# HDL FOR REGISTER

(Lecture-33)

# Introduction

- Registers and counters can be described in Verilog at either the behavioural or the structural level.
- Behavioural modelling describes only the operations of the register, as prescribed by a function table, without a preconceived structure.
- A structural-level description shows the circuit in terms of a collection of components such as gates, flip- flops, and multiplexers.
- The various components are instantiated to form a hierarchical description of the design similar to a representation of a multilevel logic diagram. Both are useful.
- When a machine is complex, a hierarchical description creates a physical partition of the machine into simpler and more easily described units.

# HDL Example (Universal Shift Register-Behavioral Model)

```
// Behavioral description of a 4-bit universal shift register
// Fig. 6.7 and Table 6.3
module Shift_Register_4_beh (                    // V2001, 2005
  output reg        [3: 0]    A_par,           // Register output
  input             [3: 0]    I_par,           // Parallel input
  input             s1, s0,                    // Select inputs
                    MSB_in, LSB_in,            // Serial inputs
                    CLK, Clear_b               // Clock and Clear
);

always @ (posedge CLK, negedge Clear_b)          // V2001, 2005
   if (Clear_b == 0) A_par <= 4'b0000;
   else
    case ({s1, s0})
      2'b00: A_par <= A_par;                     // No change
      2'b01: A_par <= {MSB_in, A_par[3: 1]};     // Shift right
      2'b10: A_par <= {A_par[2: 0], LSB_in};     // Shift left
      2'b11: A_par <= I_par;                     // Parallel load of input
    endcase
endmodule
```

Variables of type **reg** retain their value until they are assigned a new value by an assignment statement. Consider the following alternative **case** statement for the shift register model:

```
    case ({s1, s0})
      // 2'b00: A_par <= A_par;                  // No change
      2'b01: A_par <= {MSB_in, A_par [3: 1]};    // Shift right
      2'b10: A_par <= {A_par [2: 0], LSB_in};    // Shift left
      2'b11: A_par <= I_par;                     // Parallel load of input
    endcase
```

# HDL Example (Universal Shift Register-Structural Model)

```
// Structural description of a 4-bit universal shift register
module Shift_Register_4_str (                    // V2001, 2005
  output [3: 0] A_par,                           // Parallel output
  input [3: 0]  I_par,                           // Parallel input


 input        s1, s0,                            // Mode select
 input        MSB_in, LSB_in, CLK, Clear_b       // Serial inputs, clock, clear
);

// bus for mode control
 assign   [1:0]   select = {s1, s0};

// Instantiate the four stages
 stage ST0 (A_par[0], A_par[1], LSB_in, I_par[0], A_par[0], select, CLK, Clear_b);
 stage ST1 (A_par[1], A_par[2], A_par[0], I_par[1], A_par[1], select, CLK, Clear_b);
 stage ST2 (A_par[2], A_par[3], A_par[1], I_par[2], A_par[2], select, CLK, Clear_b);
 stage ST3 (A_par[3], MSB_in, A_par[2], I_par[3], A_par[3], select, CLK, Clear_b);
endmodule

// One stage of shift register
module stage (i0, i1, i2, i3, Q, select, CLK, Clr_b);
 input        i0,          // circulation bit selection
              i1,          // data from left neighbor or serial input for shift-right
              i2,          // data from right neighbor or serial input for shift-left
              i3;          // data from parallel input
 output       Q;
 input [1: 0] select;      // stage mode control bus
 input        CLK, Clr_b;  // Clock, Clear for flip-flops
 wire         mux_out;

// instantiate mux and flip-flop
 Mux_4_x_1  M0       (mux_out, i0, i1, i2, i3, select);
 D_flip_flop  M1     (Q, mux_out, CLK, Clr_b);
endmodule

// 4x1 multiplexer              // behavioral model
module Mux_4_x_1 (mux_out, i0, i1, i2, i3, select);
 output       mux_out;
 input        i0, i1, i2, i3;
 input [1: 0] select;
 reg          mux_out;
 always @ (select, i0, i1, i2, i3)
  case (select)
    2'b00:    mux_out = i0;
    2'b01:    mux_out = i1;
    2'b10:    mux_out = i2;
    2'b11:    mux_out = i3;
  endcase
endmodule

 // Behavioral model of D flip-flop
 module D_flip_flop (Q, D, CLK, Clr_b);
  output      Q;
  input       D, CLK, Clr;
  reg         Q;

  always @ (posedge CLK, negedge Clr_b)
   if (!Clr_b) Q <= 1'b0; else Q <= D;
 endmodule
```