Embedded Systems

Week 8: State Machines

Dr. Vecdi Emre Levent

Instructors

Assist. Prof. Dr. Vecdi Emre Levent

Email : emre@levent.tc

emre.levent@marmara.edu.tr

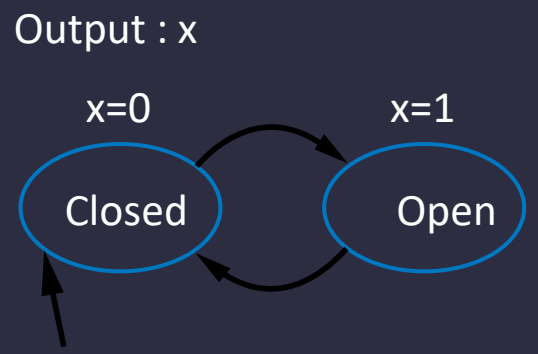
Web: www.levent.tc

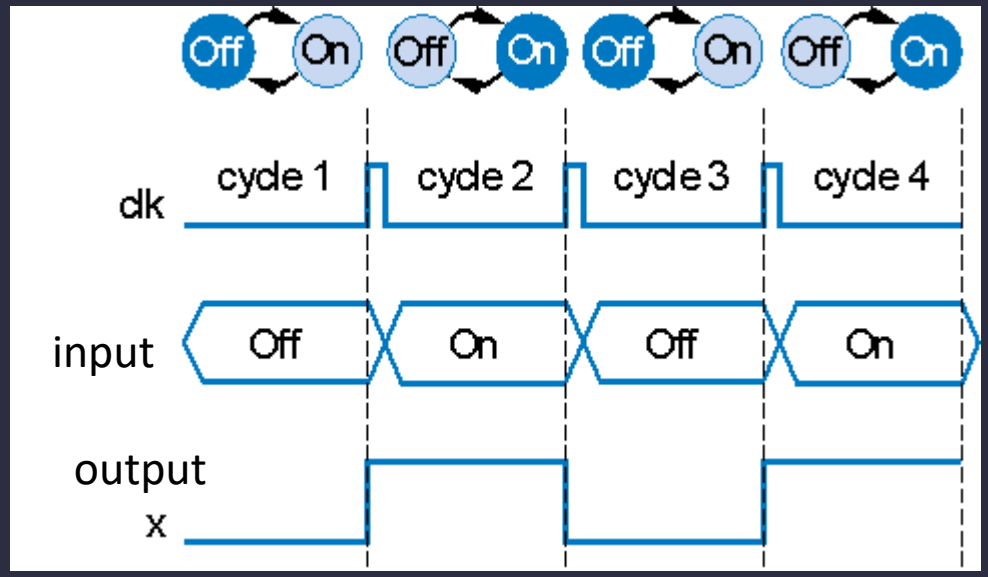
Course

- Finite State Machines

Determining the Behavior of the Sequential Circuit FSM

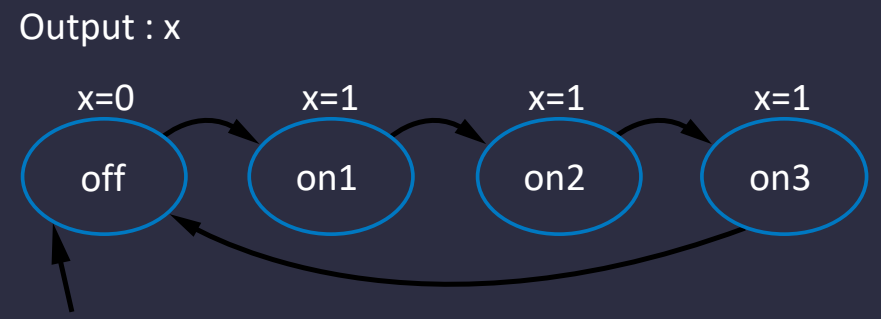
- Finite-State Machine (FSM)
  - It is a representation of the behavior of the sequential circuit according to the states.
  - Lists states and transitions between states
    - Example : Let there be an output signal named X and each clock change value in cycle
    - Two states : “Off” (x=0), and “On” (x=1)
    - There are transitions from the Off state to the On state and from the On state to the Off state.
    - The initial state is indicated by an arrow.

