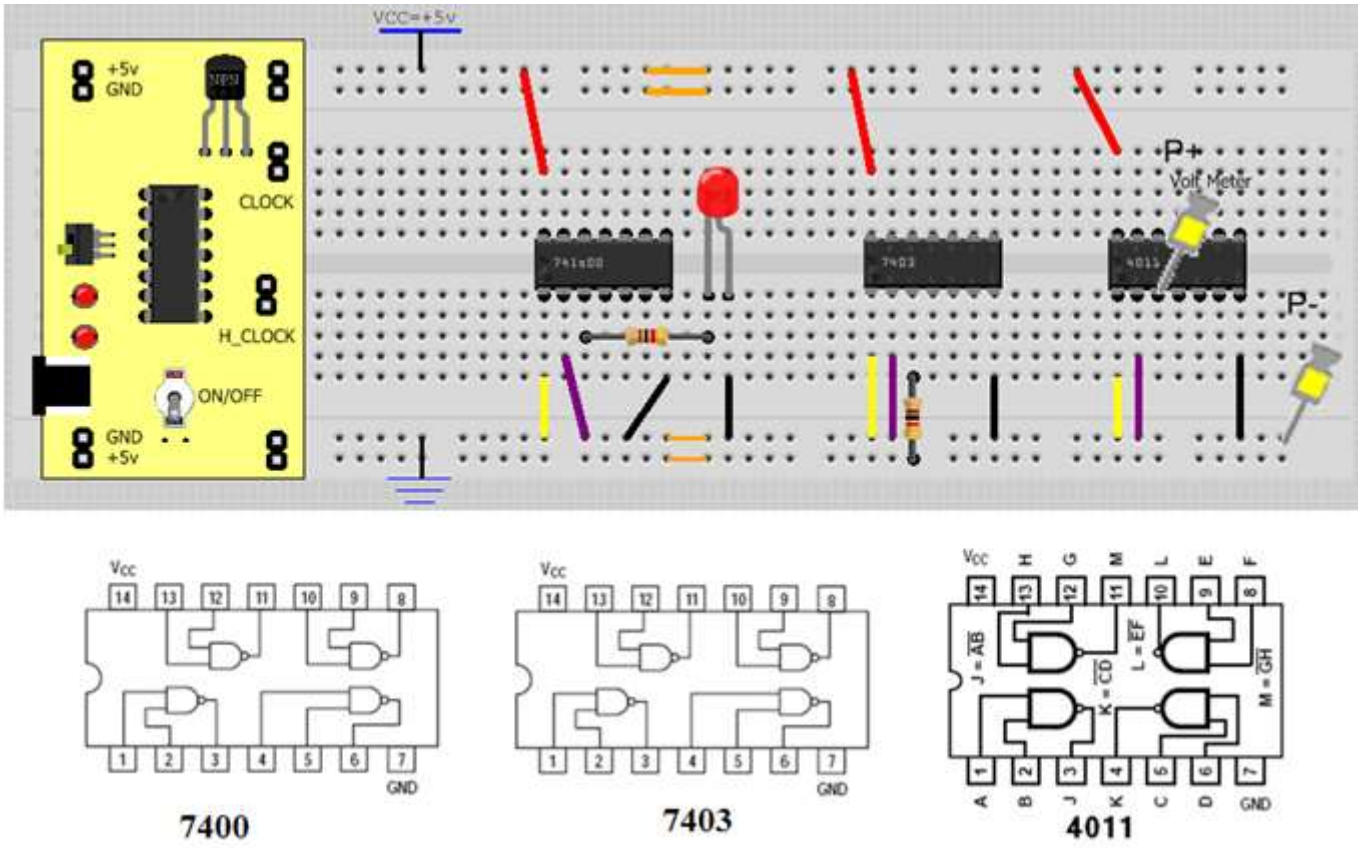
وسائل مورد نیاز بِردبورد ، تراشه‌های ۷۴۰۰ (NAND)، ۷۴۰۳ (NAND) و ۴۰۱۱ (NAND CMOS)

آزمایش ۱-۱

در این آزمایش یک دروازهٔ NAND در دو خانوادهٔ TTL و CMOS با دو نوع خروجی متفاوت را مورد بررسی قرار میدهیم . تراشه ۷۴۰۰و ۷۴۰۳ هر دو از خانوادهٔ TTL و هر کدام حاوی چهار عدد دروازهٔ NAND می باشد . اولی دارای خروجی استاندارد (totem pole) و دومی دارای خروجی کلکتور باز است . تراشهٔ ۴۰۱۱ نیز دارای چهار عدد دروازهٔ NAND از خانوادهٔ CMOS است .

الف - ابتدا با مراجعه به شکل ۱ (پایین شکل) با پیکر بندی داخلی آی سی های ۷۴۰۰ ، ۷۴۰۳ و ۴۰۱۱ آشنا شوید. به پایه های تغذیهٔ مثبت و زمین در هر آی سی توجه کنید .

سپس طبق شکل (۱) آی سی ها را بصورت مناسب و بترتیب روی بردبورد قرار داده و سیم بندی را انجام دهید . اکنون بصورت همزمان و طبق جدول ۱ با تغییر ورودیها ، خروجی هر سه گیت را با استفاده از ولت متر اندازه گیری کرده و نتایج را در جدول(۱) یادداشت نمایید .



شکل ۱- مدار آزمایش ۱-۱ و راهنمای تراشه ها

	جدول (۱) : نتایج آزمایش ۱-۱									
	7400			7403				4011		
ورودی		On/off	0/1	بدون مقاومت		با مقاومت				
A	B	ولتاژ	LED	منطق	ولتاژ	منطق	ولتاژ	منطق	ولتاژ	منطق
0	0									
0	1									
1	0									
1	1									
0	باز									
1	باز									

جدول ۱

گزارش کار

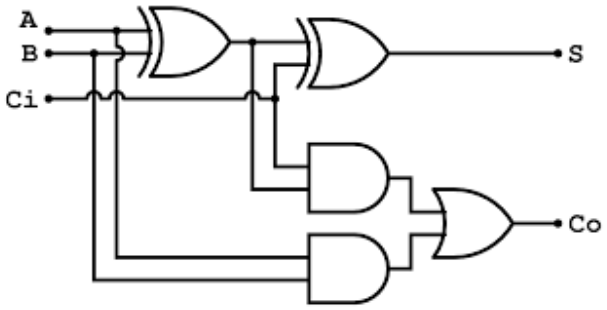
سؤال (۱) : مقدار مقاومت بین خروجی و LED بر اساس چه ملاحظاتی انتخاب میشود؟ رابطهٔ آنرا بدست آورده و در گزارش کار بیاورید .

سؤال(۲) : در مورد سه نوع خروجی استاندارد ، کلکتور باز و سه حالته تحقیق کنید و نتایج آنرا در چند خط در گزارش کار بنویسید . با توجه به نتایج تحقیق ، مقادیر بدست آمده برای آی سی ۷۴۰۳ را توجیه کنید.

سوال (۳) : ورودی باز در هر یک از خانوادهٔ TTL و CMOS دارای چه منطقی است ؟در مورد مدار داخلی یک گیت NAND خانواده TTL و CMOS تحقیق کرده و مدار آنرا در گزارش کار بیاورید. با توجه به مدار گیت در مورد نتایج دو حالت آخر (حالت ورودی باز) توضیح دهید.

پروژه (۱) : تحلیل مدار با پروتئوس

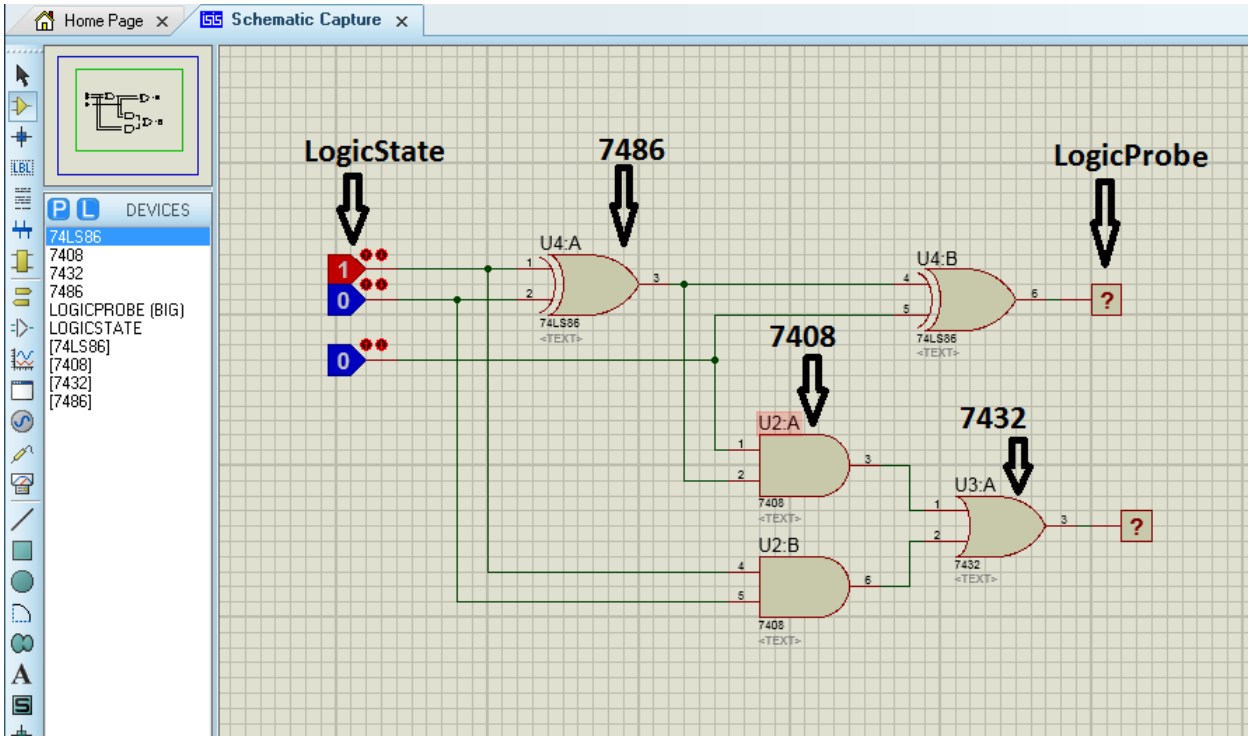
شکل ۲ مدار یک جمع کنندهٔ کامل یک بیتی را نشان می دهد . در محیط پروتئوس مدار را ببندید (شکل ۳) و مقادیر جدول زیر را به ورودی اعمال کرده خروجی را مشاهده کنید. فایل مدار را بعنوان ضمیمه گزارش کار تحویل دهید.



شکل ۲

Ci	A	B	Sum	Co
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	1	0		
1	1	1		

جدول ۲



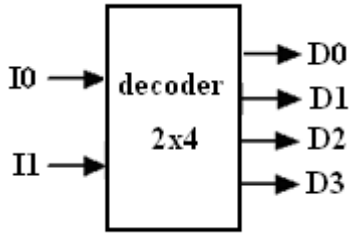
شکل ۳ – مدار جمع کننده در محیط پروتئوس

آزمایش دوم

آشنایی با مدارهای ترکیبی

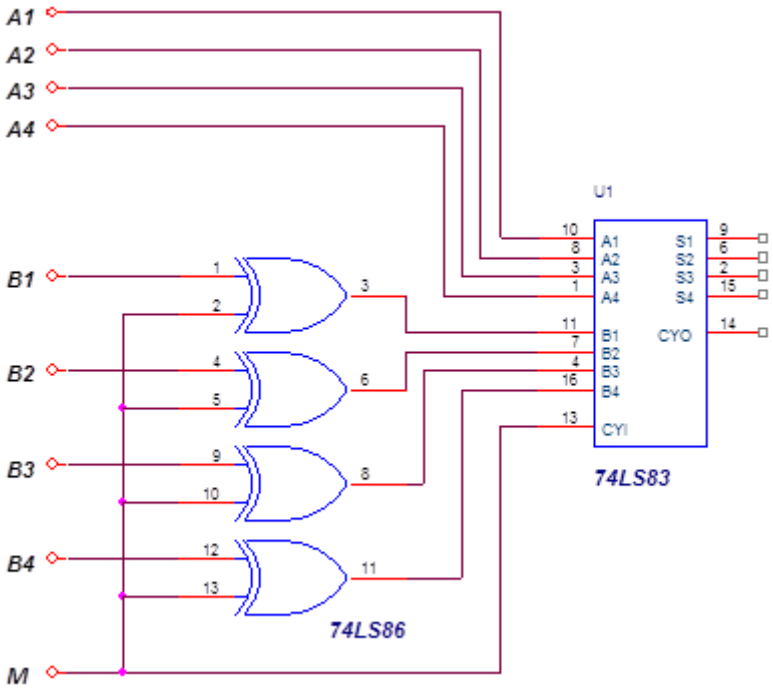
پیش گزارش :

۱- مدار دیکودر ۲ به ۴ مطابق شکل و جدول زیر را با استفاده گیتهای AND (۷۴۰۸) و NAND (۷۴۰۰) طراحی کنید. مراحل طراحی مانند جدول درستی ، جداول ساده سازی و در نهایت شکل مدار را روی کاغذ بیاورید.



I1	I0	D3	D2	D1	D0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

۲- شکل زیر یک مدار جمع کننده/تفریق کننده است. در حالت M=0 عمل جمع و در حالت M=1 عمل تفریق انجام می شود. چگونگی عملکرد مدار را توضیح دهید. نقش گیتهای XOR چیست؟(شکل راهنمای تراشهٔ جمع کنندهٔ چهار بیتی ۷۴۸۳ در صفحهٔ بعد آمده است)



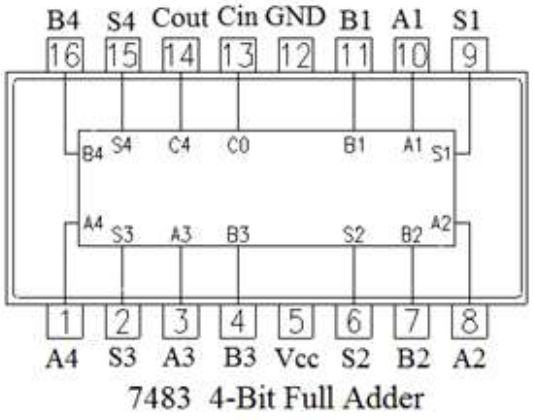
آزمایش۲-۱

با استفاده از مدار طراحی شده در پیش گزارش ، دیکودر ۲ به ۴ را روی بردبورد ببندید و سپس آنرا مورد آزمایش قرار داده و درستی جدول حالات آنرا بررسی کنید .

آزمایش۲-۲

مدار جمع کننده و تفریق کننده چهار بیتی را روی برد بورد ببندید. با توجه به مقادیر ورودی در جدول زیر، نتایج خروجی را یادداشت کنید.

مدار را بسته و با مقدار دهی به ورودیها طبق جدول زیر ، مقادیر خروجی را مشاهده و در جدول یادداشت نمایید . خروجیها را با LED نمایش دهید .



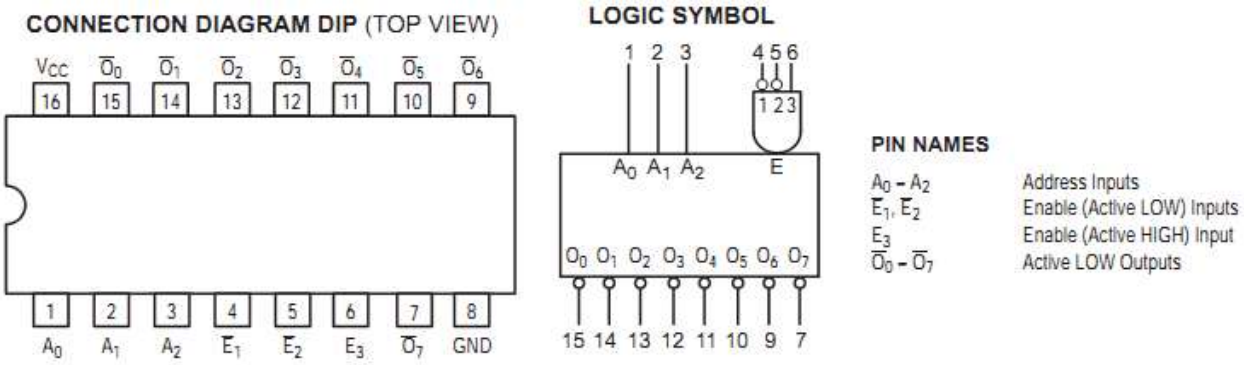
M	A	B	Sum	Co
0	0011	0010		
1	0011	0010		
0	0010	0011		
1	0010	0011		
0	1001	1000		
1	1001	1000		

گزارش کار

پروژه :

تراشهٔ ۷۴۱۳۸ یک دیکودر سه به هشت است. با مراجعه به راهنمای این تراشه با عملکرد آن آشنا شوید .

با استفاده از تراشهٔ ۷۴۱۳۸ و گیتهای مناسب یک دیکودر چهار به شانزده طراحی کنید . مدار طرح شده را توسط نرم افزار پروتئوس شبیه سازی کنید . فایل مدار را بعنوان گزارش کار تحویل دهید.



آزمایش سوم

آشنایی با تراشه های شمارنده
و کدبردارده‌دهی به هفت قسمتی

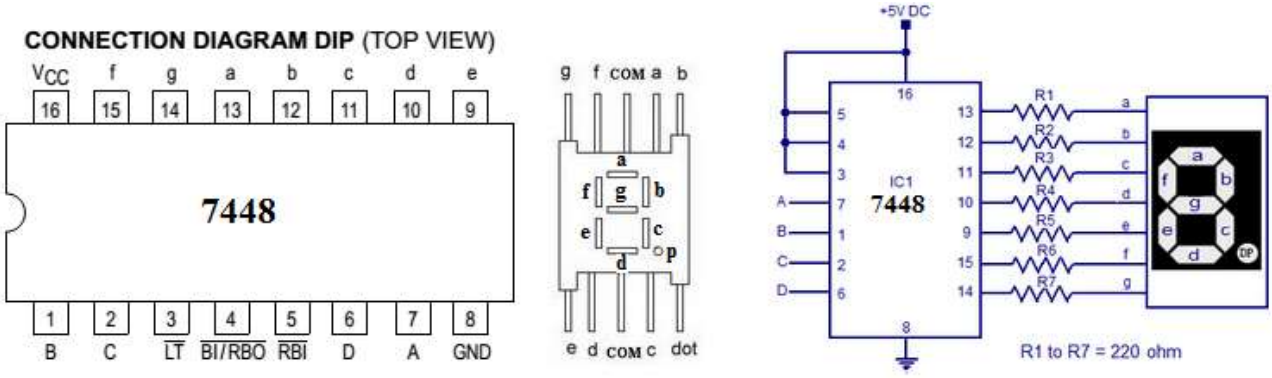
آزمایش۴-۱

الف – تراشه ۷۴۴۷ یا ۷۴۴۸ را به یک نمایشگر هفت قسمتی همانند شکل زیر متصل کرده و به ازاء حالات مختلف ورودی (۰۰۰۰ تا ۱۱۱۱) ، علائم مشخص شده روی نمایشگر را یادداشت کنید . توجه داشته باشید قرار دادن مقاومت بین تراشه و نمایشگر جهت جلوگیری از سوختن نمایشگر و تراشه الزامی است .

ب – پایه شماره ۳ تراشه (Lamp Test) را به ولتاژ LOW متصل کنید و به ازاء حالات مختلف ورودی ، تغییرات خروجی را مشاهده کنید .

ج – پایه شماره ۵ تراشه (blank-in) و پایهٔ شمارهٔ ۴ (blank out)را از ولتاژ ۵ ولت جدا کنید سپس پایه ۵ را به زمین و پایه ۴ را به ولت متر متصل کنید . به ازاء عدد ورودی صفر و یک عدد ورودی دلخواه دیگر ، اَشکال نمایش داده شده توسط نمایشگر وهمچنین مقدار ولت متر را یادداشت نمایید .(مدار را برای آزمایش بعد نگهدارید)

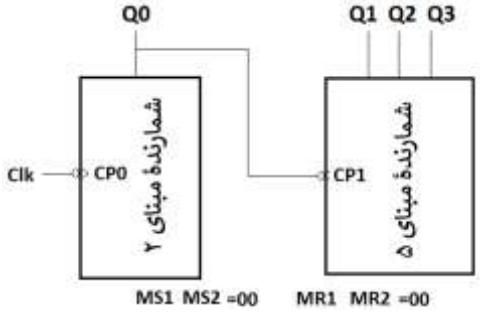
برای آگاهی از کاربرد این حالت و این دو پایه به سؤال ۳ دستور کار توجه کنید .



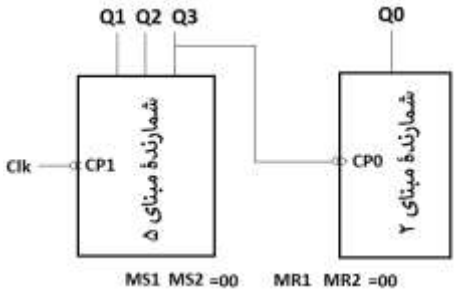
آزمایش۴-۲

تراشهٔ ۷۴۹۰ می‌تواند بصورت یک شمارندهٔ مبنای ۱۰ ناهمگام عمل کند . با مراجعه به کاتالوگ این تراشه ، با شیوهٔ کار آن آشنا شوید .

۱- با استفاده از تراشهٔ ۷۴۹۰ یک شمارندهٔ مبنای ۱۰ را پیاده سازی کنید . خروجی را توسط مدار7_segment مشاهده نمایید .



