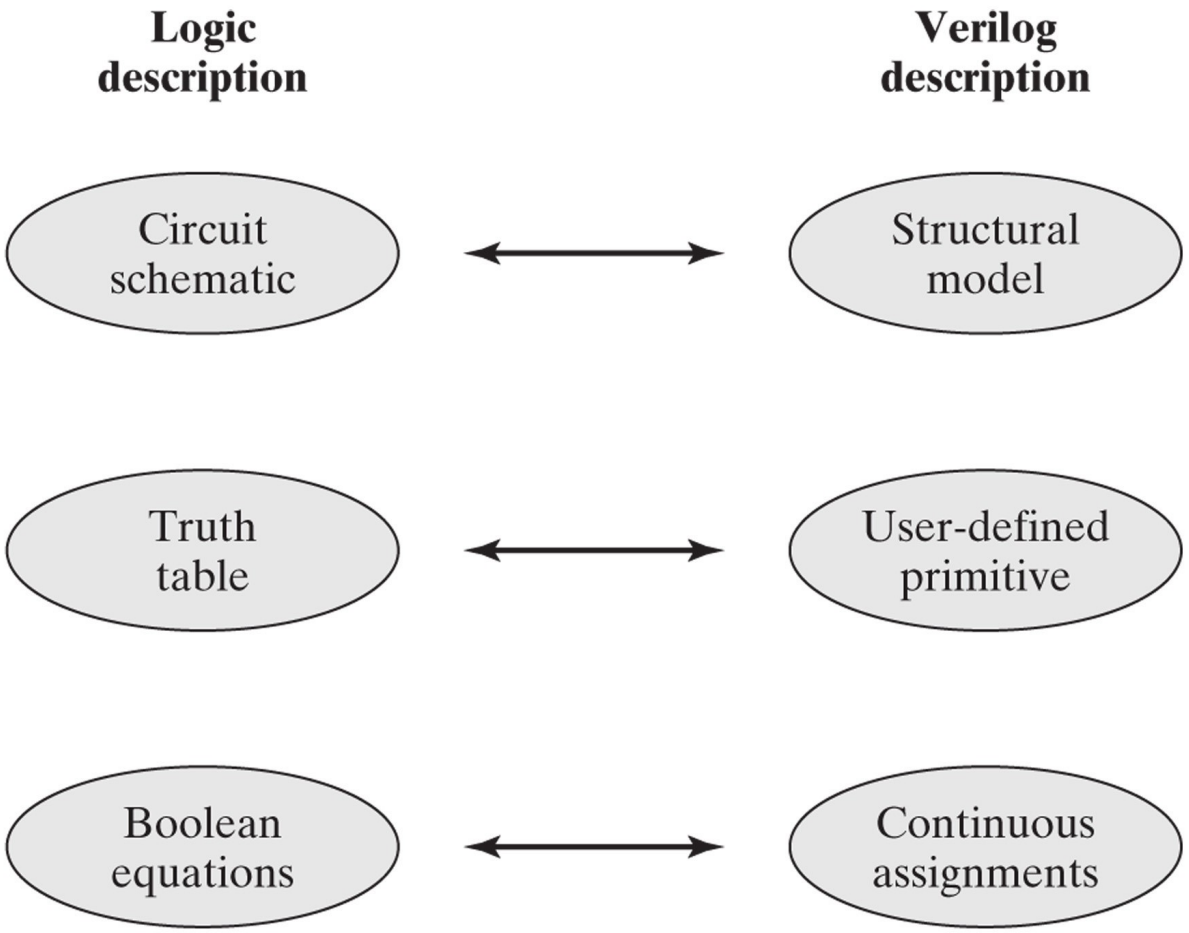
EEL 4783: HDL in Digital System Design

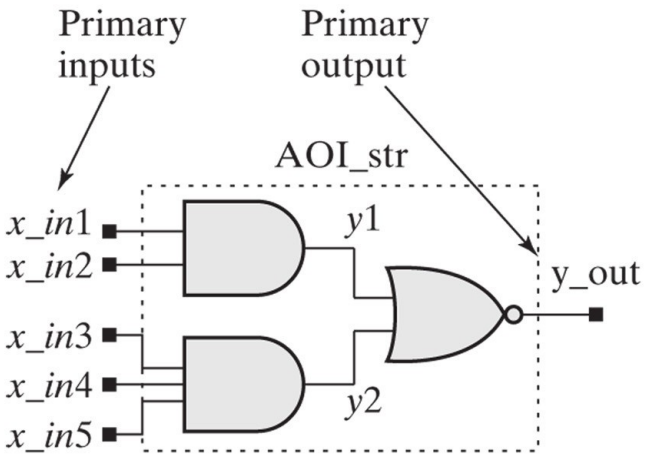
Lecture 3: Logic Design with Behavioral Models

Prof. Mingjie Lin

HDL Levels



An AOI Circuit



Module name

Module ports

```
module AOI_str_1995 (y_out, x_in1, x_in2, x_in3, x_in4, x_in5);
  output y_out;
  input  x_in1, x_in2, x_in3, x_in4, x_in5;

  wire   y1, y2;

  nor    (y_out, y1, y2);
