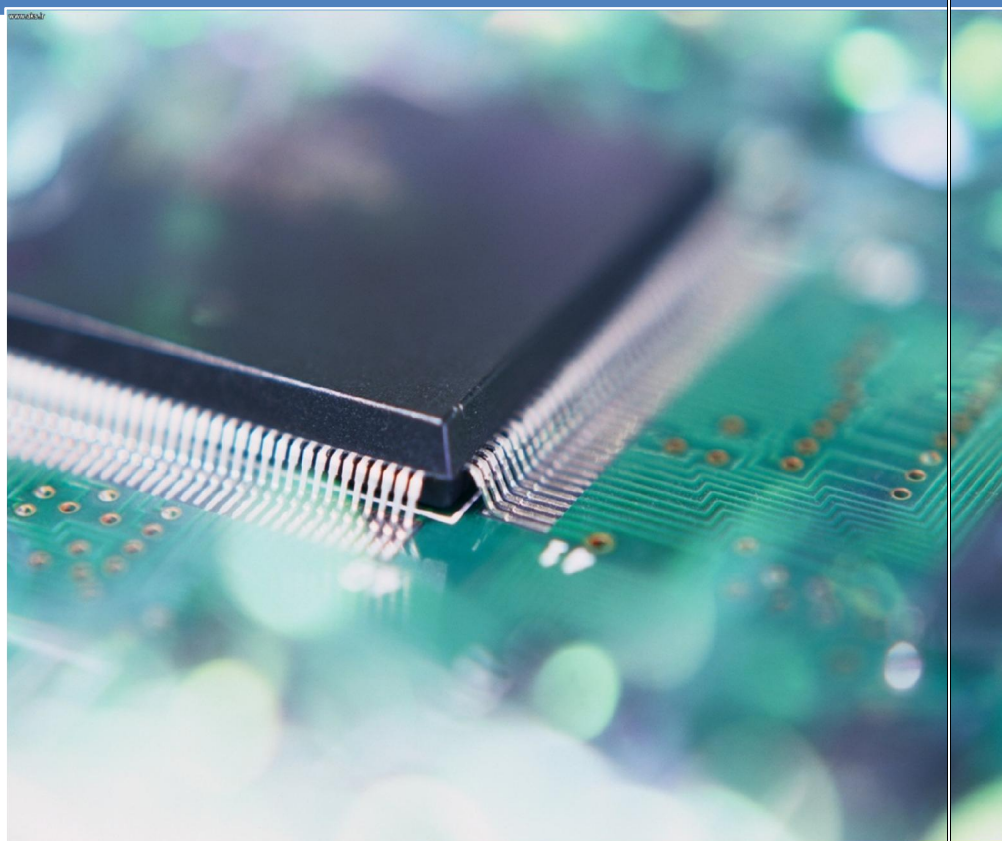


بسمه تعالی



آزمایشگاه مدارهای منطقی ۱



دانشکده برق و کامپیوتر
آزمایشگاه مدار منطقی
محمد رضا فتاح

بسمه تعالی

هدف از ارائه آزمایشگاه مدارهای منطقی ، آشنایی دانشجویان با چگونگی کار با تراشه‌های دیجیتال و همچنین چگونگی طراحی و پیاده سازی مدارات ساده ترکیبی و ترتیبی دیجیتال با روش دستی و همچنین با استفاده از زبانهای برنامه نویسی پیاده سازی 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 آنروز. راهنمای این آی سی ها در دستور کار آزمایش موجود است . لازم به ذکر است که در ابتدای هر جلسه ممکن است یک امتحانک (کوئیز) در مورد چگونگی کار یک آی سی به صورت کتبی گرفته شود .

ب- تهیه پیش گزارش : بسیاری از مدارهای که قرار است مورد آزمایش قرار گیرند لازم است که 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 . پیش گزارش باید در دو نسخه و در کاغذ A4 تهیه شود که یک نسخه آن به مربی آزمایشگاه تحویل داده می شود و نسخه دیگر در اختیار دانشجو خواهد بود تا بر اساس آن آزمایش مربوطه را انجام دهد . در بخش دوم آزمایشگاه که طراحی توسط نرم افزار انجام میشود باید پیش گزارش بصورت فایل تحویل داده شود .

۲- حضور در تمام جلسات آزمایشگاه الزامی است و در صورت غیبت ، دانشجو باید در همان هفته و در یکی دیگر از گروههای 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 سه 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 نتایج هر آزمایش به صورت شکل و جداول و نتایج خروجی مدارها ، 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 مضور دانشجو در امتحان آزمایشگاه الزامیست . در غیر این صورت بقیه درصد نمرات

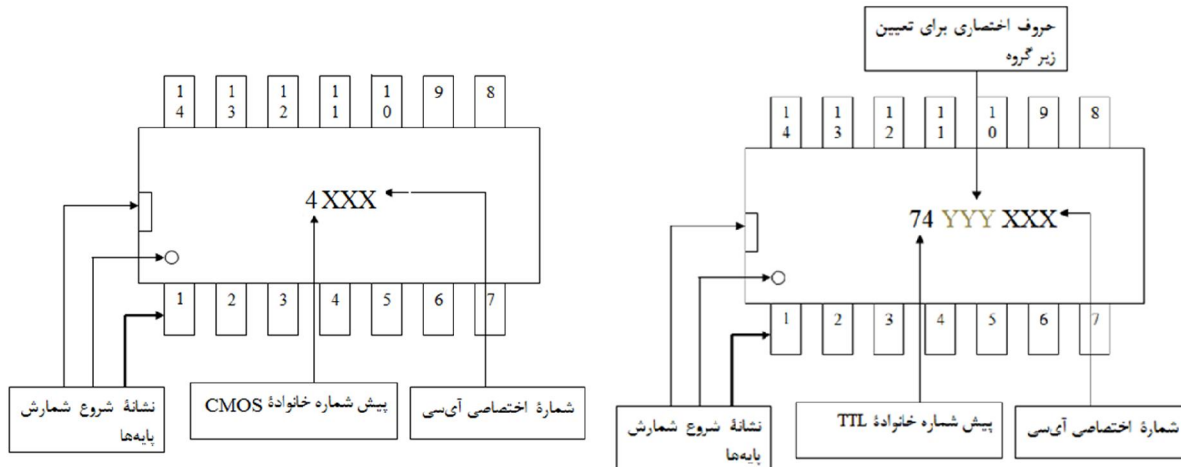
نیز لحاظ نخواهد شد .

با آرزوی توفیق برای شما

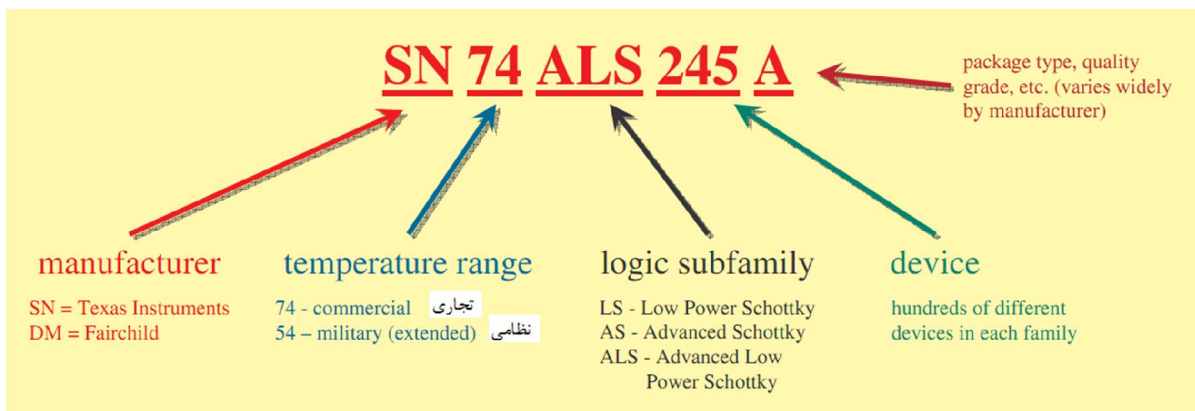
مربی آزمایشگاه : محمدرضا فتاح

مقدمه : قبل از شروع کار لازم است آشنایی اولیه در مورد انواع آی سی های منطقی و همچنین نحوه کار با آنها داشته باشیم .

متداولترین خانواده مدارهای منطقی که حاوی مدارهای پایه هستند عبارتند از خانواده CMOS و خانواده TTL. برای شناخت یک تراشه دیجیتال از نظر نوع تکنولوژی ساخت و عملکرد آن ، می توان از شماره مخصوصی که روی هر آی سی نوشته شده است استفاده کرد . با توجه به این شماره و مراجعه به کتابچه های CMOS یا TTL می توان با عملکرد آن تراشه بصورت کامل آشنا شد . متداولترین آی سی های TTL با پیشوند 74 و آی سی های CMOS با پیشوند 4 متمايز می گردند . در شکل زیر طریقه شناخت تراشه و ترتیب قرار گرفتن پایه های آن آورده شده است .



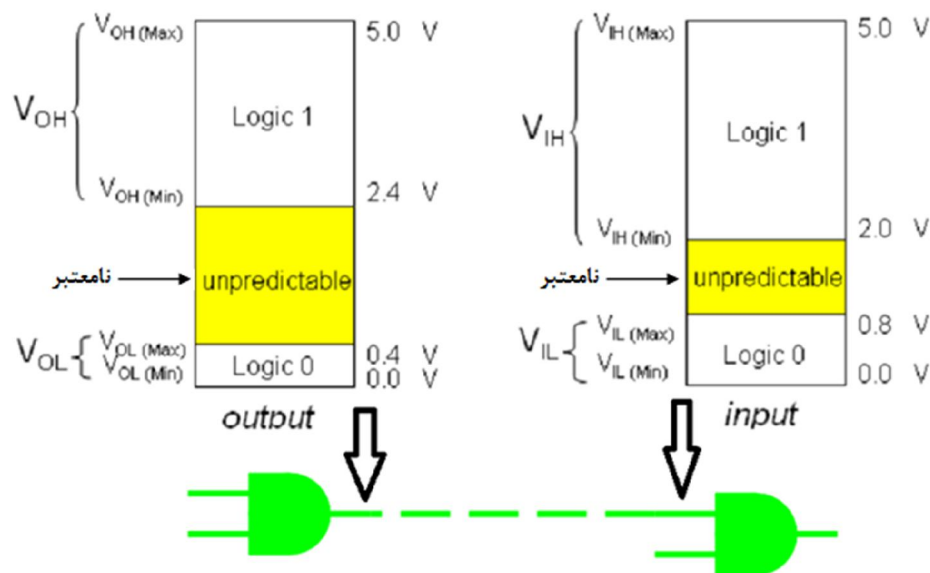
برای شناخت آی سی های TTL شکل زیر و جدول صفحه بعد را به دقت مورد توجه قرار دهید .



خانواده TTL دارای چند زیر گروه است که در جدول زیر به مهمترین آنها اشاره شده است:

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

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

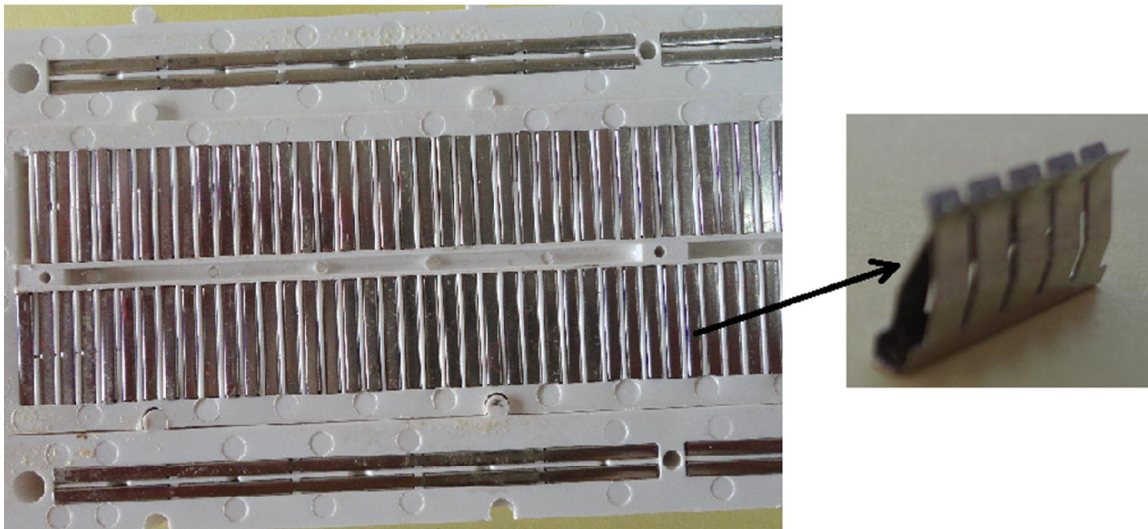


چگونگی استفاده از برد بورد در آزمایشگاه مدارهای منطقی:

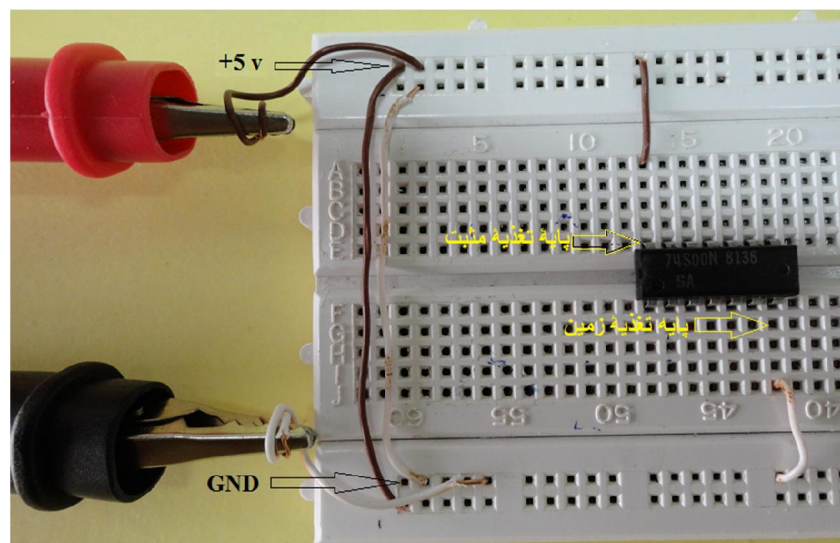
به نمای ظاهری بردبورد در شکل زیر توجه کنید . سوراخهای دو ردیف کناری در بالا و پایین بردبورد از داخل به صورت افقی به هم متصل هستند و ردیفهای عمودی دو طرف شیار وسط برد بورد به صورت عمودی به هم متصل هستند . (به خطوط قرمز توجه کنید).



همچنین اتصالات داخلی بردبورد در شکل زیر نشان داده شده است:



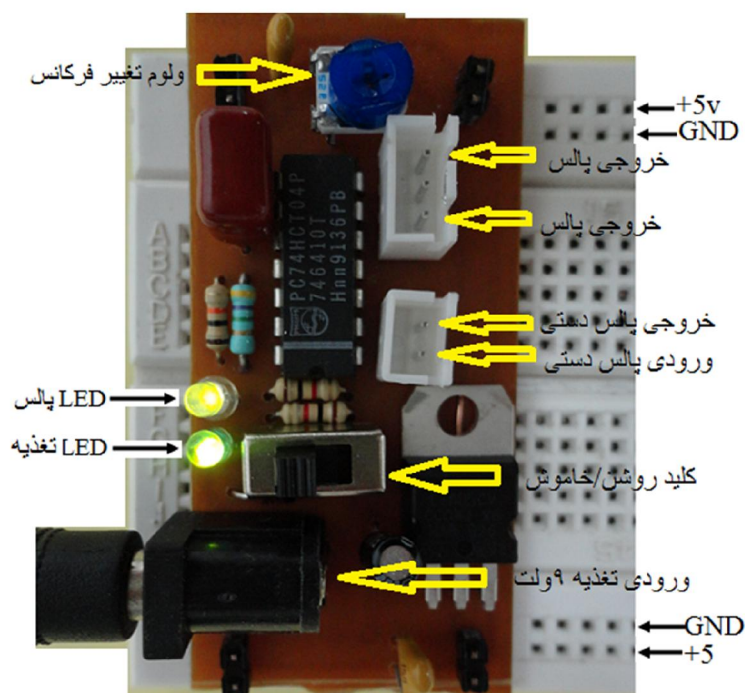
معمولاً ردیفهای دو طرف بردبورد در بالا و پایین بعنوان ردیفهای تغذیه مورد استفاده قرار میگیرد. همانطور که در شکل زیر می بینید یکی از ردیفهای بالا برای تغذیه مثبت و ردیف دیگر بعنوان ردیف زمین و یا صفر در نظر گرفته شده سپس با استفاده از دو سیم دو ردیف دیگر بردبورد نیز به تغذیه مثبت و زمین متصل گردیده است. 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 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 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.



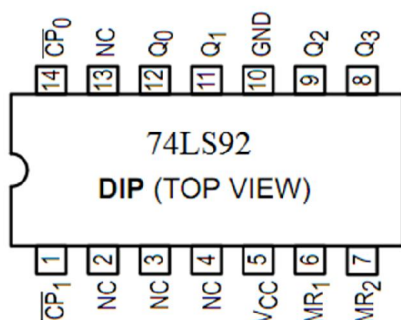
هنگام استفاده از این بورد باید به LED تغذیه و پالس توجه داشته باشید در هنگام روشن بودن بورد باید LED تغذیه در حالت روشن باشد. خاموش بودن این LED، نشانگر وجود اتصال کوتاه در مدار خواهد بود در این صورت کلید منبع را در حالت خاموش قرار داده و نسبت به رفع عیب مدار اقدام کنید.

خواندن راهنمای آی سی (Datasheet)

حتماً قبل از انجام آزمایش باید با چگونگی عملکرد یک آی سی آشنا شوید. برای این منظور سازندگان آی سی راهنمای آنرا نیز در اختیار کاربران قرار می دهند. راحتترین راه برای دسترسی به این راهنما استفاده از اینترنت است. البته در مورد آی سی های استفاده شده در هر آزمایش راهنمای آن نیز در همان آزمایش و یا انتهای آن آورده شده است. در این جا برای آشنایی بیشتر جهت استفاده بهتر از راهنما، چند اصطلاح مهم را توضیح میدهیم.

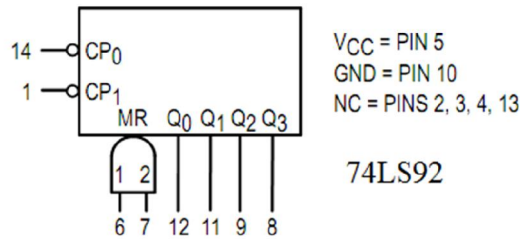
۱- CONNECTION DIAGRAM

شکل ظاهری آی سی با ذکر شماره هر پایه و اسم پایه با توجه به عملکرد آن. 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.



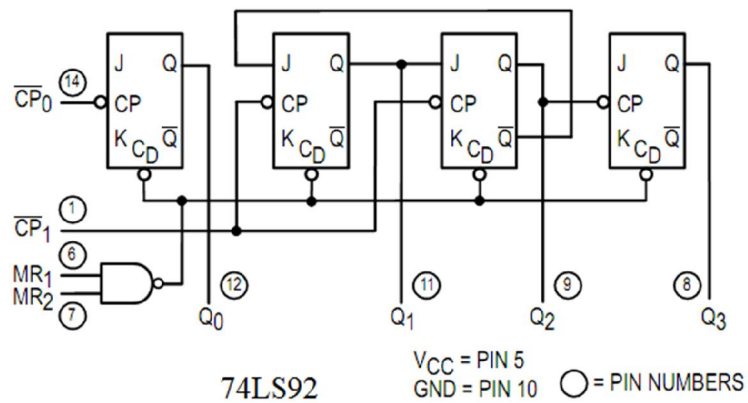
۲- LOGIC SYMBOL

معمولاً این نوع از نمایش آی سی مناسب برای ترسیم مدارات منطقی روی کاغذ و یا نرم افزارهای الکترونیک است.



۳- LOGIC DIAGRAM

در این نوع از نمایش آی سی جزئیات مدار در سطح گیت نشان داده می شود .



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 ما قرار می دهد.

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 Q₀ is connected to Input CP₁.

آزمایش اول

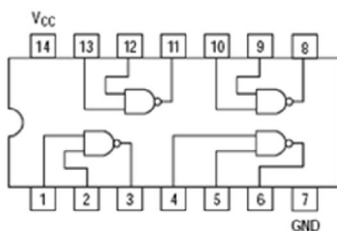
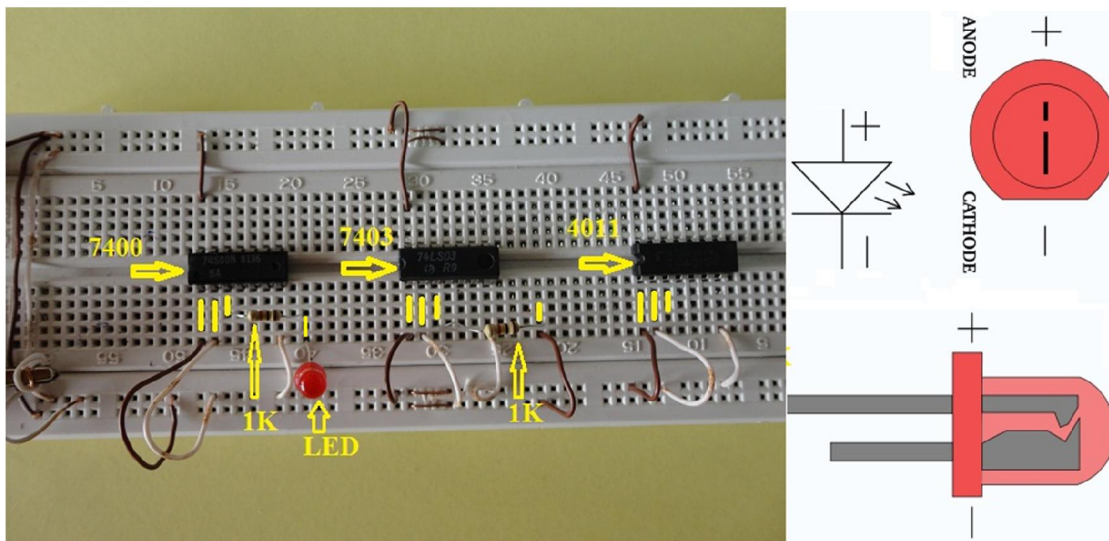
آشنایی با تراشه‌های منطقی

وسایل مورد نیاز بردبرد ، تراشه‌های ۷۴۰۰ (NAND) ، ۷۴۰۳ (NAND) و ۴۰۱۱ (NAND CMOS) و ۷۴۸۳ (جمع کننده چهار بیتی)

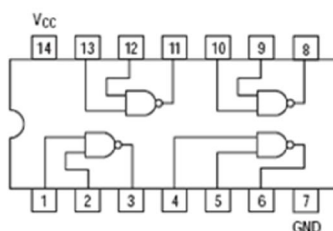
آزمایش ۱-۱

در این آزمایش یک دروازه NAND در دو خانواده متفاوت TTL و CMOS با نوع خروجی متفاوت را مورد بررسی قرار می‌دهیم . تراشه ۷۴۰۳ و ۷۴۰۰ هر دو از خانواده TTL و هر کدام حاوی چهار عدد دروازه NAND می‌باشند . اولی دارای خروجی استاندارد () و دومی دارای خروجی کلکتور باز است . همچنین تراشه ۴۰۱۱ که آنهم دارای چهار عدد دروازه NAND از خانواده CMOS است .