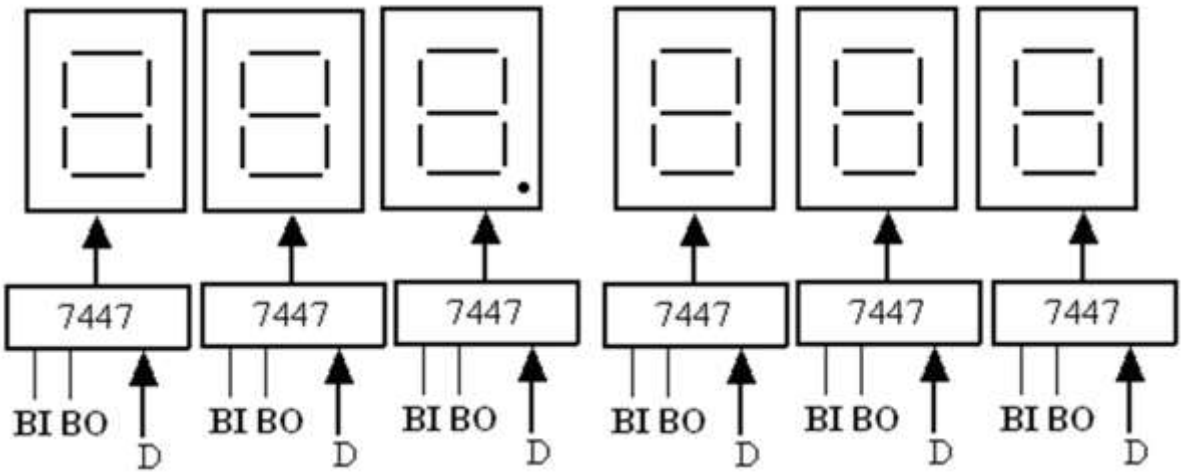
۲- با تغیر ترکیب مدار ۷۴۹۰، شمارنده را به یک شمارندۀ زوج و فرد تبدیل کنید.



گزارش کار

سؤال :

۱- جهت نمایش دادن یک عدد اعشاری از شش نمایشگر استفاده شده است که طبق شکل زیر سه عدد آن ارقام اعشاری و سه نمایشگر دیگر آن ارقام صحیح را نمایش میدهند . اگر بخواهیم صفرهای قبل از عدد صحیح و صفرهای بعد از عدد اعشاری را نمایش ندهیم ، اتصالات لازم بین تراشه‌ها را طراحی و رسم نمایید .



۲- چگونه می توان شمارنده در مبنای مختلف از ۲ تا ۱۰ را توسط تراشۀ ۷۴۹۰ ساخت؟ بطور مثال شکل مدار را برای شمارندۀ مبنای هشت (شمارش از صفر تا هفت) رسم کنید.(همانند شکلهای آزمایش ۴-۲)

پروژه :

با استفاده از تعدادی تراشه ۷۴۹۰ و نمایشگر سون سگمنت یک ثانیه شمار با دقت دهم ثانیه در محیط پروتئوس بسازید. این ثانیه شمار از 00.0 تا 99.9 ثانیه را شمارش می کند.

توجه: کتابخانه نرم افزار پروتئوس دارای سون سگمنت دهدهی با چهار ورودی است که مستقیما به خروجی شمارنده متصل می شود و نیاز به تراشه ۷۴۴۸ نیست.

MOTOROLA

BCD TO 7-SEGMENT DECODER

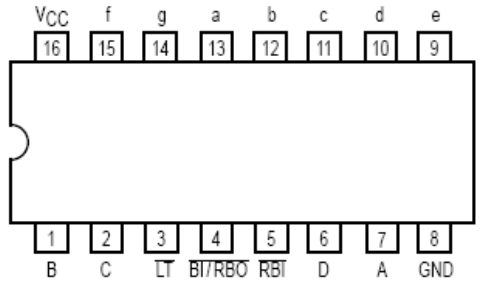
The SN54/74LS48 is a BCD to 7-Segment Decoder consisting of NAND gates, input buffers and seven AND-OR-INVERT gates. Seven NAND gates and one driver are connected in pairs to make BCD data and its complement available to the seven decoding AND-OR-INVERT gates. The remaining NAND gate and three input buffers provide lamp test, blanking input/ripple-blanking input for the LS48.

The circuit accepts 4-bit binary-coded-decimal (BCD) and, depending on the state of the auxiliary inputs, decodes this data to drive other components. The relative positive logic output levels, as well as conditions required at the auxiliary inputs, are shown in the truth tables.

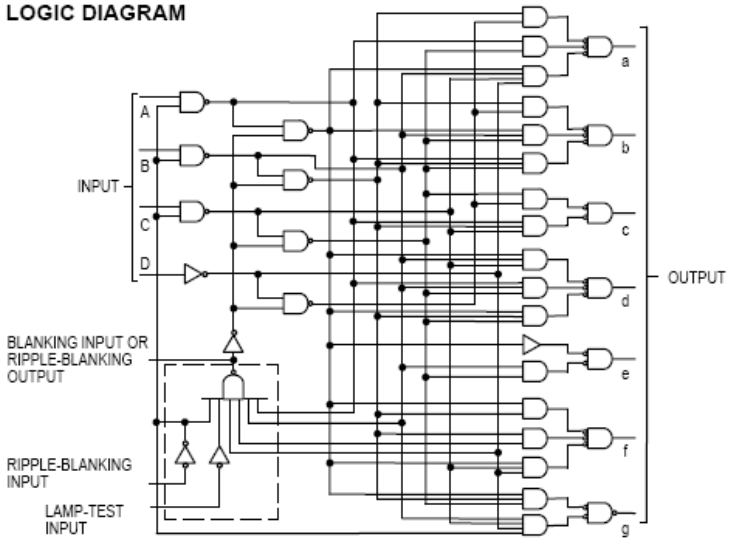
The LS48 circuit incorporates automatic leading and/or trailing edge zero-blanking control (RBI and RBO). Lamp Test (LT) may be activated any time when the BI/RBO node is HIGH. Both devices contain an overriding blanking input (BI) which can be used to control the lamp intensity by varying the frequency and duty cycle of the BI input signal or to inhibit the outputs.

- Lamp Intensity Modulation Capability (BI/RBO)
- Internal Pull-Ups Eliminate Need for External Resistors
- Input Clamp Diodes Eliminate High-Speed Termination Effects

CONNECTION DIAGRAM DIP (TOP VIEW)



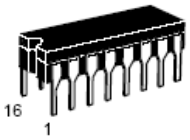
LOGIC DIAGRAM



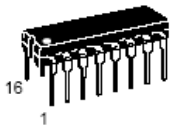
SN54/74LS48

BCD TO 7-SEGMENT DECODER

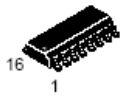
LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 620-09



N SUFFIX
PLASTIC
CASE 648-08

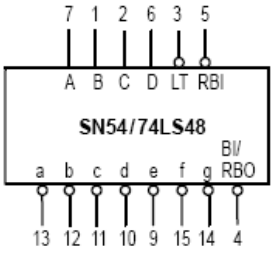


D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC

LOGIC SYMBOL



V_{CC} = PIN 16
GND = PIN 8

SN54/74LS48

PIN NAMES

		LOADING (Note a)	
		HIGH	LOW
A, B, C, D	BCD Inputs	0.5 U.L.	0.25 U.L.
$\overline{RBI}$	Ripple-Blanking (Active Low) Input	0.5 U.L.	0.25 U.L.
$\overline{LT}$	Lamp-Test (Active Low) Input	0.5 U.L.	0.25 U.L.
$\overline{BI/RBO}$	Blanking Input or Ripple-Blanking Output (Active Low)	0.5 U.L. 1.2 U.L.	0.75 U.L. 2(1) U.L.
$\overline{BI}$	Blanking (Active Low) Input	0.5 U.L.	0.25 U.L.
		Open-Collector	3.75 (1.25) U.L. (48)

NOTES:
a) Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW
b) Outut current measured at V_{OUT} = 0.5 V
Output LOW drive factor is SN54LS/74LS48: 1.25 U.L. for Military (54), 3.75 U.L. for Commercial (74).

NUMERICAL DESIGNATIONS — RESULTANT DISPLAYS

TRUTH TABLE
SN54/74LS48

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

NOTES:
(1) BI/RBO is wired-AND logic serving as blanking input ($\overline{BI}$) and/or ripple-blanking output ($\overline{RBO}$). The blanking out ($\overline{BI}$) must be open or held at a HIGH level when output functions 0 through 15 are desired, and ripple-blanking input ($\overline{RBI}$) must be open or at a HIGH level if blanking of a decimal 0 is not desired. X=input may be HIGH or LOW.
(2) When a LOW level is applied to the blanking input (forced condition) all segment outputs go to a LOW level, regardless of the state of any other input condition.
(3) When ripple-blanking input ($\overline{RBI}$) and inputs A, B, C, and D are at LOW level, with the lamp test input at HIGH level, all segment outputs go to a HIGH level and the ripple-blanking output ($\overline{RBO}$) goes to a LOW level (response condition).
(4) When the blanking input/ripple-blanking output ($\overline{BI/RBO}$) is open or held at a HIGH level, and a LOW level is applied to lamp-test input, all segment outputs go to a LOW level.

MOTOROLA

DECADE COUNTER; DIVIDE-BY-TWELVE COUNTER; 4-BIT BINARY COUNTER

SN54/74LS90
SN54/74LS92
SN54/74LS93

DECADE COUNTER;
DIVIDE-BY-TWELVE COUNTER;
4-BIT BINARY COUNTER

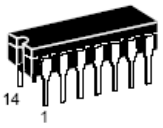
LOW POWER SCHOTTKY

The SN54/74LS90, SN54/74LS92 and SN54/74LS93 are high-speed 4-bit ripple type counters partitioned into two sections. Each counter has a divide-by-two section and either a divide-by-five (LS90), divide-by-six (LS92) or divide-by-eight (LS93) section which are triggered by a HIGH-to-LOW transition on the clock inputs. Each section can be used separately or tied together (Q to $\overline{CP}$) to form BCD, bi-quinary, modulo-12, or modulo-16 counters. All of the counters have a 2-input gated Master Reset (Clear), and the LS90 also has a 2-input gated Master Set (Preset 9).

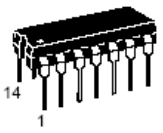
- Low Power Consumption . . . Typically 45 mW
- High Count Rates . . . Typically 42 MHz
- Choice of Counting Modes . . . BCD, Bi-Quinary, Divide-by-Twelve, Binary
- Input Clamp Diodes Limit High Speed Termination Effects

PIN NAMES		LOADING (Note a) HIGH	LOW
$\overline{CP}_0$	Clock (Active LOW going edge) Input to ÷2 Section	0.5 U.L.	1.5 U.L.
$\overline{CP}_1$	Clock (Active LOW going edge) Input to ÷5 Section (LS90), ÷6 Section (LS92)	0.5 U.L.	2.0 U.L.
$\overline{CP}_1$	Clock (Active LOW going edge) Input to ÷8 Section (LS93)	0.5 U.L.	1.0 U.L.
MR_1, MR_2	Master Reset (Clear) Inputs	0.5 U.L.	0.25 U.L.
MS_1, MS_2	Master Set (Preset-9, LS90) Inputs	0.5 U.L.	0.25 U.L.
Q_0	Output from ÷2 Section (Notes b & c)	10 U.L.	5 (2.5) U.L.
Q_1, Q_2, Q_3	Outputs from ÷5 (LS90), ÷6 (LS92), ÷8 (LS93) Sections (Note b)	10 U.L.	5 (2.5) U.L.

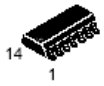
NOTES:
a. 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military, (54) and 5 U.L. for commercial (74) Temperature Ranges.
c. The Q_0 Outputs are guaranteed to drive the full fan-out plus the $\overline{CP}_1$ input of the device.
d. To insure proper operation the rise (t_r) and fall time (t_f) of the clock must be less than 100 ns.



J SUFFIX
CERAMIC
CASE 632-08



N SUFFIX
PLASTIC
CASE 646-06



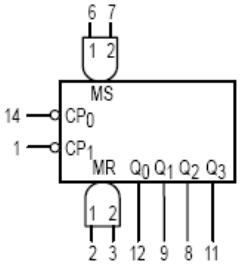
D SUFFIX
SOIC
CASE 751A-02

ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC

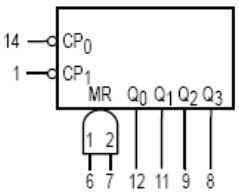
LOGIC SYMBOL

LS90



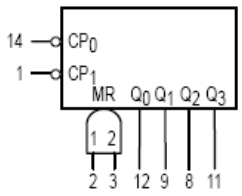
V_{CC} = PIN 5
GND = PIN 10
NC = PINS 4, 13

LS92



V_{CC} = PIN 5
GND = PIN 10
NC = PINS 2, 3, 4, 13

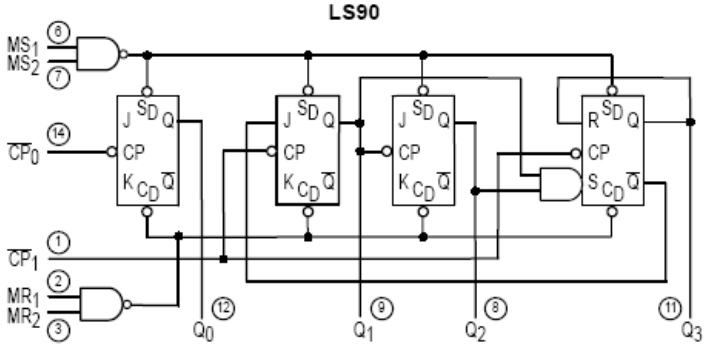
LS93



V_{CC} = PIN 5
GND = PIN 10
NC = PIN 4, 6, 7, 13

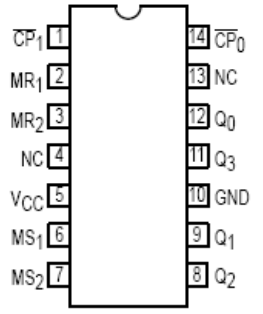
LOGIC DIAGRAM

LS90



○ = PIN NUMBERS
V_{CC} = PIN 5
GND = PIN 10

CONNECTION DIAGRAM
DIP (TOP VIEW)

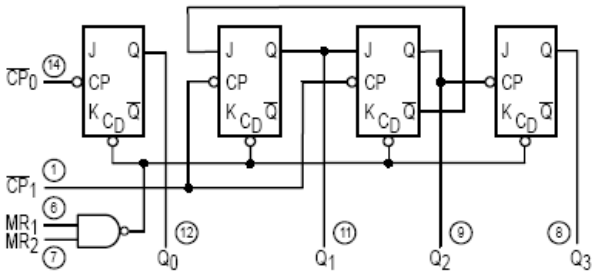


NC = NO INTERNAL CONNECTION

NOTE:
The Flatpak version has the same pinouts (Connection Diagram) as the Dual In-Line Package.

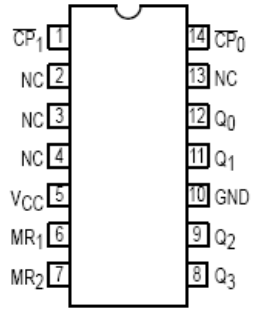
LOGIC DIAGRAM

LS92



○ = PIN NUMBERS
V_{CC} = PIN 5
GND = PIN 10

CONNECTION DIAGRAM
DIP (TOP VIEW)

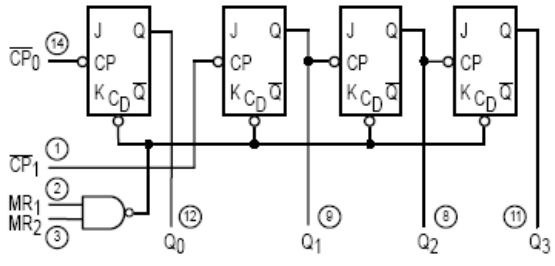


NC = NO INTERNAL CONNECTION

NOTE:
The Flatpak version has the same pinouts (Connection Diagram) as the Dual In-Line Package.

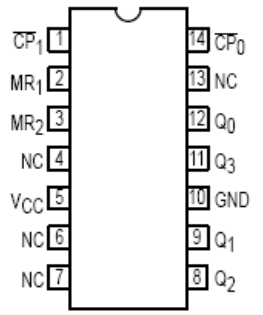
LOGIC DIAGRAM

LS93



○ = PIN NUMBERS
V_{CC} = PIN 5
GND = PIN 10

CONNECTION DIAGRAM
DIP (TOP VIEW)



NC = NO INTERNAL CONNECTION

NOTE:
The Flatpak version has the same pinouts (Connection Diagram) as the Dual In-Line Package.

FUNCTIONAL DESCRIPTION

The LS90, LS92, and LS93 are 4-bit ripple type Decade, Divide-By-Twelve, and Binary Counters respectively. Each device consists of four master/slave flip-flops which are internally connected to provide a divide-by-two section and a divide-by-five (LS90), divide-by-six (LS92), or divide-by-eight (LS93) section. Each section has a separate clock input which initiates state changes of the counter on the HIGH-to-LOW clock transition. State changes of the Q outputs do not occur simultaneously because of internal ripple delays. Therefore, decoded output signals are subject to decoding spikes and should not be used for clocks or strobes. The Q_0 output of each device is designed and specified to drive the rated fan-out plus the $\overline{CP}_1$ input of the device.

A gated AND asynchronous Master Reset ($MR_1 \bullet MR_2$) is provided on all counters which overrides and clocks and resets (clears) all the flip-flops. A gated AND asynchronous Master Set ($MS_1 \bullet MS_2$) is provided on the LS90 which overrides the clocks and the MR inputs and sets the outputs to nine (HLLH).

Since the output from the divide-by-two section is not internally connected to the succeeding stages, the devices may be operated in various counting modes.

LS90

A. BCD Decade (8421) Counter — The $\overline{CP}_1$ input must be externally connected to the Q_0 output. The $\overline{CP}_0$ input receives the incoming count and a BCD count sequence is produced.

B. Symmetrical Bi-quinary Divide-By-Ten Counter — The Q_3 output must be externally connected to the $\overline{CP}_0$ input. The input count is then applied to the $\overline{CP}_1$ input and a divide-by-ten square wave is obtained at output Q_0.

C. Divide-By-Two and Divide-By-Five Counter — No external interconnections are required. The first flip-flop is used as a binary element for the divide-by-two function ($\overline{CP}_0$ as the input and Q_0 as the output). The $\overline{CP}_1$ input is used to obtain binary divide-by-five operation at the Q_3 output.

LS92

A. Modulo 12, Divide-By-Twelve Counter — The $\overline{CP}_1$ input must be externally connected to the Q_0 output. The $\overline{CP}_0$ input receives the incoming count and Q_3 produces a symmetrical divide-by-twelve square wave output.

B. Divide-By-Two and Divide-By-Six Counter —No external interconnections are required. The first flip-flop is used as a binary element for the divide-by-two function. The $\overline{CP}_1$ input is used to obtain divide-by-three operation at the Q_1 and Q_2 outputs and divide-by-six operation at the Q_3 output.

LS93

A. 4-Bit Ripple Counter — The output Q_0 must be externally connected to input $\overline{CP}_1$. The input count pulses are applied to input $\overline{CP}_0$. Simultaneous divisions of 2, 4, 8, and 16 are performed at the Q_0, Q_1, Q_2, and Q_3 outputs as shown in the truth table.

B. 3-Bit Ripple Counter— The input count pulses are applied to input $\overline{CP}_1$. Simultaneous frequency divisions of 2, 4, and 8 are available at the Q_1, Q_2, and Q_3 outputs. Independent use of the first flip-flop is available if the reset function coincides with reset of the 3-bit ripple-through counter.

LS90
MODE SELECTION

RESET / SET INPUTS				OUTPUTS			
MR_1	MR_2	MS_1	MS_2	Q_0	Q_1	Q_2	Q_3
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
L	X	L	X	Count			
X	L	X	L	Count			
L	X	X	L	Count			
X	L	L	X	Count			

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

LS92 AND LS93
MODE SELECTION

RESET INPUTS		OUTPUTS			
MR_1	MR_2	Q_0	Q_1	Q_2	Q_3
H	H	L	L	L	L
L	H	Count			
H	L	Count			
L	L	Count			

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

LS90
BCD COUNT SEQUENCE

COUNT	OUTPUT			
	Q_0	Q_1	Q_2	Q_3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	H	H	L
7	H	H	H	L
8	L	L	L	H
9	H	L	L	H

NOTE: Output Q_0 is connected to Input CP_1 for BCD count.

LS92
TRUTH TABLE

COUNT	OUTPUT			
	Q_0	Q_1	Q_2	Q_3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	L	L	H
7	H	L	L	H
8	L	H	L	H
9	H	H	L	H
10	L	L	H	H
11	H	L	H	H

NOTE: Output Q_0 is connected to Input CP_1.

LS93
TRUTH TABLE

COUNT	OUTPUT			
	Q_0	Q_1	Q_2	Q_3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	H	H	L
7	H	H	H	L
8	L	L	L	H
9	H	L	L	H
10	L	H	L	H
11	H	H	L	H
12	L	L	H	H
13	H	L	H	H
14	L	H	H	H
15	H	H	H	H

NOTE: Output Q_0 is connected to Input CP_1.

آزمایش چهارم

آشنایی با تراشه شیفت رجیستر

آزمایش۴-۱

تراشهٔ ۷۴۱۹۴ یک ثبات انتقالی چهار بیتی با امکانات ورودی موازی و سری و انتقال به راست و چپ می‌باشد با مراجعه به راهنمای آن با نحوهٔ کار آن آشنا شوید . در ادامه آزمایش زیر را با این تراشه انجام دهید .

۱- ابتدا مقدار بایتری ۰۱۱۰ را به ورودی موازی اعمال کنید سپس با استفاده حالت بارگذاری ، این عدد را در ثبات ذخیره کنید.

۲- با استفاده از حالت شیفت به چپ مقدار ۰۱۱۰ را به چپ شیفت دهید. با تغییر ورودی سریال شیفت به چپ به مقدار صفر و یا یک تغییرات خروجی را مشاهده کنید.

۳- با بارگذاری دوبارهٔ عدد باینری ۰۱۱۰ اینبار این مقدار را به راست شیفت دهید. با تغییر ورودی سریال شیفت به راست به مقدار صفر و یا یک تغییرات خروجی را مشاهده کنید.

