دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

دستورکار آزمایشگاه معماری کامپیوتر

دکتر حمیدرضا زرندی
با همکاری مدرسین آزمایشگاه
تابستان ۱۳۹۷

بسمه تعالی

درس معماری کامپیوتر، همنیاز این درس است، چنانچه دانشجویی چندین ترم پیش، این درس را گذرانده است، لازم است مجدداً این درس را بازنگری کرده و مطالب آن را به یاد آورد. همچنین، در این آزمایشگاه، از زبان توصیف سخت‌افزار VHDL برای توصیف، شبیه‌سازی، سنتز و در نهایت پیاده‌سازی بر روی بردهای FPGA استفاده خواهد شد، در نتیجه لازم است دانشجویان نسبت به این زبان توصیف یا زبانی مشابه همچون Verilog یا SystemC آشنا باشند. **هر آزمایش باید در سطح تجرید خواسته شده (گیت، RTL) طراحی و پیاده‌سازی شود.**

- **منظور از توصیف در سطح گیت، توصیفی است که تنها از گیت‌های منطقی استفاده شده باشد.**
- **منظور از توصیف در سطح RTL، توصیفی است که از Decoder, Encoder, MUX, DeMux, Register, Bus, Buffer استفاده شده باشد.**

- تعداد آزمایش‌های که در طول ترم انجام می‌شود، در جلسه اول، توسط مدرس آزمایشگاه تعیین خواهد شد، لازم است که دانشجویان در قالب اعضای دو نفری (و در موارد خاص با تایید مدرس، سه نفری) تشکیل گروه داده و به مدرس آزمایشگاه اطلاع دهند.

- توصیه می‌شود مدرس آزمایشگاه در ابتدای ترم یک ساختار استاندارد برای کامپیوتر پایه مورد نظر خود (عرض بیتی گذرگاه‌ها، تعداد و عرض بیتی ثبات‌ها، سایز حافظه‌ها، عملیات ALU) قرارداد نماید و بر مبنای آن آزمایشات مورد نظر خود را طراحی نماید. به عنوان مثال، در آزمایش جمع‌کننده‌ها و یا ضرب کننده‌ها، ورودی‌های ماژول بر مبنای عرض بیتی ثبات‌ها تنظیم شود.

- زمان اتمام هر آزمایش، توسط مدرس آزمایشگاه با توجه به محتوای آزمایش مشخص خواهد شد و دانشجویان قبل از شروع هر آزمایش نسبت به مهلت انجام آن اطلاع کسب کنند.

- هر گروه، قبل از شروع آزمایش لازم است، پیش‌گزارشی تهیه کرده و قبل از شروع کلاس تحویل مدرس آزمایشگاه دهد. مدرس آزمایشگاه قبل از شروع هر کلاس ممکن است پرسش‌های شفاهی یا کتبی نسبت به آزمایش مورد نظر مطرح نماید و دانشجویان موظف هستند که به آن‌ها پاسخ کامل و صحیح دهند.

- از آنجایی که شروع کلاس‌های آزمایشگاه پس از زمان حذف و اضافه است، تعداد جلسات برگزار شده کمتر بوده و لذا حضور در کلیه جلسات الزامی است و تنها یک جلسه غیبت مجاز خواهد بود. همچنین از ورود افراد بیش از ۱۰ دقیقه تاخیر ممانعت به عمل خواهدآمد.

- نمره دهی نهایی بر اساس موارد زیر انجام خواهد شد؛ (مدرسین آزمایشگاه در صورت لزوم می‌توانند تغییراتی ایجاد نمایند):

پیش‌گزارش‌های تحویل داده شده	مجموع آزمایش‌ها حدود ۱۰ درصد
نمره پرسش‌های شفاهی/کتبی قبل از شروع هر آزمایش	مجموع آزمایش‌ها حدود ۱۵ درصد
انجام کامل هر آزمایش	مجموع آزمایش‌ها حدود ۳۰ درصد
کیفیت انجام هر آزمایش و پیاده‌سازی آن	مجموع آزمایش‌ها حدود ۳۰ درصد
حضور فعال، موثر در گروه همکاری با مدرس (کار کلاسی)	حدود ۱۵ درصد
موارد دیگر (با صلاحدید مدرس آزمایشگاه)	به انتخاب مدرس آزمایشگاه

- انجام کلیه آزمایش‌ها برای هر نوع داده‌ای زیر امکان‌پذیر است، قبل از شروع این مجموعه آزمایش‌ها، مدرس انتخاب نوع نمایش اعداد را برای شما مشخص خواهد ساخت و کلیه دانشجویان موظف به انجام آزمایش‌ها براساس همان نوع نمایش اعداد هستند.

نوع نمایش اعداد		توضیحات
بی علامت	Unsigned	مجموعه اعداد طبیعی
مکمل دو	Two’s Complement	اعداد صحیح (مثبت و منفی) در منطق مکمل ۲
اندازه علامت	Sign Magnitude	علامت و اندازه عدد بصورت جداگانه ذخیره می‌گردد
مکمل ۱	One’s Complement	اعداد صحیح (مثبت و منفی) در منطق مکمل ۱
ممیز ثابت	Fixed Point	اعداد حقیقی (مثبت و منفی) متناهی با تعداد بیت اعشار ثابت
ممیز شناور	Floating Pint	اعداد حقیقی (مثبت و منفی) متناهی با تعداد بیت اعشار شناور

- شبیه‌سازی‌ها توسط یکی از دو شبیه‌ساز ISim و یا ModelSim انجام خواهد گرفت.

فهرست آزمایش‌ها

شماره آزمایش	موضوع	عنوان آزمایش	صفحه
۱	VHDL مقدماتی	توصیف مدارهای پایه آزمایش‌ها و انجام شبیه‌سازی	۵
۲	مدارهای پایه	طراحی مدارهای اولیه (Decoder, Encoder, Multiplexer و Comprator) ۴ بیتی	۱۱
۳	پیاده‌سازی مدار ترتیبی	Processها و چگونگی پیاده‌سازی شمارنده‌ها	۱۲
۴	جمع کننده‌ها	پیاده‌سازی انواع جمع کننده‌ها و تحلیل سرعت آنها	۱۵
۵	ضرب کننده‌ها	پیاده‌سازی انواع ضرب‌کننده ۴ بیتی	۱۶
۶	تاخیر در مدارهای دیجیتال	ایجاد تاخیر مشخص در مدارهای دیجیتال	۱۹
۷	حافظه‌ها	طراحی حافظه ROM, RAM و CAM	۲۱
۸	تقسیم‌کننده‌ها	تقسیم‌کننده دودویی در منطق مکمل ۲	۲۳
۹	کامپیوتر پایه(۱)	پیاده‌سازی یک کامپیوتر پایه	۲۵
۹	کامپیوتر پایه(۲)	پیاده‌سازی یک کامپیوتر پایه	۲۸
۱۰	طراحی واحد کنترل	طراحی واحد کنترل سیم بندی شده(Hard wired Control Unit)	۳۴
۱۱	خط لوله	پیاده‌سازی خط لوله و بررسی تسریع آن	۳۶
۱۲	ورودی/خروجی	پیاده‌سازی ارسال/دریافت ناهمگام در گذرگاه‌های یکطرفه و دوطرفه	۳۸
۱۳	ورودی/خروجی	پیاده‌سازی مکانیزم‌های داوری گذرگاه	۴۱
۱۴	محاسبات ممیز شناور	پیاده‌سازی عملیات جمع یا تفریق واحد ممیز شناور	۴۴
۱۵	ارزیابی کارآیی	اندازه‌گیری پارامترهای IPC، CPI، MIPS و DMIPS یک پردازنده نوعی	۴۵
۱۶	حافظه‌ها	طراحی سلسله مراتب حافظه	۴۷

فهرست ضمایم

شماره ضمیمه	عنوان	صفحه
۱	توصیف سیستم با VHDL	۴۹
۲	شبیه‌سازی با استفاده از ModelSim	۵۲
۳	نحوه کارکردن با بورد FPGA	۶۱

موضوع: VHDL مقدماتی

شماره آزمایش: ۱

عنوان:

توصیف مدارهای پایه آزمایش‌ها و انجام شبیه‌سازی

هدف:

آشنایی با نحوه مدل کردن مدارهای پایه آزمایش‌ها در زبان VHDL و معرفی ساختارهای همروند

آمادگی پیش از آزمایش:

شرح آزمایش:

توصیف سخت‌افزار در VHDL دارای دو بخش است: توصیف پورت‌های ورودی و خروجی و توصیف معماری سخت‌افزار. به عنوان مثال، توصیف گیت And در مثال زیر ذکر شده است:

توصیف گیت And

```
library IEEE;
use IEEE.std_logic_1164.all;

Entity and_gate is
   Port(
      A, B: in std_logic;
      C   : out std_logic
   );
End entity and_gate;

Architecture behavioral of and_gate is
Begin
   C <= A and B;
End behavioral;
```

در توصیف سخت‌افزار می‌توان از عملگرهای and, or, xor, not استفاده کرد. این عملگرها مثل گیت‌های ساده منطقی عمل می‌کنند. عملگر مقداردهی به سیگنال <= است. در مثال بالا std_logic می‌تواند یک مقدار منطقی داشته باشد. در صورتی که نیاز به تعریف پورتی بیش از یک بیت باشد، می‌توان از std_logic_vector استفاده کرد. به مثال زیر توجه نمایید:

توصیف گیت And به صورت برداری

```
library IEEE;
use IEEE.std_logic_1164.all;

Entity and_gate_vector is
   Port(
      A, B: in std_logic_vector(3 downto 0);
      C   : out std_logic_vector(3 downto 0)
   );
End entity and_gate_vector;
```

```
Architecture behavioral of and_gate_vector is
   Signal internal_signal   : std_logic_vector(3 downto 0);
Begin
   internal_signal <= A and B;
   C <= internal_signal;
End behavioral;
```

توجه شود که در مثال بالا خروجی ۴ بیتی است و هر بیت آن با AND دو مقدار متناظر در A و B بدست آمده است. علاوه بر آن به تعریف سیگنال در داخل architecture توجه شود. به چند روش می‌توان به سیگنال‌های از نوع std_logic_vector مقدار داد. در جدول زیر چند نمونه ذکر شده است:

مقدار دهی به std_logic_vector

```
   Port(
      A: in  std_logic_vector(9 downto 0);
      C: out std_logic_vector (9 doento 0)
   );
---

Signal b: std_logic_vector(9 downto 0);
---

B(9 downto 1) <= B(8 downto 0); -- shift
B(0) <= '1';
B(2 downto 0) <= "110";
B <= A;
B(9 downto 8) <= A(2 downto 1);
B(4) <= B(5);
B <= (others => '0');
B(5 downto 0) <= A(9 downto 7) & "011";
```

استفاده از when-else و with-select

```
Signal q: std_logic_vector(3 downto 0);
Signal t: std_logic_vector(3 downto 0);

Q<= "0000" when t="1010" else
      "1011" when t="0010" else
      "1111";

with t select
q<= "0000" when "1010",
     "1011" when "0010",
     "1111" when others;
```

معرفی Component و نحوه نمونه‌سازی از componentها و تأخیر برای testbench

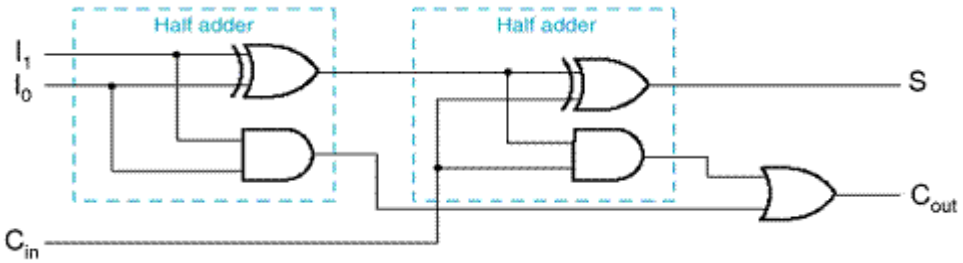
```
library ieee;
use ieee.std_logic_1164.all;
entity Decoder_vec is
  port(
```

```
    input: in std_logic_vector(2 downto 0);
    output: out  std_logic_vector(7 downto 0)
  );
end entity Decoder_vec;
architecture behav of Decoder_vec is
 begin
        output(0) <= '1' when input="000" else '0';
        output(1) <= '1' when input="001" else '0';
        output(2) <= '1' when input="010" else '0';
        output(3) <= '1' when input="011" else '0';
        output(4) <= '1' when input="100" else '0';
        output(5) <= '1' when input="101" else '0';
        output(6) <= '1' when input="110" else '0';
        output(7) <= '1' when input="111" else '0';
 end behav;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity decoder_tb is
end entity decoder_tb;

architecture test of decoder_tb is
        component Decoder_vec is
           port(
                 input: in std_logic_vector(2 downto 0);
                 output: out  std_logic_vector(7 downto 0)
                );
        end component;
        signal i :std_logic_vector(2 downto 0);
        signal o : std_logic_vector(7 downto 0);
begin
  dec:Decoder_vec port map ( input => i, ouput => o);
  i <= "000", "001" after 100 ns, "101" after 200 ns;
end test;
```

طراحی ساختاری

در این روش طراحی، هر موجودیت از چند کامپوننت تشکیل شده است. مثلا یک جمع کننده تک بیتی از دو نیم جمع کننده و یک گیت OR تشکیل شده است (مانند شکل ۱).



شکل ۱ جمع کننده تک بیتی

علاوه بر آن هر نیم جمع کننده از دو گیت AND و XOR تشکیل شده است. طراحی ساختاری این جمع کننده به صورت زیر است:

```
library IEEE;
use IEEE.std_logic_1164.all;
Entity and_gate is
   Port(
      A, B: in std_logic;
      C   : out std_logic
   );
End entity and_gate;
Architecture gatelevel of and_gate is
Begin
   C <= A and B;
End gatelevel;
```

```
library IEEE;
use IEEE.std_logic_1164.all;
Entity or_gate is
   Port(
      A, B: in std_logic;
      C   : out std_logic
   );
End entity or_gate;
Architecture gatelevel of or_gate is
Begin
   C <= A or B;
End gatelevel;
```

```
library IEEE;
use IEEE.std_logic_1164.all;
Entity xor_gate is
   Port(
      A, B: in std_logic;
      C   : out std_logic
   );
End entity xor_gate;
Architecture gatelevel of xor_gate is
Begin
   C <= A xor B;
End gatelevel;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity half_adder is
   Port(
      in1, in2: in std_logic;
      out1, out2   : out std_logic
   );
end entity half_adder;
architecture structure of half_adder is
         component xor_gate is
                  Port(
```

```
                A, B: in std_logic;
                C   : out std_logic
            );
        End component xor_gate;
        component and_gate is
            Port(
                A, B: in std_logic;
                C   : out std_logic
            );
        End component and_gate;
begin
  xor_gate_instance0: xor_gate port map ( A => in1, B => in2, C=>out1 );
  and_gate_instance0: and_gate port map ( A => in1, B => in2, C=>out2 );
end structure;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity full_adder is
  Port(
     i0, i1, cin: in std_logic;
     s, cout   : out std_logic
  );
end entity full_adder;
architecture structure of full_adder is
        component or_gate is
            Port(
                A, B: in std_logic;
                C   : out std_logic
            );
        End component or_gate;
        component half_adder is
            Port(
                in1, in2: in std_logic;
                out1, out2   : out std_logic
            );
        end component half_adder;

        signal internal_signal0, internal_signal1, internal_signal2: std_logic;

begin

  half_adder_instance0: half_adder port map ( in1 => i0, in2 => i1, out1 => internal_signal0, out2 =>
internal_signal1);
  half_adder_instance1: half_adder port map ( in1 => internal_signal0, in2 => cin, out1 => s, out2 =>
internal_signal2);
  or_gate_instance0: or_gate port map ( A => internal_signal1, B => internal_signal2, C => cout );
end structure;
```

خروجی‌های مورد انتظار آزمایش:

انتظار می‌رود دانشجویان بتوانند سخت افزار شکل۱ را توصیف کرده و واحد تست مربوط به آن را نیز بنویسند.

و توصیف‌های انجام شده در شبیه‌ساز بدون خطا کامپایل و شبیه‌سازی شوند.

خروجی‌های این آزمایش عبارت است از:

- تحویل درستی عملکرد مدار موردنظر
- پیاده‌سازی طراحی ذکر شده، کامپایل و شبیه‌سازی آن
- پیاده‌سازی بر روی FPGA و نیز اطمینان از درستی عملکرد آن با استفاده از ورودی‌ها و خروجی‌های قابل استفاده روی برد FPGA

مراجع مطالعه/ضمایم:

موضوع:	مدارهای پایه
شماره آزمایش:	۲
عنوان:	طراحی مدارهای اولیه (Decoder2to4, Encoder4to2, Multiplexer4to1 و Comprator) ۴ بیتی
هدف:	

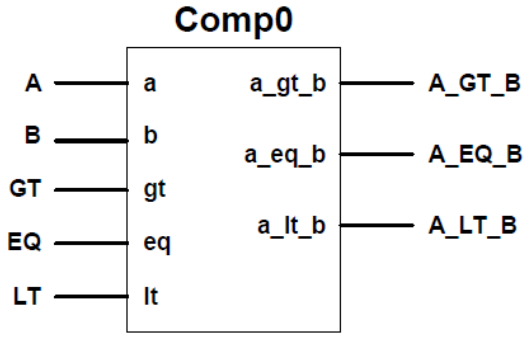
در این آزمایش هدف آشنایی با نحوه عملکرد و پیاده‌سازی هر یک از مدارهای پایه در **سطح تجرید گیت** است.

آمادگی پیش از آزمایش:

مدارهای فوق بر روی کاغذ طراحی و قبل از شروع آزمایش به مدرس آزمایشگاه تحویل داده شوند.

شرح آزمایش:

در این آزمایش، یک مالتی‌پلکسر، کدکننده، کدگشا و یک مقایسه‌گر ۴ بیتی براساس ماژول‌های تک‌بیتی آن‌ها طراحی و شبیه‌سازی گردد. طراحی انجام شده باید توسط زبان توصیف سخت‌افزار VHDL و در **سطح تجرید گیت** انجام شود. تهیه TestBench برای این جمع‌کننده نیز جزء الزام‌های آزمایش است. مدار تک بیتی مورد استفاده برای مقایسه‌گر مطابق با ساختار شکل۱ پیاده‌سازی شود.



شکل۱ ساختار ورودی/خروجی‌های مقایسه‌گر تک‌بیتی

خروجی‌های مورد انتظار آزمایش:

هر یک از موارد زیر باید تحویل داده شود:

- طراحی شماتیک طرح (محتوای توصیف) به صورت فیزیکی (بر روی کاغذ)
- بررسی درستی سیگنال‌های خروجی مدارهای مورد نظر با توجه به ورودی‌ها در شبیه‌سازی
- پیاده‌سازی مقایسه‌گر ۴ بیتی بر روی FPGA و نیز اطمینان از درستی عملکرد آن با استفاده از ورودی و خروجی‌های قابل استفاده روی برد FPGA

مراجع مطالعه/ضمایم:

[1] B. Parhami, *Computer Arithmetic-Algorithms and Hardware Designs*, Oxford Univ. Press, 2000.

موضوع:	آشنایی با پیاده‌سازی مدارهای ترتیبی در VHDL
شماره آزمایش:	۳
عنوان:	Processها و چگونگی پیاده‌سازی شمارنده‌ها و ماشین‌های حالت
هدف:	

آشنایی با Processها به عنوان بخش‌هایی که به صورت ترتیبی اجرا می‌شوند و استفاده از آن‌ها برای طراحی انواع فلیپ فلاپ‌ها، و پیاده‌سازی نمونه‌ای از شمارنده‌ها.

آمادگی پیش از آزمایش:

- مطالعه بخش زیر و کسب اطلاعات اولیه در رابطه با ساختار Process از ملزومات پیش از انجام این آزمایش می‌باشد.

Processها ماهیت ترتیبی دارند و باید در بدنه‌ی Architecture نوشته شوند. این ساختارها به صورت ترتیبی اجرا می‌شوند لذا درون آن‌ها می‌توان از دستورالعمل‌های If ... then ... else استفاده کرد. هر Process برای اجرا به Sensitivity list نیاز دارد. منظور از Sensitivity list سیگنال‌هایی هستند که وقتی Event روی آن‌ها رخ می‌دهد Process را تحریک می‌کند. مطالعه شیوه توصیف و استفاده از این قابلیت‌ها قبل از آزمایش الزامی است.

برای طراحی ماشین‌های حالت از Processها استفاده می‌شود. همان‌طور که گفته شد اگر مقدار سیگنالی که در Sensitivity list وجود دارد تغییر کند Process اجرا می‌شود. اگر Process مدل کننده‌ی یک بلوک ترکیبی است، تمام ورودی‌های آن باید در Sensitivity list لحاظ شود. ماشین‌های حالت در لبه‌های کلاک تغییر حالت می‌دهند، لذا کلاک باید در ورودی Sensitivity list وارد شود. علاوه بر این حالت درونی سیستم هم باید در یک متغیر داخلی نگه‌داری شود. برای نگه‌داری حالت‌ها در خود VHDL می‌توان Type جدید تعریف کرد که در این‌جا مثلا دو مقدار A و B نشان‌دهنده دو حالت از سیستم هستند:

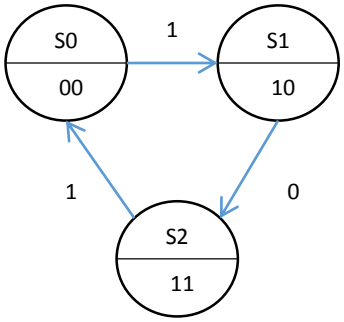
Type *state* is (*A,B*);

حال برای استفاده از Type تعریف شده می‌توان از آن Type یک سیگنال یا متغیر تعریف کرد. سپس از دستورالعمل Case روی سیگنال یا متغیر تعریف شده برای انشعاب به حالت‌های سیستم استفاده می‌شود. واضح است که این دستورالعمل Case بسته به اینکه ماشین حالت به صورت Mealy یا Moore باید طراحی شود، در داخل Process تعریف می‌شود. به عنوان مثال:

```
Signal st : state;
Process (...)
Case st is
When A => ....
When B => ....
```

برای طراحی مدارات باحافظه مدلی به نام مدل هافمن وجود دارد که قسمت ترکیبی مدار را از قسمتی ترتیبی آن جدا می‌کند. قسمت ترتیبی آن معمولاً با Processی که نسبت به سیگنال کلاک و reset حساس است نوشته می‌شود و قسمت ترکیبی آن در Processی که نسبت به حالت‌های مدار حساس است نوشته می‌شود.

در ادامه مثالی از ماشین حالت Moore و نحوه پیاده‌سازی آن آورده شده است.



شکل ۱- نمونه‌ای از ماشین حالت Moore

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity moore_machine is
	port(
		input : in std_logic ;
		output : out std_logic_vector(1 downto 0);
		clk : in std_logic
	);
end moore_machine;
architecture Behavioral of moore_machine is
	type state_t is (s0 , s1 , s2);
	signal state : state_t := s0;
	signal next_state : state_t := s0;
begin
	CMB : process(state , input)
	begin
		case state is
			when s0=>
				if(input = '1') then
					next_state <= s1;
				else
					next_state <= state ;
				end if;
			when s1=>
				....
			when s2=>
				....
			when others=>
				next_state <= s0;
		end case;
	end process;
	REG : process(clk)
		begin
			if(clk'event and clk = '1') then
				state <= next_state;
			end if;
	end process;
	output <= “00” when state = s0   else
		  “10” when state = s1   else
		  “11” when state = s2;
end Behavioral;
```

شرح آزمایش:

۱) یک فلیپ‌فلاپ از نوع D(DFF) با سیگنال Reset ناهمگام (asynchronous) در منطق منفی (active low) طراحی کنید.

۲) یک فلیپ‌فلاپ از نوع T(TFF) با سیگنال Reset ناهمگام طراحی کنید.

۳) یک Ripple Counter، ۴ بیتی که نمونه‌ای از شمارنده‌های ناهمگام می‌باشد را با استفاده از TFF ساخته شده در (۲) طراحی کنید.

۴) مدار یک Sequence detector برای رشته‌ی "۱۱۰۱" را ابتدا به صورت Mealy و سپس Moore طراحی کرده و آن را با استفاده از زبان VHDL طراحی کنید.

۵) مداری طراحی کنید که رخداد هر یک از دو رشته "۰۱۱۰" و "۰۱۰۱" را در ورودی تشخیص دهد.

خروجی‌های مورد انتظار آزمایش:

هریک از بخش‌های این آزمایش در قالب‌های زیر به ترتیب تحویل مدرس آزمایشگاه گردد:

- طراحی شماتیک طرح (محتوای توصیف) بر صورت فیزیکی
- پیاده‌سازی مدارها با استفاده از زبان‌های توصیف
- شبیه‌سازی مدارهای توصیف شده و تهیه Testbench برای آن و نشان دادن درستی عملکرد آن‌ها.
- سنتز و پیاده‌سازی مدارها بر روی برد FPGA و اثبات درستی عملکرد مدارها. (برای استفاده از کلاک برد و مشاهده نتایج نیاز به کاهش فرکانس برد می‌باشد، بدین جهت متناسب با شرایط کلاس این کاهش فرکانس توسط مدرس آزمایشگاه و یا دانشجویان انجام گیرد.)

مراجع مطالعه/ضمایم:

[1] M. Zwolinski, “Digital System Design with VHDL,” Pearson Educated, 2000, pp. 104-128.

موضوع:	جمع کننده‌ها
شماره آزمایش:	۴
عنوان:	پیاده‌سازی انواع جمع کننده‌ها و تحلیل سرعت آنها
هدف:	

هدف از این آزمایش، آشنایی با چگونگی عملکرد هر یک از جمع‌کننده‌ها است. تفاوت این جمع‌کننده‌ها در سرعت محاسبه عملیات جمع می‌باشد. در این آزمایش هدف آشنایی با نحوه عملکرد و پیاده‌سازی سخت‌افزاری آن‌ها در **سطح تجرید گیت** است.

آمادگی پیش از آزمایش:

- مطالعه انواع جمع‌کننده‌های زیر و طراحی و تحویل شماتیک سخت‌افزاری آن با استفاده از واحدهای Full Adder
  - الف) Ripple Adder (Cascaded Adder)
  - ب) Carry-Lookahead Adder
  - ج) Carry Select Adder
- تحلیل تئوری و مقایسه سرعت عملکرد جمع کننده‌های فوق

شرح آزمایش:

در این آزمایش، یک جمع‌کننده ۴ بیتی از نوع‌های زیر، طراحی و پیاده‌سازی گردد. طراحی انجام شده با زبان توصیف سخت‌افزاری VHDL و در **سطح تجرید گیت** انجام شود. در این پیاده‌سازی از واحدهای Full Adder و HalfAdder استفاده گردد. تهیه TestBench برای این جمع کننده نیز جزء الزام‌های آزمایش است.

الف) Ripple Adder (Cascaded Adder)

ب) Carry-Lookahead Adder

ج) Carry Select Adder

خروجی‌های مورد انتظار آزمایش:

هر یک از موارد زیر باید تحویل داده شود:

- طراحی شماتیک طرح (محتوای توصیف) بر روی کاغذ
- بررسی درستی سیگنال‌های خروجی مدار مورد نظر با انجام شبیه‌سازی
- پیاده‌سازی بر روی FPGA و نیز اطمینان از درستی عملکرد آن با استفاده از ورودی‌ها و خروجی‌های قابل استفاده روی برد FPGA
- تحلیل و مقایسه سرعت عملکرد جمع کننده‌های فوق با یکدیگر و در عرض بیت‌های مختلف

مراجع مطالعه/ضمایم:

[1] B. Parhami, *Computer Arithmetic-Algorithms and Hardware Designs*, Oxford Univ. Press, 2000.
[2] https://en.wikipedia.org/wiki/Carry-select_adder

موضوع:	ضرب‌کننده‌ها
شماره آزمایش:	۵
عنوان:	پیاده‌سازی انواع ضرب‌کننده ۴ بیتی
هدف:	

آشنایی با پیاده‌سازی انواع ضرب‌کننده آرایه‌ای و ضرب‌کننده‌های سریع با زبان VHDL.

پیاده‌سازی سخت‌افزاری تمامی واحدها در **سطح تجرید گیت** صورت گیرد.

آمادگی پیش از آزمایش:

طریقه عملکرد ضرب کننده‌های الف، ب و ج را بیان نموده و آن‌ها را با یکدیگر مقایسه کنید. (تفاوت‌ها و شباهت‌ها موجود بیان شود.)

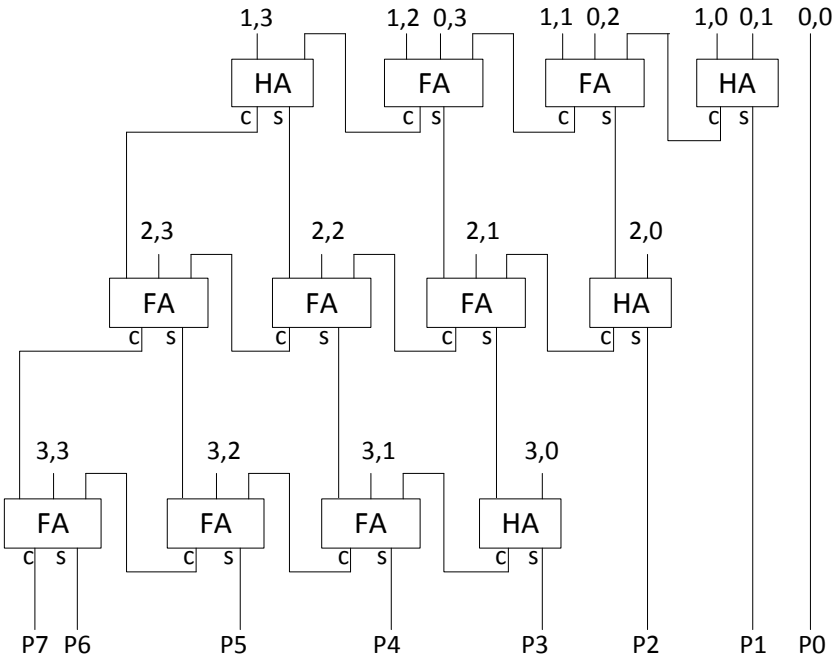
شرح آزمایش:

ضرب‌کننده‌ها یکی از مهمترین واحدهای محاسباتی در طراحی پردازنده‌ها محسوب می‌شوند و روش‌های گوناگونی برای پیاده‌سازی این واحد ارائه شده است. در این آزمایش چهار روش متفاوت برای انجام عملیات ضرب ۴ بیتی صحیح بی‌علامت ارائه شده است. در هریک از این روش‌ها دو عدد ۴ بیتی به عنوان ورودی و یک عدد ۸ بیتی در خروجی محاسبه می‌شود.

الف) ضرب‌کننده معمولی:

در این ضرب‌کننده مثل ضرب دهدهی (decimal)، در هر مرحله یک بیت از مضروب (multiplier) در تمام بیت‌های مضروب فیه (multiplicand) ضرب شده و با یک بیت شیفت با خروجی مرحله قبل جمع می‌شود.

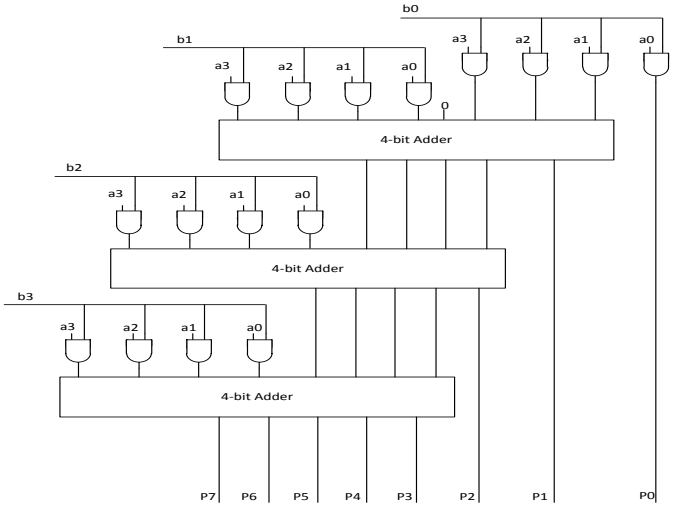
نمودار این ضرب‌کننده در شکل ۱ نشان داده شده است.



شکل۱- نمودار بلوکی ضرب کننده معمولی

ب) ضرب‌کننده آرایه‌ای:

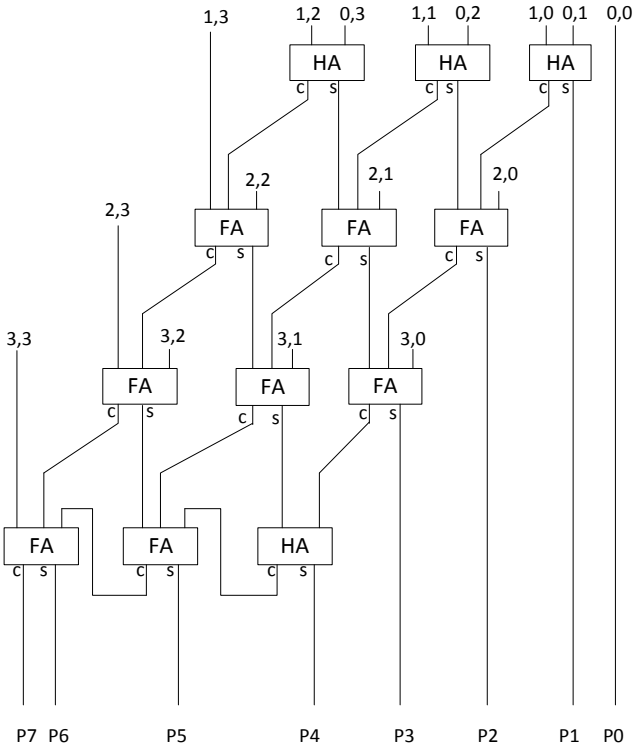
در این ضرب‌کننده از واحدهای جمع‌کننده ۴ بیتی استفاده شده است. نمودار این ضرب‌کننده در شکل ۲ نشان داده شده است.



شکل۲- نمودار بلوکی ضرب کننده آرایه ای

ج) ضرب‌کننده Carry-Save Adder

در این نوع ضرب کننده با انتقال بیت نقلی به مرحله بعد سرعت انجام ضرب افزایش می‌یابد. نمودار این ضرب‌کننده در شکل ۳ نشان داده شده است.

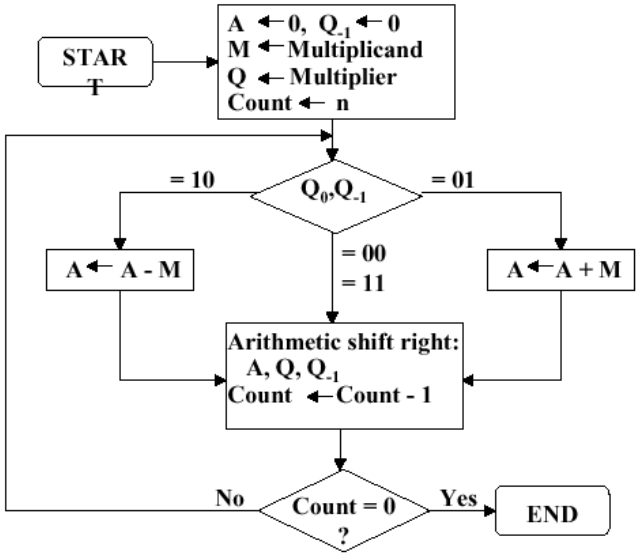


شکل۳-نمودار بلوکی ضرب کننده Carry save adder

د) ضرب‌کننده Booth (اختیاری)

در این نوع ضرب کننده که از روش‌های ضرب سریع محسوب می‌شود ضرب‌های جزئی کمتری خواهیم داشت که می‌توان این ضرب‌های جزئی را به روش‌های مختلف با هم جمع کرد. این روش هر چند روند ضرب را سریع‌تر می‌کند ولی آن را پیچیده‌تر خواهد کرد.

فلوچارت این روش در شکل ۴ نشان داده شده است. تفاوت اصلی این روش با روش‌های قبل در ترتیبی بودن آن است. به این صورت که مضروب فیل (multiplicand) در یک shift register ۸ بیتی قرار می‌گیرد و در هر پالس ساعت یک بیت به سمت راست shift داده می‌شود. خروجی این ضرب کننده پس از ۸ پالس ساعت قابل استفاده است.



شکل۴-فلوچارت الگوریتم ضرب Booth

خروجی‌های مورد انتظار آزمایش:

- شبیه‌سازی ضرب کننده‌ها و بررسی صحت عملکرد توسط TestBench
- تفاوت‌های بیان شده در پیش‌گزارش دانشجویان در صورت امکان توسط ابزار پیاده‌ساز درستی‌سنجی شود.
- پیاده‌سازی بر روی FPGA و بررسی صحت عملکرد با استفاده از ورودی‌ها و خروجی‌های قابل اعمال روی برد FPGA

مراجع مطالعه/ضمایم:

[1] M. Morris Mano, *Computer system architecture*, Prentice-Hall, 3rd edition.

موضوع: تاخیر در مدارات دیجیتال

شماره آزمایش: ۶

عنوان: ایجاد تاخیر مشخص در مدارهای دیجیتال

هدف:

هدف از این آزمایش، آشنایی با چگونگی ایجاد تاخیر در مدارهای دیجیتال و بررسی آن بر روی برد می‌باشد.

آمادگی پیش از آزمایش:

- ساختار و عملکرد مدارهای شیفت دهنده (منطقی و ریاضی) را تشریح و تفاوت‌ها را بیان کنید.
- ساختار و نحوه عملکرد 7-seg را به طور کامل توضیح دهید.

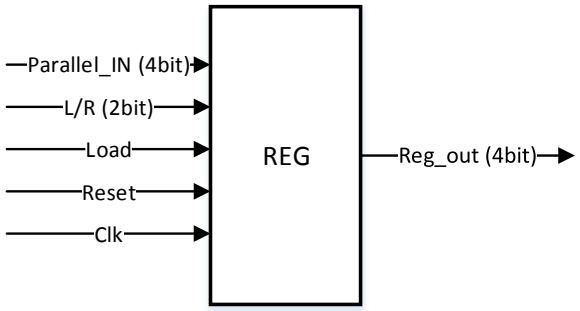
شرح آزمایش:

یکی از بحث‌های اصلی در مدارهای دیجیتال، بخش زمان‌بندی است. زمان‌بندی نامناسب مدارات می‌تواند مشکلات زیادی را در هنگام شبیه‌سازی و پیاده‌سازی به وجود آورد. در این آزمایش به چگونگی ایجاد تاخیر در مدارات دیجیتال پرداخته می‌شود. برای نیل به این منظور از ثبات‌های دارای قابلیت انتقال (منطقی و ریاضی) استفاده می‌کنیم.

الف) پیاده‌سازی ثبات با قابلیت انتقال (Shift):

یک ثبات چهار بیتی با قابلیت‌های زیر طراحی نمایید. ساختار بلوک ثبات در شکل ۱ نشان داده شده است. قابلیت‌های این بلوک به شرح زیر می‌باشد: (ثبات حساس به لبه بالارونده کلاک است.)

- انتقال منطقی (چپ/ راست)
- انتقال ریاضی (چپ / راست)



شکل ۱- ساختار ورودی/خروجی‌های بلوک ثبات

در بلوک ثبات فوق درگاه‌های ورودی خروجی به شرح زیر است:

- ورودی LR دوبیتی است که برای انتخاب بین حالات شیفت است. ("LR="00 ثبات مقدار قبلی خود را حفظ می‌کند.) (انتقال به چپ ریاضی و منطقی یک حالت فرض شوند.)

- ورودی Load تک بیتی است که در صورت یک بودن، ورودی چهاربیتی Parallel_IN همزمان با لبه بالارونده کلاک در ثبات بارگذاری می‌شود. و در صورت صفر بودن ثبات مقدار قبلی خود را نگه می‌دارد.

ب) چگونگی ایجاد مقدار تاخیر مشخص برای نمایش خروجی روی برد:

برای ایجاد تاخیر و قابل رویت شدن تغییرات خروجی‌ها بر روی برد، از شمارنده برای تنظیم و کاهش فرکانس کلاک روی برد استفاده می‌شود. فرکانس کاری برد موجود در آزمایشگاه ۴۰ مگاهرتز می‌باشد، بنابراین برای دستیابی به تاخیر ۱ ثانیه، نیاز به شمارنده‌ای است که به میزان $۴\times۱۰^{۷}-۱$ بار بشمارد تا تاخیری به میزان ۱ ثانیه ایجاد گردد.

در این بخش از آزمایش شمارنده‌ای طراحی و به کد بخش الف اضافه کنید تا بتوان خروجی‌های شیفت رجیستر را با تاخیرهای مناسب بر روی برد مشاهده نمود.

ج) نمایش خروجی بر روی 7-Seg:

چگونگی اتصال پایه‌ها به منظور نمایش بر روی 7-seg به صورت زیر می‌باشد. (کلاک برد به پایه 184 متصل شده است.)