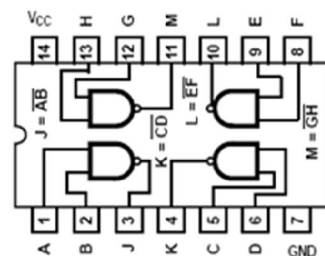
الف - ابتدا طبق شکل زیر مدار را برای انجام آزمایش آماده کنید . توجه داشته باشید قبل از اتصال خروجیهای منبع تغذیه به برد و ولتاژ آنرا برای ۵ ولت تنظیم نمایید . اکنون بصورت همزمان و طبق جدول (۱) با تغییر ورودیها ، خروجی هر سه گیت را با استفاده از ولت متر اندازه گیری کرده و در جدول یادداشت نمایید .



7400



7403



4011

		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
0	0								
0	1								
1	0								
1	1								
0	باز								
1	باز								

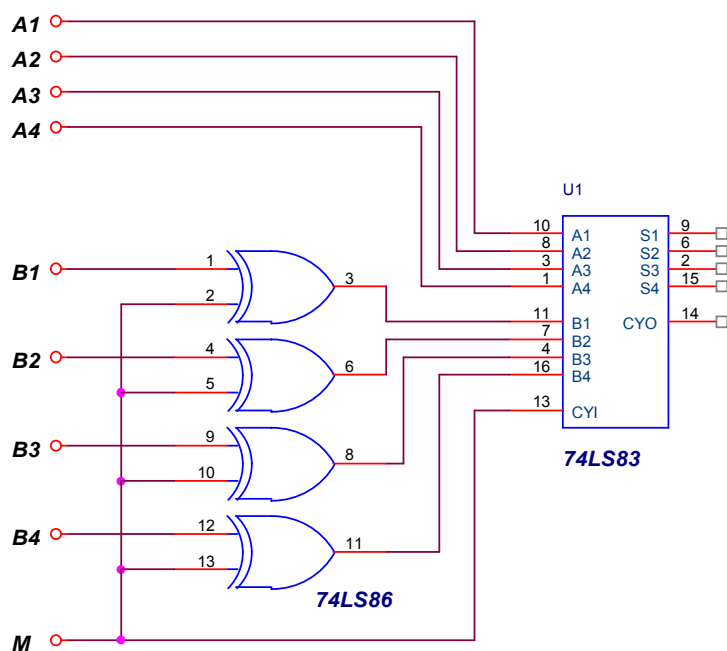
سؤال (۱) : مقدار مقاومت بین خروجی و LED بر اساس چه ملاحظات انتخاب میشود؟ رابطه آنرا بدست آورده و در گزارش کار بیاورید .

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.

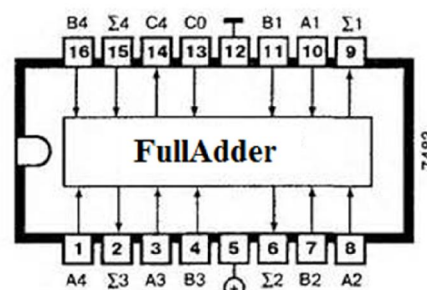
سوال (۳) ورودی باز در هر یک از خانواده TTL و CMOS دارای چه منطقی است ؟ در مورد مدار داخلی یک گیت NAND خانواده TTL و CMOS تحقیق کرده و مدار آنرا در گزارش کار بیاورید. با توجه به مدار گیت در مورد نتایج دو حالت آخر (حالت ورودی باز) توضیح دهید.

آزمایش ۲-۱

شکل زیر یک جمع کننده و تفریق کننده چهار بیتی است . ضمن توضیح در مورد طرز کار آن ، مدار را بسته و طبق جدول زیر ، مورد آزمایش قرار دهید. 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		

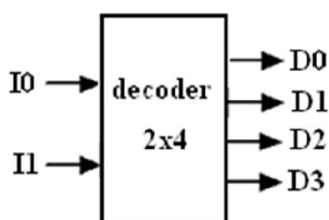


آزمایش دوم

آشنایی با مدارهای ترکیبی و ترتیبی

آزمایش ۱-۲

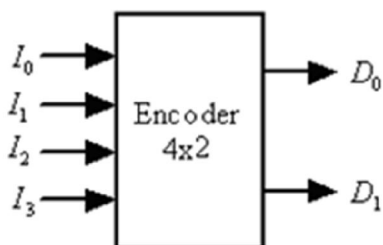
DECODER یا کدبردارها مدارهایی هستند ، با n ورودی که بر اساس هر یک از حالات ورودی یکی از 2^n خروجی آن فعال می‌گردد . اکنون با استفاده از تراشه ۷۴۰۸ (AND) و ۷۴۰۰ (NAND) یک کدبردار ۲ به ۴ را طراحی کنید و سپس آنرا مورد آزمایش قرار داده و درستی جدول حالات آنرا بررسی کنید .



A	B	D3	D2	D1	D0
0	0				
0	1				
1	0				
1	1				

آزمایش ۲-۲

ENCODER و یا کدگذارها عمل عکس کدبردارها را انجام می‌دهند . به اینصورت که دارای 2^n ورودی (حداکثر) و n خروجی می‌باشند . اگر یکی از ورودیها فعال باشد ، خروجی یک عدد n بیتی متناظر با همان ورودی خواهد بود . اکنون با استفاده از تراشه ۷۴۰۰ یک کدگذار ۴ به ۲ را طراحی کرده و جدول درستی آنرا با توجه به آزمایش مورد بررسی قرار دهید .

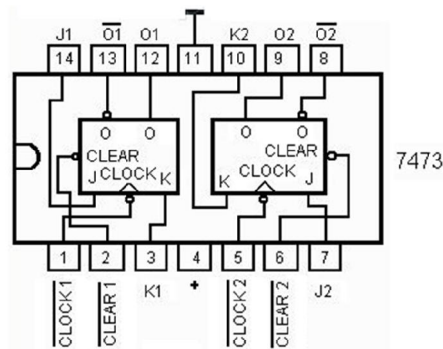


I3	I2	I1	I0	D1	D0
0	0	0	1		
0	0	1	0		
0	1	0	0		
1	0	0	0		

آزمایش ۳-۲

تراشه ۷۴۷۳ حاوی دو عدد فلیپ فلاپ JK می‌باشد ، یکی از فلیپ فلاپها را انتخاب کرده مورد آزمایش قرار دهید و جدول درستی آنرا بنویسید . خروجی را با LED مشاهده کنید .فرکانس کلاک را یک هرتز انتخاب کنید . سؤال - این فلیپ فلاپ به چه لبه ای از پالس ساعت حساس است ؟

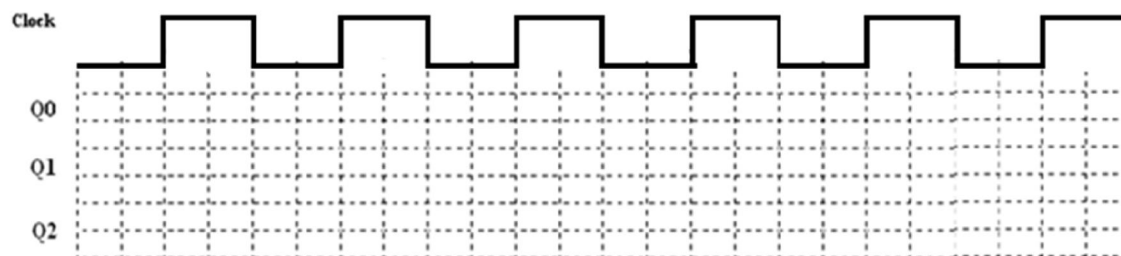
Reset	J	K	Q	/Q
0	x	x		
1	0	0		
1	0	1		
1	1	0		
1	1	1		



آزمایش ۲-۴

الف - با استفاده از فلیپ فلاپ JK و گیت NAND یک شمانده مبنای ۵ طراحی کنید (شمارش از صفر تا چهار). مدار را ببندید و با انتخاب فرکانس مناسب برای کلاک ورودی ، خروجیها را توسط LED مشاهده کنید .

ب- فرکانس کلاک را به ۱۰ کیلوهرتز افزایش دهید . خروجیهای مدار را بترتیب وبصورت جداگانه از Q0 تا Q2 همزمان با کلاک ورودی ، توسط اسیلوسکوپ مشاهده کنید . شکل موج هر خروجی را در یک دوره کامل در شکل زیر رسم نمایید.

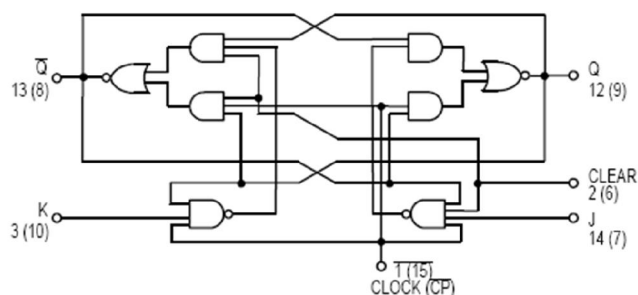




DUAL JK NEGATIVE EDGE-TRIGGERED FLIP-FLOP

The SN54LS/74LS73A offers individual J, K, clear, and clock inputs. These dual flip-flops are designed so that when the clock goes HIGH, the inputs are enabled and data will be accepted. The logic level of the J and K inputs may be allowed to change when the clock pulse is HIGH and the bistable will perform according to the truth table as long as minimum set-up times are observed. Input data is transferred to the outputs on the negative-going edge of the clock pulse.

LOGIC DIAGRAM (Each Flip-Flop)



MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS			OUTPUTS	
	$\overline{C}_D$	J	K	Q	$\overline{Q}$
Reset (Clear)	L	X	X	$\overline{L}$	H
Toggle	H	h	h	$\overline{q}$	q
Load "0" (Reset)	H	L	h	L	H
Load "1" (Set)	H	h	L	H	L
Hold	H	L	L	q	$\overline{q}$

H, h = HIGH Voltage Level

L, l = LOW Voltage Level

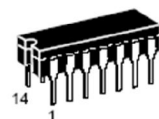
X = Don't Care

l, h (q) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the HIGH to LOW clock transition.

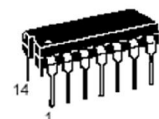
SN54/74LS73A

DUAL JK NEGATIVE EDGE-TRIGGERED FLIP-FLOP

LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 632-08



N SUFFIX
PLASTIC
CASE 646-06

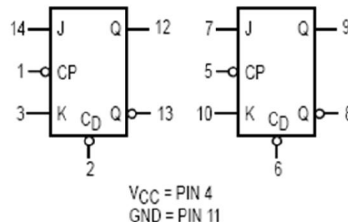


D SUFFIX
SOIC
CASE 751A-02

ORDERING INFORMATION

SN54LSXXJ Ceramic
SN74LSXXN Plastic
SN74LSXXD SOIC

LOGIC SYMBOL



آزمایش سوم

آشنایی با تراشه شمارنده ۷۴۹۰ و ثبات ۷۴۱۹۴ و کدبرد/اردهدی به هفت

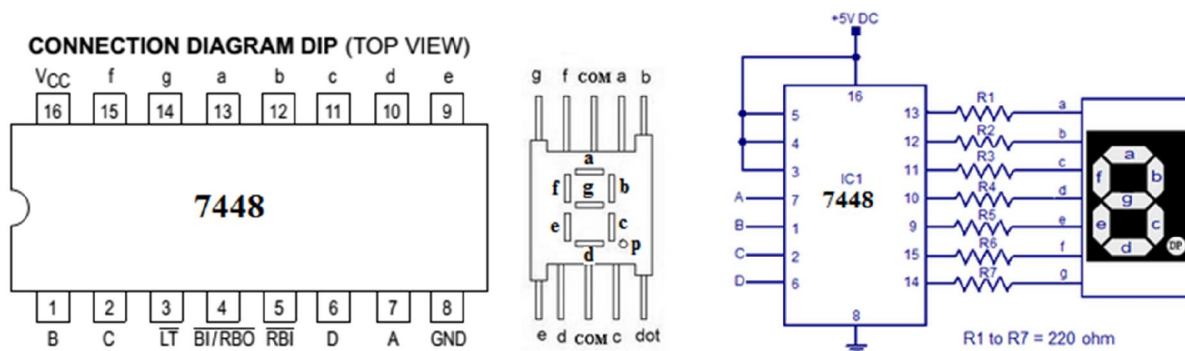
قسمتی

آزمایش ۳-۱

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 پایه شماره 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 ، اشکال نمایش داده شده توسط نمایشگر و همچنین 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 .



آزمایش ۳-۲

تراشه ۷۴۹۰ می تواند بصورت یک شمارنده مبنای ۱۰ ناهمگام عمل کند . با مراجعه به کاتالوگ این تراشه ، با شیوه کار آن آشنا شوید .

با استفاده از تراشه ۷۴۹۰ یک شمارنده مبنای ۱۰ را پیاده سازی کنید . خروجی را توسط مدار 7_seg مشاهده نمایید .

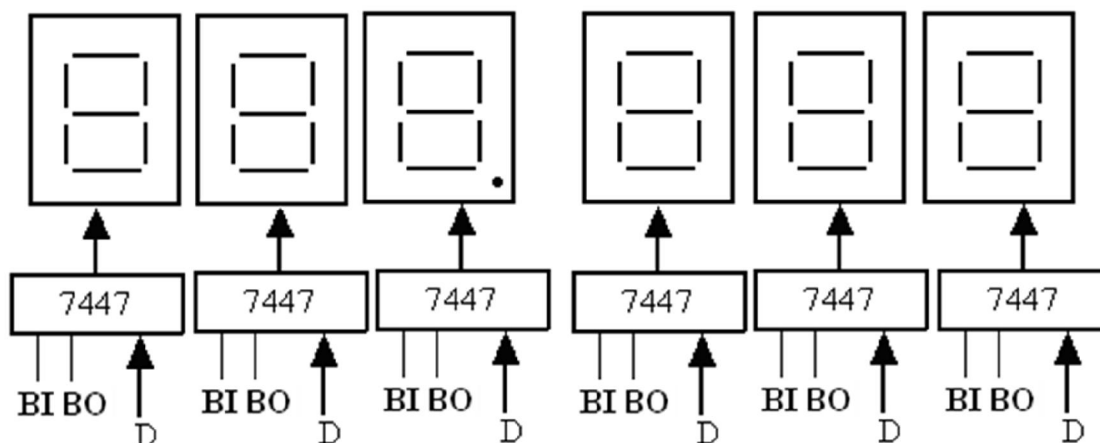
آزمایش ۳-۳

تراشه ۷۴۱۹۴ یک ثبات انتقالی چهار بیتی با امکانات ورودی موازی و سری و انتقال به راست و چپ می باشد با مراجعه به کتابچه TTL با نحوه کار آن آشنا شوید . در ادامه آزمایش زیر را با تراشه انجام دهید .

ابتدا مقدار بایتری 0110 را به ورودی موازی اعمال کنید سپس با استفاده خط Load ، آنرا بارگذاری کرده و سپس آنرا در خروجی به راست شیفต์ دهید. برای شیفต์ به چپ نیز این کار را تکرار کنید .

سؤالات :

- ۱- نقش پایه Lamp-Test در تراشه‌های ۷۴۴۷ و ۷۴۴۸ چیست ؟
- ۲- نحوه استفاده از پایه‌های blank-in و blank-out در تراشه ۷۴۴۷ را شرح دهید .
- ۳- 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 .



- ۴- با استفاده از دو شیفت رجیستر 74194، یک شیفت رجیستر هشت بیتی انتقال به راست و چپ طرح کنید . مدار را به صورت کامل ترسیم نمایید (با استفاده از Logic Symbol آی سی).



BCD TO 7-SEGMENT DECODER

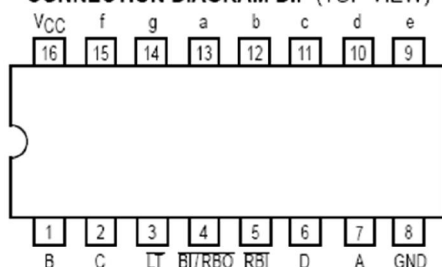
The SN54/74LS48 is a BCD to 7-Segment Decoder consisting of NAND gates, input buffers and seven AND-OR-INVERT gates. Seven NAND gates and one driver are connected in pairs to make BCD data and its complement available to the seven decoding AND-OR-INVERT gates. The remaining NAND gate and three input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

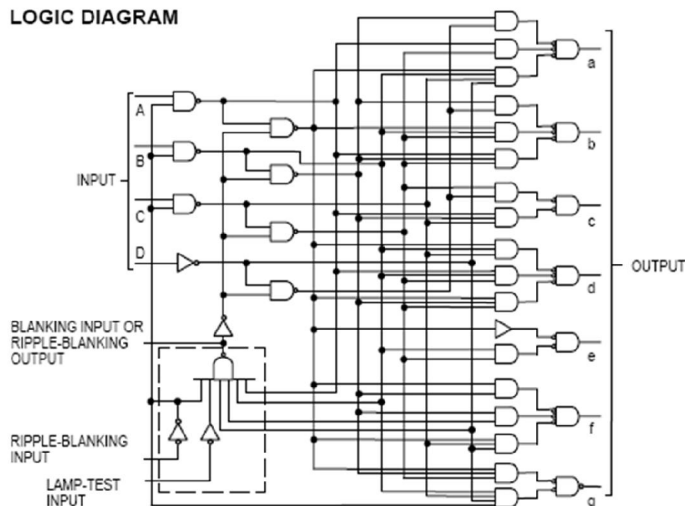
The LS48 circuit incorporates automatic leading and/or trailing edge zero-blanking control (RBI and RBO). Lamp Test (LT) may be activated any time when the BI/RBO node is HIGH. Both devices contain an overriding blanking input (BI) which can be used to control the lamp intensity by varying the frequency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



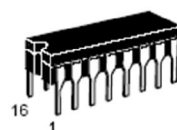
LOGIC DIAGRAM



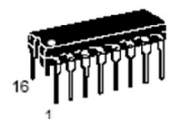
SN54/74LS48

BCD TO 7-SEGMENT DECODER

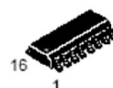
LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 620-09



N SUFFIX
PLASTIC
CASE 648-08

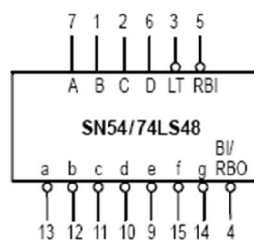


D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC

LOGIC SYMBOL



VCC = PIN 16
GND = PIN 8

SN54/74LS48

PIN NAMES

A, B, C, D	BCD Inputs
RBI	Ripple-Blanking (Active Low) Input
LT	Lamp-Test (Active Low) Input
BI/RBO	Blanking Input or Ripple-Blanking Output (Active Low)
BT	Blanking (Active Low) Input

LOADING (Note a)

HIGH	LOW
0.5 U.L.	0.25 U.L.
0.5 U.L.	0.25 U.L.
0.5 U.L.	0.25 U.L.
0.5 U.L.	0.75 U.L.
1.2 U.L.	2(1) U.L.
0.5 U.L.	0.25 U.L.
Open-Collector	3.75 (1.25) U.L. (48)

NOTES:

a) Unit Load (U.L.) = 40 μ A HIGH/1.6 mA LOW

b) Output 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

DECIMAL OR FUNCTION	INPUTS						OUTPUTS							NOTE	
	LT	RBI	D	C	B	A	BI/RBO	a	b	c	d	e	f		g
0	H	H	L	L	L	L	H	H	H	H	H	H	L	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	L	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	
BT	X	X	X	X	X	X	L	L	L	L	L	L	L	L	2
RBI	H	L	L	L	L	L	L	L	L	L	L	L	L	L	3
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 (BI) and/or ripple-blanking output (RBO). The blanking out (BI) must be open or held at a HIGH level when output functions 0 through 15 are desired, and ripple-blanking input (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 (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 (RBO) goes to a LOW level (response condition).
- (4) When the blanking input/ripple-blanking output (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.



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

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 ($\overline{Q}$ 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

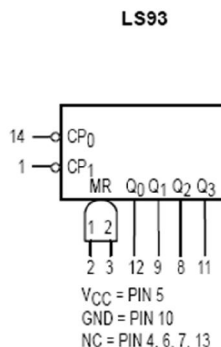
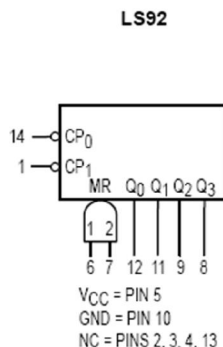
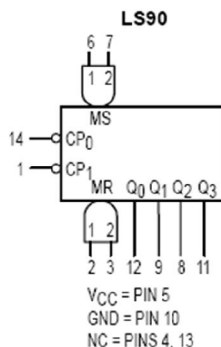
PIN NAMES

		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 μ A 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 Q₀ 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.