آزمایش۴-۲

با استفاده از دو شیفت رجیستر 74194، یک شیفت رجیستر هشت بیتی انتقال به راست و چپ طرح کنید . مدار را ببندید و برای این مدار نیز حالات مختلف آزمایش قبل را برای عدد باینری ۰۰۰۰،۰۰۰۱ تکرار کنید.

آزمایش۴-۳

با استفاده از مدار شیفت رجیستر هشت بیتی یک شمارندهٔ جانسون هشت بیتی بسازید. ترتیب شمارش در این شمارنده بصورت زیر است:

➔ 0000,0000 –> 0000,0001 –> 0000,0011 –> 0000,0111 -> ….. –> 1111,1111 –> 1111,1110-> 1111,1100 –> 1111,1000 -> ….. –> 0000,0000

MOTOROLA

SN54/74LS194A

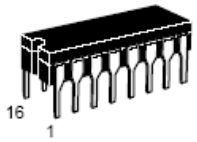
4-BIT BIDIRECTIONAL UNIVERSAL SHIFT REGISTER

The SN54/74LS194A is a High Speed 4-Bit Bidirectional Universal Shift Register. As a high speed multifunctional sequential building block, it is useful in a wide variety of applications. It may be used in serial-serial, shift left, shift right, serial-parallel, parallel-serial, and parallel-parallel data register transfers. The LS194A is similar in operation to the LS195A Universal Shift Register, with added features of shift left without external connections and hold (do nothing) modes of operation. It utilizes the Schottky diode clamped process to achieve high speeds and is fully compatible with all Motorola TTL families.

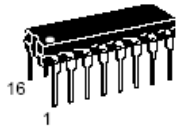
- Typical Shift Frequency of 36 MHz
- Asynchronous Master Reset
- Hold (Do Nothing) Mode
- Fully Synchronous Serial or Parallel Data Transfers
- Input Clamp Diodes Limit High Speed Termination Effects

4-BIT BIDIRECTIONAL UNIVERSAL SHIFT REGISTER

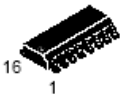
LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 620-09



N SUFFIX
PLASTIC
CASE 648-08

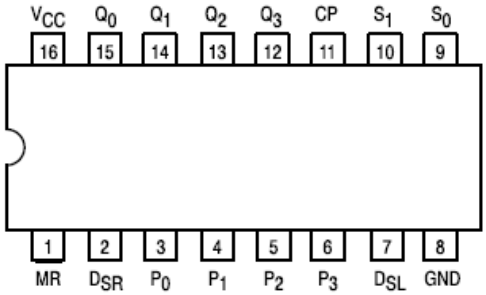


D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

SN54LSXXXJ	Ceramic
SN74LSXXXN	Plastic
SN74LSXXXD	SOIC

CONNECTION DIAGRAM DIP (TOP VIEW)

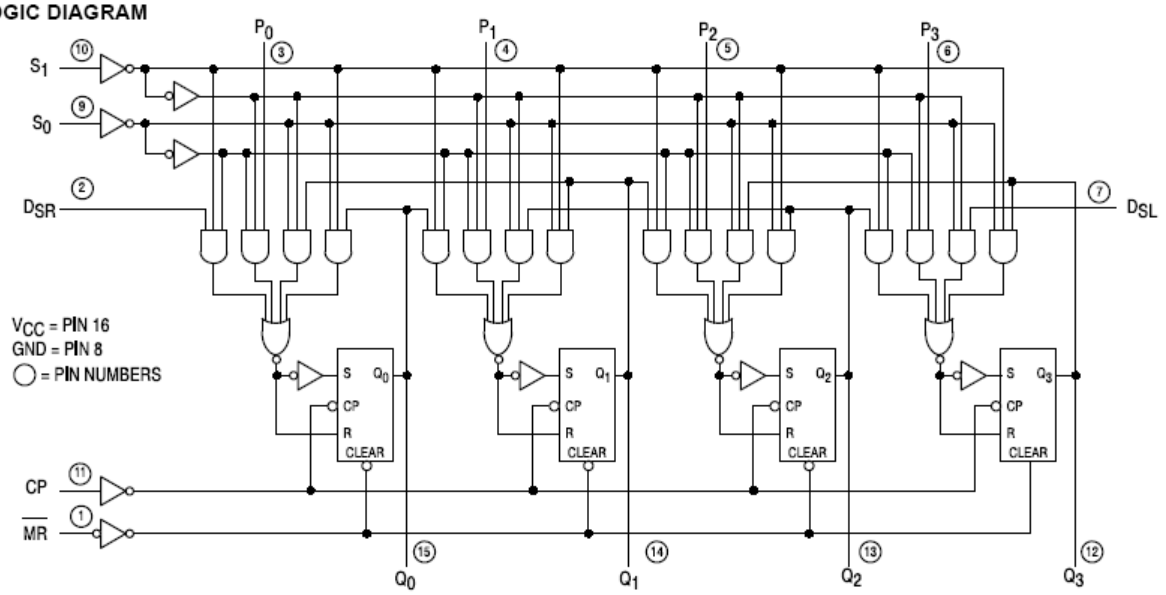


PIN NAMES

		LOADING (Note a) HIGH	LOW
S_0, S_1	Mode Control Inputs	0.5 U.L.	0.25 U.L.
P_0–P_3	Parallel Data Inputs	0.5 U.L.	0.25 U.L.
D_{SR}	Serial (Shift Right) Data Input	0.5 U.L.	0.25 U.L.
D_{SL}	Serial (Shift Left) Data Input	0.5 U.L.	0.25 U.L.
$\overline{CP}$	Clock (Active HIGH Going Edge) Input	0.5 U.L.	0.25 U.L.
$\overline{MR}$	Master Reset (Active LOW) Input	0.5 U.L.	0.25 U.L.
Q_0–Q_3	Parallel Outputs (Note b)	10 U.L.	5 (2.5) U.L.

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

LOGIC DIAGRAM



FUNCTIONAL DESCRIPTION

The Logic Diagram and Truth Table indicate the functional characteristics of the LS194A 4-Bit Bidirectional Shift Register. The LS194A is similar in operation to the Motorola LS195A Universal Shift Register when used in serial or parallel data register transfers. Some of the common features of the two devices are described below:

All data and mode control inputs are edge-triggered, responding only to the LOW to HIGH transition of the Clock (CP). The only timing restriction, therefore, is that the mode control and selected data inputs must be stable one set-up time prior to the positive transition of the clock pulse.

The register is fully synchronous, with all operations taking place in less than 15 ns (typical) making the device especially useful for implementing very high speed CPUs, or the memory buffer registers.

The four parallel data inputs (P_0, P_1, P_2, P_3) are D-type inputs. When both S_0 and S_1 are HIGH, the data appearing on P_0, P_1, P_2, and P_3 inputs is transferred to the Q_0, Q_1, Q_2, and Q_3 outputs respectively following the next LOW to HIGH transition of the clock.

The asynchronous Master Reset ($\overline{MR}$), when LOW, overrides all other input conditions and forces the Q outputs LOW.

Special logic features of the LS194A design which increase the range of application are described below:

Two mode control inputs (S_0, S_1) determine the synchronous operation of the device. As shown in the Mode Selection Table, data can be entered and shifted from left to right (shift right, Q_0 Q_1, etc.) or right to left (shift left, Q_3 Q_2, etc.), or parallel data can be entered loading all four bits of the register simultaneously. When both S_0 and S_1,are LOW, the existing data is retained in a "do nothing" mode without restricting the HIGH to LOW clock transition.

D-type serial data inputs (D_{SR}, D_{SL}) are provided on both the first and last stages to allow multistage shift right or shift left data transfers without interfering with parallel load operation.

MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS						OUTPUTS			
	MR	S_1	S_0	D_{SR}	D_{SL}	P_n	Q_0	Q_1	Q_2	Q_3
Reset	L	X	X	X	X	X	L	L	L	L
Hold	H	l	l	X	X	X	q_0	q_1	q_2	q_3
Shift Left	H	h	l	X	l	X	q_1	q_2	q_3	L
	H	h	l	X	h	X	q_1	q_2	q_3	H
Shift Right	H	l	h	l	X	X	L	q_0	q_1	q_2
	H	l	h	h	X	X	H	q_0	q_1	q_2
Parallel Load	H	h	h	X	X	P_n	P_0	P_1	P_2	P_3

L = LOW Voltage Level
H = HIGH Voltage Level
X = Don't Care
l = LOW voltage level one set-up time prior to the LOW to HIGH clock transition
h = HIGH voltage level one set-up time prior to the LOW to HIGH clock transition
p_n (q_n) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the LOW to HIGH clock transition.

بخش دوم

طراحی مدارهای منطقی بر اساس زبان برنامه نویسی وریلاگ

بسمه تعالی

مقدمه

در بخش اول آزمایشگاه با روش معمول طراحی و پیاده سازی مدارهای منطقی توسط آی سی های پایه و معمولی آشنا شدید . امروزه یکی از روشهای مرسوم و پیشرفته طراحی، استفاده از نرم افزارهای طراحی و همچنین پیاده سازی طرح توسط آی سی های برنامه پذیر است . در بخش دوم آزمایشگاه سعی می شود به صورت کلی و مختصر با این شیوه جدید آشنا شویم . برای این منظور به یک نرم افزار طراحی و همچنین یک بورد آموزشی مبتنی بر تراشه های برنامه پذیر نیاز داریم .

بورد موجود در آزمایشگاه حاوی یک آی سی برنامه پذیر CPLD ساخت شرکت ALTRA است . این شرکت در زمینهٔ تولید آی سی های FPGA و CPLD فعالیت می کند . نرم افزار مورد استفاده نیز تولید همین شرکت و برای کار با آی سی های تولیدی این کمپانی است . این نرم افزار با نام QUARTUS شناخته می شود که ویرایش جدیدی از نرم افزار قدیمی MAXPLUS همین شرکت است .

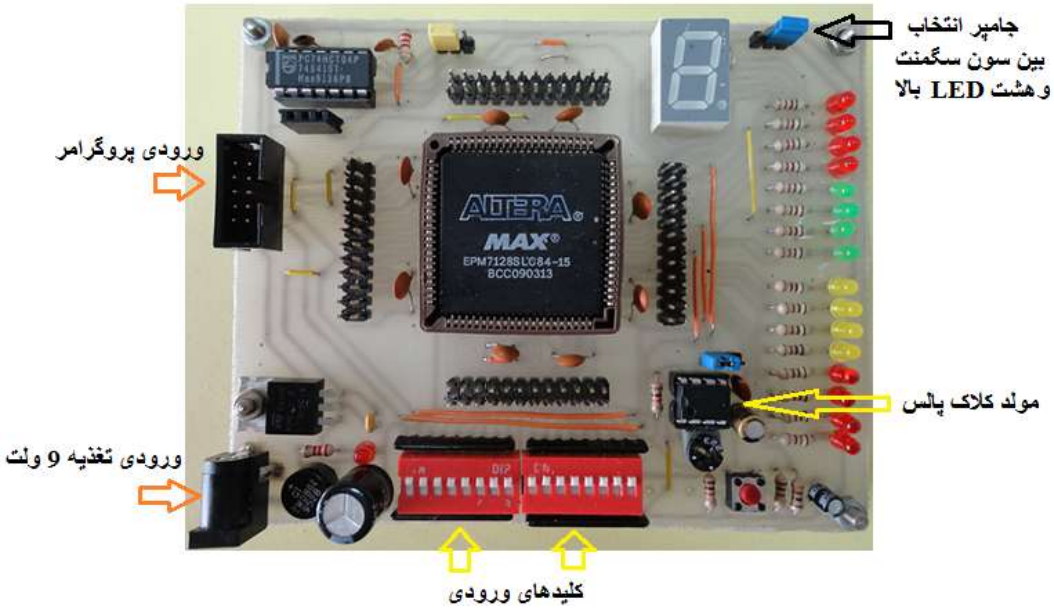
برای آشنایی دانشجویان با این نرم افزار متن جداگانه ای تهیه شده که بصورت مختصر و مفید چگونگی کار با این نرم افزار توضیح داده شده است .

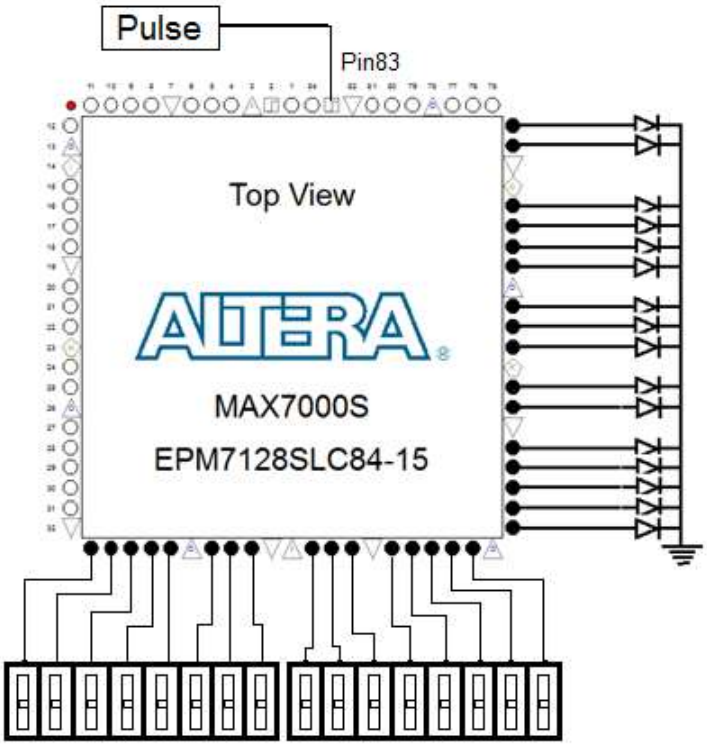
معمولا طراحی مدارهای دیجیتال توسط زبانهای برنامه نویسی که مختص این کار ساخته شده اند انجام می شوند زبان وریلاگ (Verilog) و زبان VHDL نمونه ای از این زبانهای برنامه نویسی هستند . نرم افزار کوارتوس نیز این دو زبان برنامه نویسی و طراحی را پشتیبانی می کند. همچنین در این نرم افزار می توان بصورت شماتیک نیز کار طراحی را انجام داد .

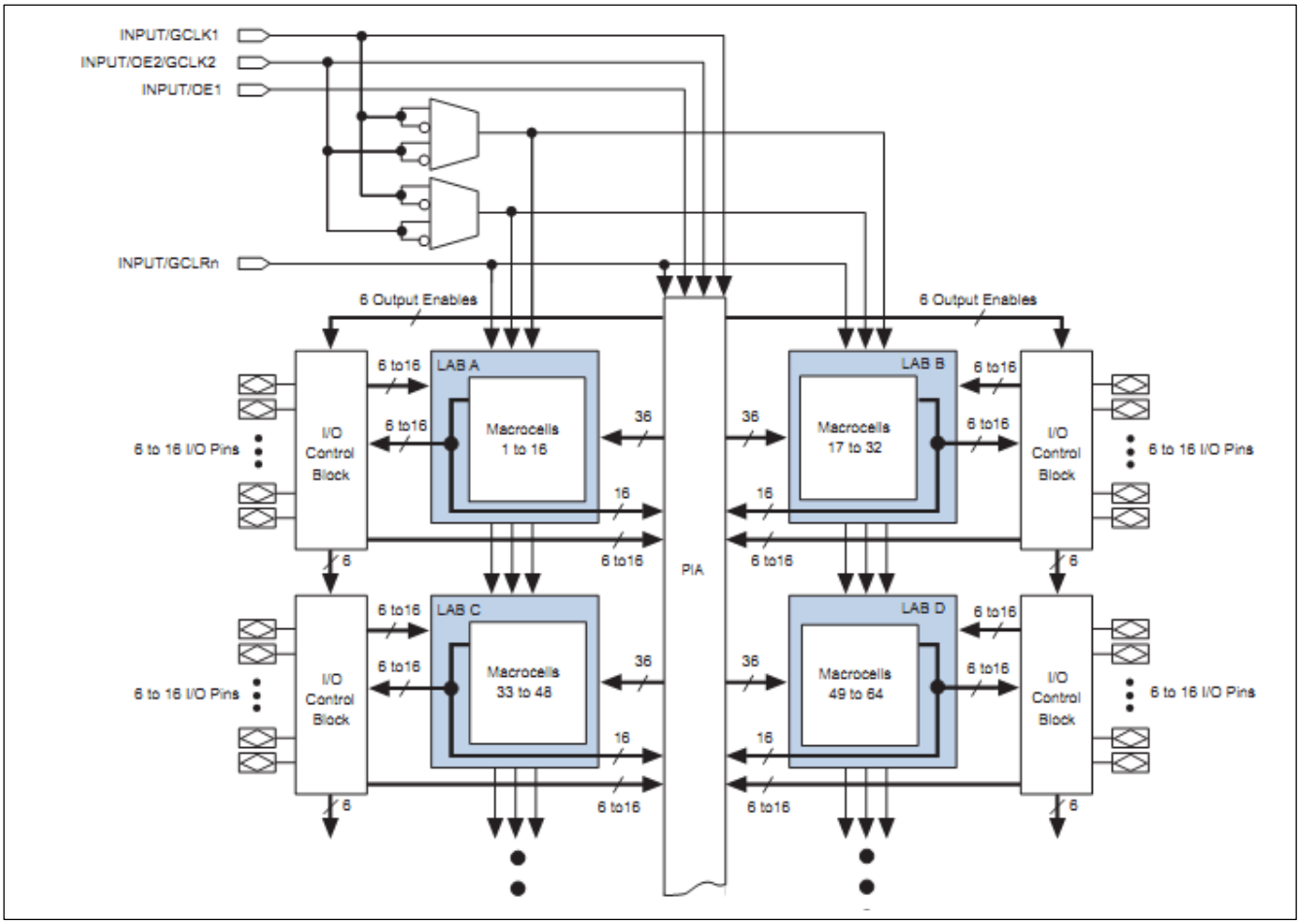
در بخش دوم دستور کار آزمایشگاه در قالب آموزش زبان وریلاگ ، آزمایشهای مورد نظر نیز انجام می شود تا در انتهای کار دانشجو مهارت مختصری برای کار با این زبان طراحی پیدا کند .

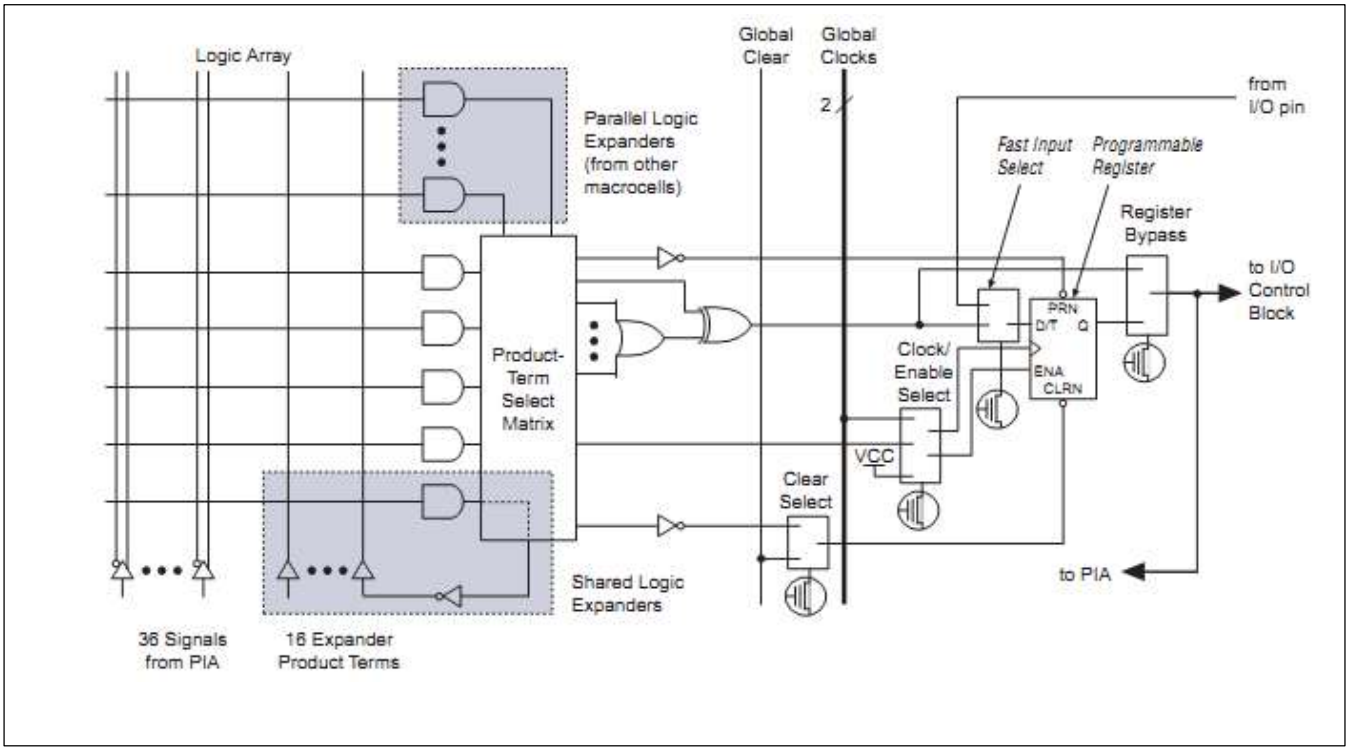
با آرزوی توفیق برای شما

مربی آزمایشگاه : محمدرضا فتاح


جامپر انتخاب
بین سون سگمنت
و هشت LED بالا
ورودی پروگرامر
مولد کلاک پالس
ورودی تغذیه 9 ولت
کلیدهای ورودی


Pulse
Pin83
Top View
MAX7000S
EPM7128SLC84-15


INPUT/GCLK1
INPUT/OE2/GCLK2
INPUT/OE1
INPUT/GCLRn
6 Output Enables
6 Output Enables
LAB A
LAB B
LAB C
LAB D
Macrocells 1 to 16
Macrocells 17 to 32
Macrocells 33 to 48
Macrocells 49 to 64
I/O Control Block
6 to 16 I/O Pins
6 to16
36
16
6
PIA


Logic Array
Global Clear
Global Clocks
2
from I/O pin
Parallel Logic Expanders (from other macrocells)
Fast Input Select
Programmable Register
Register Bypass
to I/O Control Block
Product-Term Select Matrix
PRN
D/T
Q
ENA
CLRN
Clock/ Enable Select
VCC
Clear Select
to PIA
Shared Logic Expanders
36 Signals from PIA
16 Expander Product Terms

آزمایش پنجم

زبان توصیف سخت افزار، زبانی برای شرح و تعریف سیستمهای دیجیتال به صورت متنی است . به عبارت دیگر HDL را می توان توصیف رابطهٔ بین سیگنالهای ورودی یک مدار و سیگنالهای خروجی آن تعریف کرد . پس این زبان می تواند برای نمایش مدارهای منطقی ، عبارتهای بولی و یا مدارهای پیچیده دیجیتال بکار رود .

