Digital Design with the Verilog HDL

Chapter 6 FSM with Verilog

Binh Tran-Thanh

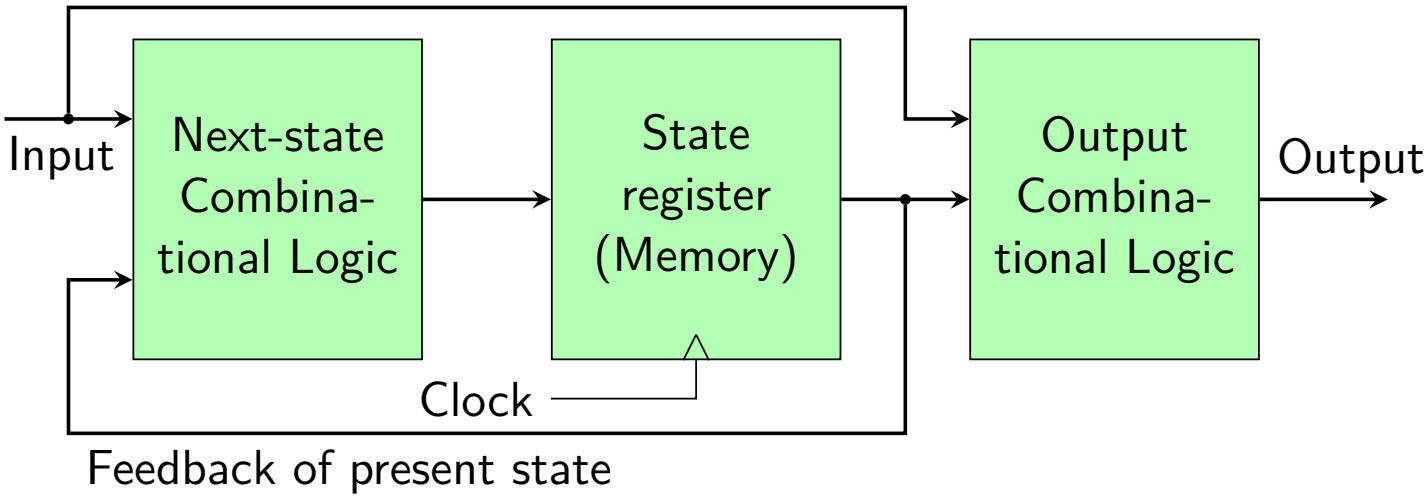
May 11, 2023

Explicit State Machines

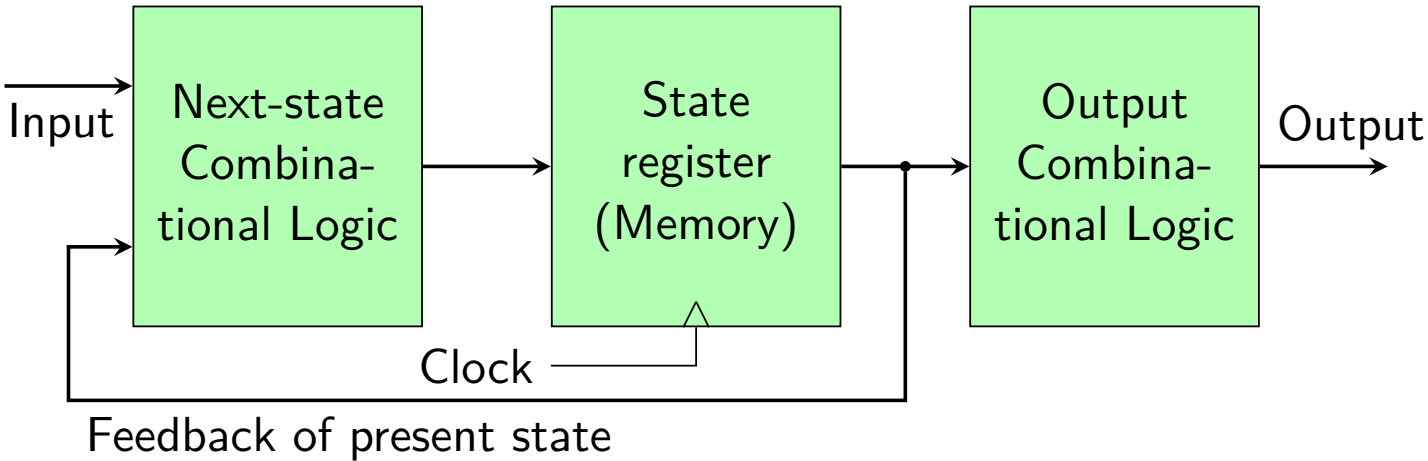
- Declare registers to store explicit states
- Combination logic circuit controls states
- Verilog:
  - Edge-trigger behaviour synchronizing the states
  - Level-trigger behaviour describing the next states and output logic

