دانشگاه شهید مدنی آذربایجان

آزمایشگاه سیستم‌های دیجیتال ۲

دانشکده فنی و مهندسی

گروه مهندسی برق

تهیه کننده: دکتر موسی یوسفی

بهار۹۷

پیش گفتار

هدف از این آزمایشگاه آشنایی با روند طراحی سیستم‌های دیجیتال با استفاده از تراشه FPGA می‌باشد، در این آزمایشگاه کدهای ساده در زبان VHDL با استفاده از نرم افزار Modelsim شبیه‌سازی می‌شود، مراحل سنتز کدها با استفاده از نرم افزار Xilinx ISE انجام خواهد شد. دانشجویان مهندسی برق با مراحل طراحی سیستم‌های دیجیتال و نهایتاً پیاده‌سازی در برد آموزشی FPGA Xc500E آشنا خواهند شد. این دستورکار مشتمل بر ۸ آزمایش می‌باشد که شامل معرفی نرم‌افزارها و شبیه سازی و پیاده سازی سیستم‌های دیجیتالی با توجه به نکات برد آموزشی موجود درآزمایشگاه می‌باشد.

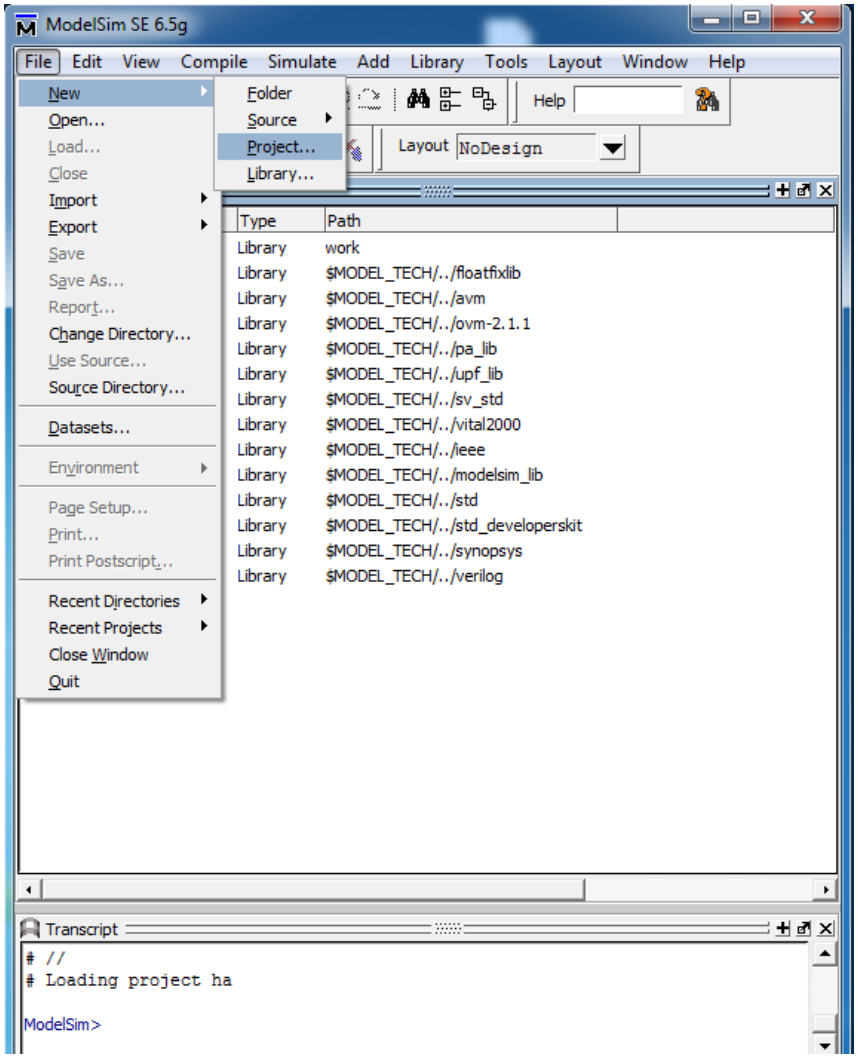
از تمام دانشجویان تقاضا داریم تمام نکات و موارد پیشنهادی خود را از طریق ایمیل m.yousefi@azaruniv.ac.ir به اینجانب ارسال کنند تا در ویرایش‌های بعدی دستور کار در نظر گرفته شود.

فهرست

آزمایش ۱

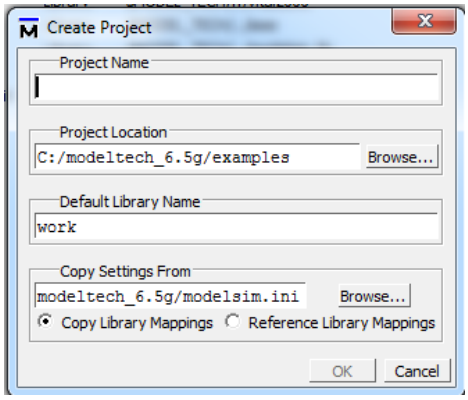
الف) مراحل ایجاد یک پروژه جدید در نرم افزار ModelSim

۱- بعد از اجرای نرم افزار ModelSim پنجره‌ای بصورت شکل۱-۱ باز می‌شود. از منوی File>New>Project را انتخاب کنید.



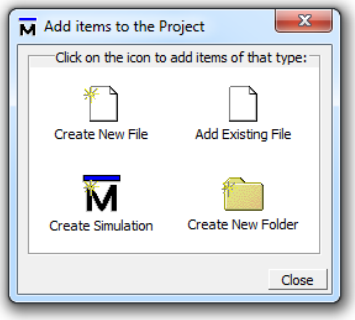
شکل۱-۱

۲- پنجره‌ای مطابق شکل ۱-۲ ظاهر خواهد شد. در این پنجره نام پروژه و مسیر ذخیره سازی پروژه از شما دریافت می‌شود. در کادر نام پروژه(Project Name) نام Test1 را وارد کنید.

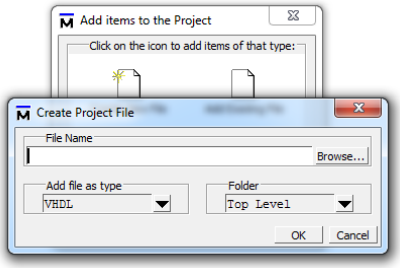


شکل ۱-۲

۳- پنجره شکل۱-۳-الف برای شما باز می‌شود. در این پنجره از گزینه Create New File برای ایجاد فایل VHDL استفاده کنید. در این شرایط پنجره‌ای مطابق شکل۱- ۳-ب برای وارد نام فایلی که کد مورد نظر خودتان را می‌خواهید بنویسید ظاهر می‌شود برای مثال اسم HA را در کادر File Name تایپ کرده و کلید OK را بزنید.



