به نام خدا

آموزش کار با نرم افزار Xilinx ISE Design Suite

قسمت سوم : تمرین کد نویسی

برای اینکه شما بتوانید با آموزش ها بیشتر همراه باشید و استفاده ی بیشتری ببرید لازم دانستم که چند نمونه کد آموزشی Verilog را قرار بدهم تا با برنامه نویسی به زبان Verilog بیشتر آشنا شوید. در ادامه ی آن هم چند سوال قرار دادم تا تمرین کد نویسی کرده باشید.

با انجام این کار ها من می توانم با اطمینان از این که مطالبم برای مخاطبان قابل فهم هست ، صحبت از نکات مربوط به برنامه نویسی کنم و وارد سطح پیشرفته ی آموزش شوم.

پس توجه شما را به ۴ مثال جلب می کنم ، پیشنهاد می کنم برای تسلط روی مطالبی که در قسمت های قبلی آموزش گفته شد ، کد های زیر را در Xilinx ISE برده و سنتز کنید و با توجه به ورودی ها و خروجی ها برای هر کد Testbench مناسب بنویسید و آن را شبیه سازی کنید .

تذکر : برخی مواقع که کدی را از جایی کپی می کنید ممکن است با این مشکل برخورد کنید که کد در عین این که درست است ولی Error دارد ، باید به این نکته توجه کنید که برخی علایم مثل (') که برای نوشتن 2b'01 بکار برده می شود ، در کپی شدن با علامتی شبیه (') در کد قرار می گیرد و طبیعتا ISE آن علامت را نمی شناسد و Error می دهد.

مثال ۱)

Encoder :

```
//-----------------------------------------------------
// Design Name : encoder_using_case
// File Name   : encoder_using_case.v
// Function    : Encoder using Case
//
//-----------------------------------------------------
module encoder_using_case(
binary_out , //  4 bit binary Output
encoder_in , //  16-bit Input
enable       //  Enable for the encoder
);
output [3:0] binary_out  ;
input  enable ;
input [15:0] encoder_in ;

reg [3:0] binary_out ;

always @ (enable or encoder_in)
begin
  binary_out = 0;
  if (enable) begin
    case (encoder_in)
      16'h0002 : binary_out = 1;
      16'h0004 : binary_out = 2;
      16'h0008 : binary_out = 3;
      16'h0010 : binary_out = 4;
      16'h0020 : binary_out = 5;
      16'h0040 : binary_out = 6;
      16'h0080 : binary_out = 7;
      16'h0100 : binary_out = 8;
      16'h0200 : binary_out = 9;
      16'h0400 : binary_out = 10;
      16'h0800 : binary_out = 11;
      16'h1000 : binary_out = 12;
      16'h2000 : binary_out = 13;
      16'h4000 : binary_out = 14;
      16'h8000 : binary_out = 15;
   endcase
  end
end

endmodule
```

مثال ۲) تقسیم بر ۲ :

```
//-----------------------------------------------------
// Design Name : clk_div
// File Name   : clk_div.v
// Function    : Divide by two counter
//
//-----------------------------------------------------
```

```
module clk_div (clk_in, enable,reset, clk_out);
 // --------------Port Declaration----------------------
 input               clk_in                  ;
 input               reset                   ;
 input               enable                  ;
 output              clk_out                 ;
 //--------------Port data type declaration-------------
 wire                 clk_in                 ;
 wire                 enable                 ;
//--------------Internal Registers----------------------
reg                   clk_out                ;
//--------------Code Starts Here------------------------
always @ (posedge clk_in)
if (reset) begin
  clk_out <= 1'b0;
end else if (enable) begin
  clk_out <= !clk_out ;
end

endmodule
```

مثال ۳) محاسبه ی Patitiy

```
//------------------------------------------------------
// Design Name : parity_using_assign
// File Name   : parity_using_assign.v
// Function    : Parity using assign
//
//------------------------------------------------------
module parity_using_assign (
data_in    , //  8 bit data in
parity_out   //  1 bit parity out
);
output  parity_out ;
input [7:0] data_in ;

wire parity_out ;

assign parity_out =  (data_in[0] ^ data_in[1]) ^
                     (data_in[2] ^ data_in[3]) ^
                     (data_in[4] ^ data_in[5]) ^
                     (data_in[6] ^ data_in[7]);

endmodule
```

مثال ۴)

Linear feedback shift register :

```
//----------------------------------------------------
// Design Name : lfsr
// File Name   : lfsr.v
// Function    : Linear feedback shift register
//
//----------------------------------------------------
module lfsr    (
out             ,  // Output of the counter
enable          ,  // Enable  for counter
clk             ,  // clock input
reset              // reset input
);

//----------Output Ports--------------
output [7:0] out;
//------------Input Ports--------------
input [7:0] data;
input enable, clk, reset;
//------------Internal Variables--------
reg [7:0] out;
wire        linear_feedback;

//-------------Code Starts Here-------
assign linear_feedback = !(out[7] ^ out[3]);

always @(posedge clk)
if (reset) begin // active high reset
  out <= 8'b0 ;
end else if (enable) begin
  out <= {out[6],out[5],
          out[4],out[3],
          out[2],out[1],
          out[0], linear_feedback};
end

endmodule // End Of Module counter
```

مثال های فوق از سایت www.asic-world.com آورده شده است.

اگر در مورد عنوان مثال های فوق اطلاعات کافی ندارید می توانید در wikipedia با یک جستجو اطلاعات لازم را بدست بیاورید.

برای علاقه مندان هم چند سوال مطرح می کنم تا بتوانند هرچه بیشتر بر کد نویسی به زبان Verilog مسلط شوند.

سوال ۱ : خروجی هر یک از ۲ کد زیر چیست ؟

کد اول :

```
module quest_for_out();

integer i;
reg clk;

initial begin
  clk = 0;
  #4 $finish;
end

always #1 clk = !clk;

always @ (posedge clk)
begin : FOR_OUT
  for (i=0; i < 8; i = i + 1) begin
    if (i == 5) begin
      disable FOR_OUT;
    end
    $display ("Current i : %g",i);
  end
end
endmodule
```

کد دوم :

```
module quest_for_in();

integer i;
reg clk;

initial begin
  clk = 0;
  #4 $finish;
end

always #1 clk = !clk;

always @ (posedge clk)
begin
  for (i=0; i < 8; i = i + 1) begin : FOR_IN
    if (i == 5) begin
      disable FOR_IN;
    end
    $display ("Current i : %g",i);
  end
end
endmodule
```

سوال ۲ : خروجی یک Mux2-1 وقتی که پایه ی انتخاب (Select) آن مقدار X دارد چیست ؟

سوال ۳ : ماژولی طراحی کنید که ورودی آن پالس ساعت باشد و عدد زوج (حداقل ۲ بیتی) ، کار این ماژول به این گونه است که فرکانس پاس ساعت ورودی را بر عدد زوج تقسیم کند و در خروجی پالس با فرکانس جدید را قرار دهد. این ماژول یک پایه ی Reset هم داشته باشد که با فعال شدن آن کد به حالت اولیه برگردد.

سوال ۴ : ماژولی طراحی کنید که فاصله ی لبه ی بالارونده ی دو ورودی را برحسب تعداد سیکل پالس ساعت محاسبه کند و در خروجی(۳۲بیتی) قرار دهد. فرض کنید که هر ورودی فقط یکبار از صفر به یک تغییر وضعیت می دهد. یک Reset هم داشته باشد که ماژول را به حالت اولیه ببرد.