Mealy machine vs. Moore machine

Block Diagram of a Mealy sequential machine

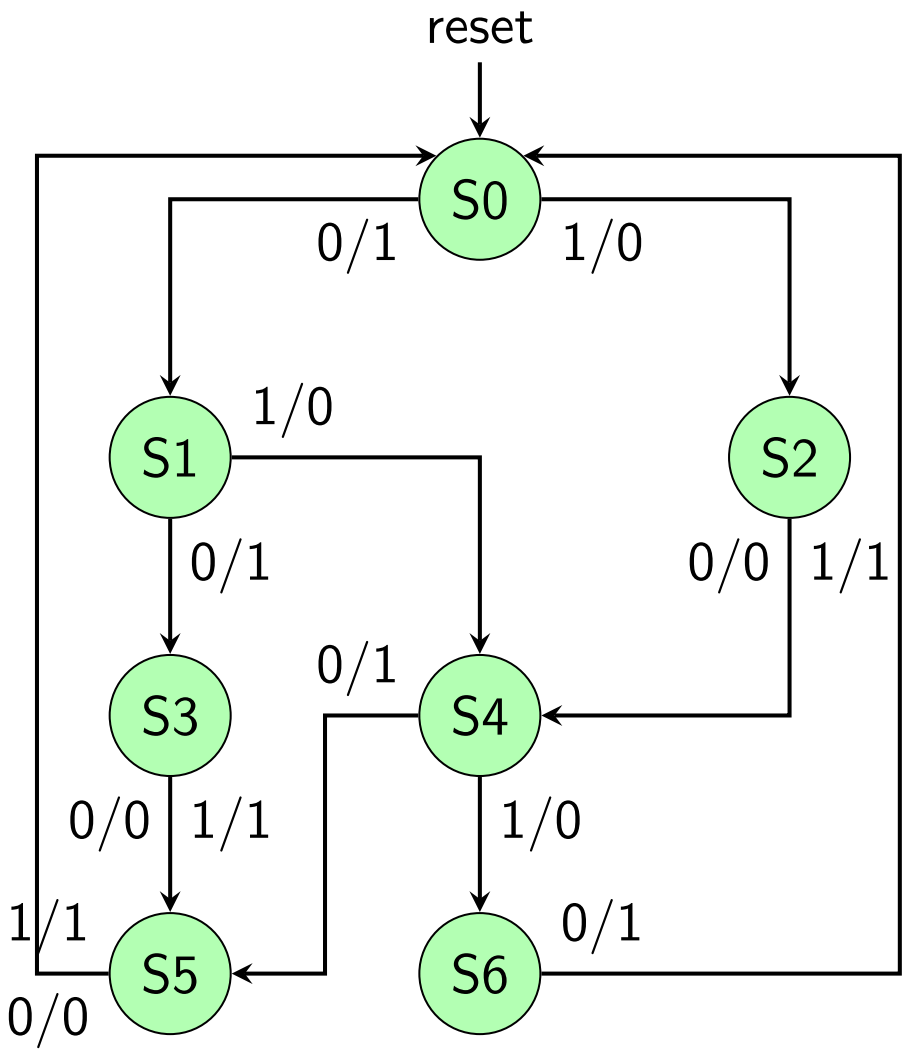


Block Diagram of a Moore sequential machine



BCD to Excess-3 Converter -FSM

State transition graph



State transition table

State	Next state/ output input	
	0	**1**
S0	S1/1	S2/0
S1	S3/1	S4/0
S2	S4/0	S4/1
S3	S5/0	S5/1
S4	S5/1	S6/0
S5	S0/0	S0/1
S6	S0/1	-/-