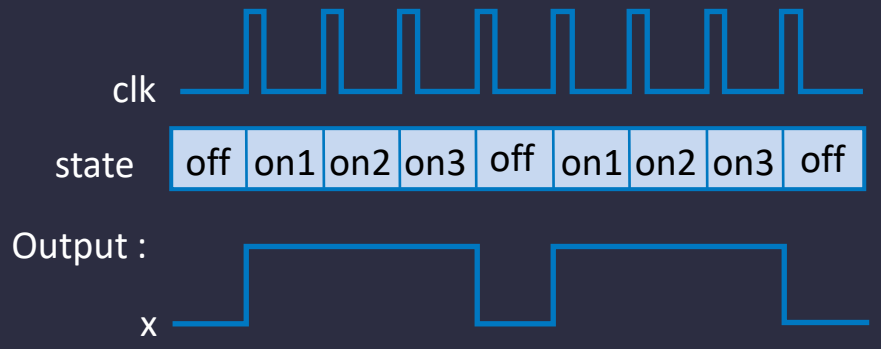




FSM Example : 0,1,1,1, repetition

- In order 0, 1, 1, 1, 0, 1, 1, 1, ... A circuit that gives its outputs will be designed.
  - Each value is a clock It appears in the cycle .
- Can be designed with FSM
  - There are 4 states