  and    (y1, x_in1, x_in2);
  and    (y2, x_in3, x_in4, x_in5);
endmodule
```

Port modes

Internal wires establish connectivity

Instantiated primitives

(a)

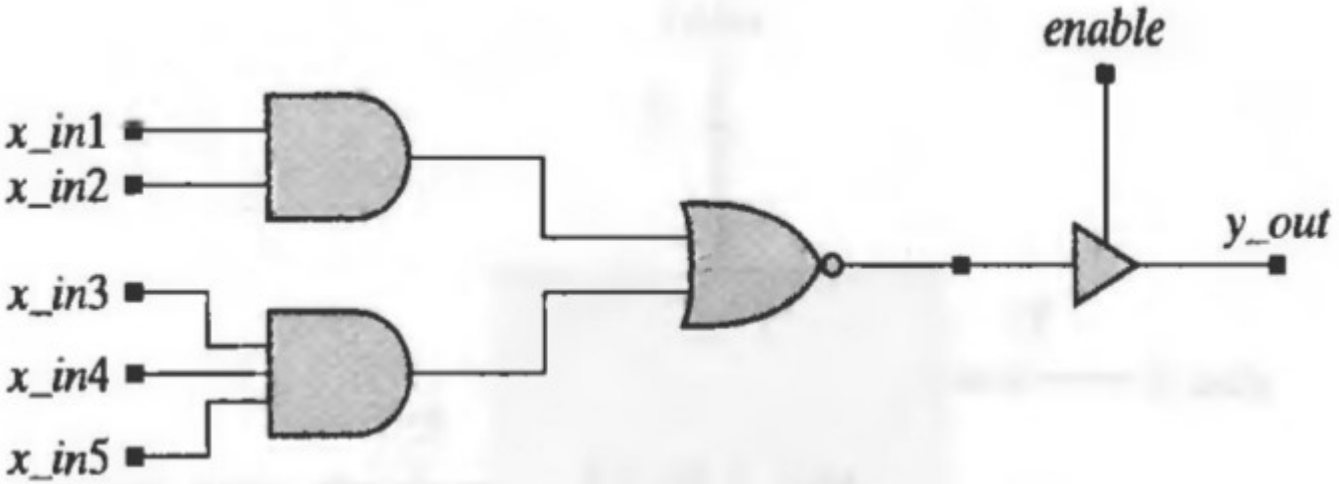
```
module AOI_str_2005 (output y_out, input x_in1, x_in2, x_in3, x_in4, x_in5);
 wire    y1, y2;

 nor     (y_out, y1, y2);
 and     (y1, x_in1, x_in2);
 and     (y2, x_in3, x_in4, x_in5);
endmodule
```

(b)