HDL های متنوعی توسط شرکتهای مختلف ارائه شده است مانند زبان VHDL و زبان Verilog . در این متن سعی شده به صورت خلاصه در مورد زبان وریلاگ که زبان ساده تری است توضیح داده شود .برای سادگی یادگیری با ارائهٔ مثالهای ساده و متعدد سعی شده یک آشنایی کلی برای دانشجو ایجاد شود .

توصیف مدارهای دیجیتال به سه روش زیر انجام می شود :

- توصیف مدار در سطح دروازه های منطقی (گیت)
- توصیف مدار در سطح جریان داده (سیگنال)
- توصیف رفتاری

توصیف مدار بر اساس جریان داده

در توصیف جریان داده برای تعیین نسبت بین ورودی و خروجی از عملگرهای منطقی و حسابی استفاده میشود . در جدول زیر عملگرهای قابل استفاده در زبان وریلاگ آورده شده است.

نوع عملگر	سمبل عملگر	عملیات	عملوند ها
عملگر های حسابی و منطقی	*	ضرب	۲
	/	تقسیم	۲
	+	جمع	۲
	-	تفریق	۲
	%	باقی مانده	۲
	!	معکوس منطقی	۱
	&&	و منطقی	۲
	||	یا منطقی	۲
عملگر های الحاق و تکرار	{ }	الحاق	نا محدود
	{{ }}	تکرار	نا محدود
عملگر های شیفت	>>	شیفت راست	۲
	<<	شیفت چپ	۲

نوع عملگر	سمبل عملگر	عملیات	عملوند ها
عملگر های رابطه ای[1] ، تساوی ، بیتی	<	کوچکتر از	۲
	>	بزرگتر از	۲
	<=	کوچکتر و مساوی	۲
	>=	بزرگتر و مساوی	۲
	==	تساوی	۲
	!=	عدم تساوی	۲
	===	تساوی نوع حروف	۲
	!==	عدم تساوی نوع حروف	۲
	~	معکوس بیتی	۱
	&	و بیتی	۲
	|	یا بیتی	۲
	^	یا اختصاصی بیتی	۲
	^~ یا ~^	معکوس یا اختصاصی بیتی	۲

جدول ۱ – عملگرهای زبان وریلاگ

برای درک بهتر این بخش به مثالهای ساده زیر توجه کنید.

مثال ۵-۱ : جمع کنندهٔ کامل یک بیتی با روش جریان داده

```
module        fulladder_1bit  ( SUM , Cout , A , B , Cin);
        output          SUM;
        output          Cout;
```

```
    input       A , B;
    input       Cin;
    assign      { Cout , SUM } = A + B + Cin;
endmodule
```

واحد ساختاری وریلاگ "ماژول" نامیده می شود. برنامه بالا در اصل یک ماژول است که با کلید واژهٔ module شروع و انتهای آن با کلید واژهٔ endmodule خاتمه می یابد . خطوط بین این دو کلیدواژه که بدنهٔ ماژول را تشکیل میدهند در اصل توصیف طرح بر اساس جریان داده است .

بعد از کلمه module یک نام مناسب برای مدار نوشته میشود سپس در دنبالهٔ آن و در داخل پرانتز نام ورودیها و خروجیها ذکر میگردد . یک جمع کننده کامل یک بیتی دارای دو ورودی اصلی A و B و یک ورودی بیت نقلی است که در اینجا بنام Cin نامگذاری شده است. همچنین این مدار دارای یک خروجی حاصل جمع بنام SUM است که نتیجه حاصل جمع دو متغیر ورودی A و B در آن قرار می گیرد. اگر حاصل جمع دارای بیت نقلی باشد مقدار آن در متغیر Cout نمایش داده می شود. متغیرهای ورودی و خروجی در زبان وریلاگ بترتیب با کلمات input و output تعریف می گردند.

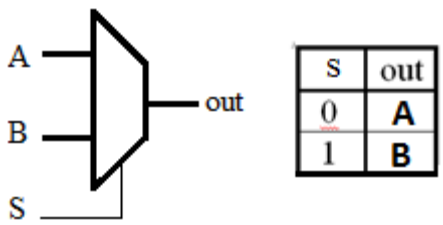
توجه : قوانین نام گذاری متغیرها و توابع در زبان وریلاگ مانند زبان C است.

در روش جریان داده بعد از تعاریف اولیهٔ ماژول ، با یک یا چند گزارهٔ حسابی و یا منطقی ارتباط بین ورودیها و خروجیها توصیف میگردد. البته قبل از هر عبارت باید کلمهٔ assign آورده شود. در مثال بالا در جمله آخر ارتباط بین ورودی و خروجی با عملگرهای انتساب (=) و جمع (+) بیان شده است. چون حاصل جمع دو متغیر یک بیتی ممکن است دو بیتی باشد ابتدا باید متغیرهای یک بیتی خروجی را با عمگر الحاق {} به یک متغیر برداری دو بیتی تبدیل و سپس حاصل جمع را به آن منتسب کرد. در این حالت بیت اول حاصل جمع به متغیر SUM و بیت دوم و پر ارزشتر به خروجی Cout تخصیص دهی می شود.

مثال ۵-۲: مالتی پلکسر ۲ به ۱

جدول درستی و شکل نمادین یک مالتی پلکسر ۲ به ۱ درزیر نشان داده شده است . در کنار آن توصیف مدار بر اساس مدل جریان داده آمده است .

```
module      mux2to1(output  out , input  a , b, s);
assign        out = (s==1)  ?  b : a ;
endmodule
```



S	out
0	A
1	B

شکل ۸

توضیح عبارت شرطی آخر برنامه:

condition ? true expression : false expression

شرط ؟ عبارت درست : عبارت نادرست

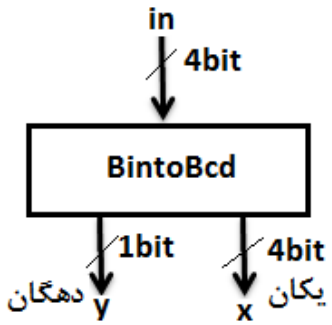
مثال ۵-۳ : مقایسه کنندهٔ چهار بیتی با روش جریان داده

```
module comp1bit (output  Gt ,  Eq , Lt , input  [3:0]a , b )
        assign          Gt = a > b ;
        assign          Lt = a < b ;
        assign          Eq = a == b ;
endmodule
```

input	output		
	Gt	Eq	Lt
a = b	0	1	0
a < b	0	0	1
a > b	1	0	0

در این روش از گزاره های شرطی برای توصیف مدار استفاده شده است در این صورت نیازی به دانستن جدول درستی نیست. همانطور که در خط اول برنامه مشاهده می کنید برای تعریف یک ورودی و یا خروجی چند بیتی از عملگر [n:0] که قبل از نام متغیر می آید میتوان استفاده کرد. n تعداد بیتهای متغیر است . در این حالت چون نام دو متغیر a و b با علامت کاما جدا شده اندازه متغیر به هر دو اطلاق می شود.

آزمایش ۵-۱ : مطابق شکل ۹ یک مبدل باینری به دهدهی را با روش جریان داده طراحی کنید. صحت عملکرد طراحی را روی بورد بررسی کنید.



شکل ۹

توجه : می توانید از عملگرهای % (باقیمانده) و / (خارج قسمت) استفاده کنید. باقیمانده یکان و خارج قسمت مقدار دهگان خواهد بود.

نکته : نمایش اعداد ثابت در وریلاگ

ساختار کلی نمایش اعداد در این زبان بصورت زیر است:

<Size> ' <Radix><Value> →<اندازه> ' <مبنا><مقدار>

در جدول زیر چند نمونه از روش نمایش اعداد ثابت در وریلاگ آمده است .

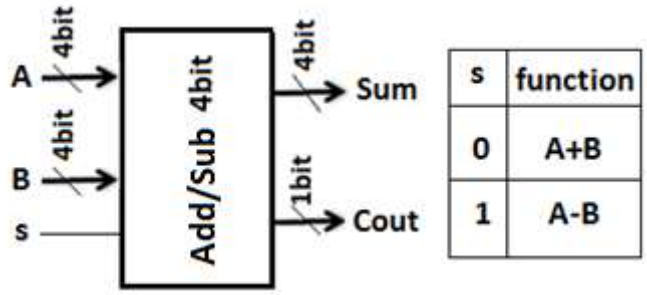
مقدار در برنامه	مقدار ذخیره شده در حافظه	توضیح
15	00000000000000000000000000001111	بدون ذکر اندازه :۳۲ بیت دسیمال
4'b1111	1111	با ذکر اندازه: ۴ بیت در مبنای باینری
8'h0f	00001111	با ذکر اندازه: هشت بیت در مبنای هگز
8'd15	00001111	با ذکر اندازه: هشت بیت در مبنای دسیمال
8'hzz		با ذکر اندازه :هشت بیت امپدانس بالا
8'hxx		با ذکر اندازه: هشت بیت مقدار نامعتبر

بعنوان مثال برای نوشتن عدد 6 چهار بیتی ابتدا عدد 4 برای مشخص کردن تعداد بیتها سپس علامت ' و بعد از آن مبنای نمایش عدد (b باینری ، d دسیمال ، h هگزادسیمال و ...) و در نهایت خود عدد آورده میشود. مثلاً عد ۶ را در وریلاگ بصورت 4'd6 (دهدهی) و یا 4'b0110 (باینری) و یا 4'h6 (مبنای شانزده) نمایش می دهیم. اگر عدد 6 را بصورت معمول بنویسیم برنامه آنرا ۳۲ بیتی فرض می کند.

گزارش کار

پروژه:

مدار جمع کننده/تفریق کنندهٔ چهاربیتی شکل زیر را با روش جریان داده طراحی کنید و درستی عملکرد آنرا در محیط شبیه سازی با مقادیر ورودی جدول زیر بررسی کنید. مقادیر آزمایش را با مقادیر آزمایش ۲-۲ در بخش اول دستور کار مقایسه کنید .



s	function
0	A+B
1	A-B

شکل ۱۰

s	A	B	Sum	Cout
0	0011	0010		
1	0011	0010		
0	0010	0011		
1	0010	0011		
0	1001	1000		
1	1001	1000		

آزمایش ششم

طراحی دیکودر و کار با نمایشگر سون سگمنت

مثال ۶-۱ : دیکودر ۲ به ۴

برنامهٔ زیر یک دیکودر ۲ به ۴ همانند آزمایش ۲-۱ را پیاده سازی می کند. این مدار دارای یک ورودی دو بیتی و یک خط خروجی چهار بیتی است . به چگونگی پیاده سازی این مدار توجه کنید.

```
module      decoder2to4 ( input  [1:0] sel , output  [3:0]q  );
assign      q =  ( sel == 0 )  ?  4'b0001 :
                 ( sel == 1 )  ?  4'b0010 :
                 ( sel == 2 )  ?  4'b0100 :
                                  4'b1000;
endmodule
```

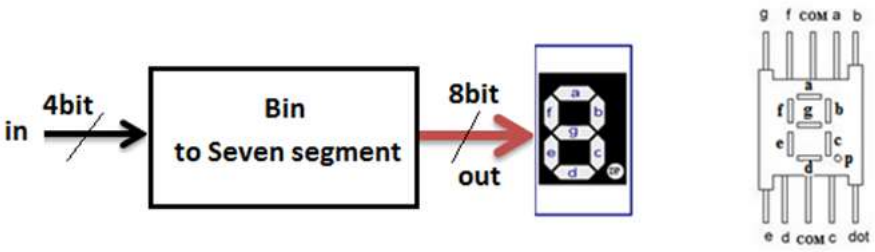
در این مثال همانند مثال ۲ از آزمایش ۵ از گزارهٔ شرطی زیر استفاده شده است:

عبارت نادرست : عبارت درست ؟ عبارت شرطی = خروجی

در این ماژول شرط اول sel==0 است که اگر درست باشد مقدار باینری 0001 در خروجی قرار می گیرد و اگر درست نباشد شرط دوم مورد ارزیابی قرار می گیرد یعنی بجای عبارت نادرست،گزاره های مربوط به شرط دوم قرار داده شده است و بجای گزارهٔ نادرست شرط دوم نیز گزاره های مر بوط به شرط سوم جایگذاری شده است الی آخر.

آزمایش ۶-۱ : طراحی مبدل باینری به کد سون سگمنت

با توجه به مثال قبل یک مبدل باینری چهاربیتی به کد سون سگمنت طراحی کنید تا بتواند اعداد مبنای هگز از 0 تا F را روی سون سگمنت نمایش دهد . از جدول صفحهٔ بعد برای تعیین وضعیت خروجیها استفاده نمایید . برای اعداد A تا F جدول را تکمیل کنید . طرح خود را روی بورد پیاده سازی کرده و با تغییر ورودی عملکرد آنرا بررسی کنید.



شکل ۱۱

در این مدار فرض بر این است که کم ارزشترین بیت خروجی مبدل به پایه سگمنت a و پرارزشترین بیت به پایهٔ سگمنت p (نقطهٔ اعشار) متصل است.

عدد	نمایش	کد هگز	وضعیت سگمنتها							
			p	g	f	e	d	c	b	a
0		3f	0	0	1	1	1	1	1	1
1		06	0	0	0	0	0	1	1	0
2		5b	0	1	0	1	1	0	1	1
3		4f	0	1	0	0	1	1	1	1
4		66	0	1	1	0	0	1	1	0
5		6d	0	1	1	0	1	1	0	1
6		7d	0	1	1	1	1	1	0	1
7		07	0	0	0	0	0	1	1	1
8		7f	0	1	1	1	1	1	1	1
9		67	0	1	1	0	0	1	1	1
A										
b										
C										
d										
E										
F										

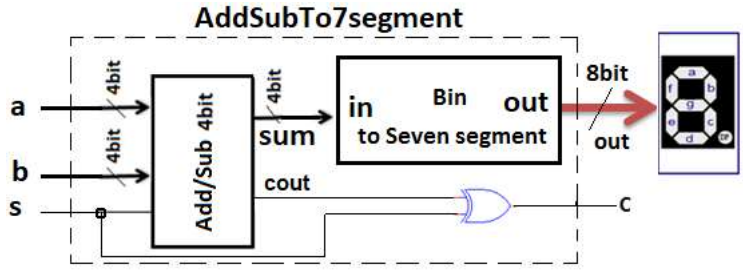
جدول ۱

روش سطح گیت (روش فراخوانی)

در بسیاری از مواقع برای طراحی یک پروژه لازم است آنرا به اجزاء کوچکتر تقسیم کرد و بعد از طراحی اجزاء کوچکتر آنها را به هم متصل نمود. در وریلاگ برای طراحی یک مدار بزرگ و پیچیده ابتدا آنرا به ماژولهای کوچکتر تقسیم می کنیم سپس ماژولهای کوچکتر را طراحی کرده و در نهایت این ماژولها را در ماژول اصلی فراخوانی می کنیم. اصول فراخوانی در وریلاگ مشابه با فراخوانی توابع در زبان C است. برای درک بهتر این مفهوم به مثال زیر توجه کنید.

آزمایش ۶-۲ : نمایش خروجی جمع کننده تفریق کننده با نمایشگر سون سگمنت

در این مثال می خواهیم مدار شکل ۱۲ را پیاده سازی کنیم. شکل زیر (داخل مستطیل خط چین) از اتصال دو ماژول جمع کننده/تفریق کننده و مبدل باینری به سون سگمنت ساخته شده است . توصیف و ساخت این مدار از طریق روش سطح گیت یا فراخوانی ممکن است . در ادامه چگونگی طراحی با این روش را توضیح می دهیم.



شکل ۱۲

در پروژهٔ آزمایش پنجم یک مدار جمع کننده / تفریق کننده طراحی کردیم که دارای قالب زیر بود :

```
module  AddSub( input   [3:0]a , b , input s , output  cout , output  [3:0]sum);
        assign      {cout , sum } =  (s==0)  ?  a+b : a-b ;
endmodule
```

همچنین در آزمایش ۶-۱ نیز یک مبدل باینری به سون سگمنت طراحی شد که دارای قالب زیر است:

```
module   BinTo7Segment (input  [3:0]in , output   [7:0]out );
        assign          out = ( in == 0 )   8'b 0011 1111 :
                              ( in == 1 )   8'b 0000 0110 :
                                ....
        endmodule
```

(البته ممکن است اسامی ماژولها و یا متغیرها با آنچه شما طراحی کرده اید متفاوت باشد.)

حال می خواهیم با اتصال این دو ماژول در یک ماژول سوم ، مقدار متغیر sum برای متغیر ورودی in در ماژول مبدل ارسال شود . ماژول سوم و توضیحات مربوط به آن در زیر آمده است:

```
module  AddSubTo7seg ( input  [3:0]a , b , input s , output c , output [7:0]out);
        wire  [3:0]w;
        wire  m;
        AddSub(a , b , s , m , w );
```

```
    BinTo7segment ( w ,  out );
    xor      ( c , m , s );
endmodule
```

ورودیهای ماژول نهایی عبارت است از متغیرهای a و b که چهاربیتی هستند و متغیر s که یک بیتی است و این ماژول دارای یک خروجی هشت بیتی بنام out و یک خروجی یک بیتی بنام c است. متغیر sum از ماژول اول در داخل ماژول اصلی به متغیر ورودی in از ماژول دوم متصل شده است این اتصال داخلی در وریلاگ wire (سیم) نامیده میشود. متغیر سیم باید در بدنهٔ ماژول معرفی شود که در برنامهٔ بالا با عبارت wire [3:0]w معرفی شده است. اسم متغیر سیم دلخواه است و البته نباید با متغیرهای ورودی و خروجی یکسان باشد. این متغیر از نوع سیم در فراخوانی ماژولها بجای متغیر sum در ماژول جمع کننده و بجای متغیر in در تابع مبدل قرار گرفته است.

برای متغیر cout که به ورودی گیت xor متصل شده است نیز یک سیم باید تعریف کرد. لازم بذکر است که متغیرهای ورودی و خروجی می توانند بعنوان سیم هم در نظر گرفته شوند. مثلاً ورودی دوم گیت xor که به ورودی s متصل است دیگر نیازی به تعریف سیم ندارد.

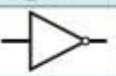
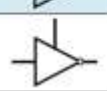
گیتهای پایه مانند گیت xor و غیره در کتابخانهٔ وریلاگ تعریف شده اند و نیازی به تعریف توسط برنامه نویس نیست. نکته ای که در فراخوانی گیتها باید در نظر گرفت این است که در فراخوانی متغیرهای مربوط به گیت ابتدا متغیر خروجی و سپس متغیرهای ورودی قرا می گیرند. (ترتیب ورودیها مهم نیست)

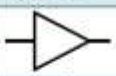
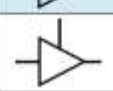
اکنون این مثال را اجرا کرده و عملکرد آنرا روی بورد نشان دهید. خروجی c را روی LED نمایش دهید.

نکتهٔ ۱ : برای انجام این مثال ابتدا فایل ماژول جمع کننده/تفرق کننده را در پوشه پروژهٔ فعلی کپی کنید سپس در یک فایل جدید د رهمین پروژه ماژول نهایی AddSubTo7seg را بنویسید. فایل جدید باید TopLevel شود.

نکتهٔ ۲ : می توان بجای اینکه هر ماژول در یک فایل جداگانه باشد هر سه ماژول را در یک فایل قرار داد. در این صورت اسم فایل باید همنام ماژول اصلی باشد که در این پروژه ماژول AddSubTo7seg است.

نکتهٔ ۳ : گیتهای تعریف شده در کتابخانهٔ وریلاگ در جدول زیر آمده است:

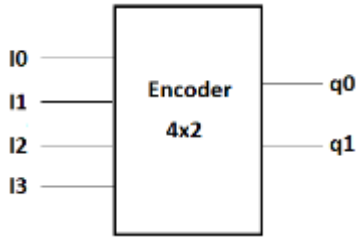
نام گیت	نماد	کلید واژه
گیت AND با N ورودی		and
گیت NAND با N ورودی		nand
گیت OR با N ورودی		or
گیت NOT با N خروجی		not
وارونگر سه حالته فعال پایین		notif0
وارونگر سه حالته فعال بالا		notif1

نام گیت	نماد	کلید واژه
گیت NOR با N ورودی		nor
گیت XOR با N ورودی		xor
گیت XNOR با N ورودی		xnor
گیت بافر با N خروجی		buf
بافر سه حالته فعال پایین		bufif0
بافر سه حالته فعال بالا		bufif1

جدول ۲

گزارش کار

پروژه : یک انکودر ۲ به ۴ را همانند شکل زیر طراحی کنید. عملکرد مدار را در محیط شبیه سازی بررسی کنید.



