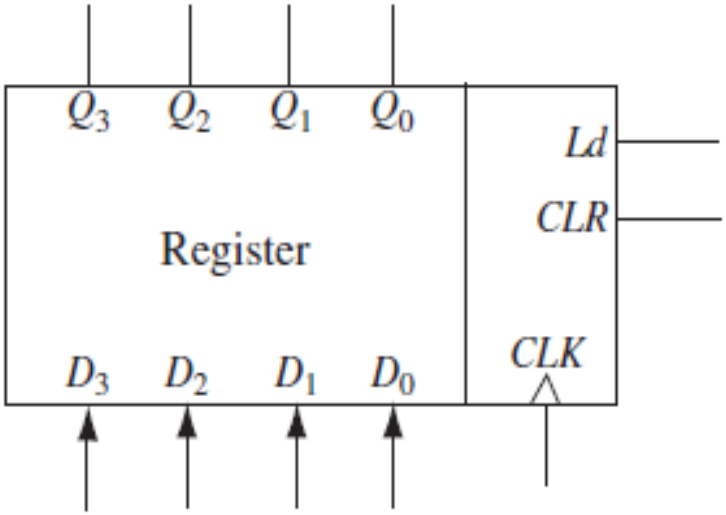
INTRODUCTION TO VERILOG

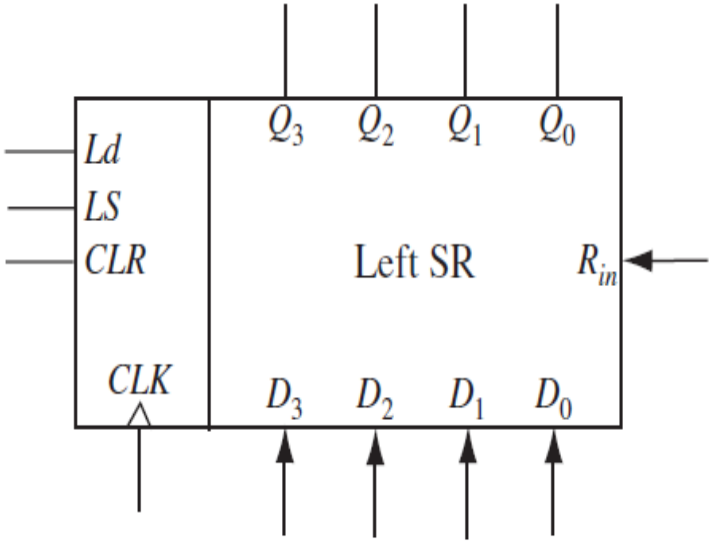
HDL models of sequential circuits




```
always @ (posedge CLK)
begin
 if (CLR)        Q <= 4'b0000;
 else if (Ld)    Q <= D;
end
```

Register with synchronous clear and load

HDL models of sequential circuits




```
always @ (posedge CLK)
begin
 if (CLR)        Q <= 4'b0000;
 else if (Ld)    Q <= D;
 else if (LS)    Q <= {Q[2:0], Rin};
end
```

Left shift register with synchronous clear and load

Q = 1101, Rin = 0

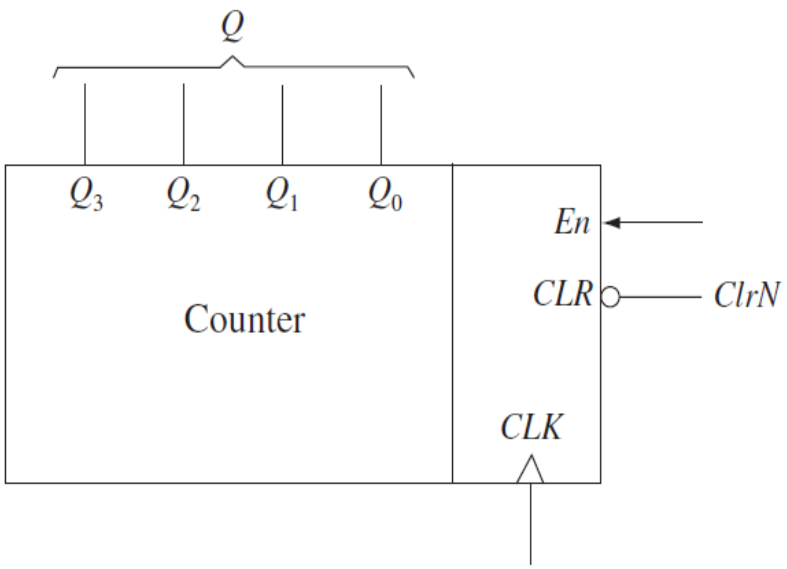
HDL models of sequential circuits

LS control input

LS = I, contents of the register shifted left and right most bit set equal to Rin.