```
#PlanAhead Generated physical constraints

NET "SEG_DATA[0]" LOC = P10;
NET "SEG_DATA[1]" LOC = P7;
NET "SEG_DATA[2]" LOC = P11;
NET "SEG_DATA[3]" LOC = P5;
NET "SEG_DATA[4]" LOC = P4;
NET "SEG_DATA[5]" LOC = P12;
NET "SEG_DATA[6]" LOC = P9;
NET "SEG_DATA[7]" LOC = P3;

NET "SEG_SEL[0]" LOC = P15;
NET "SEG_SEL[1]" LOC = P20;
NET "SEG_SEL[2]" LOC = P19;
NET "SEG_SEL[3]" LOC = P18;
NET "SEG_SEL[4]" LOC = P16;
```

خروجی‌های مورد انتظار آزمایش:

هریک از بخش‌های این آزمایش در قالب‌های زیر به ترتیب تحویل مدرس آزمایشگاه گردد:

- پیاده‌سازی مدارها با استفاده از زبان‌های توصیف سخت افزار VHDL
- شبیه‌سازی مدارهای توصیف شده و تهیه Testbench برای آن و نشان دادن درستی عملکرد آن‌ها.
- سنتز و پیاده‌سازی مدارها بر روی بورد FPGA و اثبات درستی عملکرد مدارها.

مراجع مطالعه/ضمایم:

[1] Mano, Kime, *Logic and Computer Design Fundamentals,* Prentice-Hall, 2004.

موضوع: حافظه‌ها

شماره آزمایش: ۷

عنوان: طراحی حافظه ROM, RAM و CAM

هدف:

هدف از این آزمایش، آشنایی با انواع حافظه‌ها، طراحی و پیاده‌سازی برخی از آن‌ها است.

آمادگی پیش از آزمایش:

چگونگی عملکرد چهار نوع حافظه (زیر) الف، ب، ج و د را به صورت دقیق تشریح کرده و تفاوت‌های اصلی مابینشان را بیان کنید.

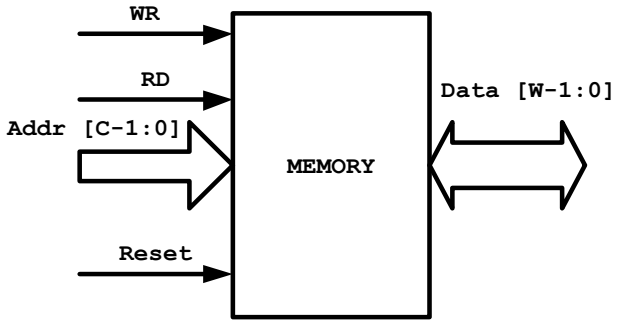
شرح آزمایش:

یکی از بخش‌های اصلی در مدارهای دیجیتال، بخش حافظه است. مدل‌سازی نامناسب حافظه علاوه بر اینکه ممکن است زمان شبیه‌سازی را طولانی کند، در هنگام سنتز نیز در صورت توصیف نادرست باعث استفاده غیرمعمول از منابع سخت‌افزاری می‌شود. در این آزمایش هدف اصلی آشنایی با نحوه مدل کردن انواع حافظه‌ها با زبان توصیف سخت‌افزار است.

الف) طراحی حافظه RAM:

یک حافظه RAM مطابق با شکل زیر طراحی کنید. پارامترهای این بلوک به شرح زیر می‌باشد:

- عرض حافظه W
- تعداد خانه‌های حافظه D
- عرض درگاه Address برابر با log2(D) می‌باشد که فعلا آن را هم در کد با پارامتر C نشان دهید.



شکل ۱- بلوک دیاگرام یک واحد حافظه

در بلوک حافظه فوق:

درگاه Data از نوع ورودی/خروجی تعریف می‌شود:

- سیگنال‌های WR و RD حساس به لبه بالاروند عمل کرده بطوریکه هنگامی WR در لبه بالارونده است مقدار Data به خانه‌ای که توسط Addr مشخص شده، نوشته می‌شود. هنگامی که RD در لبه بالا رونده است عمل خواندن از حافظه انجام شود.
- درگاه Addr نشان می‌دهد که داده به کدام خانه SRAM باید نوشته و از کدام خانه باید خوانده شود.
- سیگنال Clk پالس ساعت است.
- سیگنال Reset در منطق منفی و ناهمگام (Asynchronous active low) عمل می‌کند.

در این بلوک، خانه‌های RAM به هنگام Reset به بدین ترتیب مقداردهی شوند:
مقدار خانه شماره "۰" برابر "۰". مقدار خانه شماره "۱"برابر "۱" و به همین ترتیب مقدار اولیه هر خانه برابر با آدرس آن خانه باشد.
کد سخت افزاری فوق را نوشته و شبیه‌سازی کنید. (درحالت‌های مختلف)

ب) طراحی حافظه ROM:

با تغییری در طراحی انجام شده در بخش (الف)، این حافظه را به حافظه ROM تبدیل کنید.

ج) طراحی حافظه RAM دو درگاهه (Dual port RAM):

با تغییری در طراحی انجام شده در بخش (الف)، این حافظه را به حافظه دو درگاهه تبدیل کنید که در آن دو درگاه مستقل برای خواندن/نوشتن بطور کاملاً مستقل وجود داشته باشد.

د) طراحی حافظه CAM:

با مشخصات داده شده در (الف) حافظه آدرس‌پذیر محتوی (Content Addressable Memory) طراحی و پیاده‌سازی کنید. این حافظه بدون درگاه آدرس بوده، و خواندن یا نوشتن در آن بر اساس محتوا انجام خواهد شد. در هنگام نوشتن داده، اگر داده داخل حافظه نباشد، آن داده در اولین مکان خالی نوشته می‌شود و در هنگام خواندن، در صورتی که داده وارد شده در حافظه وجود داشته باشد، سیگنال match (تک بیتی) یک خواهد شد به معنای آنکه داده در حافظه یافت شده است و در غیر اینصورت صفر خواهد بود.

خروجی‌های مورد انتظار آزمایش:

هریک از بخش‌های این آزمایش در قالب‌های زیر به ترتیب تحویل مدرس آزمایشگاه گردد:

- طراحی شماتیک طرح و محتوای توصیف بر روی کاغذ
- پیاده‌سازی مدارها با استفاده از زبان‌های توصیف
- شبیه‌سازی مدارهای توصیف شده و تهیه Testbench برای آن و نشان دادن درستی عملکرد آن‌ها.
- سنتز و پیاده‌سازی مدارها بر روی بورد FPGA و اثبات درستی عملکرد مدارها.

مراجع مطالعه/ضمایم:

[1] Mano, Kime, *Logic and Computer Design Fundamentals,* Prentice-Hall, 2004, chapter 12.

موضوع: تقسیم‌کننده‌ها

شماره آزمایش: ۸

عنوان: تقسیم کننده دودویی

هدف:

طراحی تقسیم کننده دودویی با کمک زبان توصیف سخت‌افزار

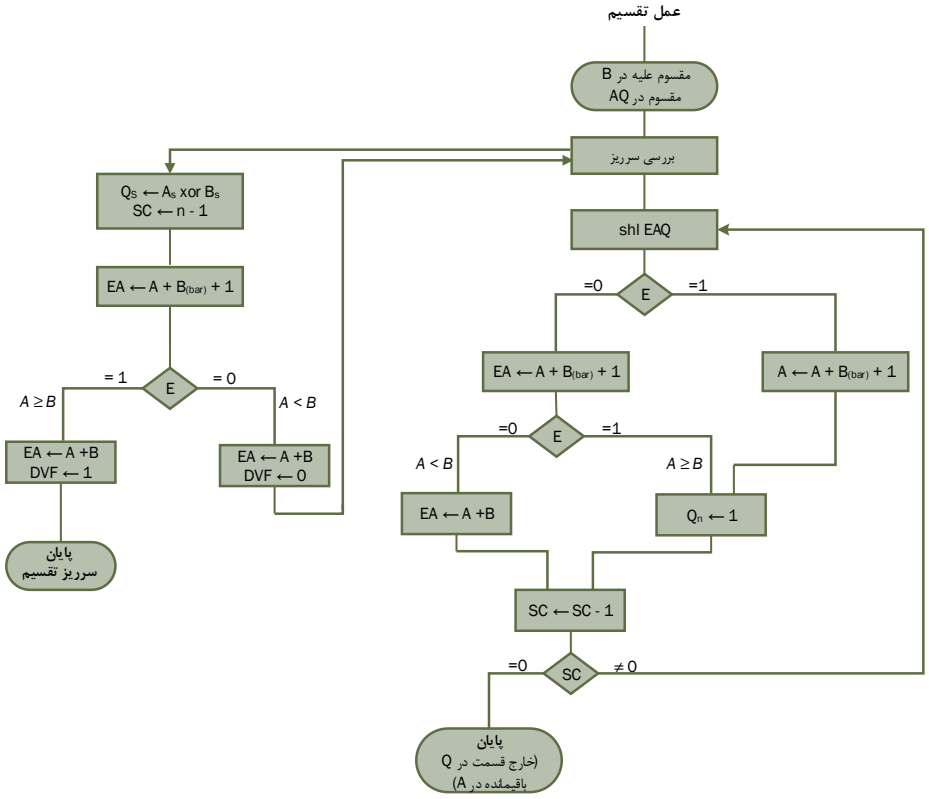
آمادگی پیش از آزمایش:

الگوریتم‌های مختلف پیاده‌سازی سخت‌افزاری عملیات تقسیم را مطالعه کنید. بدین منظور می‌توانید از منابع آورده شده در بخش مراجع این آزمایش و یا منابع فراوان در اینترنت کمک بگیرید. سپس یک الگوریتم را برای پیاده‌سازی در کلاس انتخاب کرده (در انتخاب الگوریتم به چگونگی پیاده‌سازی آن در کلاس فکر کنید.) و به صورت دقیق چگونگی عملکرد آن را همراه با یک مثال عددی تشریح کنید. (مراحل انجام تقسیم در مثال خود را به صورت کامل بنویسید.)

شرح آزمایش:

کد توصیف سخت‌افزاری تقسیم کننده دودویی مدنظر را برای تقسیم یک مقسوم ۸ بیتی، بر مقسوم علیه ۴ بیتی بنویسید.

در شکل۱، یک فلوچارت نمونه برای الگوریتم تقسیم آورده شده است. (در صورت تمایل می‌توانید پیش‌گزارش خود را مبتنی بر این فلوچارت انجام دهید.)



شکل ۱- فلوچارت الگوریتم تقسیم

خروجی‌های مورد انتظار آزمایش:

موارد زیر باید تحویل مدرس آزمایشگاه شود:

- توصیف سخت افزاری و پیاده‌سازی تقسیم‌کننده مورد نظر
- انجام شبیه‌سازی و ارایه آن به مدرس آزمایشگاه
- تمام کدهای توصیفی به همراه Testbench برای مشاهده نتایج ارائه شود.
- سنتز و پیاده‌سازی بر روی بورد FPGA و اثبات درستی عملکرد تقسیم کننده.

مراجع مطالعه/ضمایم:

[1] M. Mano, *Computer System Architecture*,Prentice-Hall, 3rd Edition, pp. 345-351, 1993.

[2] D. Patterson, J. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, Morgan Kaufmann Publishers, Inc., Third edition, 2007.

موضوع:	کامپیوتر پایه(۱)
شماره آزمایش:	۹
عنوان:	پیاده‌سازی یک کامپیوتر پایه
هدف:	

پیاده‌سازی یک کامپیوتر پایه و مشاهده عملی چگونگی عملکرد آن.

آمادگی پیش از آزمایش:

بررسی کلیات کامپیوتر پایه درس معماری کامپیوتر؛ مطالعه کد موجود در دستورکار و کسب آمادگی لازم جهت پیاده‌سازی آن.

شرح آزمایش:

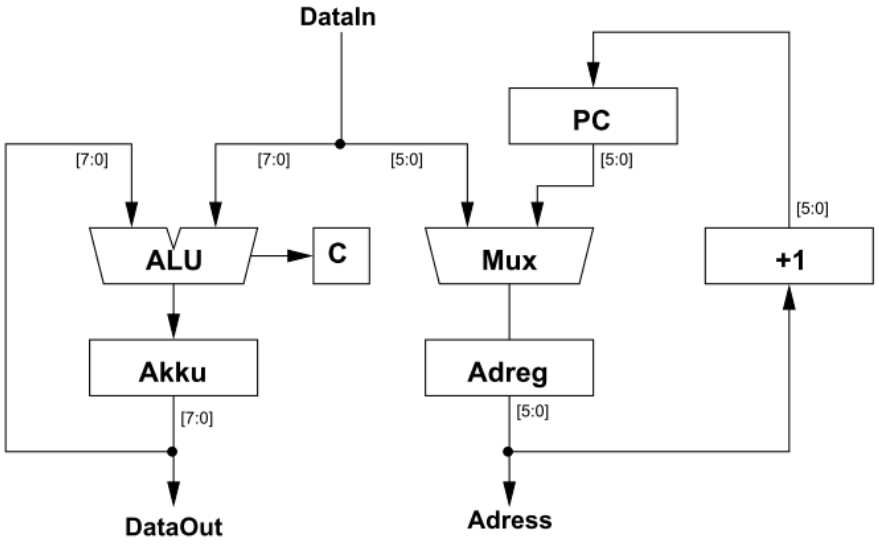
در این آزمایش یک کامپیوتر پایه طراحی و سپس روی FPGA پیاده‌سازی می‌شود. کامپیوتر پایه از سه جز تشکیل می‌شود: ۱) پردازنده، ۲) حافظه و ۳) واحد ورودی/خروجی. با توجه به بوردهای موجود، امکان ایجاد هر سه این قسمت‌ها بر روی بورد وجود دارد.

کامپیوتری که پیاده‌سازی می‌شود می‌تواند دستورات موجود در شکل۱ را داشته باشد: (دستور show محتوای یک رجیستر را روی LEDها نشان می‌دهد.)

یک نمونه کد کامپیوتر پایه و مسیر داده مربوط به آن در زیر آمده است.

Load
Store
Brz
Brnz
Add
Sub
Ror
Rol
Shr
Shl
Show
And
Or
Xor
Not
Halt

شکل۱-مجموعه دستورالعمل‌های پیشنهادی برای کامپیوتر پایه



شکل۲-دیاگرام بلوکی کامپیوتر پایه

```
-- Minimal 8 Bit CPU

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity CPU8BIT2 is
        port (  data:   inout   std_logic_vector(7 downto 0);
                adress: out     std_logic_vector(5 downto 0);
                oe:     out     std_logic;
                we:     out     std_logic;              -- Asynchronous memory interface
                rst:    in      std_logic;
                clk:    in      std_logic);
end;

architecture CPU_ARCH of CPU8BIT2 is
        signal  akku:   std_logic_vector(8 downto 0);  -- akku(8) is carry !
        signal  adreg:  std_logic_vector(5 downto 0);
        signal  pc:     std_logic_vector(5 downto 0);
        signal  states: std_logic_vector(2 downto 0);
begin
        Process(clk,rst)
        begin
          If (rst = '0') then
                adreg   <= (others => '0');     -- start execution at memory location 0
                states  <= "000";
                akku <= (others => '0');
                pc   <= (others => '0');

          elsIf rising_edge(clk) then
                -- PC / Adress path
                If (states = "000") then
                        pc      <= adreg + 1;
                        adreg   <= data(5 downto 0);
                else
                        adreg   <= pc;
                end If;

                -- ALU / Data Path
                Case states is
                        when "010" => akku <= ("0" & akku(7 downto 0)) + ("0" & data);      -- add
                        when "011" => akku(7 downto 0) <= akku(7 downto 0) nor data;        -- nor
                        when "101" => akku(8) <= '0';           -- branch not taken, clear carry
                        when others => null;            -- instr. fetch, jcc taken (000), sta (001)
                end Case;
```

```
            -- State machine
            If (states /= "000") then states <= "000";                    -- fetch next opcode
            elsIf (data(7 downto 6) = "11" and akku(8)='1') then states <= "101";   -- branch not taken
            else  states <= "0" & not data(7 downto 6);                   -- execute instruction
            end If;
      end If;
    end Process;

    -- output
    adress  <= adreg;
    data    <= "ZZZZZZZZ" when states /= "001" else akku(7 downto 0);
    oe      <= '1' when (clk='1' or states  = "001" or rst='0' or states = "101") else '0';
    -- no memory access during reset and state "101" (branch not taken)
    we <= '1' when (clk='1' or states /= "001" or rst='0') else '0';