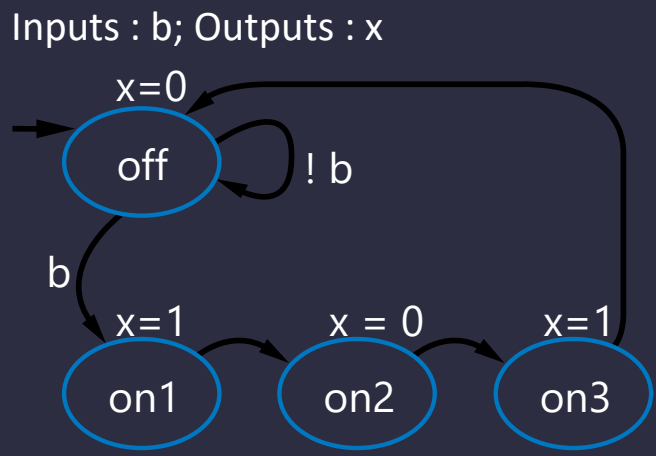


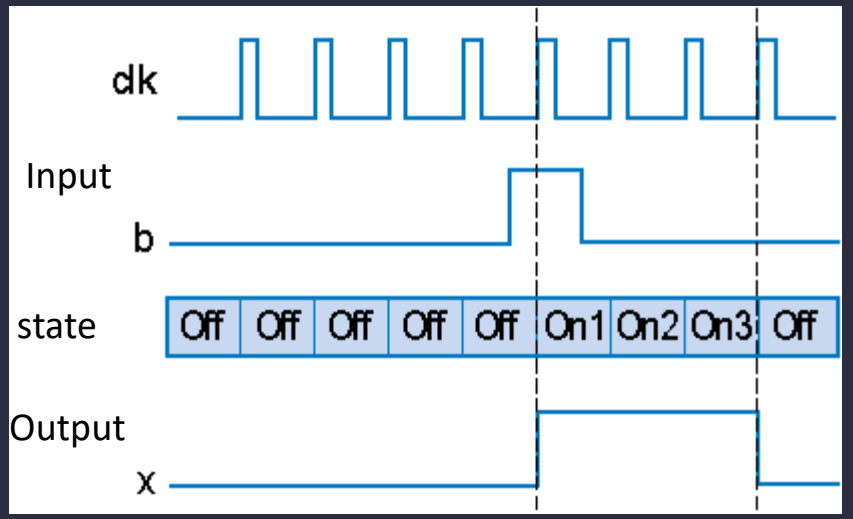


FSM Example

take a one-bit input named b,

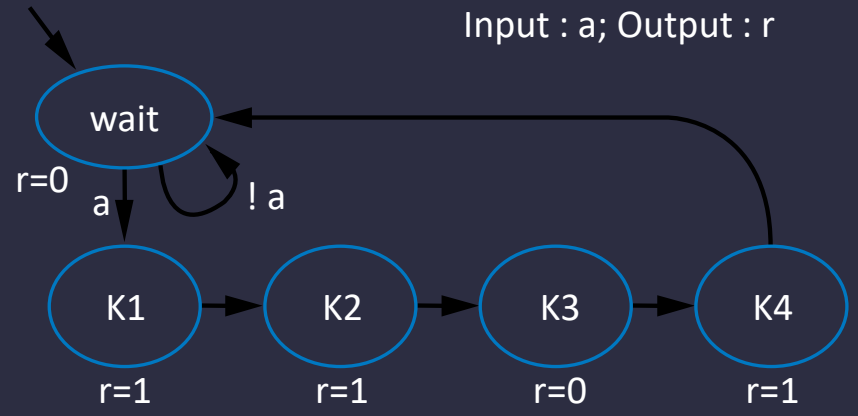
Output pattern 101 and waits for input b again.





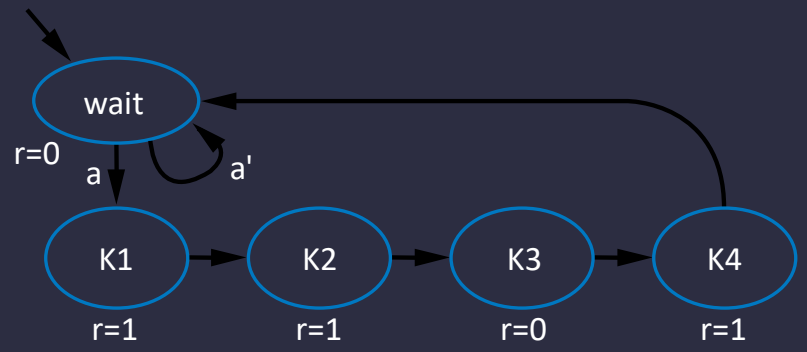
FSM Example: Car Key

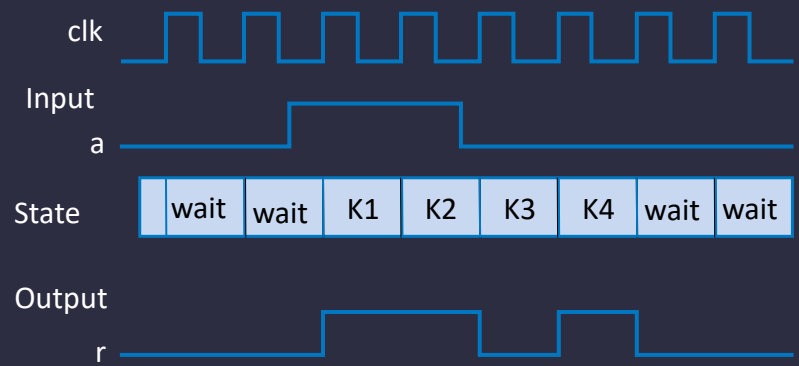
- Car keys contain a chip that holds the key's identity.
  - When the key is inserted in the car, it asks the key for identification.
  - Sends key credential, if not correct the tool won't start
- FSM
  - Wait for request (a=1)
  - Send ID (For example , 1101)

