گزارش کارِ سومِ آزمایشگاهِ معماری کامپیوتر

تهیه و تنظیم: مبین خیری

شماره دانشجویی: 994421017

استاد راهنما: دکتر حاجی‌زاده

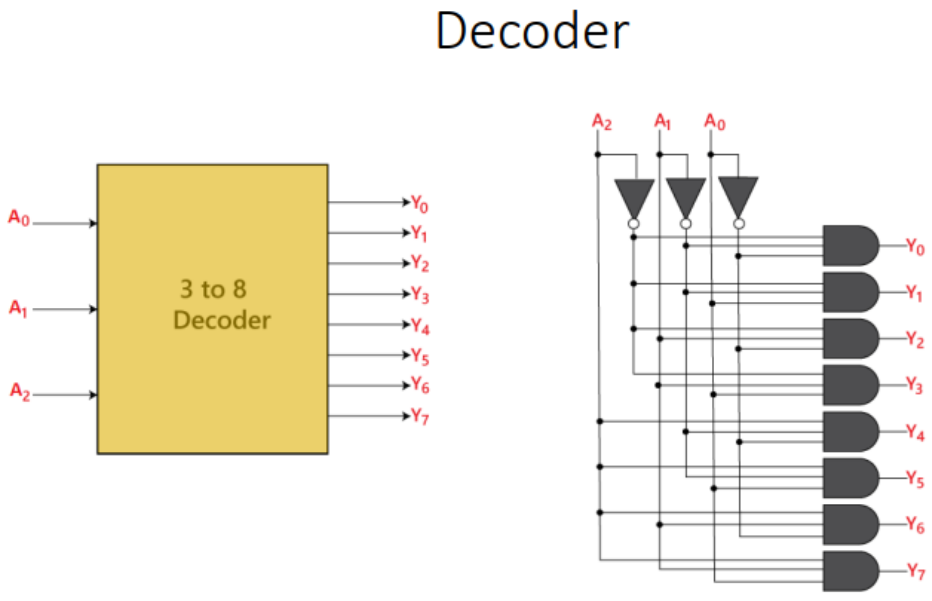
چکیده:

در این جلسه ابتدا اقداماتِ انجام‌شده در جلسه‌ی گذشته به‌طورِ کلی مرور و سپس با اجزای تشکیل‌دهنده‌ی انواعِ مختلف دیکودر و بردهای سون‌سگمنت آشنایی به عمل آمد.

بعد از پایانِ این بخش، دانشجویان طراحی و پیاده‌سازیِ بردهای سون‌سگمنت به کمک زبانِ سطح‌بالای Verilog را در گروه‌های 2 نفره آغاز کردند.

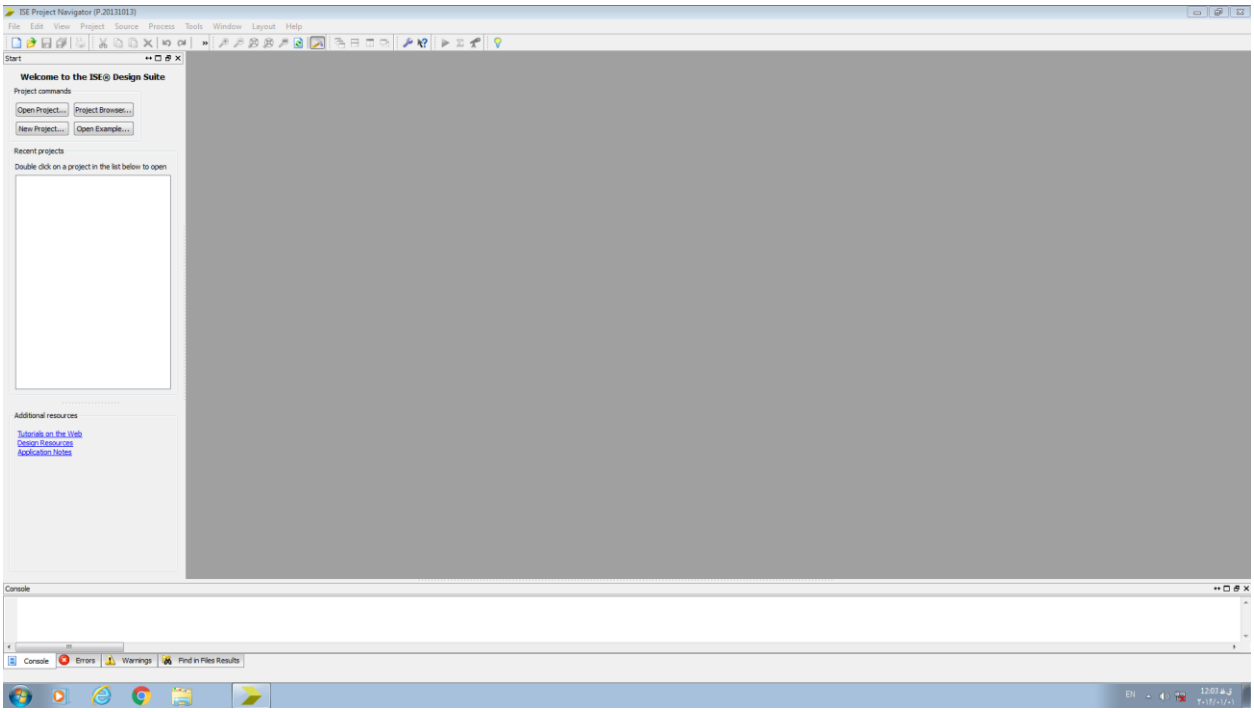
در ادامه ابتدا به معرفیِ کلیِ دیکودرها و بردهای سون‌سگمنت پرداخته و به کمک چند تصویر و جدول، با ویژگی‌های مختلف آن‌ها آشنا می‌شویم.