شکل ۱۳

I3	I2	I1	I0	q1	q0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

آزمایش هفتم

مدل سازی رفتاری

توصیف مدارهای دیجیتال در سطح آلگوریتمی و عملیاتی را توصیف رفتاری می گوییم .این مدل توصیف سریعی از رفتار مدار ارائه می دهد بدون اینکه مجبور باشد سخت افزار را معرفی کند. این مدل اغلب در توصیف مدارهای ترتیبی کاربرد دارد ولی قابل استفاده در مدارهای ترکیبی نیز هست.

مدل رفتاری مبتنی بر کنترل رخداد است یعنی تا یک رویداد خاص اتفاق نیفتد خروجیها تغییری نمی کنند توصیف مدار در این مدل با کلید واژهٔ always شروع می گردد و سپس به دنبال آن یک عبارت کنترل رخداد قرار می گیرد. عبارت کنترل رخداد تعیین می کند که گزاره ها چه زمانی باید اجرا گردند. بعد از عبارت کنترل رخداد تعداد دلخواهی از گزاره ها با انتساب رَوَندی قرار می گیرد . خروجی در بلوک always از نوع ثبات می باشد که با کلیدواژهٔ reg تعریف می گردد . این نوع متغیر برخلاف متغیر از نوع wir که مداوم می تواند تخصیص دهی شود ، مقدارش را تا تخصیص دهی جدید حفظ می کند . یک ماژول می تواند از چند بلوک always تشکیل شود و همهٔ این بلوکها بصورت همزمان اجرا می شوند.

مدل رفتاری برای مدارهای ترتیبی

در مدار های ترتیبی، رخداد باید از نوع حساس به لبه باشد. که این امر با کلید واژه های posedge (لبهٔ بالا رونده) و negedge (لبهٔ پایین رونده) بیان می شود.

مثال ۷-۱ : فلیپ فلاپ نوع D حساس به لبهٔ بالارونده با Reset سنکرون

```
module          d_ff  (input  d , clk , reset , output  reg   q ) ;
        always   @ ( posedge   clk )
                if ( reset == 1)
                        q = 1'b0;
                else
                        q = d ;
endmodule
```

در برنامهٔ بالا خروجی q با کلیدواژهٔ output بعنوان خروجی و با کلیدواژهٔ reg بعنوان یک ثبات تعریف گردیده است چرا که خروجی در مدل رفتاری باید از نوع ثبات باشد. بطور کلی بلوک always در زمان صفر شروع بکار می کند. در وریلاگ می توان یک بلوک را طوری کنترل نمود تا در زمان خاصی مثلا در لبهٔ یک پالس ساعت فعال شود. مانند همین مثال که دستورات بعد از کلیدواژهٔ always وقتی اجرا می شوند که تغییری در سیگنال ورودی clk صورت گیرد در غیر این صورت اجرا تا تغییر بعدی متوقف می شود .

در مثال۱ واکنش خروجی به ورودی ریست ، وابسته به کلاک است چون تا کلاک تغییر نکند دستور if(reset) اجرا نمی شود به همین دلیل ریست این فلیپ فلاپ از نوع همگام و یا سنکرون نامیده می شود.

مثال ۷-۲ : فلیپ فلاپ نوع D حساس به لبهٔ بالارونده با Reset آسنکرون

```
module          d_ff (input  d , clk , reset , output  reg   q ) ;
        always @ ( posedge  clk  , negedge  reset )
                if ( reset == 0)
                        q = 1'b0;
                else
                        q = d ;
endmodule
```

در این مثال گزاره های داخل بلوک هم با تغییر مقدار reset و هم با تغییر clk اجرا می شود. بنابراین Reset وابسته به کلاک نیست . دقت داشته باشید که حساسیت تعریف شده برای Reset با گزارهٔ شرطی مربوط به این ورودی در داخل بلوک always باید هماهنگی داشته باشد . مثلاً در لیست حساسیت متغیر reset به سطح صفر (negedge) تنظیم شده است پس در گزارهٔ شرطی if نیز باید با صفر مقایسه شود.

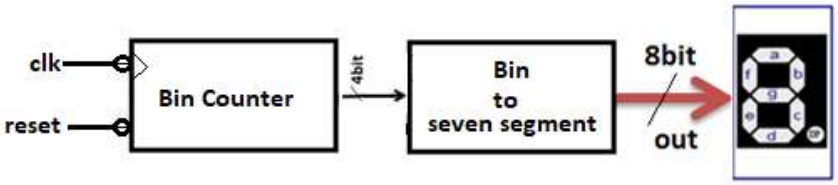
آزمایش ۷-۱ : شمارندهٔ چهار بیتی بالا شمار حساس به لبهٔ پایین رونده با Reset آسنکرون

عملکرد این ماژول را توسط شبیه سازی مشاهده کنید.

```
module          bincounter (input  clk , reset , output   reg   [3:0] q ) ;
        always    @ (negedge   clk , negedge   reset)
                if (! reset )
                        q = 4'h0;
                else
                        q = q +4'h1 ;
endmodule
```

آزمایش ۷-۲ :یک شمارندهٔ چهاربیتی دهدهی (شمارش از ’0‘ تا ’9‘) طراحی کنید. این مدار دارای پایه ورودی Reset آسنکرون حساس به سطح صفر است . خروجی را روی بورد پیاده سازی نمایید. (با اضافه نمودن یک گزارهٔ شرطی مناسب در مثال آزمایش ۷-۱ می توانید این برنامه را بنویسید)

آزمایش ۷-۳ : با استفاده از مبدل باینری به کد سون سگمنت و برنامهٔ آزمایش ۷-۱ ، مدار شکل زیر را توسط روش سطح گیت پیاده سازی کرده و جواب را روی بورد مشاهده کنید .



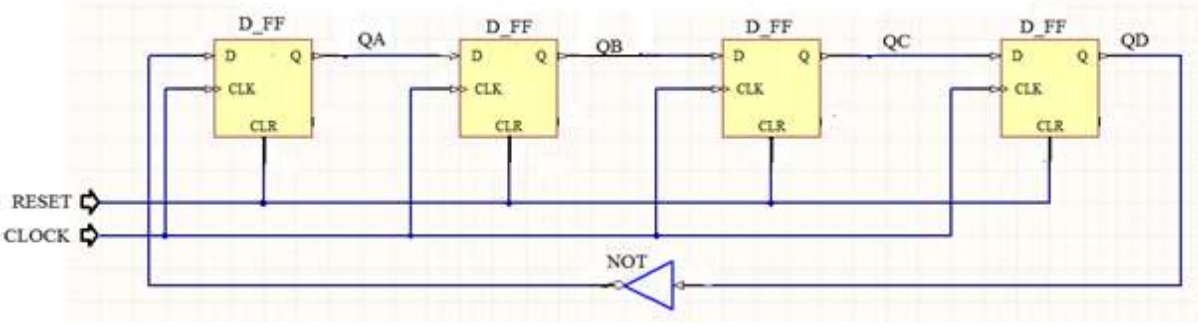
شکل ۱۴

گزارش کار

پروژه ۳ : با استفاده از روش سطح گیت ، برنامهٔ مدار شکل زیر را بنویسید و آنرا شبیه سازی نمایید. برای فلیپ فلاپ D از برنامهٔ مثال ۷-۲ استفاده کنید . ماژول را با قالب زیر طراحی کنید که متغیر خروجی چهار بیتی q بجای

خروجیهای Qa ، Qb ، Qc و Qd در شکل زیر باشد .

```
module        Johanson ( input  clk , reset , output  [3:0]q );
```



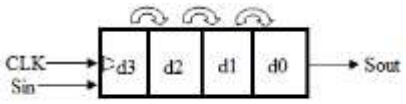
شکل ۱۵

آزمایش هشتم

طراحی شیفت رجیستر و حافظه

مثال۸-۱ : شیفت دهندهٔ چهار بیتی با ورودی و خروجی سریال با قابلیت شیفت به راست

```
module      shiftreg ( input   clk , sin , output   sout );
      reg     [3:0]temp ;
      always        @(posedge    clk )
            begin
                  temp  <=  temp >> 1 ;
                  temp[3] <= sin ;
            end
      assign        sout = temp[0]
endmodule
```



شکل۱۶

در این برنامه در گزاره های داخل بلوک always بر خلاف برنامه های قبل از عملگر انتساب <= استفاده شده است. در وریلاگ به عملگر = عملگر انتساب بلوکی و به عملگر <= عملگر انتساب غیر بلوکی گفته می شود. برای درک تفاوت این دو به مثالهای زیر توجه کنید.

```
begin
      a = b + c ;       \\ if  b=3  and  c=2   and  a(t-1) =1  =>  a=5
      d = a ;  \\  => d=5
end
```

در این بلوک از عملگر انتساب بلوکی (=) استفاده شده است .همانطور که می بینید گزاره ها به ترتیب از بالا به پایین اجرا می شوند.

```
begin
      a <= b + c ;      \\ if  b=3  and  c=2  and  a(t-1) = 1 =>  a=5
      d <= a ;          \\  => d=1
end
```

در این بلوک از عملگر انتساب غیربلوکی (<=) استفاده شده است . در این حالت ابتدا مقادیر سمت راست همهٔ گزاره ها همزمان محاسبه می گردند و سپس عمل انتساب و مقدار دهی به متغیرهای سمت چپ صورت می گیرد. در این صورت ترتیب نوشتن گزاره ها از بالا به پایین تاثیری در عملکرد مدار نخواهد داشت.

نکتهٔ دیگری که در برنامهٔ شیفت رجیستر جلب توجه می کند استفاده همزمان از دو روش توصیف رفتاری و جریان داده است .اگر توجه کرده باشید خروجی sout از نوع ثبات تعریف نشده است پس عمل انتساب مقدار به آن باید بصورت پیوسته و با کلیدواژهٔ assign صورت گیرد. باید توجه داشته باشید که بلوک always و بلوک assign بصورت همزمان و موازی اجرا می شوند.

سیگنال temp یک سیگنال داخلی و از دید کاربر مخفی است پس در تعریف ماژول و اِعلان ورودیها و خروجیها آورده نمی شود.

تعریف آرایه :

مثال ۸-۲ : طراحی حافظه با ظرفیت شانزده بایت

برای تعریف یک متغیر از نوع آرایه می توان مانند مثال زیر عمل کرد.در این مثال یک حافظه هشت بیتی با ظرفیت ۱۶ بایت تعریف کرده ایم .

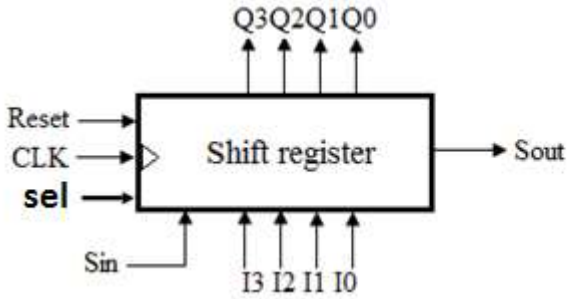
```
module      SRam (output  [7:0]Sramout , input   [7:0]Sramin , input   [3:0]Sramaddress ,
                  Input   Clock , Wren) ;
    reg     [7:0]Ram[0:15] ;
    always    @(posedge    clk )
        begin
            if( Wren )
            Ram[Sramaddress] = Sramin ;
        end
assign   Sramout  =  Ram[Sramaddress];
endmodule
```

در زبان وریلاگ برای ایجاد گزاره های شرطی می توان از دستور case استفاده نمود. الگوی چگونگی استفاده از آن در زیر آمده است.

```
case  ( sel[1:0] )
        0 : عبارت ۱
        1 : عبارت ۲
        2 : عبارت ۳
        3 : عبارت ۴
        default : عبارت ۵
endcase
```

در این مثال ، متغیر دو بیتی sel ممکن است دارای چهار مقدار ممکن 0 ، 1 ، 2 و 3 باشدکه به ازاء هر حالت یکی از عبارات چهارگانه اجرا خواهد شد. در غیر این صورت عبارت پیش فرض اجرا خواهد شد. البته عبارت default میتواند استفاده نشود.

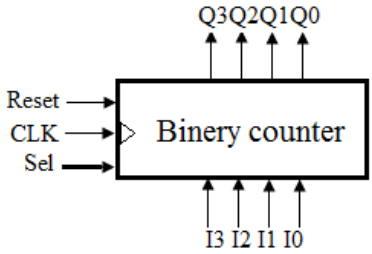
آزمایش۸-۱ : یک شیفت رجیستر مانند شکل ۱۷ با قابلیت بارگذاری موازی سنکرون ، شیفت به راست و شیفت به چپ همچنین ورودی Reset آسنکرون طراحی نمایید. عملکرد مدار را توسط شبیه سازی مشاهده کنید.(برای مشاهده بهتر در محیط شبیه سازی ، سیگنالها را بصورت باینری تنظیم کنید)



Sel	Function
00	Load
01	**Shift right**
10	**Shift left**
11	No change

شکل۱۷

آزمایش ۸-۲ : مانند شکل زیر یک شمارندهٔ چهار بیتی باینری با خط انتخاب عملکرد طبق جدول زیر، دارای ورودی و خروجی چهار بیتی و خط reset آسنکرون طراحی نمایید. مدار را روی بورد پیاده سازی نمایید.



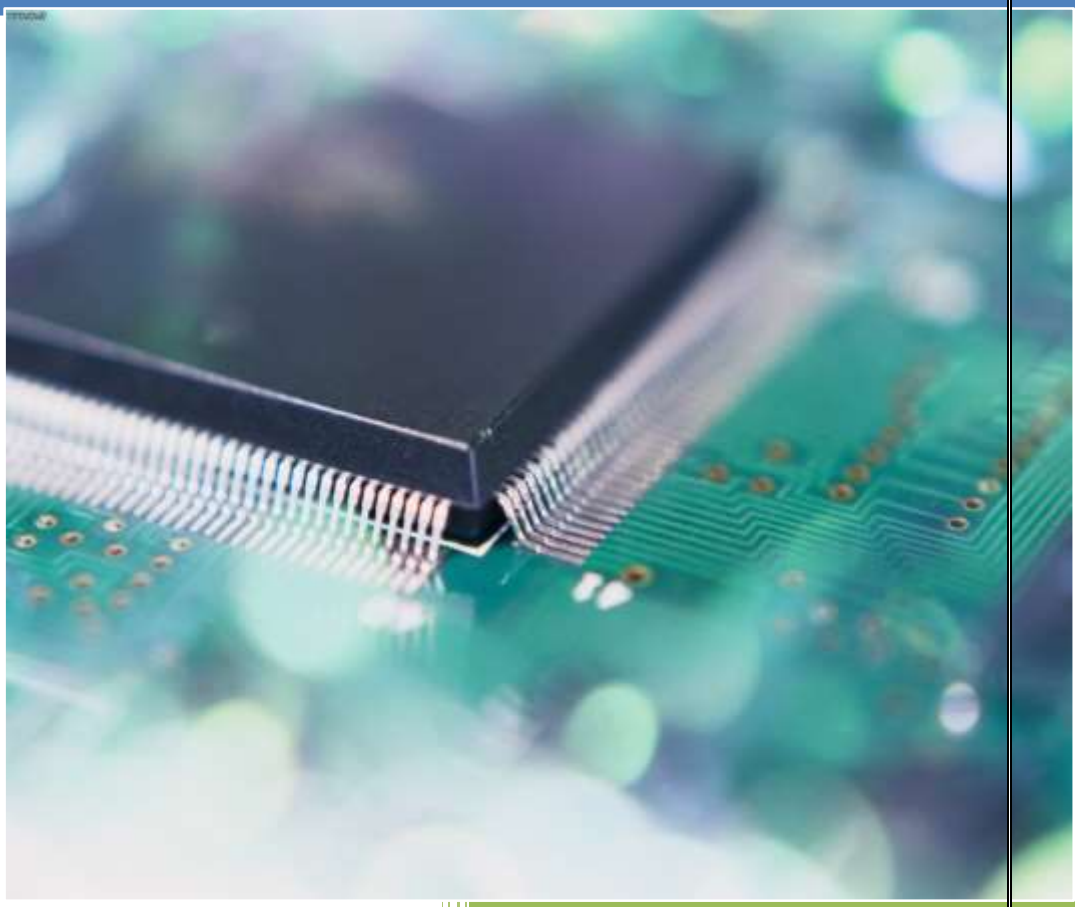
Sel	Function
00	Load
01	Count up
10	Count down
11	No change

شکل۱۸

بسمه تعالی

دانشگاه کاشان

راهنمای نرم افزار QUARTUS



دانشکدۀ برق و کامپیوتر
آزمایشگاه مدار منطقی
محمّدرضا فتّاح

راهنمای نرم افزار

آشنایی با نرم افزار کوارتوس (Quartus)

مقدمه

شرکت ALTRA که در زمینهٔ ساخت آی سی های برنامه پذیر مانند FPGA فعالیت می کند نرم افزار کوارتوس را برای طراحی مدارهای منطقی و پیاده سازی آن روی آی سی های برنامه پذیر در اختیار کاربران قرار داده است . در این نرم افزار با روشهای مختلف از جمله برنامه نویسی متنی و یا روش شماتیک می توان عمل طراحی و شبیه سازی و در نهایت سنتز مدار را انجام داد.

در این متن روش طراحی توسط زبان وریلاگ وشبیه سازی و سپس پیاده سازی آن را با استفاده از این نرم افزار توضیح خواهیم داد.

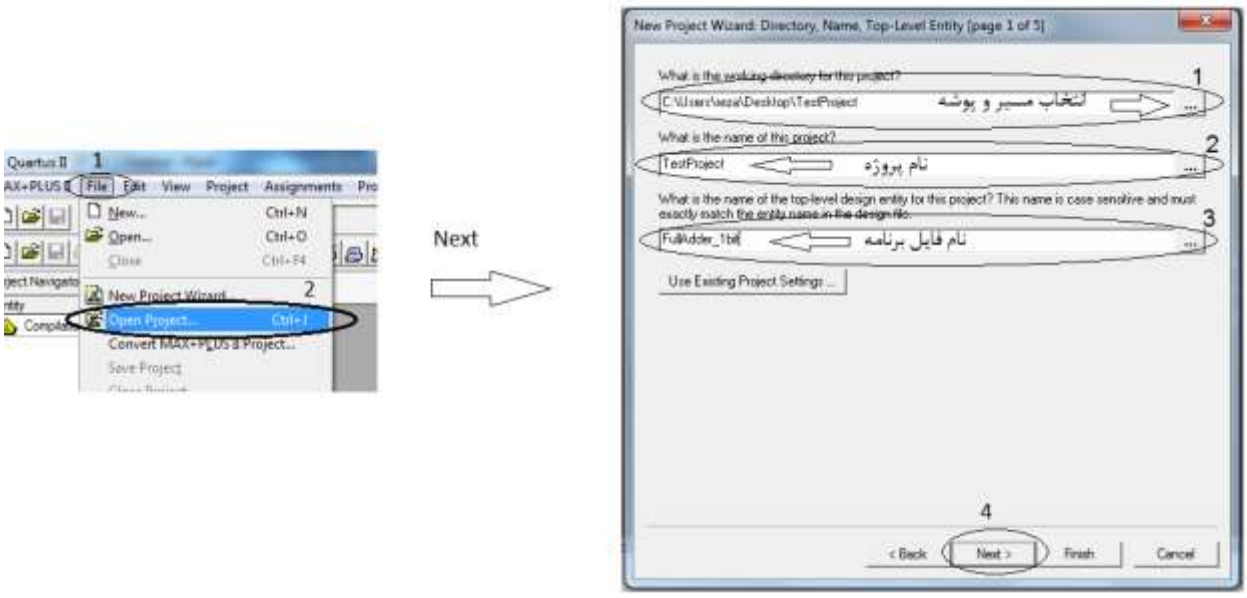
۱- ایجاد پروژه

در این نرم افزار هر پروژه شامل چندین فایل از جمله فایل اصلی برنامه و فایلهای فرعی دیگر است برای جلوگیری از پراکندگی فایلها ، حتماً سعی کنید در ابتدا یک پوشه با نام مناسب بسازید . توجه داشته باشید که نام پوشه و یا فایلها حتماً با استفاده از حروف اصلی لاتین و بدون استفاده از کارکترهایی مانند & و غیره باشد . در غیر این صورت هنگام کامپایل برنامه با خطا مواجه خواهید شد . البته ایجاد پوشه را می توانید هنگام ساخت پروژه نیز انجام دهید.

قصد ما در این جا طراحی و پیاده سازی مدار تمام جمع کنندهٔ یک بیتی است بنابراین نام فایل را FullAdder_1bit انتخاب کنید که مربوط به آزمایش اول دستور کار (بخش دوم) است .

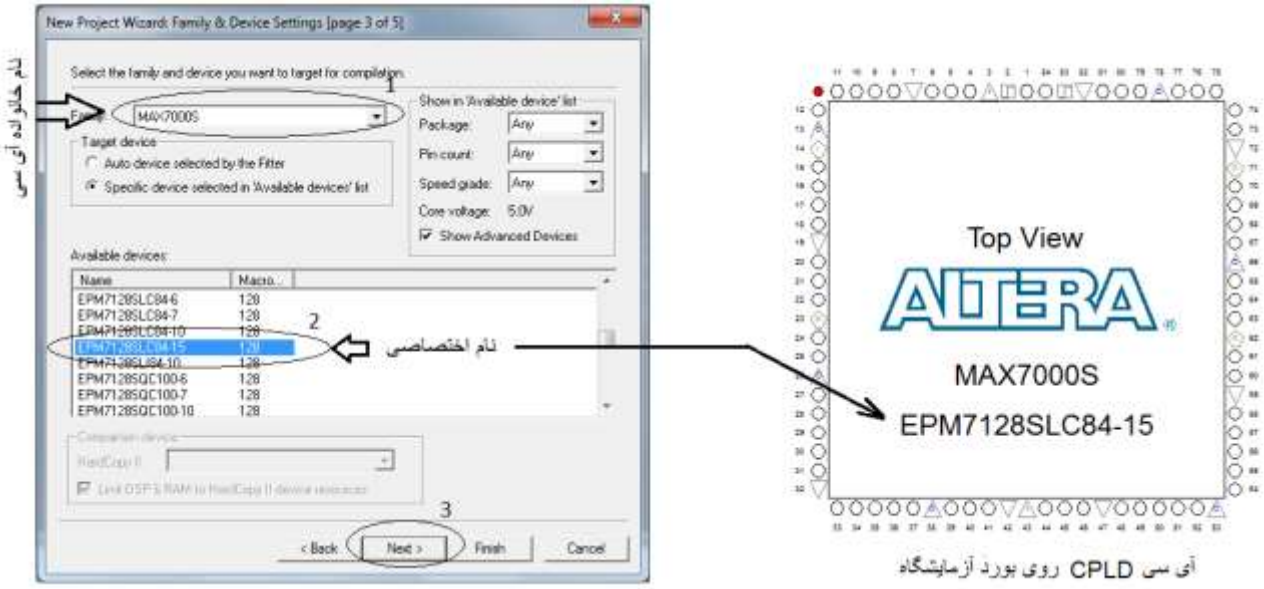
برای ایجاد پروژه مراحل زیر را دنبال کنید:

نرم افزار را اجرا کنید و طبق شکل (سمت چپ)از منوی File گزینه New Project Wizard را انتخاب کنید.