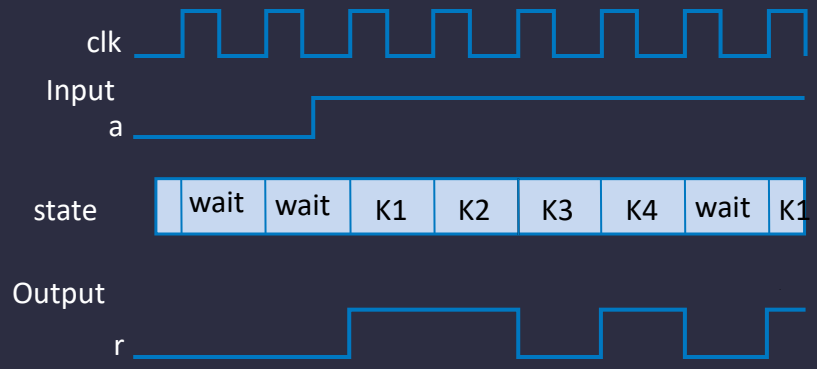


FSM Example: Car Key

- FSM Timings

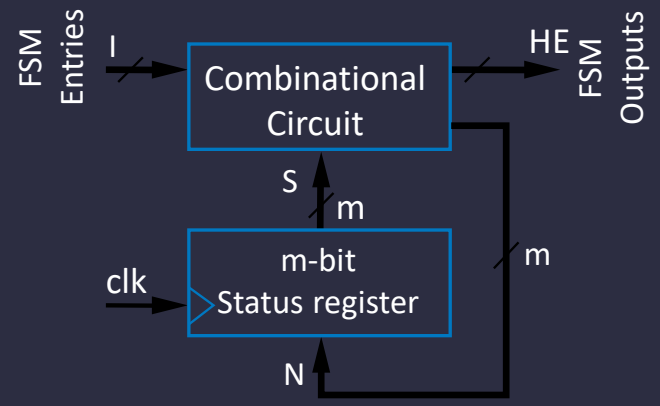


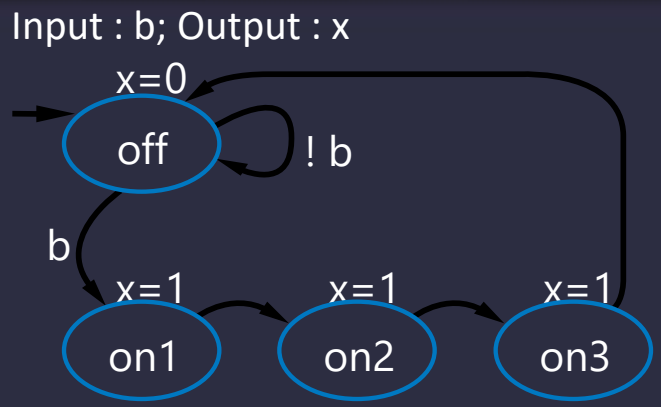


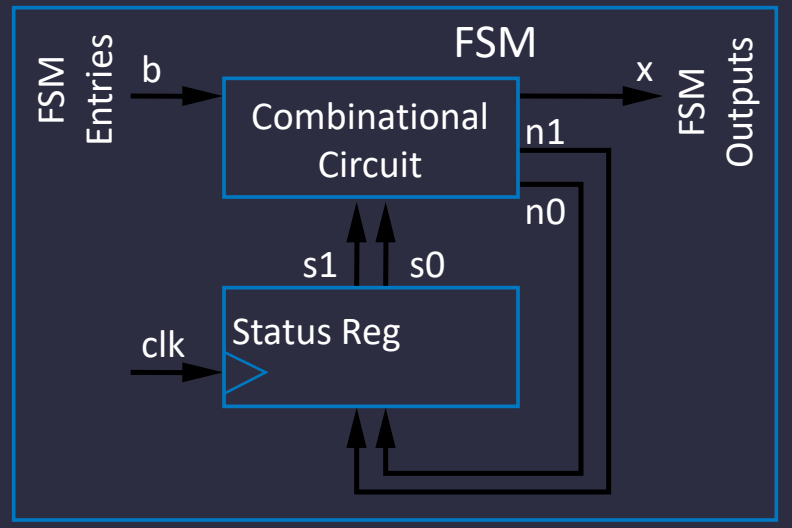


FSM Architecture

- How is FSM implemented as a sequential circuit ?
  - Status register - to keep the current state
  - Combinational circuit – to calculate output and next state