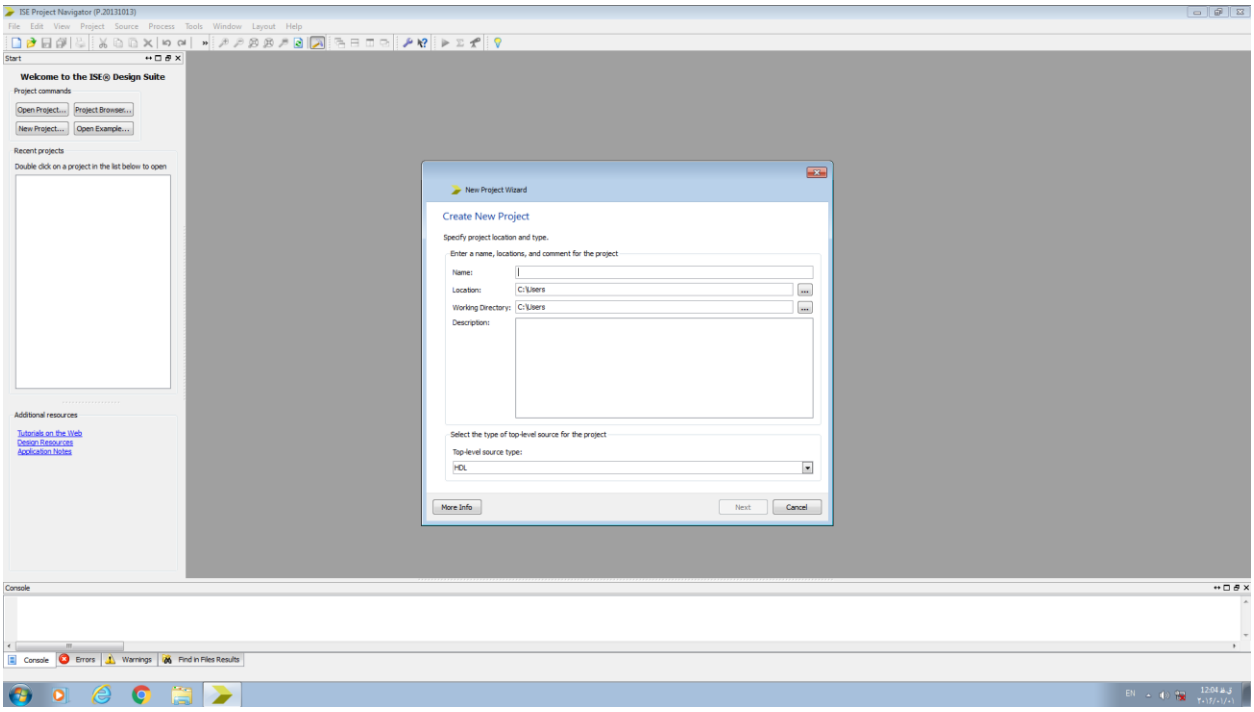
شکل زیر ترتیب اتصال گیت‌های مختلف برای طراحی و ساخت یک دیکودر 3 به 8 را نشان می‌دهد:

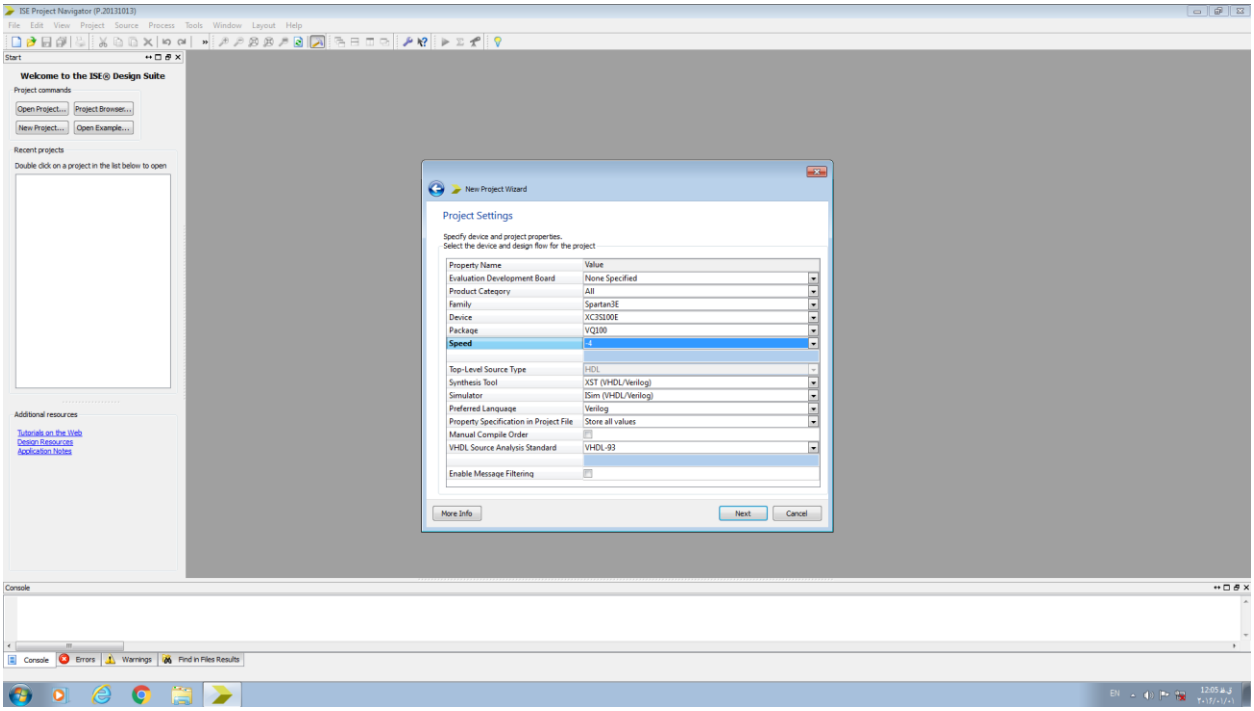


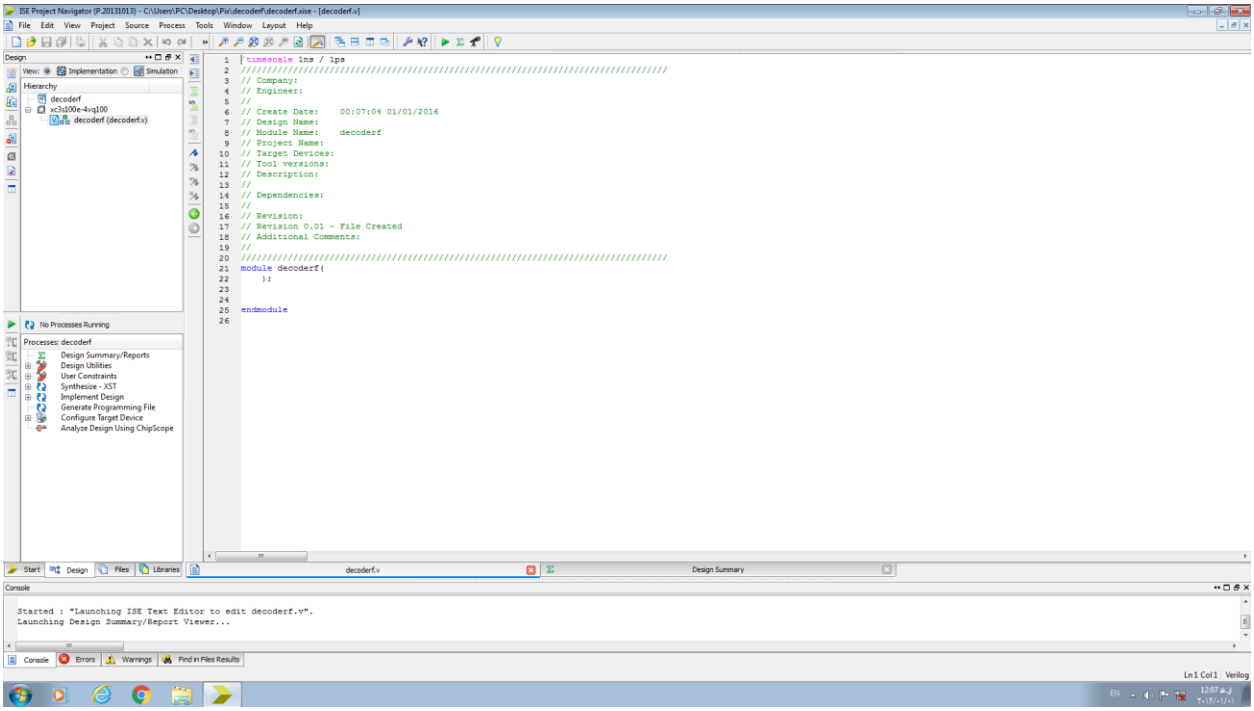
برای شبیه‌سازیِ این نوع از دیکودرها به کمک زبان برنامه‌نویسیِ Verilog همچون گذشته این بار هم پس از اجرای برنامه‌ی ISE، ابتدا پروژه‌ی جدیدی تعریف کرده و سپس یک فایلِ Source به آن اضافه می‌کنیم.

مراحلِ انجامِ این کار در تصاویرِ زیر آورده شده‌اند:

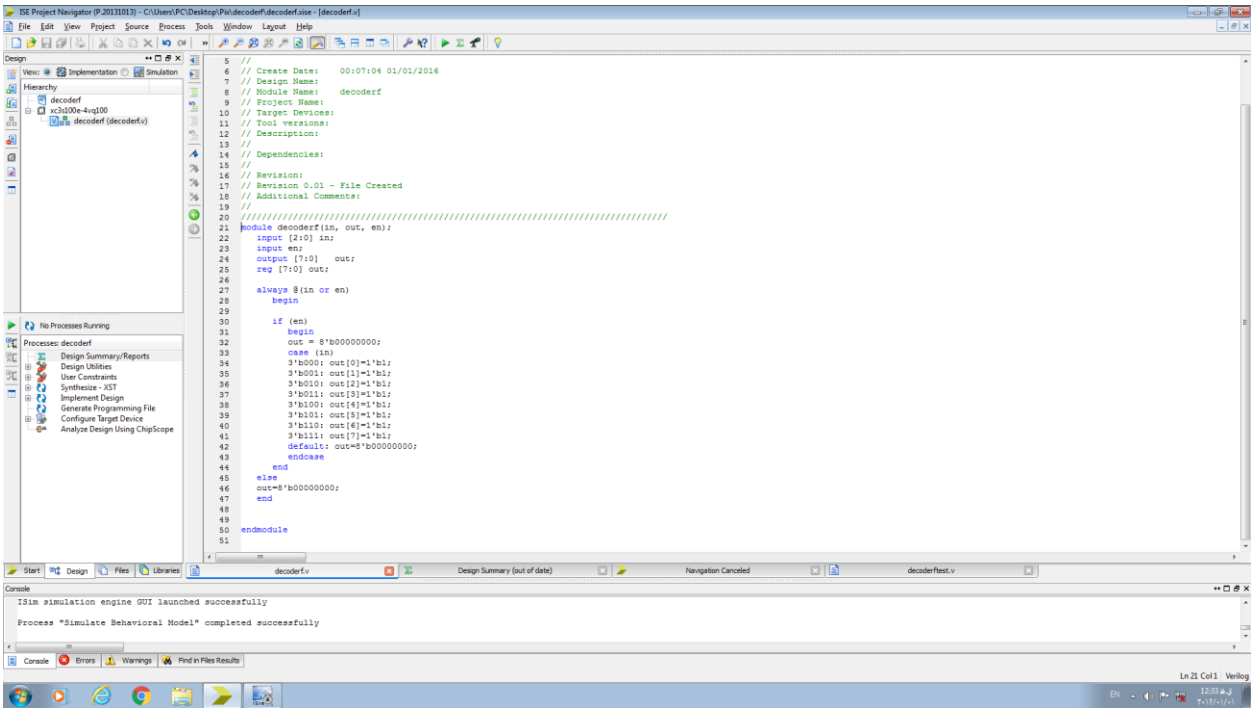





ISE Project Navigator (P.20131013)
Welcome to the ISE® Design Suite
Project commands
Open Project...
Project Browser...
New Project...
Open Example...
Recent projects
Double click on a project in the list below to open
Additional resources
Tutorials on the Web
Design Resources
Application Notes
New Project Wizard
Project Settings
Specify device and project properties.
Select the device and design flow for the project
Property Name
Value
Evaluation Development Board
None Specified
Product Category
All
Family
Spartan3E
Device
XC3S100E
Package
VQ100
Speed
-4
Top-Level Source Type
HDL
Synthesis Tool
XST (VHDL/Verilog)
Simulator
ISim (VHDL/Verilog)
Preferred Language
Verilog
Property Specification in Project File
Store all values
Manual Compile Order
VHDL Source Analysis Standard
VHDL-93
Enable Message Filtering
More Info
Next
Cancel
Console
Errors
Warnings
Find in Files Results