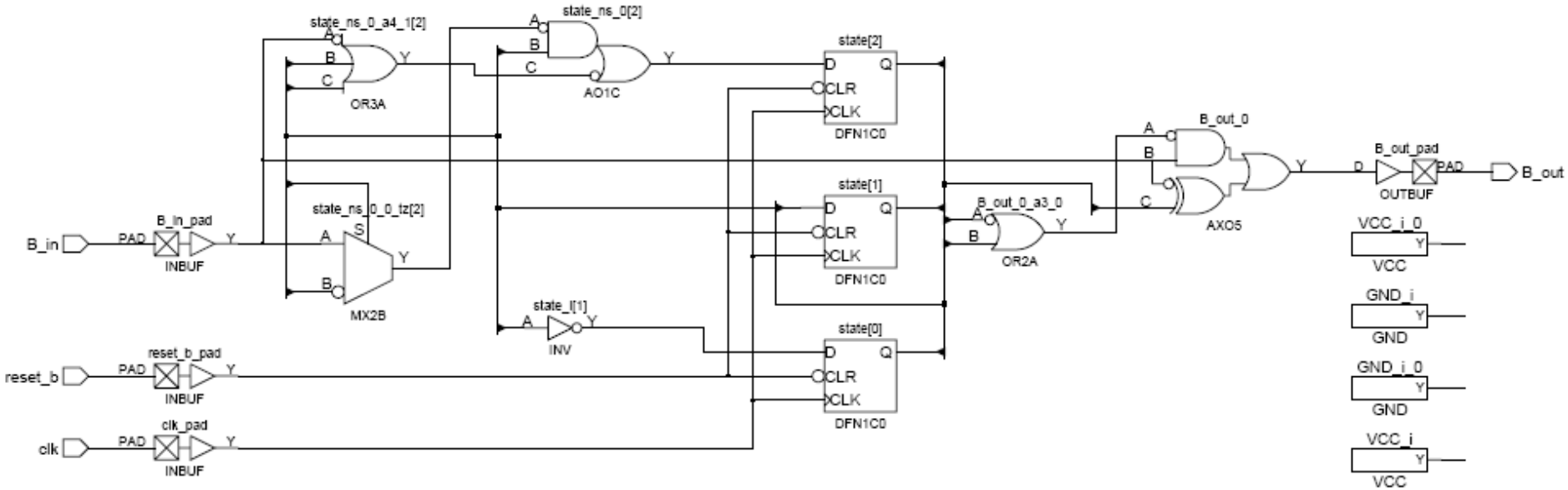
BCD to Excess-3 Converter -Verilog (1/2)

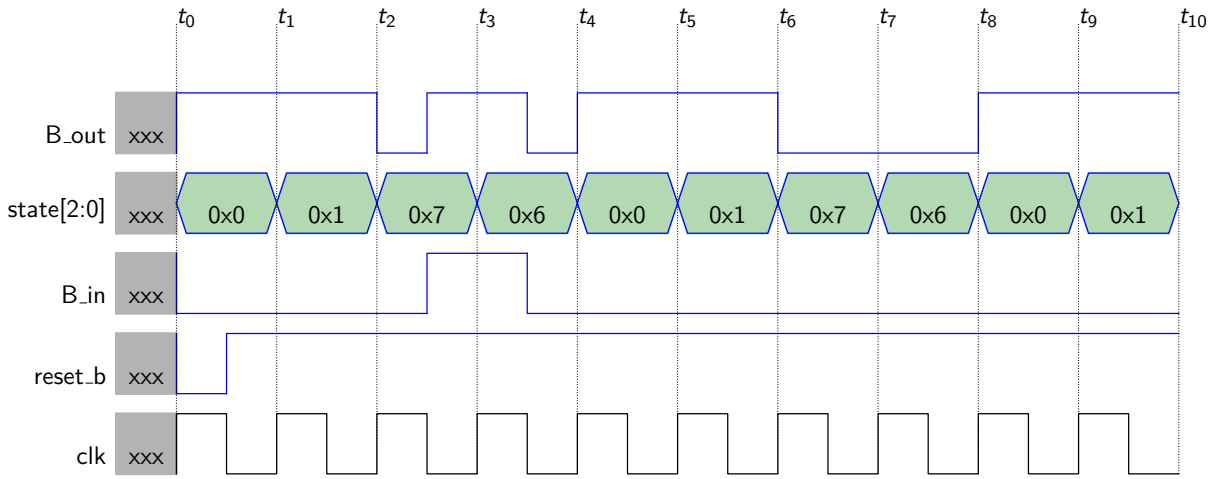
```
module BCD_to_Excess3(B_out, B_in, clk, reset);
  input B_in, clk, reset;
  output B_out;
  parameter S0 = 3'b000, //state encoding
            S1 = 3'b001,
            S2 = 3'b101,
            S3 = 3'b111,
            S4 = 3'b011,
            S5 = 3'b110,
            S6 = 3'b010,
            state_dont_care= 3'bx,
            out_dont_care= 1'bx;
  reg[2:0] state, next_state;
  reg B_out;
  //edge-trigger behaviour
  always @(posedge clk, negedge reset)
     if(reset == 1'b0) state <= S0;
     else state <= next_state;
```