Extract right most 3 bits of Q – Q[2:0], concatenate them with Rin. (1010)

HDL models of sequential circuits




```
reg Q[3:0];

always @ (posedge CLK)
begin
 if (~ClrN)     Q <= 4'b0000;
 else if (En)   Q <= Q + 1;
end
```

Synchronous counter

HDL models of sequential circuits

```
module counter(clk, CE, reset,
M, count);

//CE=Clock Enable, clk=clock,
M=Mode(up or down)

input clk, CE, reset, M;
output [3:0] count;

reg [3:0] count;

//reg [<upper>:0] <reg_name>;
```

```
always @(posedge clk)
        if (reset)
                count <= 0;
        else if (CE)
        if (M)
        count <= count + 1;
        else
        count <= count - 1;
endmodule
```

HDL models of sequential circuits

```
initial begin
// Initialize Inputs
                clk = 0;
                CE = 0;
                reset = 1;
                M = 1;
#100;
CE=1;
#100;
reset=0;
                #200;
                M=0;
                end

                always #10 clk=~clk;
                endmodule
```

HDL models of sequential circuits

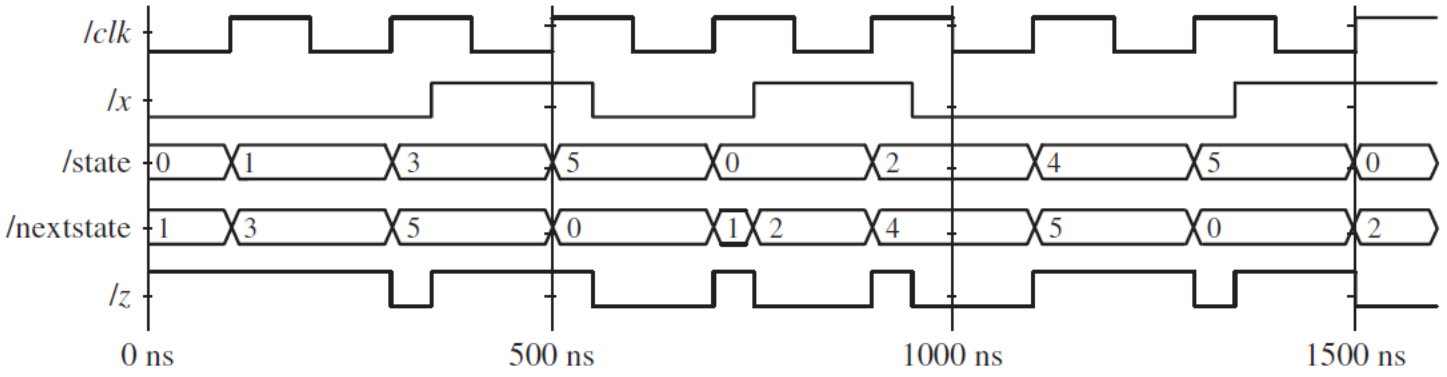
```
force CLK 0 0, 1 100 -repeat 200
force X 0 0, 1 350, 0 550, 1 750, 0 950, 1 1350
run 1600
```

- Clock with a period of 200 ns. CLK = 0 at 0ns, 1 at time 100 ns and repeats every 200 ns.

- Forcing x inputs. X = 0010 1001.

HDL models of sequential circuits

```
force CLK 0 0, 1 100 -repeat 200
force X 0 0, 1 350, 0 550, 1 750, 0 950, 1 1350
run 1600
```



HDL models of sequential circuits

- Several ways to model sequential machines.
- One approach two always blocks – represent the two parts of the circuit.
- One always block models the combinational part of the circuit – generates next state information and outputs.
- Other always block – models the state registers – update the state at the appropriate edge of the clock.
- Circuit output and next state can change – with state or input changes – sensitivity list includes both.