end CPU_ARCH;
```

خروجی‌های مورد انتظار آزمایش:

یک برنامه در حافظه کامپیوتر load شود و بتوان اجرای آن را روی LED های روی بورد FPGA دید.

مراجع مطالعه/ضمایم:

[1] http://www.tuhh.de/setb0209/cpu/.

موضوع:	کامپیوتر پایه(۲)
شماره آزمایش:	۹
عنوان:	پیاده‌سازی یک کامپیوتر پایه
هدف:	

پیاده‌سازی یک کامپیوتر پایه و مشاهده عملی چگونگی عملکرد آن.

آمادگی پیش از آزمایش:

قبل از طراحی ۱:

برای پیاده‌سازی کامپیوتر پایه می‌توان از دو روش structural و behavioral استفاده کرد. در روش structural اجزای داخلی پردازنده را به صورت مجزا طراحی و پیاده‌سازی شده و در نهایت آن‌ها کنار هم قرار می‌گیرند. در روش behavioral، پردازنده توسط توصیف رفتار آن طراحی می‌شود. به مثال زیر دقت کنید:

مثال: دستور JP را به دو روش structural و behavioral بنویسید.

[T0 : MAR ← PC ; MDR ← Mem[MAR
T1 : IR ← MDR ; PC ← PC+1
T2 : PC ← IR

پاسخ: در روش structural ابتدا هر یک از اجزای PC, MAR, MDR, IR به صورت مجزا طراحی می‌شوند. سپس یک ماشین حالت فعال شدن هر یک از آن‌ها را در زمان‌های مختلف کنترل می‌کند. درنهایت تمامی این اجزا با باس داده مشترک به یکدیگر متصل می‌شوند و پردازنده را می‌سازند. ماشین حالتی که ترتیب و چگونگی مسیر داده را کنترل می‌کند، CU یا واحد کنترل نامیده می‌شود. طراحی این واحد کنترل به دو روش hardwired و micro programmed صورت می‌گیرد.

در طراحی behavioral تمامی اجزای پردازنده در یک فایل توصیف می‌شوند. رجیسترهای PC, MAR, MDR, IR به صورت سیگنال تعریف می‌شوند. ترتیب و چگونگی مسیر داده در خود پردازنده و در process توصیف می‌شود.

قبل از طراحی ۲:

با به کارگیری rtl برای دستورات زیر، ترتیب انتقال رجیسترها را مشخص کنید.

Instruction	Operation
LD A, [ADDR]	ACC ← [ADDR]
ST A, [ADDR]	[ADDR] ← ACC
JP ADDR	JUMP TO ADDR
JPC ADDR	IF CF == 1: JUMP TO ADDR
AND A, [ADDR]	ACC ← ACC & [ADDR]
ADD A, [ADDR]	ACC ← ACC + [ADDR]
NOT A	ACC ← ~ACC
SHR A	ACC ← ACC >> 1

شرح آزمایش:

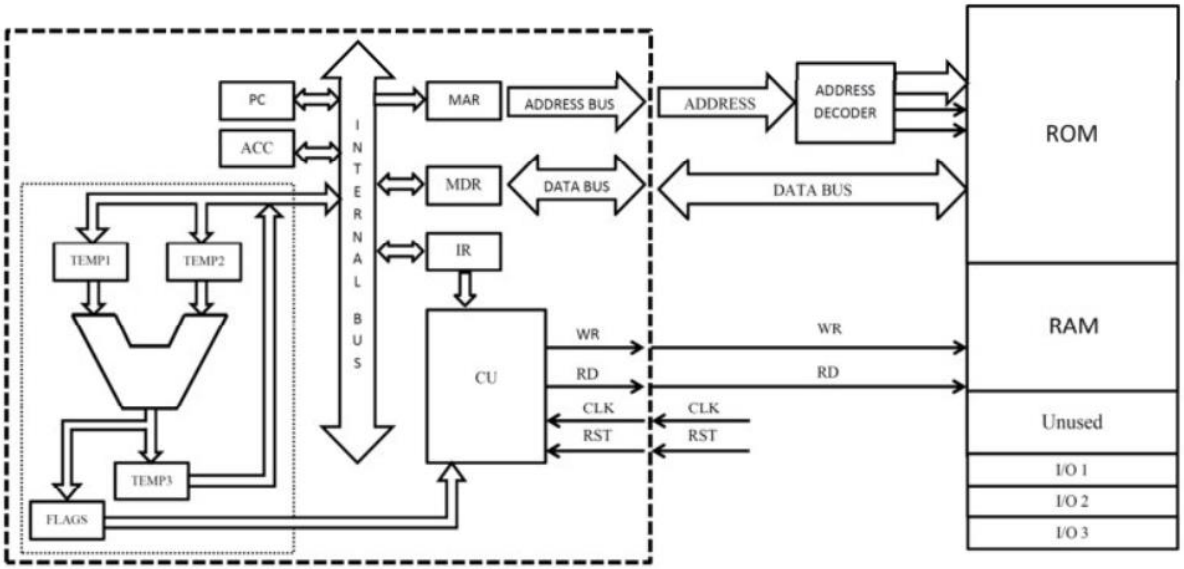
در این آزمایشگاه کامپیوتر پایه‌ای ۸ بیتی طراحی خواهد شد که از بخش‌های زیر تشکیل شده است:

- حافظه
- رمزگشای حافظه
- ورودی و خروجی
- پردازنده

پیاده‌سازی این کامپیوتر پایه در این آزمایش صورت می‌گیرد. در ابتدا پس از طراحی حافظه‌های بیرونی و گسترش حافظه در عرض، پهنای گذرگاه داده افزایش داده می‌شود. سپس این حافظه و تعدادی ورودی-خروجی در فضای آدرس‌دهی نگاشت می‌شوند. در ادامه با طراحی مدارهای کدگشا بخش پیرامونی پردازنده کامل می‌شود.

در ادامه بلوک‌های داخلی پردازنده طراحی شده و در انتها بلوک‌های طراحی شده در کنار یکدیگر قرار گرفته و کامپیوتر پایه را تشکیل می‌دهند.

شکل زیر بلوک دیاگرام پردازنده را نشان می‌دهد:



این پردازنده ۸ بیتی است و توانایی آدرس‌دهی ۳۲ بایت را دارد. تعداد خط‌های آدرس ۵ بیت و پهنای گذرگاه داده ۸ بیت است. هر دستور در این پردازنده یک بایت طول دارد و شامل دو بخش زیر است:

- Opcode: سه بیت پر ارزش دستور
- Operand: پنج بیت کم ارزش دستور

پردازنده شامل بلوک‌های داخلی زیر است که این بلوک‌ها توسط باس داخلی ۸ بیتی با هم ارتباط دارند:

- رجیستر دستور IR: رجیستری که داده دستور را از باس داخلی بارگذاری کرده و opcode را به cu و operand را به باس داخلی تحویل می‌دهد.
- رجیستر داده MDR: رجیستر دو طرفه‌ای که داده را میان باس داخلی و باس داده خارجی قرار می‌دهد.
- رجیستر آدرس MAR: رجیستری که داده آدرس را از باس داخلی بارگذاری کرده و به طور مستقیم بر باس خارجی قرار می‌دهد.
- شمارنده برنامه PC: رجیستری است که آدرس دستور بعدی در آن قرار می‌گیرد.
- انباره ACC: رجیستری که می‌تواند داده را از باس داخلی بگیرد یا بر آن قرار دهد.
- واحد کنترل CU: بخش‌های گوناگون پردازنده را با هم هماهنگ می‌کند.
- ALU: محاسبات عددی و منطقی در این بخش صورت می‌گیرد.

خروجی‌های مورد انتظار آزمایش:

۱- طراحی حافظه‌های مورد نظر

یک حافظه ROM شانزده بایتی طراحی کنید که شامل سیگنال rd، cs، خروجی یک بایتی و باس آدرس ۴ بیتی است.

یک حافظه RAM هشت بایتی طراحی کنید که شامل سیگنال wr, rd، cs، خروجی یک بایتی و باس آدرس ۳ بیتی است.

۲- طراحی مدارهای رمزگشایی حافظه و ورودی-خروجی

یک مدار رمزگشای حافظه طراحی کنید که خروجی‌های زیر را به ازای بازه‌های موردنظر بدهد:

Address Range	Signal
00h – 0fh	Chip_Select_ROM
10h – 17h	Chip_Select_RAM
1dh	Chip_Select_IO0
1eh	Chip_Select_IO1
1fh	Chip_Select_IO2

یک درگاه خروجی، یک درگاه ورودی و یک درگاه ورودی-خروجی طراحی کنید.

درگاه خروجی دارای سیگنال یک بایتی ورودی باس داده، سیگنال یک بایتی خروجی و دو سیگنال تک بیتی cs و wr است. هنگامی که cs و wr فعال باشد، مقدار موجود در باس داده بر روی پورت خروجی قرار می‌گیرد.

درگاه ورودی دارای سیگنال یک بایتی ورودی، سیگنال یک بایتی خروجی باس داده و دو سیگنال تک بیتی cs و rd است. هنگامی که cs و rd فعال باشد، مقدار موجود در پورت ورودی بر روی باس داده قرار می‌گیرد.

درگاه ورودی-خروجی دارای یک سیگنال یک بایتی ورودی-خروجی، یک سیگنال یک بایتی ورودی-خروجی باس داده، سیگنال یک بایتی خروجی و سه سیگنال تک بیتی wr،cs و rd است. این پورت دو طرفه می‌تواند هم ورودی و هم خروجی باشد.

۳- طراحی رجیسترهای داخلی

رجیسترهای داخلی زیر را طراحی کنید:

ACC شامل:

- سیگنال دوطرفه ۸ بیتی D
- سیگنال ورودی تک بیتی LE
- سیگنال ورودی تک بیتی OE

MAR شامل:

- سیگنال ورودی ۸ بیتی D
- سیگنال خروجی ۵ بیتی A
- سیگنال ورودی تک بیتی LE

MDR شامل:

- سیگنال دوطرفه ۸ بیتی DIN
- سیگنال دوطرفه ۸ بیتی DOUT
- دو سیگنال ورودی تک بیتی LE_IN و LE_OUT
- دو سیگنال ورودی تک بیتی OE_IN و OE_OUT

IR شامل:

- سیگنال دوطرفه ۸ بیتی D
- سیگنال خروجی ۳ بیتی I
- سیگنال ورودی تک بیتی LE
- سیگنال ورودی تک بیتی OE

PC شامل:

- سیگنال دوطرفه ۸ بیتی D
- سیگنال ورودی تک بیتی LE
- سیگنال ورودی تک بیتی OE
- سیگنال ورودی تک بیتی CLR
- سیگنال ورودی تک بیتی CLK
- سیگنال ورودی تک بیتی INC

۴- طراحی ALU

واحد محاسبه و منطق را طوری طراحی کنید که دارای دو رجیستر ورودی ۸ بیتی temp1 و temp2 و یک رجیستر ۸ بیتی temp3 باشد. هم‌چنین شامل:

- سیگنال دوطرفه ۸ بیتی D
- سیگنال خروجی تک بیتی CF
- سیگنال ورودی دو بیتی C
- سه سیگنال ورودی تک بیتی LE_temp1، LE_temp1 و LE_temp1
- سیگنال ورودی تک بیتی OE

واحد ALU با کدهای کنترلی C عملیات زیر را انجام می‌دهد:

C[1:0]	Operation
00	AND (TEMP3 ←TEMP1 & TEMP2)
01	ADD (TEMP3 ←TEMP1 + TEMP2 , CF ← carry)
10	NOT (TEMP3 ←! TEMP1)
11	SHR (TEMP3 ← {0, TEMP1[7:1]})

۵- طراحی CU

واحد کنترلی با روش hardwired طراحی کنید که شامل دستورات زیر باشد:

OP-Code	Instruction	Operation
000	LD A, [ADDR]	ACC ← [ADDR]
001	ST A, [ADDR]	[ADDR] ← ACC
010	JP ADDR	JUMP TO ADDR
011	JPC ADDR	IF CF == 1: JUMP TO ADDR
100	AND A, [ADDR]	ACC ← ACC & [ADDR]
101	ADD A, [ADDR]	ACC ← ACC + [ADDR]
110	NOT A	ACC ← ~ACC
111	SHR A	ACC ← ACC >> 1

۶- تکمیل پردازنده

طراحی‌های بخش ۳ تا ۵ را بهم متصل کنید تا پردازنده کامل شود.

۷- تکمیل کامپیوتر پایه

برای اتمام کامپیوتر پایه، پردازنده تکمیل شده در بخش ۶ را به بخش ۱ و ۲ متصل کنید.
با کمک جدول ارائه شده در بخش ۵، برنامه‌ای بنویسید که دو مقدار از ورودی خوانده و میانگین آن دو را در درگاه خروجی بنویسد. کد نهایی را در ROM قرار دهید.

$- بخش امتیازی

کامپیوتر پایه‌ای که به صورت structural طراحی کردید را بهینه کرده و به صورت رفتاری توصیف کنید.

** در هر بخش برای اطمینان از صحت اجرای کدهای نوشته شده می‌توانید با نوشتن فایل آزمون مناسب عملکرد آن را ارزیابی کنید.

مراجع مطالعه/ضمایم:

موضوع: طراحی واحد کنترل

شماره آزمایش: ۱۰

عنوان: طراحی واحد کنترل سیم‌بندی شده(Hard wired Control Unit)

هدف:

آشنایی با طراحی واحد کنترل سیم بندی شده

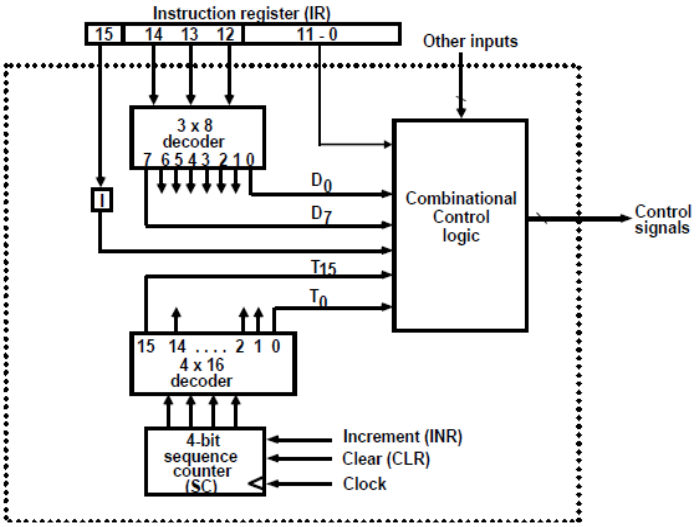
آمادگی پیش از آزمایش:

بخش تشریح کامل کامپیوتر (بخش۹، فصل۵) که در مراجع آزمایش به آن اشاره شده است را جهت راهنمایی مطالعه کنید و بر اساس جدول زیر ساختار گیتی ورودی‌های کنترل ثبات‌های AR و PC را تشریح کنید.

Fetch	$R'T_0$:	$AR \leftarrow PC$
	$R'T_1$:	$IR \leftarrow M[AR],\quad PC \leftarrow PC + 1$
Decode	$R'T_2$:	$D_0, \ldots, D_7 \leftarrow \text{Decode } IR(12\text{–}14),$
		$AR \leftarrow IR(0\text{–}11),\quad I \leftarrow IR(15)$
Indirect	$D_7'IT_3$:	$AR \leftarrow M[AR]$
Interrupt:		
$T_0'T_1'T_2'(IEN)(FGI + FGO)$:		$R \leftarrow 1$
	RT_0:	$AR \leftarrow 0,\quad TR \leftarrow PC$
	RT_1:	$M[AR] \leftarrow TR,\quad PC \leftarrow 0$
	RT_2:	$PC \leftarrow PC + 1,\quad IEN \leftarrow 0,\quad R \leftarrow 0,\quad SC \leftarrow 0$
Memory-reference:		
AND	D_0T_4:	$DR \leftarrow M[AR]$
	D_0T_5:	$AC \leftarrow AC \wedge DR,\quad SC \leftarrow 0$
ADD	D_1T_4:	$DR \leftarrow M[AR]$
	D_1T_5:	$AC \leftarrow AC + DR,\quad E \leftarrow C_{out},\quad SC \leftarrow 0$
LDA	D_2T_4:	$DR \leftarrow M[AR]$
	D_2T_5:	$AC \leftarrow DR,\quad SC \leftarrow 0$
STA	D_3T_4:	$M[AR] \leftarrow AC,\quad SC \leftarrow 0$
BUN	D_4T_4:	$PC \leftarrow AR,\quad SC \leftarrow 0$
BSA	D_5T_4:	$M[AR] \leftarrow PC,\quad AR \leftarrow AR + 1$
	D_5T_5:	$PC \leftarrow AR,\quad SC \leftarrow 0$
ISZ	D_6T_4:	$DR \leftarrow M[AR]$
	D_6T_5:	$DR \leftarrow DR + 1$
	D_6T_6:	$M[AR] \leftarrow DR,\quad \text{if } (DR = 0) \text{ then } (PC \leftarrow PC + 1),\quad SC \leftarrow 0$
Register-reference:		
	$D_7I'T_3 = r$ (common to all register-reference instructions)	
	$IR(i) = B_i\ (i = 0, 1, 2, \ldots, 11)$	
	r:	$SC \leftarrow 0$
CLA	rB_{11}:	$AC \leftarrow 0$
CLE	rB_{10}:	$E \leftarrow 0$
CMA	rB_9:	$AC \leftarrow \overline{AC}$
CME	rB_8:	$E \leftarrow \overline{E}$
CIR	rB_7:	$AC \leftarrow \text{shr } AC,\quad AC(15) \leftarrow E,\quad E \leftarrow AC(0)$
CIL	rB_6:	$AC \leftarrow \text{shl } AC,\quad AC(0) \leftarrow E,\quad E \leftarrow AC(15)$
INC	rB_5:	$AC \leftarrow AC + 1$
SPA	rB_4:	$\text{If } (AC(15) = 0) \text{ then } (PC \leftarrow PC + 1)$
SNA	rB_3:	$\text{If } (AC(15) = 1) \text{ then } (PC \leftarrow PC + 1)$
SZA	rB_2:	$\text{If } (AC = 0) \text{ then } PC \leftarrow PC + 1)$
SZE	rB_1:	$\text{If } (E = 0) \text{ then } (PC \leftarrow PC + 1)$
HLT	rB_0:	$S \leftarrow 0$

شرح آزمایش:

وظیفه واحد کنترل در یک کامپیوتر، دنبال کردن دنباله‌ای از ریز دستورالعمل‌هاست. که دستورالعمل های زبان ماشین را به سیگنال-های کنترلی برای کامپیوتر تبدیل می‌کند. شکل زیر شمای کلی واحد کنترل کامپیوتر پایه را نشان می‌دهد.



شکل۱: واحد کنترل کامپیوتر

این واحد به دو صورت می‌تواند طراحی شود، واحد کنترل سیم‌کشی شده که برای تولید سیگنال‌های کنترلی از مدارهای ترتیبی و ترکیبی استفاده می‌کند و واحد کنترل ریزبرنامه نویسی شده که متشکل از یک حافظه‌ی کنترل است که شامل ریزبرنامه‌هایی است که سیگنال‌های کنترلی لازم را فعال می‌کند.

در این آزمایش واحد کنترلی بر اساس مجموعه دستورالعمل‌های داده شده از سوی مدرس آزمایشگاه، طراحی و پیاده‌سازی گردد.

خروجی‌های مورد انتظار آزمایش:

یک دستورالعمل ۱۶ بیتی را بعنوان ورودی انتخاب نموده و سیگنال‌های کنترلی که واحد کنترل تولید می‌کند را در شبیه‌ساز نشان دهید.

مراجع مطالعه/ضمایم:

[1] Mano, Kime, *Logic and Computer Design Fundamentals*, Prentice-Hall, 2004, chapter 5.

موضوع:	خط لوله
شماره آزمایش:	۱۱
عنوان:	پیاده‌سازی خط لوله و بررسی تسریع آن
هدف:	

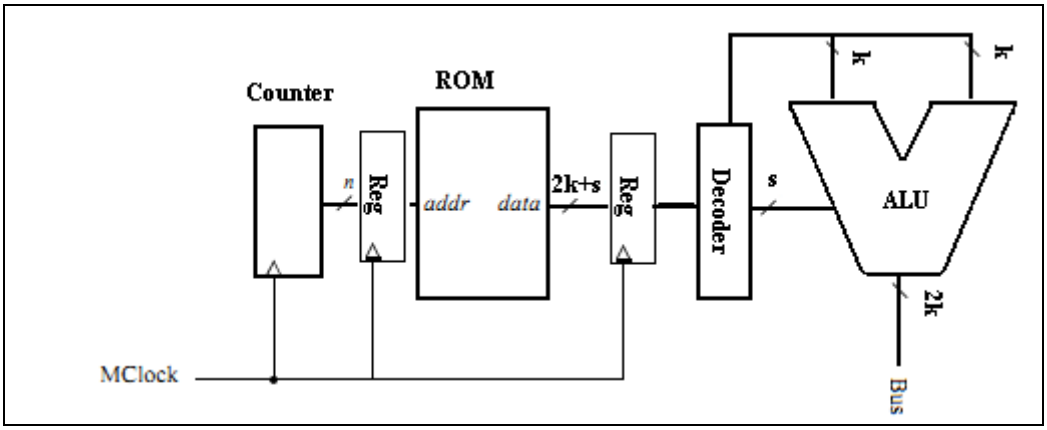
هدف از این آزمایش، آشنایی با روش خط‌لوله در تسریع سازی مدارات دیجیتالی است.

آمادگی پیش از آزمایش:

مفهوم خط‌لوله را با ذکر یک مثال توضیح دهید؛ بکارگیری این مفهوم در مدارهای دیجیتال با چه هدفی است، چگونه انجام می‌پذیرد و با چه محدودیت‌هایی مواجه است؟

شرح آزمایش:

استفاده از خط لوله برای موازی‌سازی پردازنده‌ها در سطح دستورالعمل به منظور افزایش کارایی پردازنده‌ها راهکاری مؤثر است. شکل زیر یک نمودار بلوکی برای خط لوله ساده را نشان می‌دهد.



شکل ۱- نمایی از یک خط لوله ۳ مرحله‌ای

۱) مداری که حاصل ضرب a*b*c*data را بدون ساختار خط‌لوله انجام می‌دهد، در زیر نمایش داده شده است. آن را با خط‌لوله دو مرحله‌ای پیاده‌سازی کنید. حداکثر فرکانس کاری در هر دو حالت را مقایسه کنید.

```
Process (clock)
Begin
If rising_edge (clock)then
--multiplication is done in a single stage.
      result <= a * b * c * data;
end If;
End Process;
```

۲) خط‌لوله‌ای که دارای پنج مرحله برای ضرب دو ماتریس ۴×۴ است را با زبان VHDL طراحی و توصیف کنید. حداکثر فرکانس کاری در هر دو حالت را مقایسه کنید.

خروجی‌های مورد انتظار آزمایش:

- کدهای VHDL مربوطه به همراه فایل‌های TestBench ارایه شود.
- حداکثر فرکانس کاری قابل اعمال به مدار، برای حالات مختلف از طریق گزارشات خروجی ابزار مشاهده شده و مقایسه گردد.
- در مورد تمرین دوم ابتدا چهار مرحله خط‌لوله طراحی شود و سپس کدهای توصیفی آن و درستی عملکرد آن را توسط شبیه‌ساز نشان دهید.

مراجع مطالعه/ضمایم:

[1] D. Patterson, J. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, Morgan Kaufmann Publishers, Inc., Third edition, 2007.

موضوع: ورودی/خروجی

شماره آزمایش: ۱۲

عنوان: پیاده‌سازی ارسال/دریافت ناهمگام در گذرگاه‌های یکطرفه و دوطرفه

هدف:

آشنایی با پیاده‌سازی انواع روش‌های ارسال/دریافت ناهمگام داده‌ها روی گذرگاه‌های یکطرفه و دوطرفه

آمادگی پیش از آزمایش:

مطالعه مواردی که در بخش مراجع آزمایش آورده شده است و انجام برخی شبیه‌سازی‌های مورد نظر.

شرح آزمایش:

در سیستم‌های پردازشی، انتقال داده‌ها میان پردازنده و واحدهای حافظه‌ای و ورودی/خروجی‌ها اهمیت بالایی دارد. گذرگاه (باس) یکی از مهمترین ارتباطاتی است که چندین گیرنده و فرستنده داده می‌توانند به آن دسترسی داشته باشند. بنابر این لازم است در هر لحظه از زمان تنها یک فرستنده روی گذرگاه داده قرار دهد و سایر واحدها تنها می‌توانند آن را دریافت کنند.

در این آزمایش ابتدا یک گذرگاه دو طرفه ساده پیاده‌سازی می‌شود و در ادامه آن برخی از پروتکل‌های همگام‌سازی انتقال اطلاعات و داوری (Arbitration) بررسی می‌شود.

الف) پیاده‌سازی گذرگاه داده دو طرفه

گذرگاه داده دو طرفه که قابلیت نوشتن و خواندن روی آن وجود دارد با سیگنال‌های کنترلی clock و output enable قابل پیاده‌سازی است.

```
ENTITY bidir IS
   PORT(
      bidir   : INOUT STD_LOGIC_VECTOR (7 DOWNTO 0);
      oe, clk : IN STD_LOGIC;
      inp     : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
      outp    : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
END bidir;
```

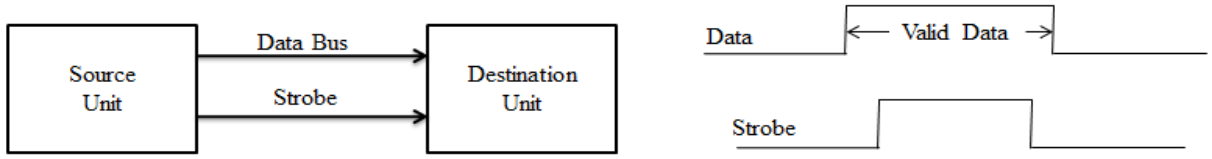
نمونه‌ای از این گذرگاه ۸ بیتی را با زبان VHDL پیاده‌سازی نمایید.

ب) انتقال داده ناهمگام

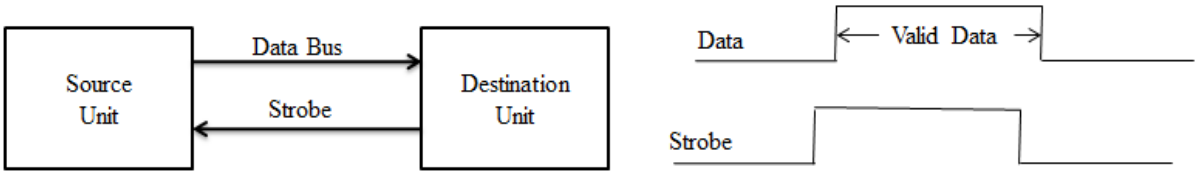
انتقال داده میان یک گیرنده و فرستنده بدون استفاده از clock را انتقال ناهمگام می‌نامند. برای هماهنگی میان گیرنده و فرستنده دو روش متداول استفاده از سیگنال strobe و روش دست‌دهی وجود دارد. که در ادامه بررسی می‌شوند.

۱) استفاده از سیگنال کنترلی strobe

سیگنال strobe یکی از روش‌های هماهنگ‌سازی میان گیرنده و فرستنده داده است. در این روش تنها یک بیت سیگنال کنترلی به سیستم افزوده می‌شود. پیاده‌سازی این روش به دو شیوه "به ابتکار گیرنده" و "به ابتکار فرستنده" صورت می‌گیرد. شکل ۱ و ۲ نمودار این دو روش به همراه زمان‌بندی سیگنال‌ها را نشان می‌دهد.



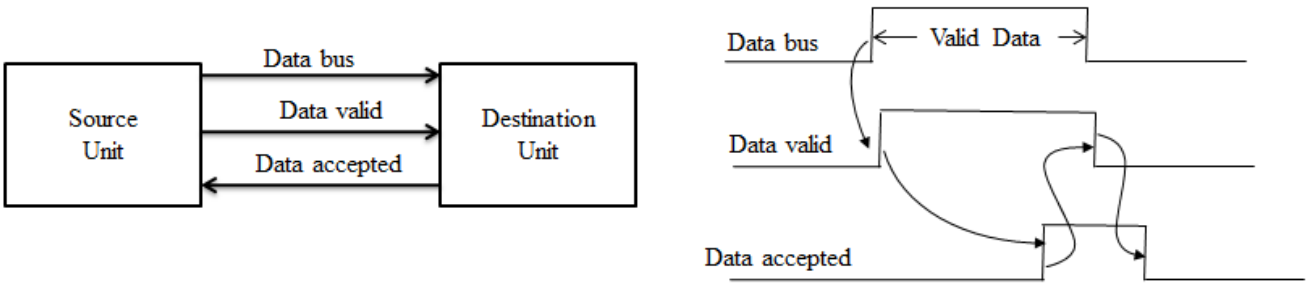
شکل۱- انتقال داده با سیگنال strobe به ابتکار فرستنده



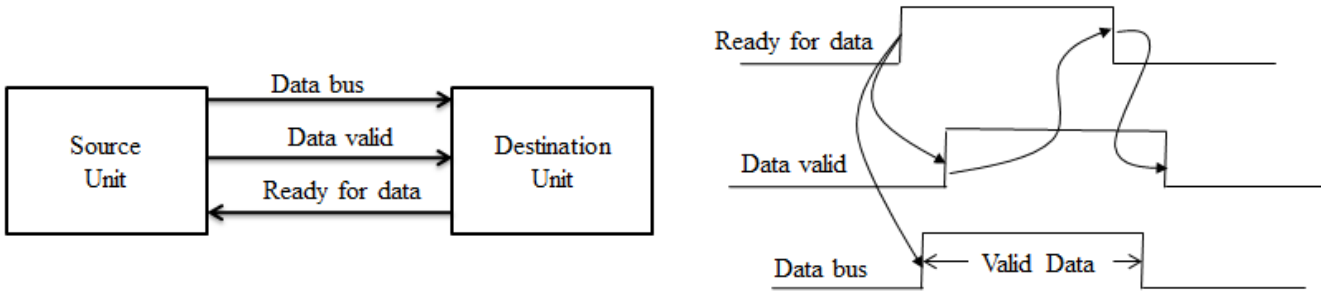
شکل۲- انتقال داده با سیگنالstrobe به ابتکار گیرنده

۲) روش دست‌دهی

روش دست‌دهی یکی دیگر از روش‌های متداول انتقال غیرهمگام داده است که با استفاده از دو سیگنال تک بیتی Data valid و Data accepted سعی می‌کند میان گیرنده و فرستنده هماهنگی ایجاد کند. پیاده‌سازی این روش نیز به دو شیوه "به ابتکار گیرنده" و "به ابتکار فرستنده" صورت می‌گیرد. شکل ۳ و ۴ نمودار این دو روش به همراه زمان‌بندی سیگنال‌ها را نشان می‌دهد.



شکل۳- انتقال داده با روش دست‌دهی به ابتکار فرستنده



شکل۴- انتقال داده با روش دست‌دهی به ابتکار گیرنده

خروجی‌های مورد انتظار آزمایش:

لازم است هر گروه از دانشجویان از هریک از بخش‌های (الف) و (ب) را با استفاده از زبان توصیف سخت‌افزار پیاده‌سازی و شبیه‌سازی نمایند و در رابطه با زمان‌بندی سیگنال‌ها تحلیل ارائه دهند.

هر یک از گروه‌ها می‌توانند دو entity جداگانه با عملکرد درونی انتخابی به عنوان گیرنده و فرستنده پیاده‌سازی کنند. واحد فرستنده در زمان‌های دلخواه، اطلاعات خود را برای گیرنده ارسال نمایند. واحد گیرنده نیز پردازش درونی خود را روی داده‌های دریافتی ارسال می‌کند.

مراجع مطالعه/ضمایم:

[1] M. Morris Mano, *Computer system architecture*, Prentice-Hall, 3rd edition, Chapter 11.

موضوع: ورودی/خروجی

شماره آزمایش: ۱۳

عنوان: پیاده‌سازی مکانیزم‌های داوری گذرگاه

هدف:

آشنایی با تداخل در ارتباطات مبتنی بر گذرگاه و استفاده از مکانیزم‌های داوری برای رفع تداخل همزمان

آمادگی پیش از آزمایش:

مطالعه مواردی که در بخش مراجع آزمایش به آن اشاره شده است و انجام برخی شبیه‌سازی‌های مورد نظر.

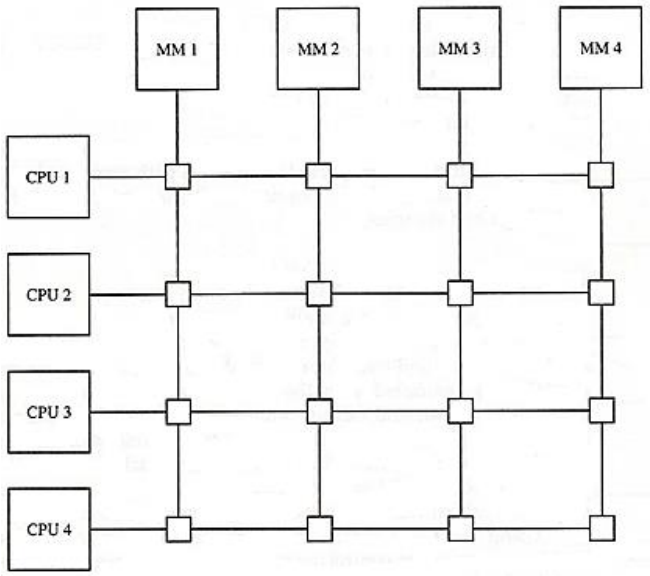
شرح آزمایش:

مکانیزم‌های داوری در گذرگاه مشترک:

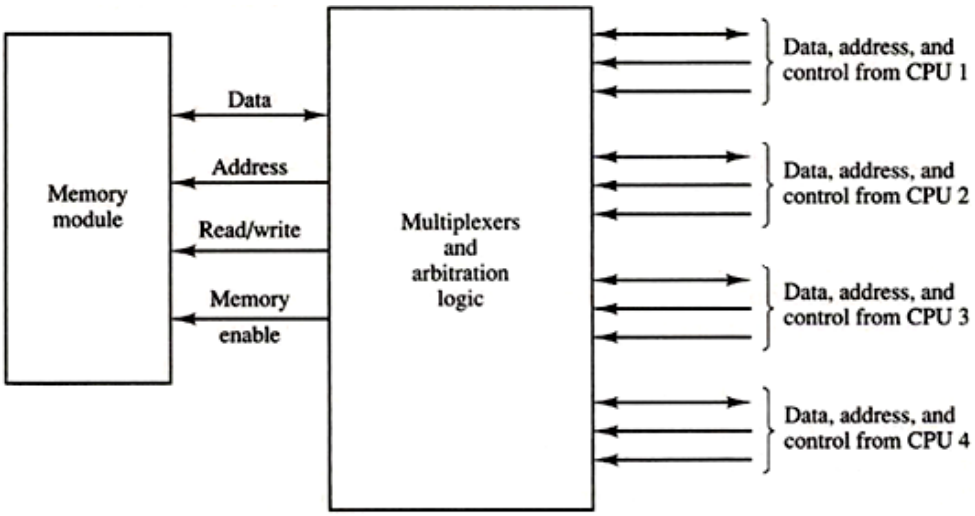
در بسیاری از سیستم‌های محاسباتی، چندین فرستنده و گیرنده داده به طور مشترک از یک گذرگاه داده استفاده می‌کنند؛ بنابراین لازم است برای جلوگیری از ایجاد اختلال میان داده‌هایی که که روی گذرگاه قرار می‌گیرد، در هر لحظه از زمان حداکثر یکی از سیستم‌ها روی گذرگاه داده قرار دهد. برای این منظور مکانیزم‌هایی جهت ایجاد نظم برای استفاده از باس مشترک طراحی شده است.

۱) استفاده از سوییچ تقاطعی (Crossbar switch)

روش سوییچ تقاطعی یکی از روش‌هایی است که برای ایجاد نظم ارتباط M پردازنده با N واحد حافظه مورد استفاده قرار می‌گیرد. در این روش با استفاده از چند واحد تسهیم‌کننده (Multiplexer) و مدارات منطقی می‌توان چند ارتباط همزمان میان یک زوج پردازنده و واحد حافظه را ایجاد کرد. شکل ۱ بلوک سوییچ تقاطعی ۴×۴ را نشان می‌دهد و شکل ۲ جزئیات بیشتری از سیگنال‌های کنترلی مورد نیاز برای ارتباط ۴ پردازنده با یک واحد حافظه مشترک به وسیله یک سوییچ تقاطعی را نشان می‌دهد.



شکل۱- یک سوییچ تقاطعی نوعی4*4



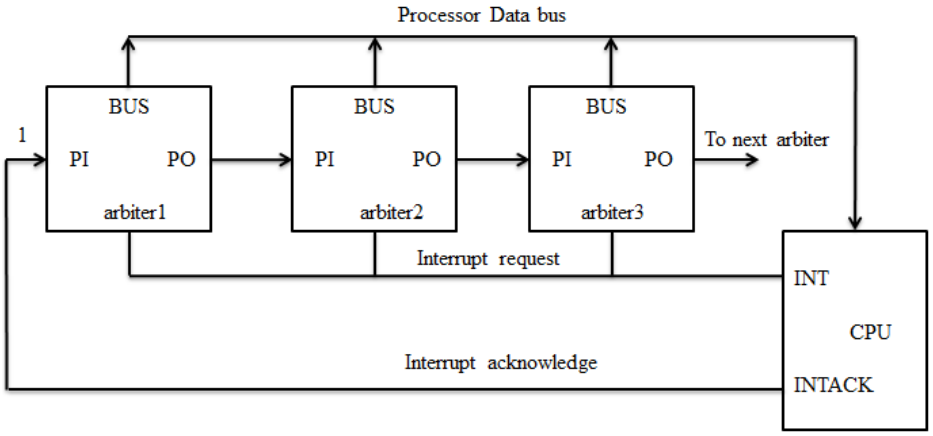
شکل۲- بلوک نمودار یک سوییچ تقاطعی با جزئیات بیشتر

۲) روش‌های اولویت‌دهی

یکی دیگر از روش‌هایی که می‌توان انتقال داده‌های همزمان را نظم‌دهی کرد استفاده از اولویت‌دهی به دستگاه‌ها است. به این ترتیب که هر یک از واحدهایی که به باس داده متصل هستند دارای یک اولویت هستند و اگر دو بخش همزمان داده‌ای برای قرار دادن روی باس داشته باشند، باس در اختیار واحد با اولویت بیشتر خواهد بود.

۲-۱) روش زنجیره‌ای (سری)

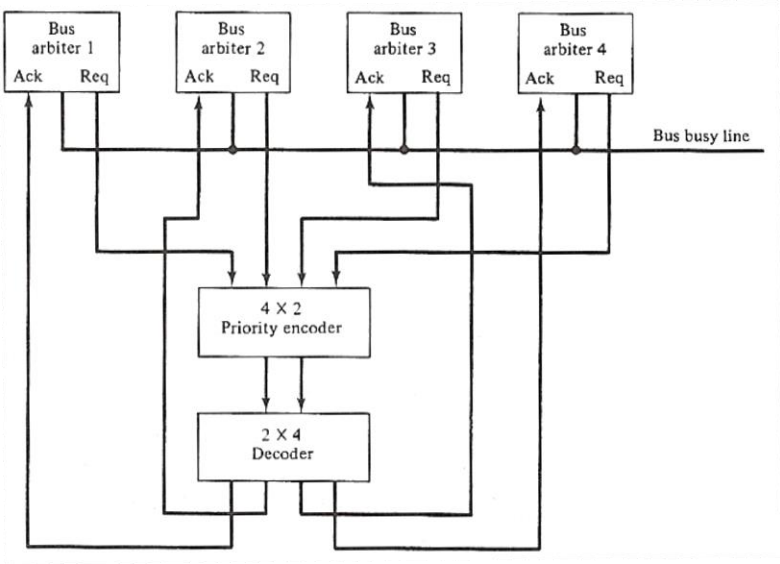
در این روش، چند واحد که به یک پردازنده متصل هستند در یک زنجیره در کنارهم قرار می‌گیرند. واحد با اولویت بیشتر در اولین محل قرار دارد و به ترتیب اولویت واحدها کم می‌شود. وقتی یکی از واحدها داده‌ای برای ارسال دارد، سیگنال وقفه پردازنده را فعال می‌کند. پردازنده پاسخ این وقفه را به اولین واحد ارسال می‌کند. در صورتی که این واحد دارای داده باشد آن را روی باس قرار خواهد داد و در غیر اینصورت به واحد بعد انتقال خواهد داد. شکل ۳ بلوک نمودار این روش را نشان می‌دهد.



شکل ۳- روش اولویت دهی زنجیره‌ای (daisy-chain)

۲-۲) روش موازی

در این روش واحدهایی دارای داده می‌توانند همزمان با هم درخواست ارسال دهند. این درخواست دریک کدگذار اولویت (Priority Encoder) وارد می‌شود و پس از تصمیم‌گیری راجع به واحدی که اولویت بیشتری دارد، پاسخ درخواست به واحد مربوطه ارسال می‌شود و پس از آن می‌تواند داده خود را روی باس قرار دهد. شکل۴ بلوک نمودار این روش را نشان می‌دهد.



شکل ۴- روش اولویت‌دهی موازی

خروجی‌های مورد انتظار آزمایش:

لازم است هر گروه از دانشجویان از هریک از بخش‌های (۱) و (۲) را با استفاده از زبان توصیف سخت‌افزار پیاده‌سازی و شبیه‌سازی نمایند و در رابطه با زمان‌بندی سیگنال‌ها تحلیل ارائه دهند.

هر یک از گروه‌ها می‌توانند سه یا چهار entity جداگانه با عملکرد درونی انتخابی به عنوان واحدهای متصل به باس پیاده سازی کنند. هر یک از واحدها در زمان‌های اختیاری درخواست باس به arbiter ارسال می‌کنند. در صورت پذیرش درخواست، واحد مورد نظر داده‌های تولیدی خود را با یک زمان‌بندی مشخص روی باس که خروجی اصلی entity است قرار خواهد داد.

مراجع مطالعه/ضمایم:

[1] M. Morris Mano, *Computer system architecture*, Prentice-Hall, 3rd edition, Chapter 11.

موضوع: محاسبات ممیز شناور

شماره آزمایش: ۱۴

عنوان: پیاده سازی عملیات جمع یا تفریق واحد ممیز شناور

هدف:

در این آزمایش هدف طراحی و پیاده‌سازی یک واحد محاسبات ساده برای انجام عملیات جمع و تفریق اعداد ممیز شناور به‌صورت سخت‌افزاری است.

آمادگی پیش از آزمایش:

مطالعه مواردی که در بخش مراجع آزمایش به آن اشاره شده است و انجام برخی شبیه‌سازی‌های مورد نظر الزامی است.

شرح آزمایش:

برای این منظور از استاندارد IEEE 754 که در آن نحوه نمایش اعداد ممیز شناور، نحوه انجام محاسبات و نحوه انجام اصلاحات بر روی نتایج شرح داده شده است، استفاده می‌شود. یک ALU ساده برای پیاده‌سازی عملیات زیر بر روی اعداد ممیز شناور بر اساس استاندارد IEEE 754 طراحی و پیاده‌سازی کنید.

الف) جمع ب) تفریق

مدارات طراحی شده را شبیه‌سازی کرده و در انتها بر روی FPGA پیاده‌سازی کنید، برای سادگی نمایش اعداد ۸ بیت در نظر بگیرید. لازم است خروجی این واحد نیز به‌صورت استاندارد باشد و نیز کل طرح قابل سنتز باشد. برای تست سیستم، یک برنامه تست که مقادیر عددی آنرا به‌صورت دستی تعیین کرده‌اید، بنویسید.

خروجی‌های مورد انتظار آزمایش:

با دادن اعداد ممیز شناور دلخواه، عملیات جمع و تفریق روی آن‌ها انجام شود.

مراجع مطالعه/ضمایم:

[1] Mano, Kime, *Logic and Computer Design Fundamentals*, Prentice-Hall, 2004, chapter 10.

موضوع: ارزیابی کارآیی

شماره آزمایش: ۱۵

عنوان: اندازه‌گیری پارامترهای IPC، CPI، MIPS و DMIPS یک پردازنده نوعی

هدف:

آشنایی با پارامترهای کارآیی و نحوه اندازه‌گیری آن‌ها

آمادگی پیش از آزمایش:

مطالعه منابع معرفی شده در انتهای آزمایش. در ابتدای جلسه سؤالات کتبی یا شفاهی کوتاه صورت خواهد گرفت.

شرح آزمایش:

یکی از روش‌های مقایسه میان دو سیستم پردازنده‌ای، ارزیابی کارایی آن‌هاست. برای این منظور پارامترهایی مانند تعداد دستورالعمل اجرا شده در واحد زمان(IPC)، زمان متوسط اجرای یک دستور العمل (CPI) تعریف شده است. برای قابل استناد بودن مقایسه‌های صورت گرفته، لازم است واحدهایی که با هم مقایسه می‌شوند برنامه‌های یکسان و استانداردی را اجرا نمایند. مجموعه برنامه‌های SPEC و MiBench از متداول‌ترین برنامه‌های استاندارد هستند که دسته اول برای پردازنده‌های Desktop و دسته دوم برای پردازنده-های Embedded مورد استفاده قرار می‌گیرند.

الف) آشنایی با برنامه‌های استاندارد

در این آزمایش لازم است هر گروه از دانشجویان ۴ نمونه از برنامه‌های مجموعه SPEC2000 یا SPEC2006 را روی سیستم خود اجرا نمایند و زمان اجرای آن را محاسبه کنند. پس از آن هر دو گروه نتایج به دست آمده خود را مقایسه کرده و مورد بررسی قرار دهند.

ب) محاسبه پارامترهای ارزیابی کارایی

در این آزمایش هر گروه از دانشجویان با هماهنگی مربی آزمایشگاه روشی برای محاسبه IPC، CPI، MIPS و DMIPS سیستم خود ارائه می‌دهند و پس از آن این پارامترها را برای سیستم خود محاسبه کرده و نتایج را با سایر گروه‌ها مقایسه می‌کنند.

ج) ارزیابی تأثیر برنامه‌نویسی چند نخی برای افزایش سرعت برنامه *

در این آزمایش هر گروه از دانشجویان ابتدا یک برنامه را به صورت تک نخی روی سیستم خود اجرا می‌کنند. در مرحله بعد دانشجویان تلاش می‌کنند همان برنامه را با تکنیک‌های برنامه نویسی چند نخی پیاده‌سازی و اجرا کنند و تأثیر موازی‌سازی در اجرای برنامه را گزارش نمایند.

د) استفاده از کتابخانه MPI برای برنامه‌نویسی موازی *

ه) استفاده از ابزار Simple Scalar برای ارزیابی کارایی یک سیستم نوعی *

خروجی‌های مورد انتظار آزمایش:

مقدارهای عددی بدست آمده برای هر یک از پارامترهای فوق در هر مرحله به همراه توضیح دقیق و فنی روشی که از آن استفاده کرده‌اید.

مراجع مطالعه/ضمایم:

[1] John. L. Hennessy and David A. Patterson, *Computer Architecture, A Quantitative Approach*, Prentice-Hall, 4th Edition, pp. 28 -37.

موضوع: حافظه‌ها

شماره آزمایش: ۱۶

عنوان: طراحی سلسله مراتب حافظه

هدف:

هدف از این آزمایش، آشنایی با سلسله مراتب حافظه و نحوه نگاشت آدرس و دریافت داده از RAM و Cache است.

آمادگی پیش از آزمایش:

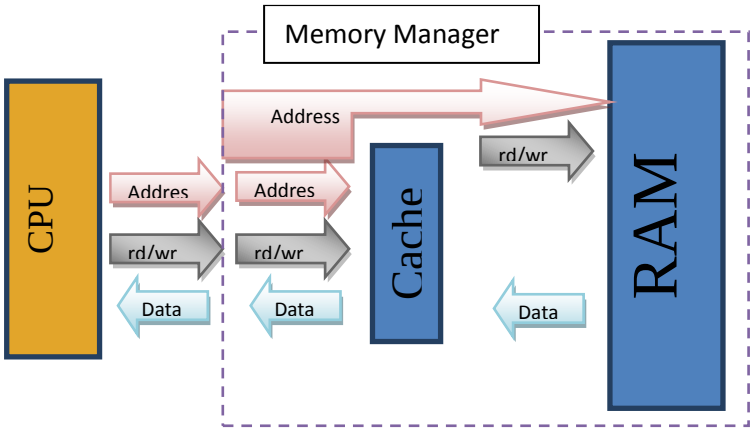
مطالعه مواردی که در بخش مراجع آزمایش به آن اشاره شده است و انجام برخی شبیه‌سازی‌های مورد نظر.

شرح آزمایش:

حافظه نهان (Cache) یکی از مهمترین راهکارهایی است که برای کاهش اختلاف سرعت پردازنده و حافظه RAM ارائه شده و طراحی می‌شود. این حافظه به صورت یک واحد میانی در کنار پردازنده قرار می‌گیرد و سعی می‌کند بخشی از داده‌هایی که پردازنده به صورت متناوب به آن نیاز دارد را در خود ذخیره کند تا در زمان نیاز پردازنده بتواند در زمان کوتاهی آن را فراهم نماید.

الف) چگونگی عملکرد حافظه سلسله مراتبی:

شکل زیر نمودار بلوکی یک حافظه سلسله مراتبی را نشان می‌دهد. مطابق این شکل، پردازنده آدرس داده مورد نظر خود را به همراه سیگنال خواندن/نوشتن به واحد مدیریت حافظه ارسال می‌کند. این واحد کنترل Cache و RAM را بر عهده دارد.



شکل ۱- دیاگرام بلوکی سلسله مراتب حافظه در ارتباط با پردازنده

در ابتدا آدرس‌ها به Cache داده می‌شوند و هر یک از حالات زیر امکان رخداد خواهند داشت:

۱. خواندن آدرسی که در Cache وجود دارد: در این حالت، داده مورد نظر بلافاصله برگردانده می‌شود.
۲. خواندن آدرسی که در Cache وجود ندارد: در این حالت، لازم است آدرس مورد نظر به RAM ارجاع شود و داده از RAM خواند شود. همچنین این داده باید درون Cache نیز به روزرسانی شود.

۳. نوشتن در آدرسی که در Cache وجود دارد: در این صورت داده جدید در Cache و RAM به طور همزمان جایگزین می‌شود. (مکانیزم Write Through)
۴. نوشتن در آدرسی که در Cache وجود ندارد: در این حالت داده مورد نظر تنها در RAM نوشته خواهد شد.

ب) طراحی سلسله مراتب حافظه:

با مشخصات داده شده در (الف) یک نمونه ساده از سلسله مراتب حافظه (Memory Hierarchy) طراحی و پیاده‌سازی کنید. طول آدرس پردازنده، سایز RAM و Cache و بلوک‌بندی را به گونه‌ای منطقی طراحی کند و تعداد بیت های بخش‌های Tag، Index و Offset را محاسبه نمایید. مکانیزم شرکت‌پذیری Cache را به صورت Direct map در نظر بگیرید.

خروجی‌های مورد انتظار آزمایش:

هریک از بخش‌های این آزمایش در قالب‌های زیر به ترتیب تحویل مدرس آزمایشگاه گردد:

- طراحی شماتیک طرح (محتوای توصیف) بر روی کاغذ
- پیاده‌سازی مدارها با استفاده از زبانهای توصیف
- شبیه‌سازی مدارهای توصیف شده و تهیه Testbench برای آن و نشان دادن درستی عملکرد آنها.

مراجع مطالعه/ضمایم:

[1] J. L. Hennessy and D. A. Patterson, “Computer Architecture, A Quantitative Approach” 4th edition, appendix C.

ضمیمه ۱

توصیف سیستم با VHDL

توصیف یک سیستم با VHDL : موجودیت سخت افزاری
Entity mobile is … End entity mobile;

- هر سیستم شامل دو جزء اصلی: ورودی/خروجی و بدنه سیستم
  - مثل: RAM، CPU، مودم، ماوس، کامپیوتر، موبایل و مانند آن.

توصیف یک سیستم با VHDL : پورت‌های ورودی و خروجی

Entity and_gate is
  Port(
  A, B: in bit;
  C : out bit
  );
End entity and_gate;

توصیف یک سیستم با VHDL : بدنه سیستم

Entity and_gate is
  Port(
  A, B: in bit;
  C : out bit
  );
End entity and_gate;
 	
Architecture behavioral of and_gate is
Begin
  C <= A and B;
End behavioral;

در مثال بالا:

And_gate	نام ماجولی که آن را توصیف می‌کنیم. این ماجول یک گیت AND است.
A, B	دو ورودی گیت AND
C	خروجی گیت AND
behavioral	سطحی که این کد را در آن توصیف می‌کنیم.
bit	یک نوع (type) است. مثل int در زبان C. سیگنال‌های از این نوع مقادیر '0' یا '1' می‌گیرند.
<=	عملگر مقداردهی است.

معرفی کتابخانه IEEE

توصیف یک سیستم با VHDL : بدنه سیستم

