Verilog

INEL 4205 - November 2009

```
module counter(out, clk, reset);
    parameter WIDTH = 4;

    output    [WIDTH-1 : 0] out;
    reg       [WIDTH-1 : 0] out;
    input clk, reset;
    wire      clk, reset;

    always @(posedge clk)
      if (~reset)
          out = out + 1;

    always @(reset)
     if (reset)  out = 0;

endmodule // counter
```

```
module test;

    reg reset, clk;
    wire [3:0] value;
    initial
        begin
            // system command to create a file for gtkview
            $dumpfile("icounter.vcd");
            $dumpvars ;

            // Monitor variables "reset" or "value"
             $monitor("Time = %g reset = %b count = %b",$time, reset, value);

            // Do the simulation
             reset = 0;
             clk = 0;
            # 17 reset = 1;
            # 11 reset = 0;
            # 29 reset = 1;
            # 11 reset = 0;
            # 300 $finish;
         end

     /* Make a regular pulsing clock. */
     always #5 clk = !clk;

     counter c1 (value, clk, reset);   // instantiate module counter

endmodule // test
```

Table 4-10
Verilog HDL Operators

Symbol	Operation
+	binary addition
−	binary subtraction
&	bit-wise AND
|	bit-wise OR
^	bit-wise XOR
~	bit-wise NOT
==	equality
>	greater than
<	less than
{ }	concatenation
? :	conditional

Signals

- Nets - represent structural connections
  - "wire" is the most common net type
- reg - represent variables used to store data
  - store the last value assigned to them until another assignment statement changes their value
- Constants:

```
3'b010; // 3 bits, equal to binary 010

3'd2; // 3 bits, equal to decimal 2

8'hAB; // 8 bits, equal to hex AB = 10101011₂
```

Continuous Assignment Statement:

- used to assign a value to a net type (i.e. a wire) outside of an *always* block
- taken as concurrent (happening at the same time)
- assign statement is implied if done when declaring the variable

HDL Example 3-4

```
//Circuit specified with Boolean expressions
module circuit_bln (x,y,A,B,C,D);
    input A,B,C,D;
    output x,y;
    assign x = A | (B & C) | (~B & D);
    assign y = (~B & C) | (B & ~C & ~D);
endmodule
```

HDL Example 4-4

```
//Dataflow description of 4-bit adder
module binary_adder (A,B,Cin,SUM,Cout);
    input [3:0] A,B;
    input Cin;
    output [3:0] SUM;
    output Cout;
    assign {Cout,SUM} = A + B + Cin;
endmodule
```

HDL Example 4-5

```
//Dataflow description of a 4-bit comparator.
module magcomp (A,B,ALSB,AGTB,AEQB);
    input [3:0] A,B;
    output ALTB,AGTB,AEQB;
    assign ALTB=(A < B),
           AGTB = (A > B),
           AEQB = (A == B);
endmodule
```

HDL Example 4-6

```
//Dataflow description of 2-to-1-line multiplexer
module mux2x1_df (A,B,select,OUT);
    input A,B,select;
    output OUT;
    assign OUT = select ? A : B;
endmodule
```

Behavioral constructs

initial blocks - provide initial values for simulations

always block - use to contain the precedural constructs.

Always @(sensitivity_list)
[begin]
[procedural statements consisting of assignment, if, case, while, and for loops. May also include task and function calls.]
[end]

```
//HDL Example 5-1
//Description of D latch (See Fig.5-6)
module D_latch (Q,D,control);
   output Q;
   input D,control;
   reg Q;
   always @ (control or D)
   if (control) Q = D;
         //Same as: if (control = 1)
endmodule
```

```
module ccounter(clock, control, count);
   input clock ;
   input control ;
   output     count ;
   reg [2:0]  count ;

initial
  count=3'h0 ;

always @ (posedge clock)
   if (control)
      case (count)
         3'h0: count = 3'h1;
         3'h1: count = 3'h4;
         3'h4: count = 3'h5;
         3'h5: count = 3'h7;
         3'h7: count = 3'h0;
         default:
           count = 3'h5;
      endcase // case (count)
   else  // counting down
      case (count)
         3'h1: count = 3'h6;
         3'h2: count = 3'h1;
         3'h3: count = 3'h2;
         3'h5: count = 3'h3;
         3'h6: count = 3'h5;
         default:
           count = 3'h5;
      endcase // case (count)

endmodule // ccounter
```

```
module testb ; // stimulus or testbench

reg control, clk;
wire [2:0] value;

initial
   begin
      // create a dump of all variables to use gtkwave
     // to view
      $dumpfile("ccounter.vcd");
      $dumpvars ;

      // print a message on the screen each time a
     //  variable changes
      $monitor("Time = %g control = %b count = %b",
          $time, control, value);

      // now describe the simulation
      control = 1'b0 ;
      clk = 1'b0 ;
      #60 control = 1'b1;
      #60 control = 1'b0;
      #60 $finish;
   end // initial begin

// create clock
always
   #5 clk = ~clk;

// instantiate module under test
ccounter mycount(clk, control, value);
endmodule // testb
```