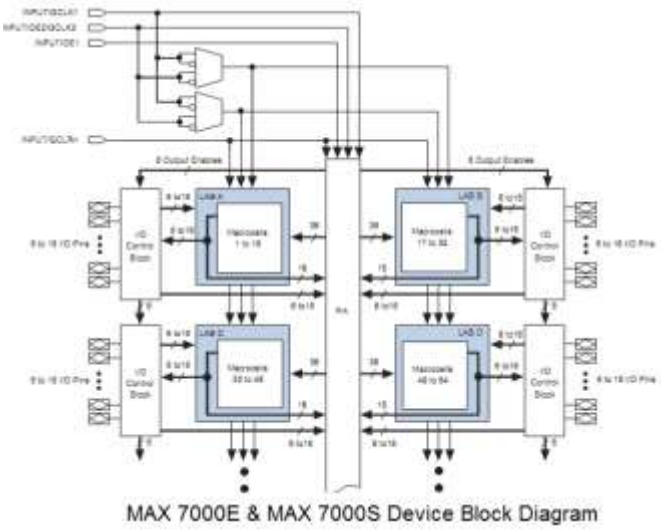


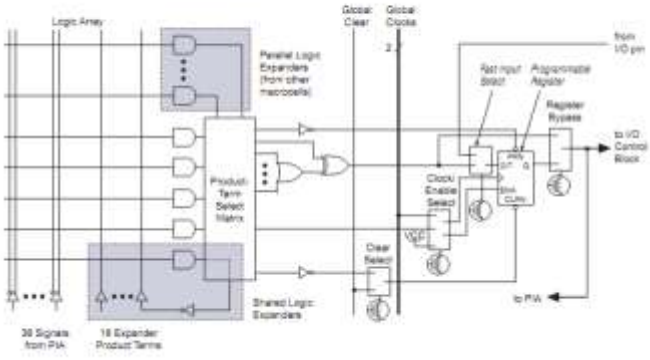
توجه داشته باشید در این مرحله فقط پروژهٔ معرفی شده ایجاد می شود ولی فایل مربوطه با نام داده شده در مرحلهٔ ۳ ، بعداً باید با همین نام ساخته شود. نوع و فرمت این فایل (متنی یا شماتیک) در هنگام ساخت مشخص می شود

توجه داشته باشید که این فایل اصطلاحاً TopLevel نامیده می شود. اگر در پروژه ساخته شده فایلی با این نام وجود نداشته باشد در هنگام کامپایل ، پیغام خطای عدم وجود فایل TopLevel داده خواهد شد.
کلید Next را بزنید تا پنجرهٔ زیر (سمت چپ) ظاهر شود.



در این مرحله آی سی برنامه پذیری که قصد داریم طرح نهایی مدار خود را روی آن پیاده سازی کنیم انتخاب میکنیم. بورد آموزشی آزمایشگاه مدارهای منطقی که آزمایشهای خود را با آن انجام میدهیم مبتنی بر آی سی با نام EPM7128SLC84-15 است. این المان یک آی سی برنامه پذیر CPLD ساخت شرکت ALTRA است .
حروف EP در ابتدای نام ، نشانهٔ این است که حافظهٔ برنامه پذیر این آی سی از نوع EPROM است .
حروف M7xxxS نشان دهندهٔ خانوادهٔ این آی سی یعنی MAX7000S و عدد 128 نشانگر ظرفیت مدارات داخلی این المان بر حسب MacroCell است (۱۲۸ ماکروسل). عدد84 تعداد پایه های آی سی را نشان میدهد. تعدادی از این پایه ها مربوط به تغذیه آی سی و تعدادی دیگر مربوط به برنامه ریزی و باقیمانده می توانند بعنوان ورودی و یا خروجی (I/O) استفاده گردند.

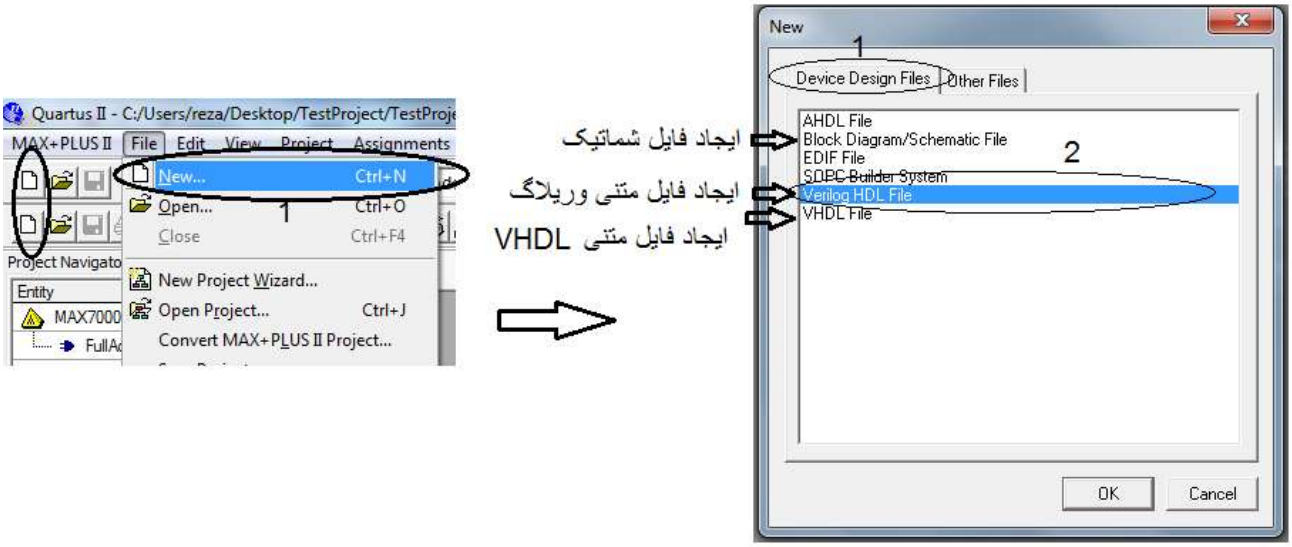

MAX 7000E & MAX 7000S Device Block Diagram


MAX 7000E & MAX 7000S Device Macrocell

شکل بالا دیاگرام داخلی CPLD و مدار داخلی یک ماکروسل را نشان می دهد.

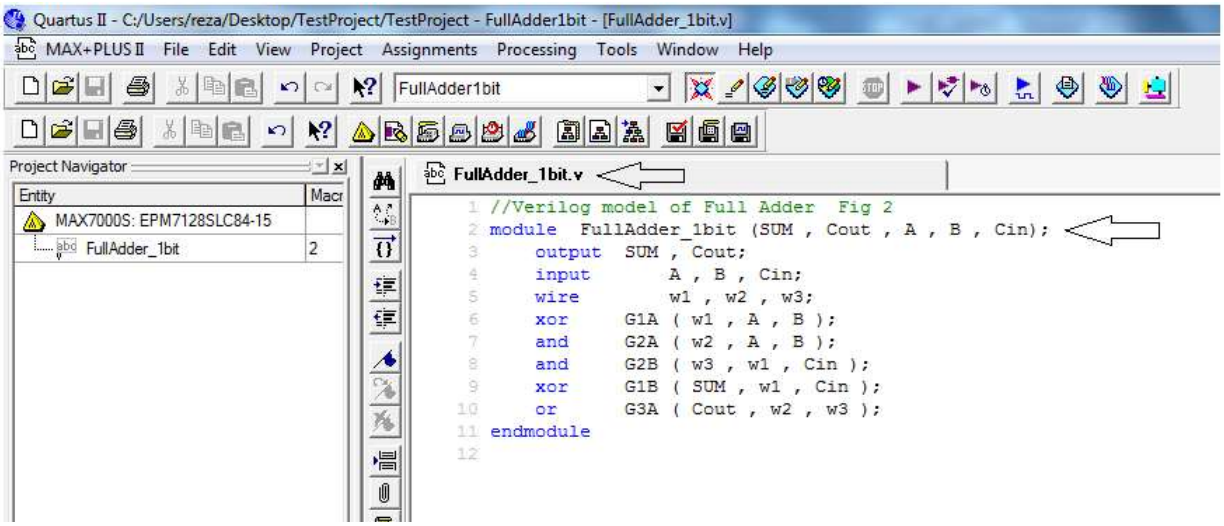
۲- ایجاد *فایل* و کامپایل برنامه

تا این مرحله چگونگی ایجاد یک پروژه به پایان رسید اکنون می خواهیم یک فایل متنی برای نوشتن و طراحی یک جمع کنندهٔ کامل یک بیتی را شروع کنیم. طبق شکل زیر(سمت چپ) از منوی File گزینهٔ New را انتخاب و در پنجرهٔ باز شده گزینه Verilog HDL را انتخاب نمایید.

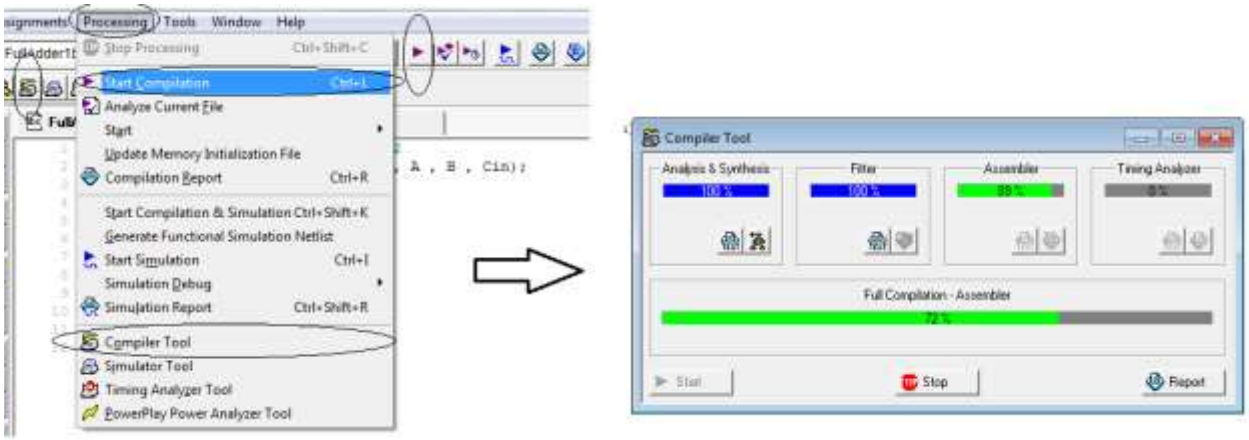


اکنون فایل متنی وریلاگ ایجاد شد با انتخاب گزینهٔ Save از منوی File فایل ایجاد شده را با نامی که قبلاً در مرحلهٔ ایجاد پروژه برای فایل TopLevel انتخاب کرده بودید ذخیره نمایید .توجه داشته باشید اگر مراحل بالا را بدرستی انجام داده باشید نام فایل پیش فرض بصورت خودکار همان نام مورد نظر و در مسیر پوشه ای که ساخته اید خواهد بود.

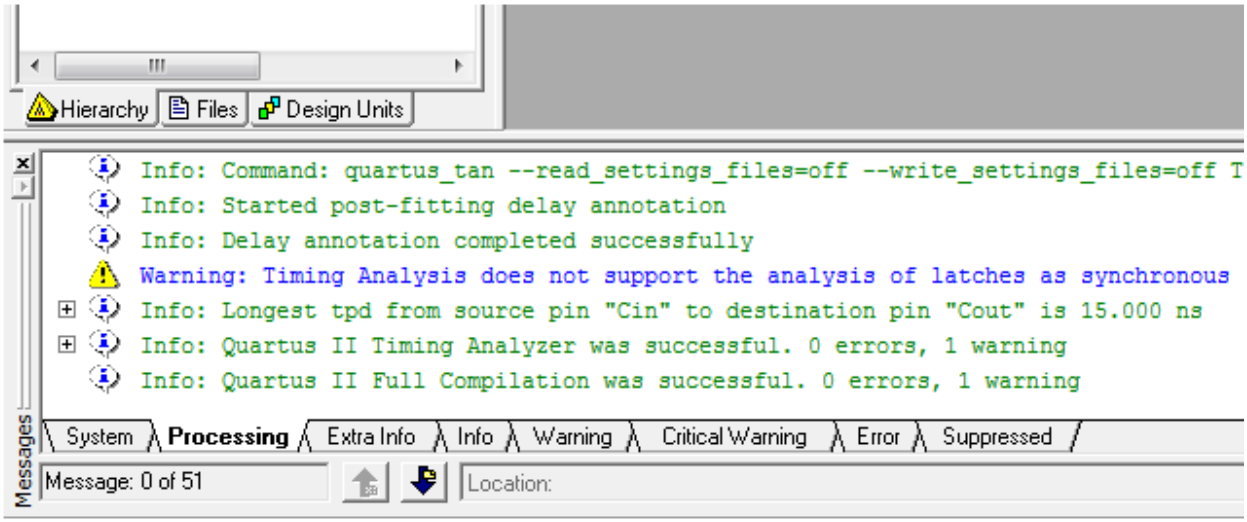
با ساخت فایل متنی اکنون می خواهیم طراحی و شبیه سازی و سنتز مدار جمع کنندهٔ کامل را شروع کنیم.ابتدا برنامهٔ مدار را در فایل ایجاد شده با زبان وریلاگ بنویسید(می توانید آنرا از داخل دستور کار کپی کنید) .



حال می توانیم برنامه را کامپایل کرده از خطاهای احتمالی مطلع شده وآنرا برطرف کنیم .برای این منظور از هر یک از گزینه های مشخص شده در شکل بعد(سمت چپ) می توانید استفاده کنید.



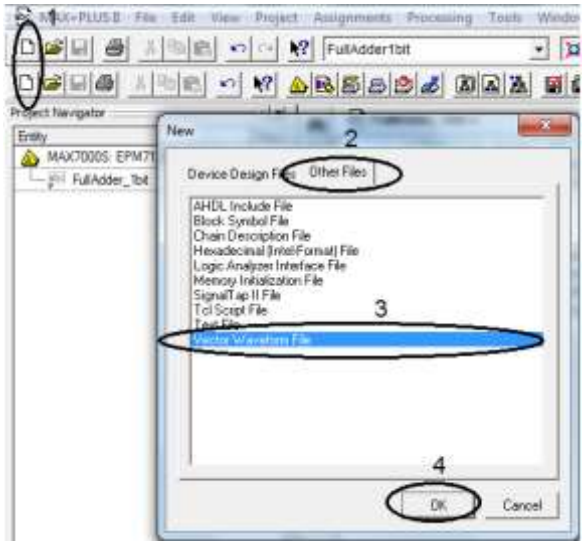
پیامهای خطا و اخطار بصورت شکل زیر در پنجرهٔ پیغامها که در زیر پنجرهٔ فایل قرار دارد ظاهر خواهد شد .



با دوبار کلیک روی پیغام خطای احتمالی محل آن در فایل برنامه مشخص می شود .

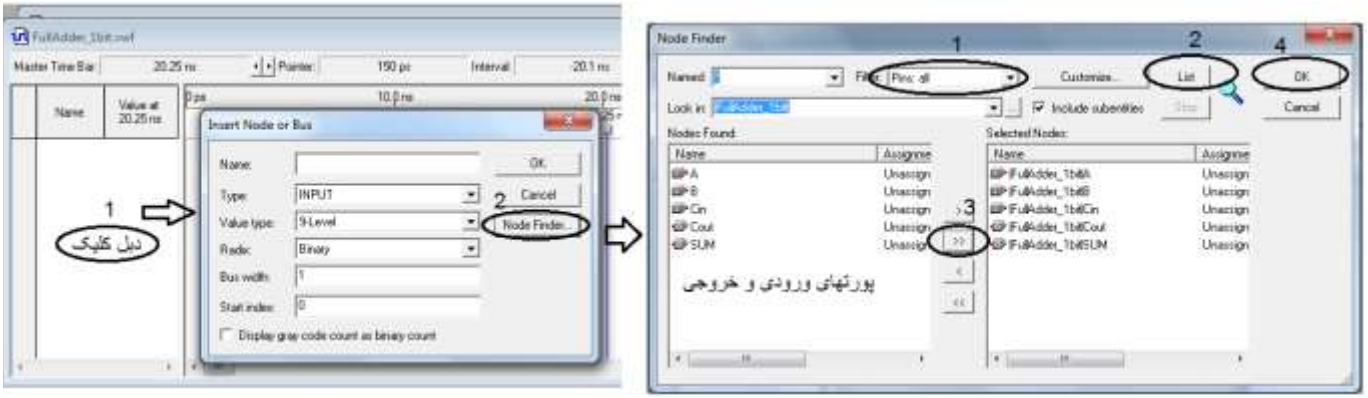
۳- شبیه سازی

در مرحلهٔ بعد با استفاده از ابزار شبیه سازی درستی عملکرد مدار را بررسی میکنیم . برای این منظور از منوی File گزینهٔ New را انتخاب کنید. در پنجره ظاهر شده و از برگهٔ OtherFile گزینهٔ Vector Waveform را گزینش نمایید.

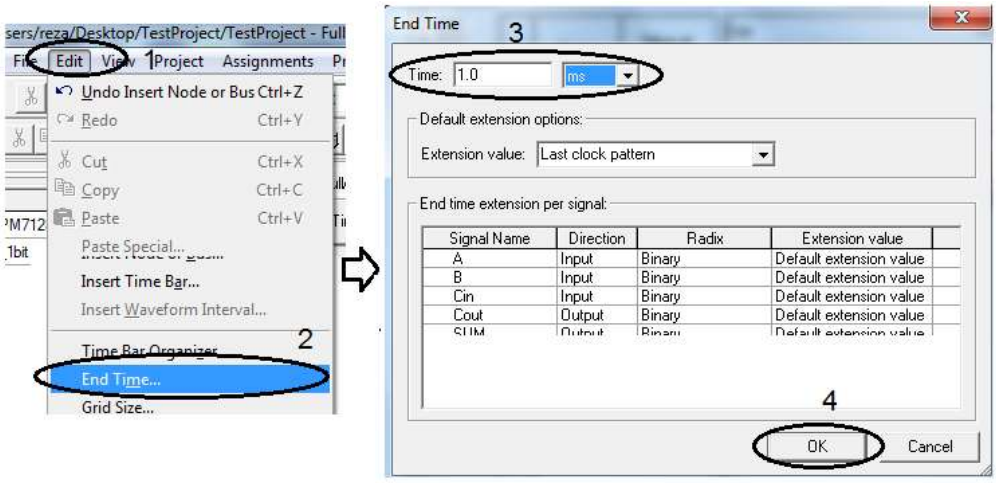


فایل شبیه سازی مانند شکل زیر ایجاد می شود:

اکنون باید سیگنالهای مدار را به فایل شبیه سازی وارد کنیم برای این منظور در پنجرهٔ لیست سیگنال، دوبار کلیک کرده و سپس مانند شکل زیر (سمت راست)مراحل مختلف را به ترتیب شماره های ذکر شده انجام دهید .

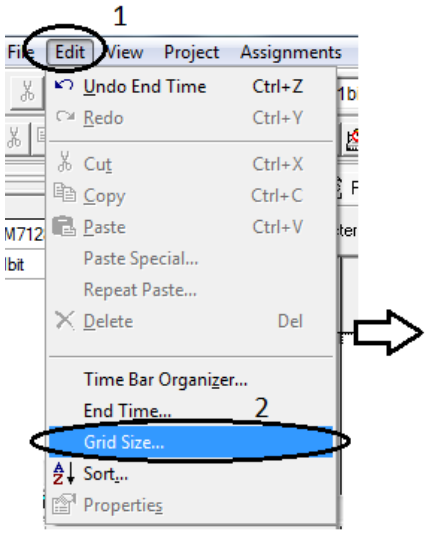


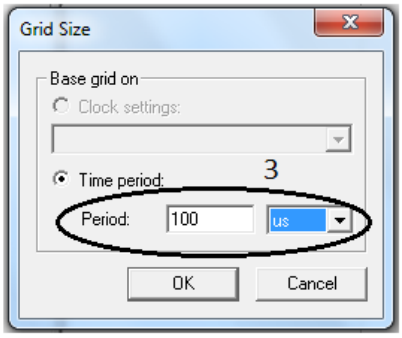
حال سیگنالهای ورودی و خروجی مدار به فایل شبیه سازی اضافه شده است. در مرحلهٔ بعد ابتدا بازهٔ زمانی شبیه سازی و همچنین بازهٔ زمانی برای هر مرحله و حالت را باید مشخص نماییم . برای بازهٔ زمان کل از شکل زیر و برای زمانی هر حالت از شکل صفحهٔ بعد استفاده کنید .



انتخاب بازهٔ زمانی شبیه سازی

در تمام آزمایشها زمان شبیه سازی را یک میلی ثانیه (1ms) انتخاب کنید.