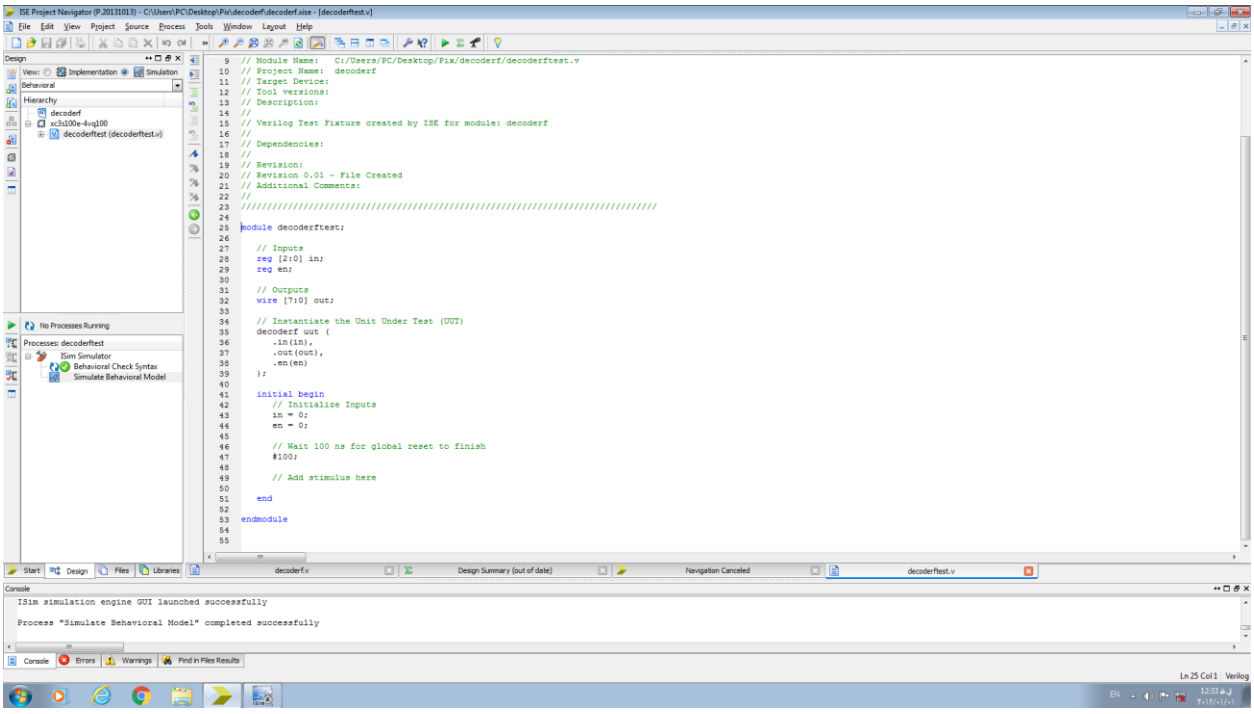

ISE Project Navigator (P.20131013) - C:\Users\PC\Desktop\Pix\decoderf\decoderf.xise - [decoderf.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
decoderf
xc3s100e-4vq100
decoderf (decoderf.v)
No Processes Running
Processes: decoderf
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
timescale 1ns / 1ps
// Company:
// Engineer:
// Create Date: 00:07:04 01/01/2016
// Design Name:
// Module Name: decoderf
// Project Name:
// Target Devices:
// Tool versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
module decoderf(
);
endmodule
Start
Design
Files
Libraries
decoderf.v
Design Summary
Console
Started : "Launching ISE Text Editor to edit decoderf.v".
Launching Design Summary/Report Viewer...
Console
Errors
Warnings
Find in Files Results
Ln 1 Col 1
Verilog

اولین قطعه کدی که برای طراحیِ یک دیکودر نیاز به اجرای آن داریم در تصویر زیر آورده شده:



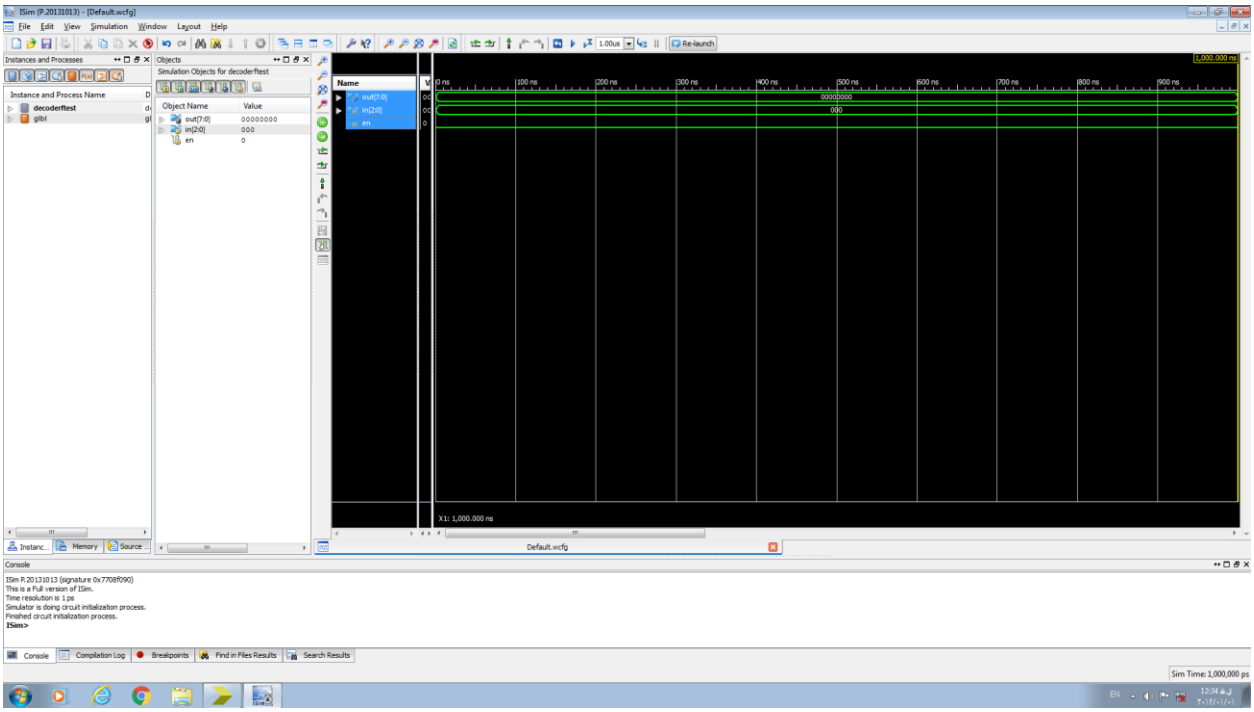
```
//
// Create Date:    00:07:04 01/01/2016
// Design Name:
// Module Name:    decoderf
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module decoderf(in, out, en);
    input [2:0] in;
    input en;
    output [7:0]   out;
    reg [7:0] out;

    always @(in or en)
       begin

          if (en)
             begin
             out = 8'b00000000;
             case (in)
             3'b000: out[0]=1'b1;
             3'b001: out[1]=1'b1;
             3'b010: out[2]=1'b1;
             3'b011: out[3]=1'b1;
             3'b100: out[4]=1'b1;
             3'b101: out[5]=1'b1;
             3'b110: out[6]=1'b1;
             3'b111: out[7]=1'b1;
             default: out=8'b00000000;
             endcase
          end
    else
    out=8'b00000000;
    end


endmodule
```

تصویر زیر نیز نشان‌دهنده‌ی کدهای مورد نیاز برای ارزیابی و تست کردن عملکرد برنامه است:



```
// Module Name:   C:/Users/PC/Desktop/Pix/decoderf/decoderftest.v
// Project Name:  decoderf
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: decoderf
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module decoderftest;

   // Inputs
   reg [2:0] in;
   reg en;

   // Outputs
   wire [7:0] out;

   // Instantiate the Unit Under Test (UUT)
   decoderf uut (
      .in(in),
      .out(out),
      .en(en)
   );

   initial begin
      // Initialize Inputs
      in = 0;
      en = 0;

      // Wait 100 ns for global reset to finish
      #100;

      // Add stimulus here

   end

endmodule
```

سپس با کامپایل کردن برنامه و اجرای قطعه‌ی تست به نتایج زیر دست می‌یابیم:

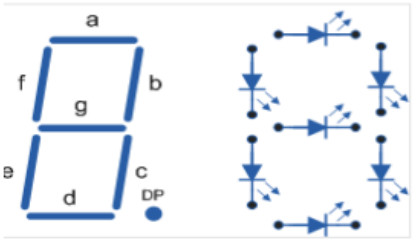


در قسمتِ بعدیِ این گزارش ابتدا به طور مختصر، با قطعاتِ سون‌سگمنت آشنا می‌شویم.