BCD to Excess-3 Converter -Verilog (2/2)

```
always @(state, B_in) begin
  B_out= 0;
  case(state)
   S0: if(B_in== 1'b0) begin
          next_state= S1; B_out= 1'b1; end
       else if(B_in== 1'b1)
          next_state= S2;
   S1: if(B_in== 1'b0) begin
          next_state= S3; B_out= 1'b1; end
       else if(B_in== 1'b1)
          next_state= S4;
   S2: ...
   ...
   S6: ...
   default: next_state= state_dont_care;
 endcase
end
```

Synthesized Circuit





Sequence Recognizer: Mealy

```
module Seq_Rec_3_1s_Mealy (output Dout, input Din, En, clk,
    reset);
  parameter Sidle = 0, S0 = 1, S1 = 2, S2 = 3; // Binary
    code
  reg[1: 0] state, next_state;

  always @(negedge clk)
    if(reset == 1) state <= Sidle;
    else state <= next_state;

  always @(state, Din, En) begin
    case(state)
      Sidle:if ((En == 1) && (Din == 1)) next_state = S1;
        else if((En == 1) && (Din == 0)) next_state = S0;
        else next_state = Sidle;
      S0:if(Din == 0) next_state = S0;
        else if(Din == 1) next_state = S1;
        else next_state = Sidle;
      S1:if(Din == 0) next_state = S0;
        else if(Din == 1) next_state = S2;
        else next_state = Sidle;
      S2:if(Din == 0) next_state = S0;
        else if(Din == 1) next_state = S2;
        else next_state = Sidle;
      default: next_state = Sidle;
    endcase
  end
  assign Dout = ((state == S2) && (Din == 1 ));// Mealy
    output
endmodule
```

Sequence Recognizer: Moore

```
module Seq_Rec_3_1s_Moore (output Dout,
                           input Din, En, clk, reset);

  parameter Sidle =0, S0 = 1, S1 = 2, S2 = 3, S3 = 4;
  reg[2: 0] state, next_state;

  always @(negedge clk)
    if(reset == 1) state <= Sidle;
    else state <= next_state;

  always @(state or Din) begin next_state = Sidle;
    case(state)
      Sidle:if((En == 1) && (Din == 1)) next_state = S1;
        else if((En == 1) && (Din == 0)) next_state = S0;
        // else next_state = Sidle; // Remove!
      S0:if(Din == 0) next_state = S0;
        else if(Din == 1) next_state = S1;
        // else next_state = Sidle;
      S1:if(Din == 0) next_state = S0;
        else if(Din == 1) next_state = S2;
        // else next_state = Sidle;
      S2, S3:if(Din == 0) next_state = S0;
        else if(Din == 1) next_state = S3;
        // else next_state = Sidle;
      default: next_state = Sidle; // Why not 3'bx?
    endcase
  end
  assign Dout = (state == S3);// Moore output
endmodule
```