```
library IEEE;
use IEEE.std_logic_1164.all;

Entity and_gate is
   Port(
      A, B: in std_logic;
      C   : out std_logic
   );
End entity and_gate;

Architecture behavioral of and_gate is
Begin
   C <= A and B;
End behavioral;
```

استفاده از std_logic_vector

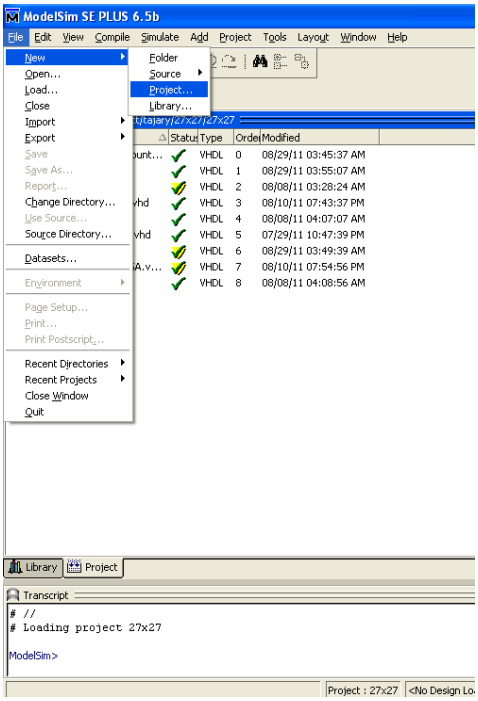
```
   Port(
      A: in  std_logic_vector(9 downto 0);
      C: out std_logic_vector (9 doento 0)
   );