انتخاب بازهٔ زمانی برای هر حالت

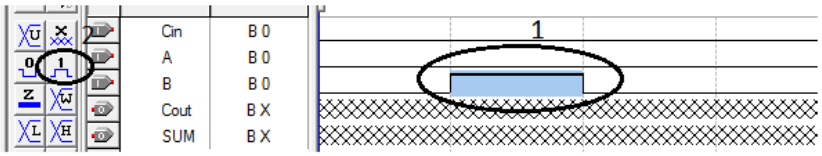
Cin	A	B	Cout	SUM
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

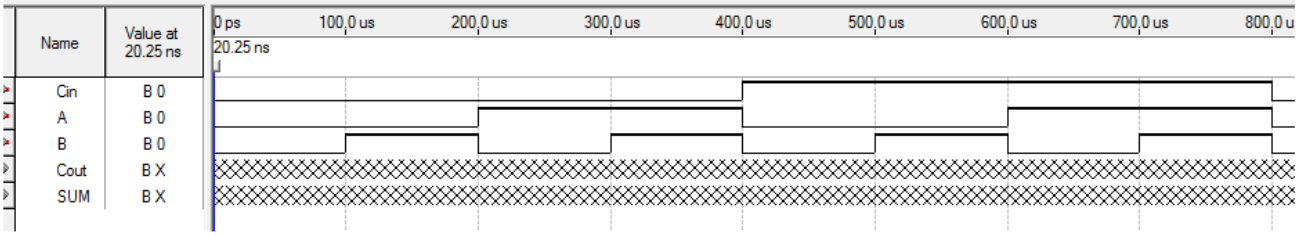
جدول حالات مدار جمع کننده

برای انتخاب بازهٔ زمانی هر حالت به تعداد حالات موجود در جدول درستی مدار نگاه می کنیم و بازهٔ زمانی کل را به تعداد حالات تقسیم می کنیم.تعداد حالات جدول عدد هشت است که برای جلوگیری از حاصل تقسیم با مقدار اعشاری آنرا ۱۰ حالت فرض می کنیم . پس زمان هر بازه برای این مدار ۱۰۰ میکرو ثانیه (100us) خواهد بود.

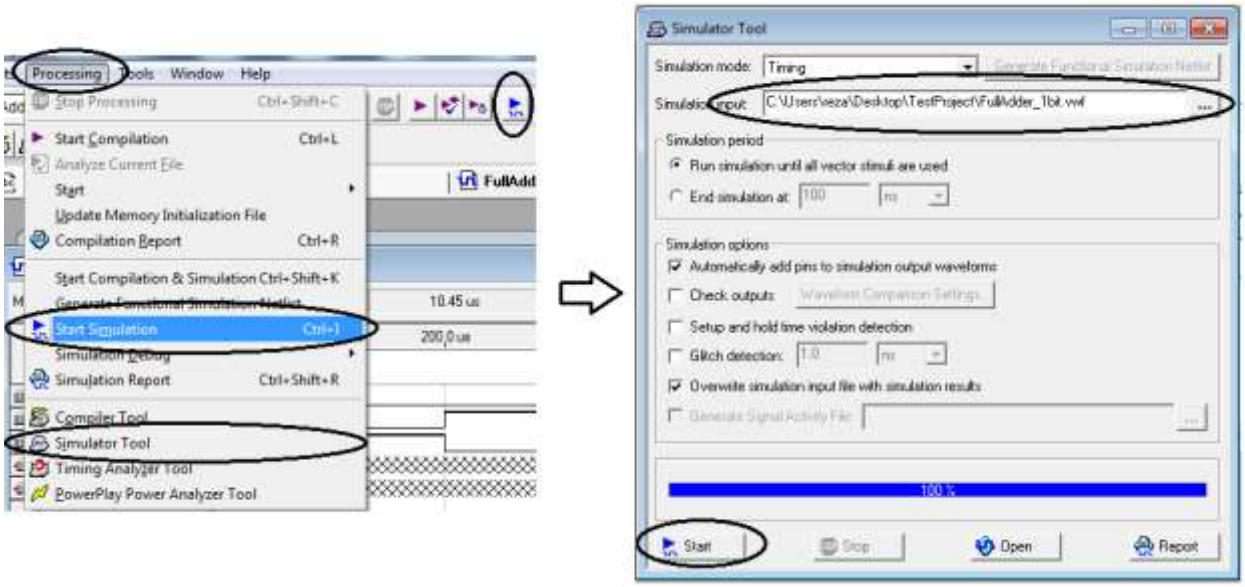
برای قرار گرفتن همهٔ حالات در پنجرهٔ نمایش سیگنال ،از کلیدهای Ctrl+W استفاده نمایید .

اکنون زمان آن رسیده است که مقادیر ورودی را برای همهٔ حالتها طبق جدول درستی مشخص نماییم به طور مثال ورودی B را در نظر بگیرید . این سیگنال در حالت اول دارای مقدار صفر و در حالت بعدی دارای مقدار یک است.همچنین مقدار پیش فرض سیگنال در شروع مقدار صفر است برای اِعمال مقدار یک به حالت دوم، نشانگر ماوس را به ابتدای قسمت دوم سیگنال B برده کلید چپ ماوس را فشار دهید و همزمان ماوس را به سمت راست و تا انتهای حالت دوم بکشید(مانند شکل زیر). این بازهٔ زمانی بصورت رنگی در می آید اکنون با انتخاب کلید  از نوار ابزار سمت چپ صفحه ، مقدار سیگنال به مقدار یک تغییر میکند. برای بقیهٔ حالات و سیگنالهای ورودی این عمل را تکرار کنید .

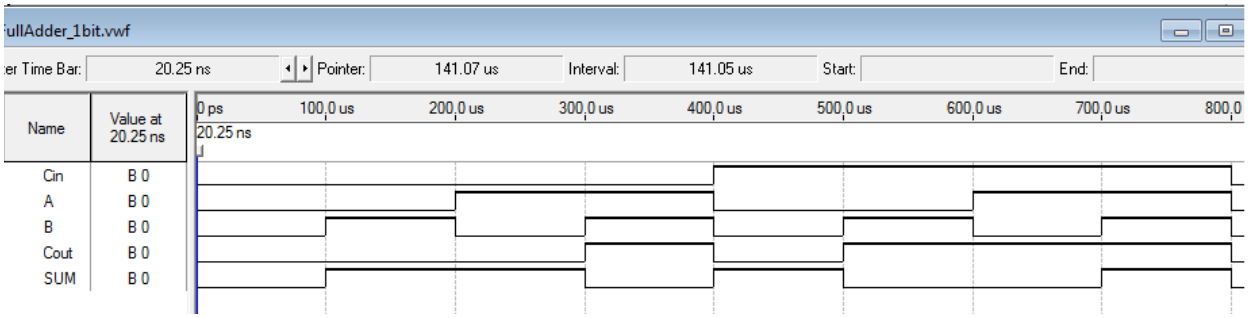




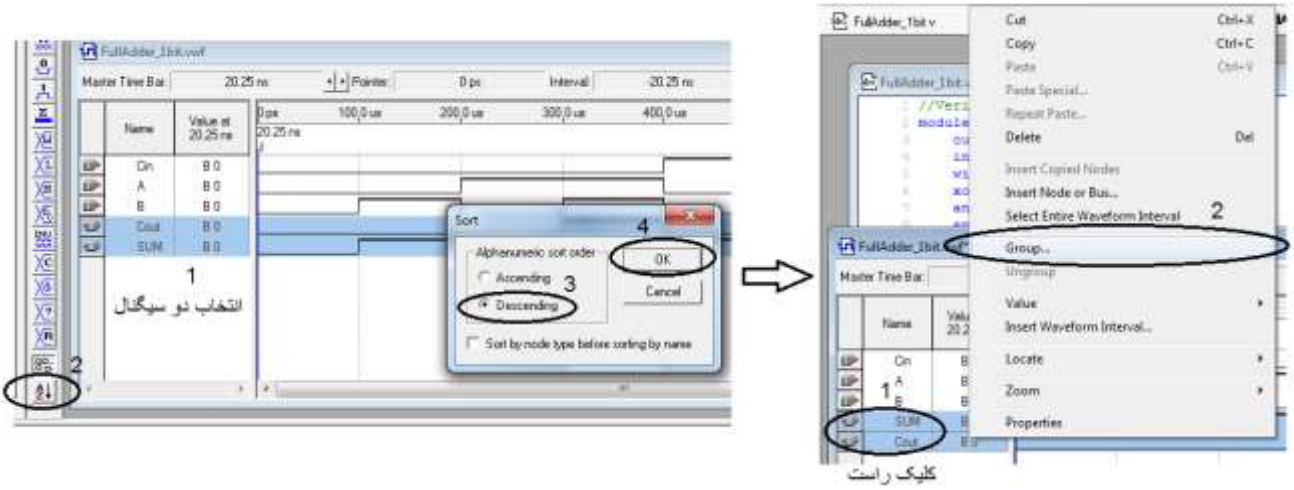
بعد از اینکه مقادیر ورودی را مشخص کردید فایل را ذخیره نمایید . نام پیش فرض پیشنهادی که همان نام فایل اصلی ولی با پسوند متفاوت است را تایید نمایید . برای آغاز شبیه سازی مراحل نشان داده شده در شکل بعد را اجرا کنید .



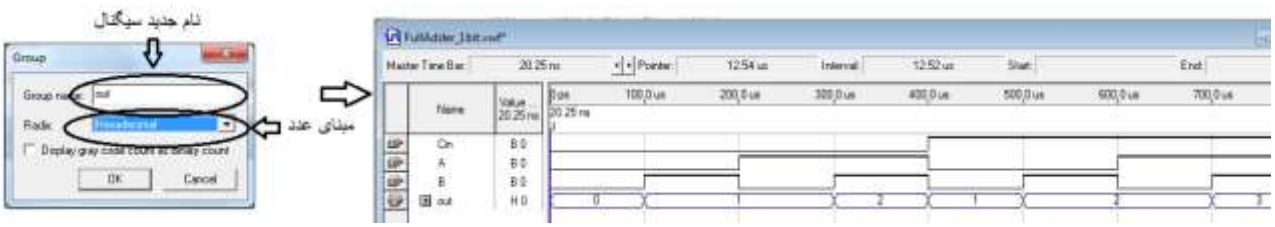
نتیجه بصورت زیر خواهد بود:



اگر بخواهیم خروجی را به صورت عددی مشاهده کنیم مراحل را طبق شکل زیر و به ترتیب انجام دهید:

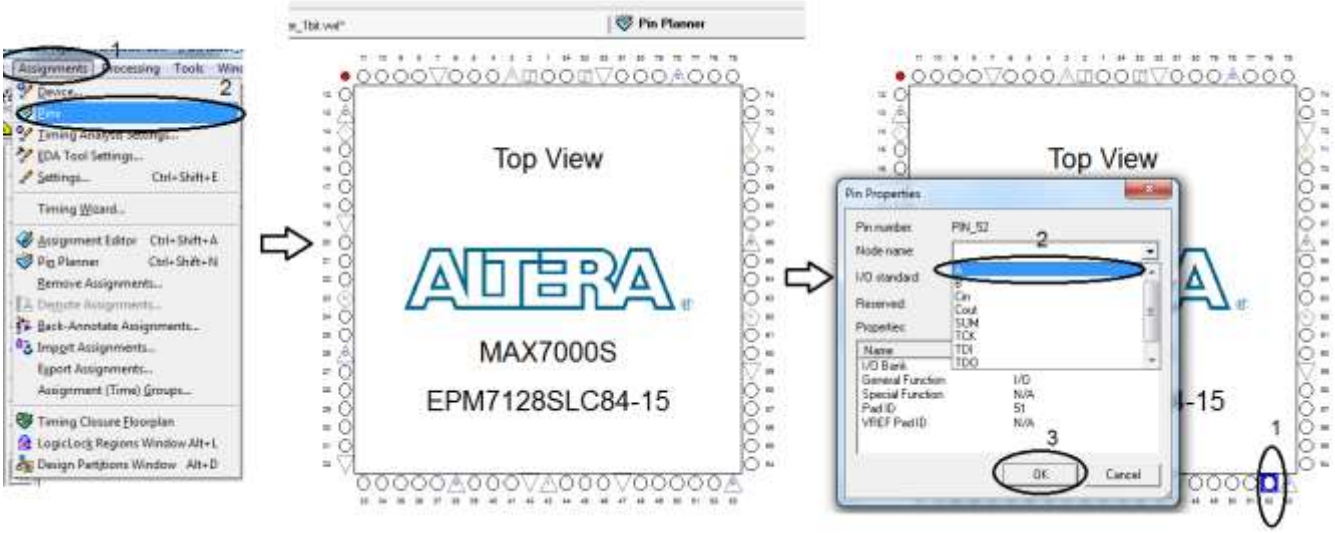


و نتیجه :



۴- پیاده سازی مدار

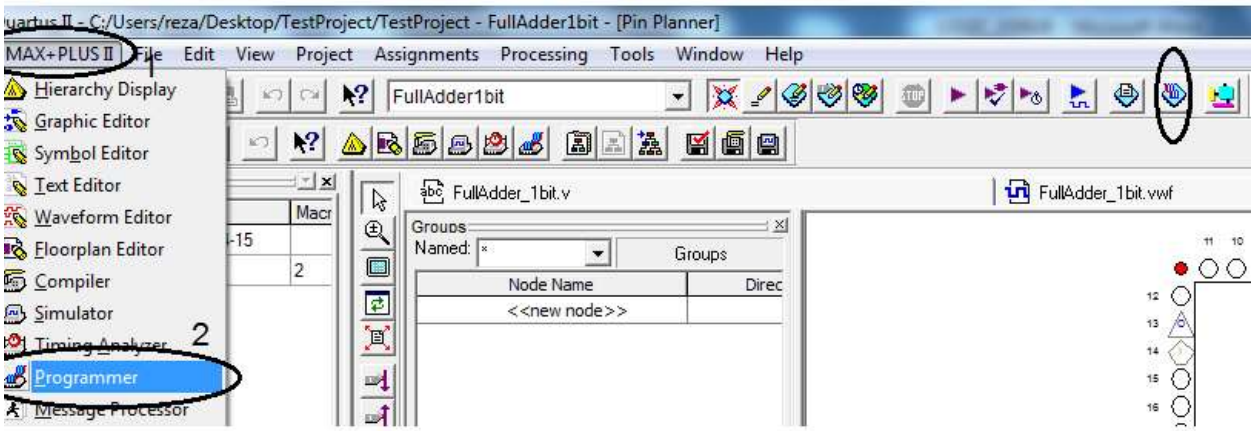
با توجه به اطمینان از درستی عملکرد مدار اکنون می خواهیم طرح خود را روی آی سی برنامه پذیر پیاده سازی و عملکرد مدار را بصورت واقعی مشاهده کنیم. برای این منظور ابتدا باید عمل اختصاص دهی پایه های آی سی به ورودی و خروجیهای مدار را انجام دهیم .با توجه به شکل زیر و به ترتیب شماره ها مراحل ذکر شده را انجام دهید .



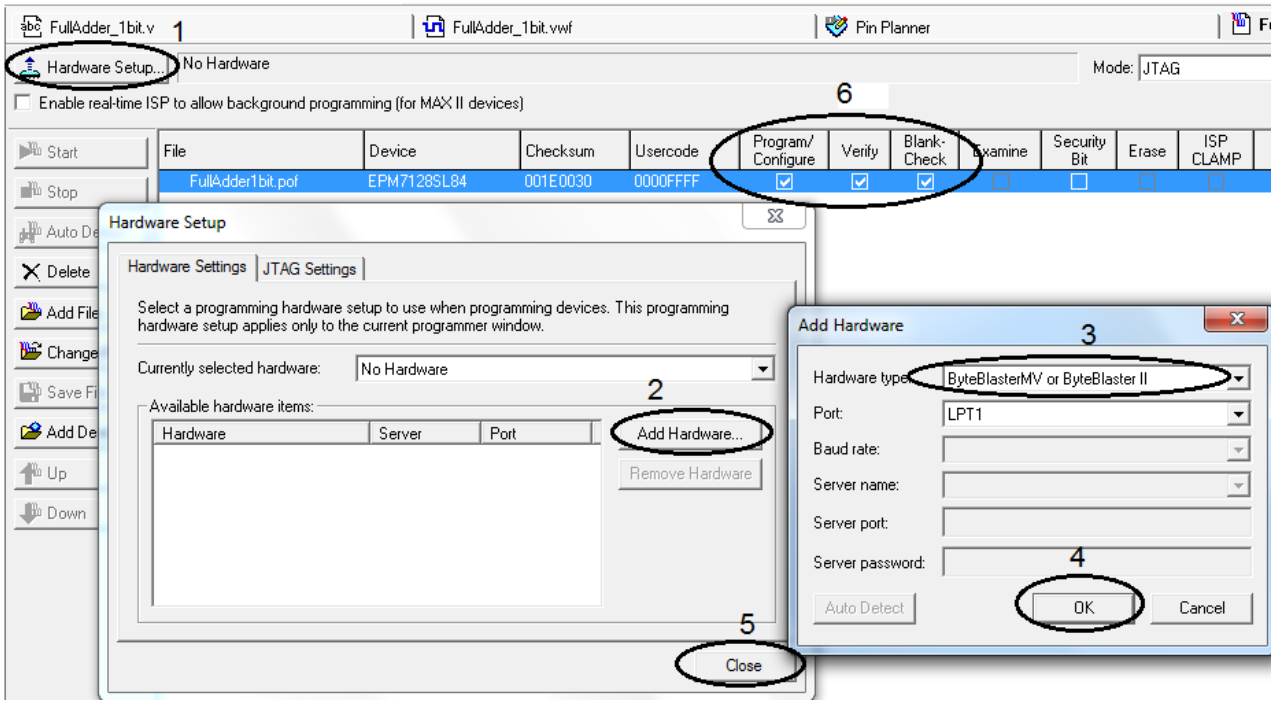
از منوی Assignments گزینهٔ Pins را انتخاب کنید . پنجره ای حاوی آی سی مورد نظر باز می شود.توجه داشته باشید پایه هایی که بصورت دایره هستند می توانند بعنوان پایه های ورودی و یا خروجی تعریف گردند.برای هماهنگی با بورد آموزشی آزمایشگاه پایه های لبهٔ پایین آی سی را برای ورودیها و پایه های لبهٔ کنار سمت راست را برای خروجیها در نظر بگیرید.

برای اختصاص دهی پایه ابتدا روی پایهٔ مورد نظر دو بار کلیک کنید. با باز شدن پنجرهٔ Pin Properties و با کلیک روی منوی باز شدنی روی اسم پایهٔ مورد نظر خود کلیک کرده تا عمل اختصاص دهی انجام شود. برای بقیهٔ پایه های مدار این کار را انجام دهید.

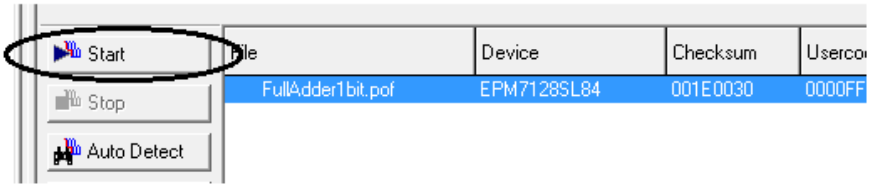
برای کامل شدن عملیات یک بار دیگر عمل کامپایل برنامه را انجام دهید . اکنون می توانید مدار خود را روی آی سی پروگرام نمایید . مانند شکل زیر از منوی Max+Plus گزینهٔ Programmer را انتخاب کنید .



بعد از باز شدن پنجرهٔ جدید ابتدا باید نوع پروگرامر خود را انتخاب کنید . مطابق شکل بعد عمل نمایید.



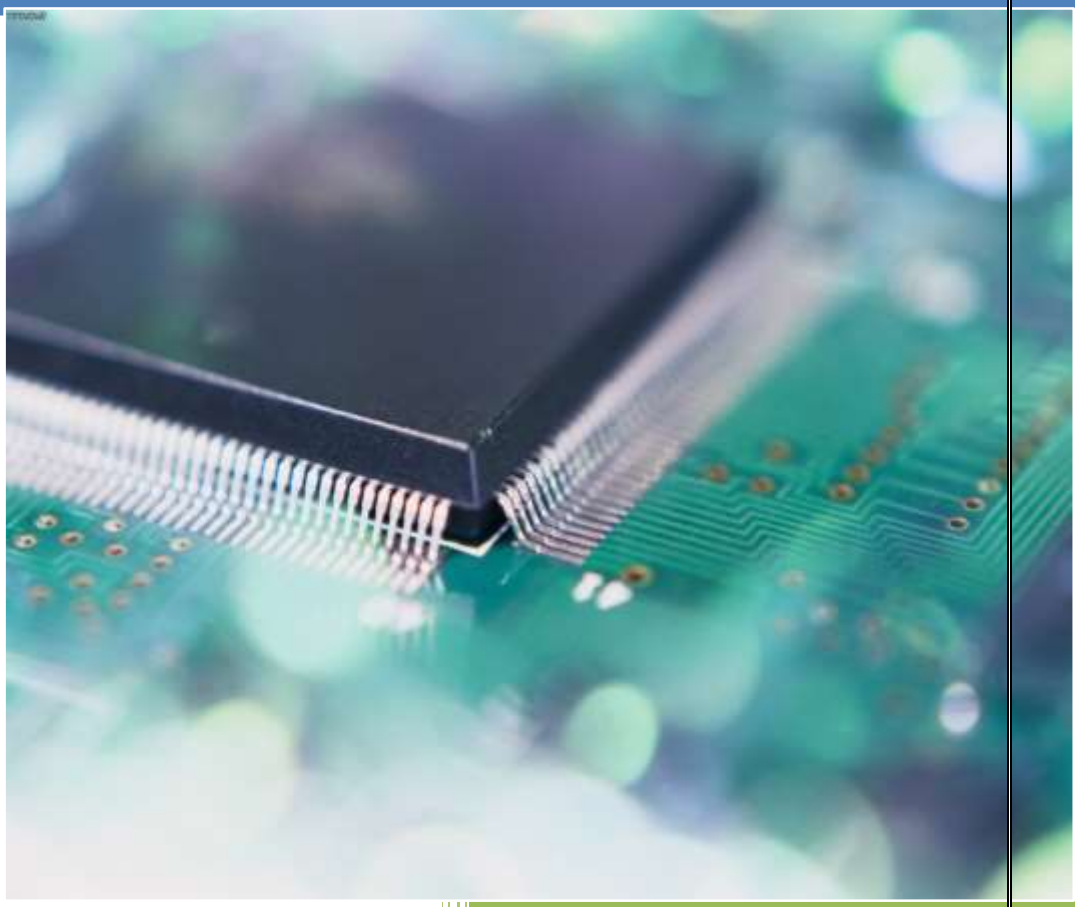
اکنون روی کلید Start کلیک کنید . عملیات برنامه ریزی شروع می شود .



