SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECB302–VLSI DESIGN

III YEAR/ V SEMESTER

1/7/2023

UNIT 5-SPECIFICATION USING VERILOG HDL

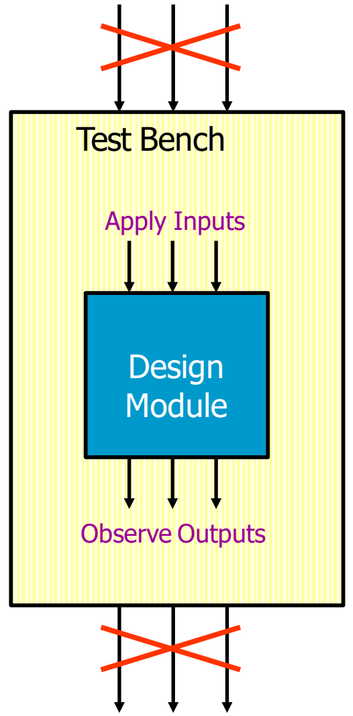
TOPIC 11,12 –**TEST BENCHES, SIX EXAMPLES**: DECODER, (EQUALITY DETECTOR) COMPARATOR, PRIORITY ENCODER, FULL ADDER, RIPPLE CARRY ADDER AND D FLIP FLOP.

OUTLINE

- TEST BENCHES
- FULL ADDER-BEHAVOURAL,STRUCTURAL EXAMPLES
- ACTIVITY
- DECODER,
- (EQUALITY DETECTOR) COMPARATOR,
- PRIORITY ENCODER,
- FULL ADDER,
- RIPPLE CARRY ADDER AND
- D FLIP FLOP.
- SUMMARY

TEST BENCH

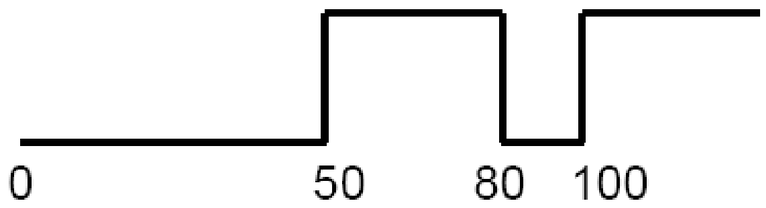
```
`timescale 1ns/100ps
module Top;
   reg PA, PB;
   wire PSum, PCarry;

   HalfAdder G1(PA, PB, PSum, PCarry);

   initial begin: LABEL
    reg [2:0] i;
    for (i=0; i<4; i=i+1) begin
     {PA, PB} = i;
     #5 $display ("PA=%b PB=%b PSum=%b
   PCarry=%b", PA, PB, PSum, PCarry);
    end // for
   end // initial
endmodule
```



TEST BENCH…

- Example: A sequence of values

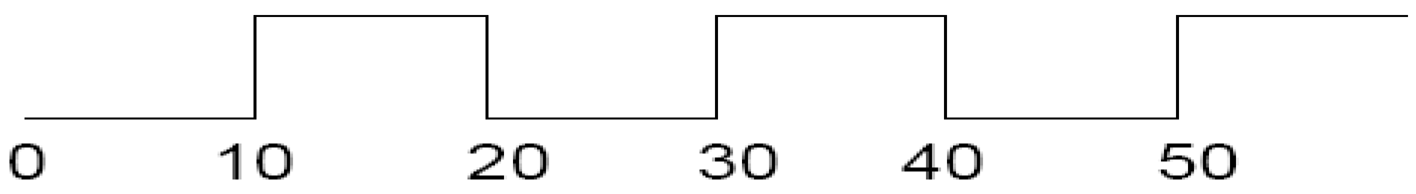
```
initial begin
  Clock = 0;
  #50 Clock = 1;
  #30 Clock = 0;
  #20 Clock = 1;
end
```



TEST BENCH…

- Repetitive Signals (clock)

Clock



- A Simple Solution:

  wire Clock;

  assign #10 Clock = ~ Clock

- Caution:
  – Initial value of Clock (***wire*** data type) = z
  – ~z = x and ~x = x