شکل۱- ۳-الف



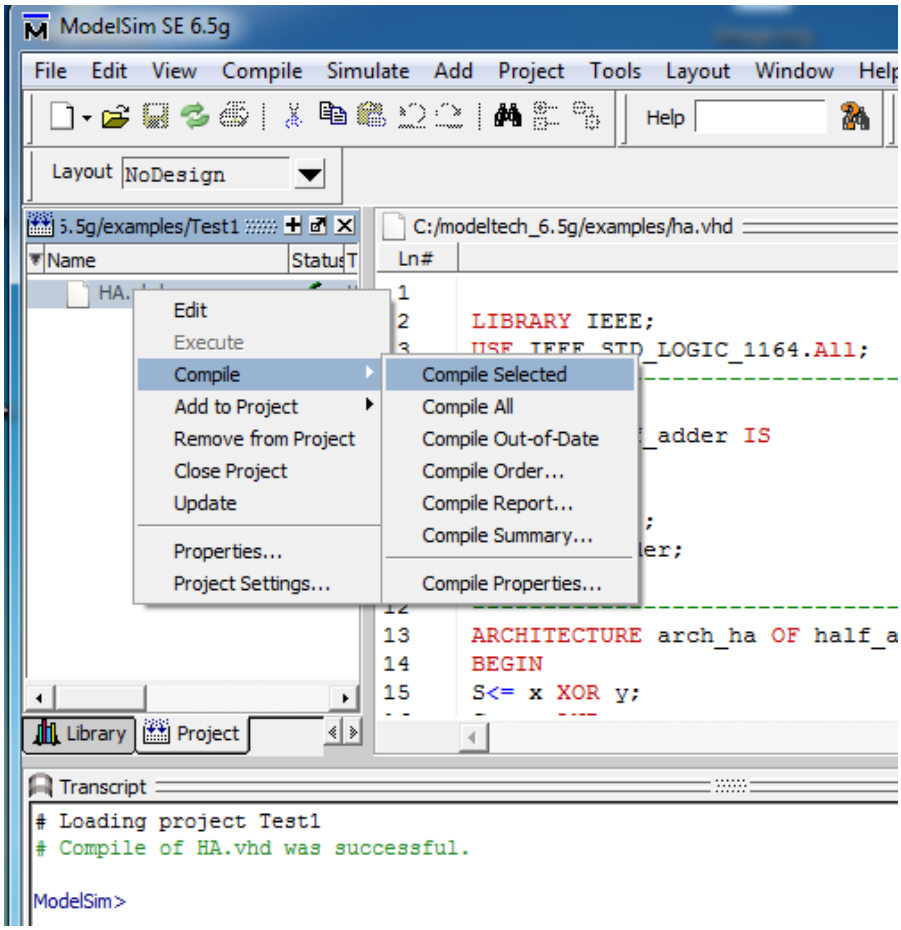
شکل۱- ۳-ب

۴- در این مرحله پنجره شکل ۱-۳ – ب را با زدن دکمه close ببندید. روی HA.vhd کلیک کرده تا صفحه کد نویسی برای شما باز شود. کد زیر را بنویسید.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.All;
-----------------------------------------
ENTITY  half_adder IS
PORT (
x,y: IN BIT;
s,c:OUT BIT);
END half_adder;
-----------------------------------------
ARCHITECTURE arch_ha OF half_adder IS
BEGIN
S<= x XOR y;
C<= x AND y;
END arch ha;
```

۵- مطابق شکل ۱-۴ برای کامپایل کد بالا از کلیک راست روی اسم HA.vhd و انتخاب Compile Selected استفاده کنید. در صورتی که کد ایرادی نداشته باشد جمله Compile of HA.vhd was successful در کادر Transcript ظاهر خواهد

شد و علامت تیک نیز در کنار HA.vhd ظاهر خواهد شد. در غیر اینصورت از کادر Transcript خطاهای احتمالی را پیدا کرده و اصلاح کنید.

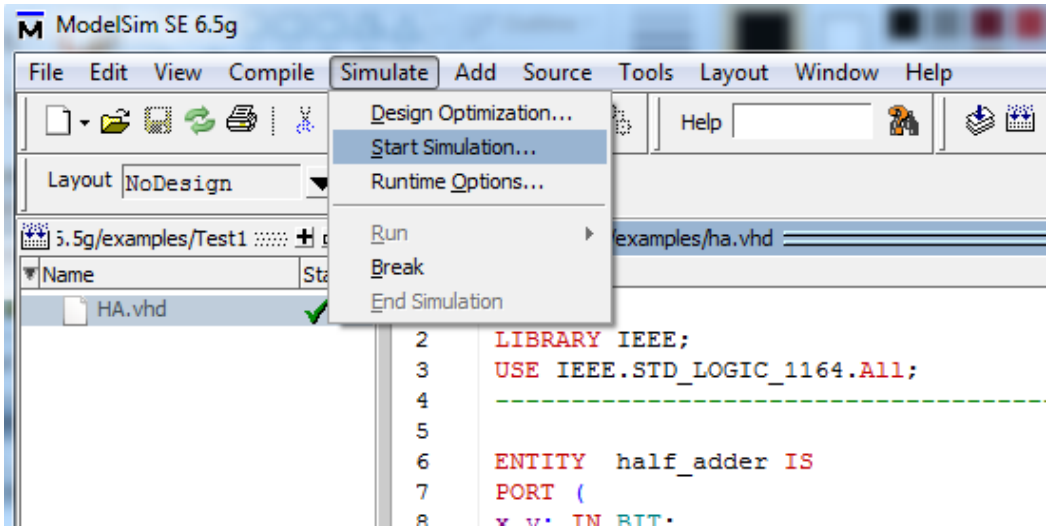


شکل ۱-۴

در صورتی که مراحل بالا بخوبی طی شود در این حالت کد را می‌توانیم شبیه سازی کنیم.

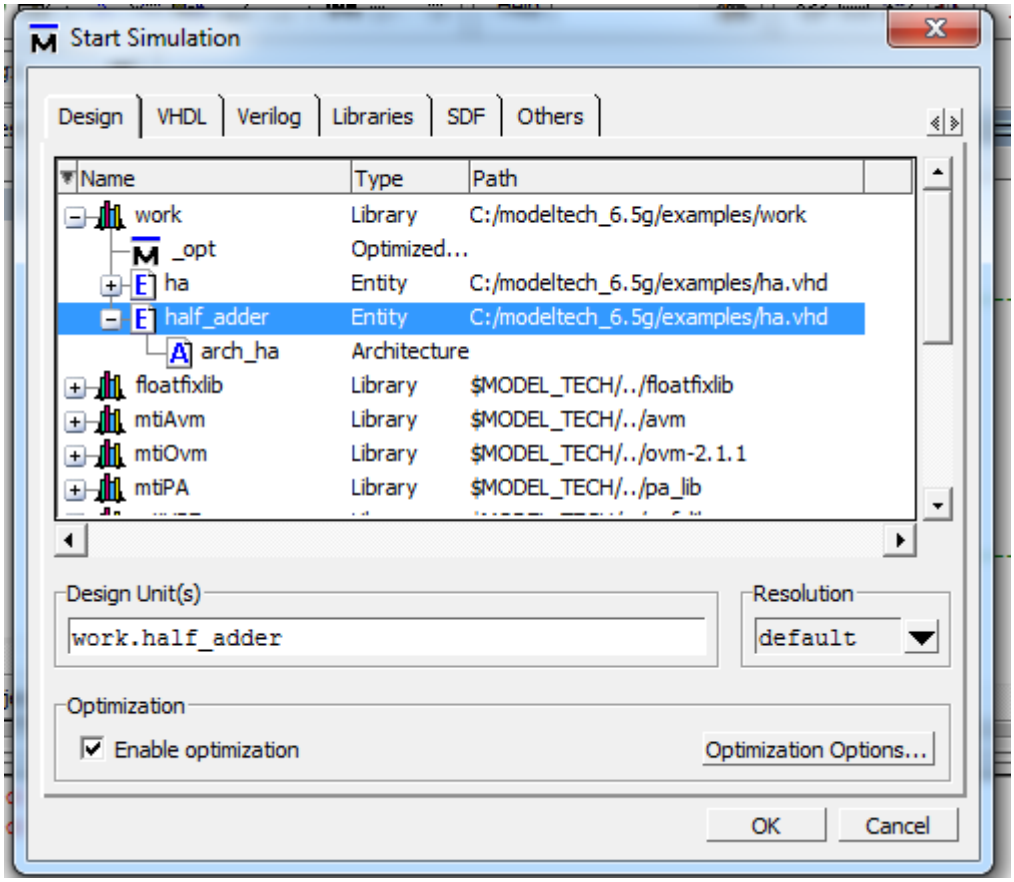
ب) مراحل شبیه سازی کد در نرم افزار ModelSim

۱-بعد از کامپایل کد، مطابق شکل۱- ۵ گزینه Start Simulation را بزنید.



شکل۱– ۵

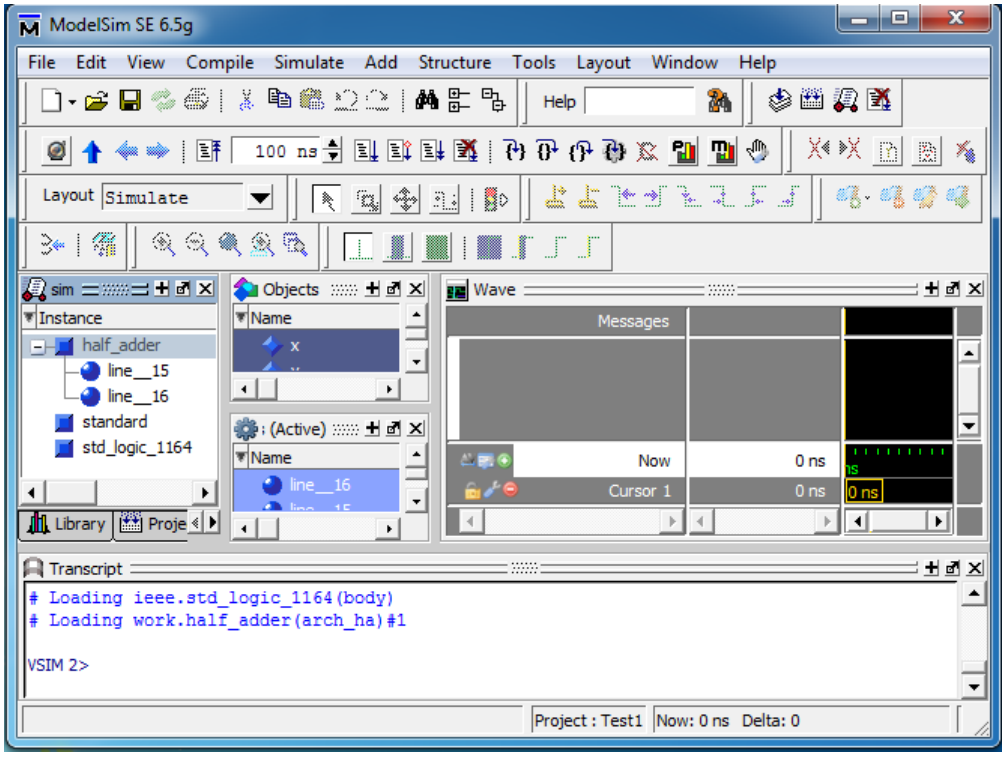
۲- از پنجره باز شده مطابق شکل ۱–۶ از میسر work-> half_adder را انتخاب کرده و کلید OK را بزنید.



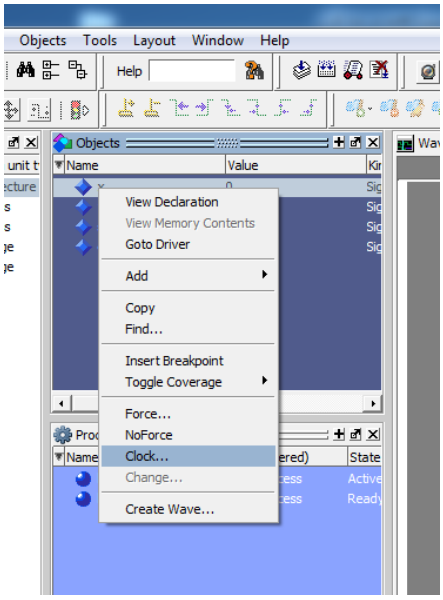
شکل۱– ۶

۳- مطابق شکل۱–۷ نرم افزارModelSim ابزارهای مختلف برای شبیه سازی و دیدن نتایج در اختیار شما قرار می‌دهد. در این قسمت به ازای ۴ حالت ممکن برای نیم جمع کننده که کد آن را نوشته وکامپایل کردیم برای دو وردی xو y به ترتیب دو سیگنال با

فرکانسهای ۱ مگا هرتز و ۲ مگا هرتز تعریف خواهیم کرد. برای این منظور مطابق شکل۱– ۸ روی سیگنال x کلیک راست کرده و گزینه clock را انتخاب کنید.

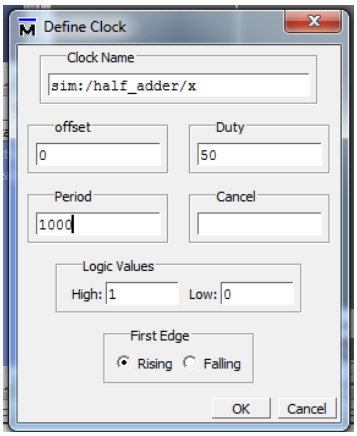


شکل ۱–۷



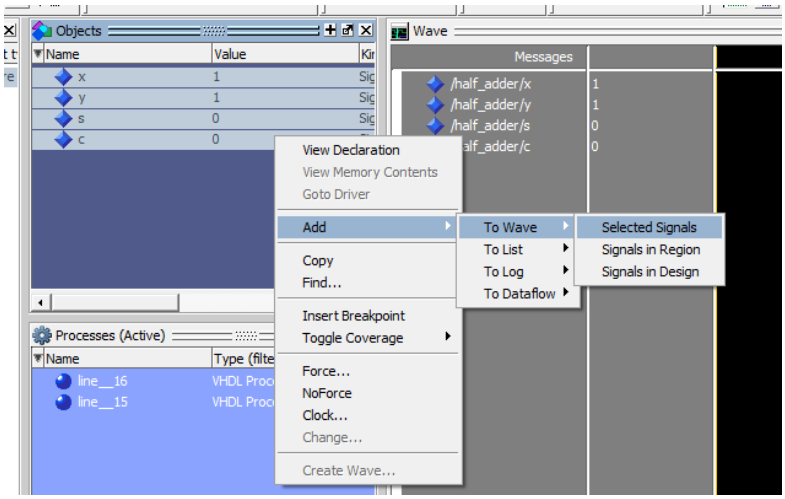
شکل ۱–۸

۴– در این مرحله پنجره‌ای مطابق شکل ۱–۹ ظاهر می‌شود در این قسمت در بخش period مقدار ۱۰۰۰ را وارد کنید. بعد کلید ok را بزنید. همین مرحله را برای سیگنال y نیز تکرار کرده و عدد ۲۰۰۰ را در کادرperiod مربوط به این سیگنال وارد کنید.



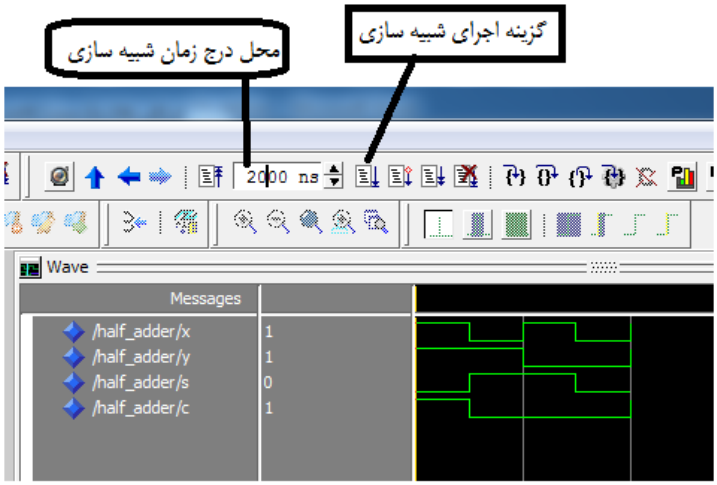
شکل ۱-۹

۵- برای دیدن نتایج بعد از شبیه سازی مطابق شکل ۱-۱۰ با استفاده از کلید shift همه سیگنالهای x, y, s, c را انتخاب کرده و کلیک راست کنید گزینه Add>To Wave> Selected Signals را انتخاب کنید.



شکل ۱-۱۰

۶- در این مرحله با تغییر زمان شبیه سازی به ۲۰۰۰ نانو ثانیه مطابق شکل۱-۱۱ از دکمه برای انجام شبیه سازی استفاده کنید .



شکل۱-۱۱

آزمایش ۱-۲ کد زیر را در محیط ModelSim نوشته و شبیه سازی کنید بعد شبیه سازی جدول عملکرد خروجی را بدست آورید.

```
Library IEEE;
use ieee.std_logic_1164.all;
-----------------------------------------
Entity Az1_1 is
  PORT ( i0,i1,i2,i3,s0,s1: in std_logic;
  y: out std_logic);
End Az1_1;
-----------------------------------------
Architecture arch_az1 of Az1_1 is

Begin
  y<= i0 When s0='0' and s1='0' else
      i1 When s0='1' and s1='0' else
      i2 When s0='0' and s1='1' else
      i3 When s0='1' and s1='1' else
      '0';
End arch az1;
```

تمرین۱- ۱: کد تمام جمع کننده را نوشته و شبیه سازی کنید.

تمرین ۱-۲: کد جمع کننده باینری ۱۲ بیتی را نوشته و به ازای یک ورودی دلخواه شبیه سازی کنید.

تمرین ۱-۳: در تمام آزمایش‌ها تمام خطاهای ایجادشده در زمان کامپایلر در هر بخش آزمایشها را یادداشت کرده و در انتهای هر گزارش ذکر کنید.

آزمایش ۲

مدار ترکیبی– مالتی پلکسر– دیکدر– انکدر

در کد VHDL زیر از عبارت WHEN/ELSE استفاده شده است در این عبارت زمانیکه Boolean_expr_1 درست ارزیابی شود مقدار متناظر آن به signal_name تخصیص داده می‌شود.

قاعده نوشتاری When/Else

```
Signal_name <= value_expr1 WHEN Boolean_epr_1 ELSE
               value_expr2 WHEN \Boolean _epr_2 ELSE
               value_expr3 WHEN Boolean_epr_3 ELSE
               ….
               Value_eprn;
```

آزمایش ۲– ۱: کد مالتی پلکسر ۴ در ۱ شکل ۲–۱ را در محیط نرم‌افزار ModelSim نوشته وشبیه سازی عملکرد آن را انجام دهید.

```
library IEEE;
use ieee.std_logic_1164.all;
---------------------------------------
Entity mux_4_1 is
Port ( d : in std_logic_vector(3 downto 0);
  S:in st_logic_vector(1 downto 0);
Y: out std_logic);
End mux_4_2;
-----------------------------------------------
Architecture arch_mux of mux_4_1 is
Begin
Y<=  d(0) when s="00" else
d(1) when s="01" else
d(2) when s="10" else
d(3) ;
End arch_mux;
```

شکل ۲–۱: کد مالتی پلکسر ۴در۱

آزمایش۲–۲: کد VHDL مالتی پلکسر ۸ در ۱ را بنویسید و شبیه سازی کنید.

تمرین‌های آزمایش دوم

تمرین۲–۱: کدVHDL دیکدر ۲ در ۴ را با امکان فعال سازی بنویسید و شبیه‌سازی کنید.

تمرین۲–۲: کدVHDL انکدر ۴ در ۲ با تقدم با ارزش را نوشته وشبیه‌سازی کنید.

آزمایش ۳

مدارهای ترتیبی منظم – فلیپ فلاپ و ثبات

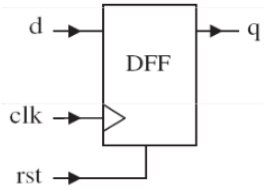
مقدمه:

عبارت Process در کد VHDL برای نوشتن دستورات ترتیبی استفاده می‌شود همانطوریکه در کد زیر نشان داده شده است عبارت‌های داخل Process زمانی اجرا می‌شوند که یکی از سیگنال‌های داخل لیست حساسیت تغییر کند.

قاعده نوشتاری Process

Process (sensitivity_list)
Begin
Sequential statement;
Sequential statement;
. . .
End process;

آزمایش ۳-۱: کد فلیپ فلاپ حساس به لبه بالا رونده زیر را در محیط نرم‌افزار ModelSim نوشته و شبیه سازی کنید.

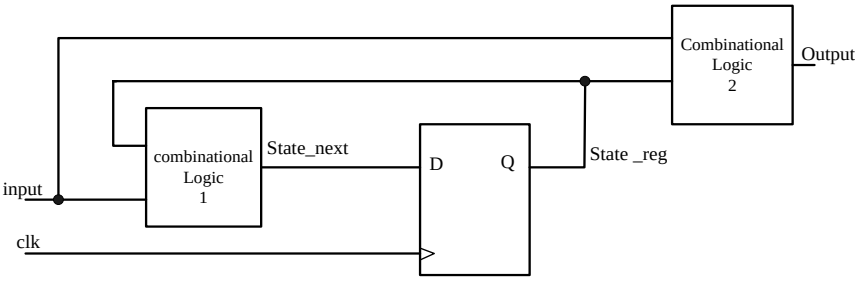


شکل۳-۱: فلیپ فلاپ D

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-----------------------------------------
ENTITY dff IS
  port(d, clk, rst: IN STD_LOGIC;
     q: OUT STD_LOGIC);
END dff;
-----------------------------------------
ARCHITECTURE behavior OF dff IS
BEGIN
   PROCESS(clk)
  BEGIN
    IF (rst='1') THEN
      q <= '0';
    ELSIF (clk'EVENT AND clk='1') THEN
      q <= d;
    END IF;
  END PROCESS;
END behavior;
```

آزمایش ۳-۲ : کد VHDL بند۳-۱ را طوری تغییر دهید که سیگنال ورودی rst بصورت غیر همزمان عمل کند. کد را نوشته و شبیه سازی کنید.

آزمایش ۳-۳: با استفاده از ساختار نشان داده شده در شکل ۳-۱ بلوک دیاگرام مدارهای ترتیبی منظم می‌باشد. در این ساختار وضعیت بعدی از طریق ورودی‌های خارجی و وضعیت فعلی مدار و با استفاده از منطق حاکم در مدار نرکیبی ۱ تهیه می‌شود و با اعمال پالس ساعت به عنوان حالت بعدی در فلیپ فلاپ حفظ می‌شود. کد VHDL فلیپ فلاپ با امکان بار شدن مطابق جدول ۳-۱ که در شکل ۳-۳ نشان داده شده را نوشته و شبیه سازی کنید.



شکل ۳-۱:بلوک دیاگرام مدار ترتیبی منظم

جدول ۳-۱ : جدول عملکرد فلیپ فلاپ با امکان بار شدن موازی

Clk	Reset	En	Output
X	1	X	0
↑	0	1	d
↑	0	0	No-change

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------------
ENTITY dff IS
  port(d, clk, rst, en: IN STD_LOGIC;
       q: OUT STD_LOGIC);
END dff;
-------------------------------------------
ARCHITECTURE behavior OF dff IS
Signal r_reg, r_next : std_logic;
BEGIN
   PROCESS(clk,rst)
  BEGIN
    IF (rst='1') THEN
 R_reg; <= '0';
    ELSIF (clk'EVENT AND clk='1') THEN
 R_reg; <= R_next
    END IF;
  END PROCESS;
R_next<= d WHEN en='1' else
  R_reg;
Q<= r_reg;
END behavior;
```

تمرین آزمایش سوم

تمرین ۳-۱: کد ثبات کامل N بیتی (بصورت پیش فرض N=8) مطابق جدول ۳-۱ را نوشته و شبیه سازی کنید. (با استفاده از عبارت Generic برای پارامتری کردن کد می‌توان استفاده کرد)

جدول ۳-۱ : جدول عملکرد ثبات کامل

Clk	Clear	Load	Shift	Operation
	1	X	X	پاک شدن همه خروجیها
	0	1	0	بار شدن موازی
	0	0	0	شیفت به راست
	0	0	1	شیفت به چپ
	0	1	1	بلا تغییر

آزمایش ۴

مدارهای ترتیبی منظم– شیفت ریجیستر و شمارنده

آزمایش ۴-۱: در زیر کد VHDL شمارنده ۴ بیتی حلقوی که از حالت "0000" تا "1111" را بصورت بالا رونده طی می‌کند و دوباره به حالت اول بر می‌گردد. کد VHDL زیر را نوشته و شبیه سازی کنید.

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
--------------------------------------------------
Entity counter_4_bit is
  Port ( clk, reset: in std_logic;
      Q: out std_logic_vector ( 3 downto 0));
End counter_4_bit;
--------------------------------------------------
Architecture arch_counter of counter_4_bit is
Signal r_reg: unsigned (3 downto 0);
Signal r_next: unsigned (3 downto 0);
Begin
Process ( clk,reset)
    Begin
      If( reset='1') then
       R_reg<=(others=>'0');
      Elsif ( clk'event and clk='1') then
       R_reg<=r_next;
      End if;
End process;
  R_next<= r_reg+1;
  q<= std_logic_vector( r_reg);
end arch_counter;
```

آزمایش ٤-۲: کد VHDL شمارنده ۴ بیتی بنویسید که بعد از رسیدن به عدد ۱۳ خروجی به اسم pulse به اندازه یک پالس ساعت یک شود.