Signal b: std_logic_vector(9 downto 0);
```

مقداردهی به std_logic_vector

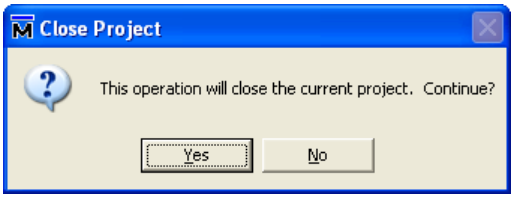
```
B(9 downto 1) <= B(8 downto 0); -- shift
B(0) <= '1';
B(2 downto 0) <= "110";
B <= A;
B(9 downto 8) <= A(2 downto 1);
B(4) <= B(5);
B <= (others => '0');
B(5 downto 0) <= A(9 downto 7) & "011";
```

ضمیمه ۲

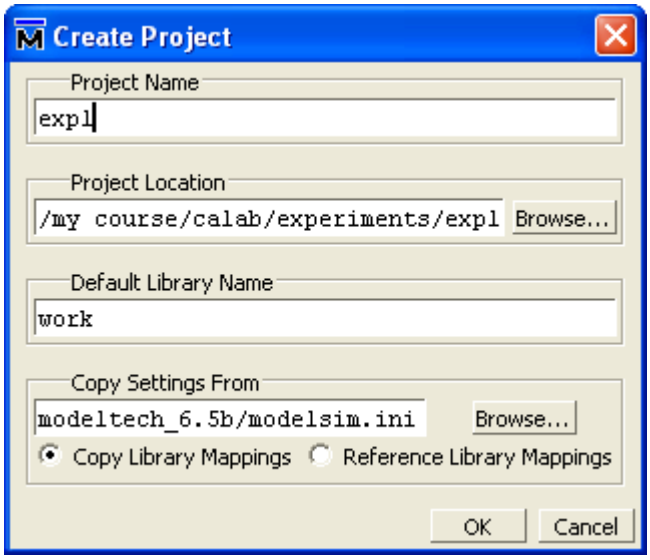
شبیه‌سازی با استفاده از ModelSim



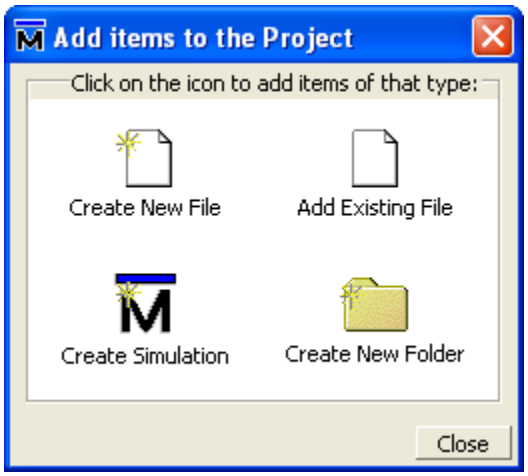
شکل ۱ ایجاد پروژه جدید



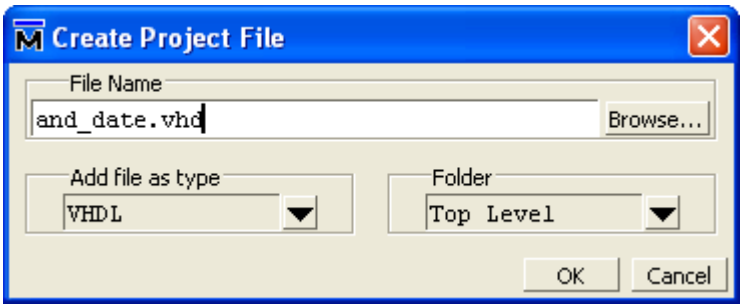
شکل ۲ پیغام هشدار بستن پروژه قبلی



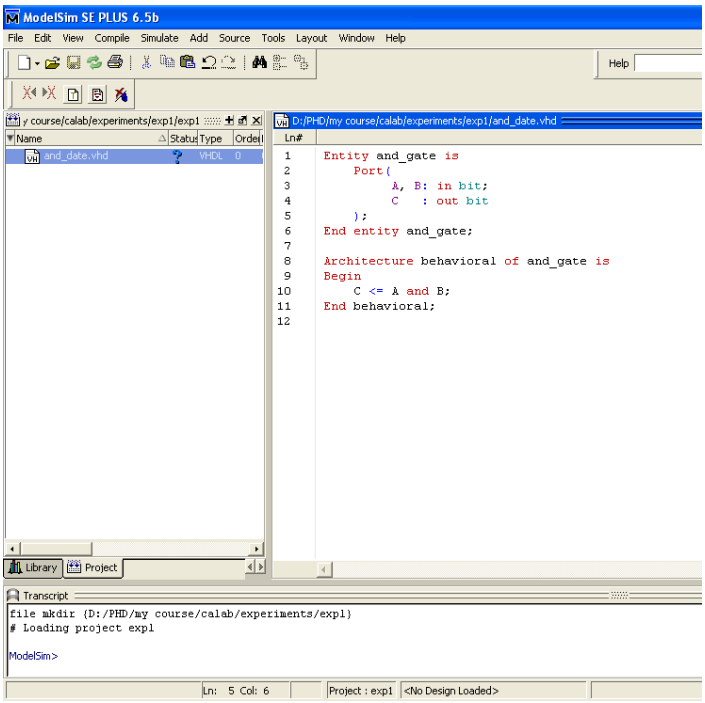
شکل ۳ تعیین نام و پوشه پروژه



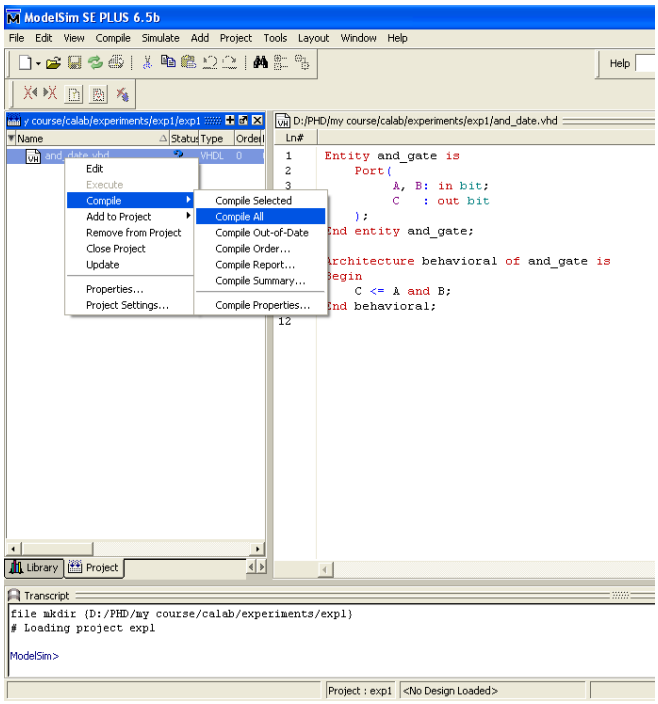
شکل ۴ افزودن فایل به پروژه



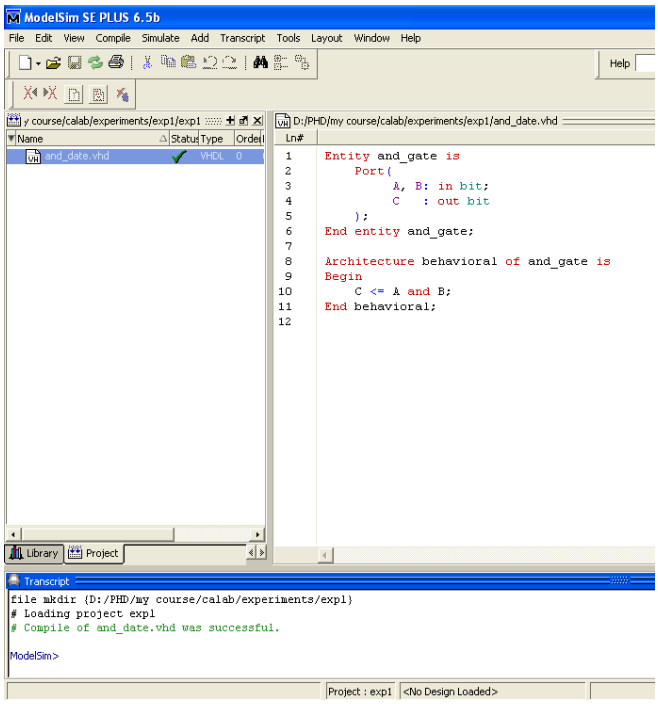
شکل ۵ تعیین نوع و نام فایل جدید



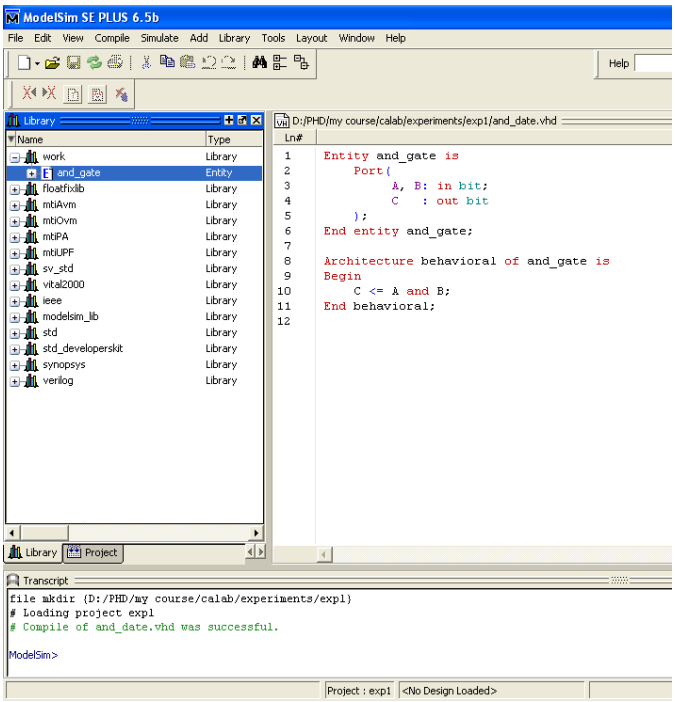
شکل ۶ نوشتن کد در ویرایشگر



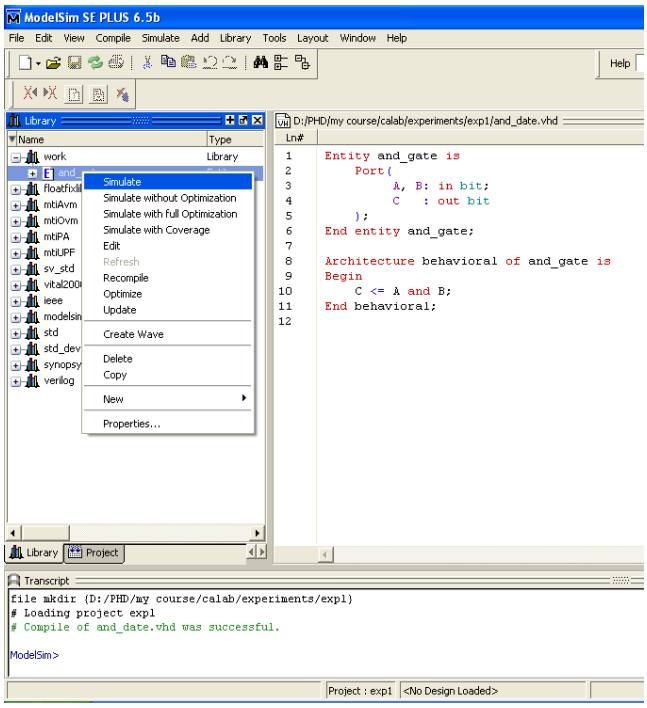
شکل ۷ منو کامپایل



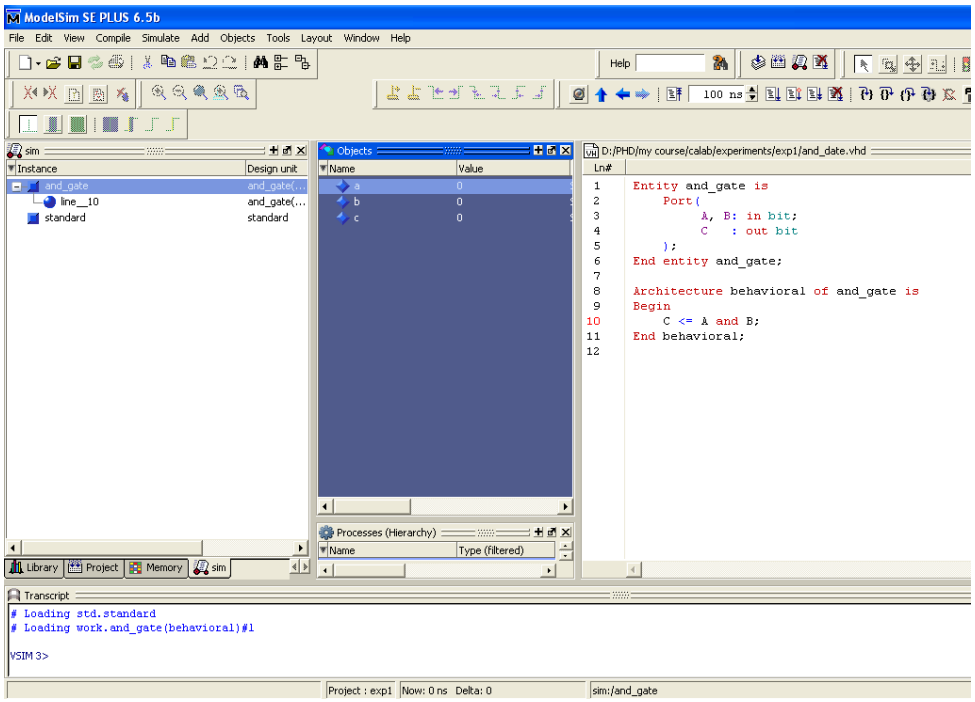
شکل ۸ نتیجه کامپایل



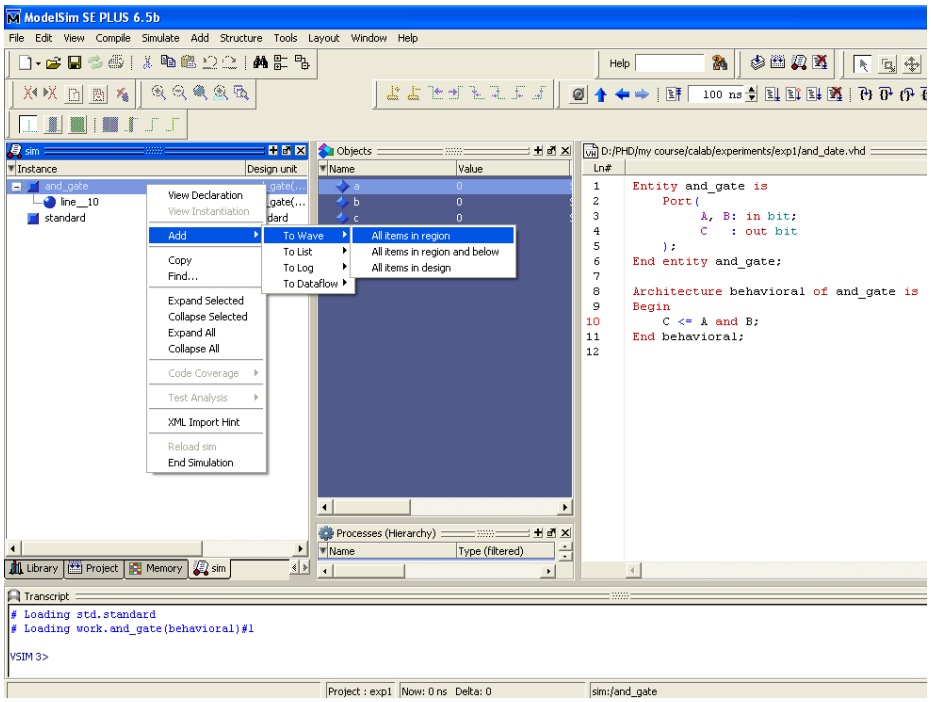
شکل ۹ بخش library برای شبیه‌سازی



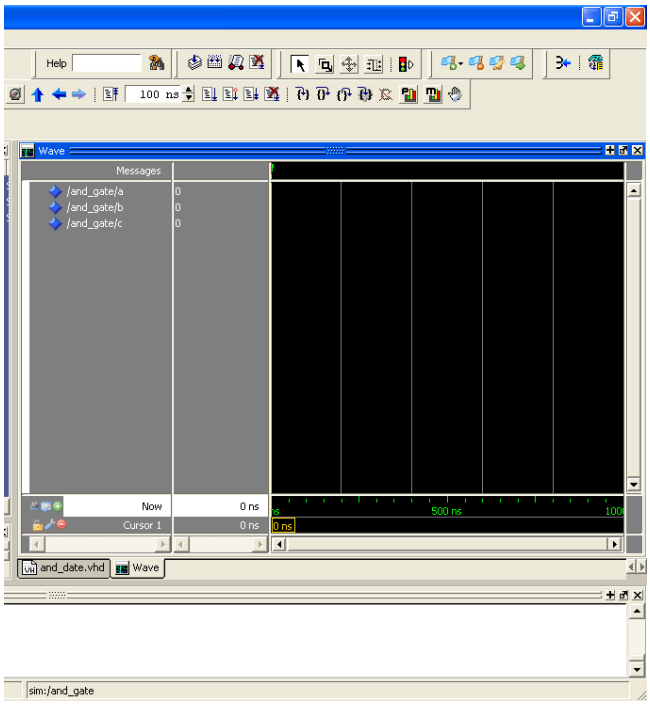
شکل ۱۰ منو شبیه‌سازی



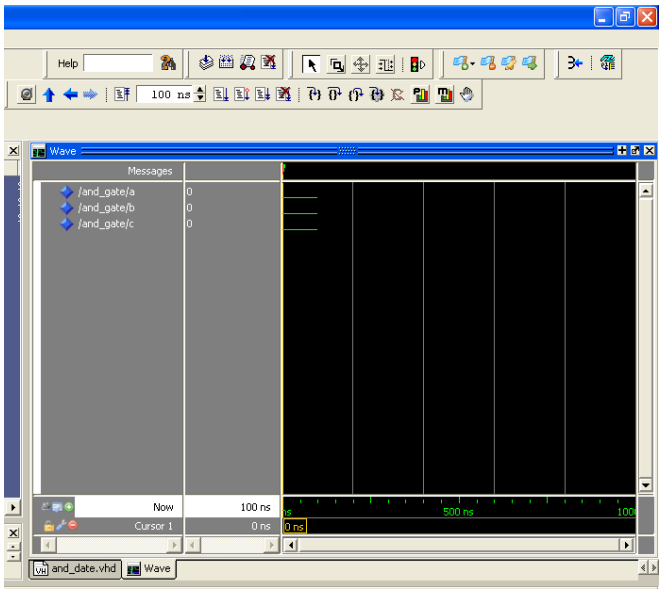
شکل ۱۱ نمای شروع شبیه‌سازی



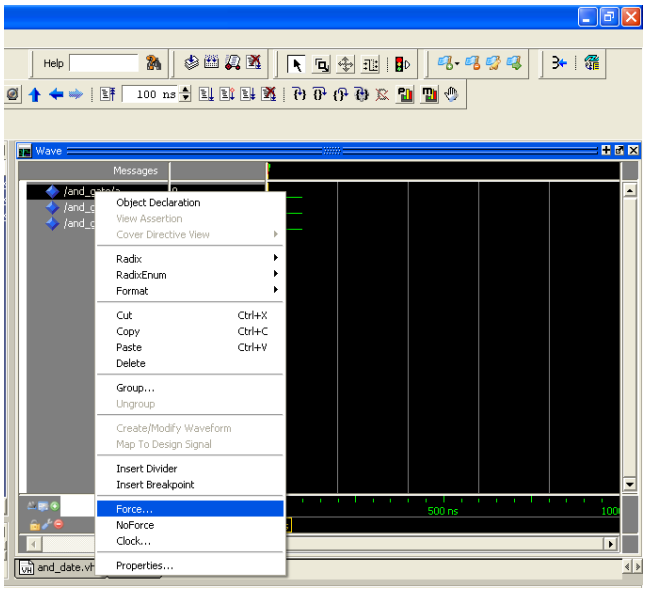
شکل ۱۲ افزودن سیگنال به پنجره wave



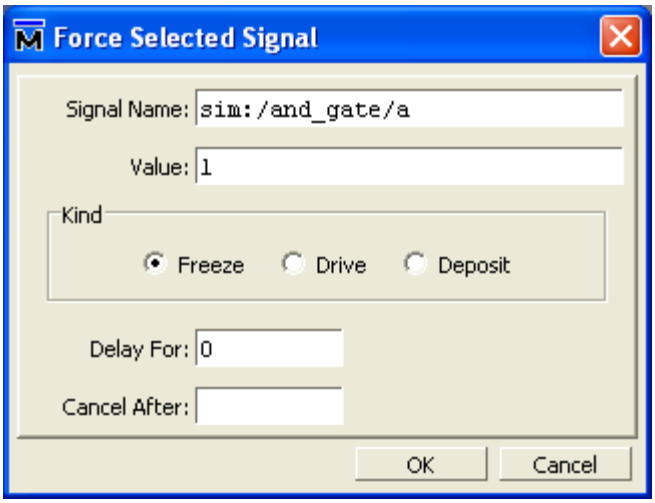
