# EEL 4783: HDL in Digital System Design

Lecture 4: Logic Design with Behavioral Models (cont.)

Prof. Mingjie Lin

# Latch vs.Flip-Flop

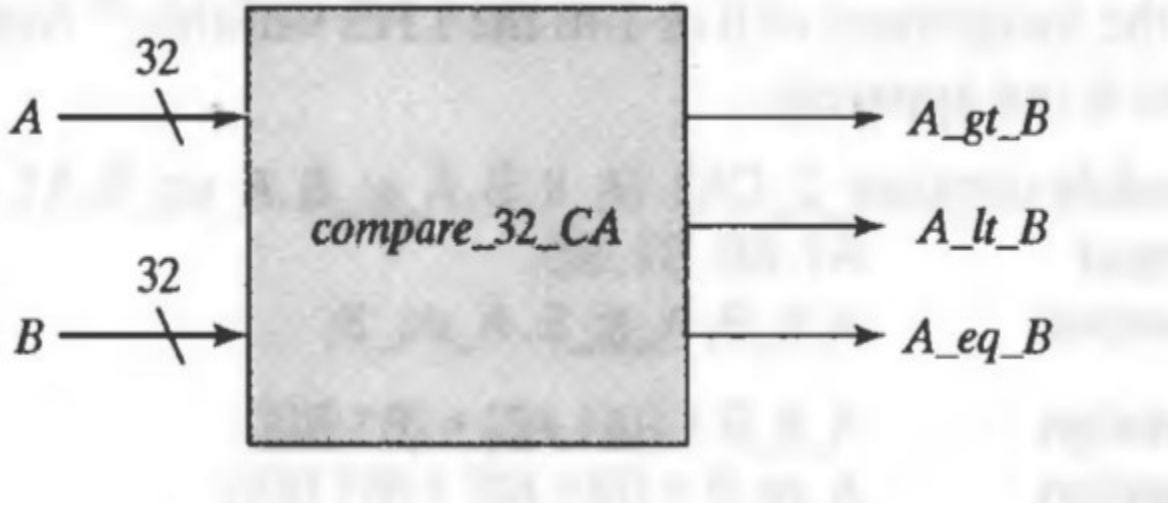
```
module tr_latch (q_out, enable, data);
  output q_out;
  input enable, data;
  reg q_out;
  always @  (enable or data)
    begin
      if (enable) q_out = data;
    end
endmodule
```

```
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
```

# Different Styles of Behavioral Modeling

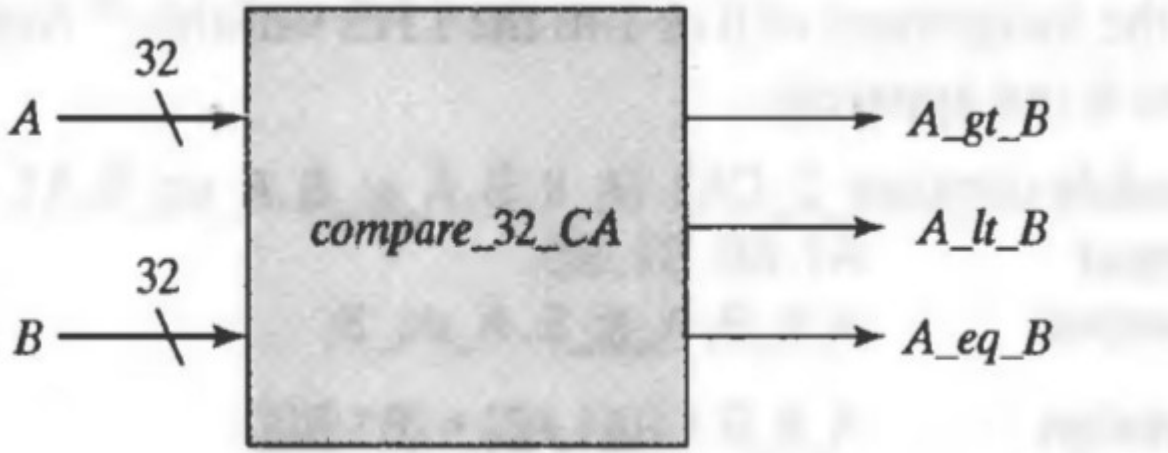
- Continuous Assignment Model
- Data-Flow/RTL Model
- Algorithm-Based Model