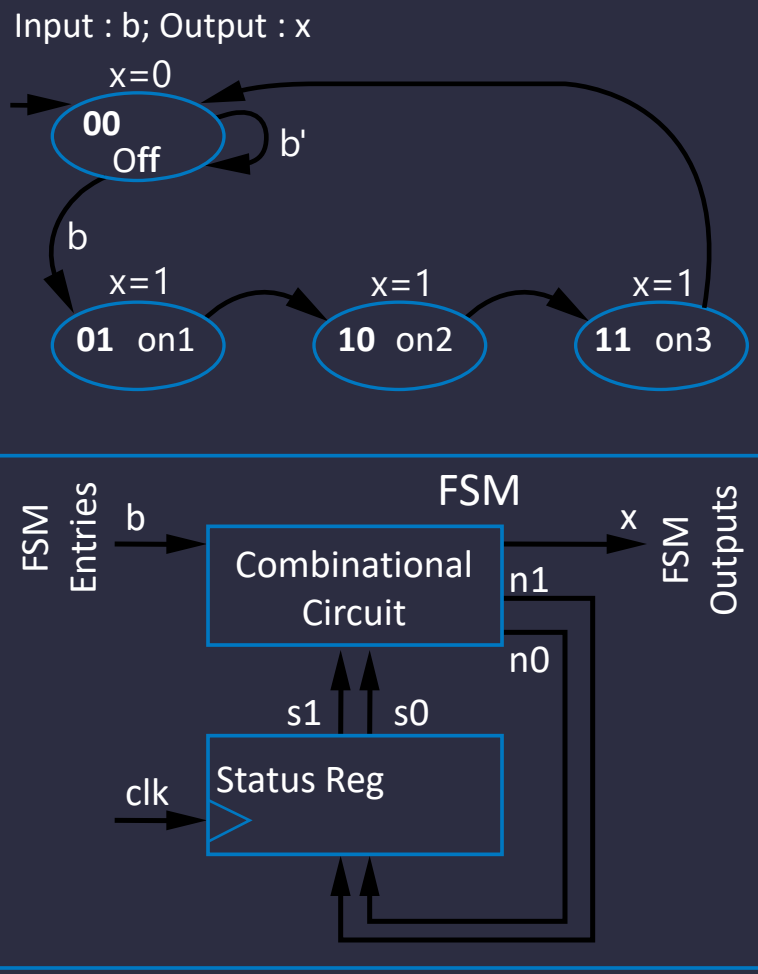






FSM Design Example

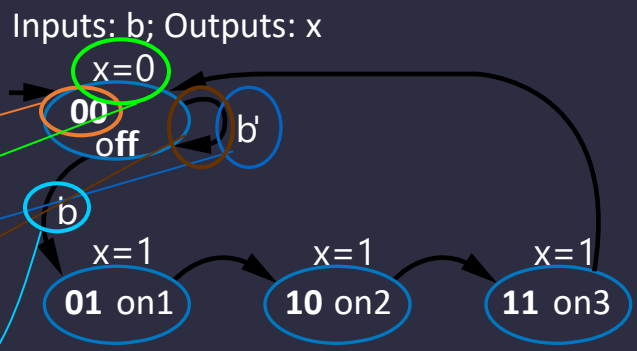
- Step 1: Analyze the requirements
- Step 2: Extract architecture
  - 2-bit state register (for 4 states)
  - Input b, output x
  - Next status signals n1, n0
- Step 3: Encode the states