HDL models of sequential circuits

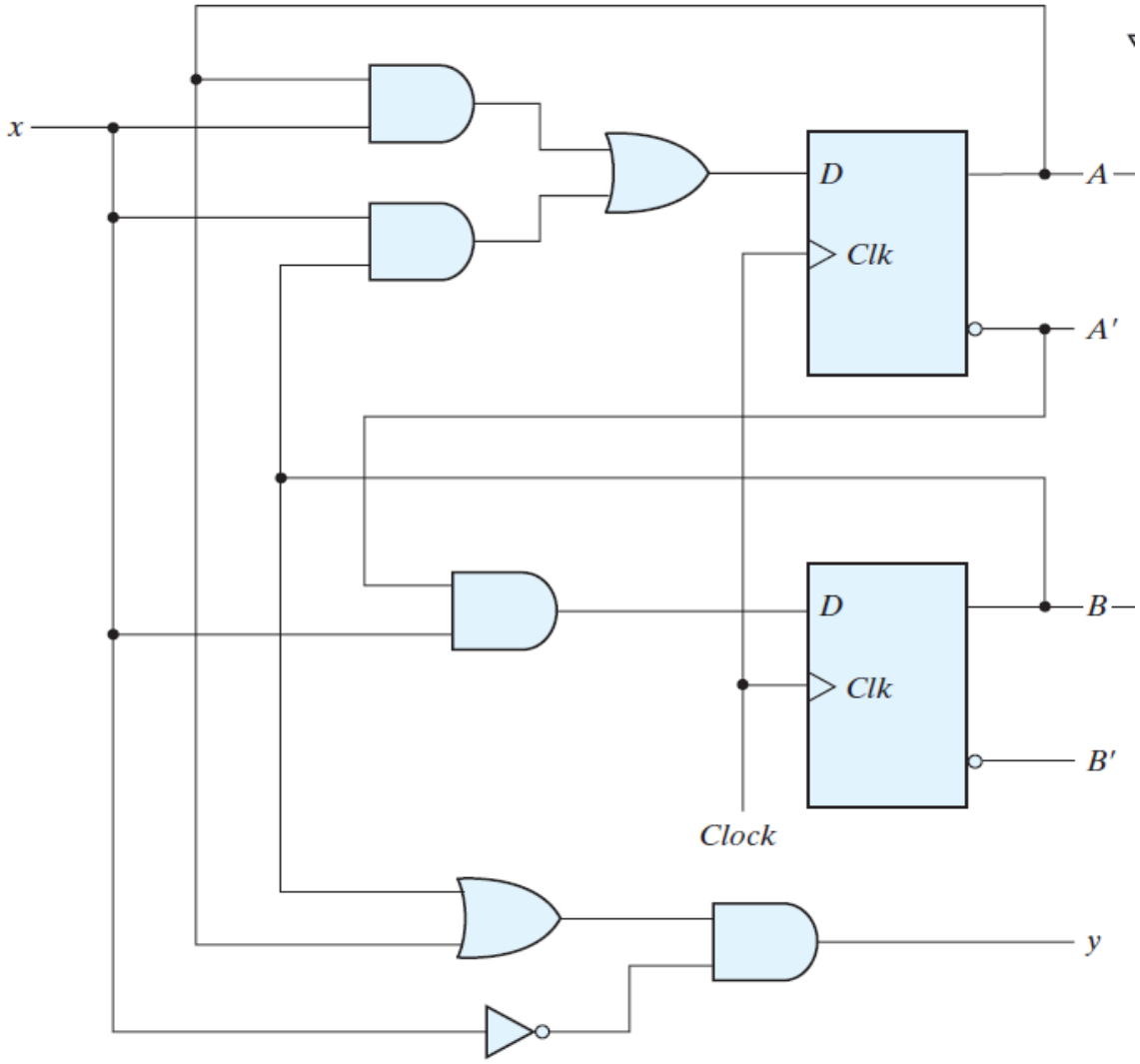
```
always @ (posedge CLK)          // State Register
begin                           // (synthesis)
  State <= Nextstate;           // rising edge of clock
end
```

- Manual design of the state machine – behavioral description – may not result in exactly same circuit.
- State assignment.

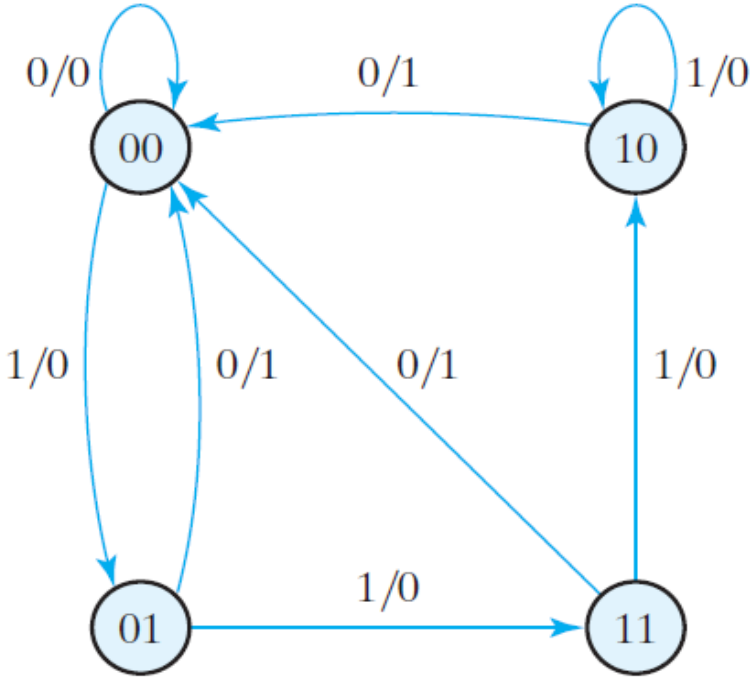
HDL models of sequential circuits

- Another approach to create Verilog module – structural description of flip-flops and gates.
- Designer manually performs the design – obtain gate level circuitry in order to create a model.
- Specification of components and interconnections by designers – synthesizer tool does not have to translate any structural description.
- More control over the generated circuitry with structural description – lot more effort to produce a structural model.