تمرین آزمایش چهارم

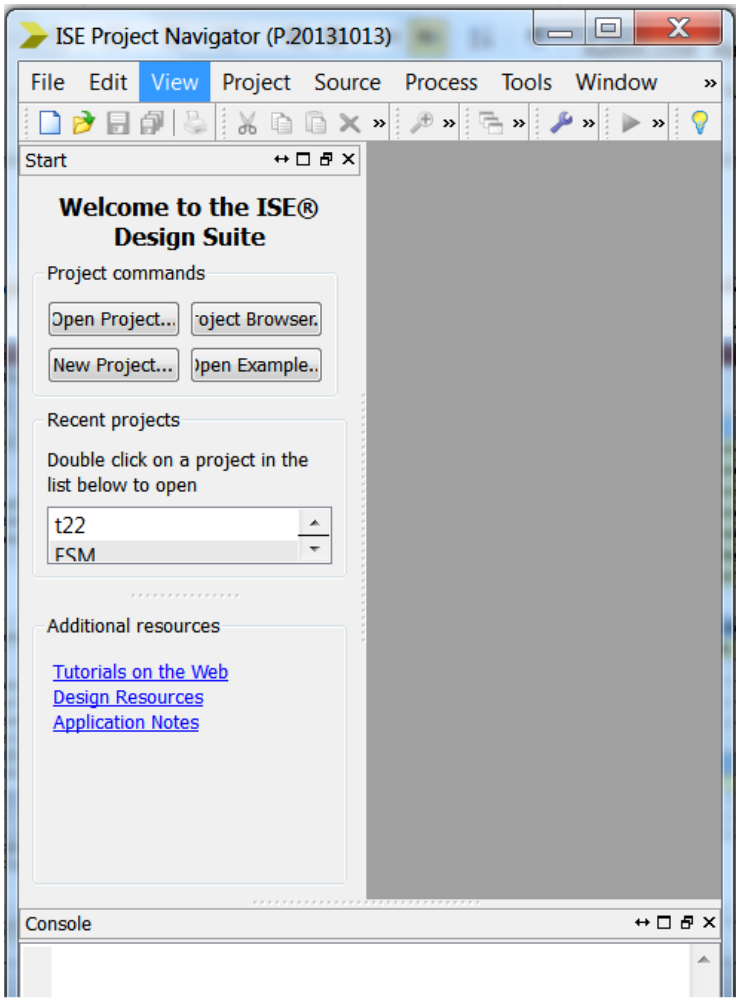
تمرین۴-۱: کد VHDL آشکار ساز کد "1011" را بنویسید و شبیه سازی کنید.

آزمایش ۵

مراحل ایجاد پروژه و سنتز کد VHDLدر محیط نرم‌افزار ISE

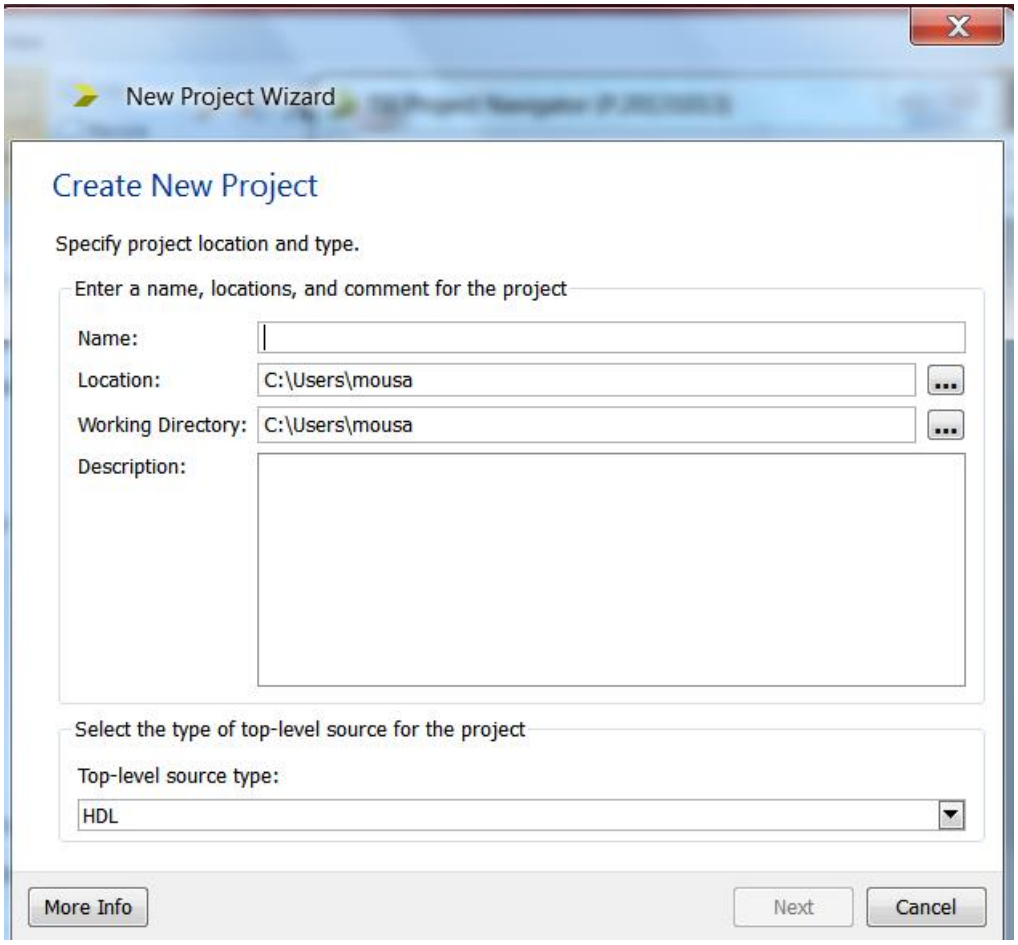
۵-۱ مراحل ایجادیک پروژه در محیط IS

۵-۱-۱ بعد از نصب نرم افزار در سیستم کامپیوتر از مسیر نصب یا از میزکار کامپیوتر (Desktop) آیکن Project Navigator را کلیک کنید. بعد از اجرای درست نرم‌افزار شکل۵-۱ را مشاهده خواهید کرد.



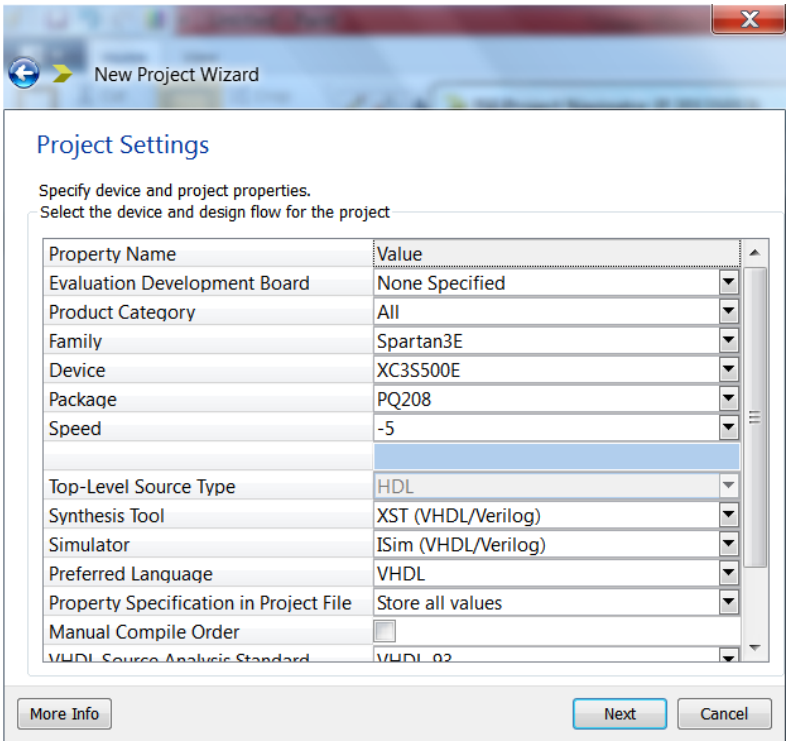
شکل ۵-۱: نمایی از محیط نرم افزار ISE

۵-۱-۲ برای ایجاد پروژه جدید روی گزینه New Project کلیلک کنید در این شرایط پنجره شکل ۵-۲ باز می‌شود.



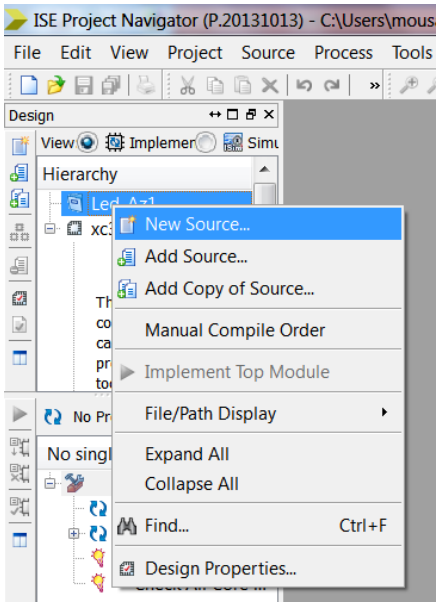
شکل۵-۲

۵-۱-۳ در این پنجره اسم پروژه مورد نظر را درکادر Name که در شکل مشاهده می کنید وارد کرده و مسیر ذخیره سازی فایل را از کادر Location می‌توانید مشخص کنید. بعد دکمه Next را بزنید. در پنجره شکل ۵-۳ مشخصات آی سی FPGA مد نظرتان را وارد کنید. در ضمیمه آزمایشگاه مشخصات کامل آی سی استفاده شده در این آزمایشگاه ذکر شده است. با کلیک روی گزینه NEXT پنجره نهایی که خلاصه‌ای از مشخصات پروژه ایجاد شده را نشان می‌دهد ظاهر می‌شود این پنجره را با زدن دکمه FINISH ببندید تا وارد محیط نرم افزار برای پروژه ایجادشده شوید.



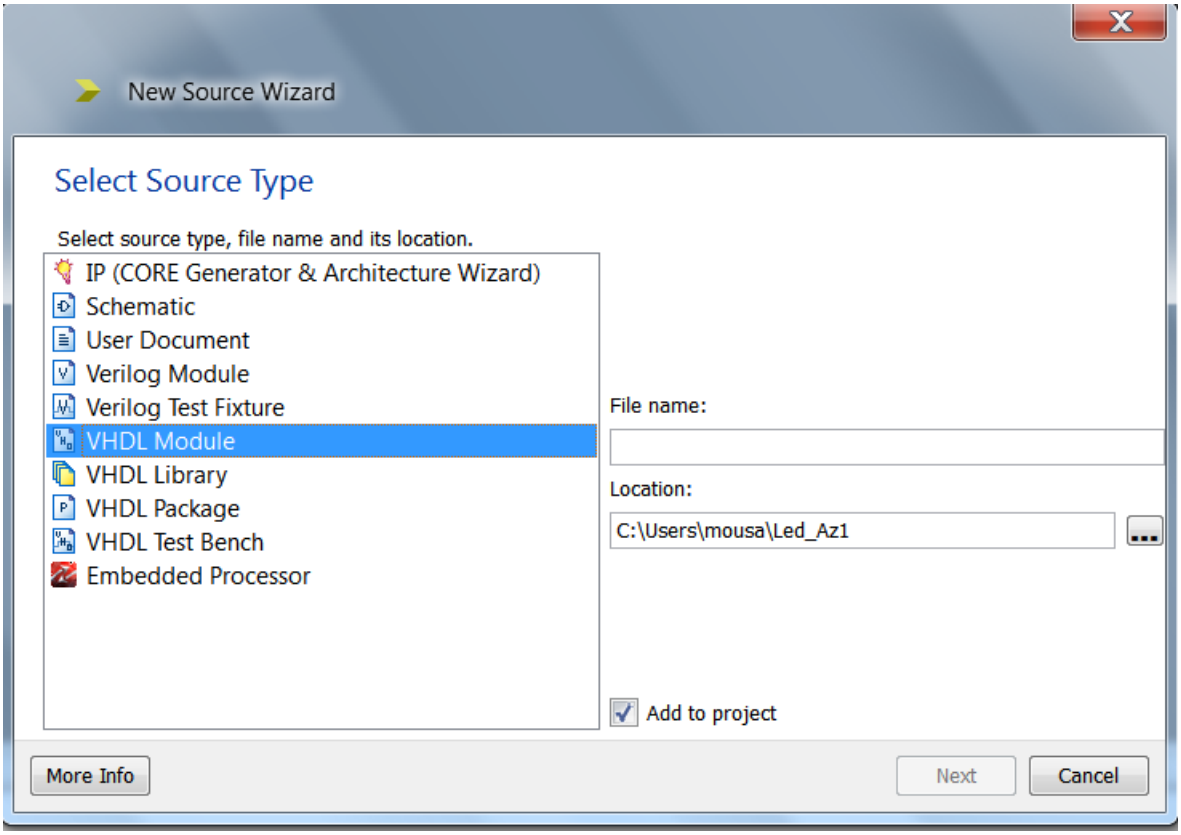
شکل ۵–۳: پنجره مربوط به وارد کردن مشخصات آی سی FPGA

۵–۱–۴ در پنجره شکل ۵–۴ در بخش Hierarchy با کلیک راست روی نام پروژه پنجره‌ای باز می‌شود در این قسمت New source را انتخاب کنید.

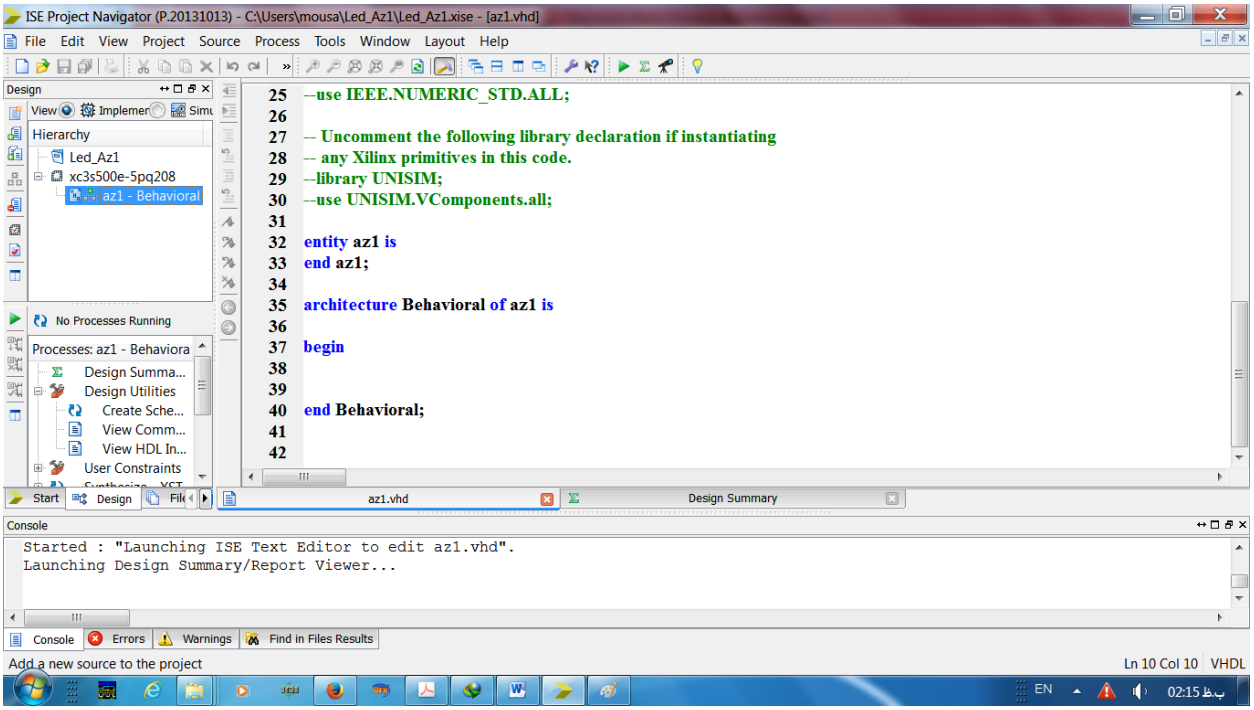


شکل ۵–۴: ایجاد یک فایل برای پروژه در محیط نرم‌افزار ISE

۵-۱-۵ در پنجره ایجاد شده مطابق شکل ۵-۵ در کادر گزینه VHDL Module را انتخاب کرده و سپس در کادر file name نام کد VHDL مورد نظر را وارد کنید، سپس کلید NEXT را بزنید.در پنجره بعدی نام ورودیها و خروجیها بطور کلی درگاه‌های سیستم دیجیتالی مورد نظر شما وارد می شود برای این قسمت در این آزمایش کاری انجام ندهید دکمه Next را زده ودر پنجره بعدی کلید Finish را بزنید. محیط مورد نیاز شما برای وارد کردن کد VHDLآماده است. دقت کنید در این صفحه برخی عبارات کد و توضیحات اضافی قرار دارد.