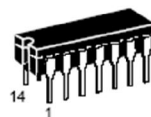
LOGIC SYMBOL



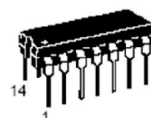
**SN54/74LS90
SN54/74LS92
SN54/74LS93**

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

LOW POWER SCHOTTKY



**J SUFFIX
CERAMIC
CASE 632-08**



**N SUFFIX
PLASTIC
CASE 646-06**



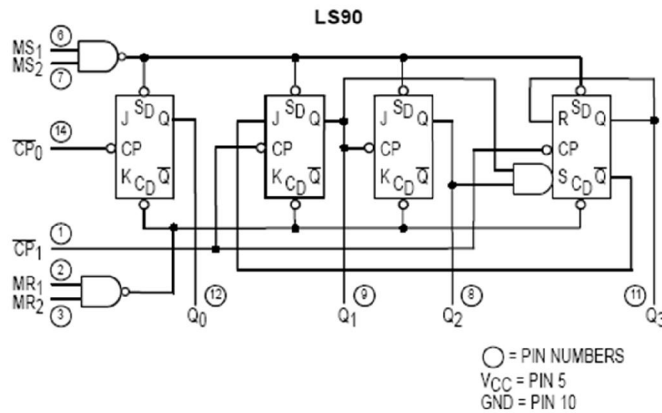
**D SUFFIX
SOIC
CASE 751A-02**

ORDERING INFORMATION

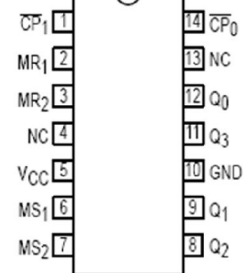
SN54LSXXJ Ceramic
SN74LSXXN Plastic
SN74LSXXD SOIC

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

LOGIC DIAGRAM



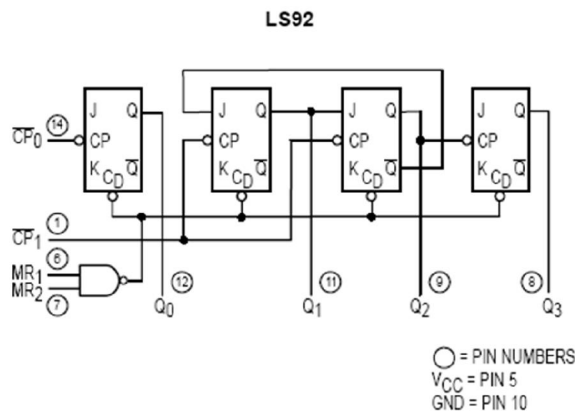
CONNECTION DIAGRAM DIP (TOP VIEW)



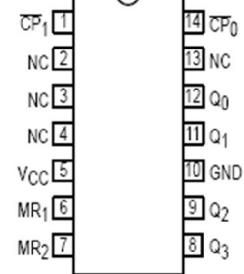
NC = NO INTERNAL CONNECTION

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

LOGIC DIAGRAM



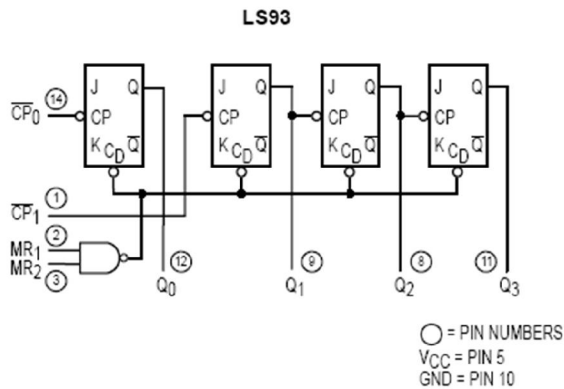
CONNECTION DIAGRAM DIP (TOP VIEW)



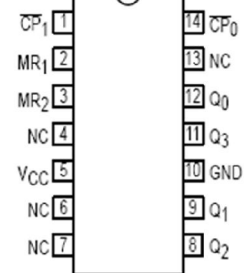
NC = NO INTERNAL CONNECTION

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

LOGIC DIAGRAM



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 Q₀ 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 ($MR_1 \cdot MR_2$) is provided on all counters which overrides and clocks and resets (clears) all the flip-flops. A gated AND asynchronous Master Set ($MS_1 \cdot MS_2$) 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 Q₀ 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 Q₃ 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 Q₀.

**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

**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 Q₀ is connected to Input $\overline{CP}_1$ 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 Q₀ is connected to Input $\overline{CP}_1$.

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

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 Q₀ is connected to Input $\overline{CP}_1$.

- 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 Q₀ as the output). The $\overline{CP}_1$ input is used to obtain binary divide-by-five operation at the Q₃ output.

LS92

- A. Modulo 12, Divide-By-Twelve Counter — The $\overline{CP}_1$ input must be externally connected to the Q₀ output. The $\overline{CP}_0$ input receives the incoming count and Q₃ 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 Q₁ and Q₂ outputs and divide-by-six operation at the Q₃ output.

LS93

- A. 4-Bit Ripple Counter — The output Q₀ 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 Q₀, Q₁, Q₂, and Q₃ 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 Q₁, Q₂, and Q₃ outputs. Independent use of the first flip-flop is available if the reset function coincides with reset of the 3-bit ripple-through counter.

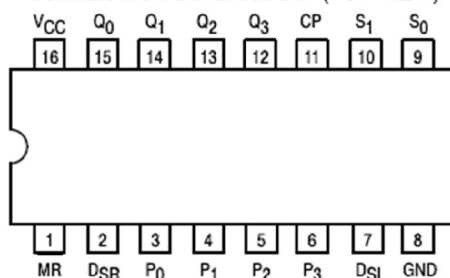


4-BIT BIDIRECTIONAL UNIVERSAL SHIFT REGISTER

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

CONNECTION DIAGRAM DIP (TOP VIEW)



PIN NAMES

S ₀ , S ₁	Mode Control Inputs
P ₀ -P ₃	Parallel Data Inputs
DSR	Serial (Shift Right) Data Input
DSL	Serial (Shift Left) Data Input
CP	Clock (Active HIGH Going Edge) Input
MR	Master Reset (Active LOW) Input
Q ₀ -Q ₃	Parallel Outputs (Note b)

LOADING (Note a)

	HIGH	LOW
S ₀ , S ₁	0.5 U.L.	0.25 U.L.
P ₀ -P ₃	0.5 U.L.	0.25 U.L.
DSR	0.5 U.L.	0.25 U.L.
DSL	0.5 U.L.	0.25 U.L.
CP	0.5 U.L.	0.25 U.L.
MR	0.5 U.L.	0.25 U.L.
Q ₀ -Q ₃	10 U.L.	5 (2.5) U.L.

NOTES:

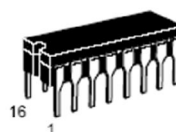
a. 1 TTL Unit Load (U.L.) = 40 μ A HIGH/1.6 mA LOW.

b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

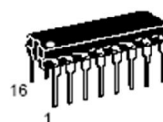
SN54/74LS194A

4-BIT BIDIRECTIONAL UNIVERSAL SHIFT REGISTER

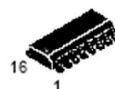
LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 620-09



N SUFFIX
PLASTIC
CASE 648-08



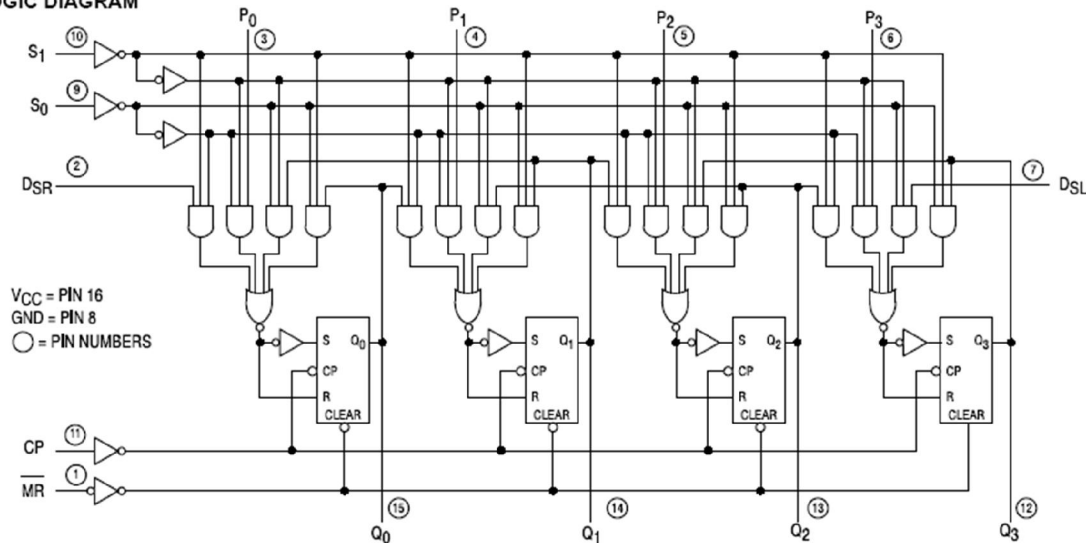
D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

SN54LSXXXJ	Ceramic
SN74LSXXXN	Plastic
SN74LSXXXD	SOIC

SN54/74LS194A

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 (P0, P1, P2, P3) are D-type inputs. When both S0 and S1 are HIGH, the data appearing on P0, P1, P2, and P3 inputs is transferred to the Q0, Q1, Q2, and

Q3 outputs respectively following the next LOW to HIGH transition of the clock.

The asynchronous Master Reset (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 (S0, S1) 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, Q0 Q1, etc.) or right to left (shift left, Q3 Q2, etc.), or parallel data can be entered loading all four bits of the register simultaneously. When both S0 and S1 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 (DSR, DSL) 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	S1	S0	DSR	DSL	Pn	Q0	Q1	Q2	Q3
Reset	L	X	X	X	X	X	L	L	L	L
Hold	H	L	L	X	X	X	q0	q1	q2	q3
Shift Left	H	h	L	X	L	X	q1	q2	q3	L
	H	h	L	X	h	X	q1	q2	q3	H
Shift Right	H	L	h	L	X	X	L	q0	q1	q2
	H	L	h	h	X	X	H	q0	q1	q2
Parallel Load	H	h	h	X	X	Pn	P0	P1	P2	P3

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

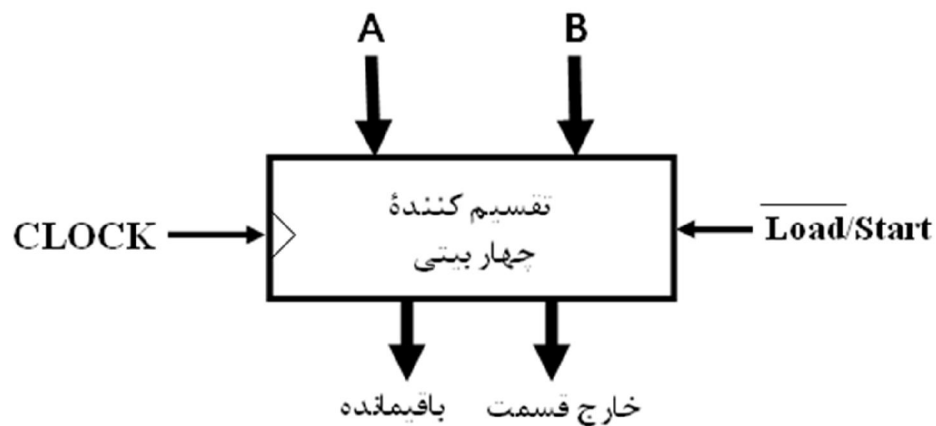
h = HIGH voltage level one set-up time prior to the LOW to HIGH clock transition

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.

آزمایش چهارم

تقسیم کننده دو عدد چهار بیت با استفاده از روش تفریق و مقایسه

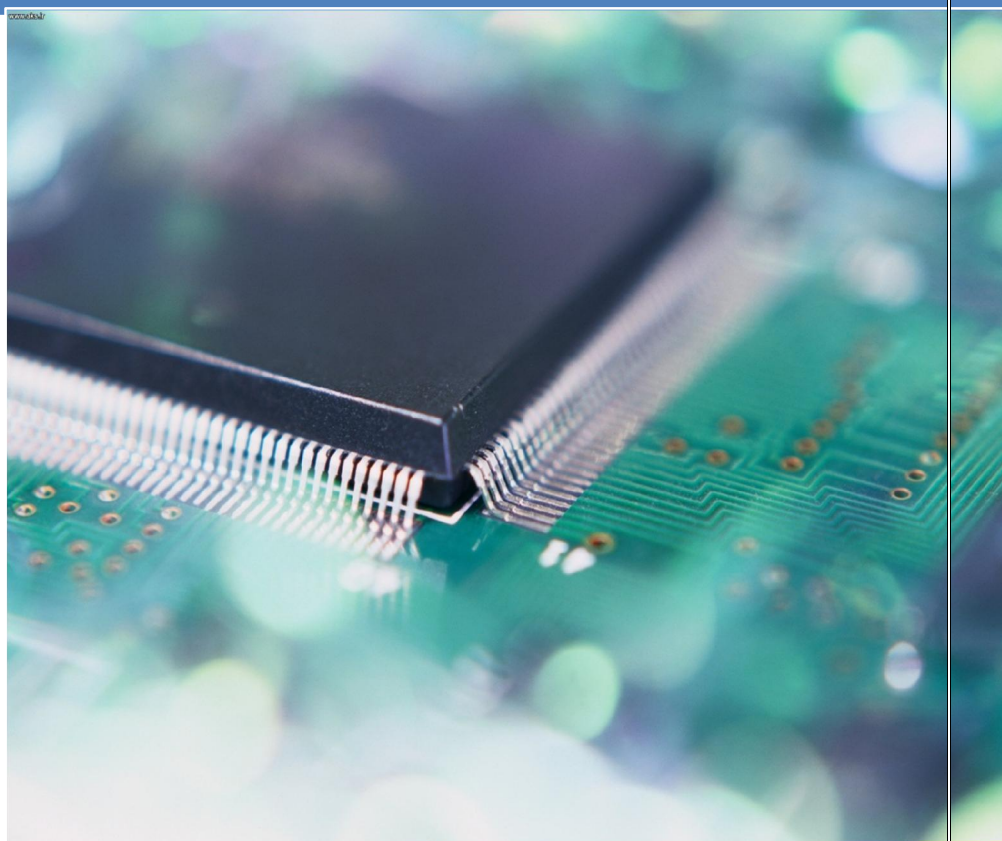
مطابق شکل زیر و با استفاده از تراشه هایی که تاکنون با آن کار کرده اید مداری طراحی کنید که بتواند دو عدد چهار بیت را به هم تقسیم کرده ($A \div B$) و در خروجیهای خود خارج قسمت و باقیمانده را نمایش دهد . مدار را با استفاده از تراشه هایی که تاکنون در آزمایشگاه از آن استفاده کرده اید طراحی کنید . توجه داشته باشید با صفر بودن خط Load/start عمل بارگذاری اعداد ورودی و با صفر شدن این خط عملیات تقسیم و نمایش خروجی انجام می شود. خروجی توسط LED نمایش داده شود.



بسمه تعالی



آزمایشگاه مدارهای منطقی ۲



دانشکده برق و کامپیوتر
آزمایشگاه مدار منطقی
محمد رضا فتاح

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 مورد آموزشی مبتنی بر آی سی های برنامه پذیر نیاز داریم .

مورد موجود در آزمایشگاه حاوی یک آی سی برنامه پذیر CPLD ساخت شرکت ALTRA است . این شرکت در زمینه 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 .

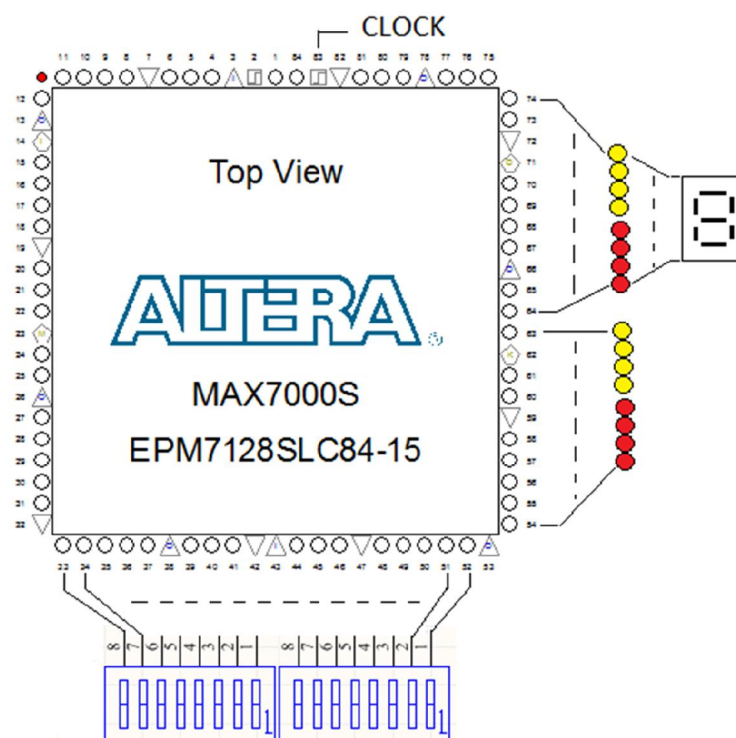
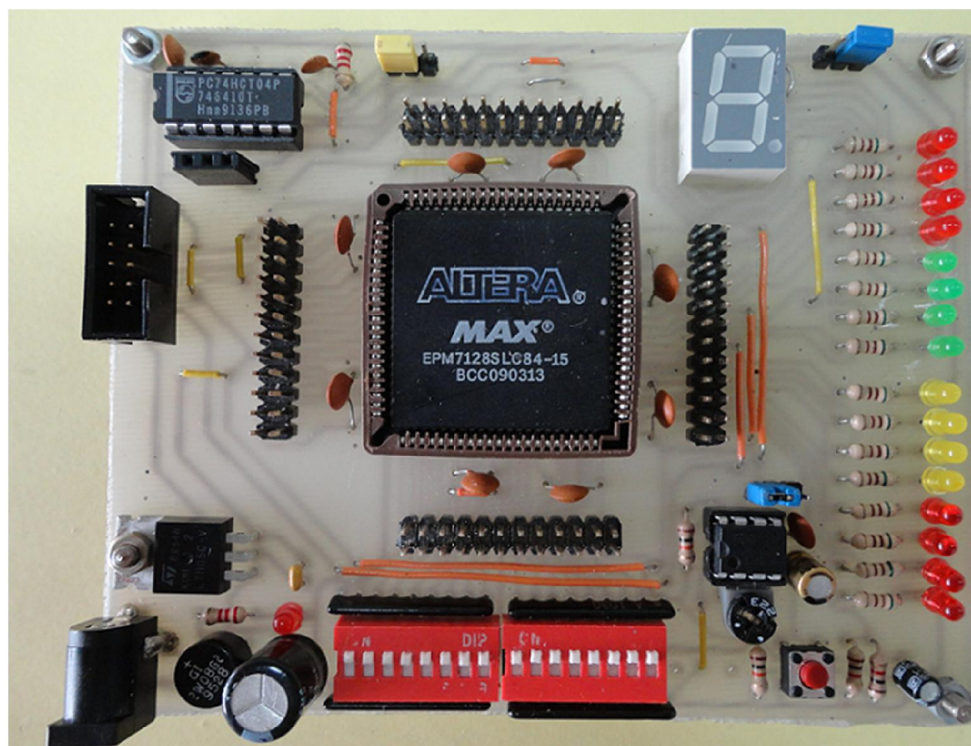
برای آشنایی دانشجویان با این نرم افزار متن جداگانه ای تهیه شده که بصورت مختصر و مفید چگونگی کار با این نرم افزار توضیح داده شده است .

معمولا طراحی مدارهای دیجیتال توسط زبانهای برنامه نویسی که مختص این کار ساخته شده اند انجام می شوند . که زبان وریلاگ (Verilog) و زبان VHDL نمونه ای از این زبانها می باشند . نرم افزار کوآرتوس 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 .البته چون متاسفانه در درس مدارهای منطقی و دروس دیگر در مورد این زبان ، آموزشی داده نمی شود دانشجو باید فعالیت بیشتری از خود بروز دهد تا بتواند نتیجه بهتری از این بخش کسب کند.

با آرزوی توفیق برای شما

مریی آزمایشگاه : محمدرضا فتاح



زبان توصیف سخت افزاری HDL

زبان توصیف سخت افزار، زبانی برای توصیف سخت افزاری سیستمهای دیجیتال به صورت متنی است. به عبارت دیگر HDL را می توان توصیف رابطه 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 ارائه 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 در سطح گیت

در مدل سازی سطح گیت مدار بصورت مجموعه ای از گیتهای پایه که به یکدیگر متصل شده اند بیان میشود. وریلاگ مانند دیگر زبانهای برنامه نویسی دارای گیتهای منطقی، خطوط انتقال و سوئیچها است. برای شروع، انواع گیتهای پایه که در این زبان وجود دارد را معرفی می کنیم. در این زبان شش گیت پایه وجود دارد. این گیتها عبارتند از گیتهای AND، OR، NOR، XOR و XNOR. همه این گیتها دارای چند ورودی دلخواه و یک خروجی هستند. همچنین این زبان دارای شش گیت انتقالی اصلی است: وارونگر، بافر، بافر سه حالته فعال پایین و بالا و وارونگر سه حالته فعال پایین و بالا. به جدول زیر دقت کنید.