HDL models of sequential circuits



HDL models of sequential circuits



HDL models of sequential circuits

```
// Mealy FSM zero detector

module Mealy_Zero_Detector (output reg y_out, input
x_in, clock, reset);

reg [1: 0] state, next_state;
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;

always @ ( posedge clock, negedge reset)

if (reset == 0) state <= S0;
else state <= next_state;
```


// Veriilog allows constants to be defined in a module by the keyword parameter.

HDL models of sequential circuits

```
always @ (state, x_in)                    // Form the next state
case (state)
S0:     if (x_in) next_state = S1; else next_state = S0;
S1:     if (x_in) next_state = S3; else next_state = S0;
S2:     if (~x_in) next_state = S0; else next_state = S2;
S3:     if (x_in) next_state = S2; else next_state = S0;
endcase

always @ (state, x_in)                    // Form the Mealy output
case (state)
S0:     y_out = 0;
S1, S2, S3:    y_out = ~x_in;
endcase
endmodule
```

HDL models of sequential circuits

- First always statement – resets the circuit to the initial state S0 = 00 – specifies synchronous clocked operation.
- At every rising edge of the clock – reset not equal to 0, state of the machine updated by first always block.
- Change in state detected by the sensitivity list mechanism of second always block – updates the value of next state – will be used by the first always block in the next clock cycle.
- Third always block detects change in state – updates the value of the output.

HDL models of sequential circuits

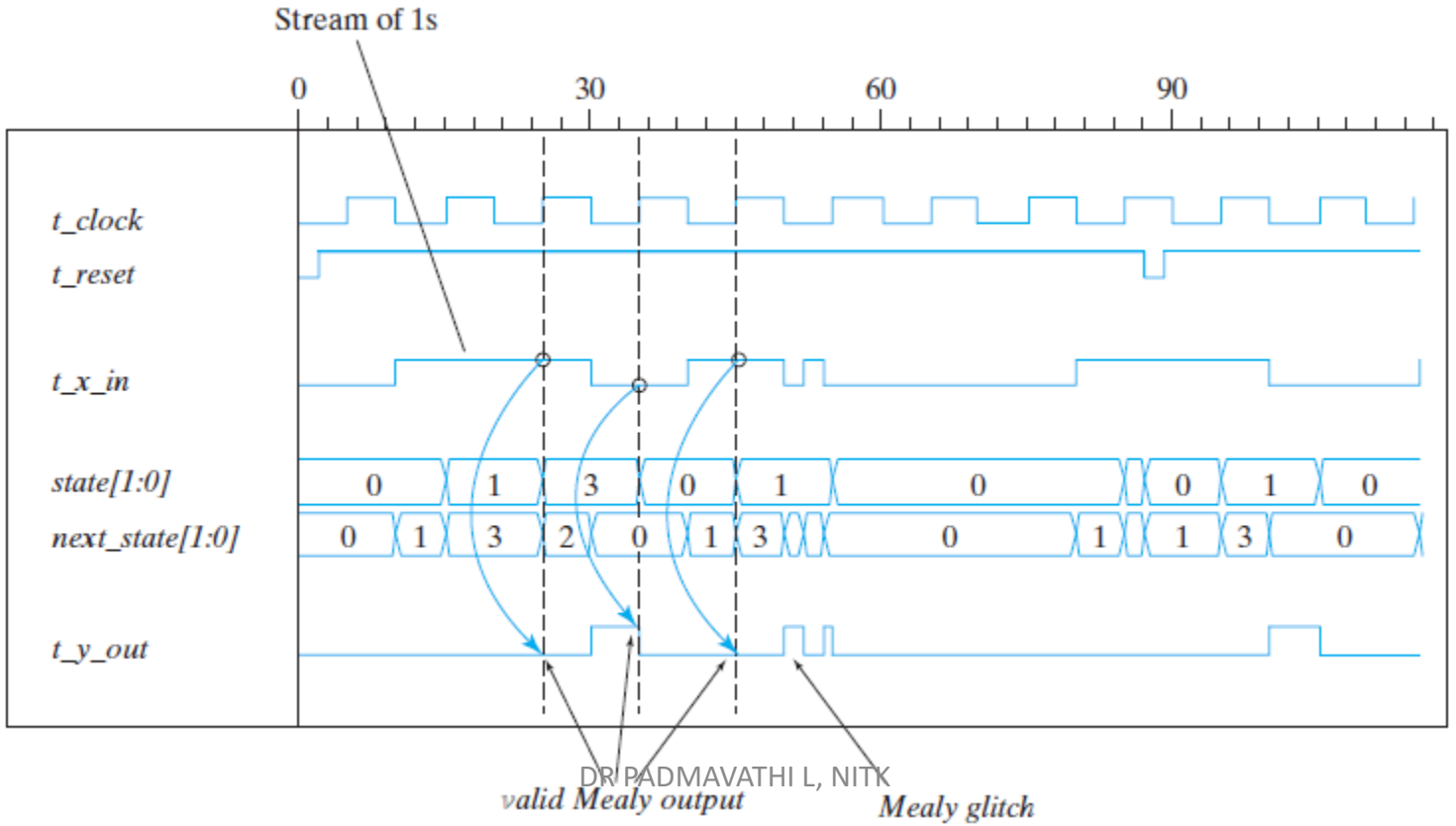
- Second and third always block responds to changes in x input – update the next state and y-out accordingly.

```
initial

begin

t_clock = 0;

forever #5 t_clock = ~t_clock;
end

initial

fork

t_reset = 0;
#2 t_reset = 1;
#87 t_reset = 0;
#89 t_reset = 1;

#10 t_x_in = 1;
#30 t_x_in = 0;
#40 t_x_in = 1;
#50 t_x_in = 0;
#52 t_x_in = 1;
#54 t_x_in = 0;
#70 t_x_in = 1;
#80 t_x_in = 1;
#70 t_x_in = 0;
#90 t_x_in = 1;
#100 t_x_in = 0;
#120 t_x_in = 1;
#160 t_x_in = 0;
#170 t_x_in = 1;

join
endmodule
```