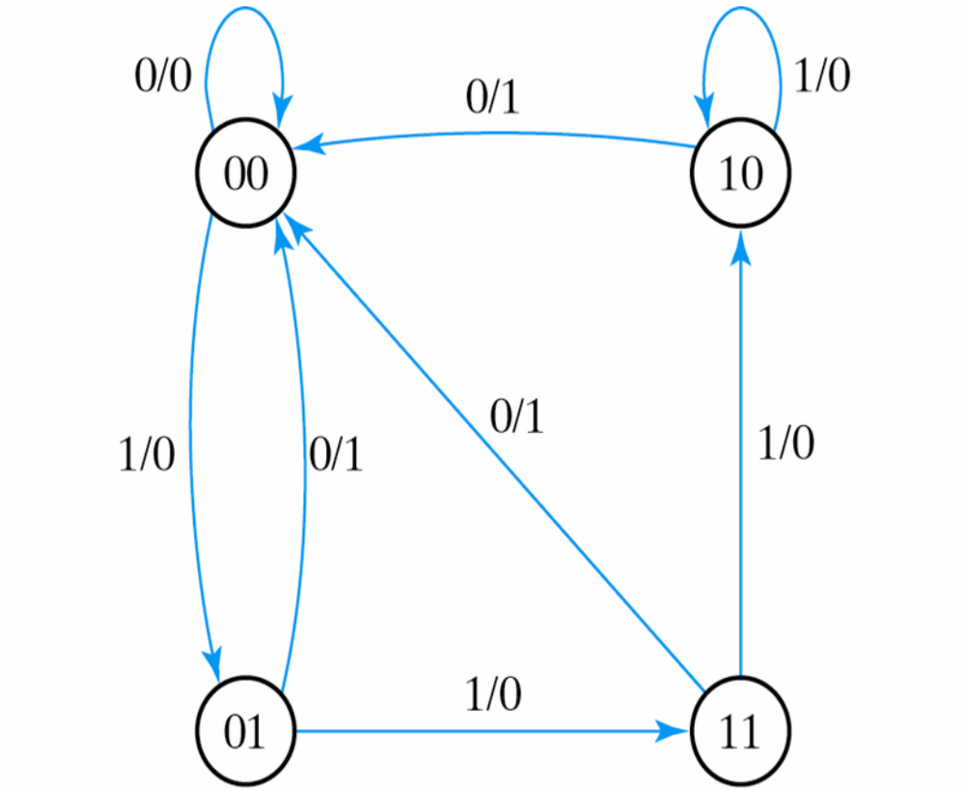

Fig. 5-16 State Diagram of the Circuit of Fig. 5-15

```
//HDL Example 5-5: Mealy state
   diagram (Fig 5-16)

module Mealy_mdl (x,y,CLK,RST);

 input x,CLK,RST;

 output y;

 reg y;

 reg [1:0] Prstate, Nxtstate;


parameter S0 = 2'b00, S1 = 2'b01, S2
  = 2'b10, S3 = 2'b11;

always @ (posedge CLK or negedge
 RST)

   if (~RST) Prstate = S0;  //Initialize
 to state S0

   else Prstate = Nxtstate; //Clock
 operations
```

```
  always @ (Prstate or x)     //Find next state
       case (Prstate)
          S0: if (x) Nxtstate = S1;
          S1: if (x) Nxtstate = S3;
                else Nxtstate = S0;
          S2: if (~x)Nxtstate = S0;
          S3: if (x) Nxtstate = S2;
                else Nxtstate = S0;
       endcase
  always @ (Prstate or x)     //Evaluate output
       case (Prstate)
          S0: y = 0;
          S1: if (x) y = 1'b0; else y = 1'b1;
          S2: if (x) y = 1'b0; else y = 1'b1;
          S3: if (x) y = 1'b0; else y = 1'b1;
       endcase
endmodule
```