# Continuous Assignment Models



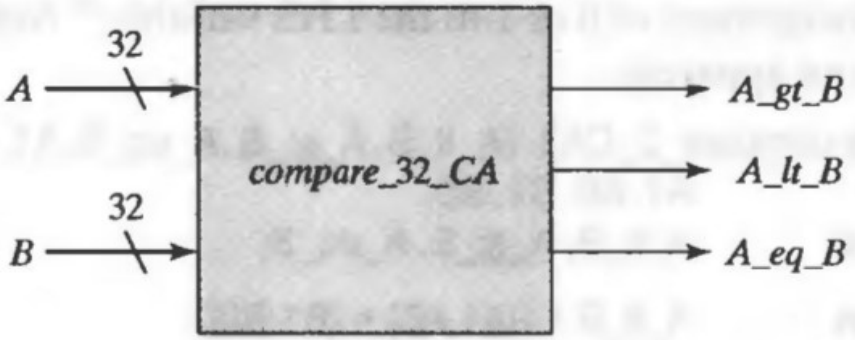

```
module compare_2_CA1  (A_lt_B, A_gt_B, A_eq_B, A1, A0, B1, B0);
 input          A1, A0, B1, B0;
 output         A_lt_B, A_gt_B, A_eq_B;

 assign         A_lt_B = ({A1,A0} < {B1,B0});
 assign         A_gt_B = ({A1,A0} > {B1,B0});
 assign         A_eq_B = ({A1,A0} == {B1,B0});
endmodule
```

# Continuous Assignment Models




```
module compare_32_CA (A_gt_B, A_lt_B, A_eq_B, A, B);
  parameter     word_size = 32;
  input         [word_size-1: 0]   A, B;
  output        A_gt_B, A_lt_B, A_eq_B;

  assign        A_gt_B = (A > B),          // Note: list of multiple assignments
                A_lt_B = (A < B),
                A_eq_B = (A == B);
endmodule
```

# Data-Flow/RTL Models

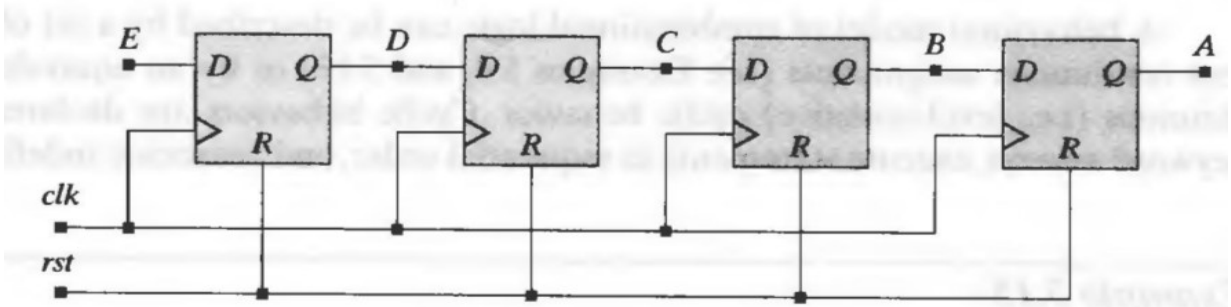



```
module compare_2_RTL (A_lt_B, A_gt_B, A_eq_B, A1, A0, B1, B0);
 Input          A1, A0, B1, B0;
 output         A_lt_B, A_gt_B, A_eq_B;
 reg            A_lt_B, A_gt_B, A_eq_B;

 always @ (A0 or A1 or B0 or B1) begin
  A_lt_B =      ({A1,A0} < {B1,B0});
  A_gt_B =      ({A1,A0} > {B1,B0});
  A_eq_B =      ({A1,A0} == {B1,B0});
 end
endmodule
```

# Blocking vs. Non-Blocking Statements

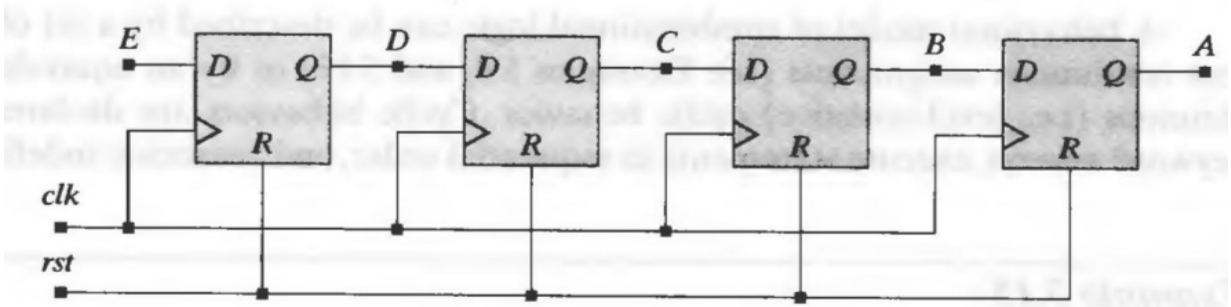
- A=B;
- A<=B;
- assign A = B;