شکل ۱۳ سیگنال‌ها در پنجره wave



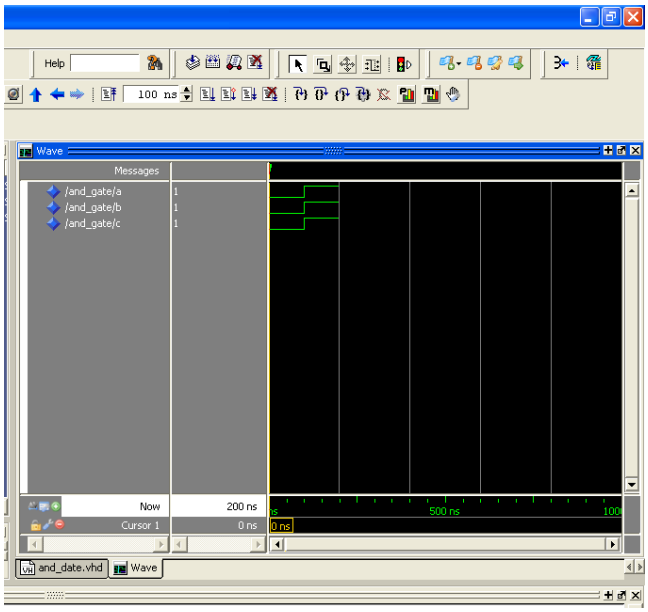
شکل ۱۴ اجرای شبیه‌سازی



شکل ۱۵ منوی force کردن سیگنال



شکل ۱۶ تعیین مقدار سیگنال در پنجره force

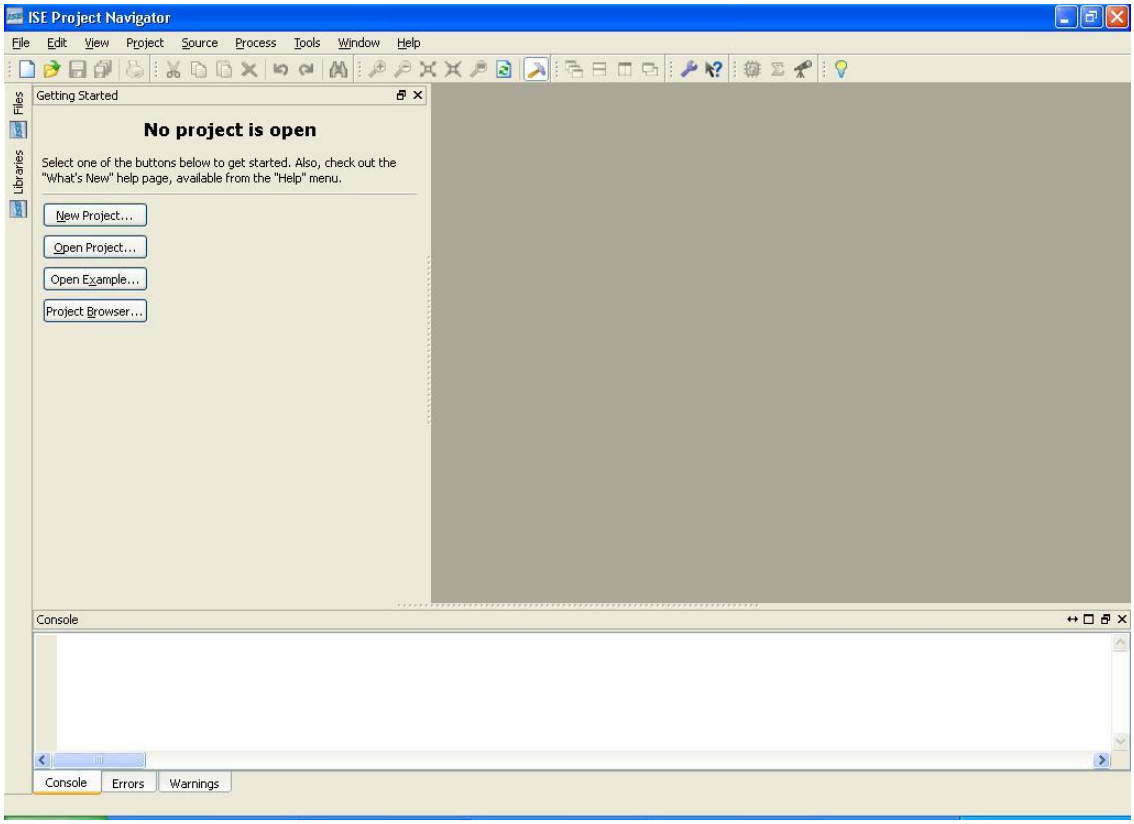


شکل ۱۷ اجرای شبیه‌سازی با مقادیر جدید

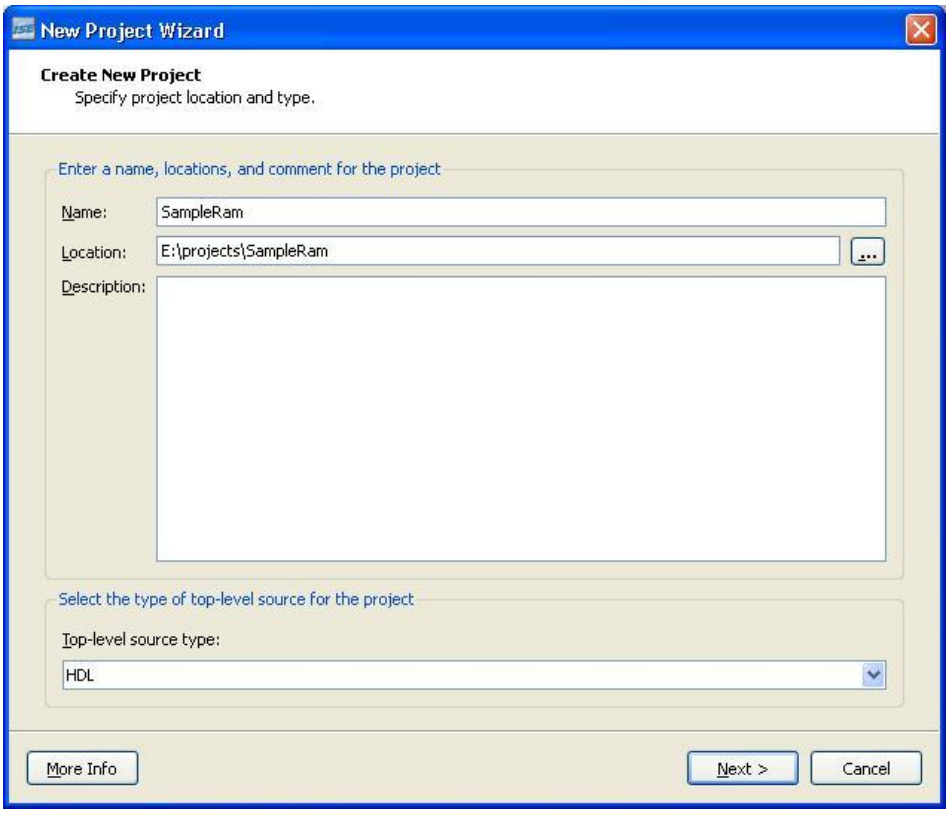
ضمیمه ۳

نحوه کارکردن با بورد FPGA

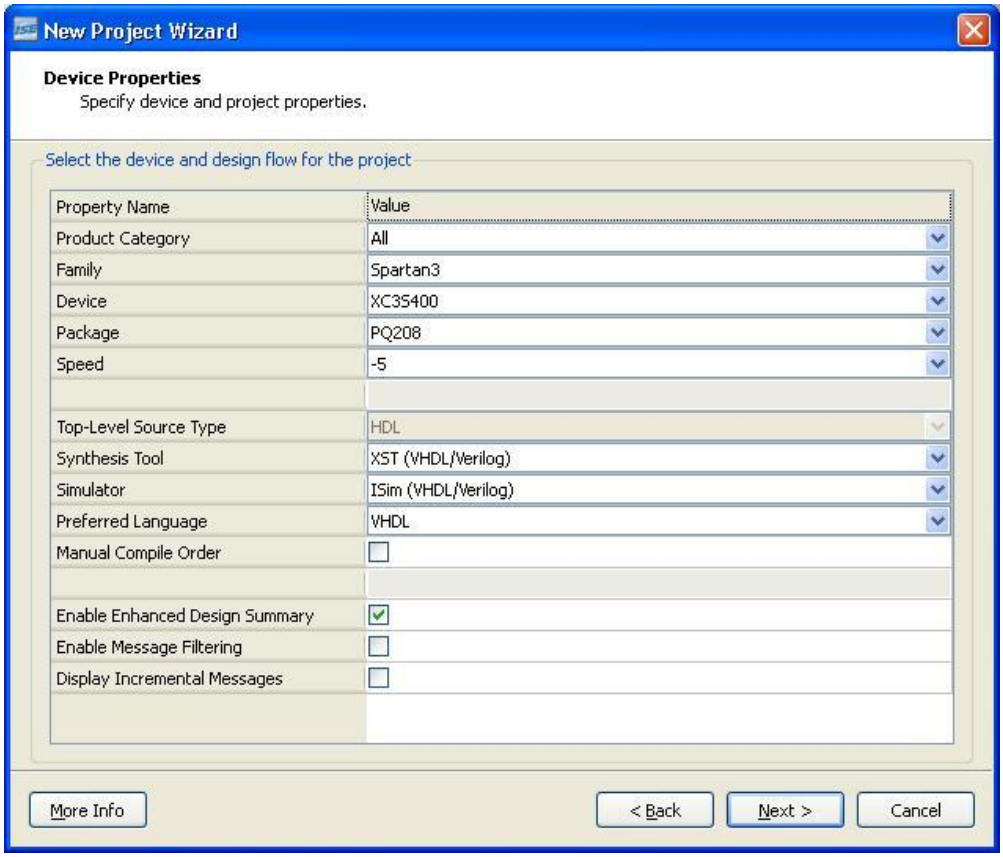
فاز اول: برنامه‌ریزی FPGA



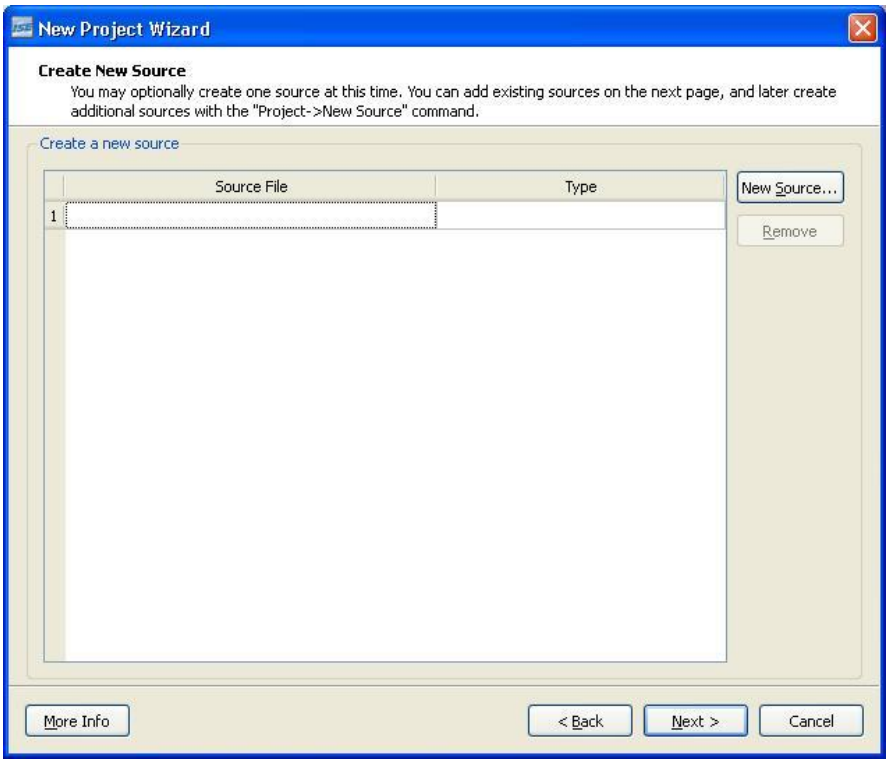
شکل ۱ صفحه شروع کار با ابزار ISE



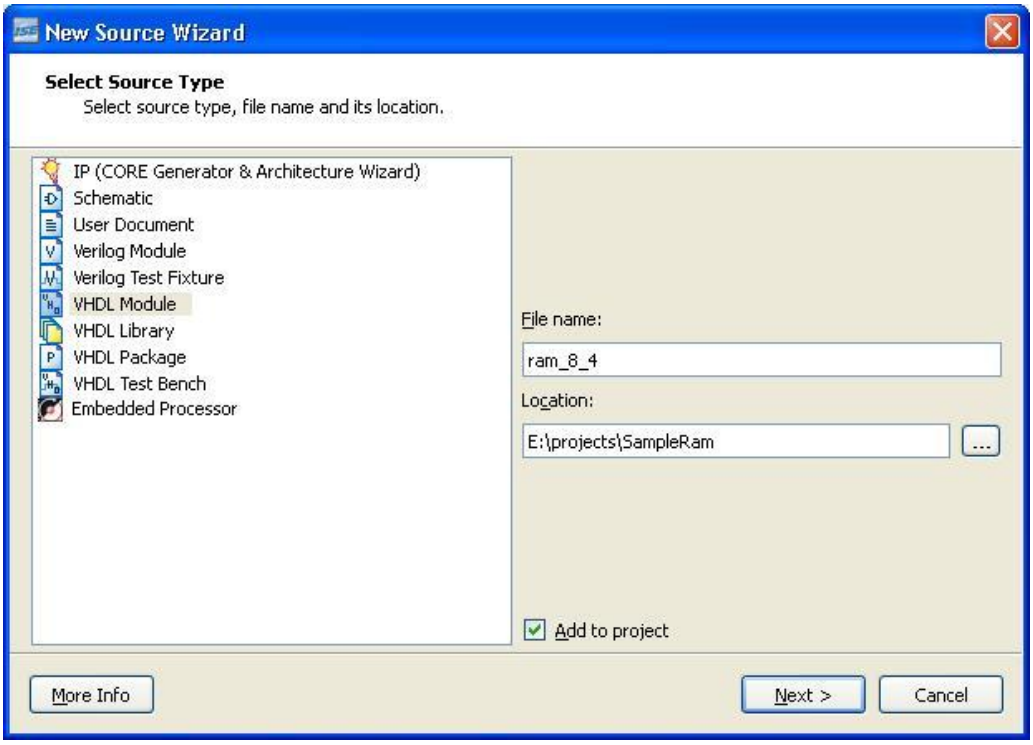
شکل ۲ ایجاد یک پروژه جدید



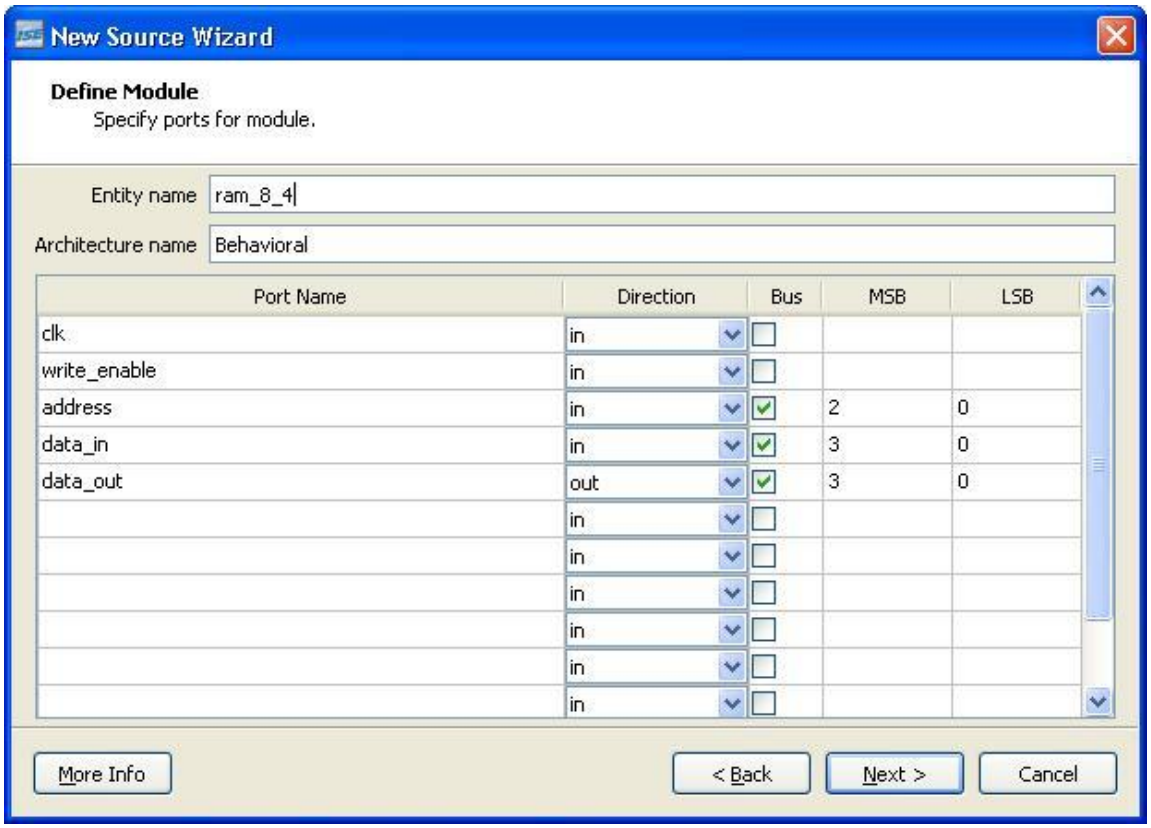
شکل ۳ تعیین ویژگی های FPGA



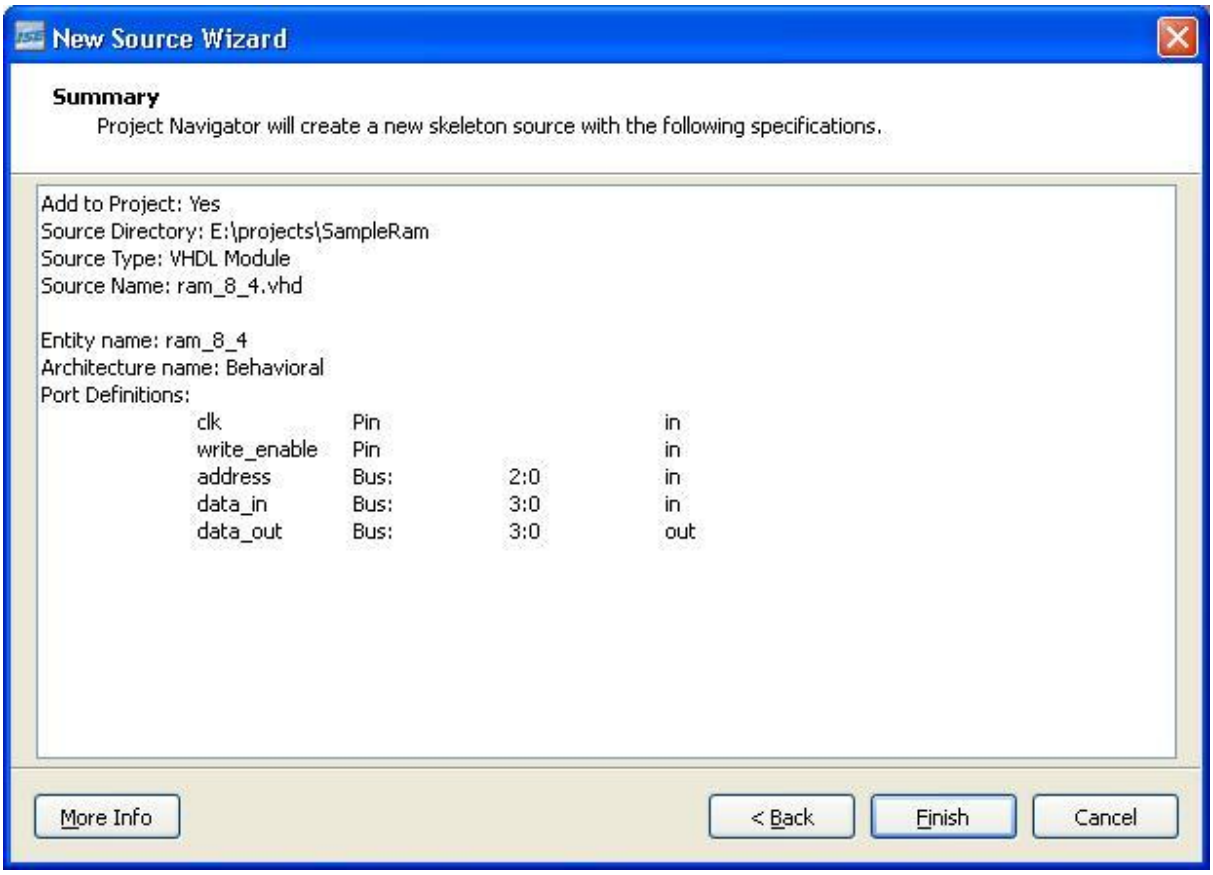
شکل ۴ ایجاد یک فایل برای entity



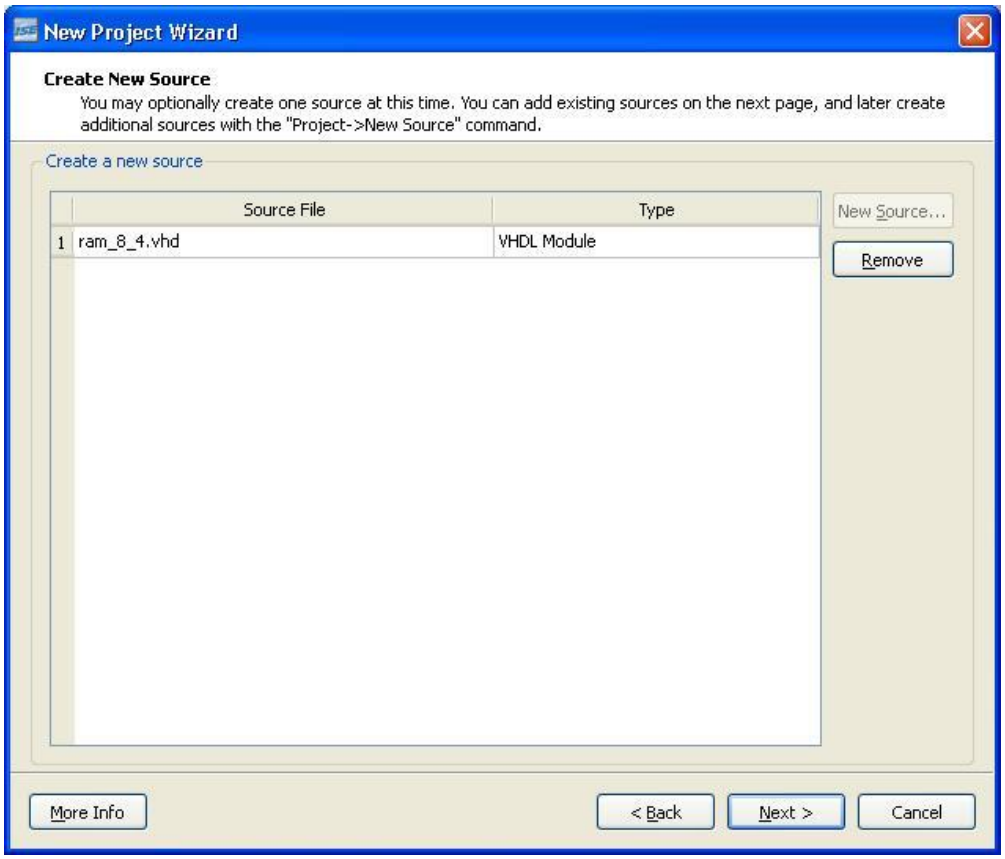
شکل ۵ تعیین نوع و نام فایل



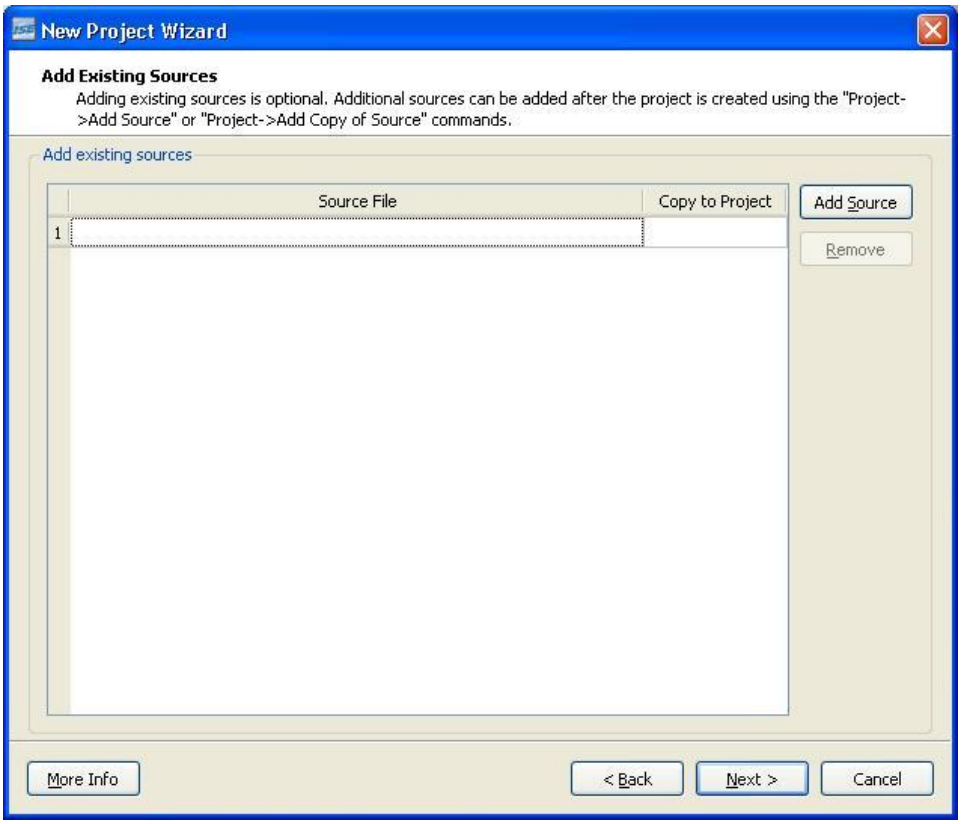
شکل ۶ تعیین پورت های ورودی و خروجی



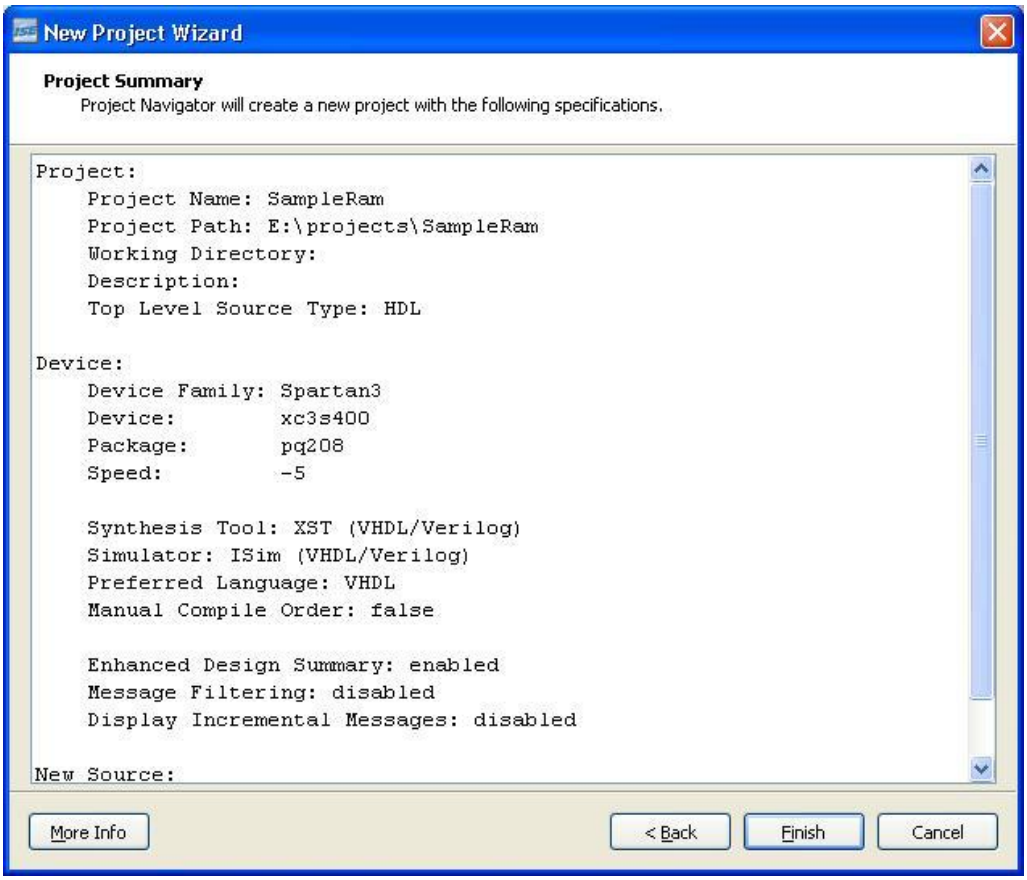
شکل ۷ خلاصه از ویژگی های فایل ایجاد شده



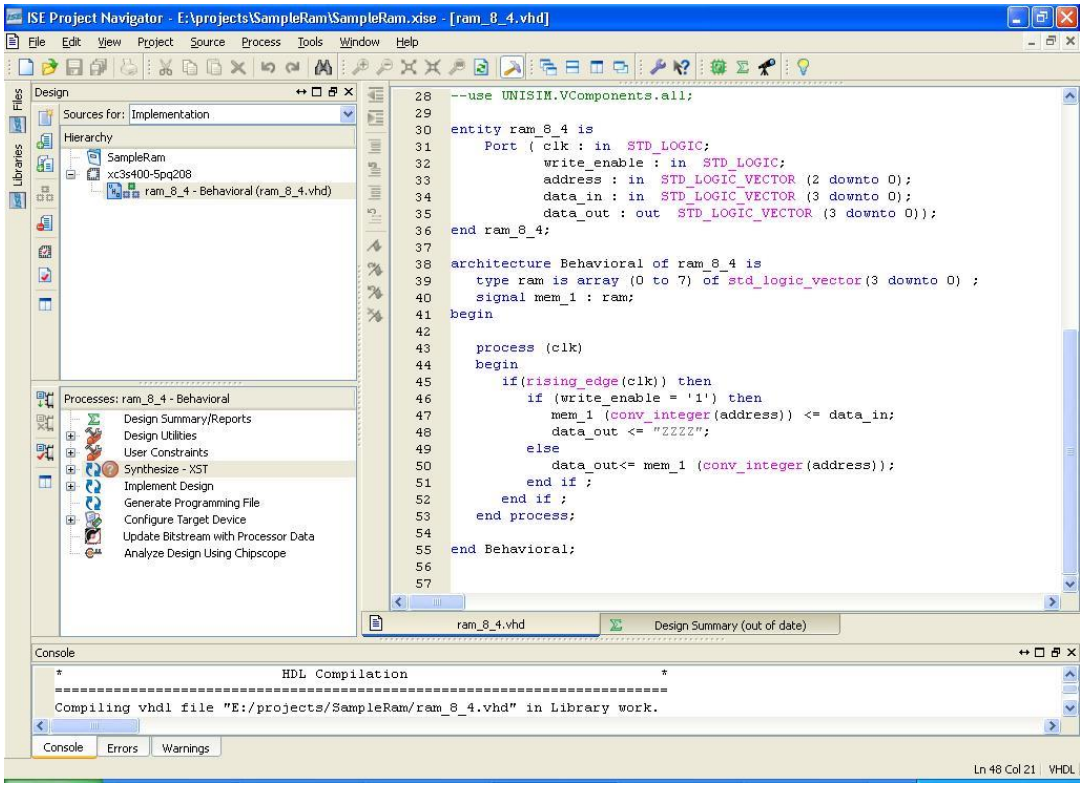
شکل ۸ لیست فایل های ایجاد شده



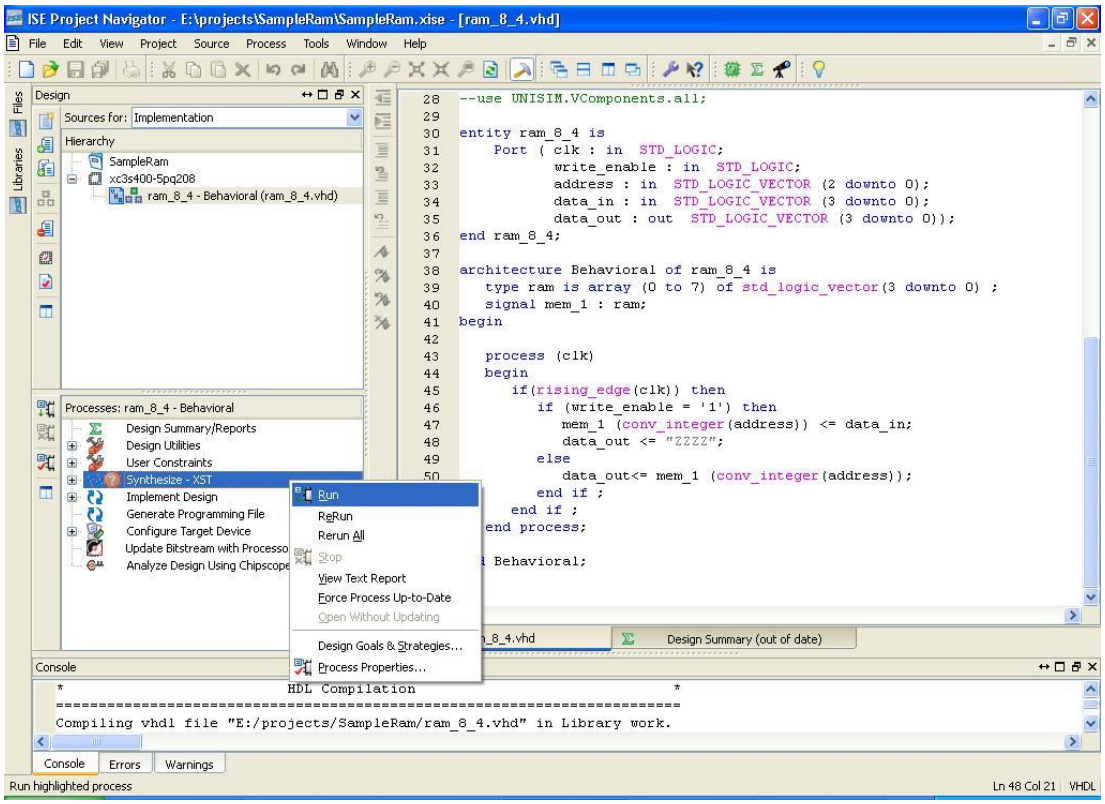
شکل ۹ افزودن فایل هایی که از قبل وجود دارند



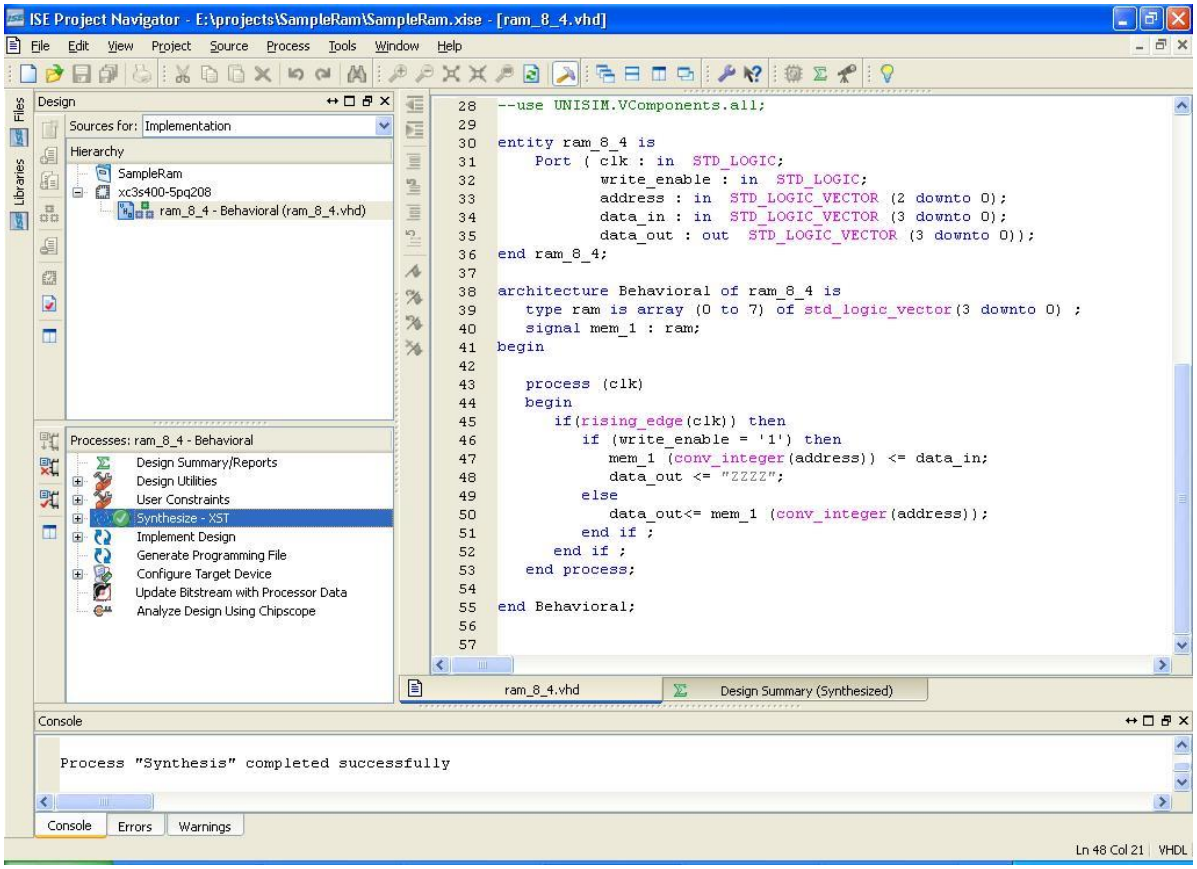
شکل ۱۰ خلاصه ای از ویژگی های پروژه ایجاد شده



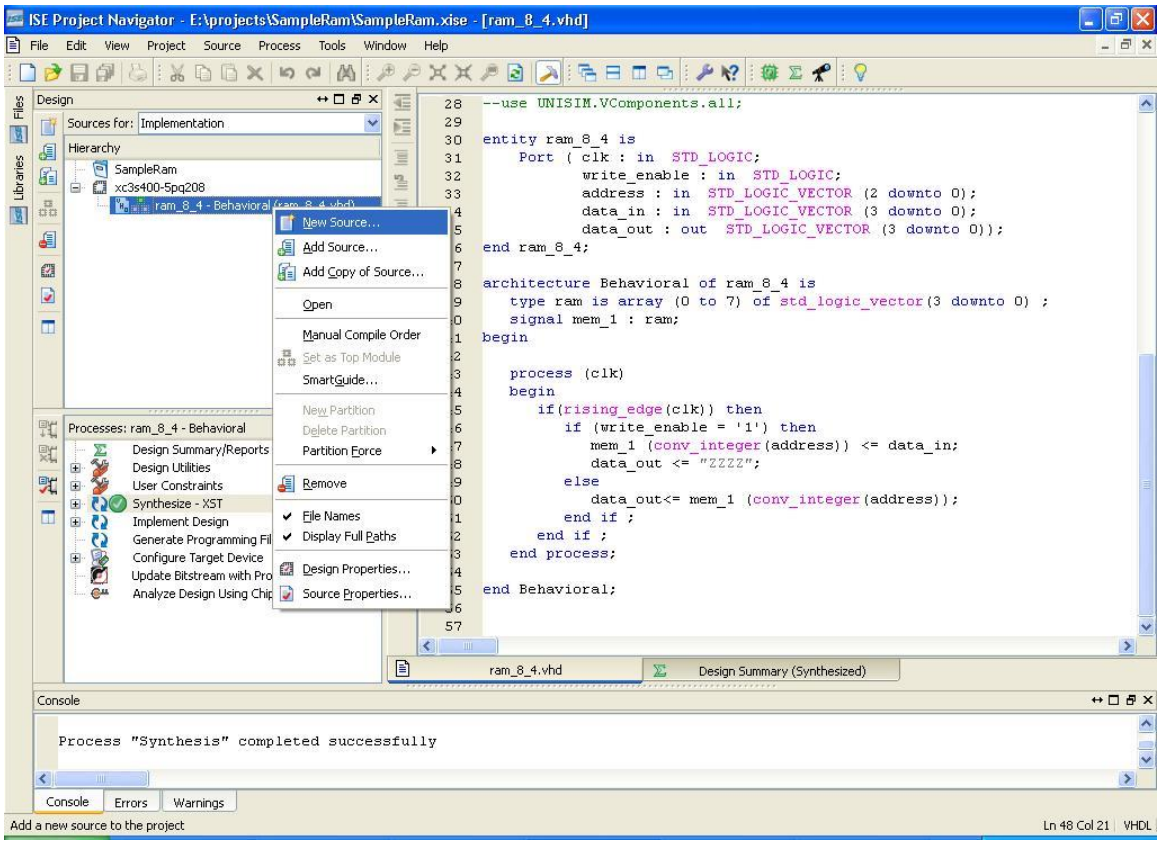
شکل ۱۱ نوشتن کد طراحی



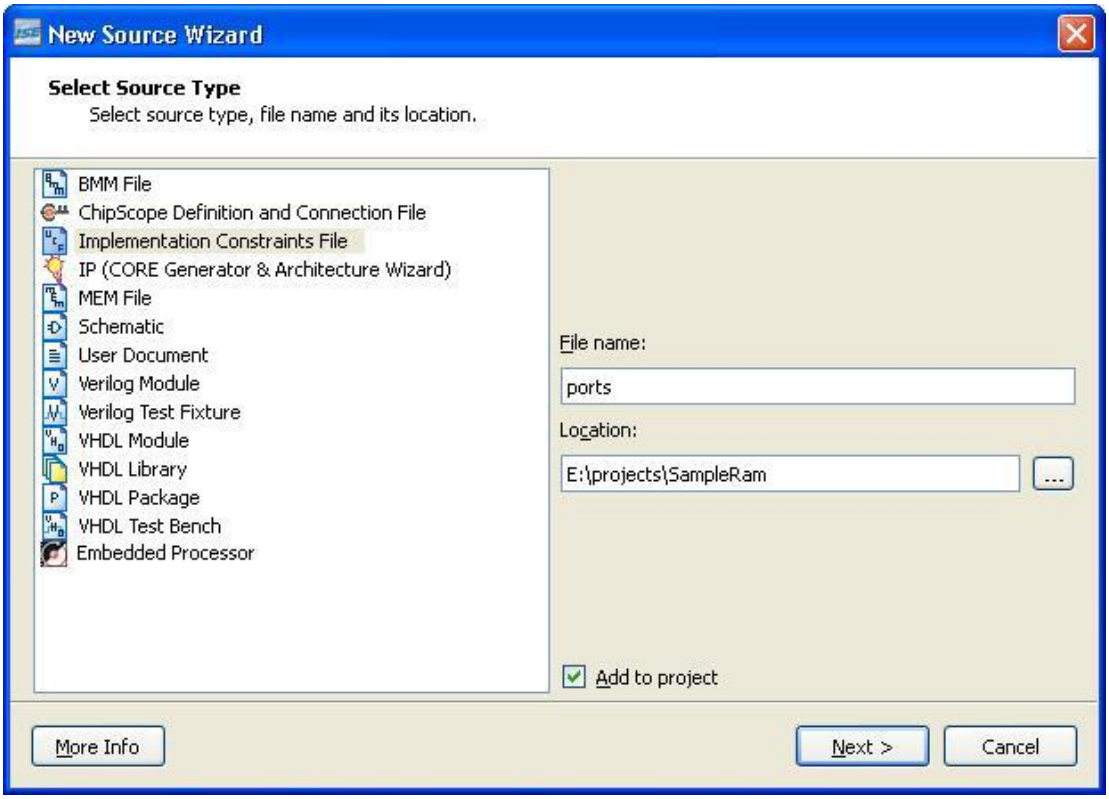
شکل ۱۲ منوی سنتز