TEST BENCH…

- Initialize the Clock signal

```
initial begin
        Clock = 0;
    end
```

- Caution: Clock is of data type ***wire***, cannot be used in an ***initial*** statement
- Solution:

```
reg Clock;
…
initial begin
Clock = 0;
end
…
always begin
#10 Clock = ~ Clock;
end
```

forever loop can also be used to generate clock

VERILOG MODULE EXAMPLE & RTL VS STRUCTURAL

```
module Full_Adder_Behavioral_Verilog(
 input X1, X2, Cin,
 output S, Cout
 );
  reg[1:0] temp;
 always @(*)
 begin
 temp = {1'b0,X1} + {1'b0,X2}+{1'b0,Cin};
 end
 assign S = temp[0];
 assign Cout = temp[1];
endmodule
```

TEST BENCH FOR BEHAVIOURAL MODEL EXAMPLE

```
timescale 10ns/ 10ps;
module Testbench_Behavioral_adder();
 reg A,B,Cin;
 wire S,Cout;
//Verilog code for the structural full adder
Full_Adder_Behavioral_Verilog Behavioral_adder
(   .X1(A),   .X2(B),   .Cin(Cin),   .S(S),   .Cout(Cout)    );
 initial begin
 A = 0;  B = 0;  Cin = 0;
 #5;  A = 0;  B = 0;  Cin = 1;
     #5;   A = 0;  B = 1;  Cin = 0;
      #5;  A = 0;  B = 1;  Cin = 1;
  #5;  A = 1;  B = 0;  Cin = 0;
 #5;  A = 1;  B = 0;  Cin = 1;
      #5;  A = 1;  B = 1;  Cin = 0;
      #5;   A = 1;  B = 1;  Cin = 1;
  #5;   end    endmodule
```

EXAMPLE WITH A TEST FIXTURE

- A Full Adder

```
module testfixture;
reg    a, b, cin;
wire   sum, carry;
adder        u0 (carry, sum, a, b, cin);   initial
begin
$monitor($time, "a=%b b=%b
cin=%b sum=%b carry=%b",  a, b,
cin, sum, carry);
a=0; b=0; cin=0;
#10 a=0; b=0; cin=1;
#10 a=0; b=1; cin=0;
#10 a=0; b=1; cin=1;
#10 a=1; b=0; cin=0;
#10 a=1; b=0; cin=1;
#10 a=1; b=1; cin=0;
#10 a=1; b=1; cin=1;
#10 $stop; #10 $finish;  end
endmodule
```

```
module adder (carry, sum, a, b, cin);
output carry, sum;
input        a, b, cin;
Wire         w0, w1, w2;
xor  u0(sum, a, b, cin);
and u1(w0, a, b);
and u2(w1, b, cin);
and u3(w2, cin, b);
or    u4(carry, w0, w1, w2)
endmodule
```

This will generate some text outputs as 0

a=0 b=0 c=0 sum=0 carry=0

10 a=0 b=0 c=1 sum=1 carry=0

… …

TEST BENCH FOR STRUCTURAL MODEL EXAMPLE

```
Module Full_Adder_Structural_Verilog ( input X1, X2, Cin, output S, Cout );
    wire a1, a2, a3;
    xor u1(a1,X1,X2);
and  u2(a2,X1,X2);
and  u3(a3,a1,Cin);
or u4(Cout,a2,a3);
     xor u5(S,a1,Cin);
 endmodule
```

TEST BENCH FOR STRUCTURAL MODEL EXAMPLE

```
timescale 10ns/ 10ps;
module Testbench_structural_adder();
 reg A,B,Cin;
wire S,Cout;
//Verilog code for the structural full adder
 Full_Adder_Structural_Verilog structural_adder
(   .X1(A),   .X2(B),   .Cin(Cin),   .S(S),   .Cout(Cout)    );
 initial begin
A = 0;  B = 0;  Cin = 0;
 #10;  A = 0;  B = 0;  Cin = 1;
     #10;    A = 0;  B = 1;  Cin = 0;
     #10;  A = 0;  B = 1;  Cin = 1;
 #10;  A = 1;  B = 0;  Cin = 0;
#10;  A = 1;  B = 0;  Cin = 1;
    #10;  A = 1;  B = 1;  Cin = 0;
     #10;    A = 1;  B = 1;  Cin = 1;
  #10;   end    endmodule
```