شکل۵-۵: پنجره تعیین نوع فایل الحاقی به پروژه



شکل ۵-۶

۵-۲ مراحل سنتزو برنامه ریزی آی سی FPGA با کد VHDL

۵-۲-۱ پروژه‌ای با نام Test1 ایجادکنید (مطابق دستورالعمل بخش ۵-۱) و کد آزمایش جمع کننده ۱۲ بیتی را بعد از پاک کردن کل محیط مربوط به فایل VHDL کپی کنید.

۵-۲-۲ برای سنتز کد VHDL از پنجره Process گزینه Synthesize را بزنید. در این حالت اگر کد ایراد نداشته باشد کد سنتز شده و نتیجه موفق بودن در پنجره console نمایش داده می‌شود.

تمرین ۵-۱: بعد از سنتز کد آزمایش بخش ۴-۲، نتایج آن در پنجره Process و قسمت Design Summery قرار دارد. نتایج سنتز این کد را با توضیحات مناسب یادداشت کرده و گزارش کنید.

آزمایش ۶

برنامه ریزی تراشه FPGA

مقدمه

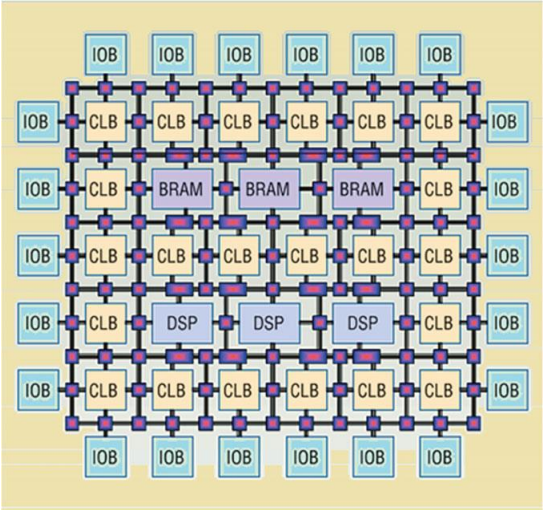
به دلیل اینکه تراشه‌های قابل برنامه‌ریزی بارها برای مشخصه‌های مختلف طراحی قابل برنامه‌ریزی هستند داشتن اینگونه تراشه‌های نیاز مبرم صنعت طراحی نیمه‌هادی است. هزینه ساخت و پیاده‌سازی سیستم‌های مبتنی بر PLDها در مقایسه با مدارهای یکپارچه فشرده با کاربرد خاص(ASICs) پایین‌تر است و امکان تزریق میلیون‌ها دلار، برای پیاده‌سازی ایده جدید در مرحله اولیه طراحی وجود ندارد. برای حداقل سرمایه‌گذاری در زمان پیاده‌سازی ایده جدید و همچنین برای تایید نهایی عملکرد سیستم توصیه همیشگی استفاده از تراشه‌های قابل برنامه‌ریزی است. اگر یک دهه گذشته را در نظر بگیریم نیاز به استفاده از میلیون تا میلیارد دروازه منطقی در SOCها باعث رشد واقعی بازار PLDها است. در طراحی SOCها بطور گسترده برای تایید نهایی رفتار سیستم از PLDها استفاده می‌شود. PLDها در بخش‌های متنوع بازار همانند خودرو، مصرف‌کننده، ارتباطات کامپیوتری، بی‌سیم و حوزه‌های صنعتی برای تایید مفهوم ایده‌ها استفاده می‌شود. جدول ۶-۱ اطلاعاتی در مورد درآمد حاصل از پروژه‌های دنیای نیمه‌هادی تا سال ۲۰۱۸ را نشان می‌دهد.

٦-١ آشنایی با تراشه FPGA[1]

آرایه گیت‌های قابل برنامه‌ریزی (FPGA) می‌تواند در حوزه مورد نظر کاربر برنامه‌ریزی یا پیکربندی شود که بطور چشم‌گیری در طراحی سیستم‌های پیچیده استفاده می‌شود. حتی امروزه، از FPGAها برای پیاده سازی سیستم‌های پیچیده SOC و اثبات مفهوم ایده‌ها استفاده می‌شوند. استفاده گسترده از FPGA بخاطر قابلیت دسترسی به ماکرو نرم افزار و سخت‌افزار مورد نیاز برای پردازنده، DSP و پردازش تصویر در طول این دهه افزایش یافته است. حتی اکثر ساختارهای پیچیده FPGAها اتصالات سرعت بالا، اینترانت، USB، و پروتکل AHB را پشتیبانی می‌کنند.

ساختار FPGA پایه را همانند یک دریای از بلوک‌های منطقی یا بلوک‌های منطقی قابل برنامه‌ریزی (CLBs)، بلوک‌های ورودی/خروجی(IOBS)، بلوک‌های حافظه (BRAMs)، بلوک‌های DSP و منابع مسیریابی دیگر می‌توان تصور کرد. بلوک‌های اساسی FPGA در شکل ۶-۱ نشان داده شده است.

[1] Field Programmable Gate Array



شکل۶-۱ ساختار FPGA پایه

هانطوریکه در شکل نشان داده شده FPGA شامل:

۱- CLB برای ایجاد منطق ترکیبی و ترتیبی استفاده می‌شود. CLB نشان داده شده از تعدادی جدول جستجو (LUTs) و ثبات‌ها تشکیل شده است. تابع ترکیبی با استفاده از LUTs ایجاد می‌شوند و این LUTs تاخیر یکنواختی دارند. اگر با استفاده از یک CLB نتوان تابع را بدست آورد نیاز است که با توجه به رفتار تابع از چند CLB پیکربندی شونده استفاده کنیم. این CLBها با استفاده از برنامه vendor-specific یا فایل bit-map پیکربندی می‌شود.

۲- IOB برای ایجاد ارتباط بین جهان پیرامونی و CBLها و برعکس استفاده می‌شود. IOB شامل بافرهای دوجهته با ثبات‌ها است ورودی می‌تواند با استفاده از بلوک IO ثبت شود (ذخیره شود) و حتی خروجی با استفاده از بلوک IO ثبت شود (ذخیره شود). ورودی و خروجی‌ممکن است ثباتی نشده باشند. و با توجه به ملزومات رفتاری در طراحی IO می‌تواند مانند IO ثبت شده یا IO ثبت نشده پیکر بندی شود.

۳- FPGA BRAM می‌تواند RAM توزیع شده و بلوک RAM (BRAM) داشته باشد. RAM توزیع شدهبا استفاده از LUTها می تواند ایجاد شود و BRAM ها توسط بلوک‌های RAM توزیع شده می‌تواند ایجاد شود که توسط ابزار طرحی vendor-specific برای پیکربندی و اندازه مورد نیاز برنامه‌ریزی شود. ایجاد RAM تک درگاهی و دودرگاهی با استفاده از BRAMها امکان پذیر است. ظرفیت BRAM ها به ساختار وابسته است.

۴- DSP بلوک‌های از پیش تعریف شده معینی وجود دارند که برای ایجاد عملکرد DSP می‌تواند پیکربندی شوند. اکثر کاربردهای DSP نیاز به ضرب‌کننده‌ها، ثبات‌های لوله‌ای، بلوک‌های رفتاری DSP معین برای عملیاتDSP و غیره دارند. برای محاسبات DSP سرعت بالا این بلوک‌ها استفاده می‌شود و می‌تواند با توجه به ملزومات طراحی پیکربندی شوند.

به غیر از بلوک‌های بالا، هر FPGA دارای ساختاری ساعتی[۲]است که در واقع همان بلوک ساعت است: Xillinx از مدیریت ساعت دیجیتال برای ایجاد حلقه قفل تاخیر DLL استفاده می‌کنددر حالیکه Altera از PLL برای شبکه ساعت استفاده می‌کند. شبکه ساعت برای ایجاد ساعت یکنواخت و ساعت بدون هازارد و گلیچ استفاده می‌شود.

هر FPGA می‌تواند منابع مسیریابی چند تایی داشته باشد و برای فراهم کردن ارتباط بین بلوک‌های مختلف FPGA استفاده می‌شود. ابزار vendor-specific برای پیکر بندی FPGA با استفاده از حداقل تاخیر در مسیر استفاده می‌شود.

[۲]Clocking

بخش زیر شامل پیاده‌سازی طرح‌های کاربردی با FPGA است. به مهندس طراح توصیه می‌شود برای بدست آوردن خروجی بهتر از طراحی با VHDL ساختارFPGA را یاد بگیرد، این باعث خواهد شد طراحی بهینه ایجاد شود، همچنین توصیه می‌شود با منابع ساختاری را قبل از نوشتن کدRTL آشنا شود که باعث خواهد شد نتیجه سنتز بهتری بدست آید حتی با استفاده از این استراتژی می‌توان فضای مصرفی را کاهش داد و باعث بهبود عملکرد توانی و سرعت سیستم شد.

۶-۲ مفهوم LUT و ایجاد منطق ترکیبی

مفهوم LUT را با استفاده از طراحی‌هایی مبتنی بر MUX به آسانی می‌تواند درک کرد. تابع منطقی با استفاده از LUT پیاده‌سازی می‌شود. LUT تاخیر یکنواختی بدون در نظر گرفتن ورودی‌ها برای یک طرح ایجاد می‌کند.

شکل ۶-۲ LUT با چهار ورودی

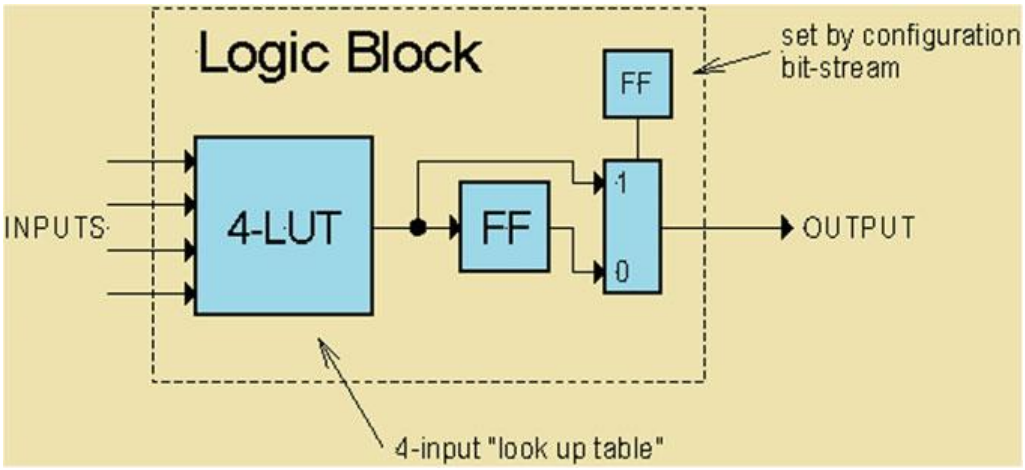
LUT چهار ورودی نشان داده شده در شکل ۶-۲ را در نظر بگیرید از این LUT برای پیاده‌سازی تابع منطقی با چهار ورودی و یک خروجی می‌توان استفاده کرد.

۶-۳ طراحی VHDL و پیاده سازی با استفاده از CLB

مطابق بحث اخیر CLB پایه شامل LUTها و ثبات‌ها است و با توجه به ساختار تراشه تعداد LUTها و ثبات‌ها می‌تواند تغییر کند همچنین حتی تعداد ورودی‌ها و خروجی‌های ساختار LUT نیز می‌تواند متغیر باشد. برای درک ساده ساختار پایه CLB در شکل ۶-۱۴ نشان داده شده است.

برای پیاده‌سازی توابع منطقی، RAM توزیع شده و همچنین شیفت ریجسترها از LUT می‌توان استفاده کرد. همانطوریکه در شکل ۶-۳ نشان داده شده با توجه به پیکربندی با فایل رشته داده خروجی ترکیبی یا ترتیبی است. اگر SRAM منطق یک داشته باشد خروجی منطق ترکیبی است و اگر پیکربندیFF منطق ۰ داشته باشد خروجی ثباتی است.

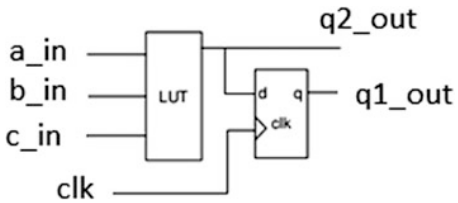
کد RTL با VHDL توصیف شده در شکل ۶-۲ را ملاحظه کنید.



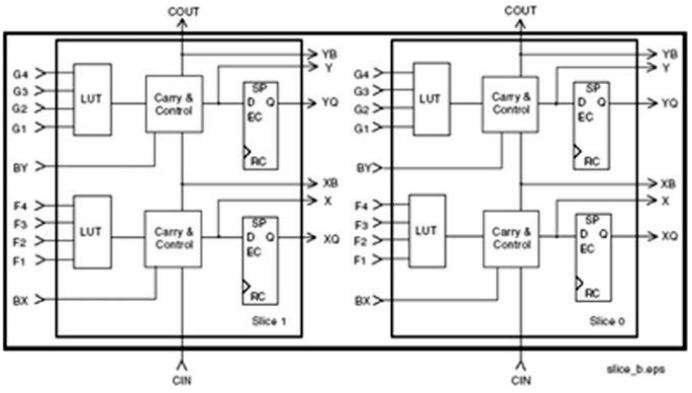
شکل ۶-۴ ساختار CLB پایه

نتیجه سنتز در شکل ۶-۱۵ نشان داده شده است. منطق با استفاده از LUT ۴ ورودی ایجاد شده است. ورودی LUT ها 'a_in' ، 'b_in' ، 'c_in' و 'd_in' با استفاده از برنامه vendor-specific پیکربندی شده و خروجی بدون ثبات 'q_out' با استفاده از منطق MUX ایجاد شده است.

اما مطابق با پیچیدگی طراحی تعداد ورودی‌ها افزایش می‌یابد در چنین شرایطی ساختار CLB می‌تواند چندین LUT و چندین ثبات با بلوک‌های نشان داده شده برای جمع کننده‌ها داشته باشد. شکل ۶-۱۶ ساختار CLB را توصیف می‌کند که شامل LUTهای ۳ ورودی است همچنین شامل جمع‌کننده و ثبات است. خروجی CLB می‌تواند ترتیبی یا ترکیبی باشد.