Behavior Descriptions

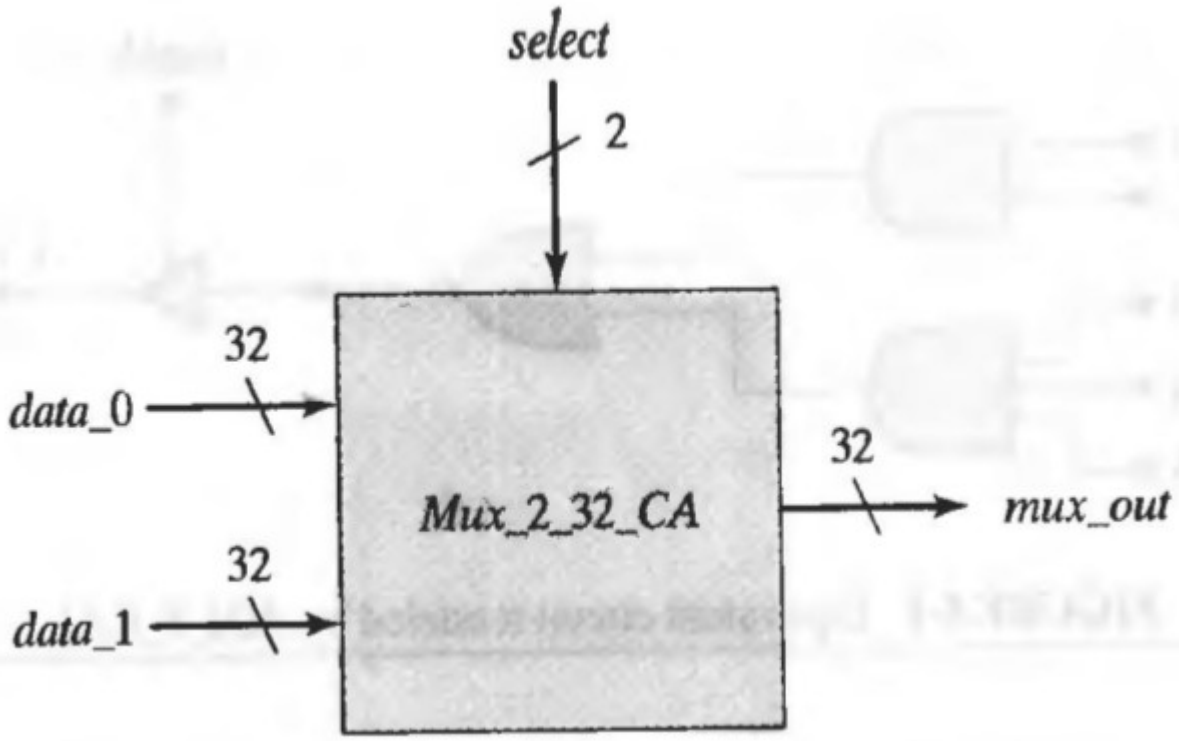
```
module AOI_5_CA0 (y_out, x_in1, x_in2, x_in3, x_in4, x_in5);
  input          x_in1, x_in2, x_in3, x_in4, x_in5;
  output         y_out;

  assign y_out =  ~((x_in1 & x_in2) | (x_in3 & x_in4 & x_in5));

endmodule
```

Modified AOI




```
module AOI_5_CA2 (y_out, x_in1, x_in2, x_in3, x_in4, x_in5, enable);
 input          x_in1, x_in2, x_in3, x_in4, x_in5, enable;
 output         y_out;

  wire y_out = enable ? ~((x_in1 & x_in2) | (x_in3 & x_in4 & x_in5)) : 1'bz;

endmodule
```

32-Bit Two-Channel MUX



32-Bit Two-Channel MUX

```
module Mux_2_ 32_CA ( mux_out, data_1, data_0, select);
 parameter                          word_size = 32;
 output          [word_size -1: 0]  mux_out;
 input           [word_size -1: 0]  data_1, data_0;
 input                              select;

 assign mux_out = select ? data_1 : data_0;
endmodule
```