HDL models of sequential circuits

- Notice how output responds to changes in both the state and the input.

- Look for transient logic and glitches



HDL models of sequential circuits

- fork…join construct
- Statements within the fork…loin block execute in parallel.
- Time delays are relative to a common reference of t = 0 – time at which the block begins execution.
- Reset can be triggered on the fly – machine recovers from an unexpected reset condition during any state.

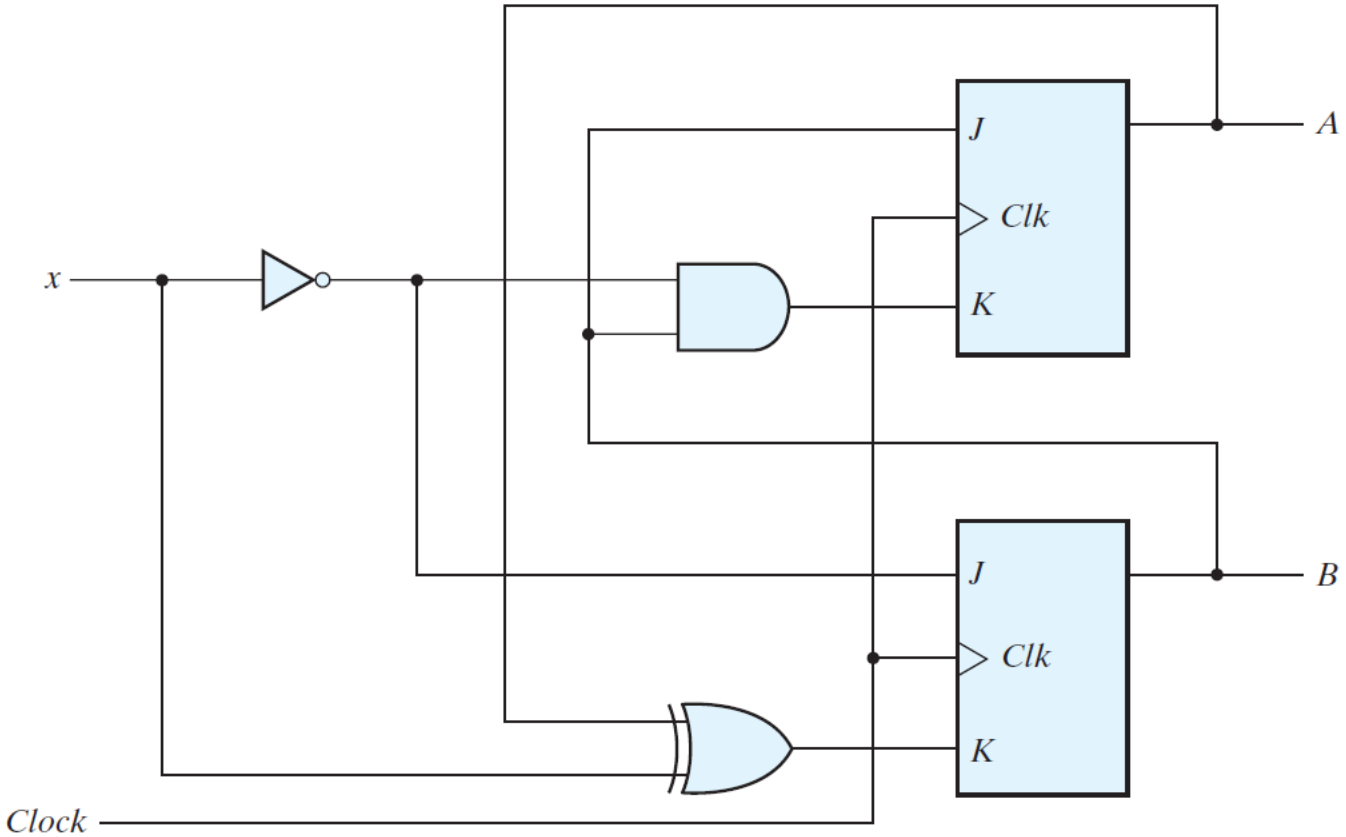
HDL models of sequential circuits

- First always block – corresponds to D flip-flop implementation of state registers.
- Second always block – combinational logic block describing the next state.
- Third always block – output combinational logic of the Mealy machine.
- Register operation of the state transition – uses non-blocking assignment operator <= – FFs of a sequential machine updated concurrently by a common clock.