MIXED MODLING STYLE

```
//mixed-design full adder
module full_adder_mixed (a, b, cin, sum, cout);
//list inputs and outputs
 input a, b, cin;
output sum, cout;
 //define reg and wires
reg cout;
wire a, b, cin;
wire sum;
wire net1;
//built-in primitive
xor (net1, a, b);
//behavioral
always @ (a or b or cin)
 begin
      cout = cin & (a ^ b) | (a & b);
end
//dataflow
assign sum = net1 ^ cin;        endmodule
```

MIXED MODLING STYLE-TEST BENCH

```
//mixed-design full adder test bench
 module full_adder_mixed_tb;
reg a, b, cin;
wire sum, cout;
//display variables
 initial
$monitor ("a b cin = %b %b %b, sum = %b, cout = %b", a, b, cin, sum, cout);
//apply input vectors
 initial
begin
#0 a = 1'b0; b = 1'b0; cin = 1'b0;
 #10 a = 1'b0; b = 1'b0; cin = 1'b1;
#10 a = 1'b0; b = 1'b1; cin = 1'b0;
#10 a = 1'b0; b = 1'b1; cin = 1'b1;
#10 a = 1'b1; b = 1'b0; cin = 1'b0;
 #10 a = 1'b1; b = 1'b0; cin = 1'b1;
 #10 a = 1'b1; b = 1'b1; cin = 1'b0;
#10 a = 1'b1; b = 1'b1; cin = 1'b1;
 #10 $stop; end
```

```
//instantiate the module into the
test bench full_adder_mixed inst1(
.a(a),
 .b(b),
.cin(cin),
 .sum(sum),
 .cout(cout) );
endmodule
```

ACTIVITY

GROUP DISCUSSION

SIX EXAMPLES: DECODER

- 3-to 8 decoder with an enable control

```
module decoder(o,enb_,sel) ;
output [7:0] o ;
 input enb_ ;
 input [2:0] sel ;
 reg [7:0] o ;
 always @ (enb_ or sel)
  if(enb_)
     o = 8'b1111_1111 ;
  else
   case(sel)
   3'b000 : o = 8'b1111_1110 ;
   3'b001 : o = 8'b1111_1101 ;
   3'b010 : o = 8'b1111_1011 ;
   3'b011 : o = 8'b1111_0111 ;
   3'b100 : o = 8'b1110_1111 ;
   3'b101 : o = 8'b1101_1111 ;
   3'b110 : o = 8'b1011_1111 ;
   3'b111 : o = 8'b0111_1111 ;
   default : o = 8'bx ;
   endcase
endmodule
```