Propagation Delay

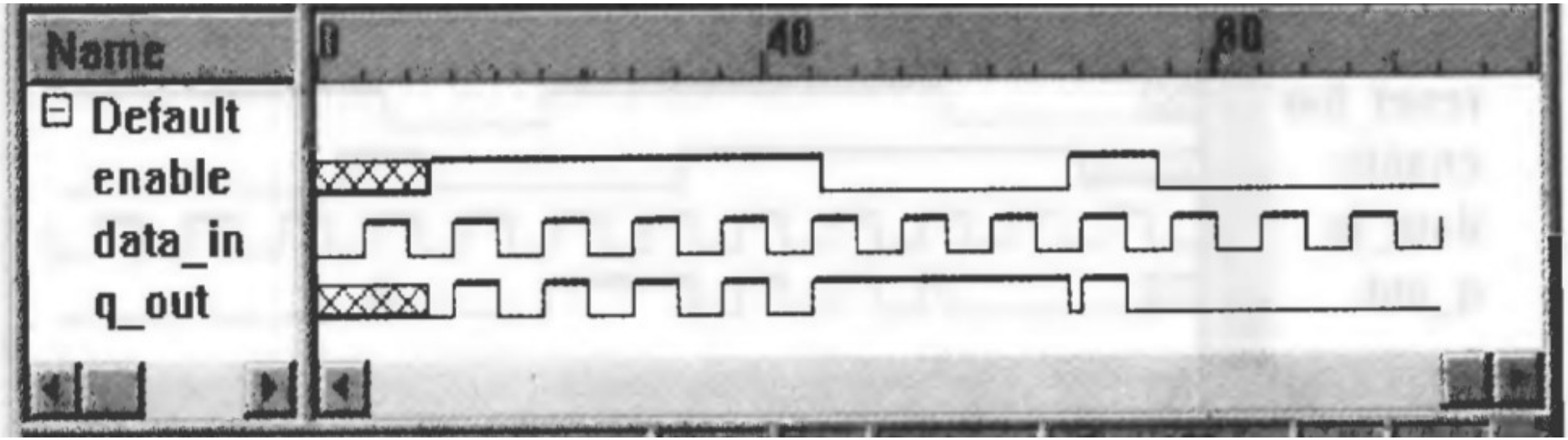
```
module AOI_5 _CA3 (y_out, x_in1, x_in2, x_in3, x_in4);
  input          x_in1, x_in2, x_in3, x_in4;
  output         y_out;

  wire #1 y1 = x_in1 & x_in2;          // Bitwise and operation
  wire #1 y2 = x_in3 & x_in_4;
  wire #1 y_out = ~ (y1 | y2);         // Complement the result of bitwise OR operation
endmodule
```

Continuous Assignments

```
module compare_2_CA0 (A_lt_B, A_gt_B, A_eq_B, A1, A0, B1, B0);
  input   A1, A0, B1, B0;
  output A_lt_B, A_gt_B, A_eq_B;

  assign A_lt_B = (~A1) & B1 | (~A1) & (~A0) & B0 | (~A0) & B1 & B0;

  assign A_gt_B = A1 & (~B1) | A0 & (~B1) & (~B0) | A1 & A0 & (~B0);

  assign A_eq_B = (~A1) & (~A0) & (~B1) & (~B0) | (~A1) & A0 & (~B1) & B0
          | A1 & A0 & B1 & B0 | A1 & (~A0) & B1 & (~B0);

endmodule
```