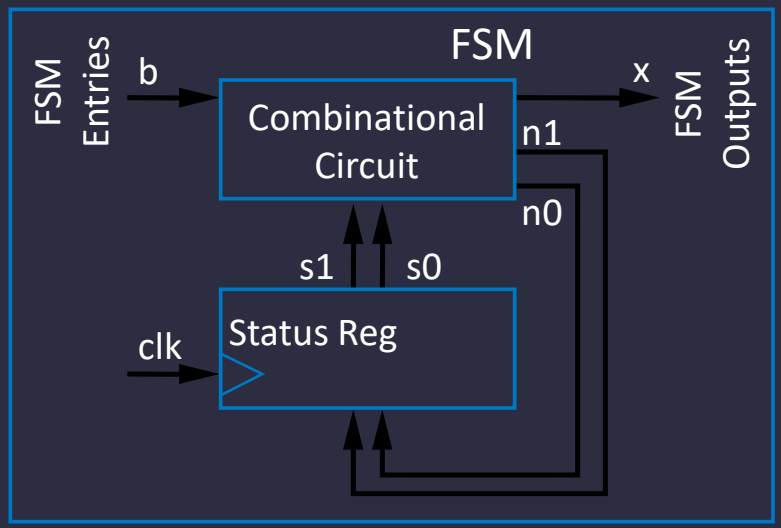


FSM Design Example

- Step 4: Create a Truth Table

	Entries			Outputs		
	s1	s0	b	x	n1	n0
Off	0	0	0	0	0	0
	0	0	1	0	0	1
On1	0	1	0	1	1	0
	0	1	1	1	1	0
On2	1	0	0	1	1	1
	1	0	1	1	1	1
On3	1	1	0	1	0	0
	1	1	1	1	0	0

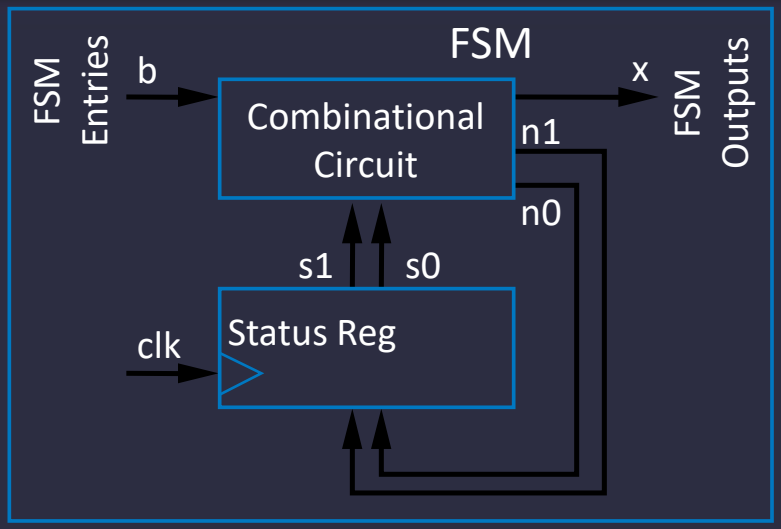




FSM Design Example

- Step 5: Combinational Circuit Implementation

	Entries			Outputs		
	s1	s0	b	x	n1	n0
Off	0	0	0	0	0	0
	0	0	1	0	0	1
On1	0	1	0	1	1	0
	0	1	1	1	1	0
On2	1	0	0	1	1	1
	1	0	1	1	1	1
On3	1	1	0	1	0	0
	1	1	1	1	0	0



x = s1 | s0

n1 = s1's0b' | s1's0b | s1s0'b' | s1s0'b

n1 = s1's0 | s1s0'

n0 = s1's0'b | s1s0'b' | s1s0'b

n0 = s1's0'b | s1s0'