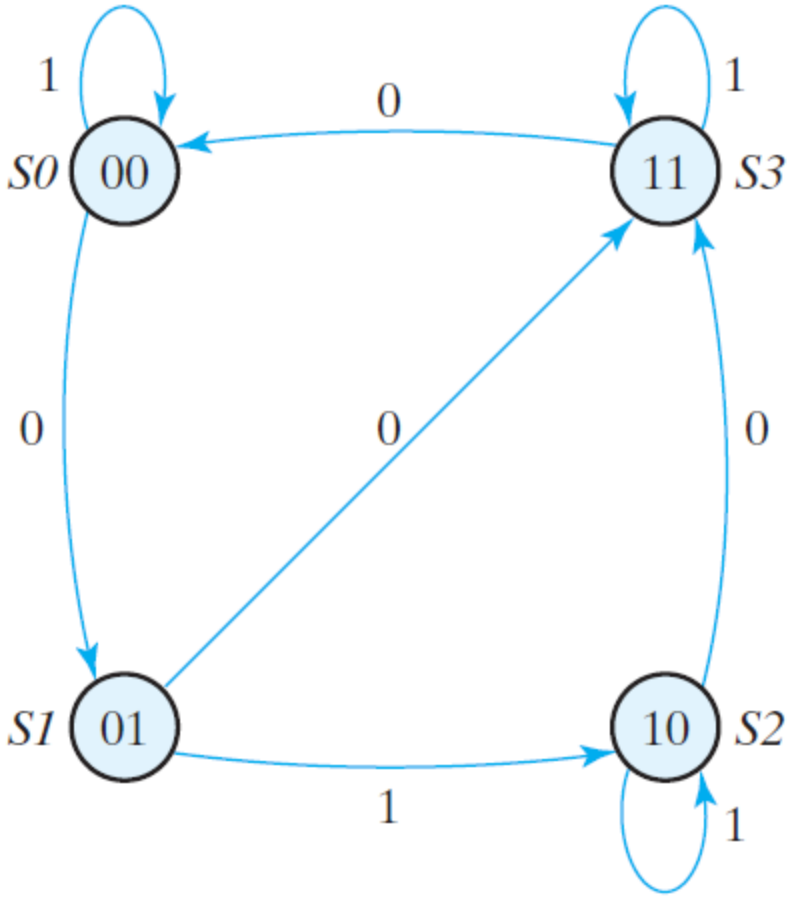
HDL models of sequential circuits

- Second and third always block uses blocking assignment operator. (=)
- Sensitivity list – includes both the state and the input – logic must respond to change in either or both of them.
- Commonly used Verilog model of Mealy machine – close relationship to the state diagram of the machine.
- Simpler and more readable description.

HDL models of sequential circuits



HDL models of sequential circuits



HDL models of sequential circuits

```
// Moore model FSM

module Moore_Model (output [1: 0] y_out, input x_in,
clock, reset);

reg [1: 0] state;

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

always @ ( posedge clock, negedge reset)
if (reset == 0) state <= S0; // Initialize to state S0

else case (state)
S0: if (~x_in) state <= S1; else state <= S0;
```

HDL models of sequential circuits

```
S1: if (x_in) state <= S2; else state <= S3;
S2: if (~x_in) state <= S3; else state <= S2;
S3: if (~x_in) state <= S0; else state <= S3;
endcase

assign y_out = state; // Output of fl ip-fl ops

endmodule
```

HDL models of sequential circuits

Alternate style – state transitions of the machine described by a single clocked cyclic behavior – one always block.