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

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

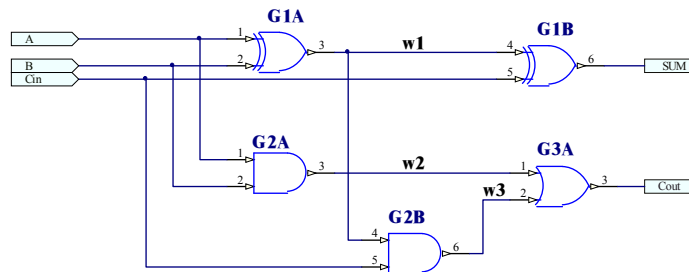
شکل ۱

توجه داشته باشید که زبان وریلاگ یک زبان متنی است و شکلهای موجود در جدول بالا فقط برای درک بهتر عملکرد هر گیت آورده شده است.

برای طراحی مدار در سطح گیت ابتدا باید آنرا به صورت مجموعه ای از گیت‌های پایه درآورد سپس با تعریف اتصالات مناسب ، این گیتها را به هم وصل نمود . در مثال زیر به توضیح این مورد می پردازیم .

جمع کننده کامل یک بیتی (Full Adder)

مدار شکل ۲ تمام جمع کننده یک بیتی را نشان می دهد که با استفاده از گیت‌های پایه ساخته شده است. در شکل به اسامی ورودی و خروجیها و همچنین به اسامی دروازه های منطقی و نام اتصالات میانی دقت کنید و سپس



شکل ۲

برنامه زیر که توصیف مدار در زبان وریلگ است را مطالعه نمایید.

//Verilog model of Full Adder Fig 2

```
module fulladder_1bit (SUM , Cout , A , B , Cin); // معرفی مدار با ذکر نام و لیست ورودیها و خروجیها
    output SUM , Cout; // تعریف خروجیها
    input A , B , Cin; // تعریف ورودیها
    wire w1 , w2 , w3; // تعریف اتصالات میانی (سیمها)
    xor G1A ( w1 , A , B );
    and G2A ( w2 , A , B );
    and G2B ( w3 , w1 , Cin );
    xor G1B ( SUM , w1 , Cin );
    or G3A ( Cout , w2 , w3 );
endmodule
```

توضیح برنامه:

در برنامه وریلگ بالا جمله اول که با // شروع شده به عنوان توضیح در نظر گرفته می شود و تاثیری در برنامه ندارد . در وریلگ عبارتهای توضیح چند جمله ای با /* شروع و با */ خاتمه می یابد.

واحد ساختاری وریلگ "ماژول" نامیده می شود. برنامه بالا در اصل یک ماژول است که با کلید واژه module شروع و انتهای آن با کلید واژه endmodule خاتمه می یابد . خطوط بین این دو کلیدواژه که بدنه ماژول را تشکیل می دهد در اصل توصیف سطح گیت مدار شکل (۲) است . بعد از کلمه module یک نام برای مدار آورده میشود که در دنباله آن و در داخل پرانتز نام ورودیها و خروجیها ذکر میگردد .

کلمه fulladder_1bit ، نام ماژول و یک شاخص (identifier) است . شاخص نامیست که به ماژولها ، متغیرها و دیگر اجزاء زبان داده می شود تا بتوان در طراحی به آن اشاره کرد . توجه داشته باشید که شاخصها به حروف کوچک و بزرگ حساس هستند که باید با حروف الفبا و یا _ (underscore) شروع شوند .

برخی از کلید واژه های مهم در زبان عبارتند از : module ، endmodule ، input ، output ، wire ، and ، or ، xor ، not . با واژه های output و input و wire دریاچه های ورودی و خروجی و همچنین سیمهای میانی مشخص می شوند. در این مدار دریاچه های A ، B و Cin ورودی و دریاچه های SUM و Cout بعنوان دریاچه خروجی تعریف میگردند . همچنین با کید واژه wire اتصالات میانی با نامهای دلخواه w1 ، w2 و w3 مشخص می شوند .

چند خط بعدی برنامه مربوط به اعلان دروازه های منطقی (گیتها) مدار است . این مدار از پنج گیت منطقی با نامهای دلخواه G1A(xor) ، G1B(xor) ، G2A(and) ، G2B(and) و G3A(or) تشکیل شده است . در هر اعلان از کلید واژه متناسب با دروازه مربوطه و یک نام دلخواه و لیست ورودیها و خروجیهای همان گیت استفاده می شود. ورودیها و خروجیها ، داخل پرانتز گذاشته می شوند به طوریکه ابتدا خروجی و سپس ورودی ذکر می شود . ترتیب ورودیها مهم نیست .

توجه داشته باشید که باید در انتهای هر گزاره علامت ; گذاشته شود بجز انتهای واژه endmodule .

تأخیر دروازه :

در مدارات واقعی ، هر گیت دارای یک تأخیر زمانی است . برای اعمال این تأخیر در شبیه سازی توسط وریلاگ میتوان برنامه را بصورت زیر بازنویسی کرد:

```
//Verilog model of Full Adder Fig 2
'timescale 1ns/100ps
module fulladder (SUM , Cout , A , B , Cin);
    output SUM , Cout;
    input A , B , Cin;
    wire w1 , w2 , w3;
    xor #(35) G1A ( w1 , A , B );
    and #(30) G2A ( w2 , A , B );
    and #(30) G2B ( w3 , w1 , Cin );
    xor #(35) G1B ( SUM , w1 , Cin );
    or #(20) G3A ( Cout , w2 , w3 );
endmodule
```

جمله 'timescale 1ns/100ps یک راهنمای کامپایلر است که قبل از اعلان مدول می آید و واحد زمانی تأخیر را تعیین می کند . عدد اول واحد زمانی را برحسب نانو ثانیه و عدد دوم دقت واحد زمانی را ۱۰۰ پیکو ثانیه مشخص می کند . در بدنه برنامه برای اعمال تأخیر برای هر دروازه از نماد # استفاده می شود و مقدار تأخیر بعد از این نماد و در داخل پرانتز تعریف می گردد.

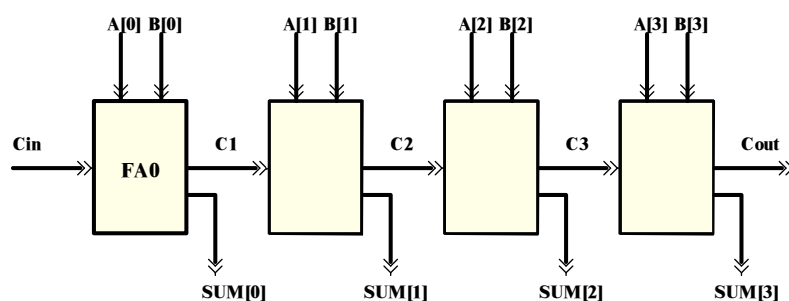
آزمایش ۵-۱: طراحی جمع کننده کامل یک بیتی

با مطالعه راهنمای نرم افزار کوارتوس این مدار را طراحی ، شبیه سازی و پیاده سازی کنید .

توجه: نتایج هر آزمایش روی برد آزمایشگاه نمایش داده شود.

توصیف یک جمع کننده چهاربیتی با استفاده از ماژول جمع کننده یک بیتی

شکل زیر یک جمع کننده چهاربیتی است که با استفاده از اتصال چهار جمع کننده یک بیتی ساخته شده است .
حال چگونه میتوانیم این مدار را در محیط وریلاگ و توصیف سطح گیت پیاده سازی کنیم؟ برنامه زیر جواب این سوال را مشخص می کند .



شکل ۵

//Verilog model of 4bit Full Adder Fig 3

```
module fulladder4bit ( SUM , Cout , A , B , Cin);  
    output [3:0] SUM;  
    output Cout;  
    input [3:0] A , B;  
    input Cin;  
    wire c1 , c2 , c3;  
    fulladder_circuit FA0 (SUM[0] , c1 , A[0] , B[0] , Cin);  
    fulladder_circuit FA1 (SUM[1] , c2 , A[1] , B[1] , c1 );  
    fulladder_circuit FA2 (SUM[2] , c3 , A[2] , B[2] , c2 );  
    fulladder_circuit FA3 (SUM[3] , Cout , A[3] , B[3] , c3 );  
endmodule
```

همانطور که در متن برنامه مشاهده می کنید چهار بار از جمع کننده یک بیتی که در مثال قبل طراحی کرده بودیم استفاده نموده و یک جمع کننده چهار بیتی ساخته ایم .

در این برنامه ورودیهای A ، B و SUM بصورت یک بردار چهار بیتی تعریف شده اند در این حالت ابتدا اندازه بردار در داخل [] (کروشه باز ، کروشه بسته) آورده می شود که اولین عدد شماره پر ارزشترین بیت سپس علامت : و بعد از آن شماره کم ارزشترین بیت قرار می گیرد و سپس نام ورودی و یا خروجی آورده می شود

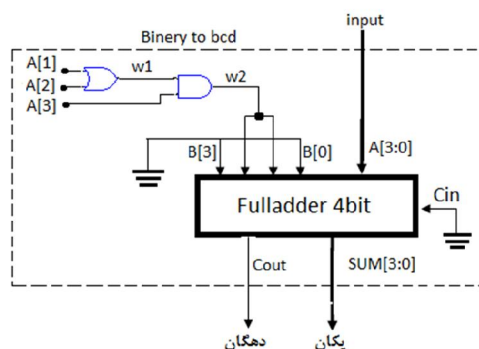
آزمایش ۵-۲ : جمع کننده چهار بیتی

برنامه جمع کننده چهار بیتی را توسط نرم افزار و در همان پروژه قبلی پیاده سازی کنید .
راهنمایی : برای انجام این تمرین ابتدا در پروژه قبلی یک فایل جدید متنی وریلاگ بسازید و برنامه بالا را وارد کنید و فایل را با نام fulladder4bit که همان نام ماژول است ذخیره نمایید .

The figure shows two screenshots of the Quartus II software interface. The first screenshot shows the 'Project' menu with 'Set as Top-Level Entity' highlighted. The second screenshot shows the 'Settings - fulladder1bit' dialog box with 'fulladder1bit.v' selected in the 'Files' list. Arrows indicate the sequence of steps: 1. Open Project menu, 2. Select 'Set as Top-Level Entity', 3. Open Settings dialog, 4. Select the file in the Files list.

شکل ۷

آزمایش ۵-۳ : مبدل باینری به BCD چهار بیتی



شکل بالا مدار مبدل باینری به BCD چهار بیتی را نشان می دهد . با استفاده از مدار جمع کننده چهاربیتی، برنامه این مدار را بنویسید. شبیه سازی و اجرای آنرا را روی بورد مشاهده نمایید .

تمرین:

یک مقایسه کننده چهار بیتی توسط وریلاگ طراحی کنید .نتیجه را در محیط شبیه سازی نمایش دهید.
توجه: تمرینهای داده شده در پایان هر آزمایش بعنوان گزارش کار در جلسه بعد بصورت فایل کوارتوس تمویل داده شود.

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 این حالت از کلمه کلیدی assigned برای تخصیص مداوم یک مقدار به یک داده از نوع net استفاده می گردد. داده نوع net برای نشان دادن اتصال فیزیکی بین دو عنصر مدار بکار می رود. نت بصورت صریح با یک کلید واژه نوع نت (wire) یا با اعلان یک متغیر بعنوان دریچه خروجی معرفی می شود. برای درک بهتر این بخش به مثالهای ساده زیر توجه کنید.

مثال ۱: توصیف یک گیت And

```
module and_gate( out , a , b );
    output out;
    input a , b ;
    assign out = a & b ;
endmodule
```

البته این توصیف را بصورت زیر نیز میتوان نوشت :

```
module and_gate( out , a , b );
    output out;
    input a , b ;
    wire out = a & b ;
endmodule
```

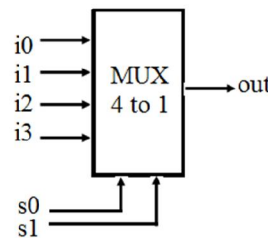
روش اول را تخصیص دهی مداوم و روش دوم را تخصیص دهی ضمنی می گویند. جدول زیر عملگرهای موجود در زبان وریلاگ را نشان می دهد.

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

آزمایش ۶-۱: مالتی پلکسر ۴ به ۱

جدول درستی و شکل نمادین یک مالتی پلکسر ۴ به ۱ در زیر نشان داده شده است. در کنار آن توصیف مدار بر اساس مدل جریان داده آمده است.

```
module mux4to1(out , i0 , i1 , i2 , i3 , s1 , s0);
output out;
input i0,i1,i2,i3,s1,s0;
assign out=s1 ? ( s0 ? i3 : i2 ) : ( s0 ? i1 : i0 )
endmodule
```



s1	s0	out
0	0	i0
0	1	i1
1	0	i2
1	1	i3

شکل ۸

توضیح عبارت شرطی آخر برنامه:

condition ? true expression : false expression

عبارت نادرست : عبارت درست ؟ شرط

مثال ۳: جمع کنندهٔ چهار بیتی (مدار شکل ۵) با روش جریان داده

```
module fulladder4bit ( SUM , Cout , A , B , Cin);
output [3:0] SUM;
output Cout;
input [3:0] A , B;
input Cin;
assign { Cout , SUM } = A + B + Cin;
endmodule
```

در مثال اخیر خروجی SUM که چهاربیتی است و خروجی Cout با عملگر {} که عملگر الحاق نامیده می شود با هم ترکیب شده و یک متغیر پنج بیتی را تشکیل می دهند. در این حالت چهار بیت نتیجه حاصل جمع مربوطه به متغیر SUM و بیت پنجم نتیجه به خروجی Cout تخصیص دهی می شود.

مثال ۴: مقایسه کنندهٔ یک بیتی با روش جریان داده

```
module comp1bit ( Gt , Eq , Lt , a , b )
output Gt , Eq , Lt ;
input a , b;
assign Gt = a & ~b ;
assign Lt = ~a & b ;
assign Eq = (~a & ~b) | ( a & b ) ;
endmodule
```

input		output		
A	B	Gt	Eq	Lt
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

این برنامه با استفاده از عملگرهای بیتی (and) & و (not) ~ و با توجه به جدول درستی نوشته شده است.

مثال ۶ را میتوان بصورت دیگری هم نوشت:

```
module      complbit (output  Gt , Eq , Lt ,
                        input  a , b );
    assign   Gt = (a > b) ,
            Lt = (a < b) ,
            Eq = (a == b) ;
endmodule
```

در این روش از گزاره های شرطی برای توصیف مدار استفاده شده است که نیازی به دانستن جدول درستی نیست. همانطور که در خط اول برنامه مشاهده می کنید میتوان تعریف ورودی و خروجیها را همزمان با اعلان آنها انجام داد.

آزمایش ۲-۴ : دیکودر ۲ به ۴

```
module      decoder2to4 (input [1:0] in , output [3:0]out , input enable );
    assign   out = enable ? ( 1 << in ) : 4'h0 ;
endmodule
```

در این مثال از عملگر شیفت به چپ << استفاده شده است. معنی جمله داخل پرانتز این است که عدد ۱ به اندازه متغیر in به سمت چپ شیفت داده شود.

نکته : نمایش اعداد ثابت در ورilaگ

ساختار کلی نمایش اعداد در این زبان بصورت زیر است:

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

مثال : در جدول زیر چند نمونه از روش نمایش اعداد ثابت در ورilaگ 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 مبناى باینری
8'h0f	00001111	با ذکر اندازه: هشت بیت در مبناى هگز
16'h1FFA	0001 1111 1111 1010	با ذکر اندازه: شانزده بیت در مبناى هگز
8'd15	00001111	با ذکر اندازه: هشت بیت در مبناى دسیمال
8'hzz		با ذکر اندازه: هشت بیت امپدانس بالا
8'hxx		با ذکر اندازه: هشت بیت مقدار نامعتبر

آزمایش ۳-۴ : برنامه مبدل باینری به BCD را با استفاده از روش جریان داده بنویسید و اجرا کنید.

توجه: به برنامه مالتی پلکسر دقت کنید.

تمرین :

مدل سازی رفتاری

توصیف مدارهای دیجیتال در سطح الگوریتمی و عملیاتی را توصیف رفتاری می گوئیم. این مدل چگونگی عملکرد مدار را توضیح و 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 کلیدواژه 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 روندی قرار می گیرد. خروجی در بلوک always از نوع ثبات می باشد که با کلیدواژه 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. همه این بلوکها بصورت همزمان اجرا می شوند.

مثال ۱: مبدل باینری به BCD

```
module bin2bcd (input [3:0] a , output [3:0] digit1 , output digit2 );
    reg digit2 , [3:0]digit1;
    always @ ( a )
        if (a > 4'b1001 )
            {digit2,digit1} = a + 4'b0110;
        else
            {digit2,digit1} = a + 4'b0000;
endmodule
```

در برنامه بالا خروجیهای digit1 و digit2 یکبار با کلیدواژه output بعنوان خروجی تعریف شده و بار دیگر با کلیدواژه reg بعنوان یک ثبات تعریف گردیده است چرا که خروجی در مدل رفتاری باید از نوع ثبات باشد. دستورات بعد از کلیدواژه always وقتی اجرا می شوند که تغییری در سیگنال ورودی a صورت گیرد در غیر این صورت اجرا تا تغییر بعدی متوقف می شود.

کنترل زمان در بلوک always

بطور کلی بلوک always در زمان صفر شروع بکار می کند. در ورپلاگ می توان یک بلوک را طوری کنترل نمود تا در زمان خاصی مثلا در لبه یک پالس ساعت فعال شود. کنترل زمان می تواند مبتنی بر روشهای زیر باشد.

- کنترل رویداد با قاعده

یک رویداد به معنای تغییر یک ثبات و یا یک نت است. در روش رفتاری از علامت @ برای کنترل رویداد استفاده می شود. در این حالت دستورات می توانند با تغییر در لبه یک سیگنال چه بالارونده و چه پایین رونده اجرا شوند. به نمونه های زیر دقت کنید.

```
@(clock) q = a ; // با هر تغییر در سیگنال کلاک دستور اجرا می شود
@(posedge clock) q = a ; // در لبه بالا رونده کلاک دستور اجرا می شود
@(negedge clock) q = a ; // در لبه پایین رونده دستور اجرا می شود
Q = @(posedge clock) a ; //
```

• کنترل چند رویداد

گاهی اوقات چند سیگنال داریم که تغییر در یکی از آنها سبب تریگر شدن و اجرای یک مجموعه از دستورات می شود. لیست این رویدادها و یا سیگنالها را لیست حساسیت می گوئیم.

مثال ۲ : دیکودر ۲ به ۴

این مثال مدل رفتاری مثال ۷ است

```
module decode2to4( in , out , enable );
    input [1:0] in ;
    input enable;
    output [3:0] out ;
    reg [3:0] out ;
    always @(enable , in )
    begin
        out = 0 ;
        if ( enable )
            case ( in )
                2'b00 : out = 4'b0001;
                2'b01 : out = 4'b0010;
                2'b10 : out = 4'b0100;
                2'b11 : out = 4'b1000;
            endcase
        end
    end
endmodule
```

عملکرد گزاره های شرطی case و if.else مشابه زبان c است. اگر داخل بلوک always چند گزاره انتساب وجود داشته باشد تعیین ابتدا و انتهای بلوک با کلید واژه های begin و end اجباری است.

مثال ۳ : جمع کننده و تفریق کننده چهار بیتی بدون carry

```
module      addsub (input  [3:0] a , b ,
inputsel ,
output reg [3:0] out );
    always @( a , b , sel )
        if ( sel == 1'b0 )
            out = a + b;
        else
            out = a - b ;
endmodule
```

آزمایش ۷-۱: یک مبدل باینری چهاربیتی به کد سون سگمنت طراحی کنید تا بتوانیم اعداد 0 تا F را روی سون سگمنت نمایش دهیم.

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 و negedge بیان می شود.

مثال ۴ : فلیپ فلاپ نوع D حساس به لبه بالارونده با Reset سنکرون

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

در این مثال واکنش خروجی به ورودی ریست ، وابسته به کلاک است.

آزمایش ۲ : فلیپ فلاپ نوع D حساس به لبه بالارونده با Reset آسنکرون

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

در این مثال گزاره های داخل بلوک هم با تغییر مقدار ریست و هم با تغییر کلاک اجرا می شود

آزمایش ۳ : شمارنده چهار بیتی بالا شمار حساس به لبه پایین رونده با Reset آسنکرون

```
module    bincounter (inputclk , reset , outputreg [3:0] q ) ;  
    always @ (posedgeclk , posedge reset)  
        if ( reset )  
            q = 4'h0;  
        else  
            q = q +1 ;  
endmodule
```

تمرین : یک شمارنده چهاربیتی دهمی (BCD) طراحی کنید و خروجی را توسط شبیه سازی نمایش دهید . این مدار دارای پایه ورودی Reset آسنکرون است.

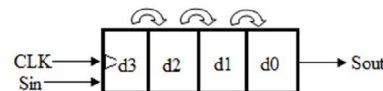
آزمایش هشتم

آزمایش ۸-۱: با استفاده از شمارندهٔ باینری چهار بیتی و مبدل باینری به سون سگمنت آزمایش قبل مداری طراحی کنید که اعداد 0 تا F را بصورت پیوسته روی سون سگمنت برد آزمایشگاه نمایش دهد .
(برای ترکیب این دو مدار از روش سطح گیت استفاده کنید)

مثال ۱: ثبات چهار بیتی با ورودی و خروجی سریال با قابلیت شیفت به راست

```
module shiftreg ( inputclk , sin , outputsout );
    reg [3:0]temp ;
    always @(posedgeclk )
        begin
            temp <= temp >> 1 ;
            temp[3] <= sin ;

        end
    assign sout = temp[0]
endmodule
```



شکل ۱

در این برنامه در گزاره های داخل بلوک always بر خلاف برنامه های قبل از عملگر انتساب <= استفاده شده است. در ورایلاگ 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.

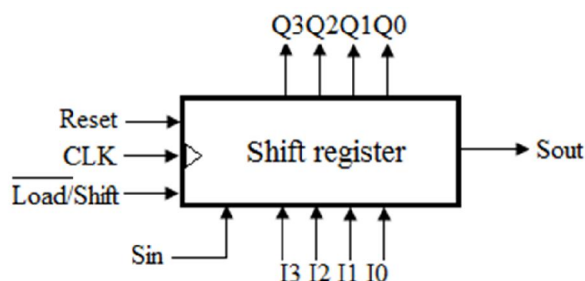
```
begin
    a <= b + c ;    \ if b=3 and c=2 and a(t-1)=1 => a=5
    d <= a ;        \ => d=1
end
```

در این بلوک از عملگر انتساب غیربلوکی (<=) استفاده شده است . در این حالت ابتدا مقادیر سمت راست همه 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 تاثیر در عملکرد مدار نخواهد داشت.