FSM Design Example

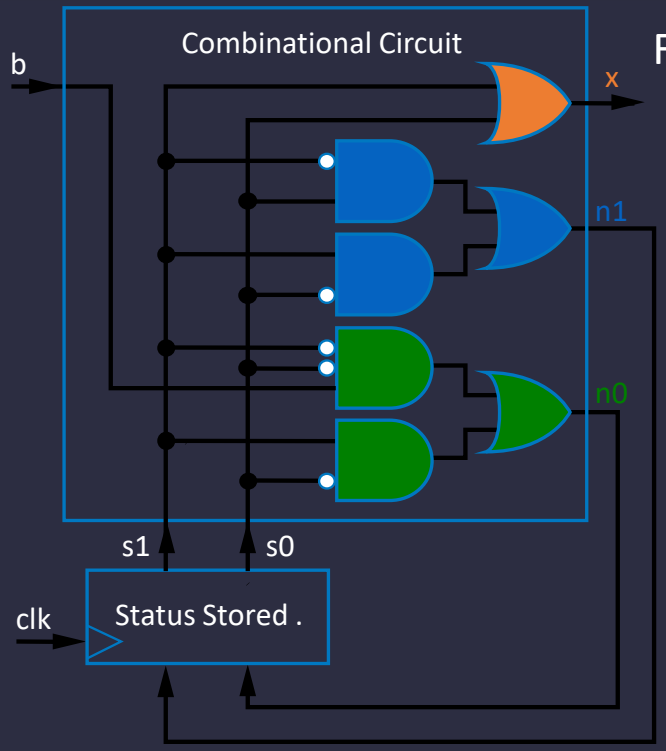
- Step 5: Build the combinational circuit

	Entries			Outputs		
	s1	s0	b	x	n1	n0
Off	0	0	0	0	0	0
	0	0	1	0	0	1
On1	0	1	0	1	1	0
	0	1	1	1	1	0
On2	1	0	0	1	1	1
	1	0	1	1	1	1
On3	1	1	0	1	0	0
	1	1	1	1	0	0

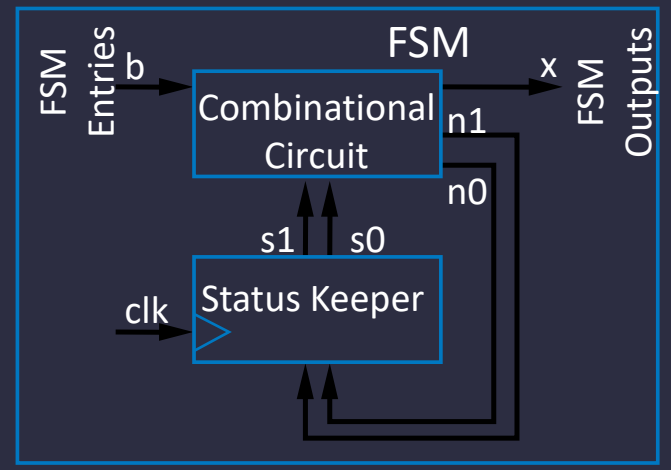
x = s1 | s0

n1 = s1's0 | s1s0'

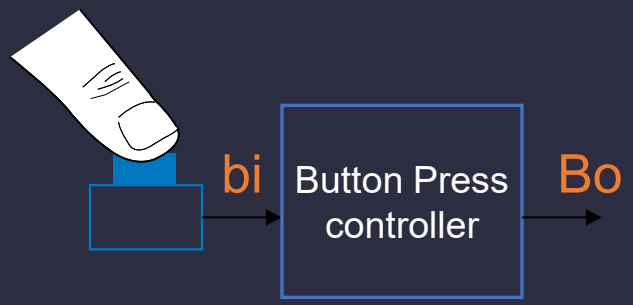
n0 = s1's0'b | s1s0'