شکل ۶-۵ پیاده‌سازی منطقی با CLB

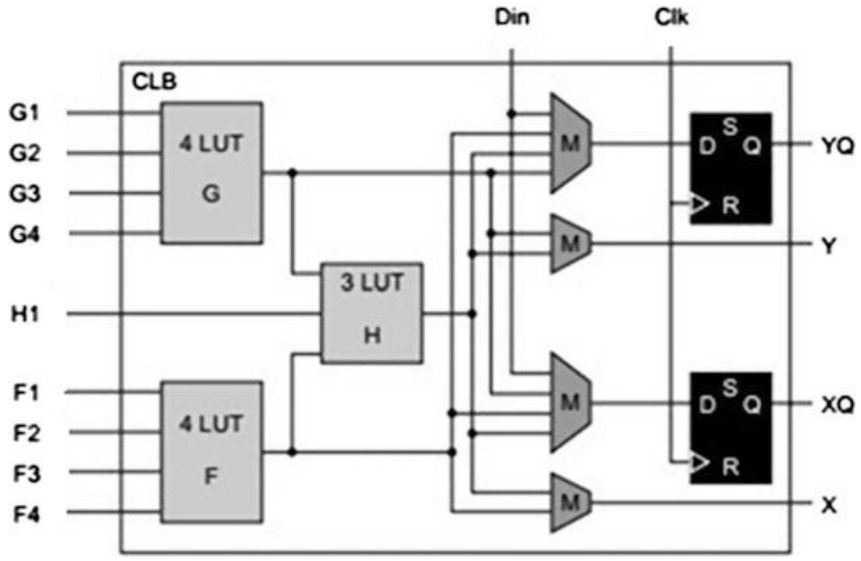


شکل ۶-۶ ساختار CLB در FPGA سری Virtex

شکل ۶-۵ اطلاعاتی درباره سنتز و پیاده‌سازی با استفاده از CLB ارائه می‌دهد.

مطابق بحث قبلی، ساختار FPGA یک تراشه خاص است. تفاوت بین تراشه های سری Altera/ XILINX در تفاوت بین بلوک‌های CLB است. ساختار CLB سری Virtex تراشه XINLINX در شکل ۶-۶ قابل مشاهده است در شکل ۶-۲ اسلایس نشان داده شده است و هر CLB دو اسلایس است هر دوSlice1 و Slice0 مشابه هم هستند و منطق دیجیتالی را با خروجی ثباتی و ترکیبی ایجاد می‌کنند.

همانطوریکه در شکل ۶-۶ نشان داده شده هر اسلایس شامل دو LUT ۴ ورودی و دو ثبات است و هر اسلایس رقم نقلی و منطق کنترل است که برای پیاده‌سازی منطق جمع استفاده می‌شود.



شکل ۶-۷ CLB با دو LUT ۴ ورودی

برای پیاده‌سازی تابع از LUT ۴ ورودی استفاده می‌شود و اسلایس صفر با اسلایس ۱ می‌تواند ارتباط داشته باشد. هر خروجی یکCLB داده را به CLB دیگری می‌تواند انتقال دهد. مسیریابی با اسافاده از الگوریتم‌های مسیر یابی با vendor-specific انجام می‌شود.

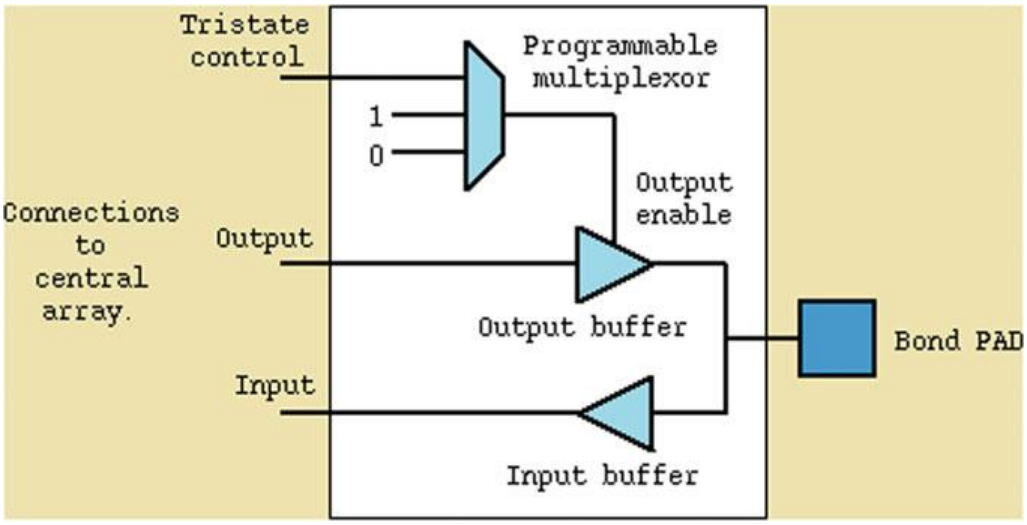
۸ ورودی CLB با خروجی ثباتی و خروجی ترکیبی در شکل ۶-۲۰ نشان داده شده است که از این برای طراحی تابع منطقی با ۸ ورودی می‌توان استفاده کرد. از CLB برای بدست آوردن خروجی ثباتی و ترکیبی می‌توان استفاده کرد.

مولد LUT یا تابع با G، H و F نامگذاری شده‌اند. حال فرض کنیم هدف طراحی دیکدر ۸ به ۲۵۶ با FPGA باشد. با استفاده از ساختار CLB نشان داده شده در شکل ۶-۷ دیکدر ۸ به ۲۵۶ را می‌تواندپیاده‌سازی کرد اما LUTهای زیادی لازم برای ۲۵۶ خروجی نیاز دارد در واقع آن نیاز به ۳ در ۲۵۶ LUT دارد یعنی ۷۶۸ تا LUT مورد نیاز است. یک تک خروجی از 'y' یا 'x' گرفته می‌شود. خروجی، خروجی ثباتی نیست.

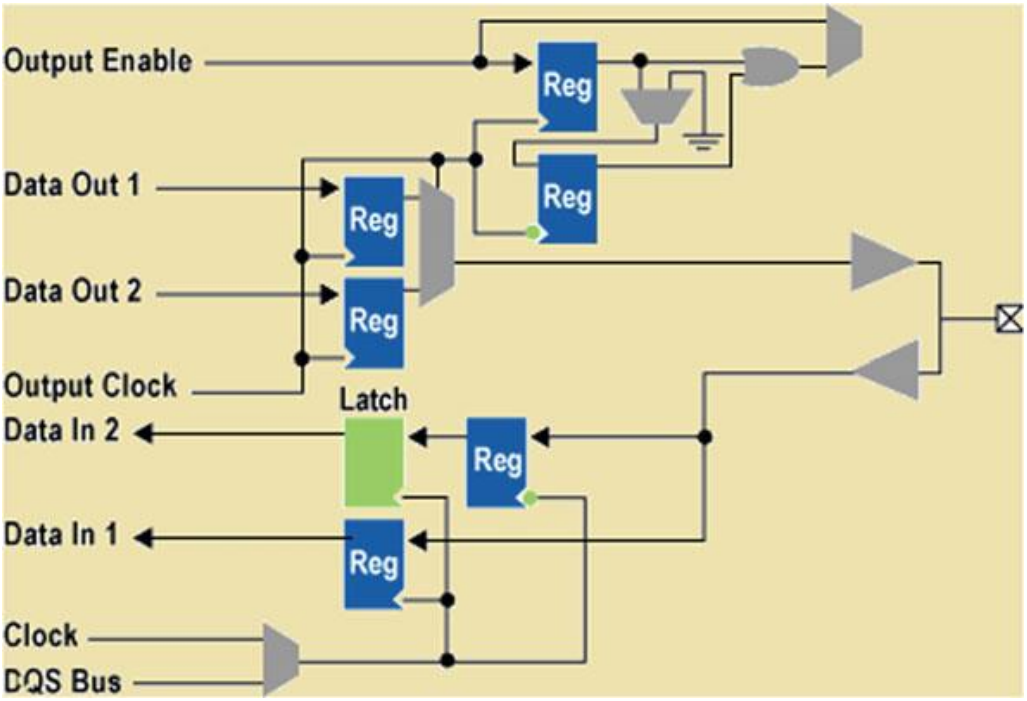
همانطوریکه در فصل ۴ بحث شد برای پیاده‌سازی دیکدر می‌توان از ساختار عبارت case استفاده کرد.اگر از case استفاده شود در واقع منطق ترکیبی ایجاد می‌شود تا یک تک خروجی را با بهره گیری از LUTهای F، G و H ایجاد کند.

۶-۴ بلوک IO

از بلوک IO برای ایجاد ارتباط با دنیای پیرامونی استفاده می‌شود. ساختار پایه بلوک IO در شکل ۶-۸ نشان داده شده است. IO را می‌توان با آرایه قابل پیکربندی برای انتقال داده‌ها به بیرون تنظیم کرد. داده منتقل شده از درگاه ورودی به PAD و بعد از گذر از بافر به CLB می‌رسد. داده می‌تواند از CLB به دنیای بیرونی از طریق بافر منتقل شود و سیگنال فعال ساز بافر می‌تواند با مالتی‌پلکسر کنترل شود.



شکل ۶-۸ ساختار پایه بلوک IO



شکل ۶-۹ ساختار پایه بلوک IO استفاده شده در تراشه Altera

در طراحی سیستم‌های عملی باید ورودی و خروجی ثباتی داشته باشیم، با این شرایط بلوک IO می‌تواند ثبات‌هایی در مسیر ورودی و خروجی داشته باشد. بلوک IO موجود در FPGA تراشه Altera در شکل ۶-۹ نشان داده شده است.

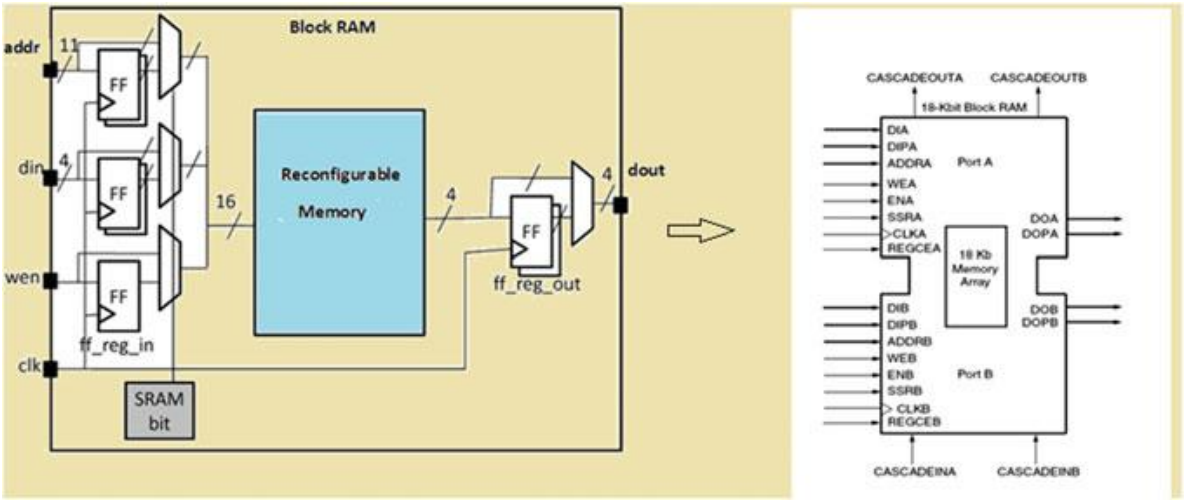
همانطوریکه در شکل ۶-۹ مشاهده می‌شود، بلوک IO از ورودی و خروجی ثباتی استفاده می‌کند. کد RTL با VHDL نشان داده شده است.

۶-۵ بلوک RAM (BRAM)

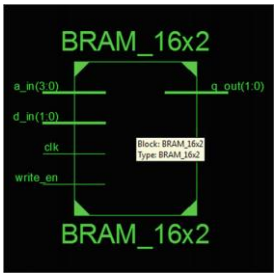
BRAM یک حافظه جاسازی شده است و یک FPGA دارای بلوک BRAM تک درگاهی و دو-درگاهی است. ساختار تراشه FPGA را ملاحظه کنید هر BRAM شامل تعدادی سلول استاتیکی RAM است. در میان آنها سلول‌های کمی برای پیکربندی حافظه استفاده می‌شود بقیه برای ذخیره‌سازی دادها استفاده می‌شوند. BRAMها بعنوان ذخیره کننده داده‌ها برای طراحی FIFO، بافرها، پشته‌ها استفاده می‌شوند همچنین برای ذخیره‌سازی در سیستم‌های FSMها استفاده می‌شوند.

هر BRAM دارای سیگنال ساعت و فعال ساز ساعت، خواندن و نوشتن می‌باشد همچنین بلوک BRAM سیستم همزمان است. اگر BRAM دو درگاهی را در نظر بگیریم هر دو درگاه قابل تعویض است ومی‌توان برای کنترل همزمانی عمل خواندن-نوشتن استفاده کرد. در تراشه‌های Spartan-3 فرکانس عملکرد BRAM ۲۰۰ مگاهرتز است. BRAM تک درگاهی و دو-درگاهی در شکل ۶-۲۵ نشان داده شده است.

همانطوریکه در شکل ۶-۱۰ مشاهده می‌کنید، BRAM شامل حافظه قابل پیکربندی، خطوط آدرس، فعال‌ساز نوشتن، سیگنال ساعت، خطوط داده ورودی و خروجی است. نتیجه سنتز BRAM ۱۶ در ۲ در شکل ۶-۱۱ نشان داده شده است.



شکل ۶-۱۰ ساختارBRAM



شکل ۶-۱۱ نتیجه سنتز BRAM

به عبارت خیلی ساده، تراشه FPGA را به صورت هر سیستم دیجیتالی می‌توان پیکربندی نمود. یک آرایه گیتی برنامه‌پذیر یک قطعه منطقی است که شامل یک آرایه دوبعدی از سلولهای منطقی قابل برمامه ریزی و کلیدهای قایل پیکربندی است سلول منطقی مبتنی بر LUT[۳] شامل یک مدار ترکیبی آرایش‌پذیر با فلیپ فلاپ می‌باشد از LUT می‌توان برای پیده سازی هر تابع ترکیبی n ورودی استفاده کرد. خروجی LUT را می‌توان در یک فلیپ فلاپ D (DFF) ذخیره کرد علاوه بر سلول منطقی قابل برنامه‌ریزی اغلب FPGA دارای بلوک‌های حافظه، ضرب‌کننده‌های ترکیبی، مدارهای مدیریت ساعت، مدارهای واسط I/O نیز هستند در FPGA های پیشرفته یک یا چند CPU نیز وجود دارد.

بلوک I/O هر پایه I/O قابل برنامه ریزی یک IOB برای سازگاری با سطح سیگنال CMOS،TTL دارد می‌توان از آن برای یک ورودی، یک خروجی یا درگاه دو جهته استفاده کرد.