بسمه تعالی

دانشگاه کاشان

راهنمای نصب نرم افزار QUARTUS



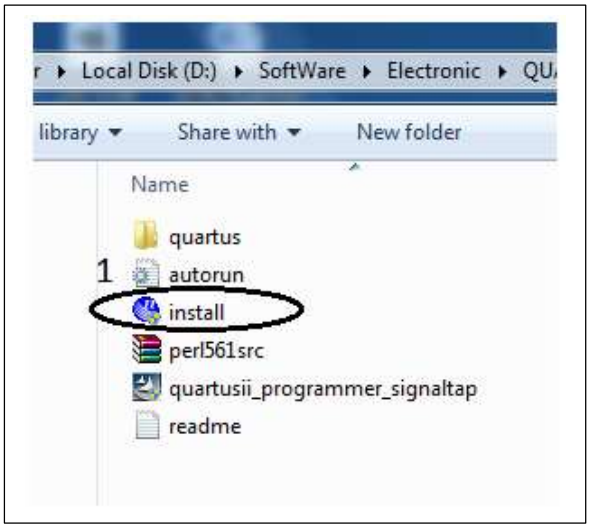
دانشکدهٔ برق و کامپیوتر

آزمایشگاه مدار منطقی

محمّدرضا فتّاح

راهنمای نصب نرم افزار

۱: فایل نصب را از داخل پوشهٔ CD1 اجرا کنید و مراحل را مطابق شکلهای زیر دنبال نمایید:



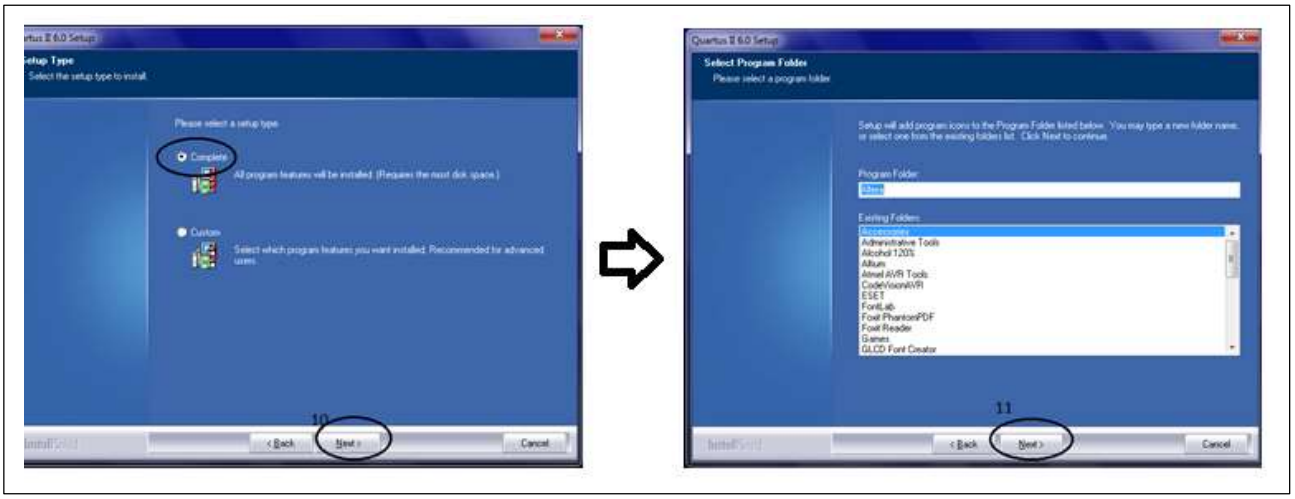
۲:

۳:

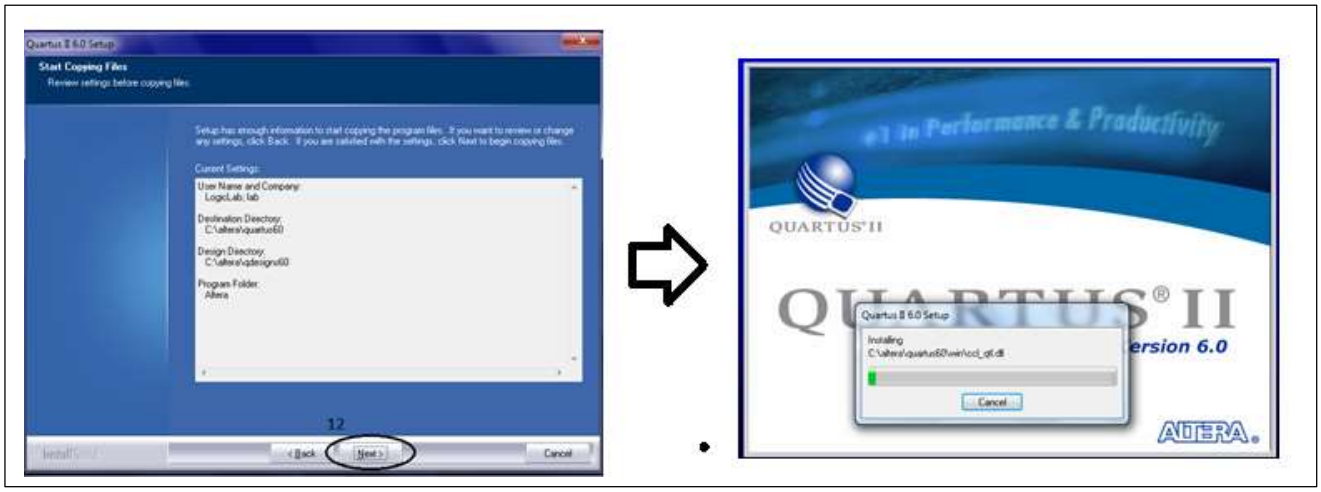
:۷


Customer Information
Please enter your information.
Please enter your name and the name of the company for which you work.
User Name:
Company Name:
lab
8
< Back
Next >
Cancel
Quartus II 6.0 Setup
Choose Destination Location
Select folder where setup will install files.
Setup will install Quartus II 6.0 in the following folder.
To install to this folder, click Next. To install to a different folder, click Browse and select another folder.
C:\altera\quartus60
Browse...
9
< Back
Next >
Cancel

:۸

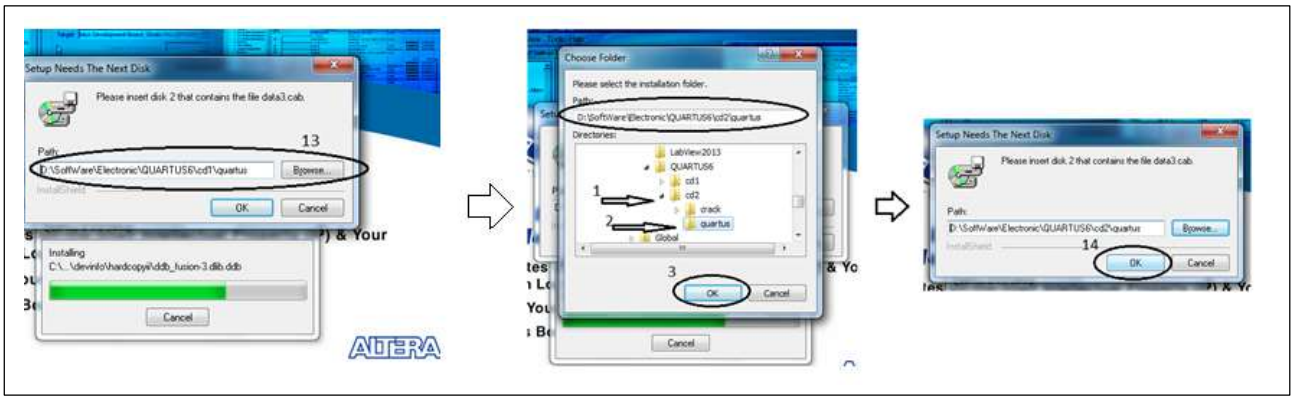

Setup Type
Select the setup type to install.
Please select a setup type.
Complete
All program features will be installed. (Requires the most disk space.)
Custom
Select which program features you want installed. Recommended for advanced users.
10
< Back
Next >
Cancel
Quartus II 6.0 Setup
Select Program Folder
Please select a program folder.
Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing folders list. Click Next to continue.
Program Folder:
Existing Folders:
Accessories
Administrative Tools
Alcohol 120%
Altium
Atmel AVR Tools
CodeVisionAVR
ESET
FontLab
Foxit PhantomPDF
Foxit Reader
Games
GLCD Font Creator
11
< Back
Next >
Cancel

:۹

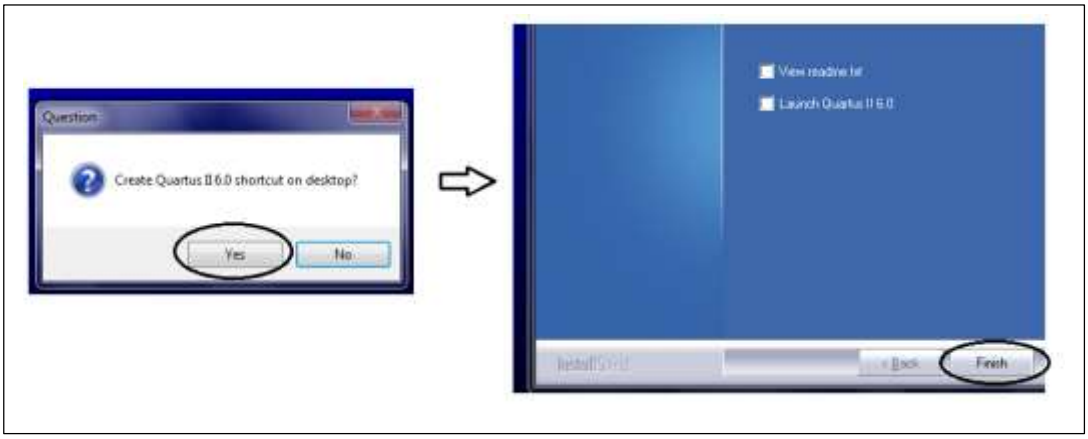

Quartus II 6.0 Setup
Start Copying Files
Review settings before copying files.
Setup has enough information to start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Next to begin copying files.
Current Settings:
User Name and Company:
LogicLab, lab
Destination Directory:
C:\altera\quartus60
Design Directory:
C:\altera\qdesigns60
Program Folder:
Altera
12
< Back
Next >
Cancel
#1 in Performance & Productivity
QUARTUS II
QUARTUS® II
Version 6.0
Quartus II 6.0 Setup
Installing
C:\altera\quartus60\win\ccl_qt.dll
Cancel
ALTERA.

•

۱۰:انتخاب بخش دوم از داخل CD2



:۱۱

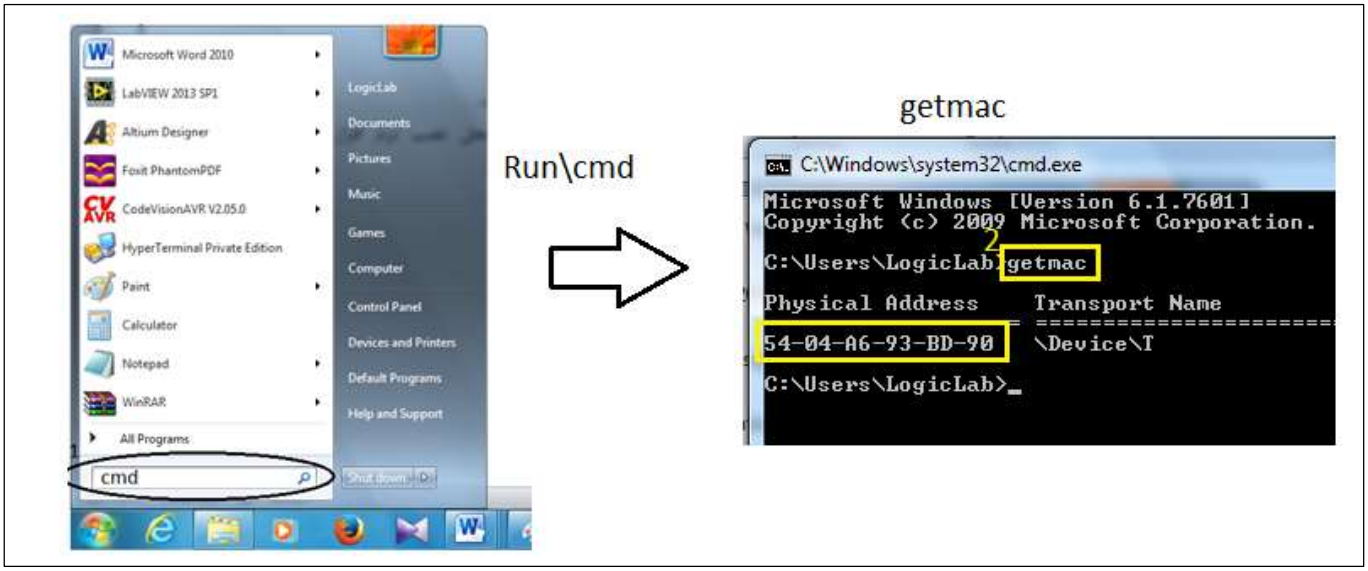


۱۷: راهنمای CRACK

۱- ابتدا پوشهٔ CD2\CRACK را در محل نصب نرم افزار کپی کنید.

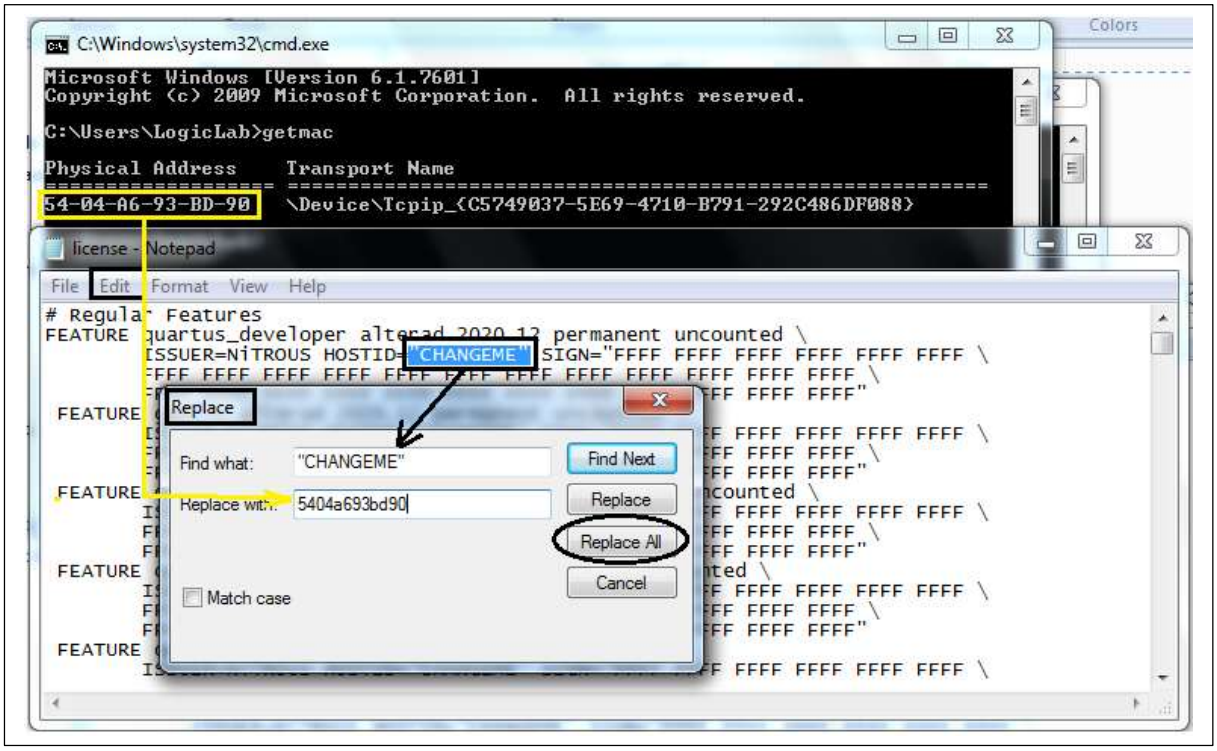
۲- فایلهای موجود در پوشهٔ CRACK\Windows را در محل نصب نرم افزار و در پوشه های altera\quartus60\bin و altera\quartus60\win کپی کنید .

۳- مراحل بعدی را طبق شکلهای زیر انجام دهید: Run\cmd

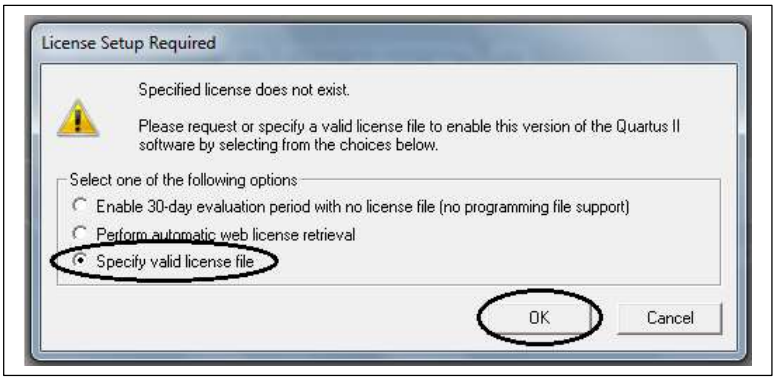


روش دیگری نیز در انتهای این متن برای بدست آوردن آدرس مک آورده شده است .

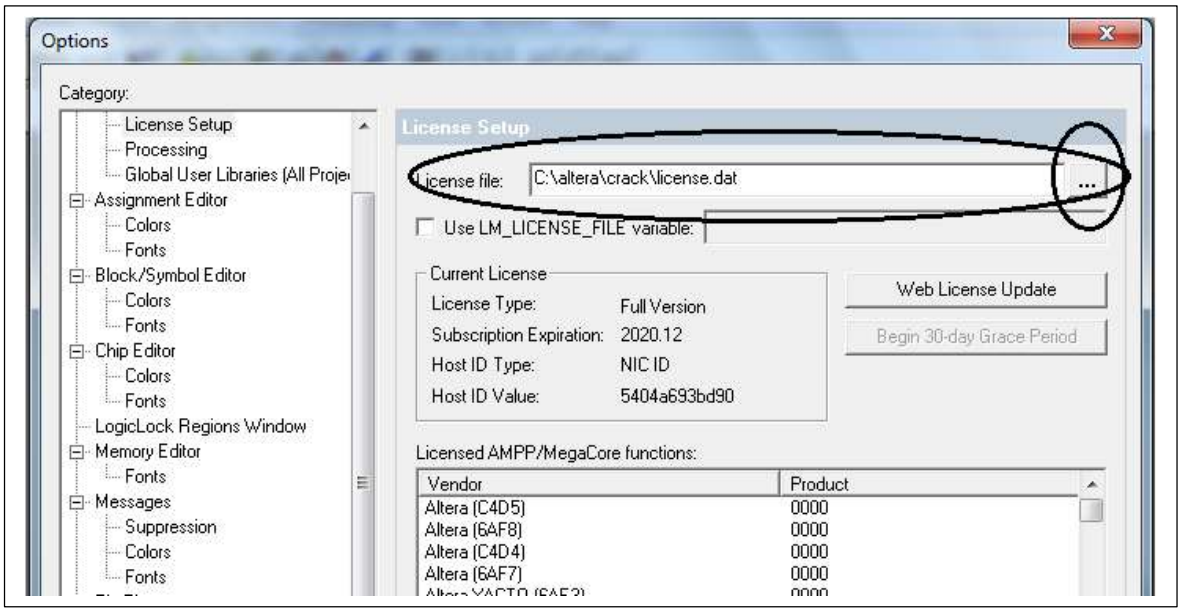
۴- از پوشهٔ CRACK فایل license را با ویرایشگر Notepad باز کنید و سپس طبق شکل زیر ازمنوی EDIT گزینهٔ Replace را انتخاب کرده و مراحل جایگزینی را انجام دهید. سپس فایل را ذخیره نمایید.



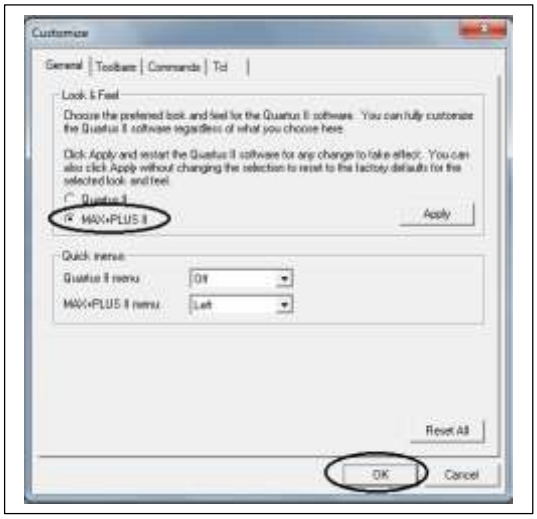
۵- نرم افزار را اجرا کنید و مراحل بعدی را مطابق شکلهای زیر دنبال کنید:



۶- انتخاب فایل license و سپس انتخاب کلید OK .



-۷



اکنون می توانید از نرم افزار استفاده کنید.