نکته دیگری که در برنامه 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 کلیدواژه assign صورت گیرد. باید توجه داشته باشید که بلوک always و بلوک assign بصورت همزمان و موازی اجرا می شوند.

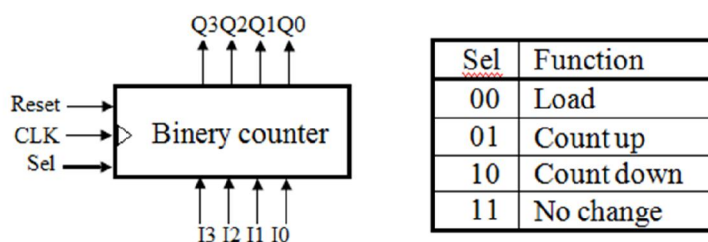
سیگنال temp یک سیگنال داخلی و از دید کاربر مخفی است پس در تعریف ماژول و اعلان ورودیها و خروجیها آورده نمی شود.

آزمایش ۸-۲: یک شیفت رجیستر مانند شکل زیر با قابلیت بارگذاری موازی سنکرون و شیفت به راست و همچنین ورودی Reset آسنکرون طراحی نمایید.



شکل ۲

آزمایش ۸-۳: یک شمارنده چهار بیتی باینری با خط انتخاب شمارش صعودی و نزولی ، دارای ورودی چهار بیتی و خط فعال ساز بارگذاری سنکرون و همچنین خط reset آسنکرون طراحی نمایید (شکل زیر). خروجی آنرا در محیط شبیه سازی در حالت‌های متفاوت مشاهده کنید .



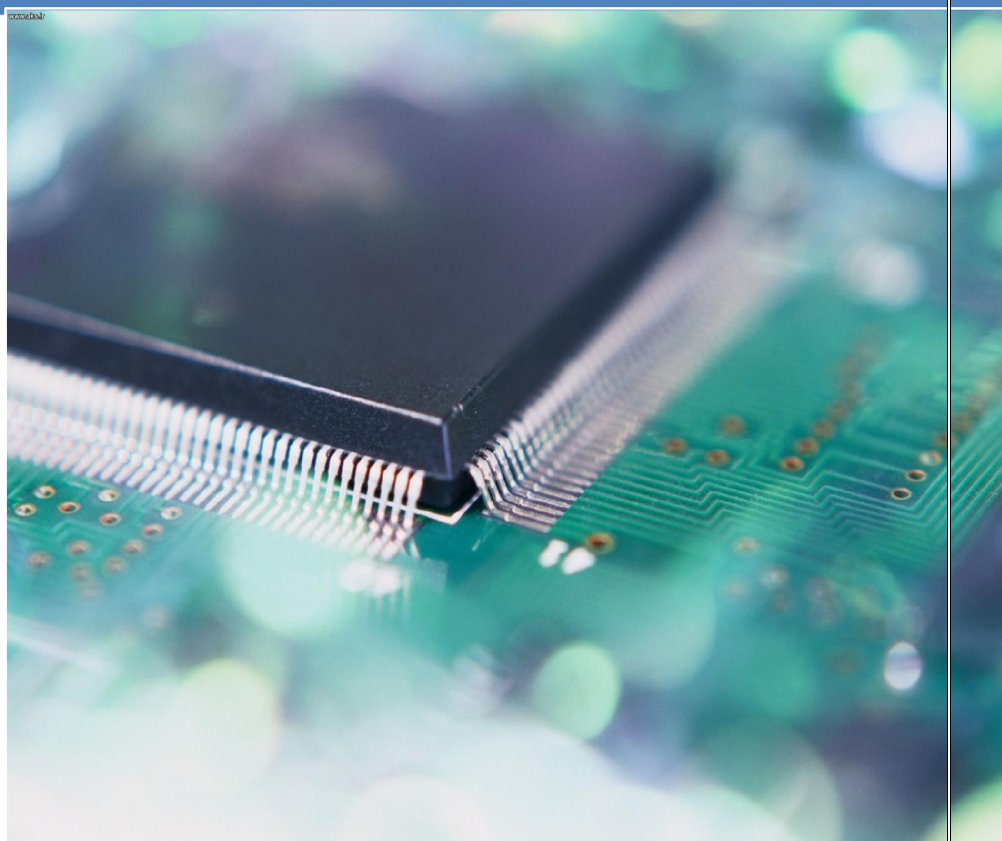
شکل ۳

تمرین: برنامه آزمایش ۸-۱ را بدون استفاده از روش سطح گیت دوباره نویسی کرده و درستی عملکرد آنرا در محیط شبیه سازی نمایش دهید.

بسمه تعالی



راهنمای نرم افزار QUARTUS



دانشکده برق و کامپیوتر
آزمایشگاه مدار منطقی
محمد رضا فتاح

راهنمای نرم افزار

آشنایی با نرم افزار کوارتوس (Quartus)

مقدمه

شرکت ALTRA که در زمینه 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 و شبیه سازی و سپس پیاده سازی آن را با استفاده از این نرم افزار توضیح خواهیم داد.

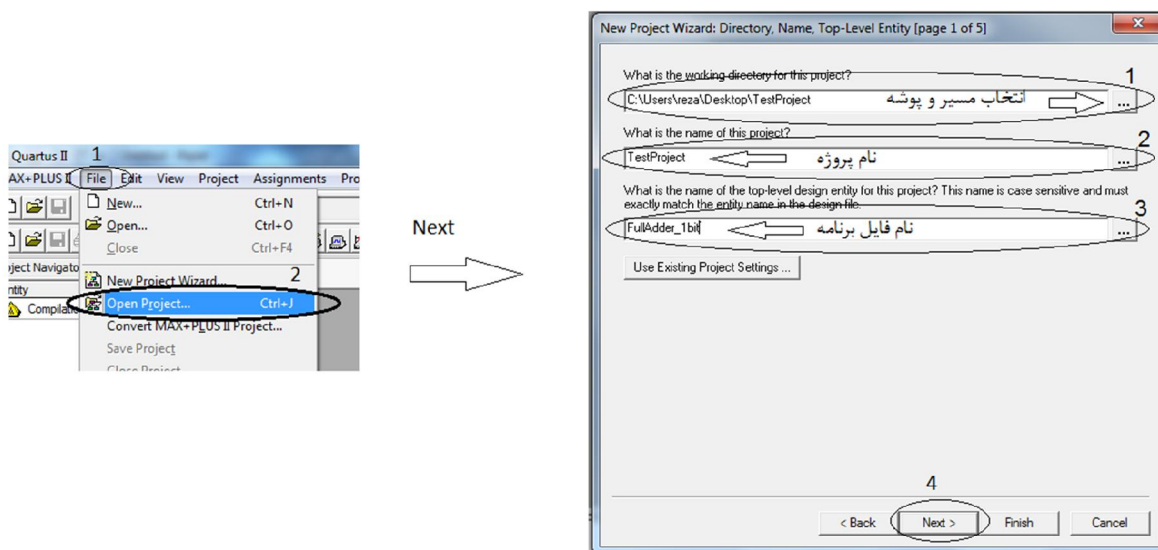
۱- ایجاد پروژه

در این نرم افزار هر پروژه شامل چندین فایل از جمله فایل اصلی برنامه و فایل های 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 کننده یک بیتی است بنابراین نام فایل را FullAdder_1bit انتخاب کنید که مربوط به آزمایش اول دستور کار (بخش دوم) است .

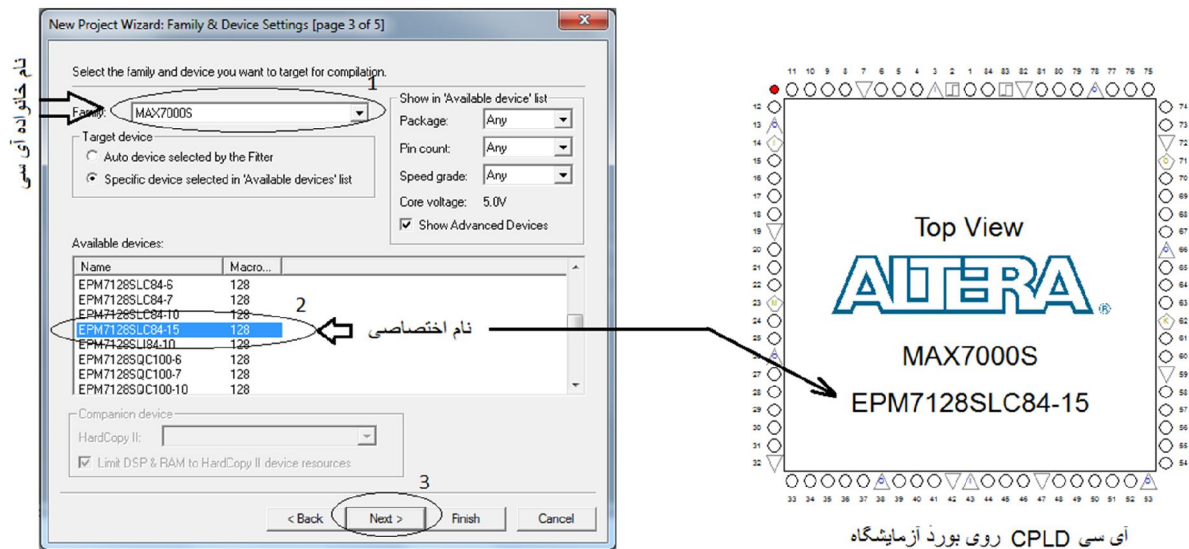
برای ایجاد پروژه مراحل زیر را دنبال کنید:

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



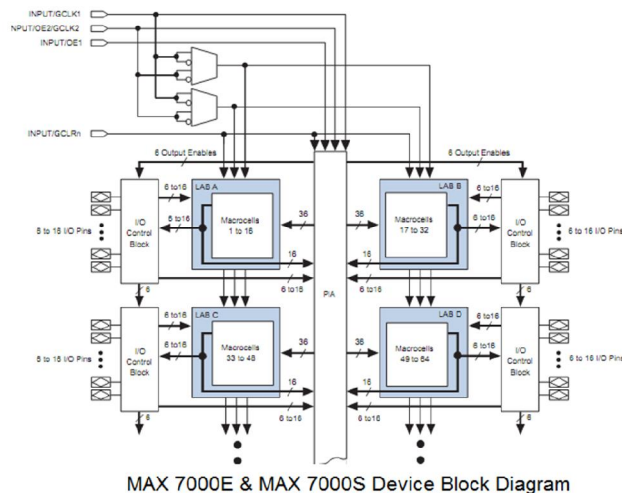
توجه داشته باشید در این مرحله فقط پروژه معرفی شده ایجاد می شود ولی فایل مربوطه با نام داده شده در مرحله ۳ ، بعداً باید با همین نام ساخته شود. نوع و فرمت این فایل (متنی یا شماتیک) در هنگام ساخت مشخص می شود

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 پنجره زیر (سمت چپ) ظاهر شود.

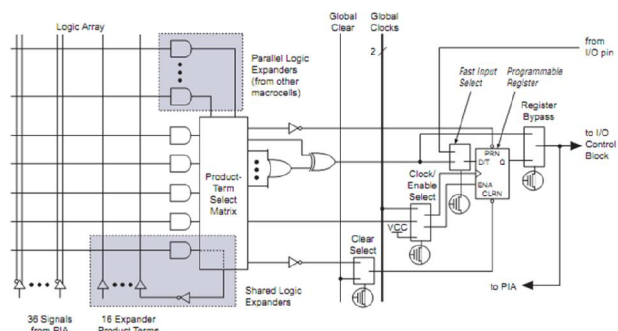


آی سی CPLD روی برد آزمایشگاه

در این مرحله آی سی برنامه پذیری که قصد داریم طرح نهایی مدار خود را روی آن پیاده سازی کنیم انتخاب میکنیم. برد آموزشی آزمایشگاه مدارهای منطقی که آزمایشهای خود را با آن انجام میدهم 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 ، نشانه این است که حافظه برنامه پذیر این آی سی از نوع EPROM است . حروف M7xxxS نشان دهنده خانواده این آی سی یعنی MAX7000S و عدد 128 نشانگر ظرفیت مدارات داخلی این المان بر حسب MacroCell است (۱۲۸ ماکروسل). عدد 84 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

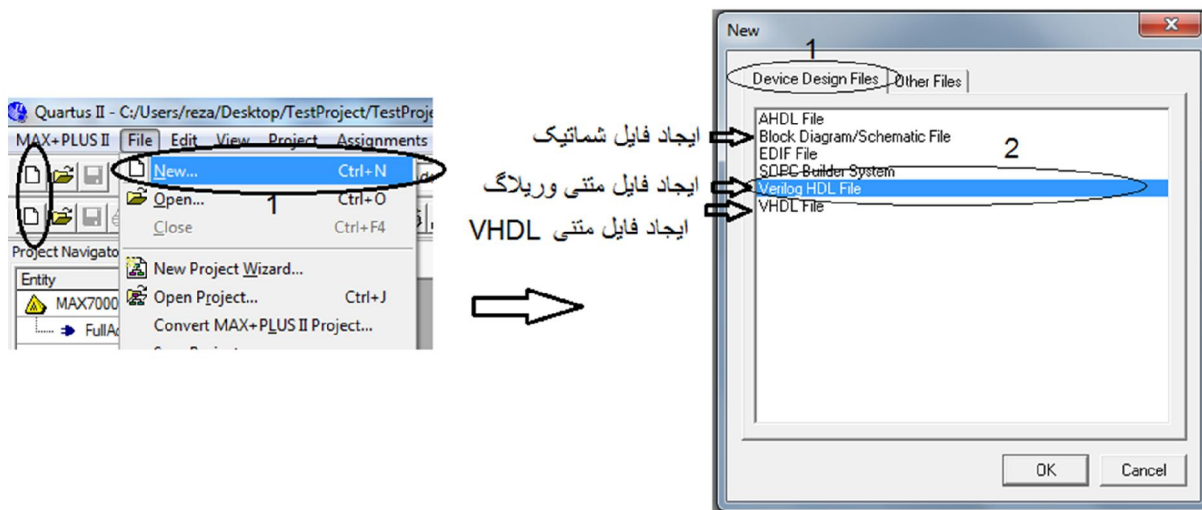


MAX 7000E & MAX 7000S Device Macrocell

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

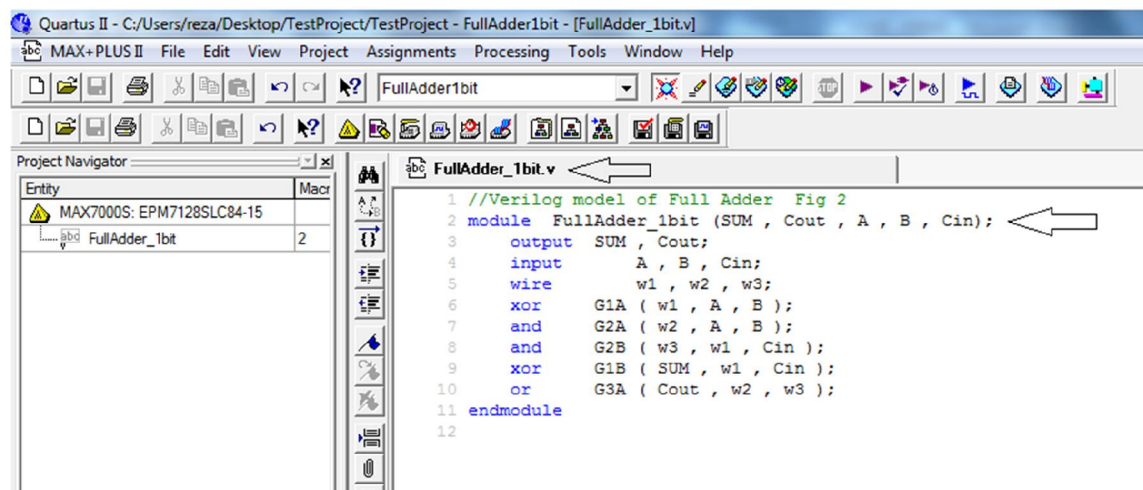
۲- ایجاد فایل و کامپایل برنامه

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

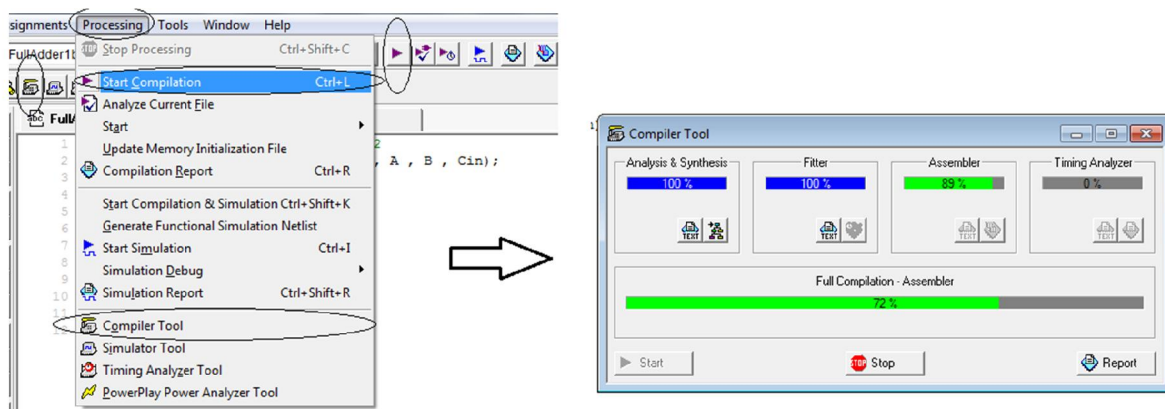


اکنون فایل متنی وریلاگ ایجاد شد با انتخاب گزینه Save از منوی File فایل ایجاد شده را با نامی که قبلاً در مرحله ایجاد پروژه برای فایل TopLevel انتخاب کرده بودید ذخیره نمایید. توجه داشته باشید اگر مراحل بالا را بدرستی انجام داده باشید نام فایل پیش فرض بصورت خودکار همان نام مورد نظر و در مسیر پوشه ای که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

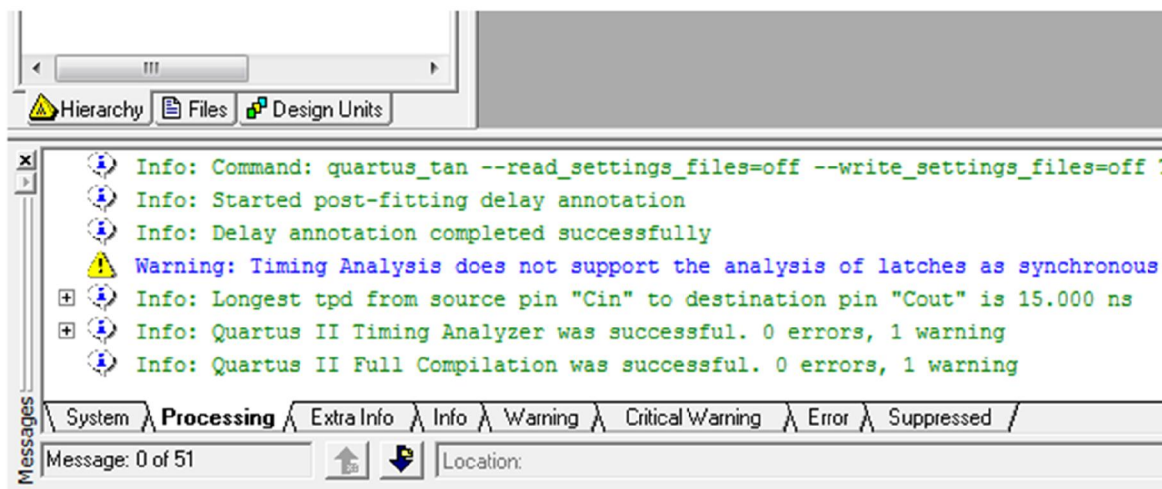
با ساخت فایل متنی اکنون می خواهیم طراحی و شبیه سازی و سنتز مدار جمع کننده کامل را شروع کنیم. ابتدا برنامه مدار را در فایل ایجاد شده با زبان وریلاگ بنویسید (می توانید آنرا از داخل دستور کار کپی کنید).



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



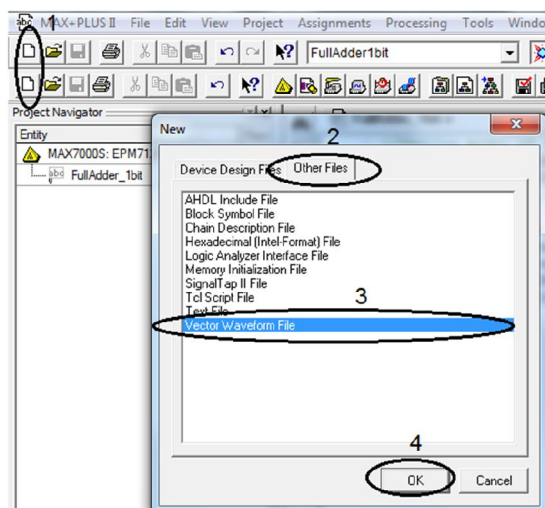
پیامهای خطا و اخطار بصورت شکل زیر در پنجره پیغامها که در زیر پنجره فایل قرار دارد ظاهر خواهد شد.