PRIORITY ENCODER

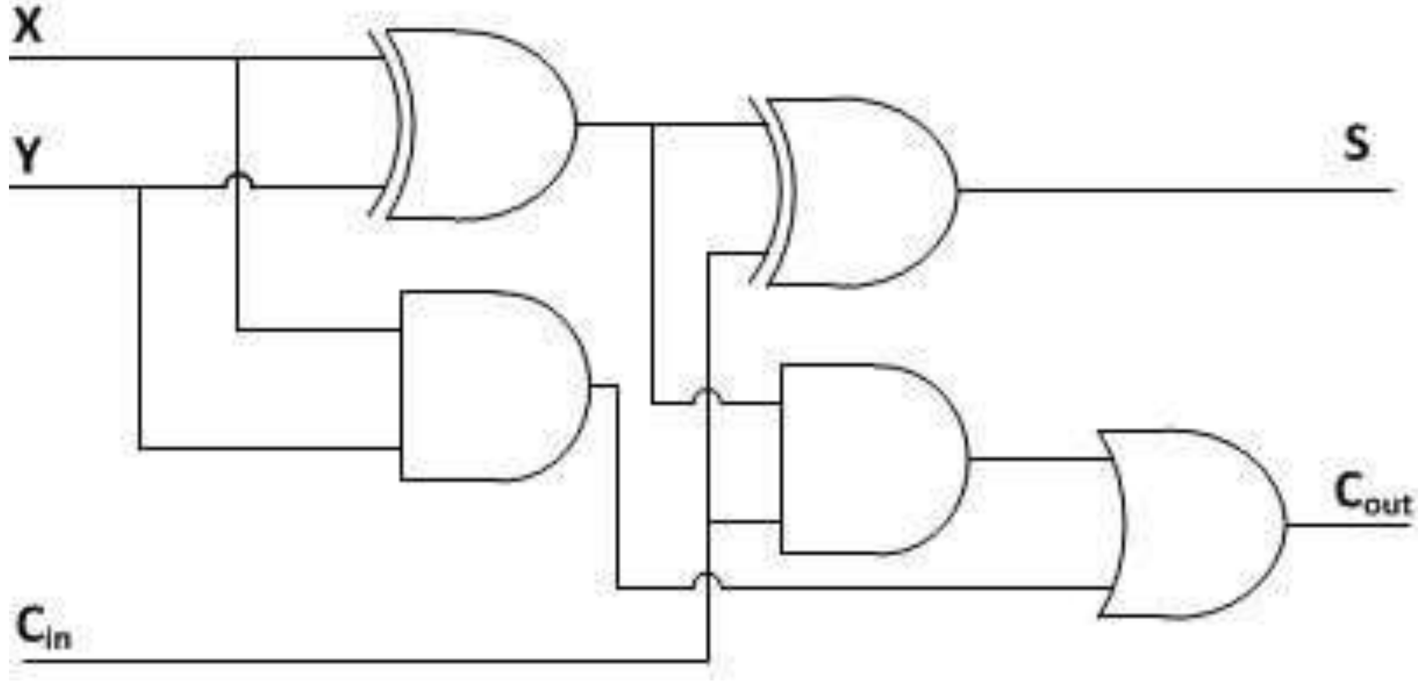
```
always @ (d0 or d1 or d2 or d3)
    if (d3 == 1)
        {x,y,v} = 3'b111 ;
    else if (d2 == 1)
        {x,y,v} = 3'b101 ;
    else if (d1 == 1)
        {x,y,v} = 3'b011 ;
    else if (d0 == 1)
        {x,y,v} = 3'b001 ;
    else
        {x,y,v} = 3'bxx0 ;
```

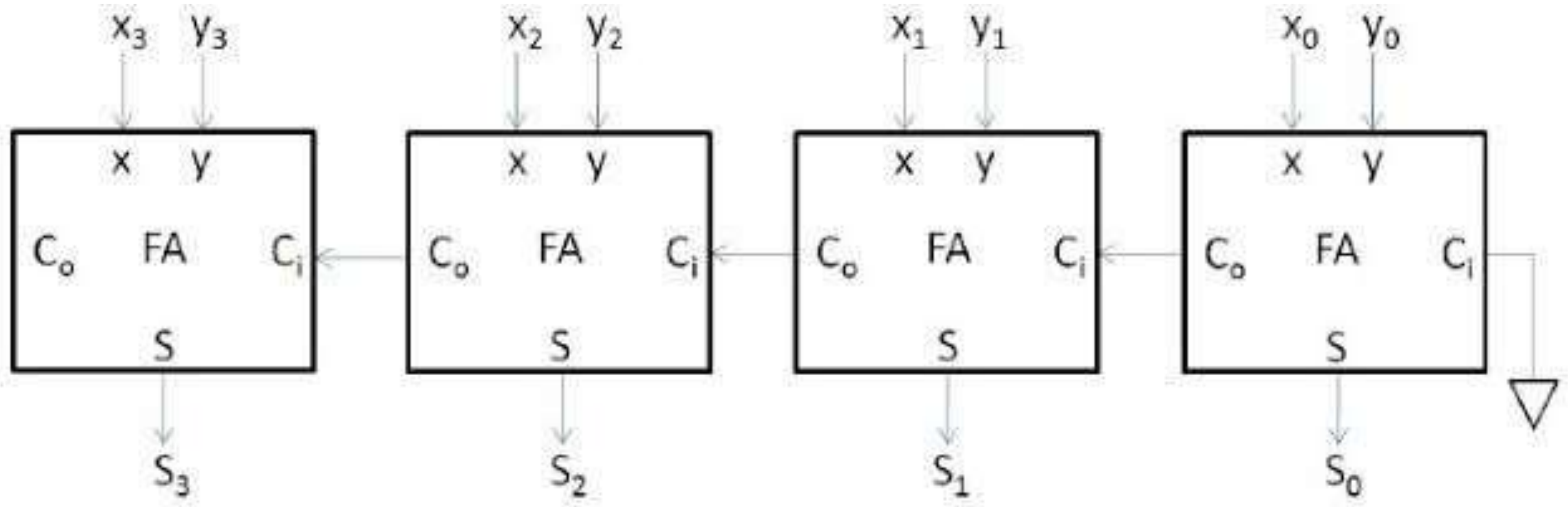
Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Priority encoder truth table

