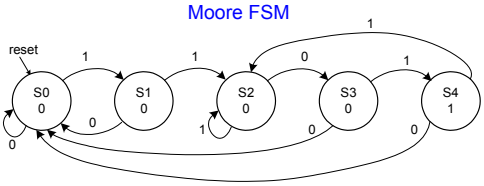
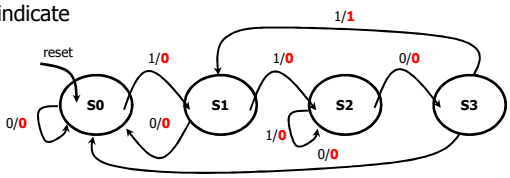
Moore vs. Mealy FSM

- Alyssa P. Hacker has a snail that crawls down a paper tape with 1's and 0's on it. The snail smiles whenever the last four digits it has crawled over are 1101. Design Moore and Mealy FSMs of the snail's brain.

Moore FSM



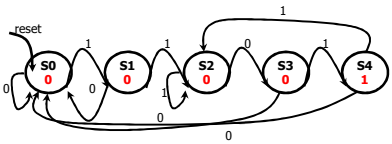
Mealy FSM: arcs indicate input/output



ARM

Moore FSM in Verilog

Moore FSM: arcs indicate input



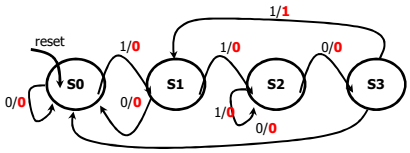

```
module moore_snail(input  clk, reset,
bnum,
                   output reg smile);

  reg [2:0] state, nextstate;

  parameter S0 = 3'b000;
  parameter S1 = 3'b001;
  parameter S2 = 3'b010;
  parameter S3 = 3'b011;
  parameter S4 = 3'b100;

  parameter delay = 1;

  // state register
  always @(posedge reset, posedge clk)
  begin
     if (reset) #delay state <= S0;
     else       #delay state <=
nextstate;
  end
```

```
// Next State Logic
always @(*)
begin
   case (state)
     S0: if (bnum) #delay nextstate <= S1;
         else      #delay nextstate <= S0;

     S1: if (bnum) #delay nextstate <= S2;
         else      #delay nextstate <= S0;

     S2: if (bnum) #delay nextstate <= S2;
         else      #delay nextstate <= S3;

     S3: if (bnum) #delay nextstate <= S4;
         else      #delay nextstate <= S0;

     S4: if (bnum) #delay nextstate <= S2;
         else      #delay nextstate <= S0;

       default:    #delay nextstate <= S0;
       endcase
  end

// Output Logic
  always @(*)
  begin
       if (state == S4) smile <= 1'b1 ;
       else             smile <=
1'b0 ;
  end

endmodule
```


ARM

Mealy FSM in Verilog

Mealy FSM: arcs indicate input/output




```
module mealy_snail(input       clk,
reset, bnum,
                   output reg  smile);

  reg [1:0] state, nextstate;

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

  parameter delay = 1;

 // state register
  always @(posedge reset, posedge clk)
  begin
        if (reset) #delay state <= S0;
        else       #delay state <=
nextstate;
  end
```

```
 // Next State and Output Logic
always @(*)
begin
 case (state)
  S0: begin
        #delay smile <= 1'b0;
        if (bnum) #delay nextstate <= S1;
        else      #delay nextstate <= S0;
      end

  S1: begin
        #delay smile <= 1'b0;
        if (bnum) #delay nextstate <= S2;
        else      #delay nextstate <= S0;
      end

  S2: begin
        #delay smile <= 1'b0;
        if (bnum) #delay nextstate <= S2;
        else      #delay nextstate <= S3;
      end

  S3: begin
         if (bnum) #delay smile <= 1'b1;
         else      #delay smile <= 1'b0;
         if (bnum) #delay nextstate <= S1;
         else      #delay nextstate <= S0;
       end

  default: begin
             #delay smile <= 1'b0;
             #delay nextstate <= S0;
             end
 endcase
 end

endmodule
```


ARM

Testbench for Snail FSM

```
`timescale 1ns/1ps

module fsm_snail_tb( );
   reg clk, reset, bnum;
   wire smile_moore;
   wire smile_mealy;

   parameter clk_period = 10;

   moore_snail   moore_snail_uut
        (clk, reset, bnum,
smile_moore);

   mealy_snail   mealy_snail_uut
        (clk, reset, bnum,
smile_mealy);

   initial
   begin
        reset = 1;
        #13
        reset = 0;
   end

   always
   begin
         clk = 1;
         forever #(clk_period/2) clk =
~clk;
   end
```

```
       initial
       begin
            bnum = 0;    #3;
            bnum = 0;    #clk_period;
            bnum = 1;    #clk_period;
            bnum = 0;    #clk_period;
            bnum = 0;    #clk_period;
            bnum = 0;    #clk_period;
            bnum = 1;    #clk_period;
            bnum = 1;    #clk_period;
            bnum = 0;    #clk_period;
            bnum = 1;    #clk_period;  // Smile
            bnum = 1;    #clk_period;
            bnum = 0;    #clk_period;
            bnum = 1;    #clk_period;  // Smile
            bnum = 0;
       end

endmodule
```


ARM