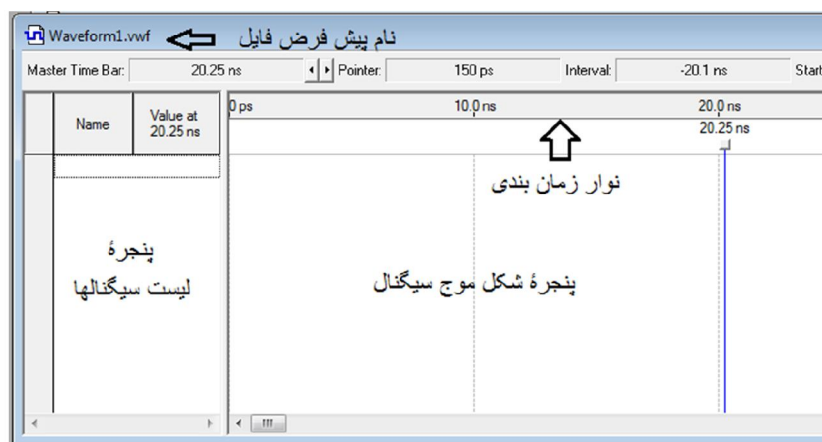
با دوبار کلیک روی پیغام خطای احتمالی محل آن در فایل برنامه مشخص می شود.

۳- شبیه سازی

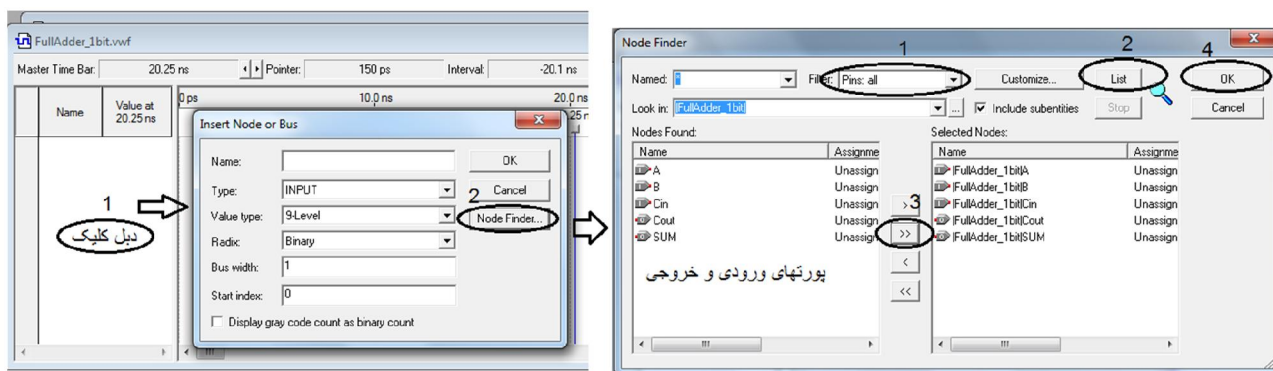
در مرحله بعد با استفاده از ابزار شبیه سازی درستی عملکرد مدار را بررسی میکنیم. برای این منظور از منوی File گزینه New را انتخاب کنید. در پنجره ظاهر شده و از برگه OtherFile گزینه Vector Waveform را گزینش نمایید.



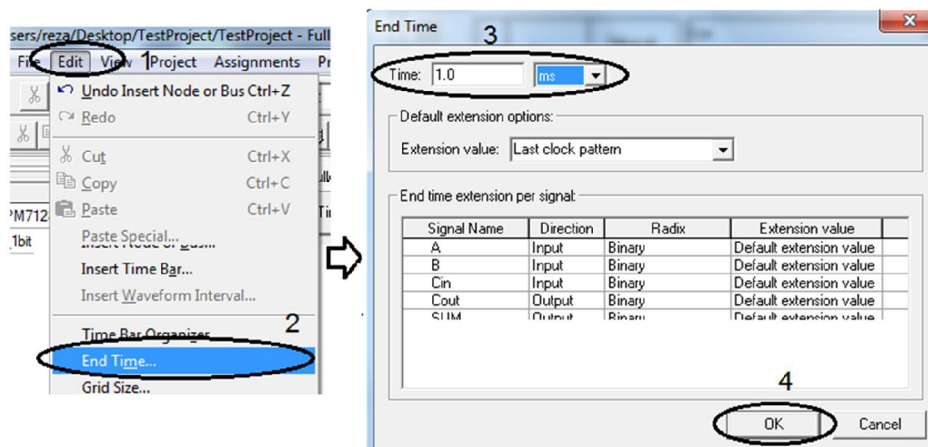
فایل شبیه سازی مانند شکل زیر ایجاد می شود:



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

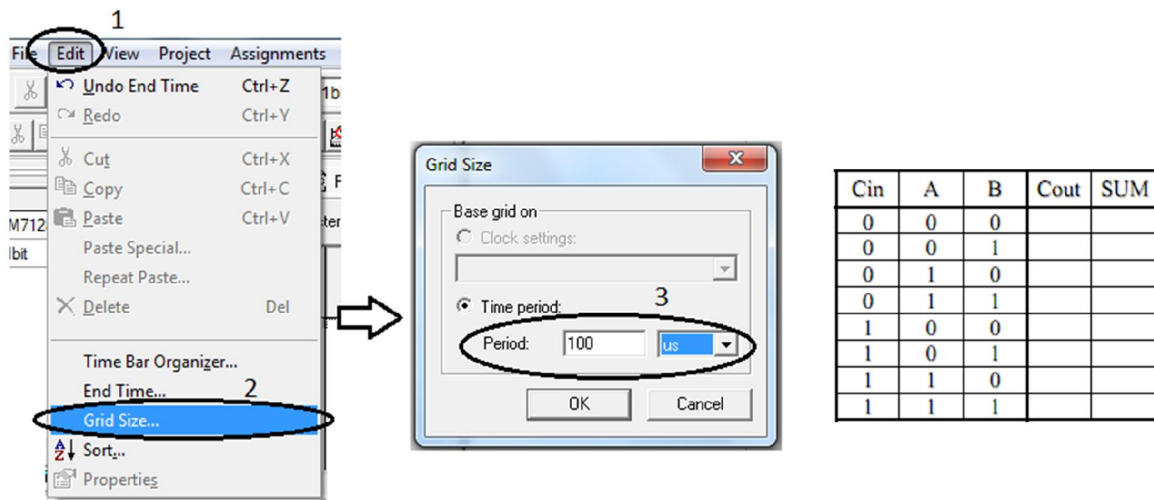


حال سیگنالهای ورودی و خروجی مدار به فایل شبیه سازی اضافه شده است. در مرحله بعد ابتدا بازه زمانی شبیه سازی و همچنین بازه زمانی برای هر مرحله و حالت را باید مشخص نماییم . برای بازه زمان کل از شکل زیر و برای زمانی هر حالت از شکل صفحه بعد استفاده کنید .



انتخاب بازه زمانی شبیه سازی

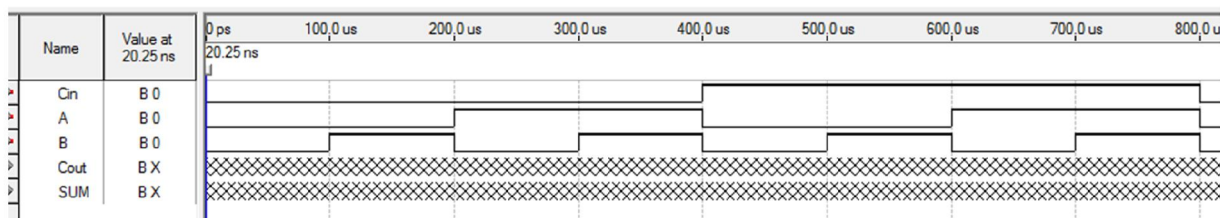
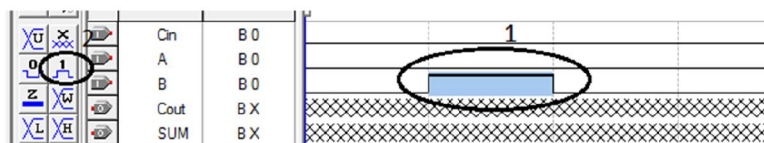
در تمام آزمایشها زمان شبیه سازی را یک میلی ثانیه (1ms) انتخاب کنید.



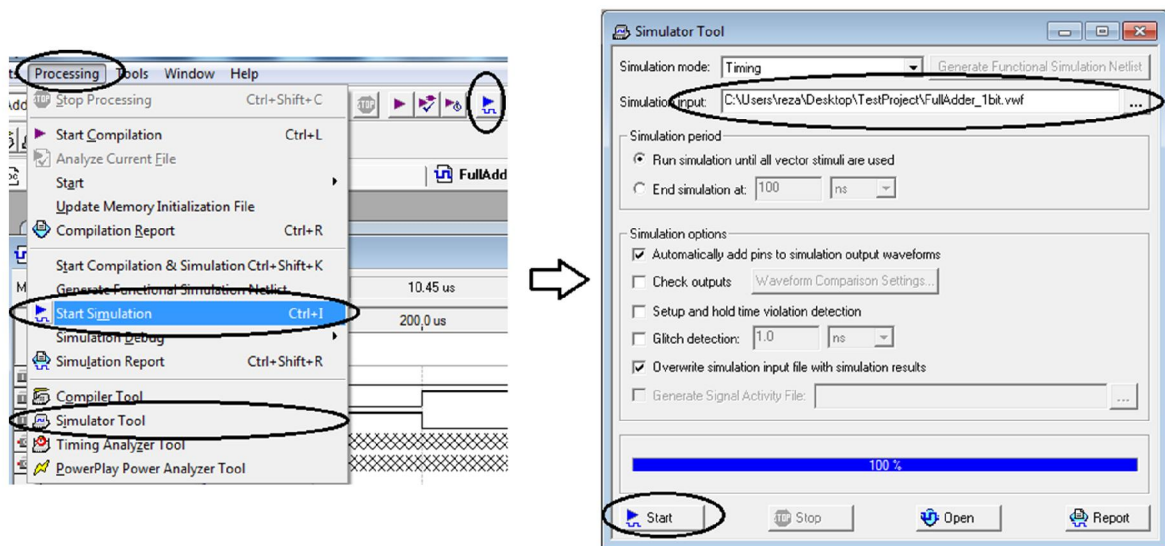
انتخاب بازه زمانی برای هر حالت

جدول حالات مدار جمع کننده

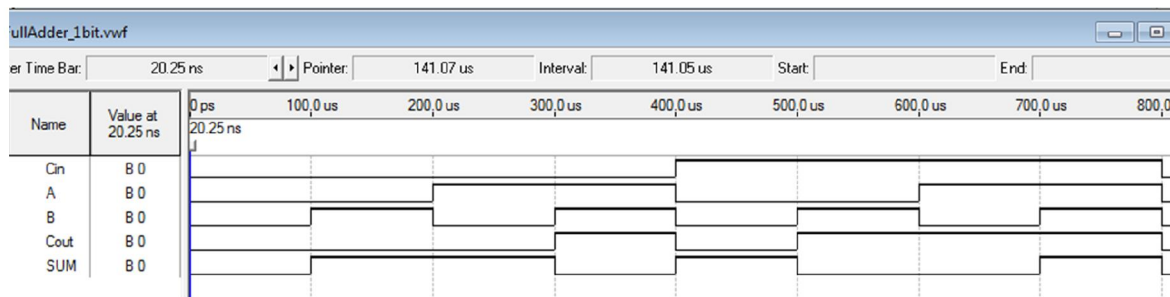
برای انتخاب بازه زمانی هر حالت به تعداد حالات موجود در جدول درستی مدار نگاه می کنیم و بازه زمانی کل را به تعداد حالات تقسیم می کنیم. تعداد حالات جدول عدد هشت است که برای جلوگیری از حاصل تقسیم با مقدار اعشاری آنرا ۱۰ حالت فرض می کنیم. پس زمان هر بازه برای این مدار ۱۰۰ میکرو ثانیه (100us) خواهد بود. برای قرار گرفتن همه حالات در پنجره نمایش سیگنال، از کلیدهای Ctrl+W استفاده نمایید. اکنون زمان آن رسیده است که مقادیر ورودی را برای همه حالات 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 اعمال 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 بازه زمانی بصورت رنگی در می آید اکنون با انتخاب کلید  از نوار ابزار سمت چپ صفحه، مقدار سیگنال به مقدار یک تغییر میکند. برای بقیه حالات و سیگنالهای ورودی این عمل را تکرار کنید.



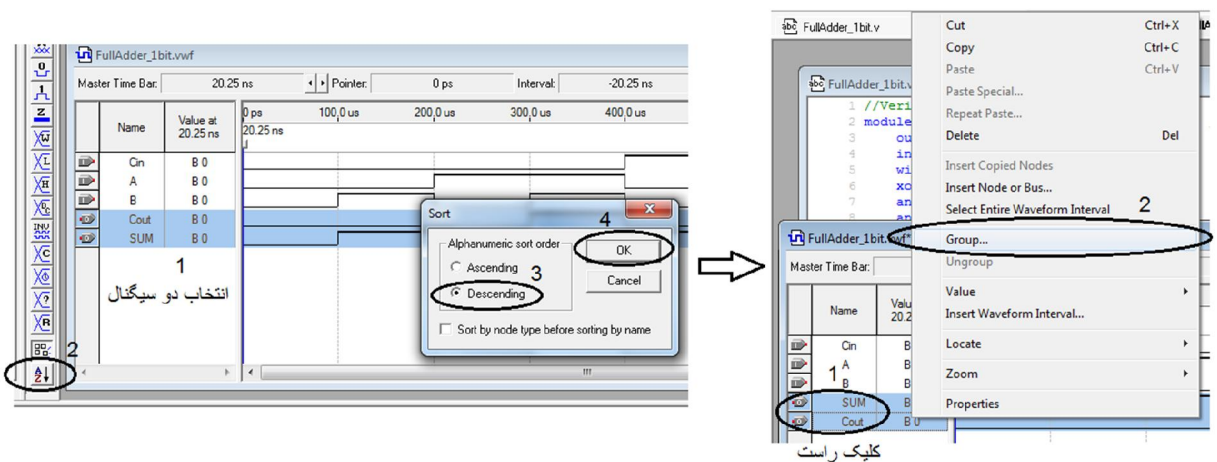
بعد از اینکه مقادیر ورودی را مشخص کردید فایل را ذخیره نمایید. نام پیش فرض پیشنهادی که همان نام فایل اصلی ولی با پسوند متفاوت است را تایید نمایید. برای آغاز شبیه سازی مراحل نشان داده شده در شکل بعد را اجرا کنید.



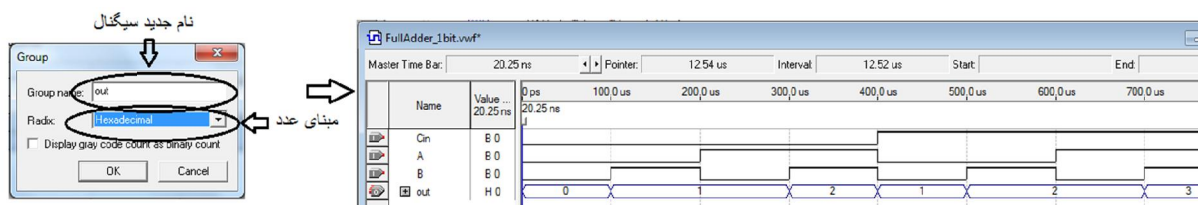
نتیجه بصورت زیر خواهد بود:



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

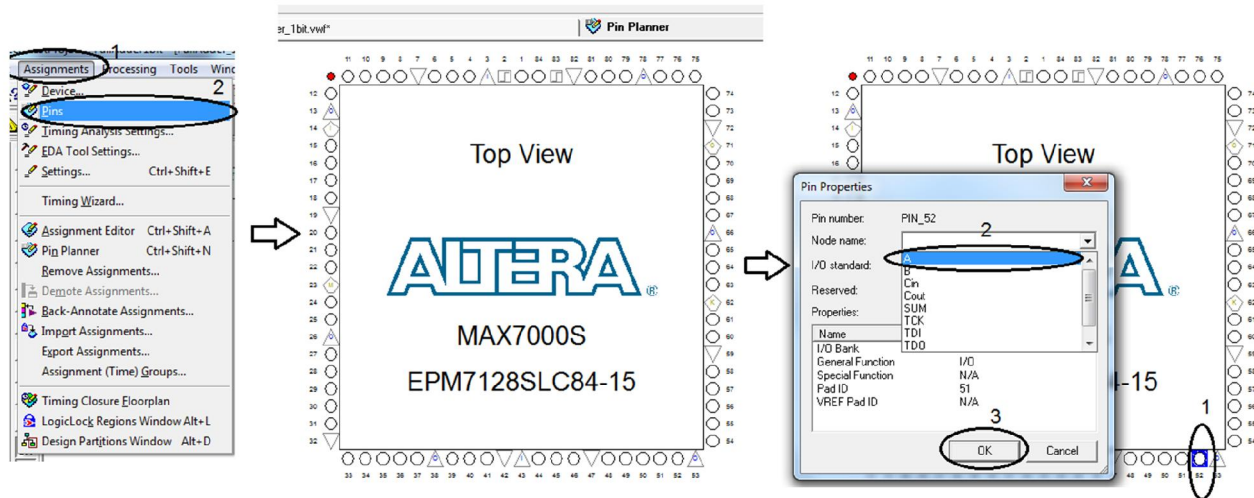


و نتیجه :



۴- پیاده سازی مدار

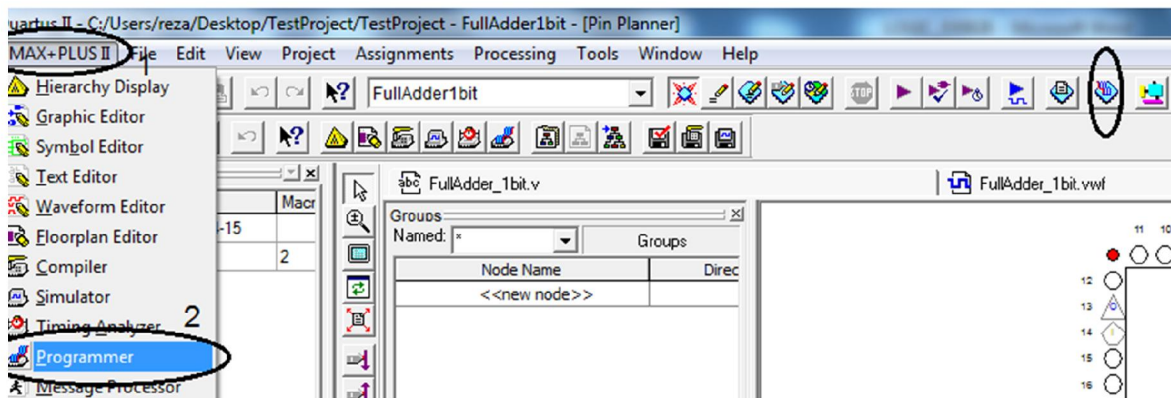
با توجه به اطمینان از درستی عملکرد مدار اکنون می خواهیم طرح خود را روی آی سی برنامه پذیر پیاده سازی و عملکرد مدار را بصورت واقعی مشاهده کنیم. برای این منظور ابتدا باید عمل اختصاص دهی پایه های آی سی به ورودی و خروجیهای مدار را انجام دهیم. با توجه به شکل زیر و به ترتیب شماره ها مراحل ذکر شده را انجام دهید.



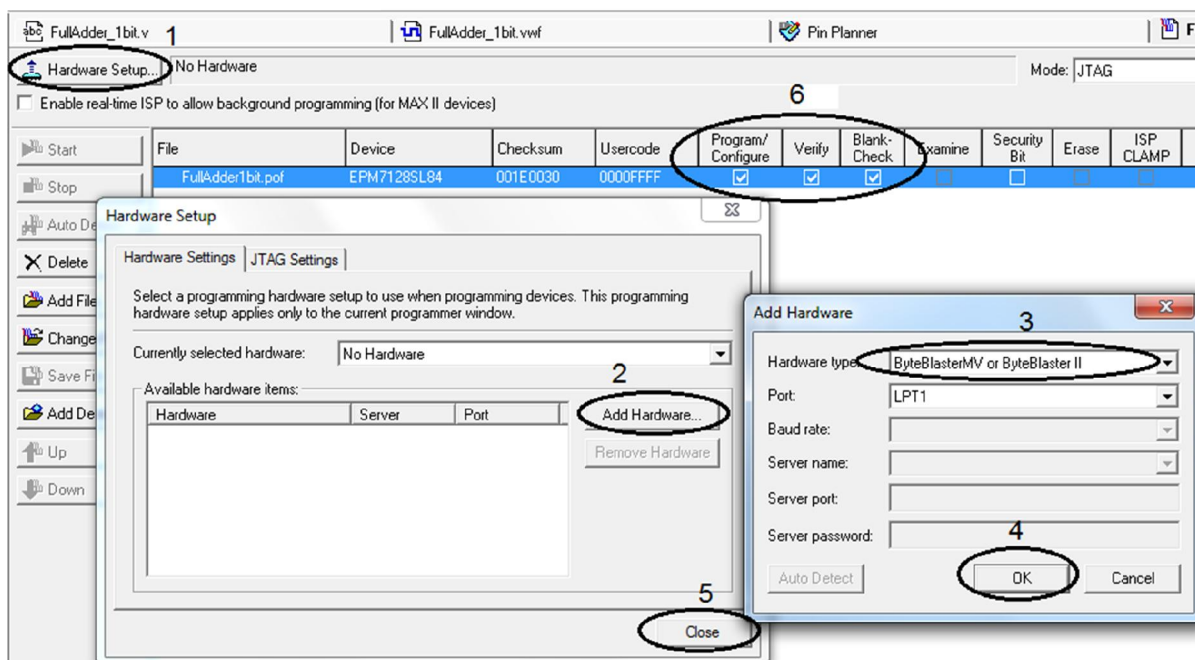
از منوی Assignments گزینه Pins را انتخاب کنید. پنجره ای حاوی آی سی مورد نظر باز می شود. توجه داشته باشید پایه هایی که بصورت دایره هستند می توانند بعنوان پایه های ورودی و یا خروجی تعریف گردند. برای هماهنگی با برد آموزشی آزمایشگاه پایه های لبه پایین آی سی را برای ورودیها و پایه های لبه کنار سمت راست را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

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

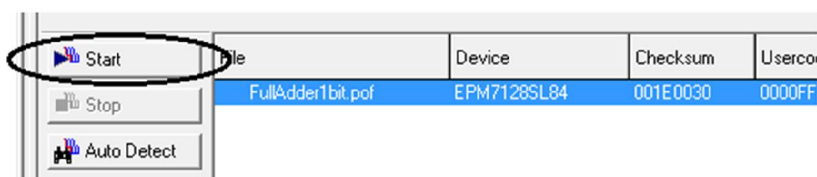
برای کامل شدن عملیات یک بار دیگر عمل کامپایل برنامه را انجام دهید. اکنون می توانید مدار خود را روی آی سی پروگرام نمایید. مانند شکل زیر از منوی Max+Plus گزینه Programmer را انتخاب کنید.