تصاویرِ زیر ساختارِ این قطعات و شیوه‌ی طراحیِ آن‌ها به کمک گیت‌های منطقی را نشان می‌دهند:

Seven segment

- Structure



کاراکتر متناظر	سگمنت‌های نمایشگر						
	a	b	c	d	e	f	g
0	×	×	×	×	×	×	
1		×	×				
2	×	×		×	×		×
3	×	×	×	×			×
4		×	×			×	×
5	×		×	×		×	×
6	×		×	×	×	×	×
7	×	×	×				

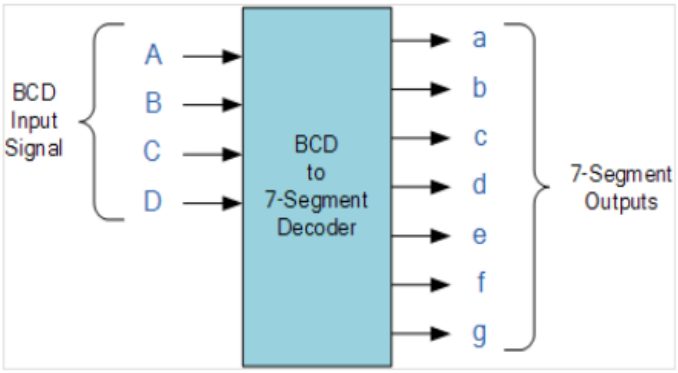
کاراکتر متناظر	سگمنت‌های نمایشگر						
	a	b	c	d	e	f	g
8	×	×	×	×	×	×	×
9	×	×	×	×		×	×
A	×	×	×		×	×	×
b			×	×	×	×	×
C	×			×	×	×	
d		×	×	×	×		×
E	×			×	×	×	×
F	×				×	×	×

Binary Coded Decimal(BCD)

دهدهی	الگوی باینری				BCD
	8	4	2	1	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7

دهدهی	الگوی باینری				BCD
	8	4	2	1	
8	1	0	0	0	8
9	1	0	0	1	9
10	1	0	1	0	Invalid
11	1	0	1	1	Invalid
12	1	1	0	0	Invalid
13	1	1	0	1	Invalid
14	1	1	1	0	Invalid
15	1	1	1	1	Invalid

BCD decoding to seven segment



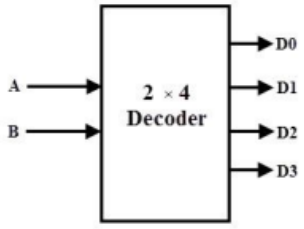
همانطور که قبلا اشاره شد، برای طراحی و پیاده‌سازیِ انواعِ مختلفِ دیکودر به کمک زبان برنامه‌نویسیِ Verilog روش‌های مختلفی وجود دارد.

در این قسمت با استفاده از مفاهیم Case و عبارت‌های شرطی و استفاده از حلقه‌ها به عنوانِ عملگرهای اصلی، یک دیکودرِ 2 به چهار را شبیه‌سازی خواهیم کرد.

Assign Statements

Nested Condition Operations

Describing a Decoder



Assign Statements

Nested Condition Operations

Describing a Decoder

```
`timescale 1ns/100ps

module dcd2_4 (input a, b, output d0, d1, d2, d3 );
   assign
      {d3, d2, d1, d0} = ( {a, b} == 2'b00 ) ? 4'b0001 :
                         ( {a, b} == 2'b01 ) ? 4'b0010 :
                         ( {a, b} == 2'b10 ) ? 4'b0100 :
                         ( {a, b} == 2'b11 ) ? 4'b1000 :
                         4'b0000;
endmodule
```

Assign Statements

Nested Condition Operations

Test for Decoder

```
module test_dcd;

    // Inputs
    reg a;
    reg b;
    // Outputs
    wire d0;
    wire d1;
    wire d2;
    wire d3;
    // Instantiate the Unit Under Test
    dcd2_4 uut (
        .a(a),
        .b(b),
        .d0(d0),
        .d1(d1),
        .d2(d2),
        .d3(d3)
    );

    initial begin
        // Initialize Inputs
        a = 0;
        b = 0;
        #50 a = 0; b = 1;
        #50 a = 1; b = 0;
        #50 a = 1; b = 1;
        // Wait 100 ns for global reset
        #100;
        // Add stimulus here
    end
endmodule
```

Assign Statements

Procedural case statement

Decoder Using case Statement

```
module dcd2_4 (input a, b, output reg d0, d1, d2, d3 );

   always @(a, b) begin
      case ( { a, b } )
         2'b00 : { d3, d2, d1, d0 } = 4'b0001;
         2'b01 : { d3, d2, d1, d0 } = 4'b0010;
         2'b10 : { d3, d2, d1, d0 } = 4'b0100;
         2'b11 : { d3, d2, d1, d0 } = 4'b1000;
         default: { d3, d2, d1, d0 } = 4'b0000;
      endcase
   end

endmodule
```