۶-۶ مروری بر قطعات SPARTAN 3

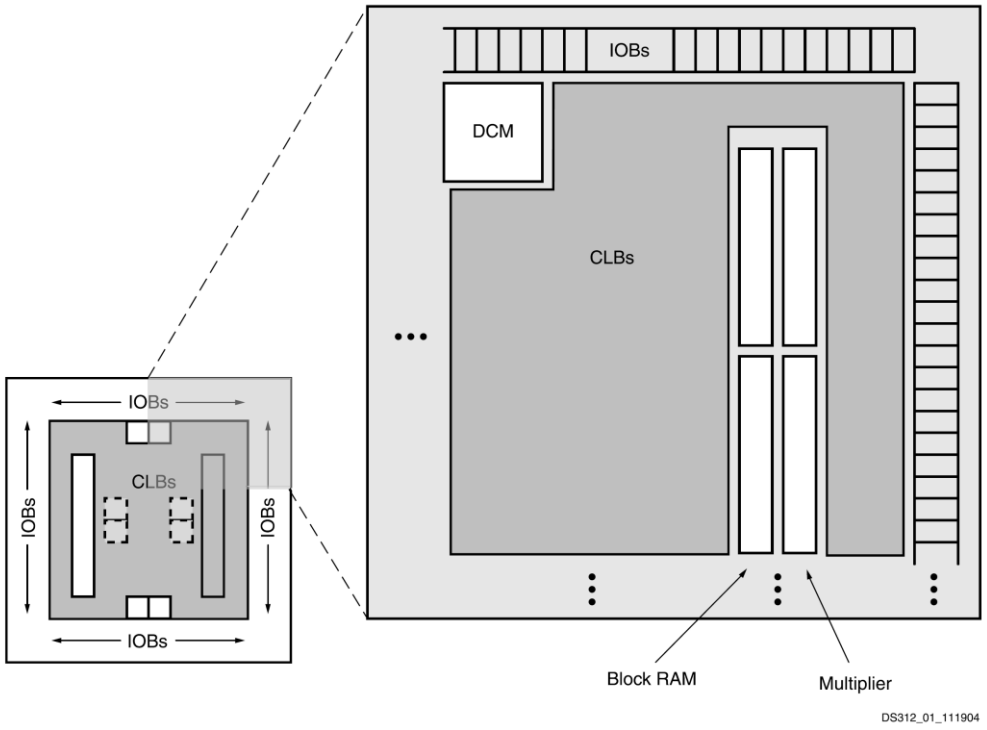
مهمترین عنصر پایه قطعه Spartan 3 یک سلول منطقی LC است که حاوی یک LUT چهار ورودی و یک فلیپ فلاپ D است. یک سلول منطقی دارای یک مدار نقلی برای پیاده سازی توابع حساب و یک مدار مالتی پلکسر است. LUT قابل آرایش بصورت یک حافظه RAM و یک شیفت رجیستر هم می‌باشد. دوسلول منطقی تشکیل یک گروه بنام Slice را می‌دهند و به چهار slice هم یک بلوک منطقی آرایش پذیر (CLB[۴]) می‌گویند.

Spartan 3 حاوی چهار نوع بلوک ماکرو شامل: ضرب کننده ترکیبی، بلوک RAM، مدیریت ساعت دیجیتال (DCM[۵]) و بلوک ورودی و خروجیIOB است.

یک DCM از یک حلقه دیجیتال تاخیری برای کاهش سرعت ساعت ، کنترل فرکانس و جابجایی فاز سیگنال ساعت استفاده می‌کند.

در جدول مشخصات چند تراشه SPARTAN 3E را نشان می دهد. تراشه مورد استفاده در برد آموزشی آزمایشگاه CX3S500E می‌باشد.

[۳] Look Up Table
[۴] Configurable Logic Block
[۵] Digital Clock Manager



Notes:

1. The XC3S1200E and XC3S1600E have two additional DCMs on both the left and right sides as indicated by the dashed lines. The XC3S100E has only one DCM at the top and one at the bottom.

شکل۶-۱۲: ساختار FPGA خانواده SPARTAN 3E

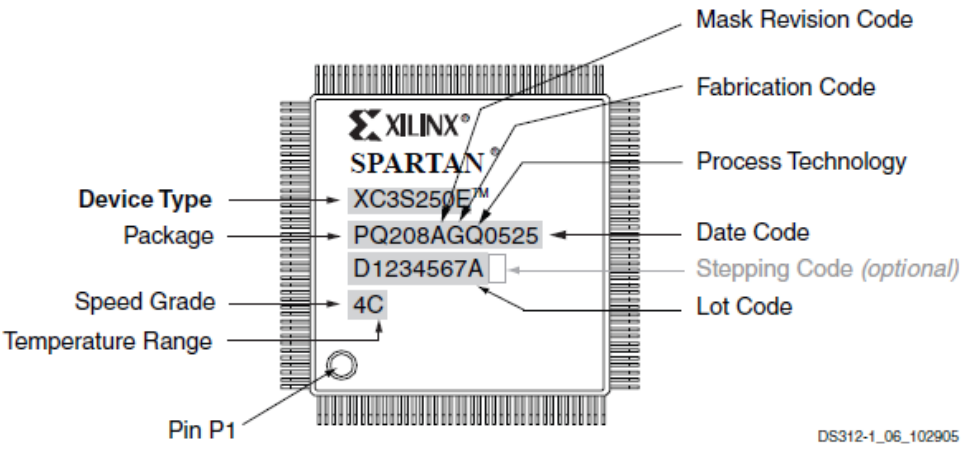
جدول ۱-۶: انواع تراشه FPGA SPARTAN 3

Device	System Gates	Equivalent Logic Cells	CLB Array (One CLB = Four Slices)				Distributed RAM bits(1)	Block RAM bits(1)	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs	Total Slices						
XC3S100E	100K	2,160	22	16	240	960	15K	72K	4	2	108	40
XC3S250E	250K	5,508	34	26	612	2,448	38K	216K	12	4	172	68
XC3S500E	500K	10,476	46	34	1,164	4,656	73K	360K	20	4	232	92
XC3S1200E	1200K	19,512	60	46	2,168	8,672	136K	504K	28	8	304	124
XC3S1600E	1600K	33,192	76	58	3,688	14,752	231K	648K	36	8	376	156

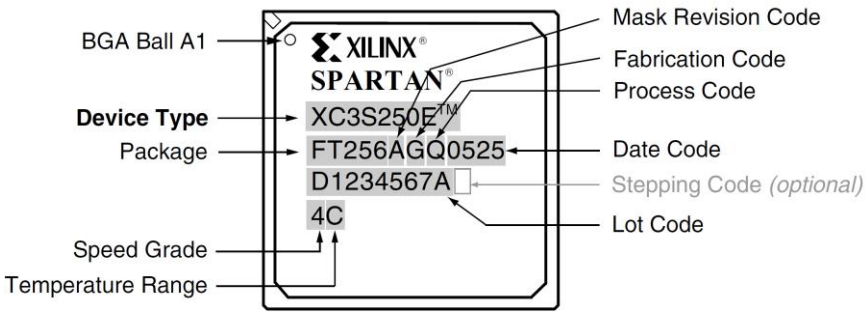
Notes:

1. By convention, one Kb is equivalent to 1,024 bits.

شکل‌های ۶-۱۲ و ۶-۱۳ معنی اعداد و علامتهای نوشته شده روی تراشه FPGA خانواده SPARTAN با نوع بسته بندی مختلف را نشان می‌دهد.

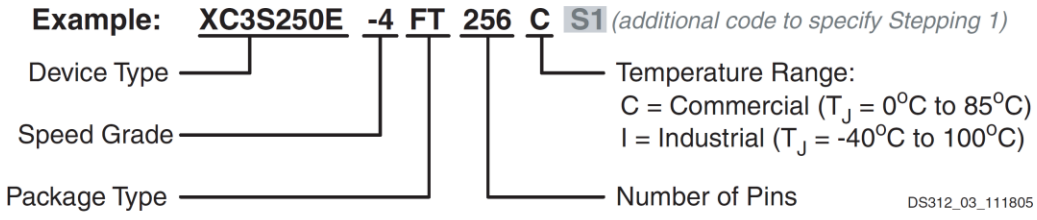


شکل ۶-۱۳: نمایش معنای اعداد و حروف روی تراشه FPGA با بسته بندی



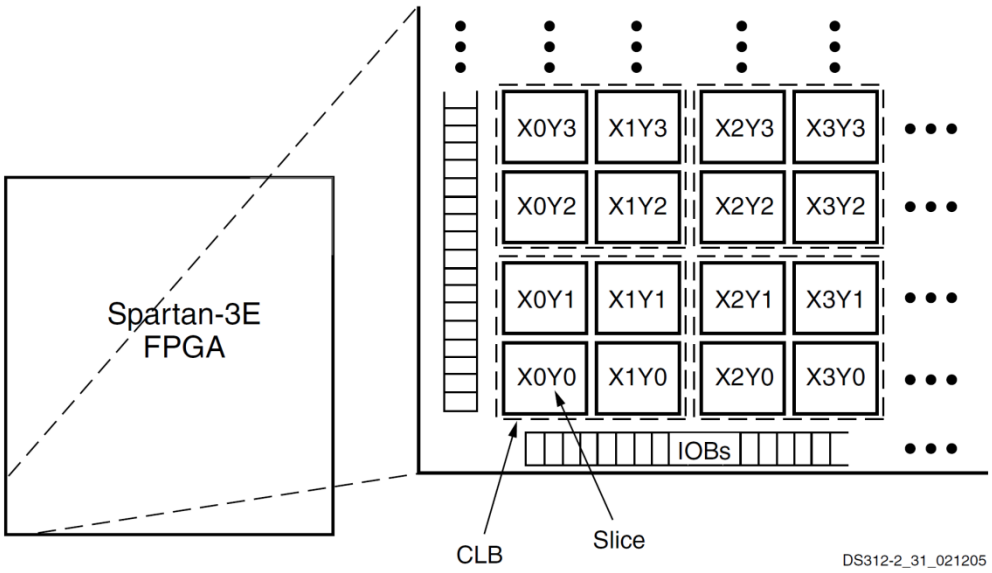
شکل ۶-۱۴: نمایش معنای اعداد و حروف روی تراشه FPGA با بسته بندی

با توجه به شکل‌های ۶-۱۳ و ۶-۱۴ می‌توان مشخصات تراشه FPGA را بدست آورد: بطور مثال نوع تراشه، سرعت کار تراشه، نوع بسته بندی، دمای کار و غیره.



شکل۶-۱۵: مفهوم اعداد و علامت‌های روی تراشه FPAG

در شکل۶-۱۶ موقعیت CLBها و Slice و LUT ها نشان داده شده است. همچنین در جدول ۶-۲ تعداد CLBها و sliceها و LUTها را برای تراشه‌های مختلف خانوادهSPARTAN 3E نشان داده شده است.



شکل ۶-۱۶: نمایش موقعیت CLBها و Sliceها در FPGA

جدول۶-۳: تعداد CLB ها وSlice ها در تراشه‌های مختلف خانواده SPARTAN3E

Device	CLB Rows	CLB Columns	CLB Total(1)	Slices	LUTs / Flip-Flops	Equivalent Logic Cells	RAM16 / SRL16	Distributed RAM Bits
XC3S100E	22	16	240	960	1,920	2,160	960	15,360
XC3S250E	34	26	612	2,448	4,896	5,508	2,448	39,168
XC3S500E	46	34	1,164	4,656	9,312	10,476	4,656	74,496
XC3S1200E	60	46	2,168	8,672	17,344	19,512	8,672	138,752
XC3S1600E	76	58	3,688	14,752	29,504	33,192	14,752	236,032

Notes:

1. The number of CLBs is less than the multiple of the rows and columns because the block RAM/multiplier blocks and the DCMs are embedded in the array (see Figure 1 in Module 1).

۶-۷ طریقه پیکر بندی FPGA

برای آشنایی با مراحل پیکربندی برد FPGA برای نمونه از یک برنامه ساده VHDL استفاده می‌کنیم. مراحل انجام پیکربندی FPGA مطابق مراحل زیر است:

در محیط نرم‌افزار ISE پروژه‌ای بانام LED_ON_OFF در مسیر c:\ISE_project\test7 ایجاد کنید مطابق آنچه که در آزمایش ۵ گفته شده است، کد زیر را در فایل VHDL با نام LED_ON_OFF ذخیره کنید.

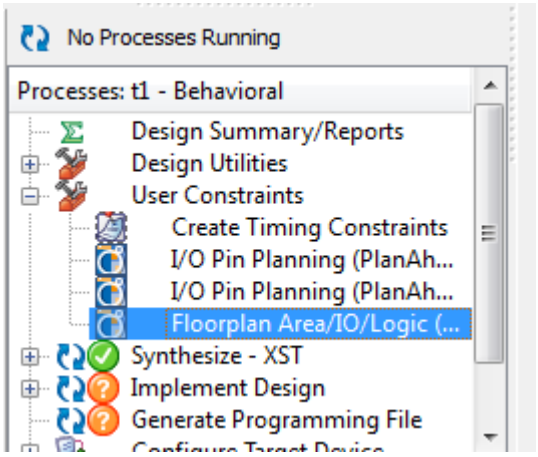
```
Library ieee;

Use ieee.std_logic_11_64.all;
-----------------------------
Entity led_on_off is
Port (
      X : in std_logic;
      Y,xbar: out std_logic);
End led_on_off;
-------------------------------------
Architecture arch_led of led_on_off is
Begin
Y<=not x;
Xbar<=not x;
End arch_led;
-------------------------------------
```

شکل۶-۱۷: کد VHDL برای روشن خاموش کردن یک LED

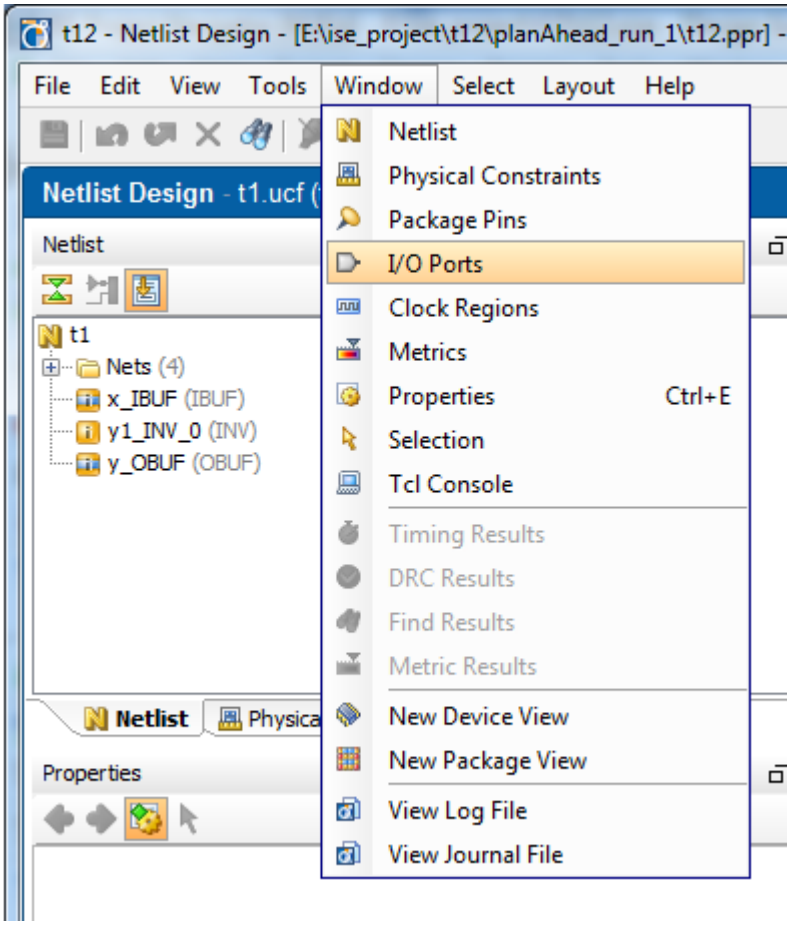
۱- در این مرحله کد نوشته شده را دوباره ذخیره کنید و سپس گزینه Synthesize را بزنید تا کد سنتز شود.

۲- از بخش Processes گزینه user constrain> Floorplan Area/IO/Logic (plan Ahead) را انتخاب کنید (مطابق شکل۶-۱۸). در این قسمت نرم‌افزار Plan Ahead باز می‌شود.