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



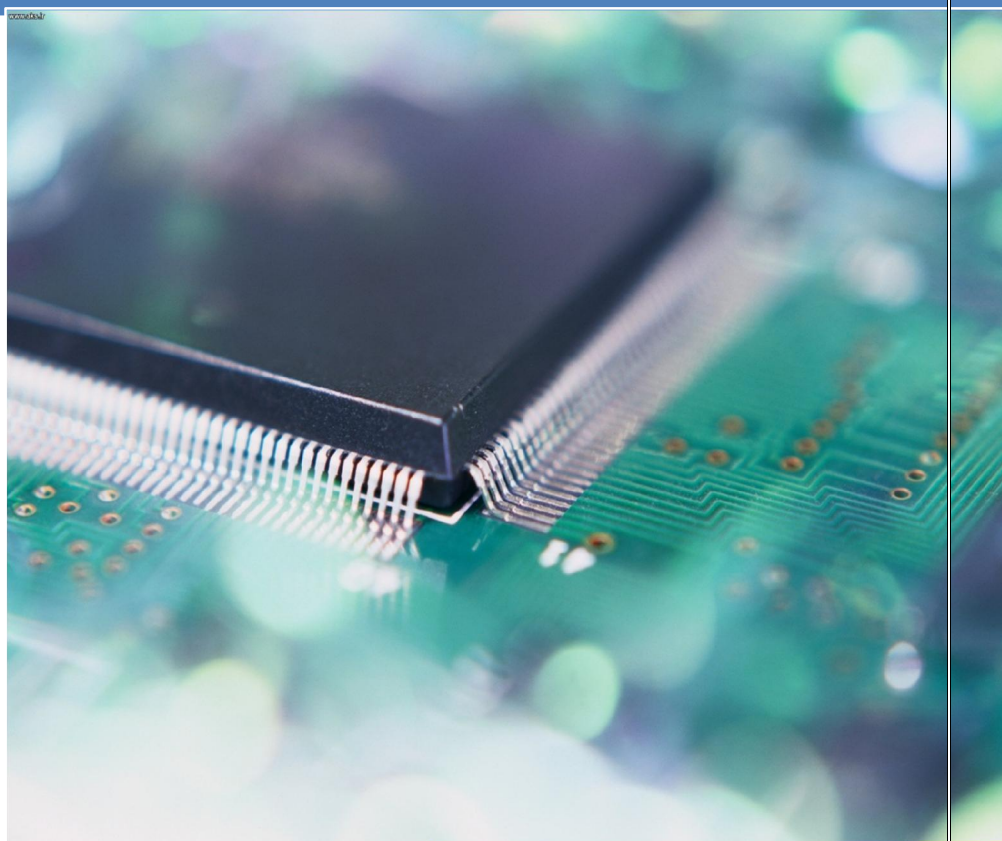
اکنون روی کلید Start کلیک کنید . عملیات برنامه ریزی شروع می شود .



بسمه تعالی



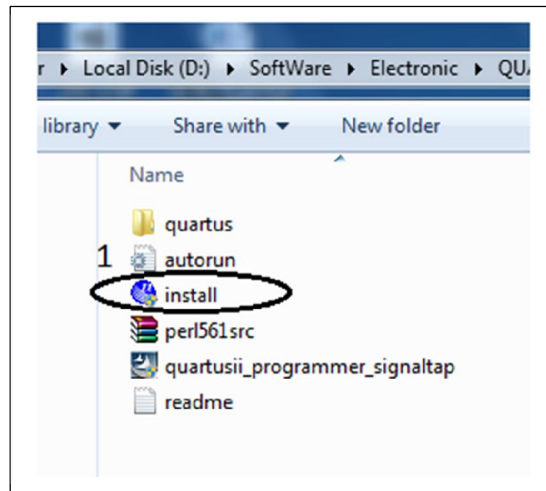
راهنمای نصب نرم افزار QUARTUS



دانشکده برق و کامپیوتر
آزمایشگاه مدار منطقی
محمد رضا فتاح

راهنمای نصب نرم افزار

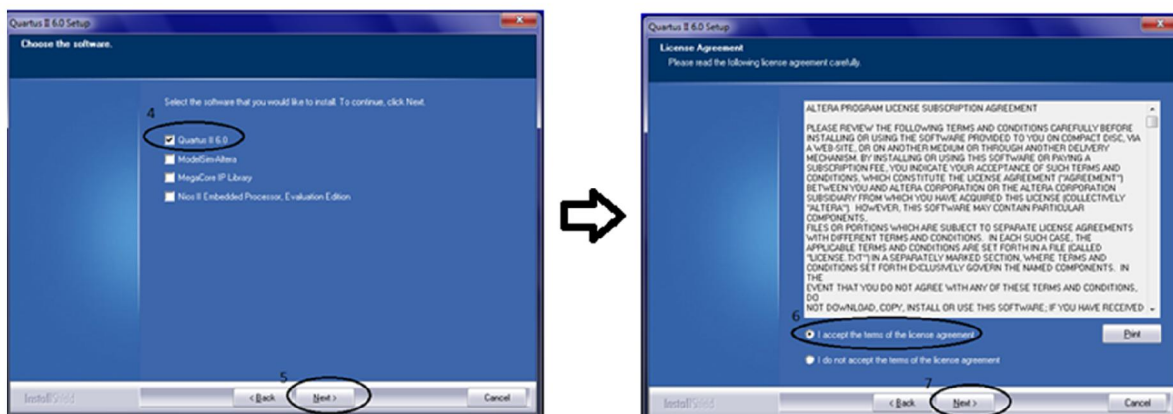
۱: فایل نصب را از داخل پوشه CD1 اجرا کنید و مراحل را مطابق شکل‌های زیر دنبال نمایید:



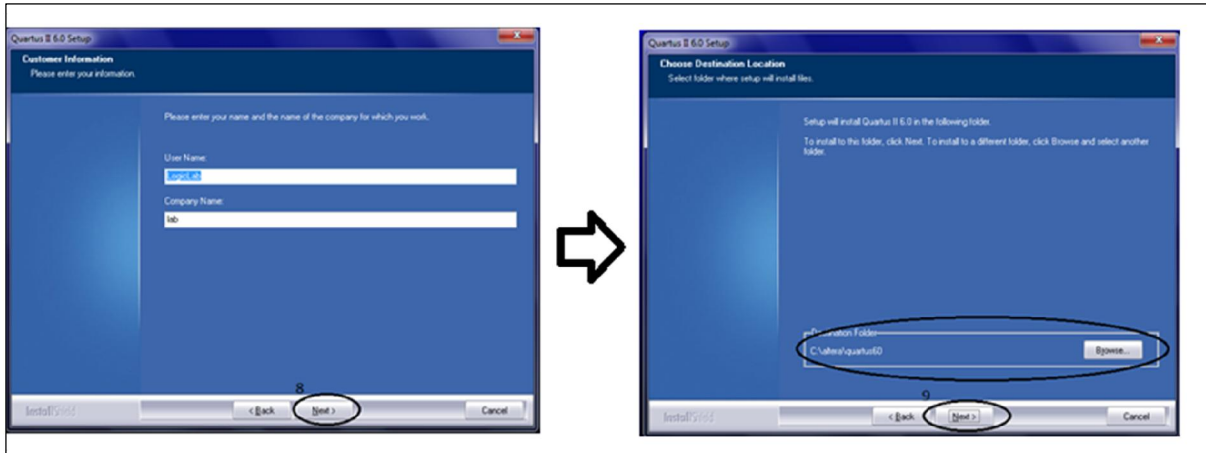
۲:



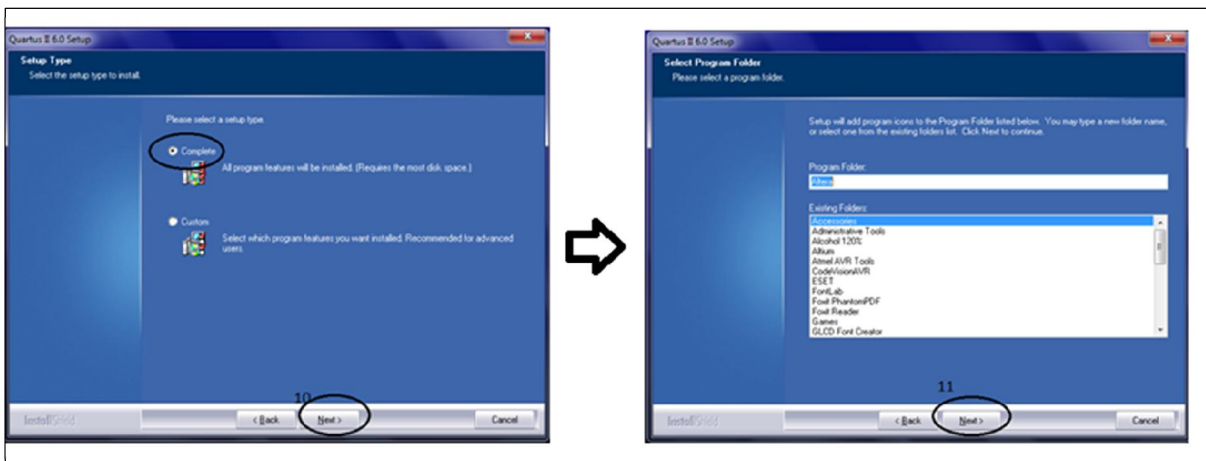
۳:



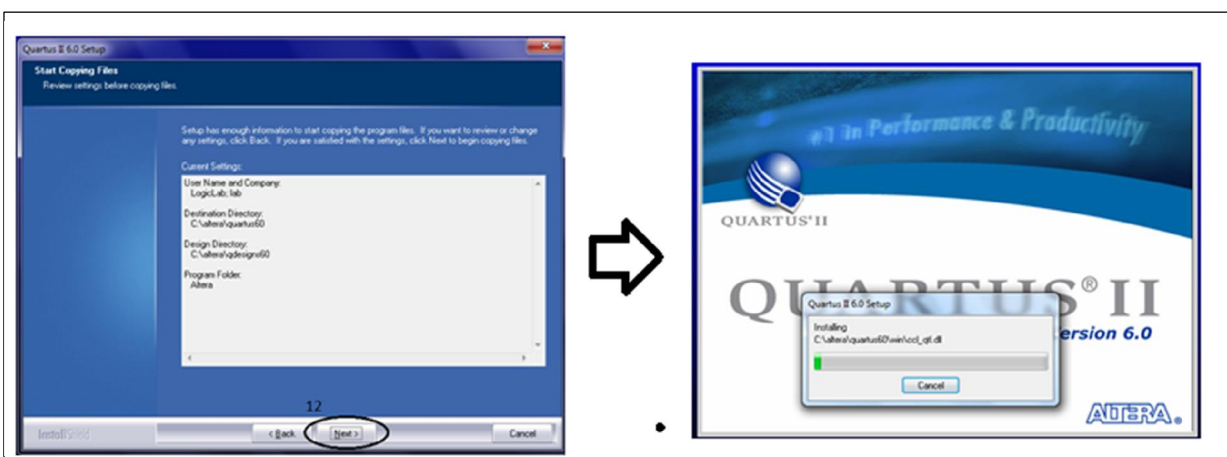
:Y



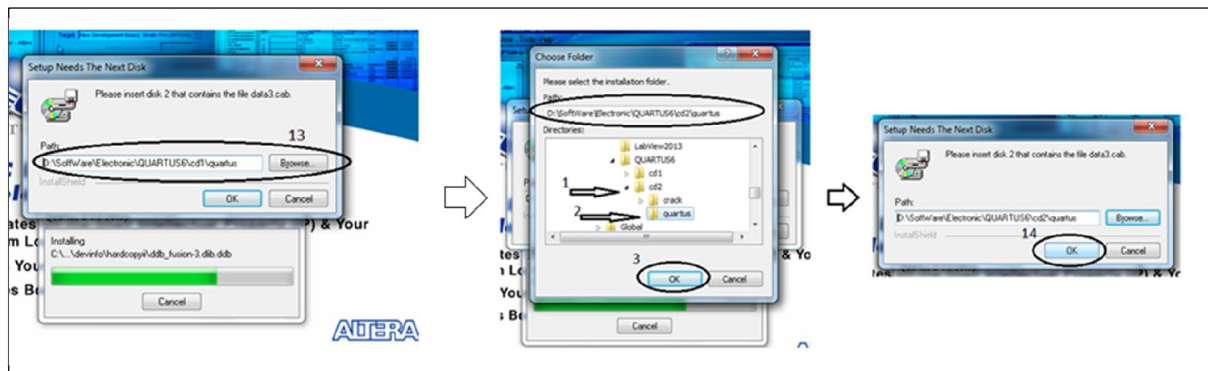
:A



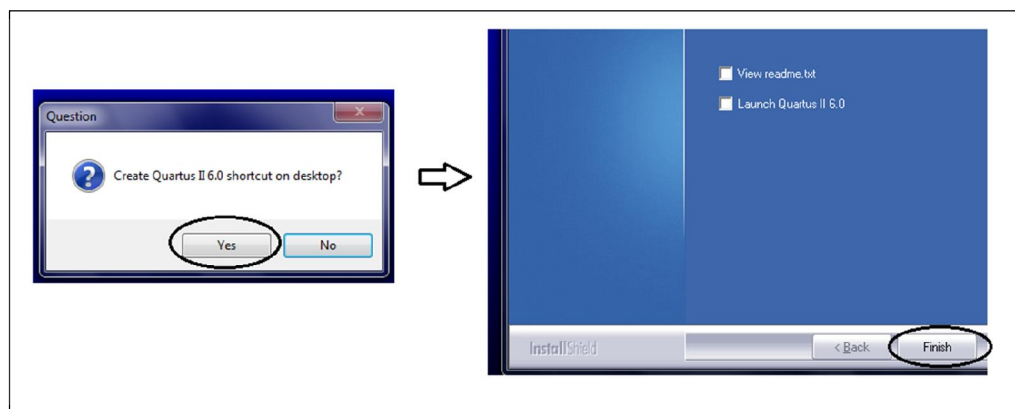
:A



۱۰: انتخاب بخش دوم از داخل CD2

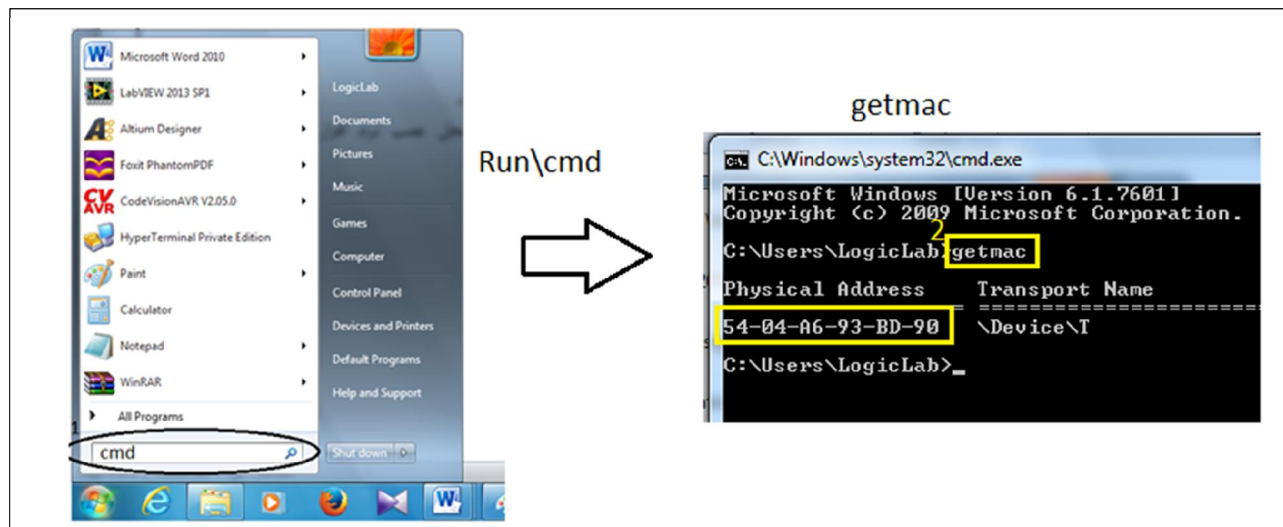


۱۱:

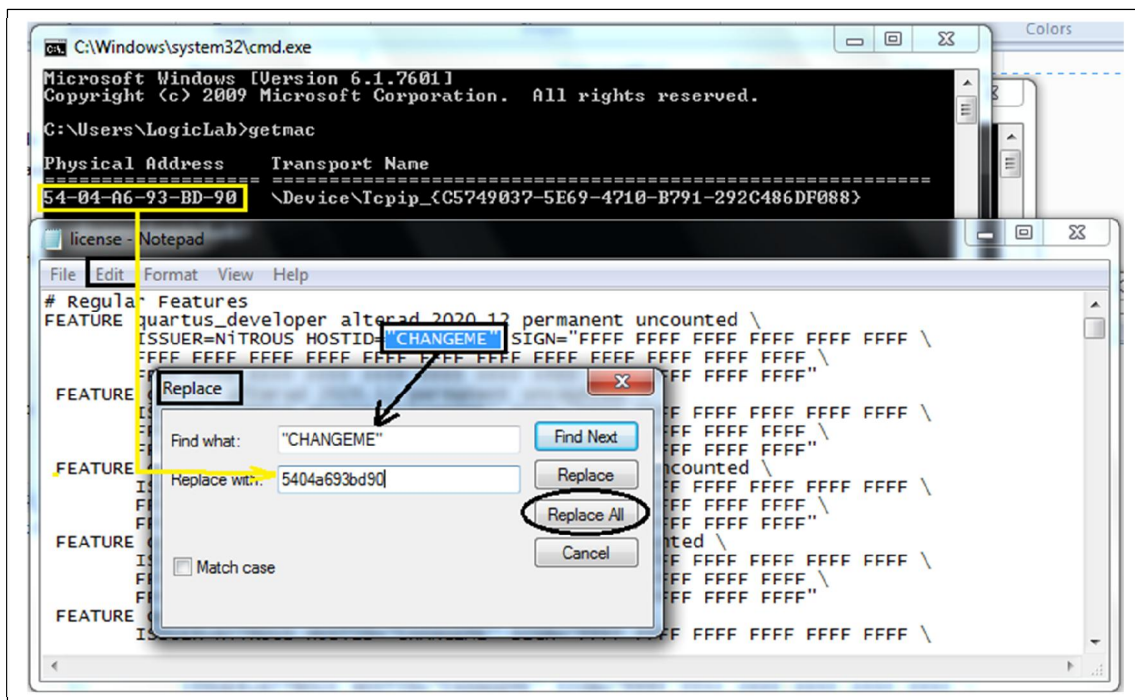


۱۷: راهنمای CRACK

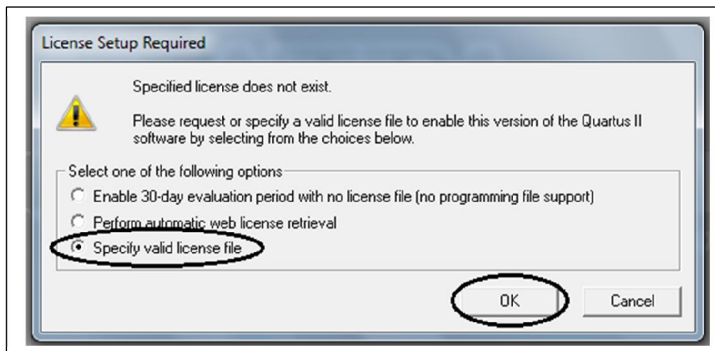
- ۱- ابتدا پوشه CD2\CRACK را در محل نصب نرم افزار کپی کنید.
- ۲- فایل‌های موجود در پوشه CRACK\Windows را در محل نصب نرم افزار و در پوشه های altera\quartus60\bin و altera\quartus60\win کپی کنید.
- ۳- مراحل بعدی را طبق شکل‌های زیر انجام دهید: Run\cmd



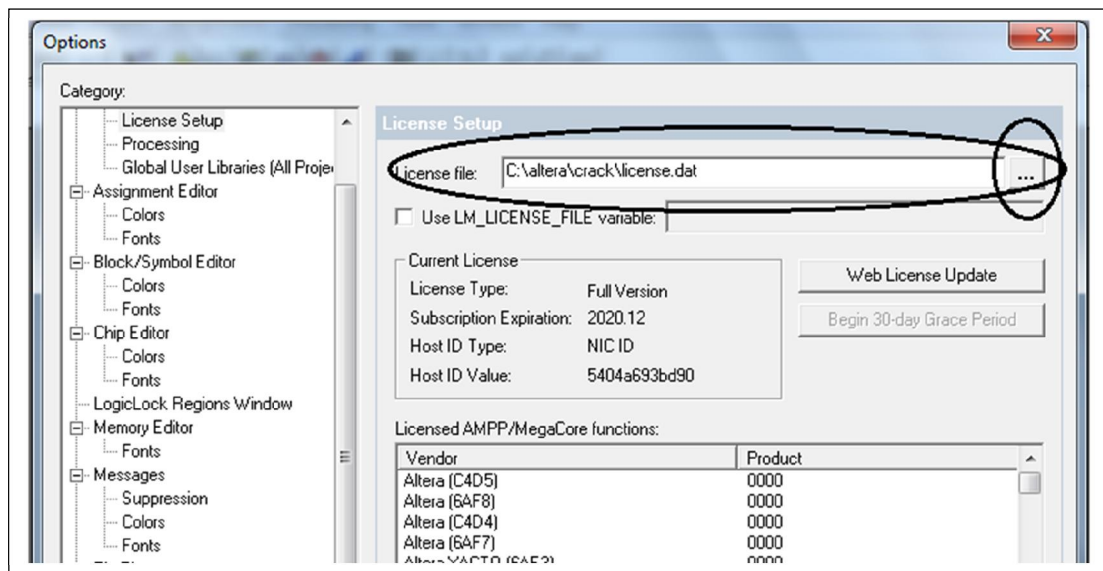
روش دیگری نیز در انتهای این متن برای بدست آوردن آدرس مک آورده شده است .
 ۴- از پوشه CRACK فایل license را با ویرایشگر Notepad باز کنید و سپس طبق شکل زیر از منوی EDIT گزینه Replace را انتخاب کرده و مراحل جایگزینی را انجام دهید. سپس فایل را ذخیره نمایید.