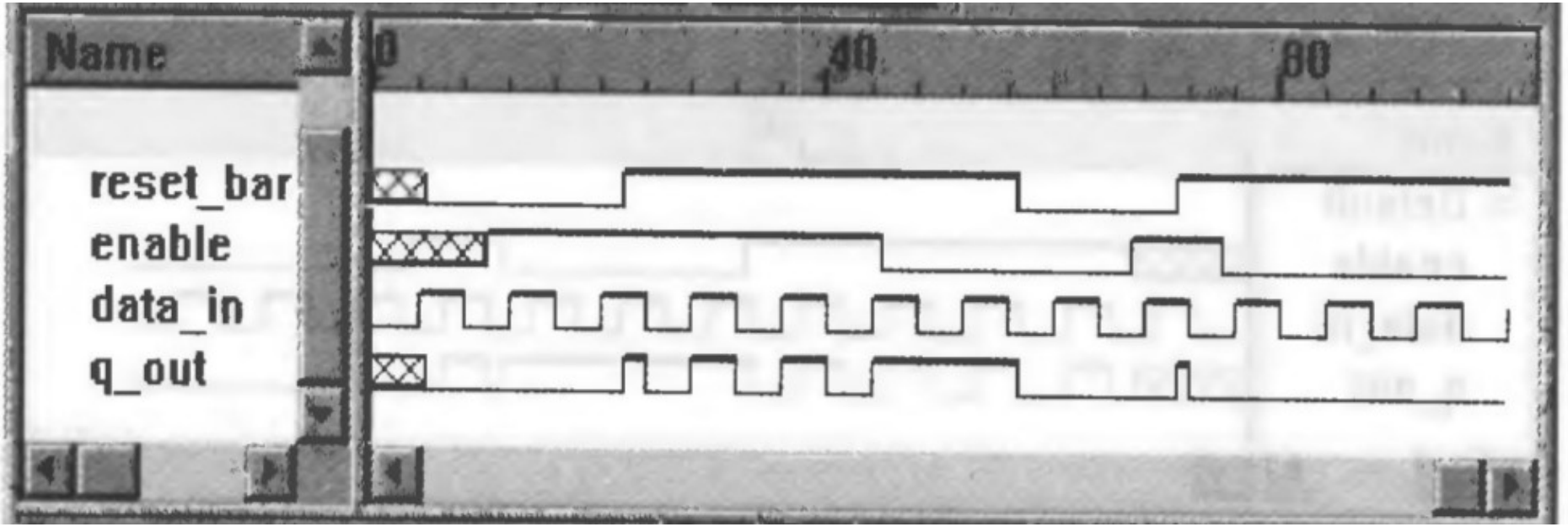
Latches and Level-Sensitive Circuits

```
assign q = set ~& qbar;
assign qbar = rst ~& q;

module Latch_CA (q_out, data_in, enable);
 output        q_out;
 input         data_in, enable;

 assign q_out = enable ? data_in : q_out;
endmodule
```



More Complex Latch

```
module Latch_Rbar_CA (q_out, data_in, enable, reset);
  output        q_out;
  input         data_in, enable, reset;

  assign q_out = !reset ? 0 : enable ? data_in : q_out;
endmodule
```



Cyclic Behavior of Flip-Flops

```
module df_behav (q, q_bar, data, set, reset, clk);
input          data, set, clk, reset;
 output        q, q_bar;
 reg           q;

 assign q_bar = ~ q;

 always @  (posedge clk)  // Flip-flop with synchronous set/reset

   begin
    if (reset == 0) q <= 0;
    else if (set ==0) q <= 1;
    else q <= data;
   end
endmodule
```

Even More Complex FFs

```
module asynch_df_behav (q, q_bar, data, set, clk, reset );
  input          data, set, reset, clk;
  output         q, q_bar;
  reg            q;

  assign         q_bar = ~q;

  always @ (negedge set or negedge reset or posedge clk)
   begin
    if (reset == 0) q <= 0;
      else if (set == 0) q <= 1;
        else q <= data;                  // synchronized activity
  end
endmodule
```

Latch vs.Flip-Flop

```
module tr_latch (q_out, enable, data);
  output q_out;
  input enable, data;
  reg q_out;
  always @  (enable or data)
    begin
      if (enable) q_out = data;
    end
endmodule
```

```
module df_behav (q, q_bar, data, set, reset, clk);
input          data, set, clk, reset;
 output        q, q_bar;
 reg           q;

 assign q_bar = ~ q;

 always @  (posedge clk)  // Flip-flop with synchronous set/reset

   begin
     if (reset == 0) q <= 0;
     else if (set ==0) q <= 1;
     else q <= data;
   end
endmodule
```

Final issues

- Please fill out the student info sheet before leaving
- Come by my office hours (right after class)
- Any questions or concerns?