شکل۶-۱۸: نمایش اجرای برنامه Plan Ahead

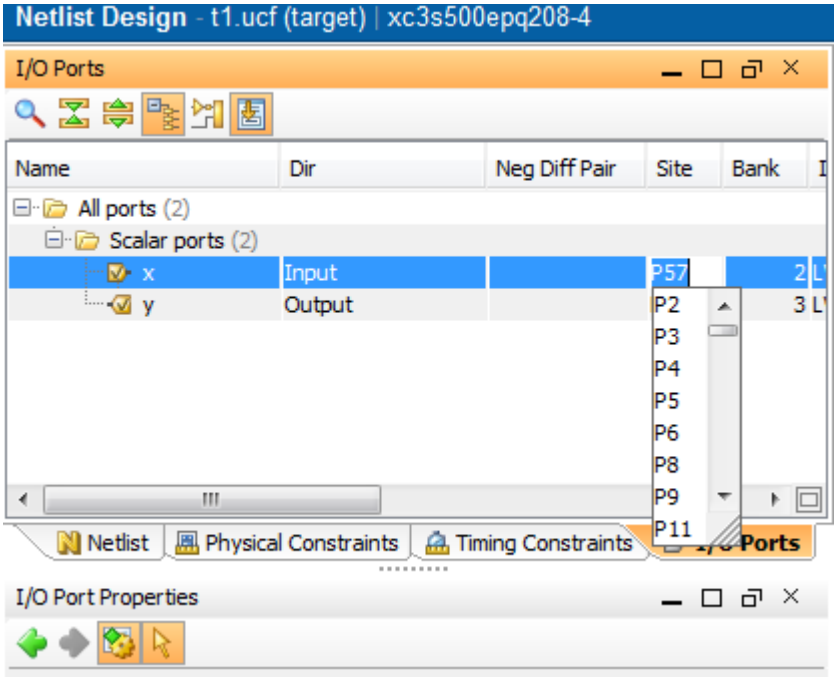
۳- از منوی Window> I/O Portsرا انتخاب کنید. (مطابق شکل۶-۱۹)



شکل ۶-۱۹: نمایش چگونگی فعال کردن بخش I/O برای تعیین پایه‌های FPGA

با توجه به اینکه در برد آموزشی FPGA آزمایشگاه کلید s1 به پایه P57 تراشه FPGA وصل می‌باشد و LED به پایه P22 وصل می‌باشد در قسمت I/O Ports تنظیمات را مطابق شکل ۶-۲۰ انجام دهید. در این قسمت برای درگاه x پایه P57 و برای پورت خروجی y از قسمت site P22 را انتخاب کنید و همچنین برای xbar در قسمت P119 site را انتخاب کنید این یعنی s1 همان x و LED2 همان y کد VHDL می‌باشد.

توجه کنید که پایه p119 به بوق وصل شده است.



شکل ۶-۲۰: چگونگی تعیین پایه‌های FPGA برای ورودی و خروجی کد VHDL مربوطه

توجه کنید: که مشخصات کلیدها و LED های متصل به پایه‌های برد تراشه FPGA در جدول‌های نشان داده شده است.

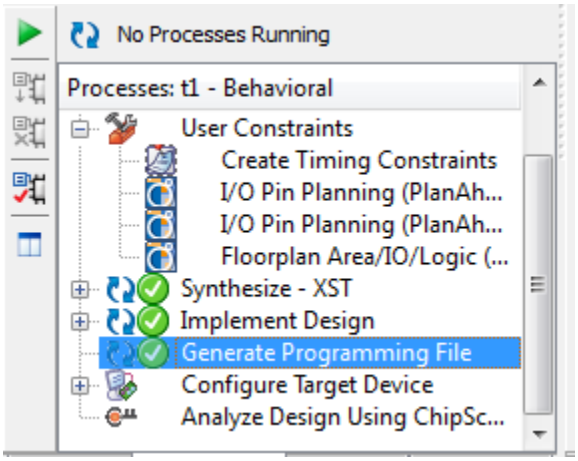
جدول۶-۳: پایه‌های FPGA و کلیدهای فشاری

Switchs	S1	S2	S3	S4	S5	S6	S7
FPGA Pins	P57	P58	P71	P72	P91	P101	P110

جدول۶-۴: مربوط به پایه‌های LEDها و پایه‌های FPGA

LED	LED2	LED3	LED4	LED5	LED6	LED7
FPGA Pins	P22	P23	P24	P25	P28	P29

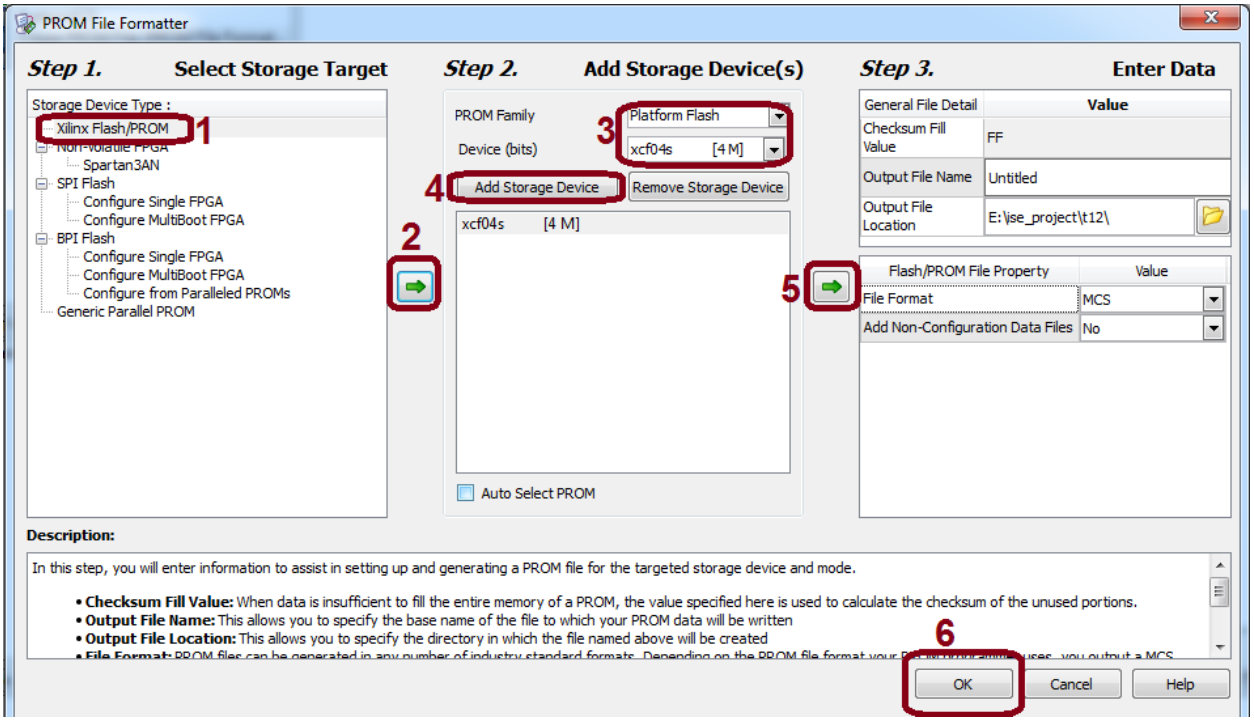
در این بخش، نرم‌افزار Plan Ahead را ببندید. در محیط نرام افزار ISE گزینه Generate Programming File را اجرا کنید. در صورتیکه عملیات درست انجام شود علامت تیک سبز رنگ مطابق شکل ۶-۲۱ ظاهر می‌شود.



شکل ۶-۲۱: نمایش مرحله تولید فایل‌های پیکربندی FPGA

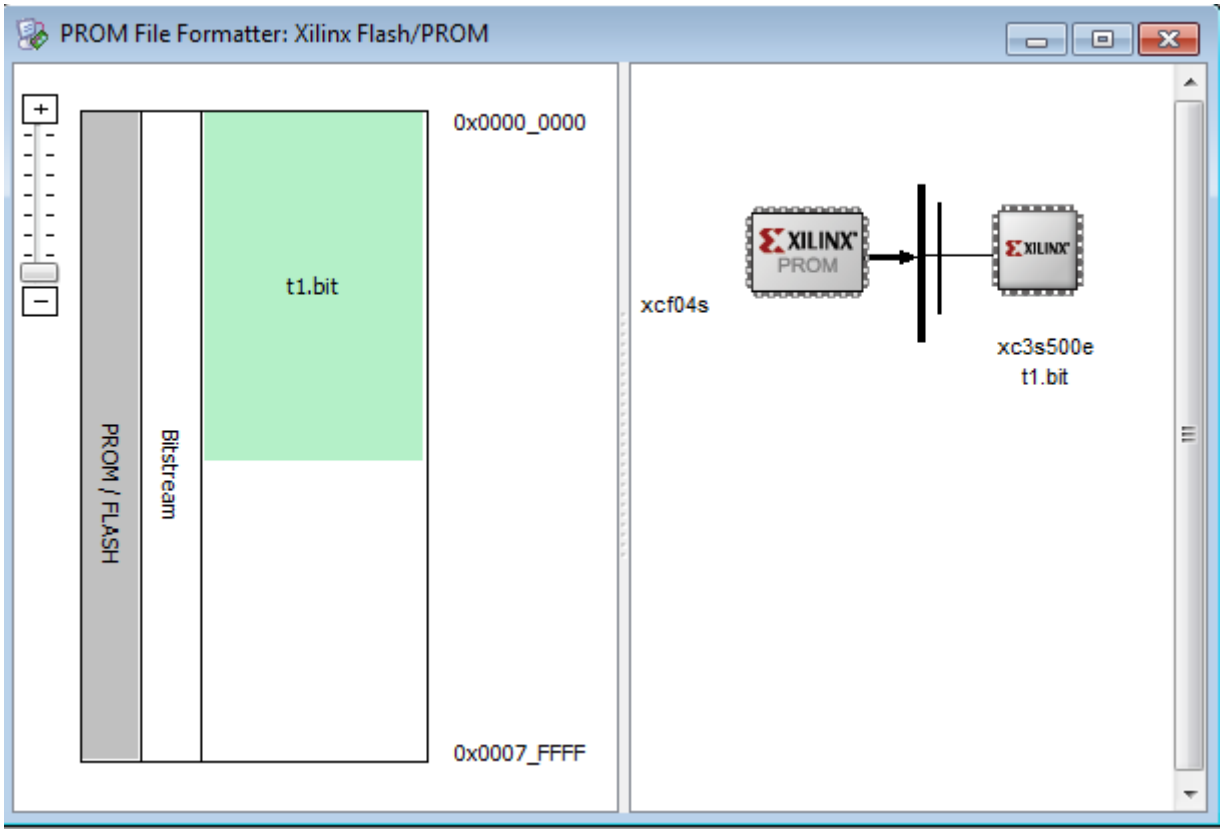
باید فایل پروگرام FPGA ایجاد شده با پسوندbit. را به فایل پروگرام PROM با پسوند mcs. تبدیل کرد بنابراین در نرم افزار iMPACT رو گزینه Create PROM File Formatter دوبار کلیک کنید. مطابق شکل ۶-۲۲ گزینه‌ها را انتخاب کنید.

توجه کنید که در Step 3 شکل ۶- ۲۲ مسیر فایل را C:\ISE_project\test7 و نام فایل را led_on_off انتخاب کنید.



شکل ۶-۲۲:مراحل تعیین مشخصات و محل ذخیره سازی و نام فایل mcs. برای پیکر بندی حافظه مورد نظر

۴- Add Device را انتخاب کنید.



شکل ۶-۲۳: نمایش فایل انتخابی با پسوند bit. برای ایجاد فایل با پسوند mcs. نوع حافظه انتخابی

در این شرایط iMPACT نمودار اتصال را رسم می‌کند در این مرحله از قسمت iMPACT Processes گزینه Generate File را دوبار کلیک کنید. در صورت موفقیت آمیز بودن مراحل، فایل با پسوند mcs. ایجاد می‌شود و پیغام موفقیت عملیات مطابق شکل ظاهر می‌شود.

شکل ۶-۲۴: پیغام موفقیت‌آمیز بودن تولید فایل با پسوند mcs.

آزمایش ۷

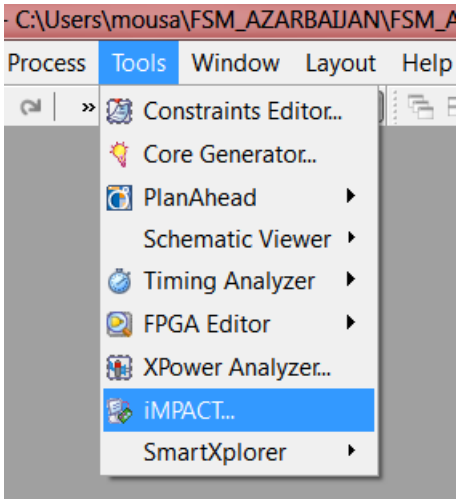
پیکربندی FPGA

۷-۱ مراحل پیکر بندی FPGA

برد آموزشی FPGA را روشن کنید. مطمئن باشید که LED POWER برد روشن پورت برنامه ریزی آن به کامپیوتر وصل است.

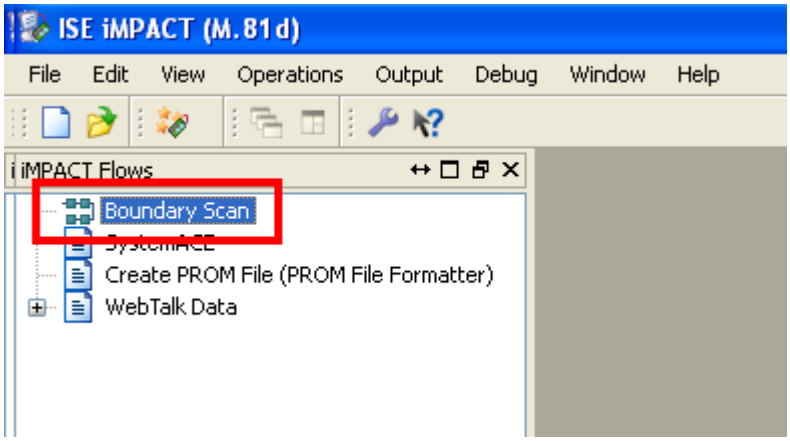
نرم افزار ISE را اجرا کنید.

از برنامه ISE منوی Tools گزینه iMPACT را انتخاب و اجرا کنید.

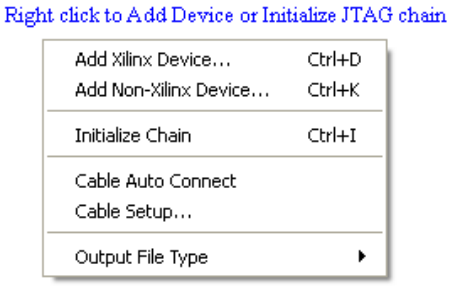


شکل ۷-۱: مراحل اجرای نرم‌افزار iMPACT

از پنجره ظاهر شده (ISE iMPACT) در بخش iMPACT Flows روی Boundary Scan کلیک کنید (شکل ۷-۲) سپس در قسمت راست پنجره در جای خالی کلیک راست کرده و گزینه Initialize Chain را انتخاب کنید (مطابق شکل ۷-۳) . در صورتیکه برد به تغذیه وصل باشد آی سی مربوطه شناسایی شده و نشان داده می شود، در ادامه تمام پنجره‌های بعدی را Cancel کنید.