# 4-Bit Shift Register




```
module shiftreg_PA (E, A, B, C, D, clk, rst);
 output A;
 input  E;
 input  clk, rst;
 reg    A, B, C, D;

 always @ (posedge clk or posedge rst) begin
   if (reset) begin A = 0; B = 0; C = 0; D = 0; end
   else begin
    A = B;
    B = C;
    C = D;
    D = E;
   end
 end
endmodule
```

# 4-Bit Shift Register



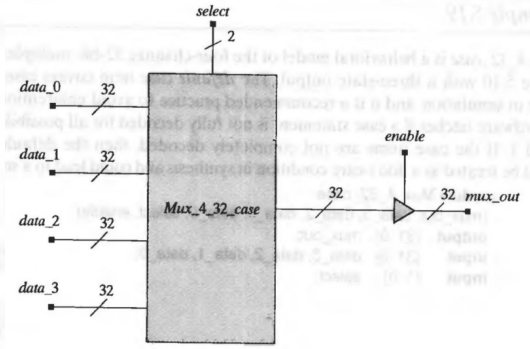

```
module shiftreg_nb (A, E, clk, rst);
  output A;
  input  E;
  input  clk, rst;
  reg    A, B, C, D;

  always @ (posedge clk or posedge rst) begin
    if (rst) begin A <= 0; B <= 0; C <= 0; D <= 0; end
    else begin
      A <= B;          //      D <= E;
      B <= C;          //      C <= D;
      C <= D;          //      B <= D;
      D <= E;          //      A <= B;
    end
  end
endmodule
```

# Algorithm-Based Model

```
module compare_2_algo (A_lt_B, A_gt_B, A_eq_B, A, B);
  output          A_lt_B, A_gt_B, A_eq_B;
  input [1: 0]    A, B;

  reg             A_lt_B, A_gt_B, A_eq_B;

  always @  (A or B)      // Level-sensitive behavior
    begin
      A_lt_B = 0;
      A_gt_B = 0;
      A_eq_B = 0;
      if (A == B)              A_eq_B = 1;      // Note: parentheses are required
      else if (A > B)          A_gt_B = 1;
      else                     A_lt_B = 1;
    end
endmodule
```

# Behavior Models of MUX

```
module Mux_4_32_case
  (mux_out, data_3, data_2, data_1, data_0, select, enable);
  output   [31: 0]   mux_out;
  input    [31: 0]   data_3, data_2, data_1, data_0;
  input    [1: 0]    select;
  input              enable;
  reg      [31: 0]   mux_int;

  assign mux_out = enable ? mux_int : 32'bz;

  always @ ( data_3 or data_2 or data_1 or data_0 or select)
    case (select)
      0:          mux_int = data_0;
      1:          mux_int = data_1;
      2:          mux_int = data_2;
      3:          mux_int = data_3;
      default:    mux_int = 32'bx;          // May execute in simulation
    endcase
endmodule
```