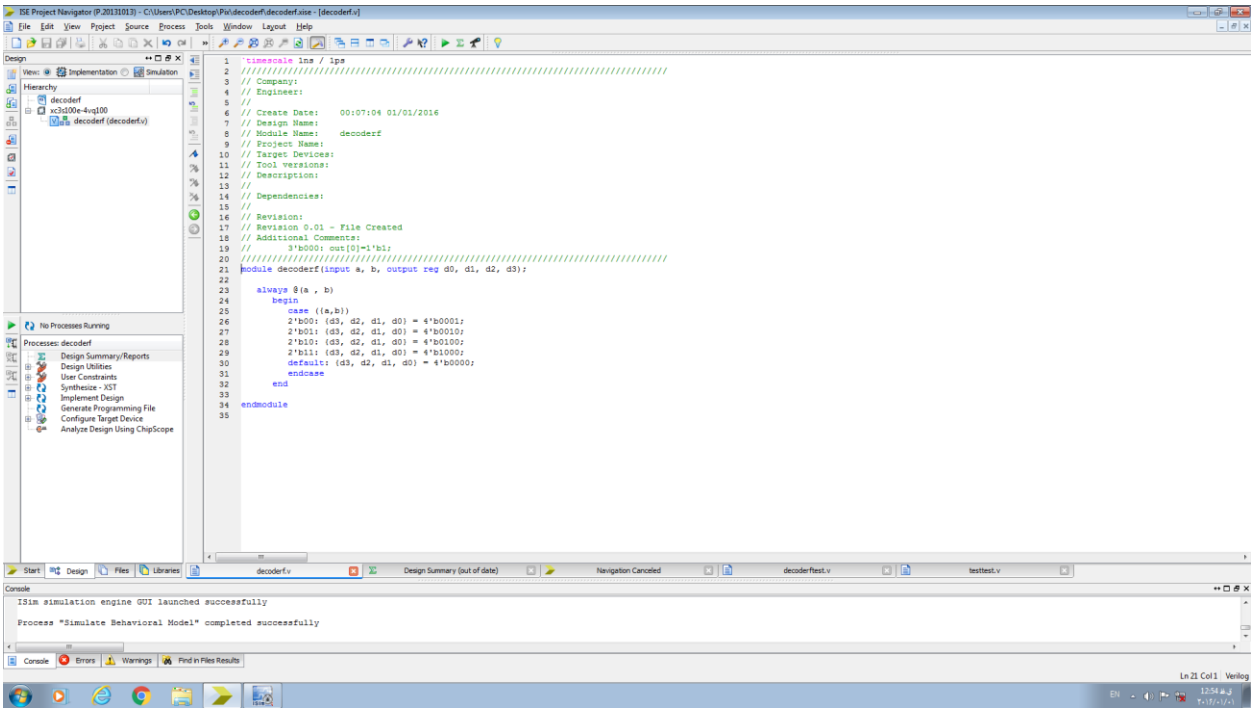
Assign Statements

Procedural case statement

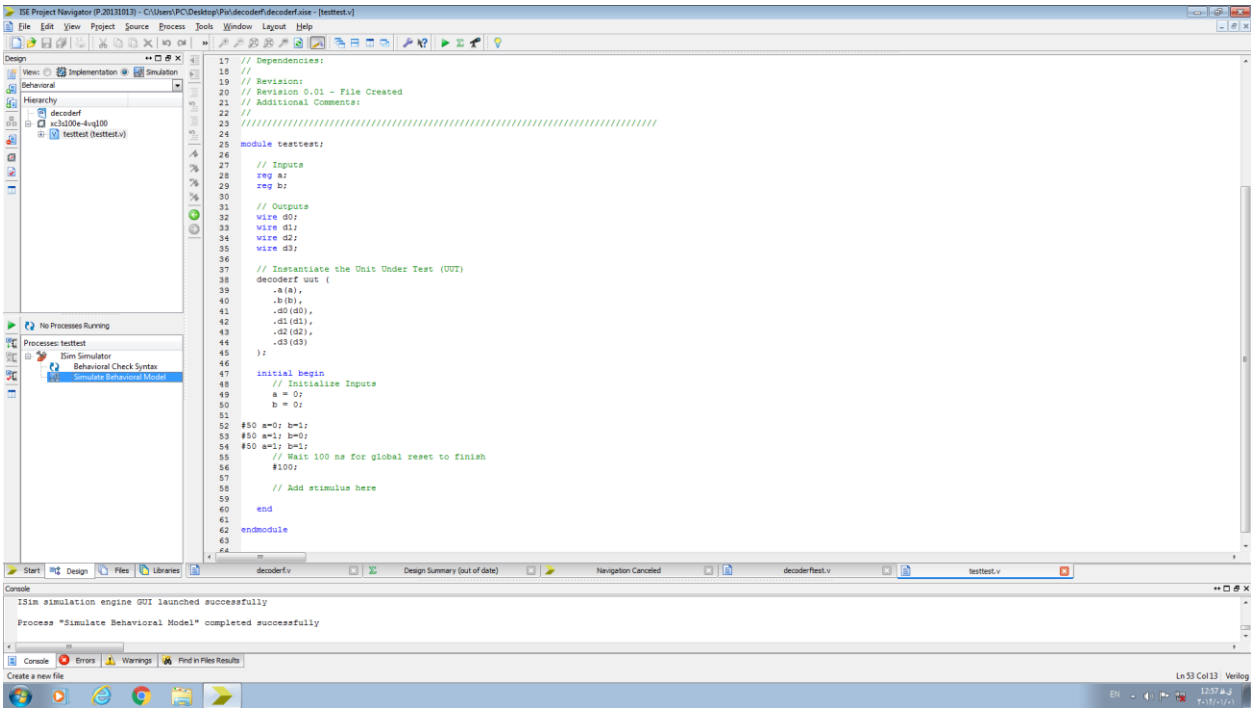
Decoder Using case Statement

```
module test_dcd;

    // Inputs
    reg a;
    reg b;
    // Outputs
    wire d0;
    wire d1;
    wire d2;
    wire d3;
    // Instantiate the Unit Under Test
    dcd2_4 uut (
        .a(a),
        .b(b),
        .d0(d0),
        .d1(d1),
        .d2(d2),
        .d3(d3)
    );

    initial begin
        // Initialize Inputs
        a = 0;
        b = 0;
        #50 a = 0; b = 1;
        #50 a = 1; b = 0;
        #50 a = 1; b = 1;
        // Wait 100 ns for global reset
        #100;
        // Add stimulus here
    end
endmodule
```