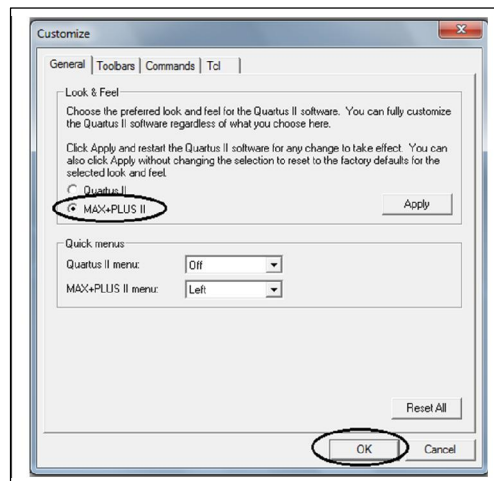
۵- نرم افزار را اجرا کنید و مراحل بعدی را مطابق شکلهای زیر دنبال کنید:



۶- انتخاب فایل license و سپس انتخاب کلید OK.



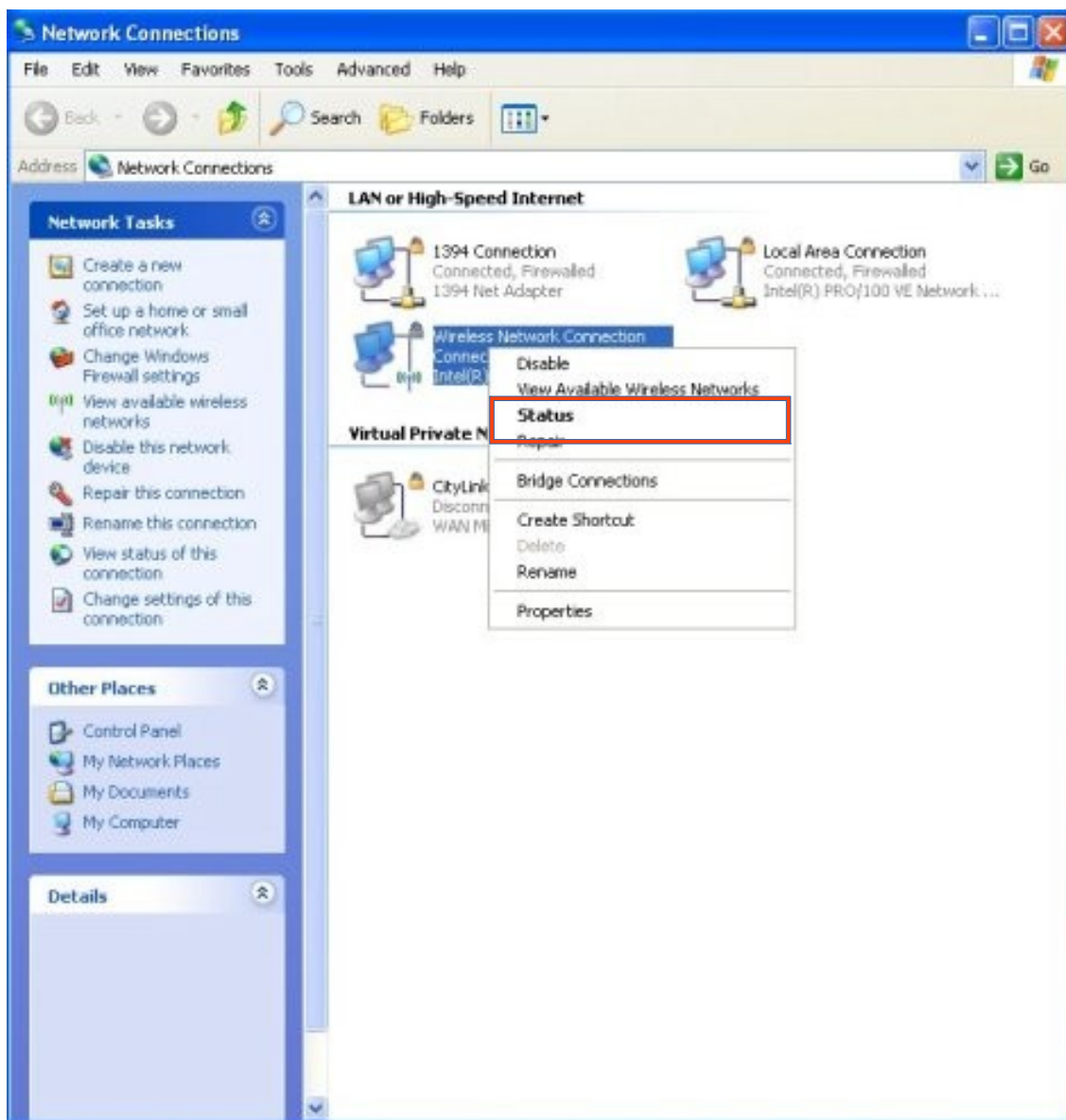
۷-

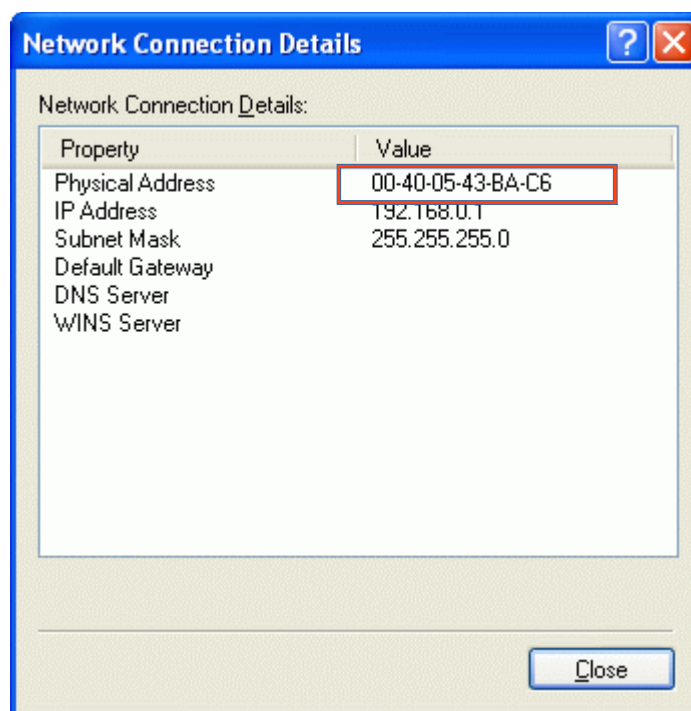
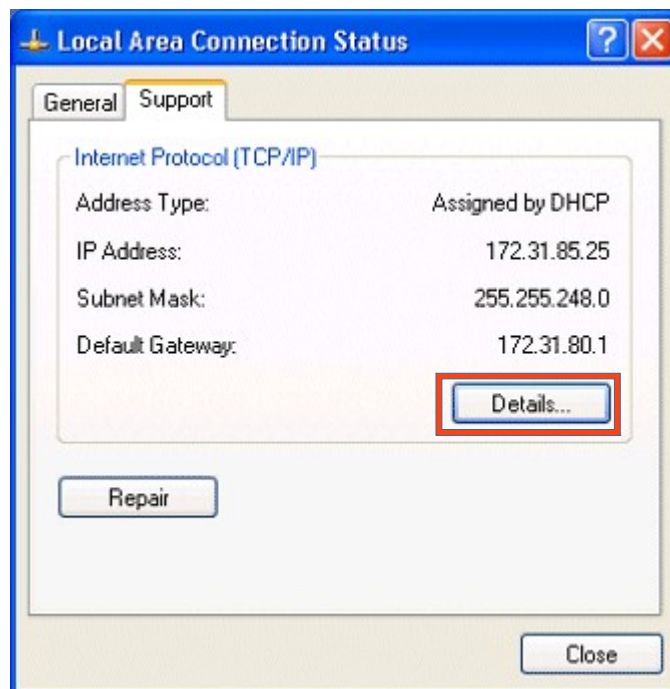


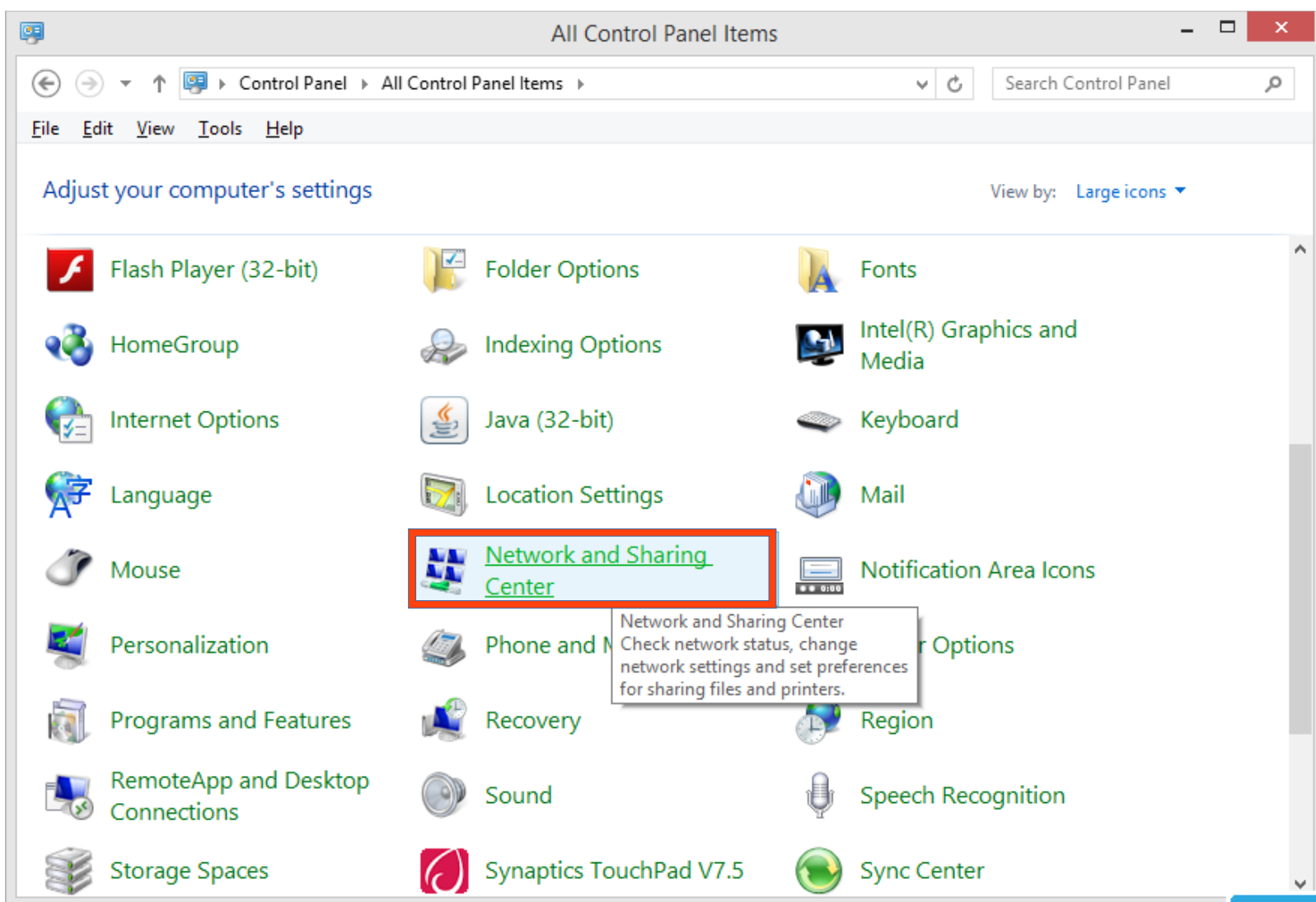
اکنون می توانید از نرم افزار استفاده کنید.

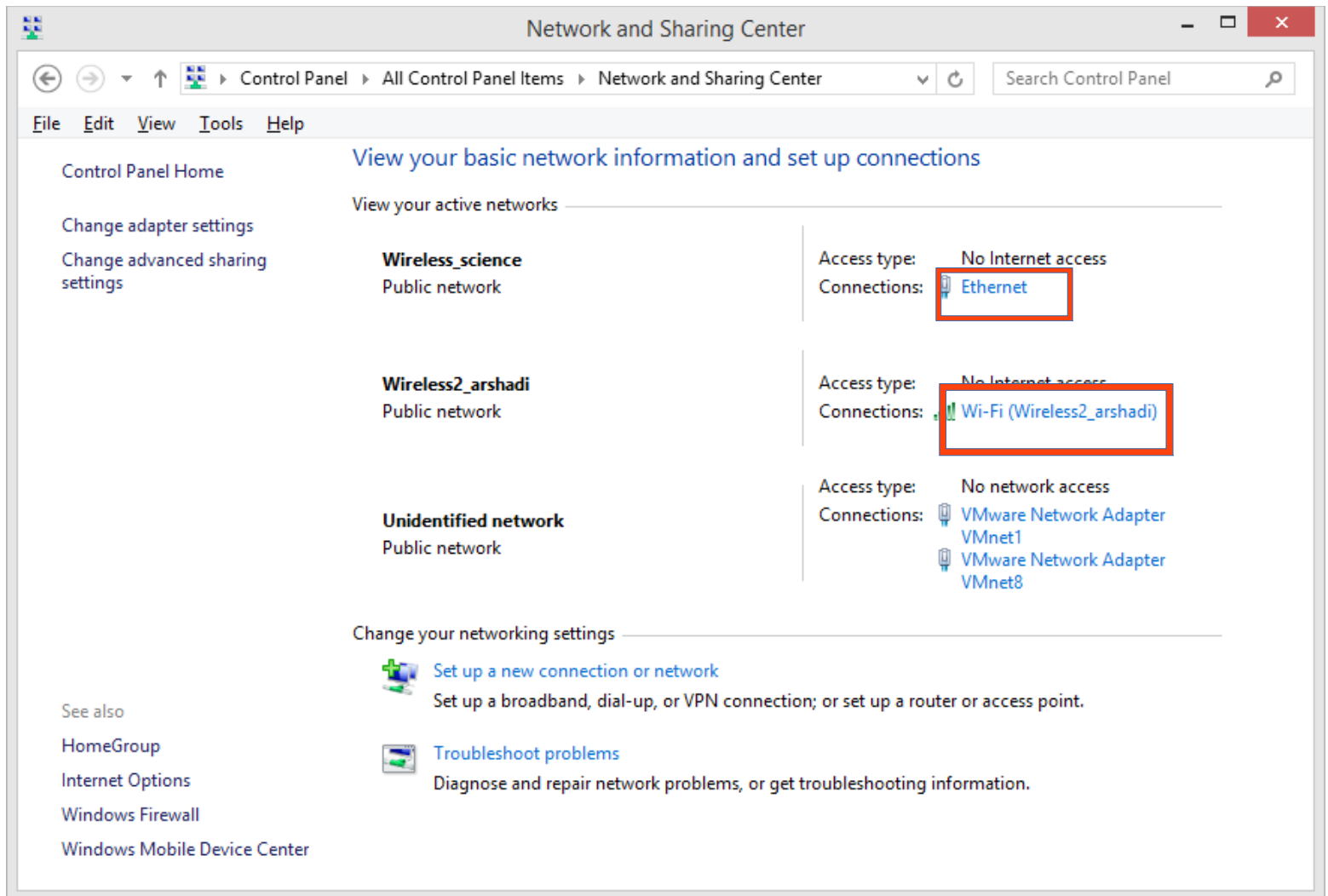
نحوه به دست آوردن آدرس مک (آدرس فیزیکی) سیستم

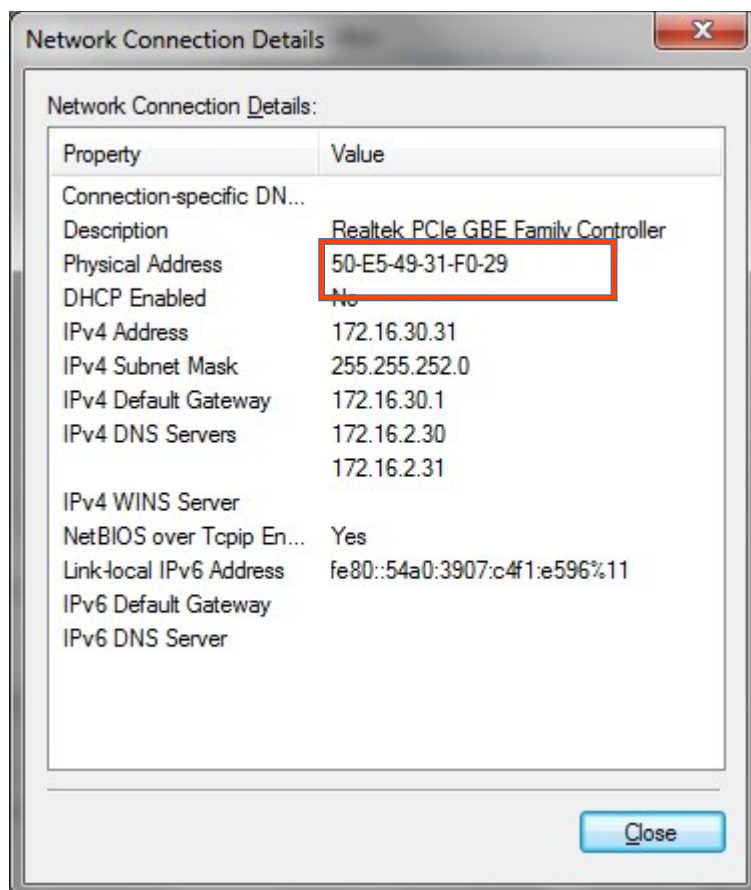
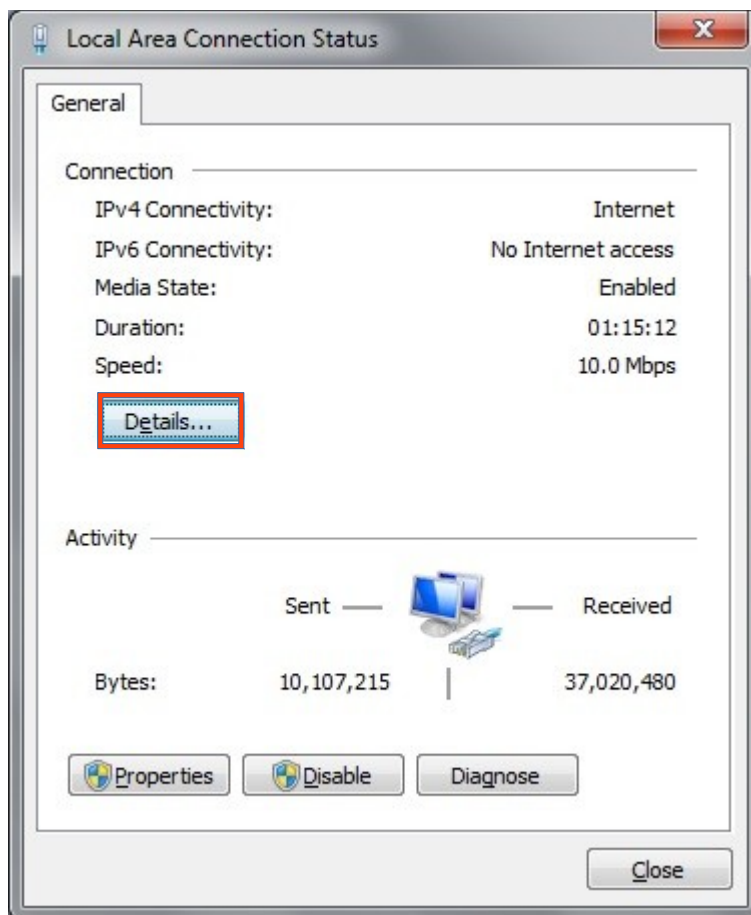
کامپیوترهای شخصی و لپ تاپ ها
برای به دست آوردن مک آدرس در ویندوز کافی است از قسمت اتصالات شبکه (در مورد اتصالات کابلی local area connection و در خصوص اتصالات بی سیم wireless connection) ، اتصال مورد نظر خود را انتخاب و پنجره وضعیت (status) را باز کنید . پس از آن با زدن دکمه جزئیات (details) پنجره جزئیات باز شده و مقدار درج شده در مقابل physical Address همان آدرس مک کارت شبکه شما می باشد. که در شکل نیز مشخص شده است.
در شکل های زیر فرآیند مذکور در به ترتیب در ویندوز XP و ویندوز 8 و 7 مشخص شده اند.
ویندوز XP:











همچنین می‌توانید از دستور getmac در command prompt (cmd) برای به دست آوردن آدرس فیزیکی کلیه کارت های شبکه سیستم استفاده نمایید.
در مورد سایر سیستم عامل ها مانند لینوکس و ios نیز می‌توانید از روش‌های مشابه استفاده کنید. به عنوان نمونه در لینوکس می‌توانید از دستور ifconfig استفاده کرده و اطلاعات مربوط به پارامتر Hwaddr که همان آدرس مک سیستم است را مشاهده نمایید.

تلفن‌های هوشمند و تبلت ها

در خصوص انواع تلفن‌های هوشمند و تبلت ها با توجه به تنوع سیستم عامل ، روش‌های مختلفی برای به دست آوردن آدرس مک وجود دارد که به عنوان نمونه برای اندروید با مراجعه به قسمت تنظیمات (setting) ، بخش سیستم (system) ، درباره دستگاه (about device) ، موقعیت (status) ، آدرس wifi mac ، می‌توانید آدرس مک دستگاه خود را مشاهده نمایید.