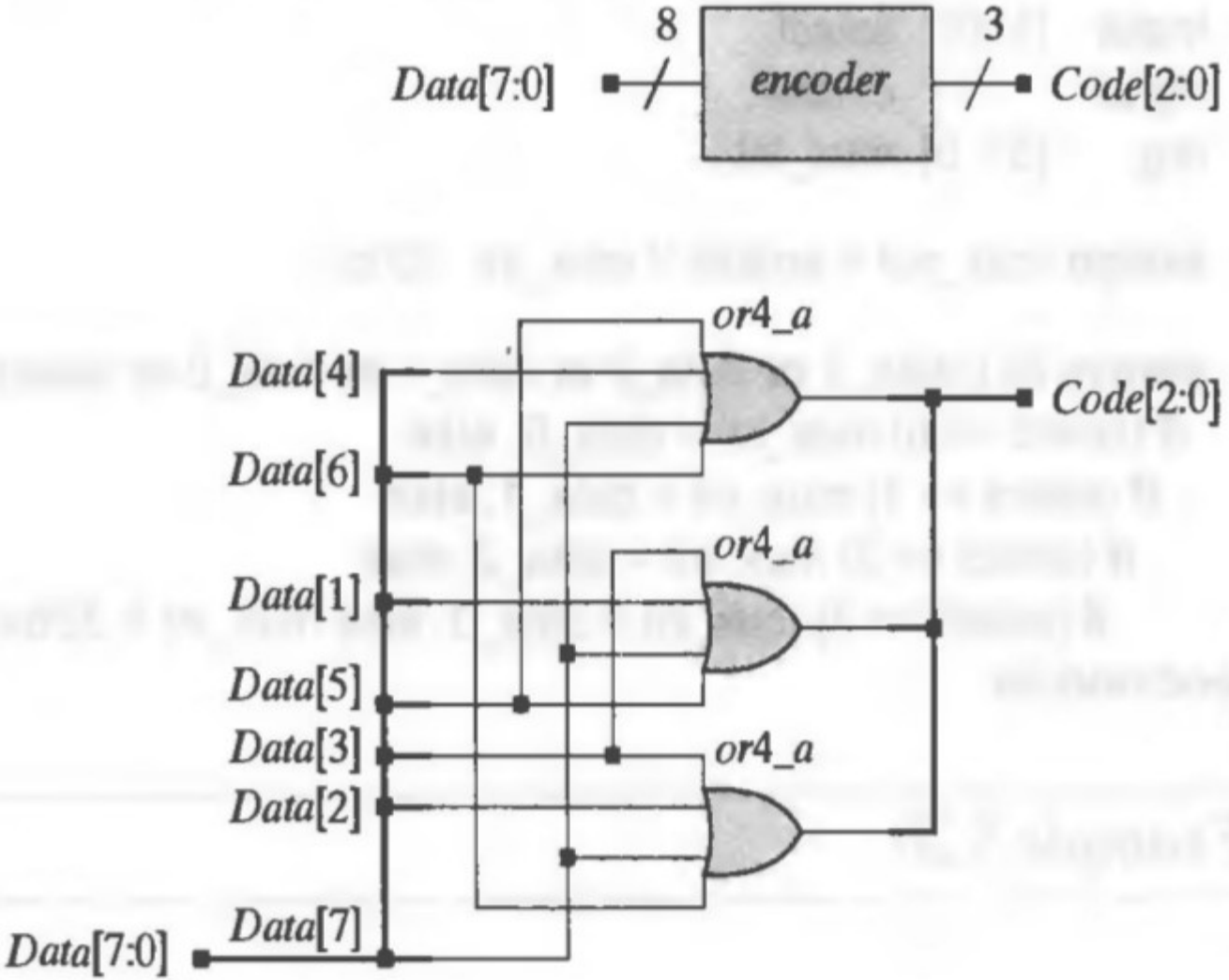
# Behavior Models of MUX (cont.)

```
module Mux_4_32_if
 (mux_out, data_3, data_2, data_1, data_0, select, enable);
 output [31: 0]  mux_out;
 input  [31: 0]  data_3, data_2, data_1, data_0;
 input  [1: 0]   select;
 input           enable;
 reg    [31: 0]  mux_int;

 assign mux_out = enable ? mux_int : 32'bz;

 always @ ( data_3 or data_2 or data_1 or data_0 or select)
  if (select == 0) mux_int = data_0; else
   if (select == 1) mux_int = data_1; else
    if (select == 2) mux_int = data_2; else
     if (select == 3) mux_int = data_3; else mux_int = 32'bx;
endmodule
```

# Behavior Models of MUX (cont.)

```
module Mux_4_32_CA (mux_out, data_3, data_2, data_1, data_0, select, enable);
  output [31: 0]  mux_out;
  input  [31: 0]  data_3, data_2, data_1, data_0;
  input  [1: 0]   select;
  input           enable;
  wire   [31: 0]  mux_int;
  assign mux_out = enable ? mux_int : 32'bz;
  assign mux_int = (select == 0)  ? data_0 :
                   (select == 1) ? data_1:
                   (select == 2) ? data_2:
                   (select == 3) ? data_3: 32'bx;
endmodule
```

# Encoder


8
3
Data[7:0]
encoder
Code[2:0]
or4_a
Data[4]
Data[6]
Code[2:0]
or4_a
Data[1]
Data[5]
Data[3]
or4_a
Data[2]
Data[7]
Data[7:0]

# Encoder

```
module encoder (Code, Data);
 output          [2: 0] Code;
 input           [7: 0] Data;
 reg             [2: 0] Code;

 always @  (Data)
  begin
    if (Data == 8'b00000001) Code = 0; else
    if (Data == 8'b00000010) Code = 1; else
    if (Data == 8'b00000100) Code = 2; else
    if (Data == 8'b00001000) Code = 3; else
    if (Data == 8'b00010000) Code = 4; else
    if (Data == 8'b00100000) Code = 5; else
    if (Data == 8'b01000000) Code = 6; else
    if (Data == 8'b10000000) Code = 7; else Code = 3'bx;
  end
```

# Encoder

```
always @  (Data)
 case (Data)
     8'b00000001  :  Code = 0;
     8'b00000010  :  Code = 1;
     8'b00000100  :  Code = 2;
     8'b00001000  :  Code = 3;
     8'b00010000  :  Code = 4;
     8'b00100000  :  Code = 5;
     8'b01000000  :  Code = 6;
     8'b10000000  :  Code = 7;
     default      :  Code = 3'bx;
 endcase
*/
endmodule
```

# Final issues

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