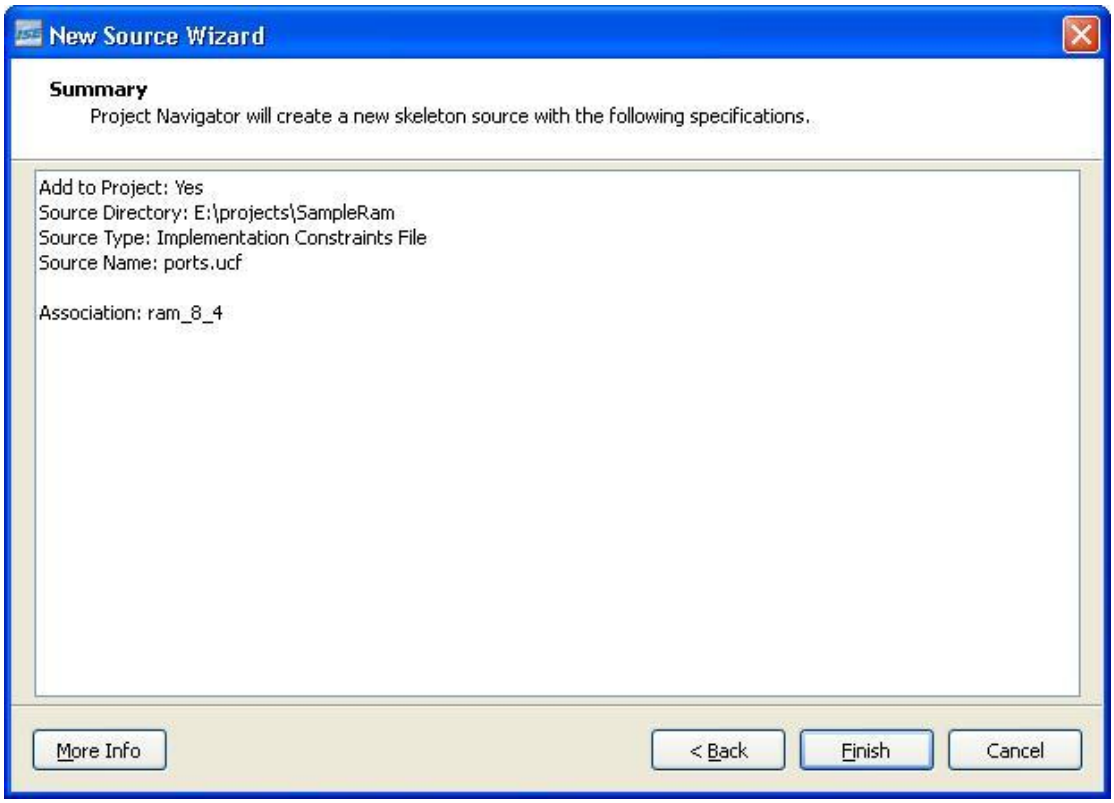
شکل ۱۳ نتیجه سنتز



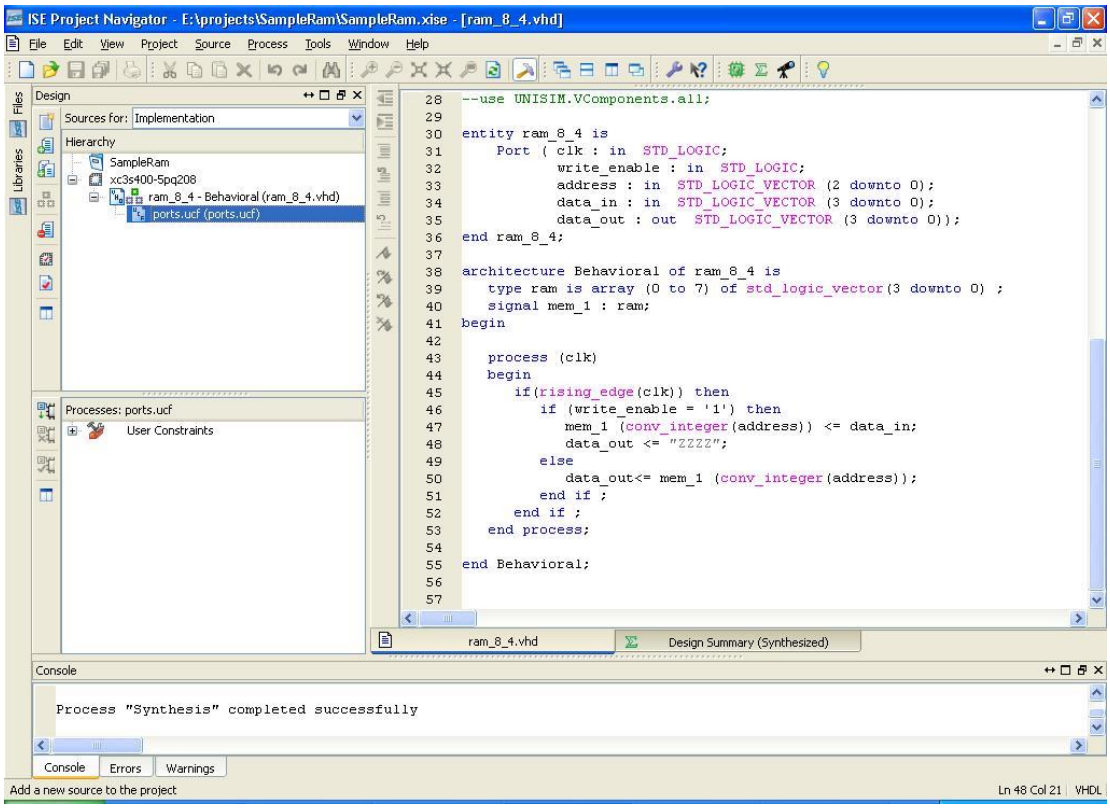
شکل ۱۴ منوی ایجاد فایل محدودیت (تعیین پورت های FPGA)



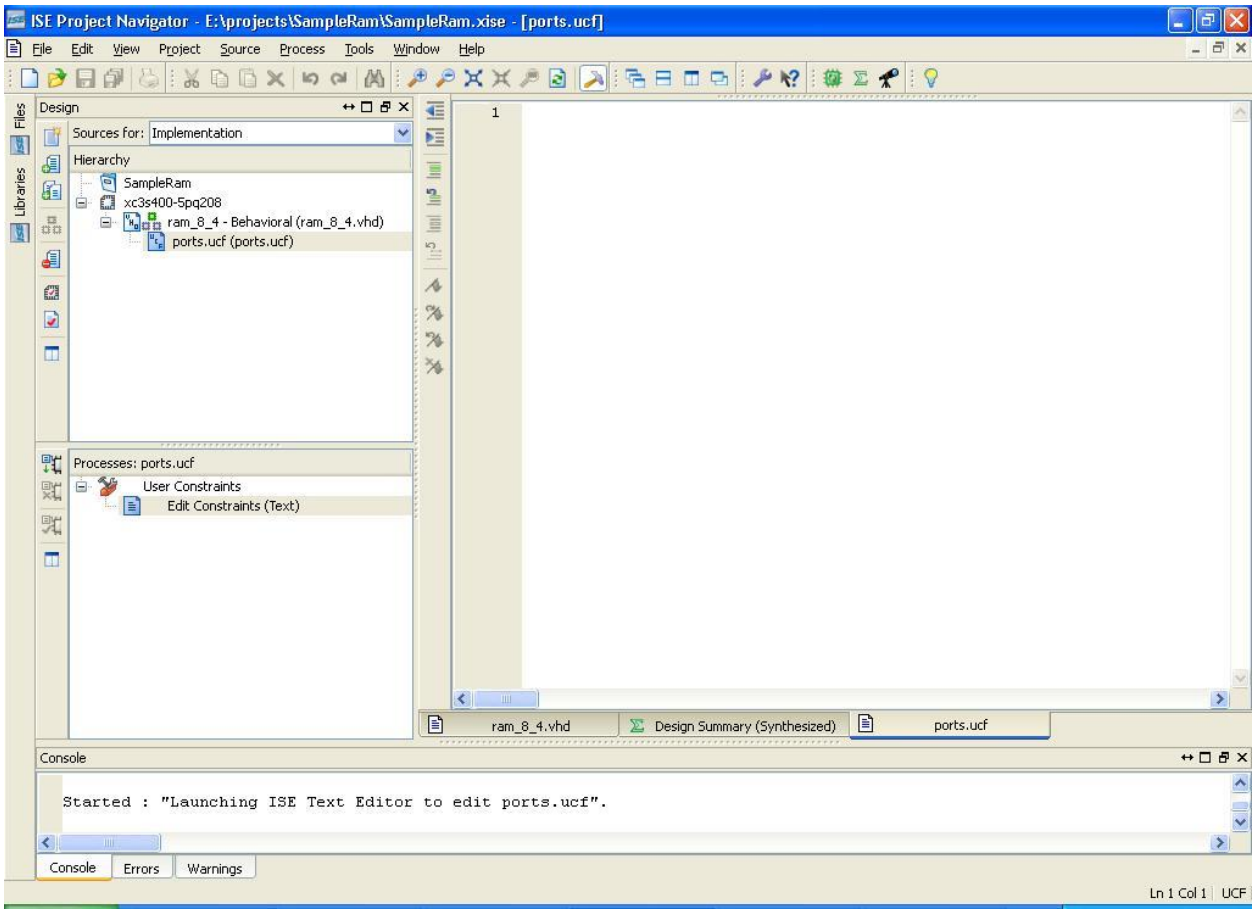
شکل ۱۵ تعیین نام فایل محدودیت



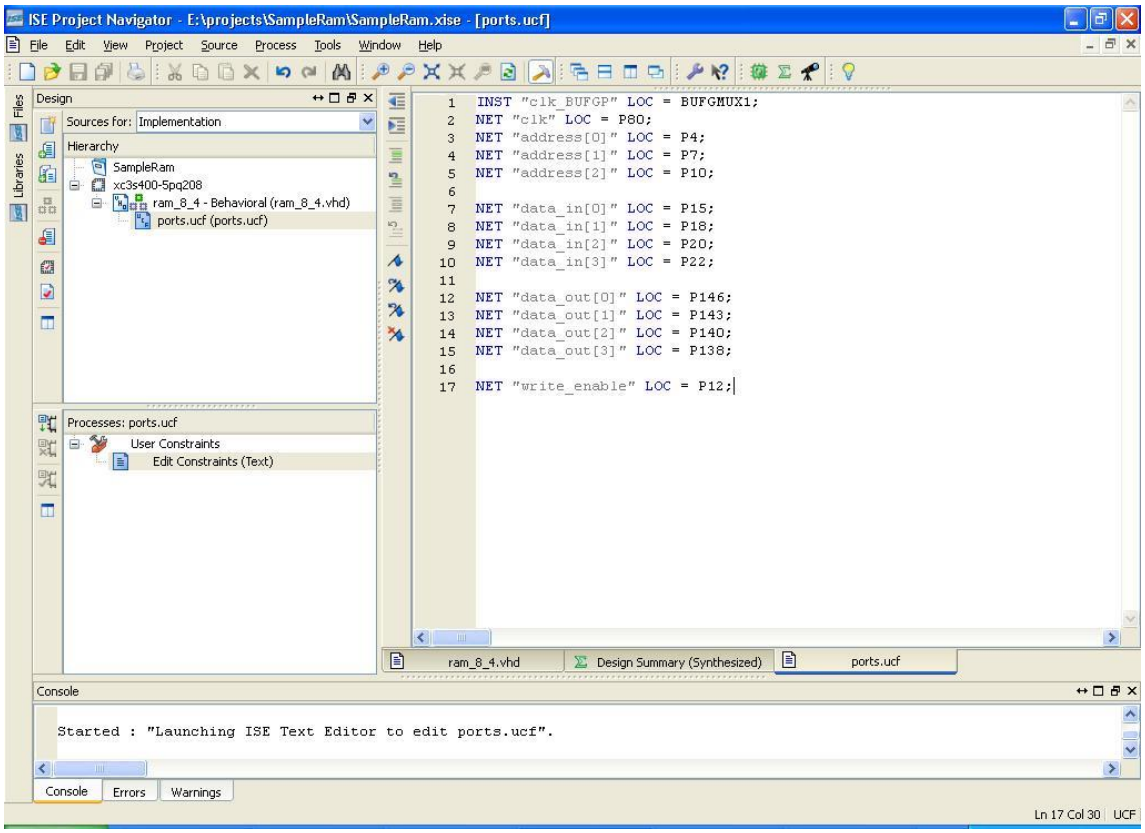
شکل ۱۶ ویژگی های فایل محدودیت ایجاد شده



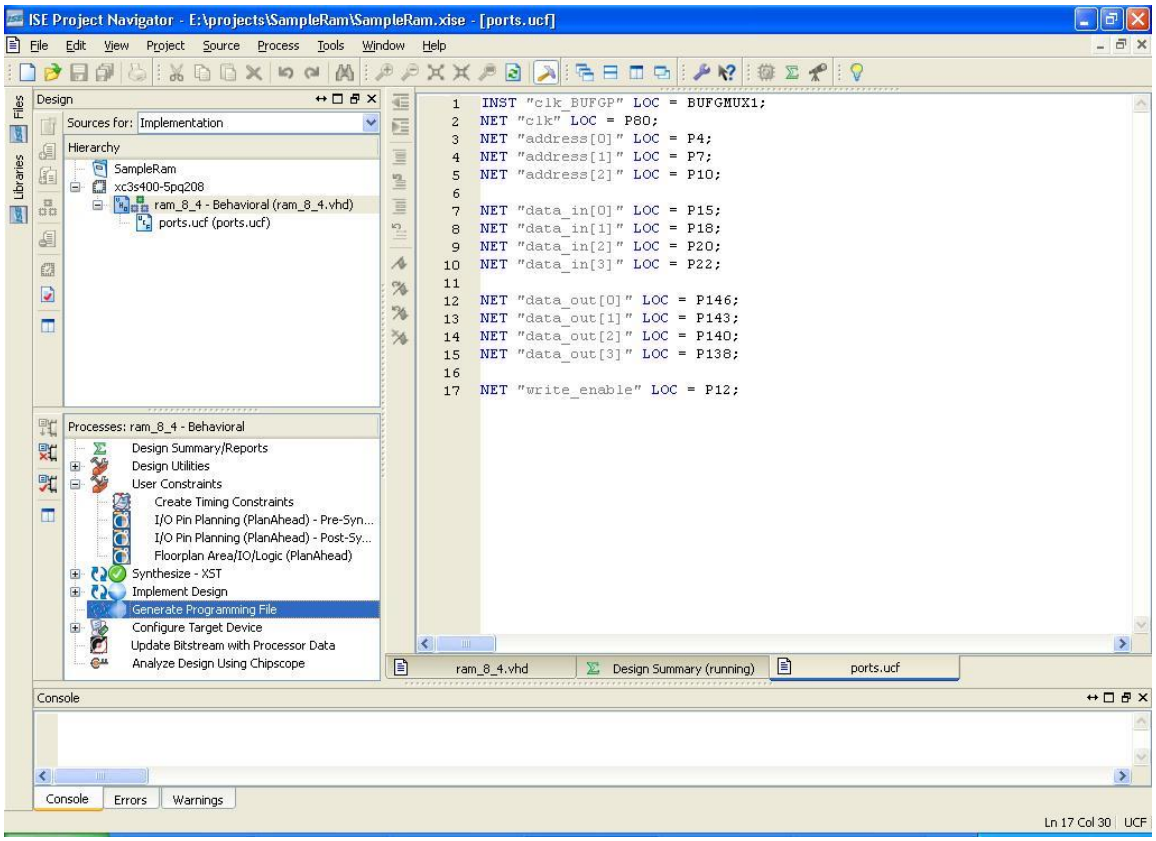
شکل ۱۷ فایل محدودیت در منوی Hirearchy



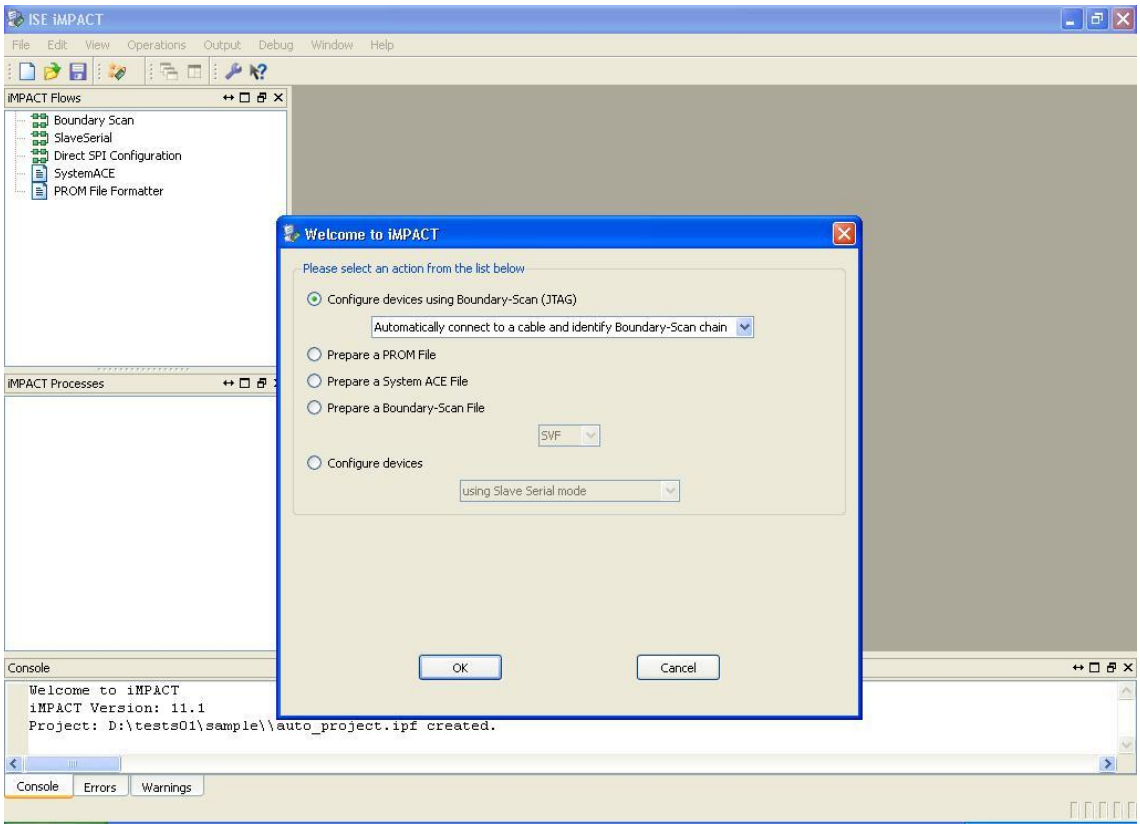
شکل ۱۸ ویرایش فایل محدودیت به صورت متنی



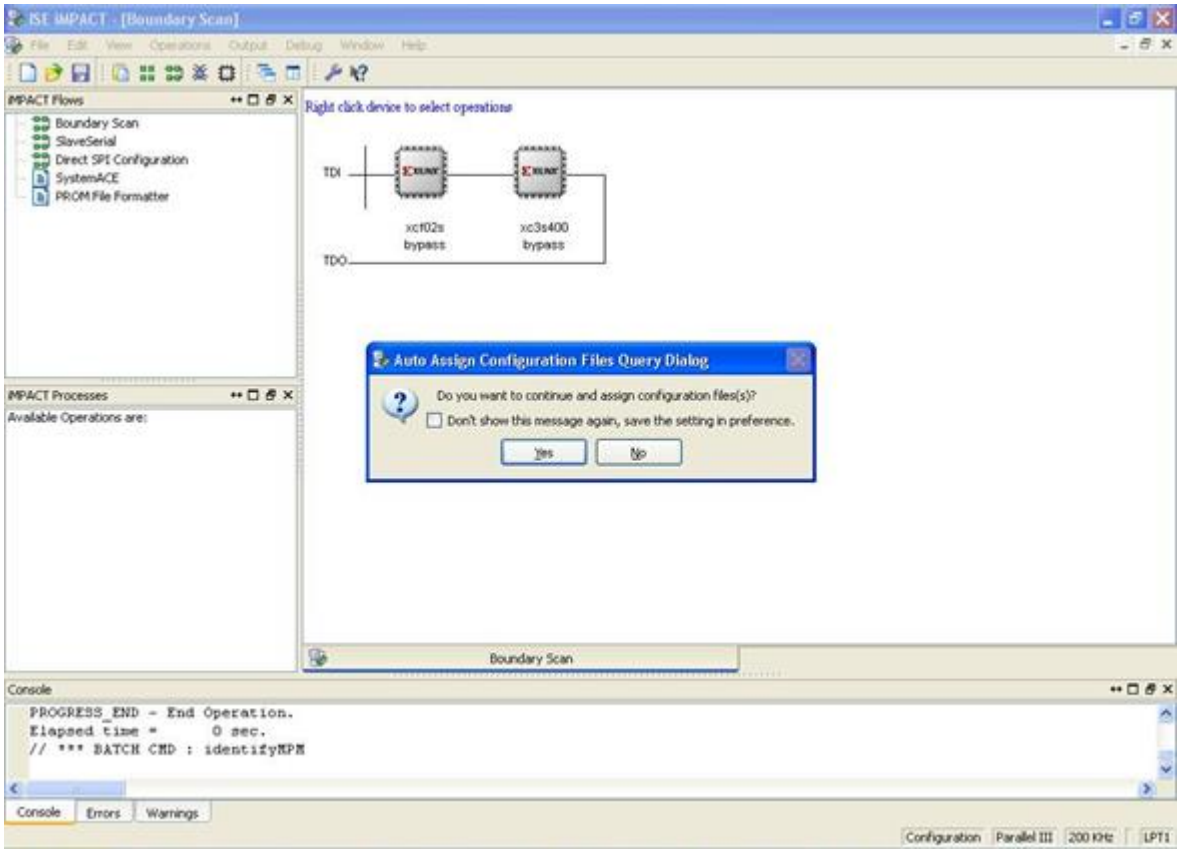
شکل ۱۹ تعیین port های طرح بر روی FPGA



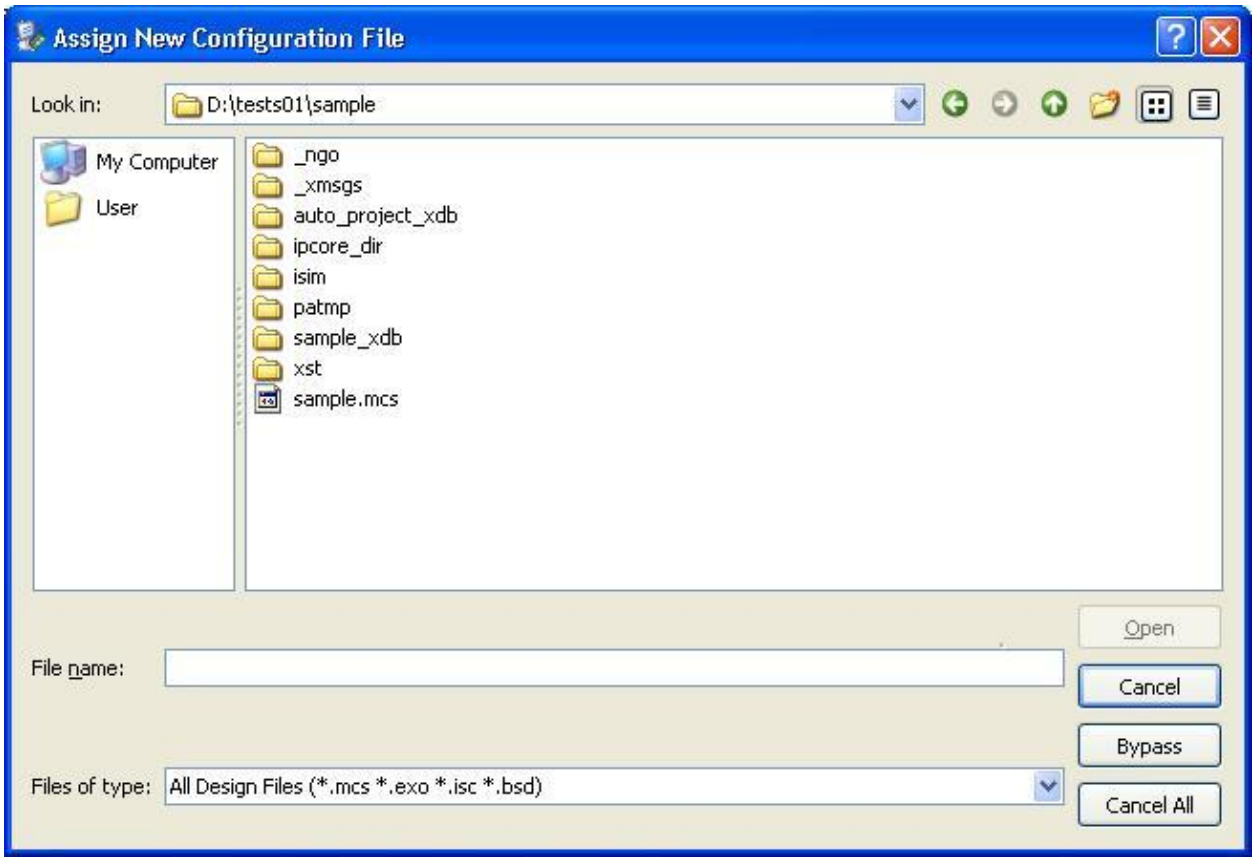
شکل ۲۰ ایجاد فایل برنامه ریزی



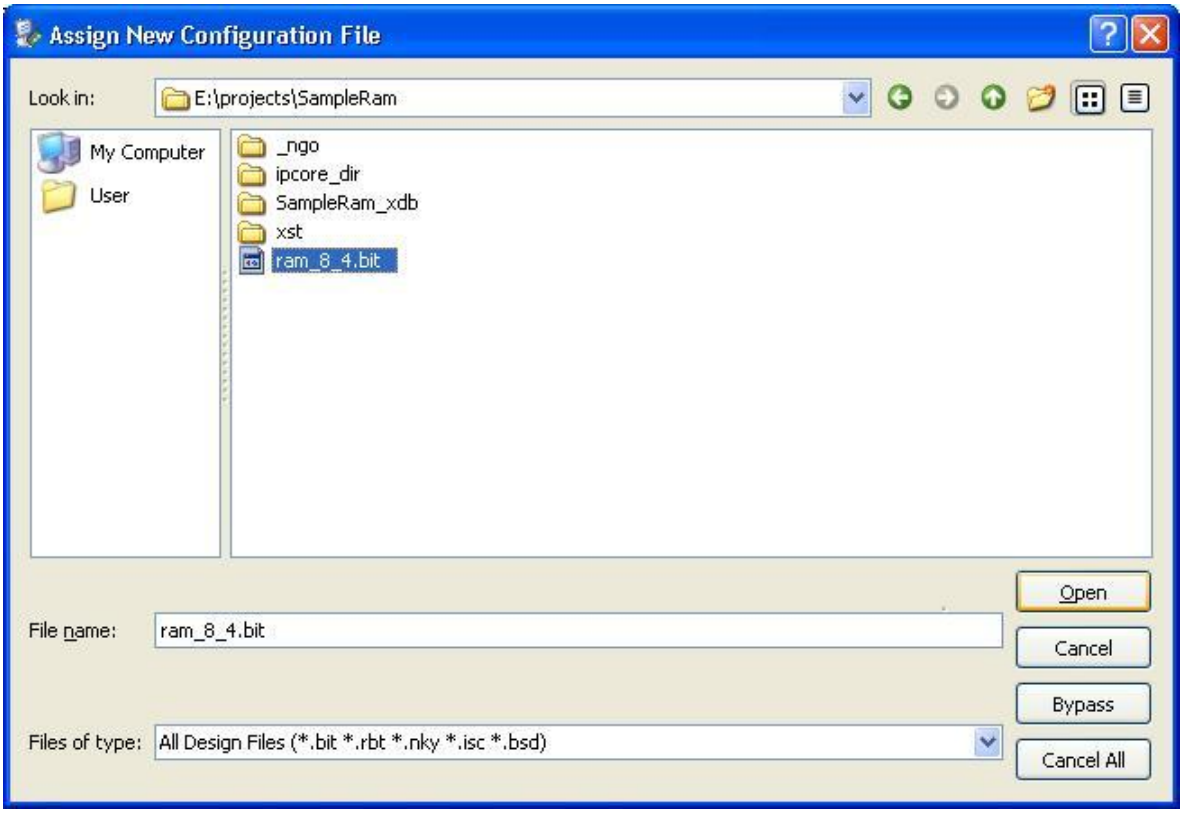
شکل ۲۱ اجرای نرم افزار IMPACT برای برنامه ریزی



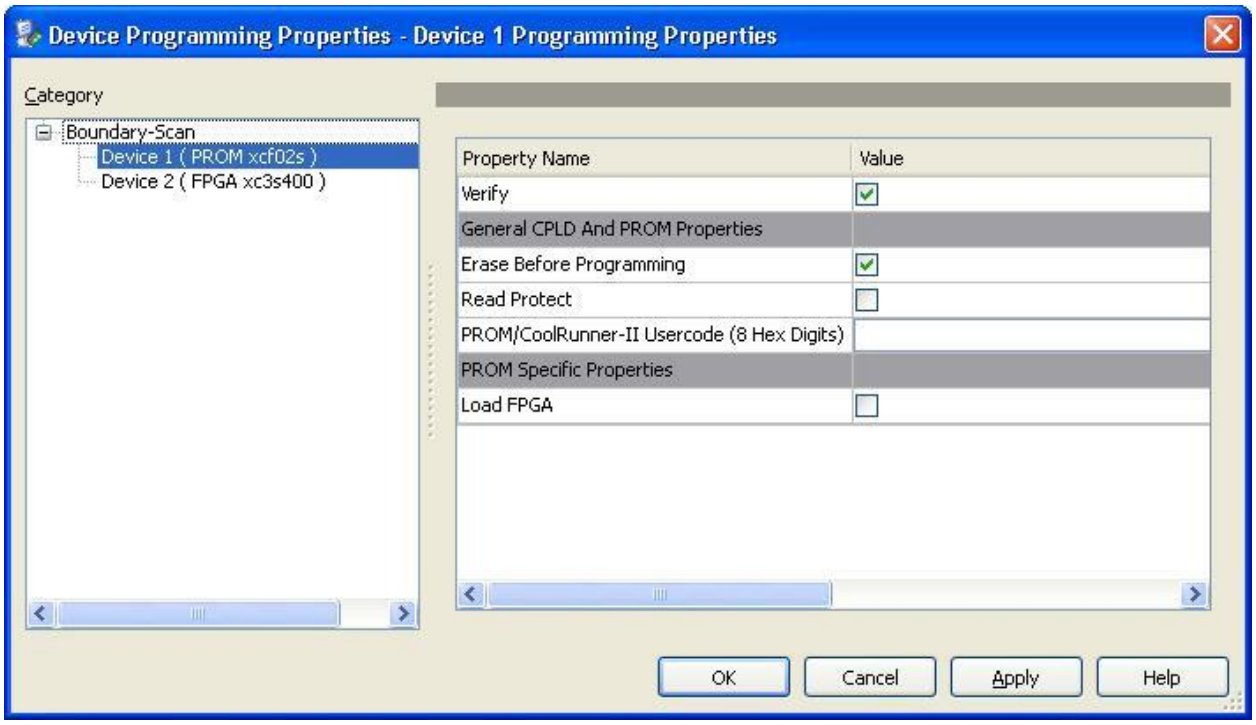
شکل ۲۲ ابزار های شناخته شده



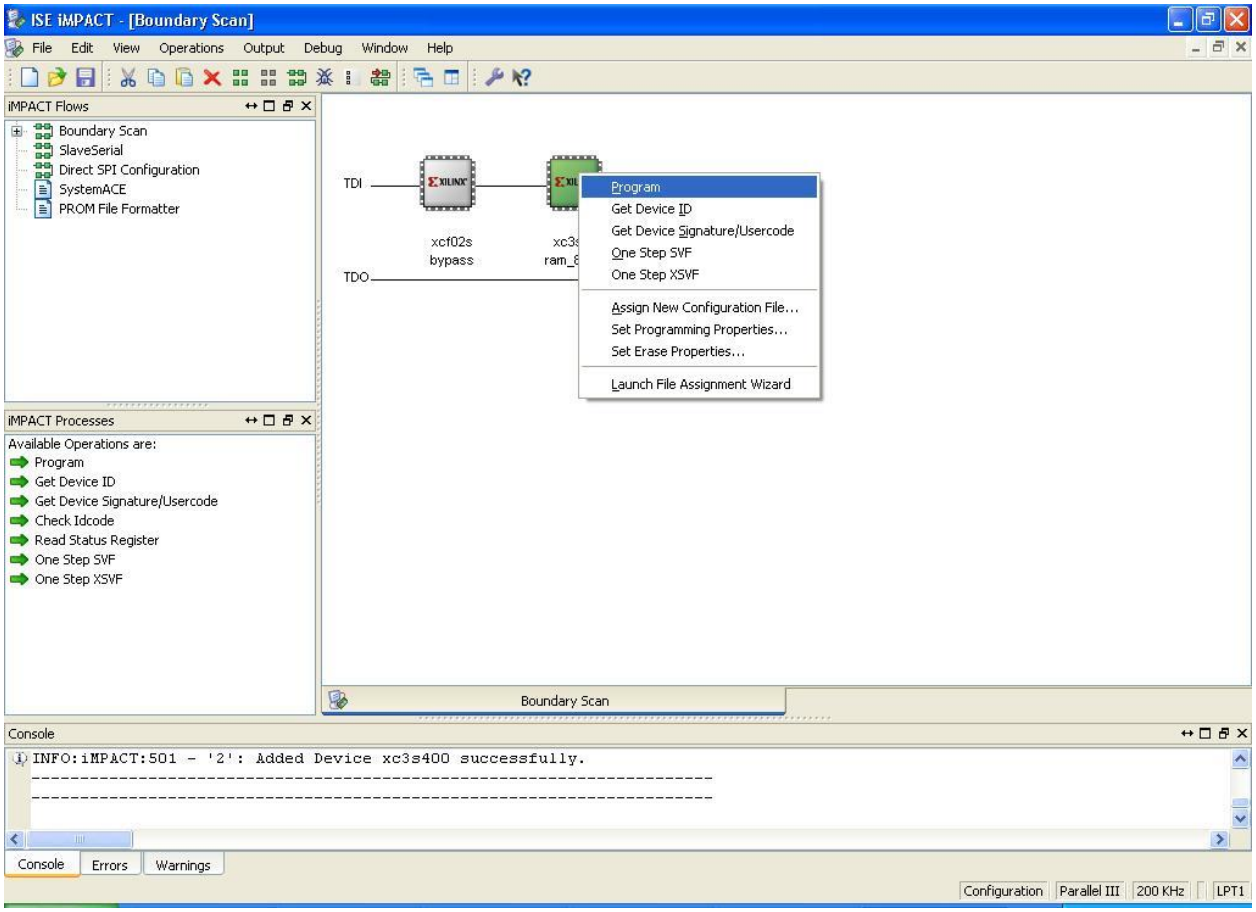
شکل ۲۳ تعیین فایل برنامه ریزی RAM موجود در روی بورد. (فعلا این مرحله را bypass می کنیم.)



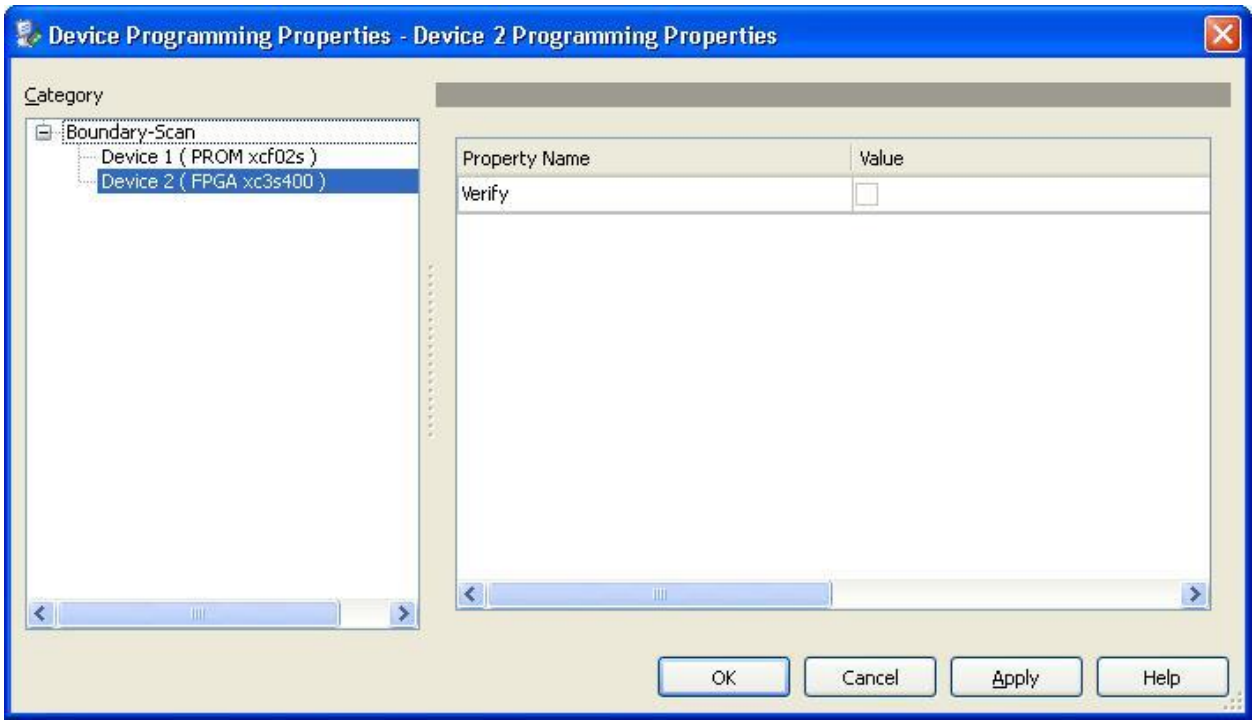
شکل ۲۴ انتخاب فایل برنامه ریزی تولید شده در مرحله قبل



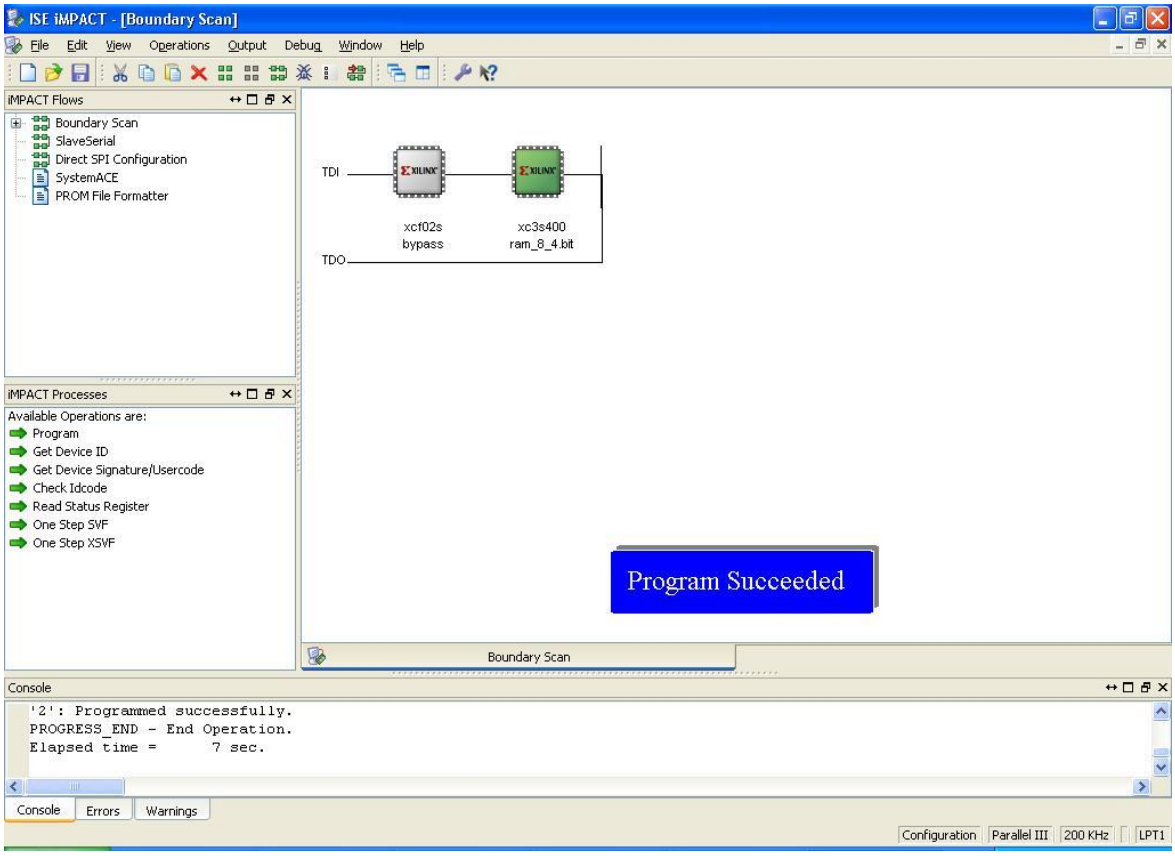
شکل ۲۵ نمایش ویژگی های برنامه ریزی ابزار



شکل ۲۶ برنامه ریزی FPGA

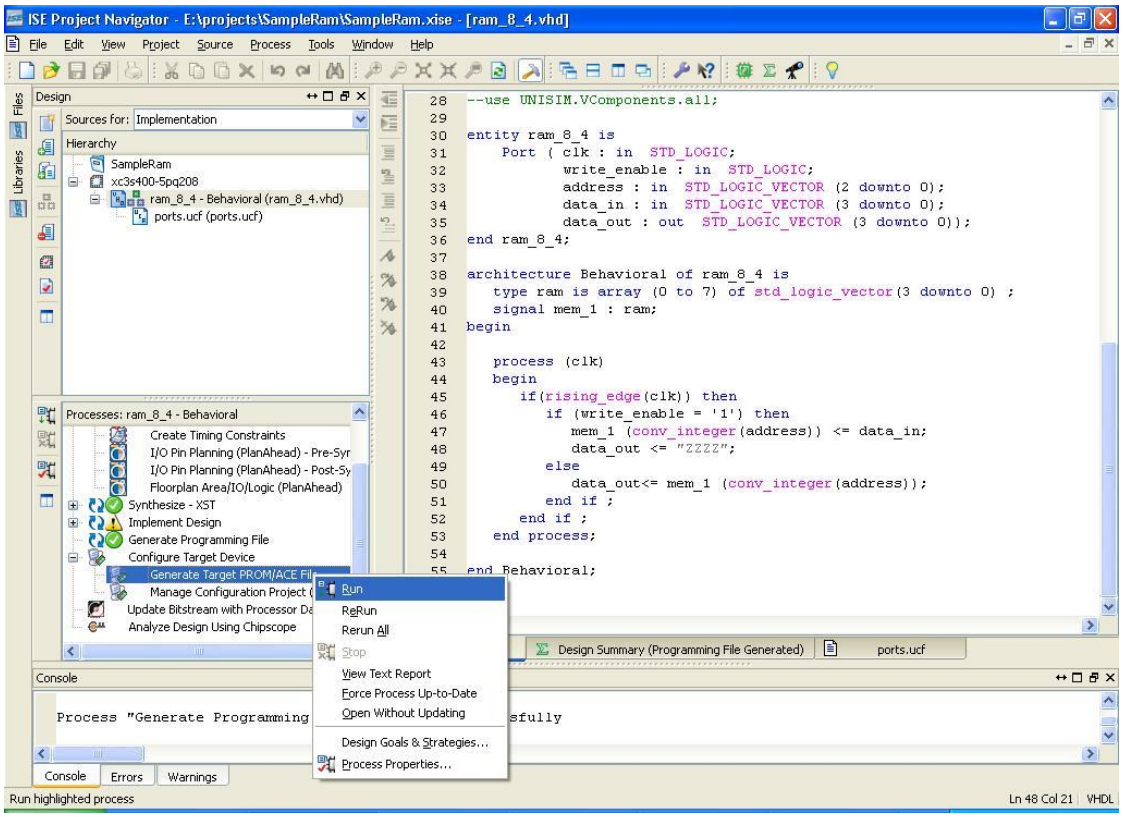


شکل ۲۷ تائید نهایی

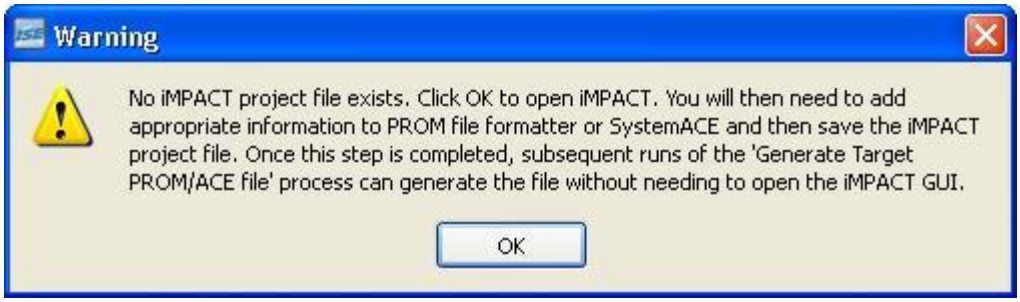


شکل ۲۸ وضعیت برنامه ریزی

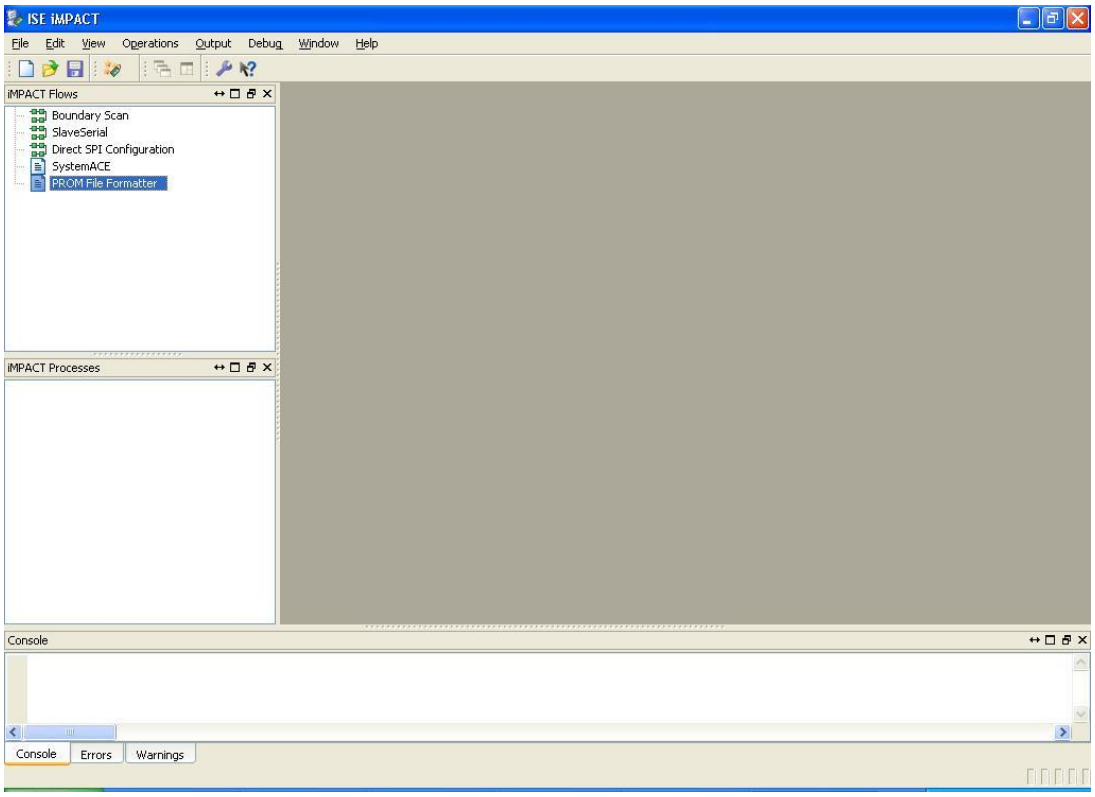