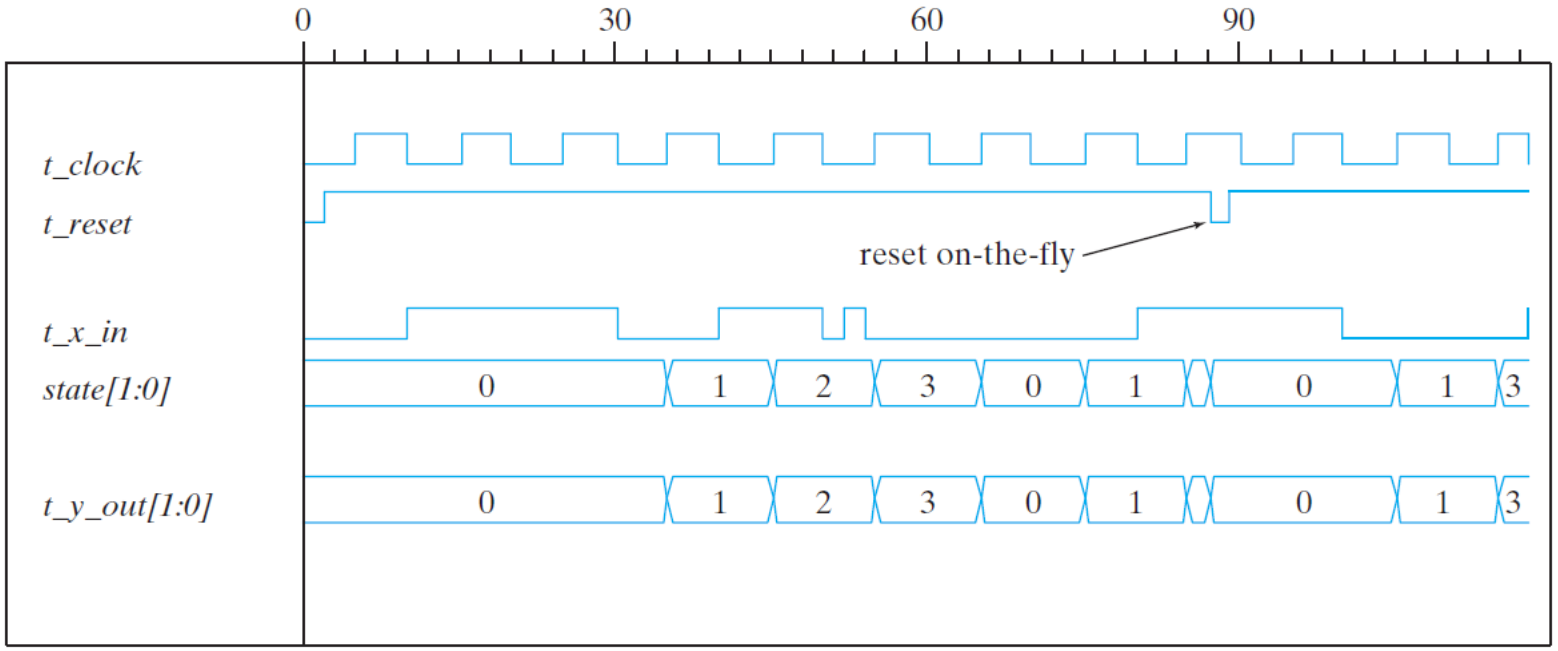
Present state – identified by the variable state.

State transitions triggered by the rising edge of the clock – according to the conditions listed in the case statement.

Moor machine – output is independent of the input.

2 bit y output specified as a continuous assignment.