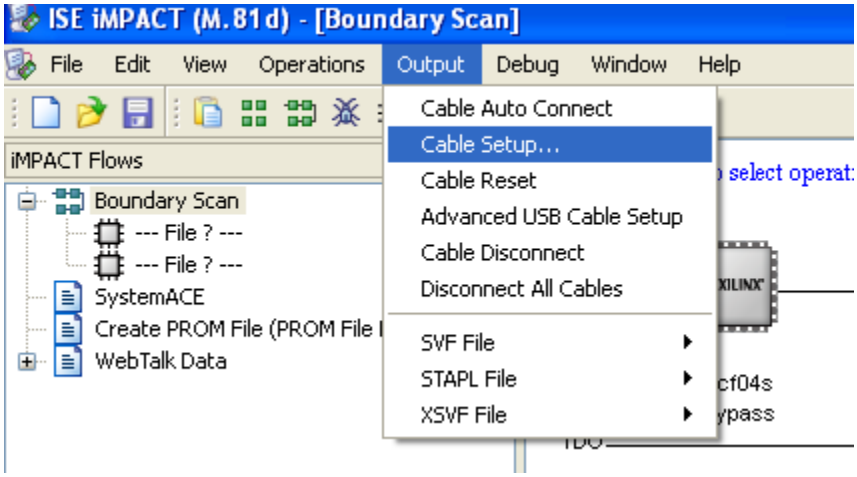


شکل ۷-۲



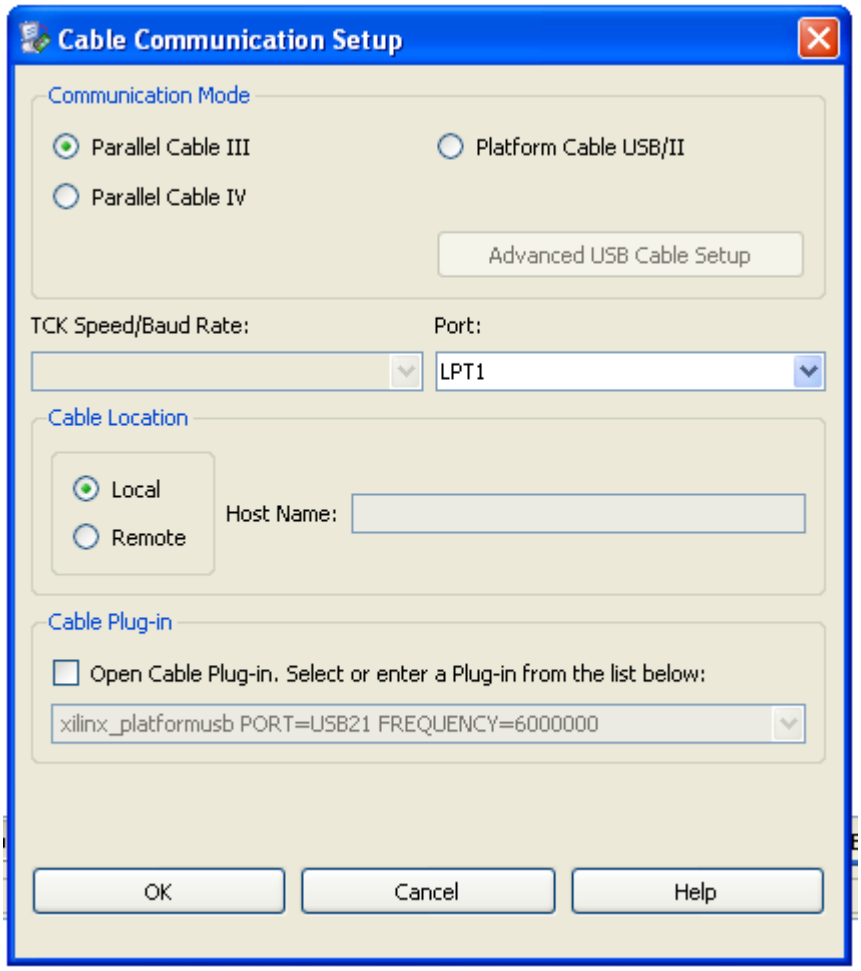
شکل ۷-۳

از منوی output گزینه Cable setup را انتخاب کنید.



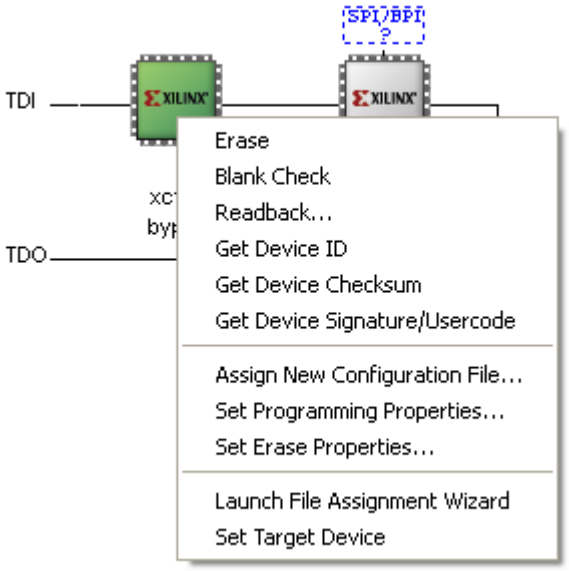
شکل ۷-۴

تنظیمات پنجره را مطابق شکل انجام دهید و در نهایت کلید Ok را بزنید.



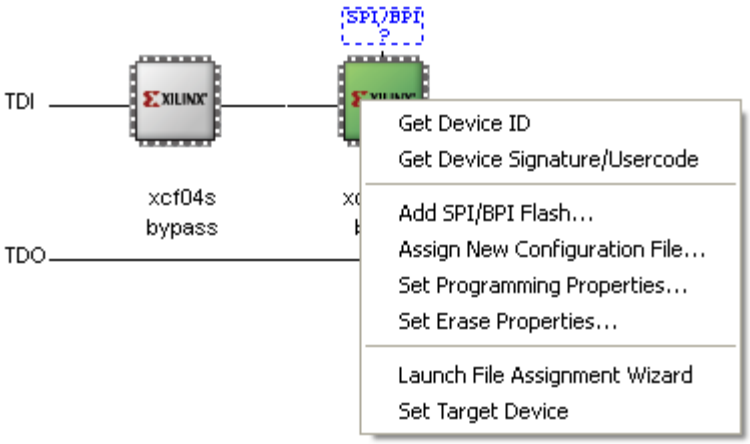
شکل ۷-۵: تنظیمات کابل

مطابق شکل ۷-۶ روی تراشه حافظه (Xcf04s) کلیک راست کرده و گزینه (Assign New Configuration File) ظاهر می‌شود این پنجره از شما فایل مورد نظر را با پسوند *.mcs در خواست می‌کند، فایل مورد نظر خودتان را وارد کنید. در این قسمت برای مثال فایل led_on_off.mcs که در مسیر c:\ISE_Project\test7 قرار دارد را انتخاب کنید. (این فایل را در آزمایش ششم بخش ۶-۲ بند ۷ ایجاد کردیم).



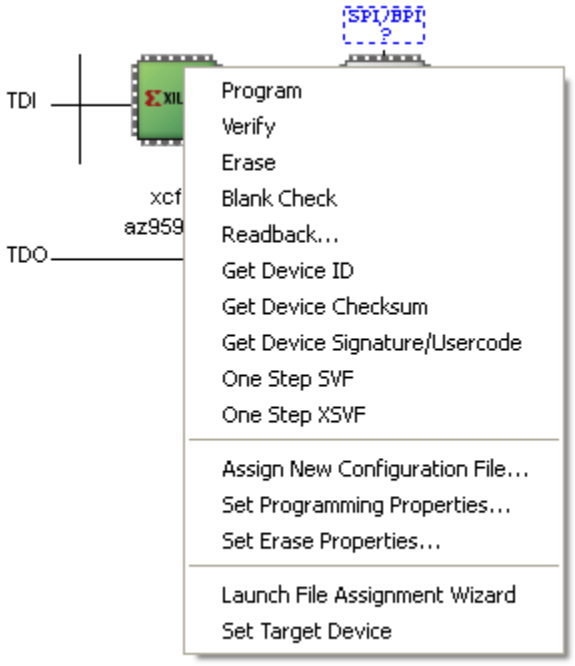
شکل ۷-۶: انتخاب تراشه حافظه برای برنامه ریزی

برای پیکر بندی تراشه FPGA مطابق مرحله بالا و شکل ۷-۷ روی تراشه FPGA کلیک راست کرده و فایل مورد نظر خودتان را با پسوند *.bit وارد کنید.



شکل ۷-۷: پیکربندی FPGA

برای برنامه ریزی حافظه روی آن کلیک راست کرده و گزینه Program را انتخاب کنید با این کار سیستم شروع به برنامه ریزی تراشه حافظه می‌کند در صورت موفقیت آمیز بودن برنامه ریزی پیغام شکل ظاهر می‌شود (شکل ۷–۸).



شکل ۷–۸: برنامه ریزی حافظه

روش فوق را برای پیکر بندی تراشه FPGA می‌تواند انجام داد. با این تفاوت که روی تراشه FPGA باید انجام شود.

در صورتیکه تمام مراحل بدون خطا باشند بعد از برنامه ریزی پیغام موفق آمیز بودن برنامه ریزی ظاهر خواهد شد (مطابق شکل ۷–۹).

شکل ۷–۹

تمرین۷–۱: کد VHDL تمام جمع کننده را نوشته و بعد از سنتز روی برد آموزشی FPGA پیکر بندی کرده و آزمایش کنید.

آزمایش ۸

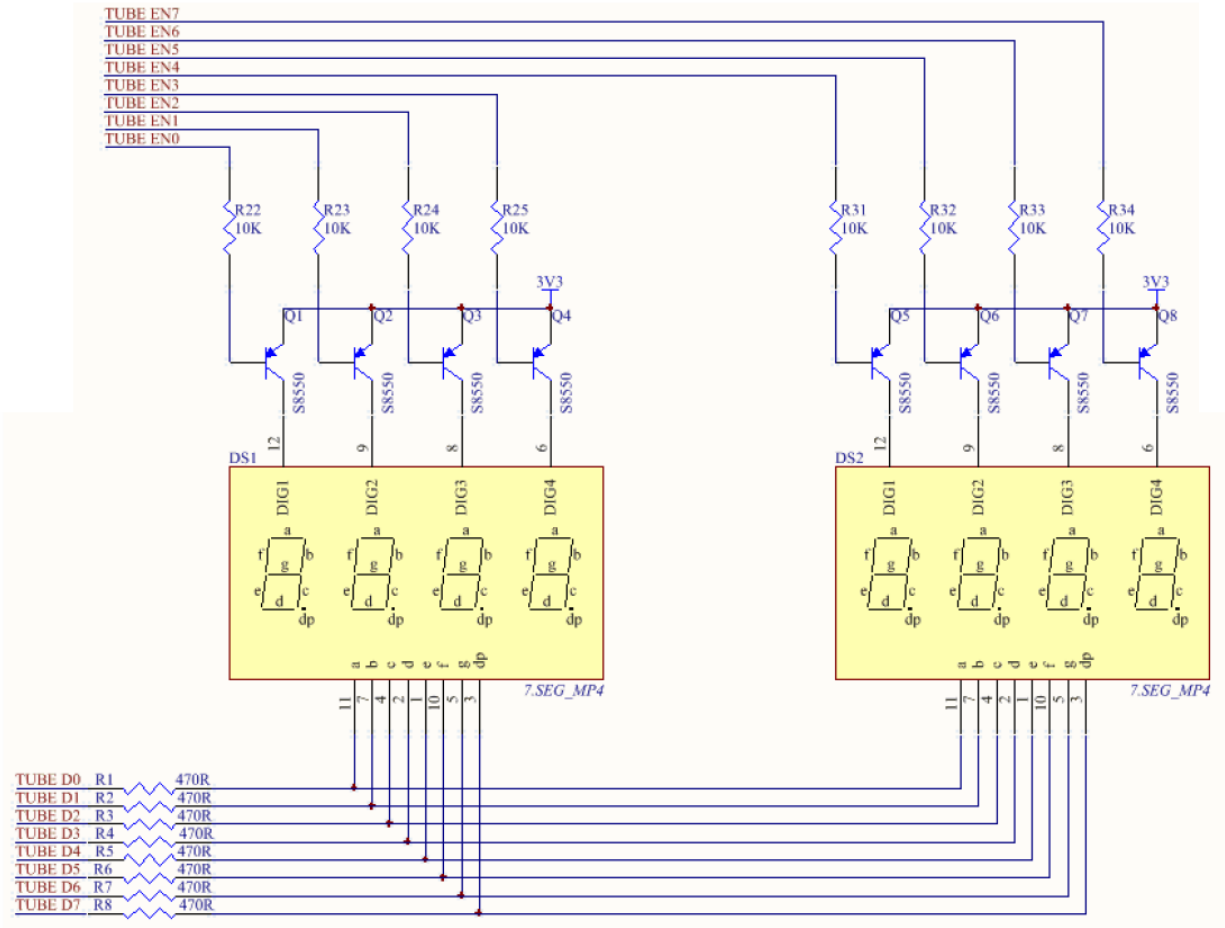
طراحی شمارنده و نمایش روی 7-segment

مقدمه

در روی برد آموزشی FPGA ۸ تا 7-segment وجود دارد که پایه‌های مربوط به FPGA و پایه‌های 7-segment های مربوط در جدول ۸–۱ ذکر شده است، همچنین طریقه اتصالات مربوط به 7-segment در شکل ۸–۱ نشان داده شده است با استفاده از کد شکل ۸–۲ FPGA را بصورت دلخواه پیکر بندی کنید و برد آموزشی را برنامه ریزی کنید.

جدول ۸–۱

FPGA Pin	نام سیگنال
P102	TUBE_D0
P99	TUBE_D1
P107	TUBE_D2
P109	TUBE_D3
P112	TUBE_D4
P100	TUBE_D5
P106	TUBE_D6
P108	TUBE_D7
P132	TUBE_EN0
P129	TUBE_EN1
P128	TUBE_EN2
P127	TUBE_EN3
P126	TUBE_EN4
P123	TUBE_EN5
P122	TUBE_EN6
P120	TUBE_EN7



شکل ۸-۲: اتصالات مربوط به 7-segmentها

کد زیر را در محیط نرم‌افزار ISE نوشته و آی سی FPGAرا با این کد پیکر بندی کنید. فایلهای مربوط برای پیکر بندی حافظه و FPGA را ایجاد کنید.

```
1  library ieee;
2  use ieee.std_logic_1164.all ;
3  ---------------------------------------------
4  entity display is
5   port (
6     input: in std_logic_vector( 3 downto 0);
7     sel : in std_logic_vector( 1 downto 0);
8     an: out std_logic_vector (3 downto 0 ) ;
9     s_seg : out std_logic_vector ( 7 downto 0 )
10 );
11 end display ;
12 ------------------------------------------------
13 Architecture arch_display of display is
14 Begin
15  process (input)
16    begin
17    case input is
18     when "0000" =>
19       s_seg<= "00000011";
20     when "0001" =>
21       s_seg<= "10011111";
22     when "0010" =>
23       s_seg<= "00100101";
24     when "0011" =>
25       s_seg<= "00001101";
26     when "0100" =>
27       s_seg<= "10011001";
28     when "0101" =>
29       s_seg<= "01001001";
30     when "0110" =>
31       s_seg<= "01000001";
32     when "0111" =>
33       s_seg<= "00011111";
34     when "1000" =>
35       s_seg<= "00000001";
```

```
36     when "1001" =>
37       s_seg<= "00001001";
38     when others =>
39       s_seg<="11111111";
40     end case;
41   end process;
42   ----------------------------------------
43  process (sel )
44   begin
45     case sel is
46       when "00" =>
47         an <= "1110";
48       when "01" =>
49         an <= "1101";
50       when "10" =>
51         an <= "1011";
52       when others =>
53         an <= "0111";
54      end case ;
55    end process ;
56  end arch_display;
```

این کد دارای دو ورودی می باشد ورودی input که ۴ بیتی است اعداد ۰ تا ۹ را شامل می‌شود و ورودی sel برای انتخاب نمایشگر مورد نظر می‌باشد. کلیدهای برد آموزشی را طوری تنظیم کنید که بتوانید اعداد را بصورت دلخواه به ورودی مدارتان اعمال کنید و با انتخاب یکی از حالت‌های 00، 10،01 و 11 می توانید 7_segment مربوطه را فعال کنید تا عدد مورد نظر شما در آن نمایش داده شود.

سوال۸-۱: از کدام پین‌های آی سی FPGA برای اعمال ورودی (input) و انتخابگر نمایشگر ها (sel) استفاده کردید؟

سوال۸-۲ : به ازای انتخاب اعداد ۱۰ تا ۱۵ چه عددی در خروجی نمایش داده می شود؟

تمرین ۸-۱: کد را طوری تغییر دهید که به ازای اعداد ۱۰ تا ۱۵ نمایشگر حرف E را نمایش دهد.