فاز دوم: برنامه ریزی RAM روی بورد



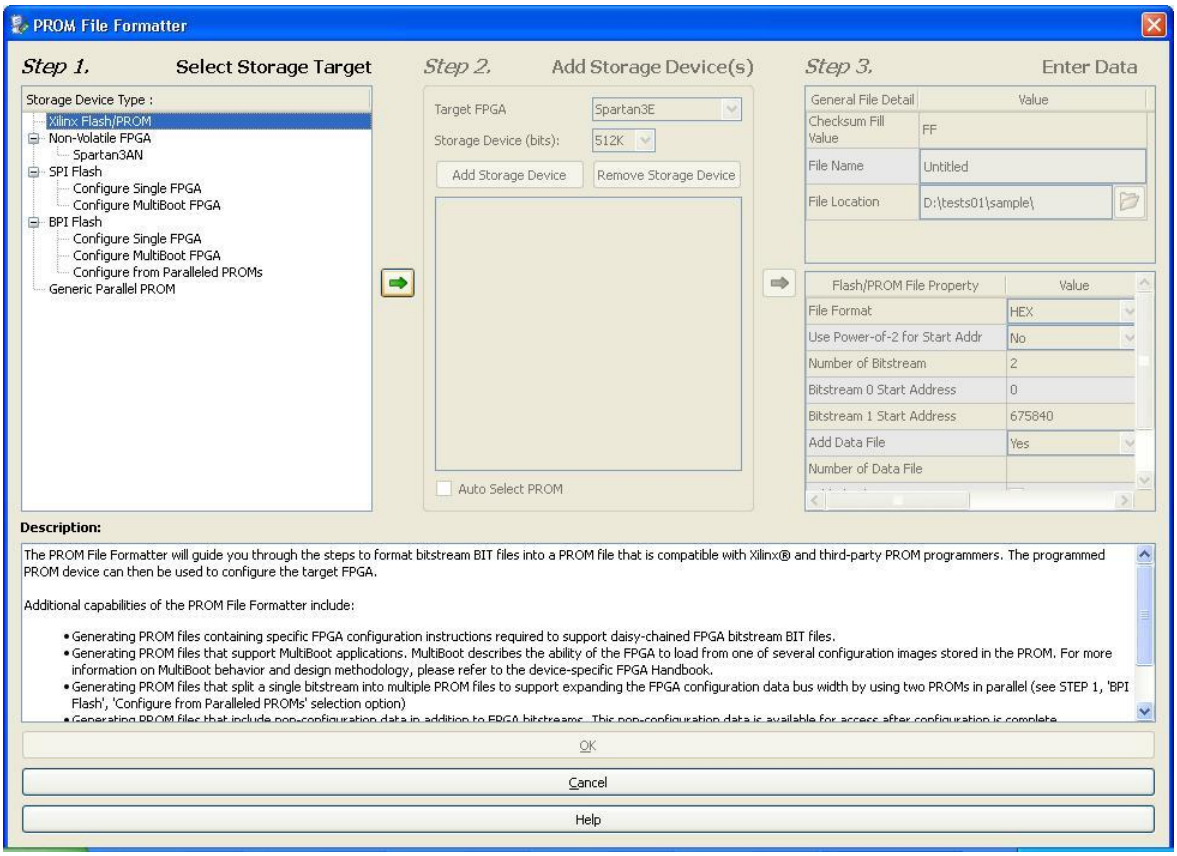
شکل ۲۹ منوی ایجاد فایل برنامه ریزی RAM



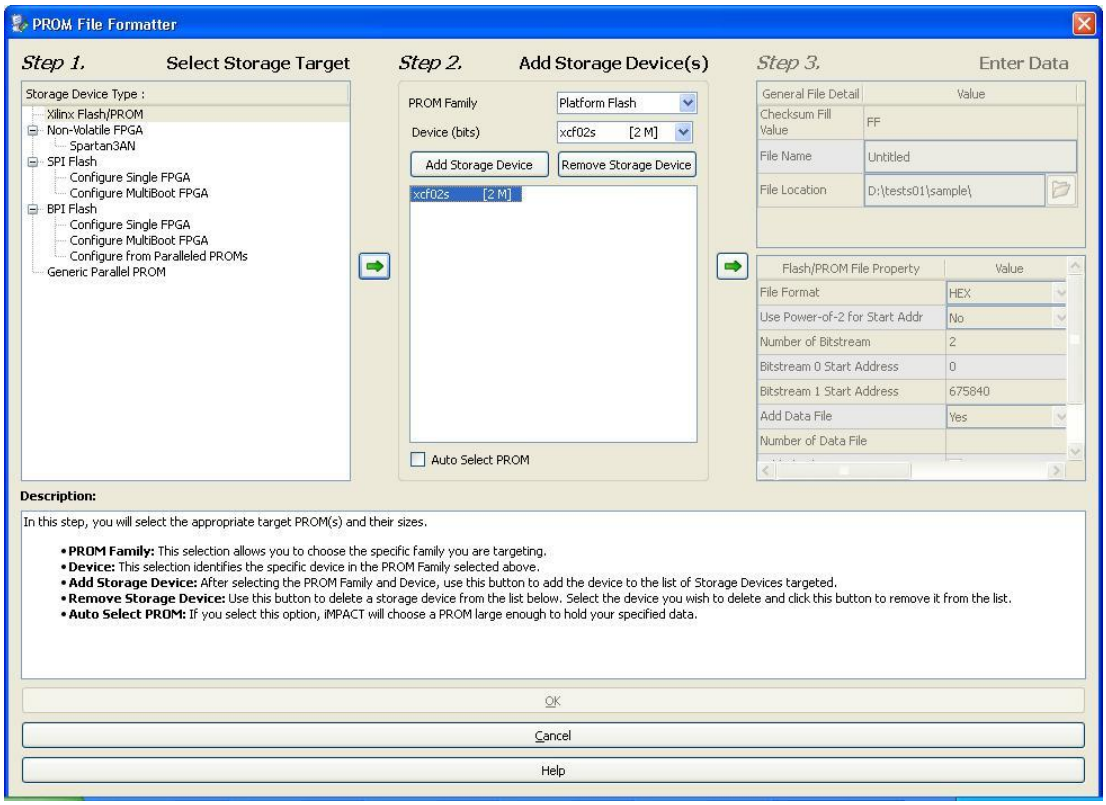
شکل ۳۰ هشدار مبنی بر ایجاد پروژه iMPACT جدید



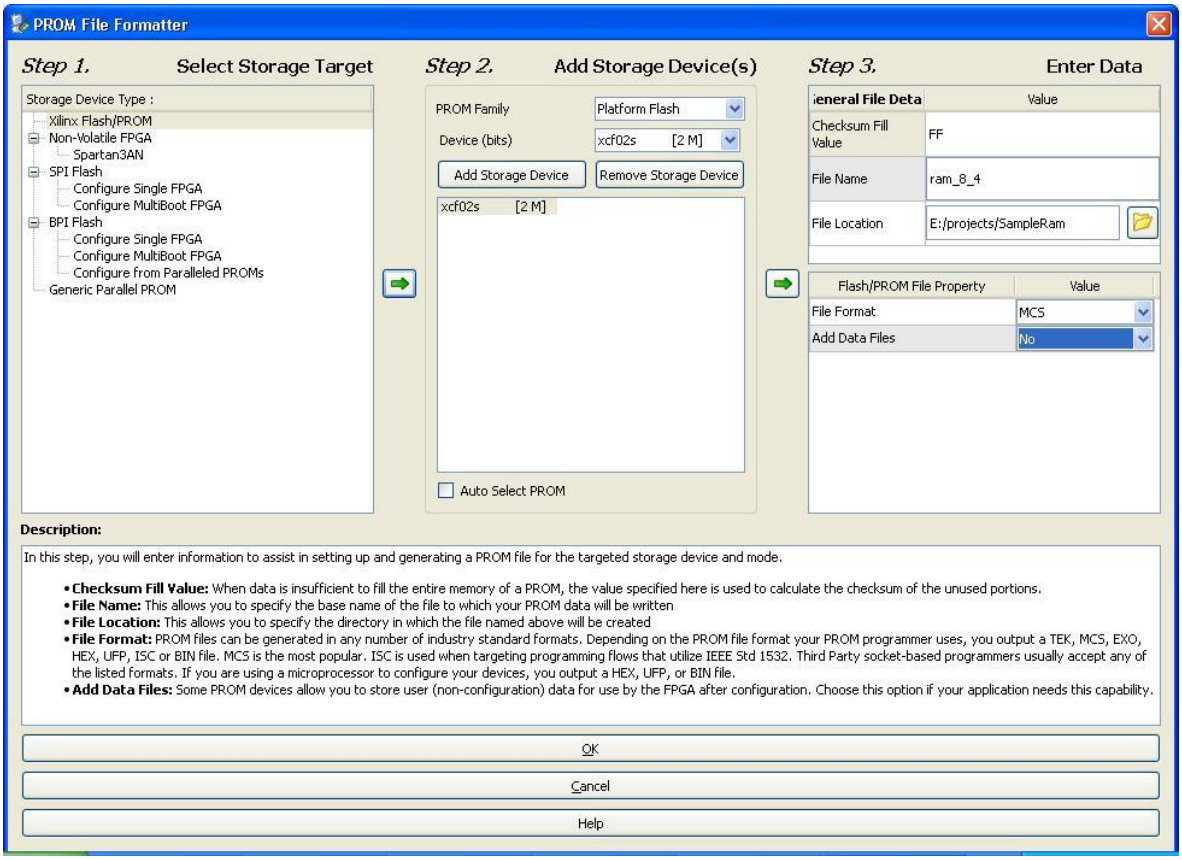
شکل ۳۱ منوی ایجاد فایل برنامه ریزی در پنجره iMPACT



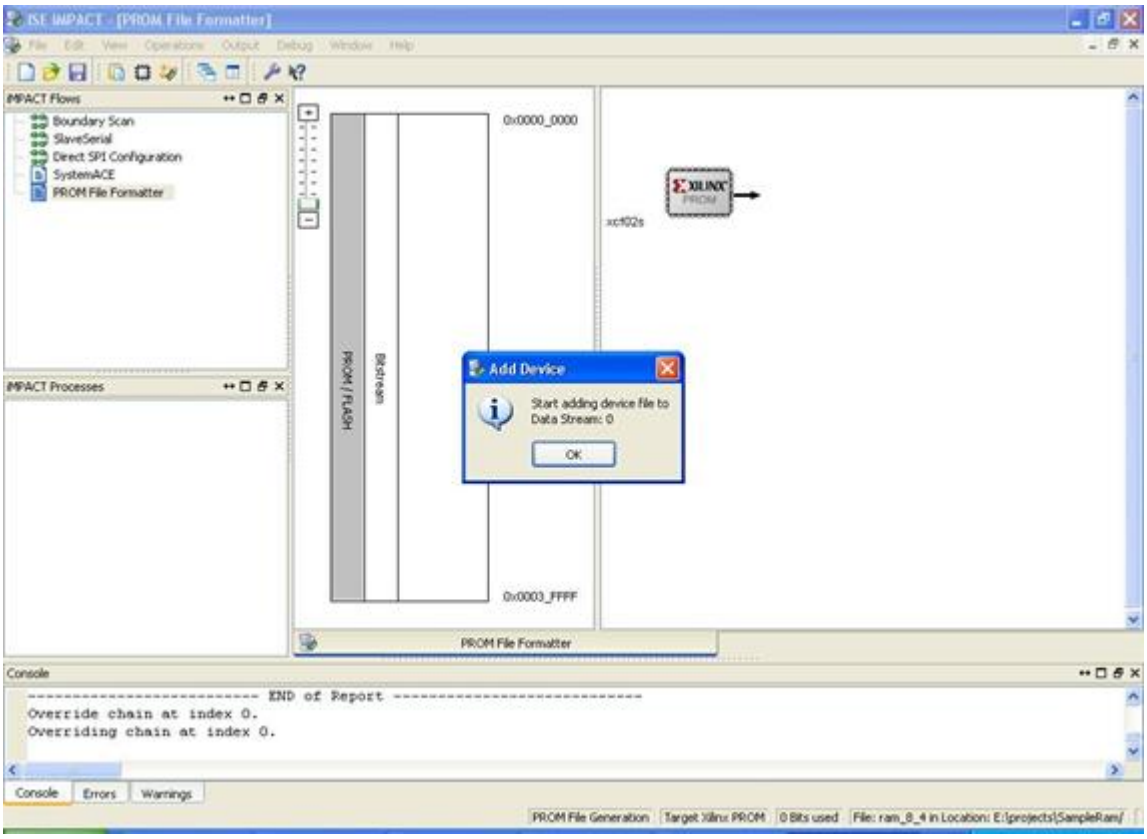
شکل ۳۲ گام اول ایجاد فایل برنامه ریزی



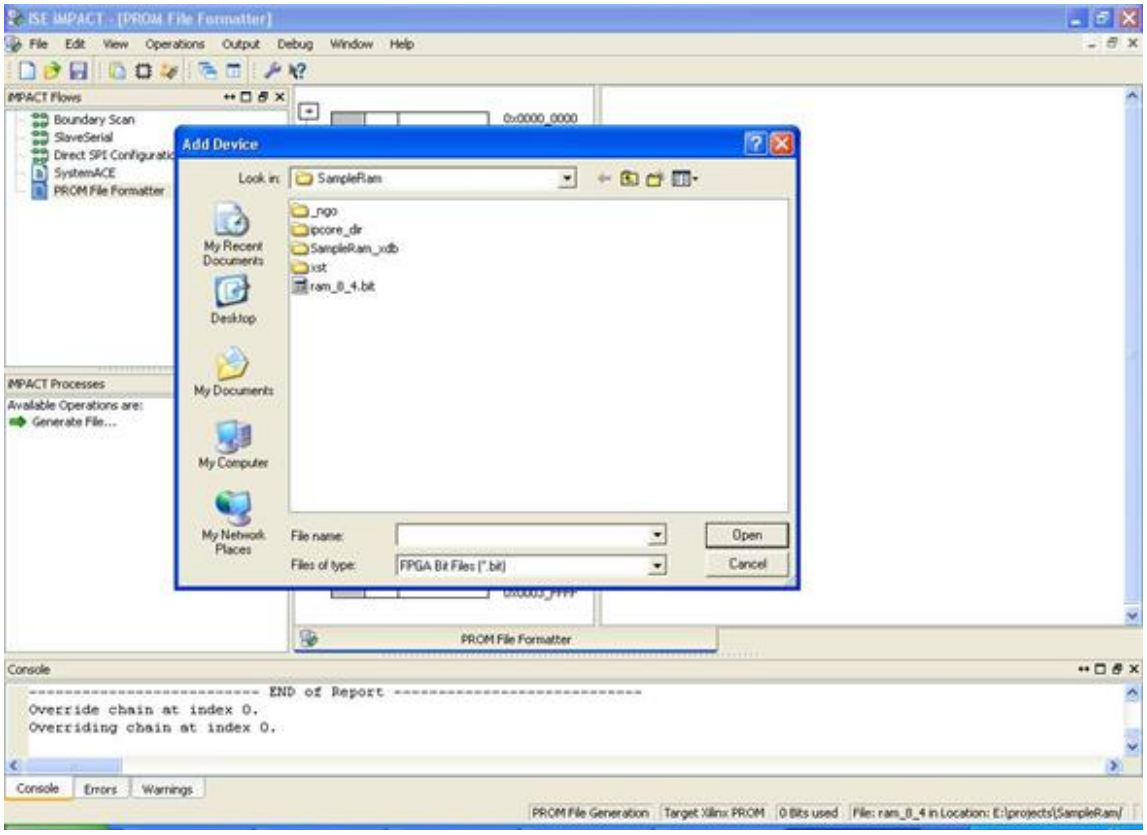
شکل ۳۳ گام دوم ایجاد فایل برنامه‌ریزی



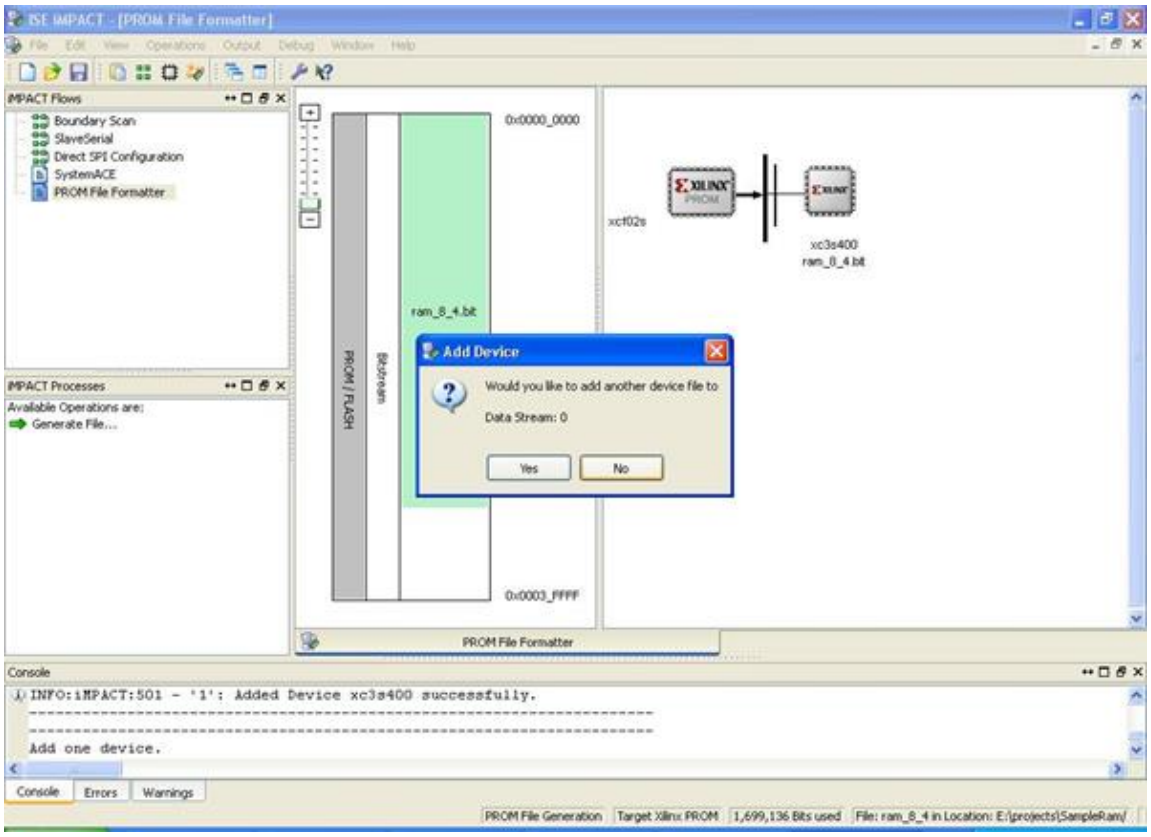
شکل ۳۴ گام سوم ایجاد فایل برنامه‌ریزی



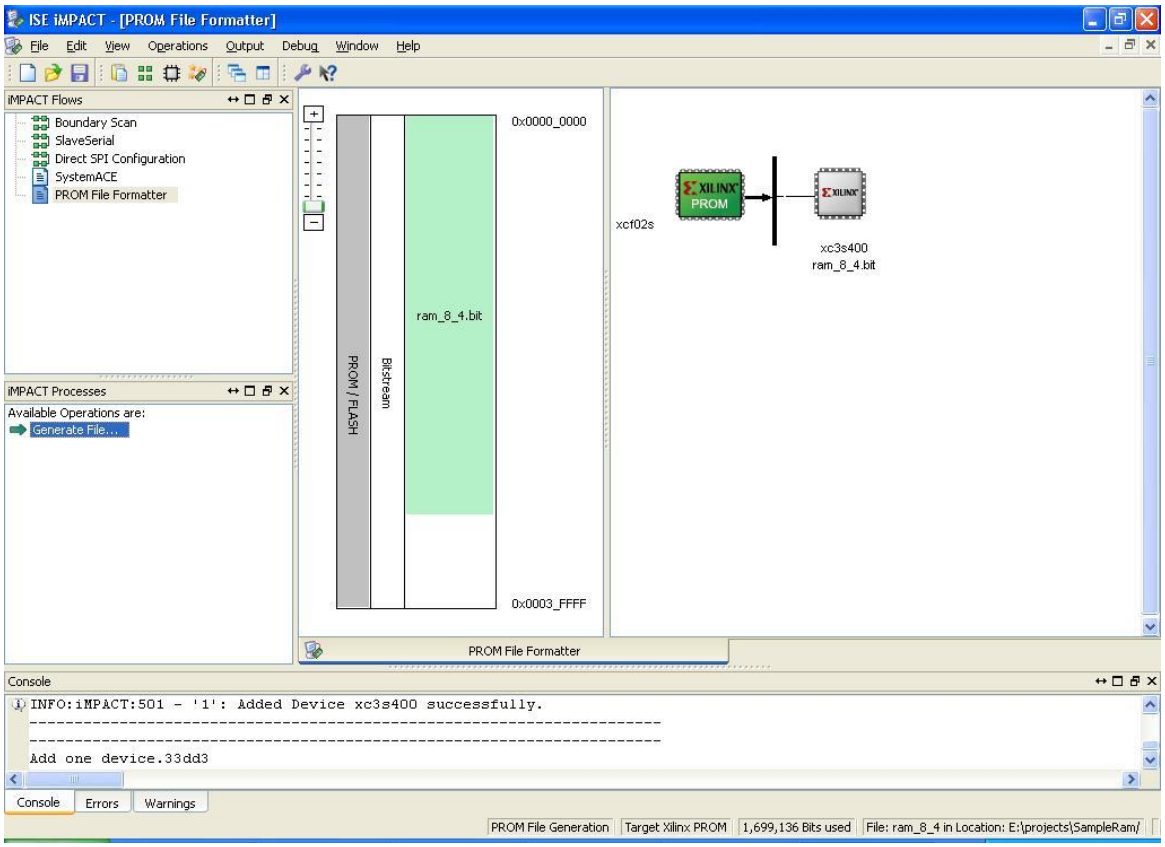
شکل ۳۵ جستجوی RAM های روی بورد توسط iMPACT



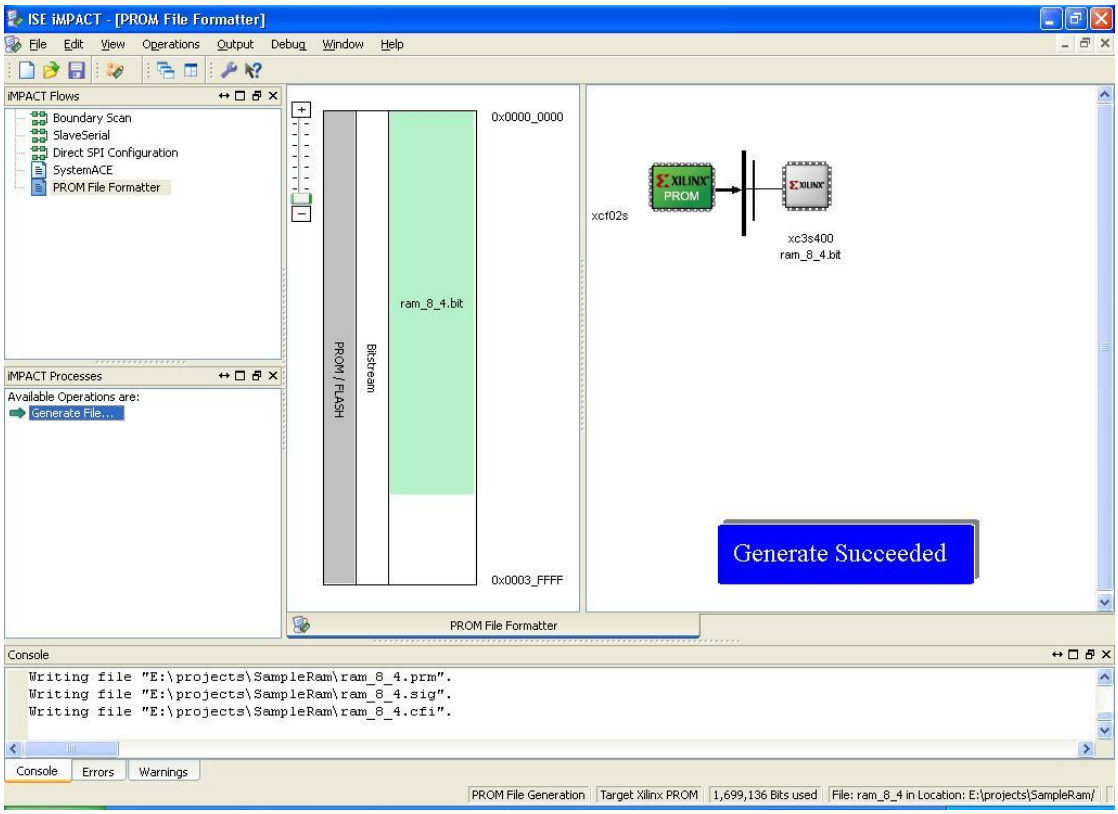
شکل ۳۶ انتخاب فایل برنامه ریزی FPGA برای برنامه ریزی RAM



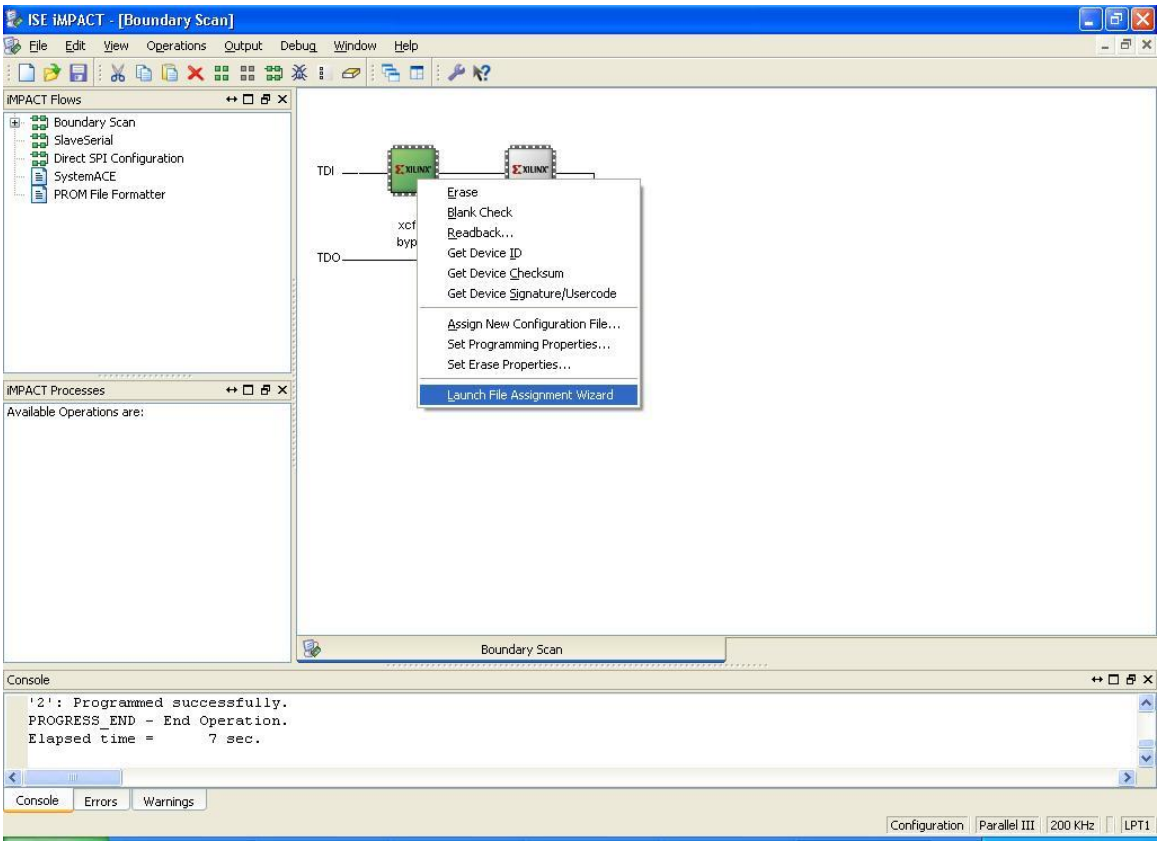
شکل ۳۷ هشدار مبنی بر افزودن فایل جدید



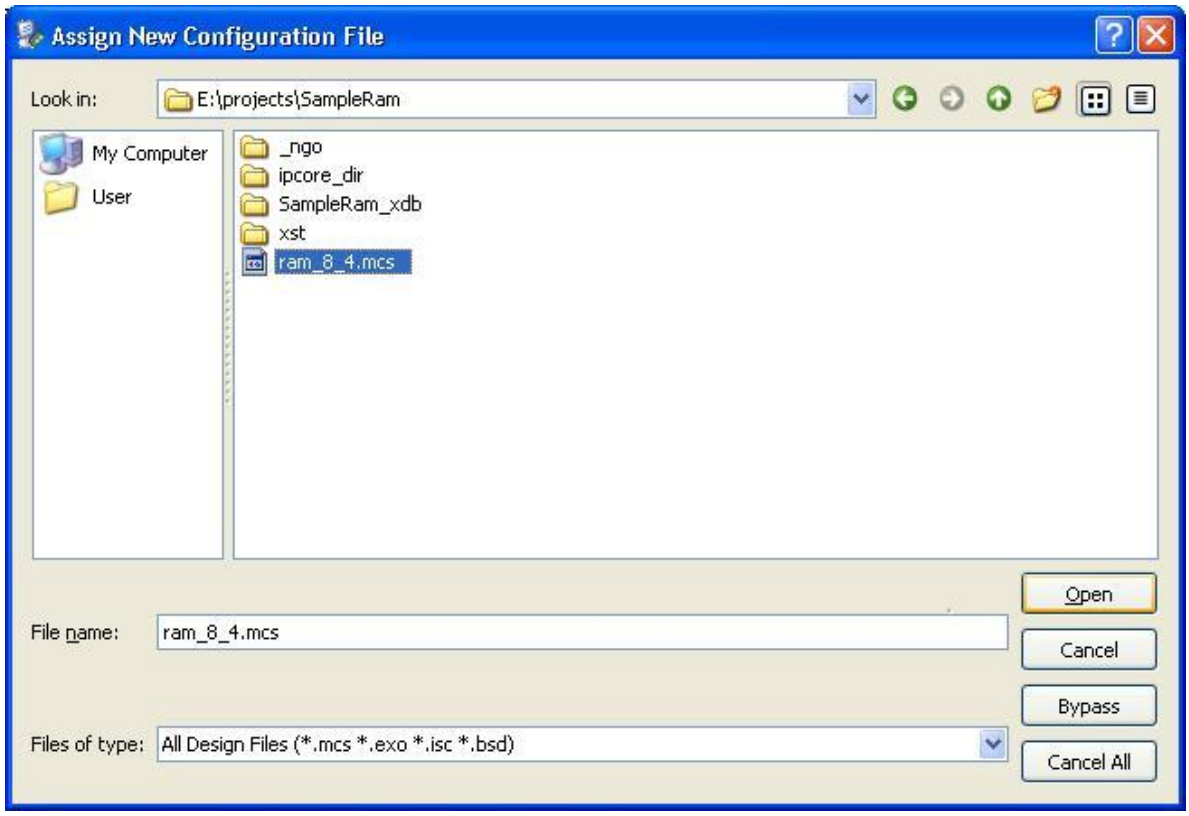
شکل ۳۸ نتیجه انتخاب فایل



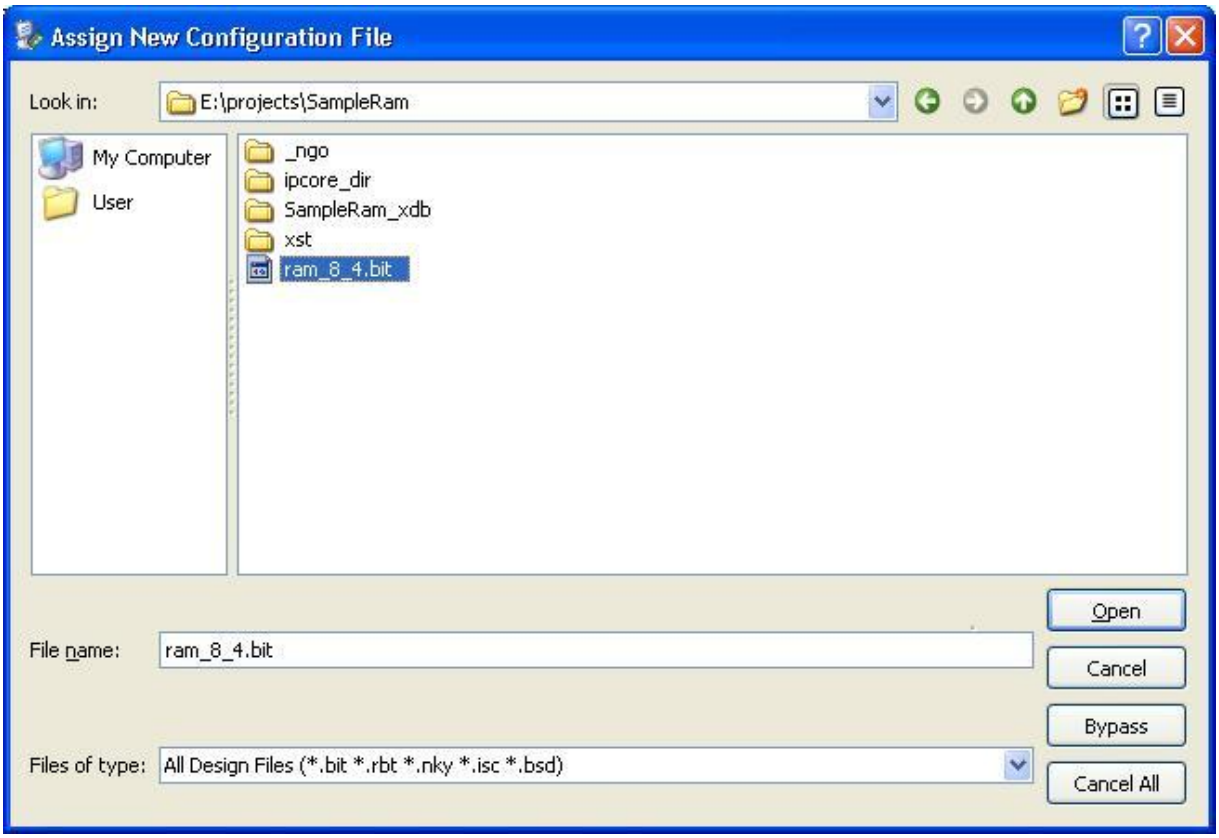
شکل ۳۹ ایجاد موفقیت آمیز فایل برنامه‌ریزی RAM



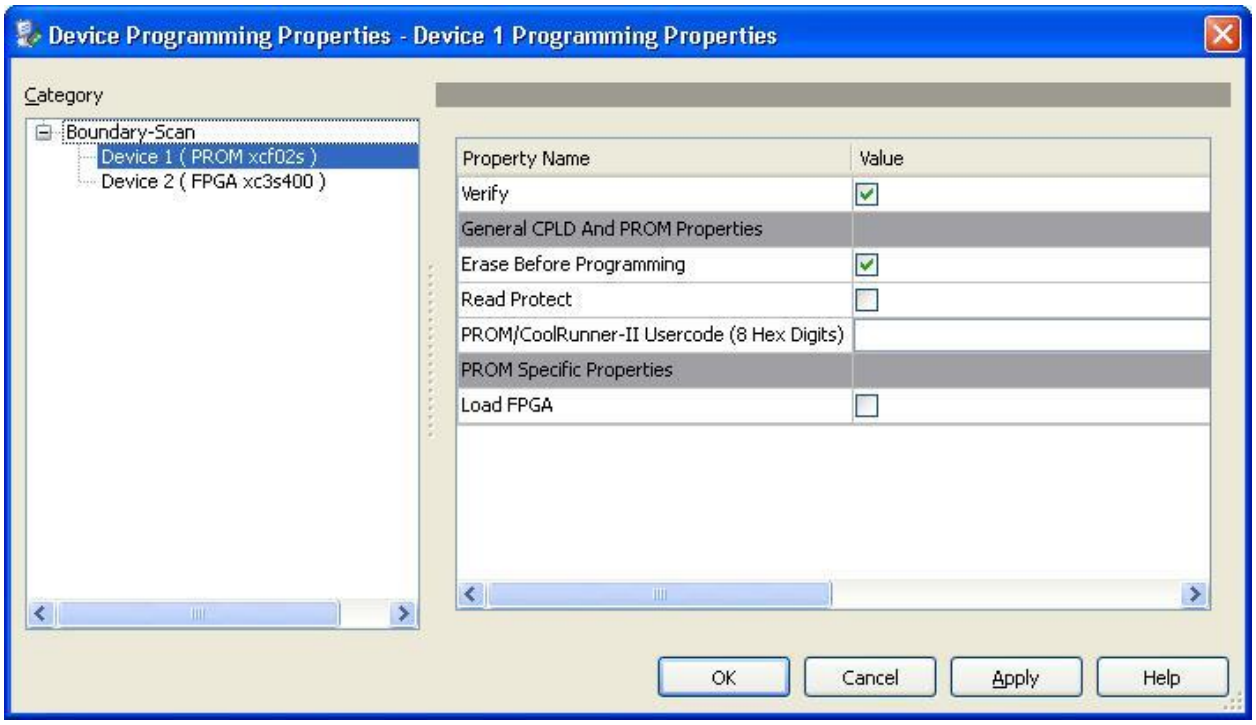
شکل ۴۰ تعیین فایل برنامه‌ریزی ram



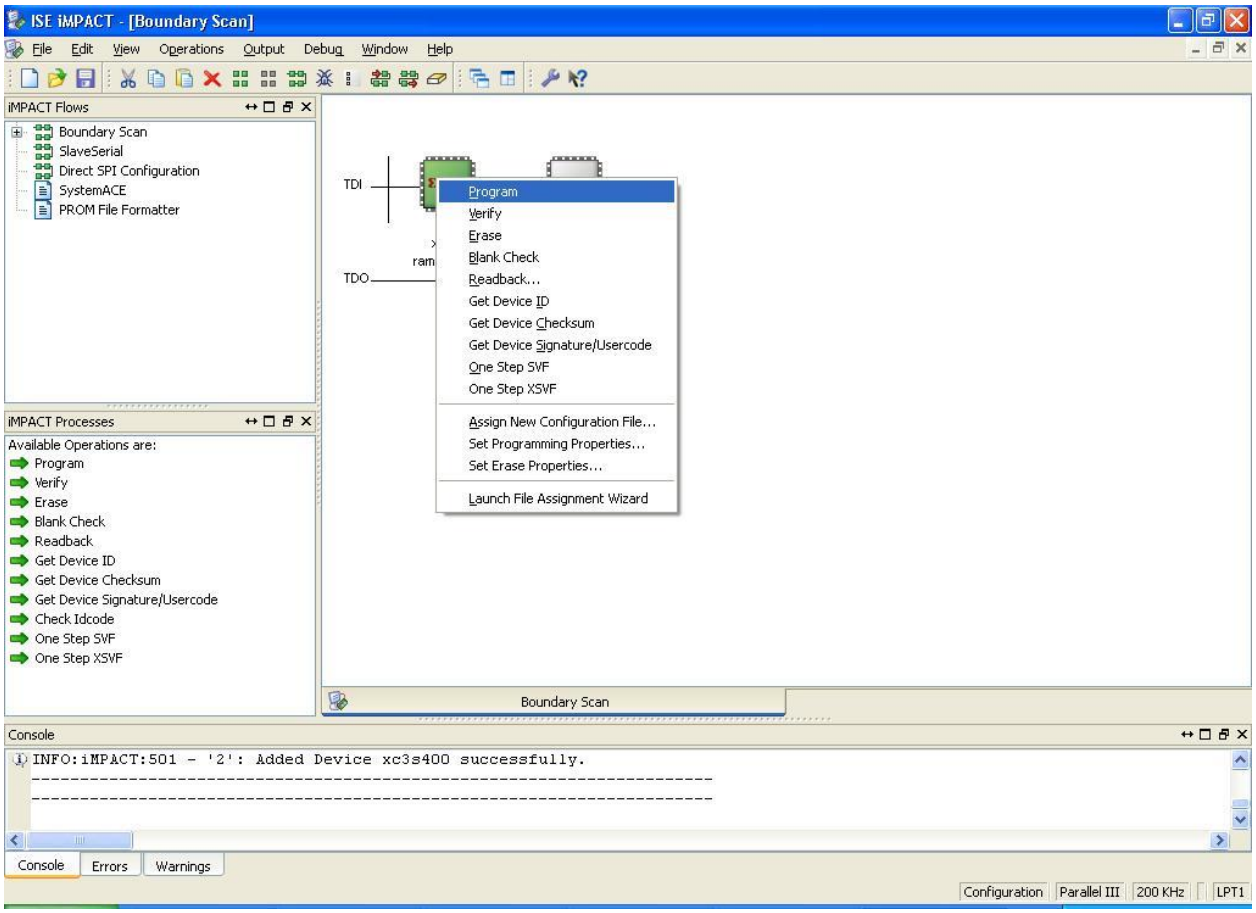
شکل ۴۱ انتخاب فایل برنامه ریزی RAM



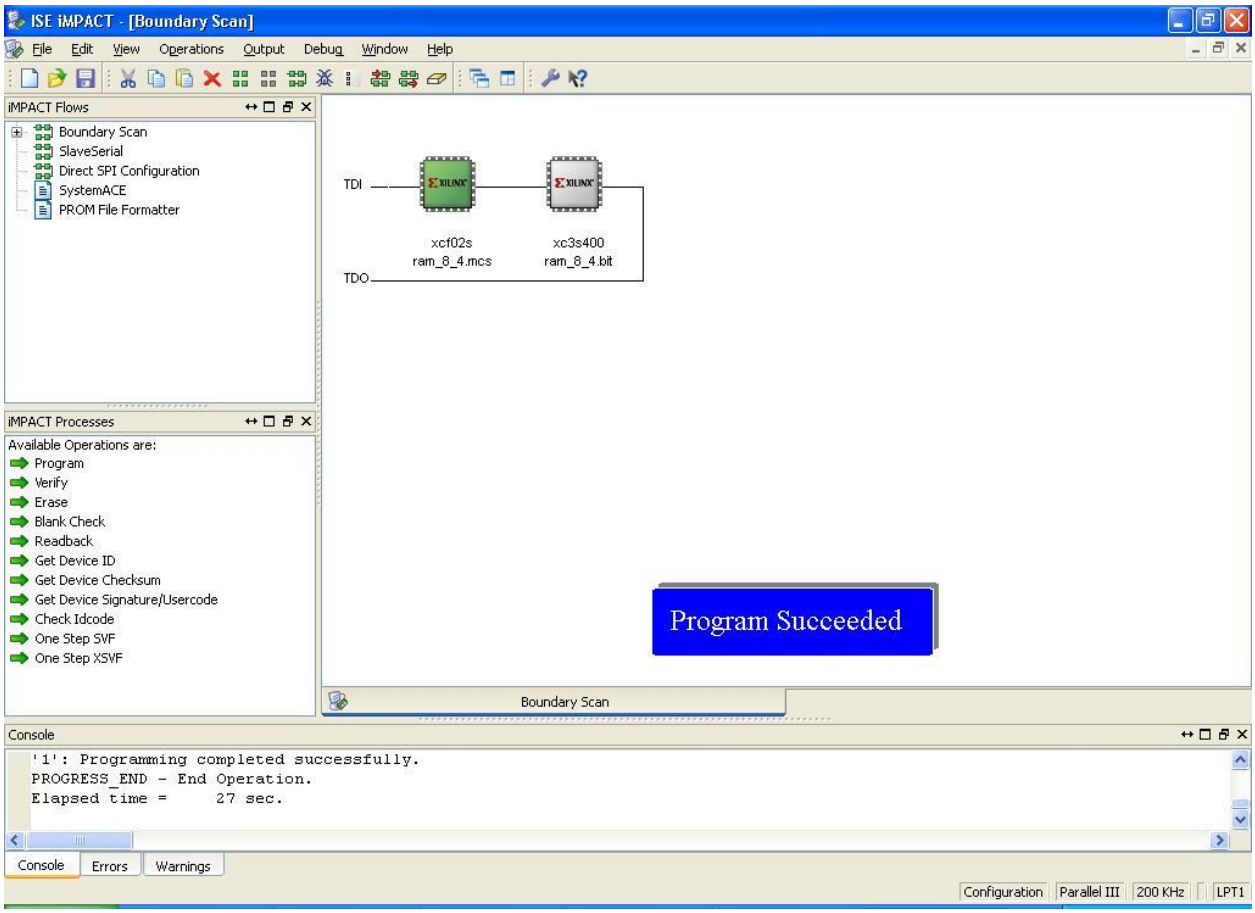
شکل ۴۲ انتخاب فایل برنامه‌ریزی FPGA



شکل ۴۳ ویژگی‌های برنامه‌ریزی دو ابزار



شکل ۴۴ برنامه‌ریزی RAM



شکل ۴۵ نتیجه موفقیت‌آمیز برنامه‌ریزی RAM