FULL ADDER, RIPPLE CARRY ADDER

full adder based on its logic diagram





The 4-bit ripple-carry adder is built using 4 1-bit full adders

VERILOG HDL CODE: 4-BIT RIPPLE-CARRY ADDER USING 4 1-BIT FULL ADDERS

```
module fulladder(X, Y, Ci, S, Co); input X, Y, Ci;
 output S, Co;
 wire w1,w2,w3;
//Structural code for one bit full adder
xor G1(w1, X, Y);
xor G2(S, w1, Ci);
and G3(w2, w1, Ci);
and G4(w3, X, Y);
or G5(Co, w2, w3);
 endmodule
```

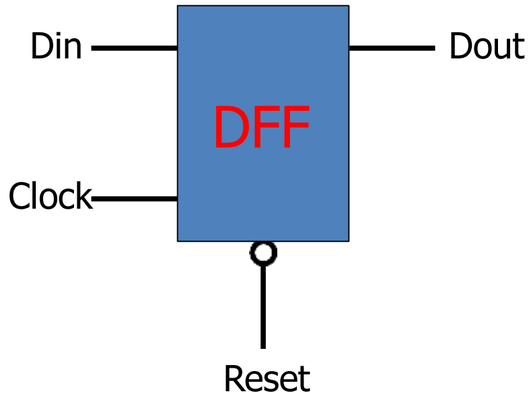
```
module rippe_adder(X, Y, S, Co);
input [3:0] X, Y;// Two 4-bit inputs
output [3:0] S; output Co;
wire w1, w2, w3;
 // instantiating 4 1-bit full adders in Verilog
fulladder u1(X[0], Y[0], 1'b0, S[0], w1);
fulladder u2(X[1], Y[1], w1, S[1], w2);
 fulladder u3(X[2], Y[2], w2, S[2], w3);
fulladder u4(X[3], Y[3], w3, S[3], Co);
endmodule
```

VERILOG HDL CODE:COMPARATOR & D FLIPFLOP

COMPARATOR

```
module comparator(large, equal, less, a, b);
output      large, equal, less;
input  [3:0] a, b;
assign      large = (a > b);
assign      equal = (a == b);
assign      less = (a < b);
endmodule
```

```
module DFF ( Din, Dout, Clock, Reset );
output Dout;
input Din, Clock, Reset;
reg    Dout;
always @( negedge Reset or posedge Clock )
begin
   if ( !Reset )
      Dout <= 1'b0;
   else
      Dout <= Din;
end
endmodule
```



ASSESSMENT

1.Write the test bench for behavioural model example

2. Write the Verilog HDL Code for : DECODER, PRIORITY ENCODER

3. Write the Verilog HDL Code for FULL ADDER, RIPPLE CARRY ADDER

4. Write the Verilog HDL Code for COMPARATOR AND D FLIP FLOP.

SUMMARY & THANK YOU