حاصلِ نوشتنِ برنامه‌های فوق در یک فایلِ Source جدید، در شکل زیر نشان داده شده است:



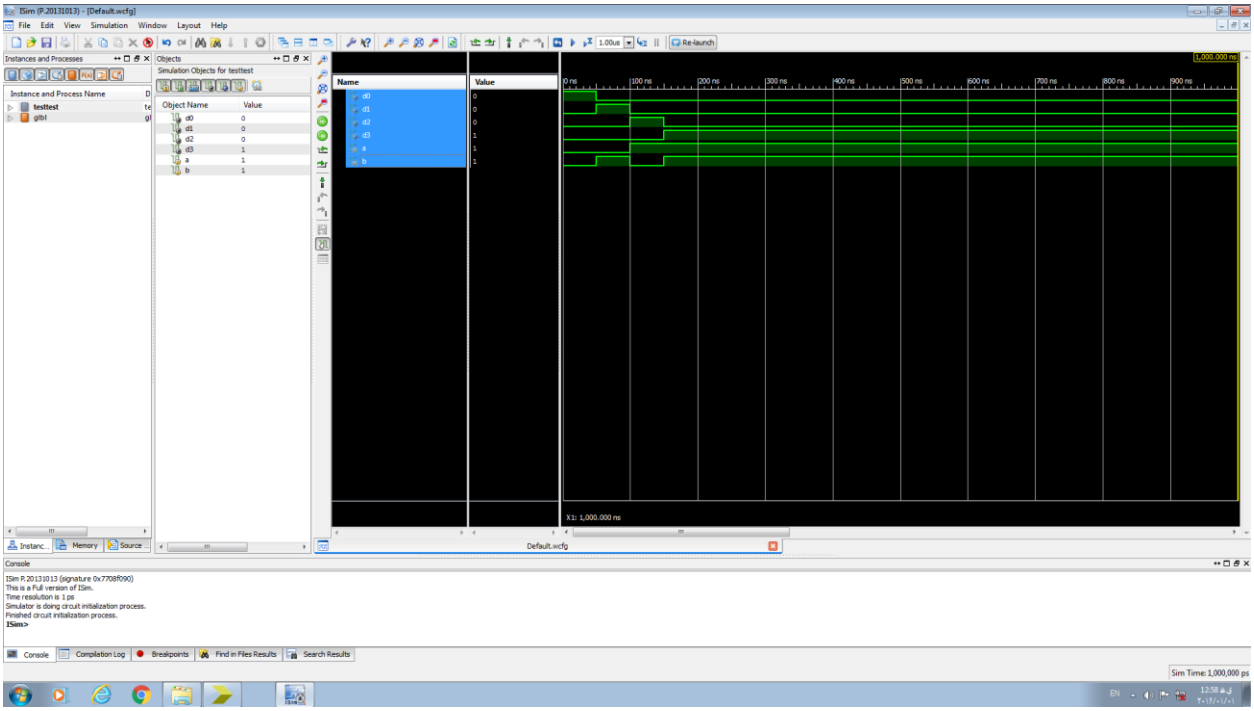
پیش از تکمیل کردن مرحله‌ی کامپایل و اجرای شبیه‌سازی، لازم است که کدهای مورد نیاز برای ارزیابی و تستِ عملکرد برنامه را نیز در یک فایل جداگانه بنویسیم:



```
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module testtest;

   // Inputs
   reg a;
   reg b;

   // Outputs
   wire d0;
   wire d1;
   wire d2;
   wire d3;

   // Instantiate the Unit Under Test (UUT)
   decoderf uut (
      .a(a),
      .b(b),
      .d0(d0),
      .d1(d1),
      .d2(d2),
      .d3(d3)
   );

   initial begin
      // Initialize Inputs
      a = 0;
      b = 0;

#50 a=0; b=1;
#50 a=1; b=0;
#50 a=1; b=1;
      // Wait 100 ns for global reset to finish
      #100;

      // Add stimulus here

   end

endmodule
```

```
ISim simulation engine GUI launched successfully

Process "Simulate Behavioral Model" completed successfully
```

با کامپایل شدن و سپس اجرای فایل در یک محیط شبیه‌سازی، به نتیجه‌ی زیر خواهیم رسید:



پایان.