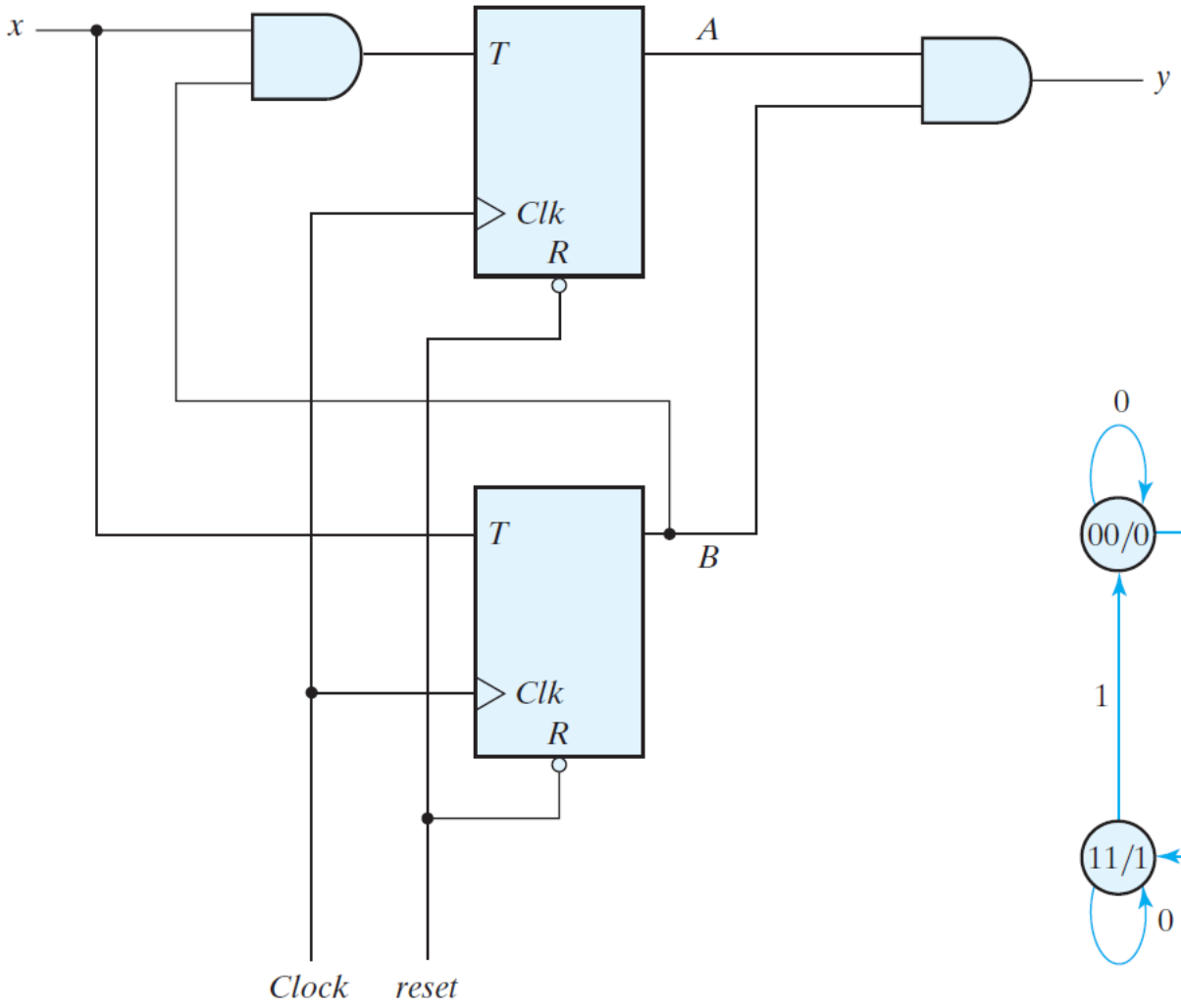
HDL models of sequential circuits



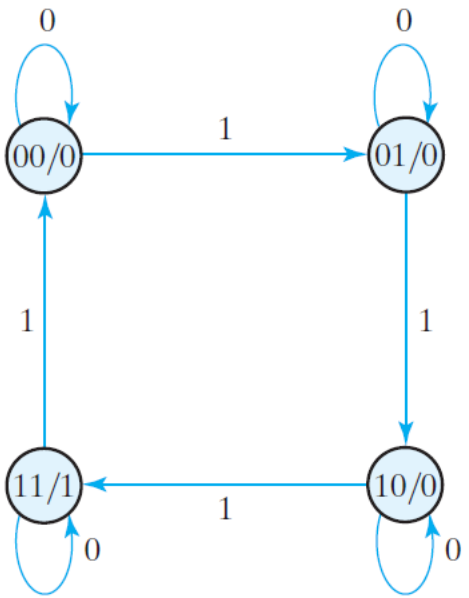
Output depends only on the present state of the input.

Reset on the fly – forces the state of the machine back to S0(00).

HDL models of sequential circuits



(a) Circuit diagram



(b) State diagram

HDL models of sequential circuits

```
// State-diagram-based model

module Moore_Model(output y_out, input x_in, clock,
reset);

reg [1: 0] state;

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

always @ ( posedge clock, negedge reset)
if (reset == 0) state <= S0;          // Initialize to state S0

else case (state)
S0: if (x_in) state <= S1; else state <= S0;
```

HDL models of sequential circuits

```
S1: if (x_in) state <= S2; else state <= S1;
S2: if (x_in) state <= S3; else state <= S2;
S3: if (x_in) state <= S0; else state <= S3;
endcase

assign y_out = state ;        // Output of fl ip-fl ops
endmodule
```

HDL models of sequential circuits

```
// Structural model

module Moore_Model_STR (output y_out, A, B, input x_in,
clock, reset);

wire TA, TB;

// Flip-fl op input equations

assign TA = x_in & B;
assign TB = x_in;
```

HDL models of sequential circuits

```
// Output equation
assign y_out = A & B;

// Instantiate Toggle flip-flops

Toggle_flip_flop_3 M_A (A, TA, clock, reset);
Toggle_flip_flop_3 M_B (B, TB, clock, reset);

endmodule
```

HDL models of sequential circuits

```
module Toggle_flip_flop (Q, T, CLK, RST_b);

output Q;
input T, CLK, RST_b;

reg Q;

always @ ( posedge CLK, negedge RST_b)

if (RST_b == 0) Q <= 1'b0;
else if (T) Q <= ~Q;

endmodule
```

HDL models of sequential circuits

- Continuous assignment statement and corresponding Boolean expressions.
- Instantiated T FFs use TA and TB – defined by the input equations.

HDL models of sequential circuits

```
initial #200 $finish ;

initial begin
t_reset = 0;
t_clock = 0;
#5 t_reset = 1;

repeat (16)
#5 t_clock = ~t_clock;
end
```

```
initial begin
t_x_in = 0;
#15 t_x_in = 1;

repeat (8)
#10 t_x_in = ~t_x_in;
end
```

HDL models of sequential circuits