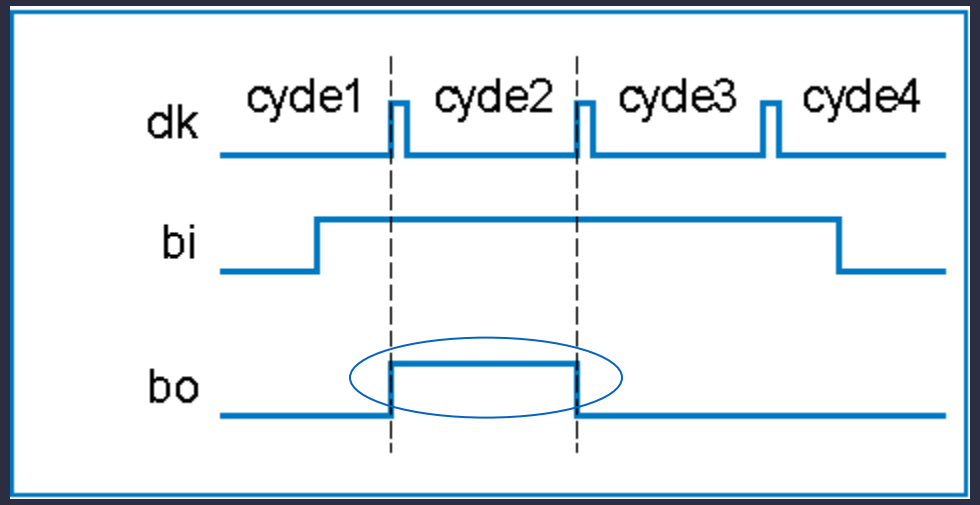


FSM Outputs



Button Filter Module Example

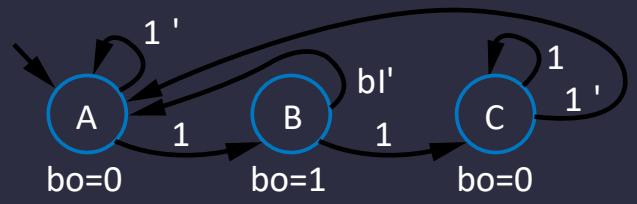




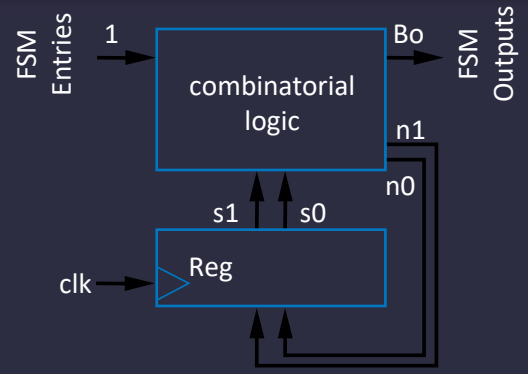
- cycle when the button is pressed A design that produces a pulse is desired.

Button Filter Module Example

FSM Inputs : bi; FSM outputs : bo

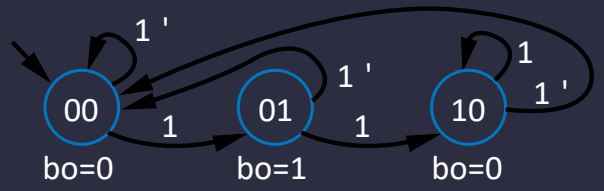


Step 1: FSM



Step 2: Build architecture

FSM input : bi; FSM output : bo

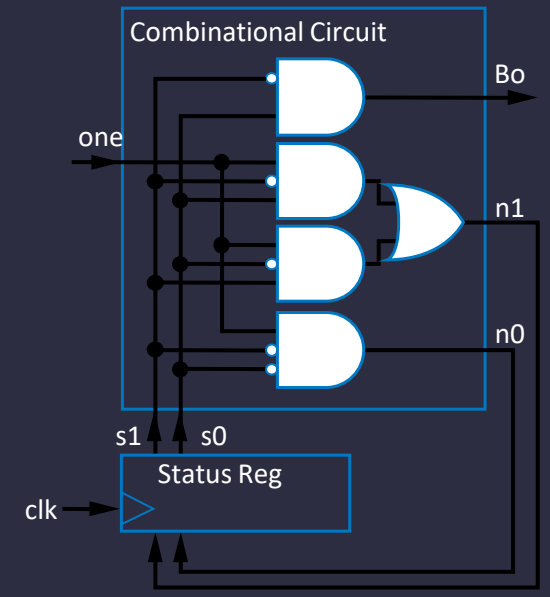


Step 3: State coding

Entries			Outputs		
s1	s0	bi	n1	n0	bo
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	0	0

Step 4: Status Chart

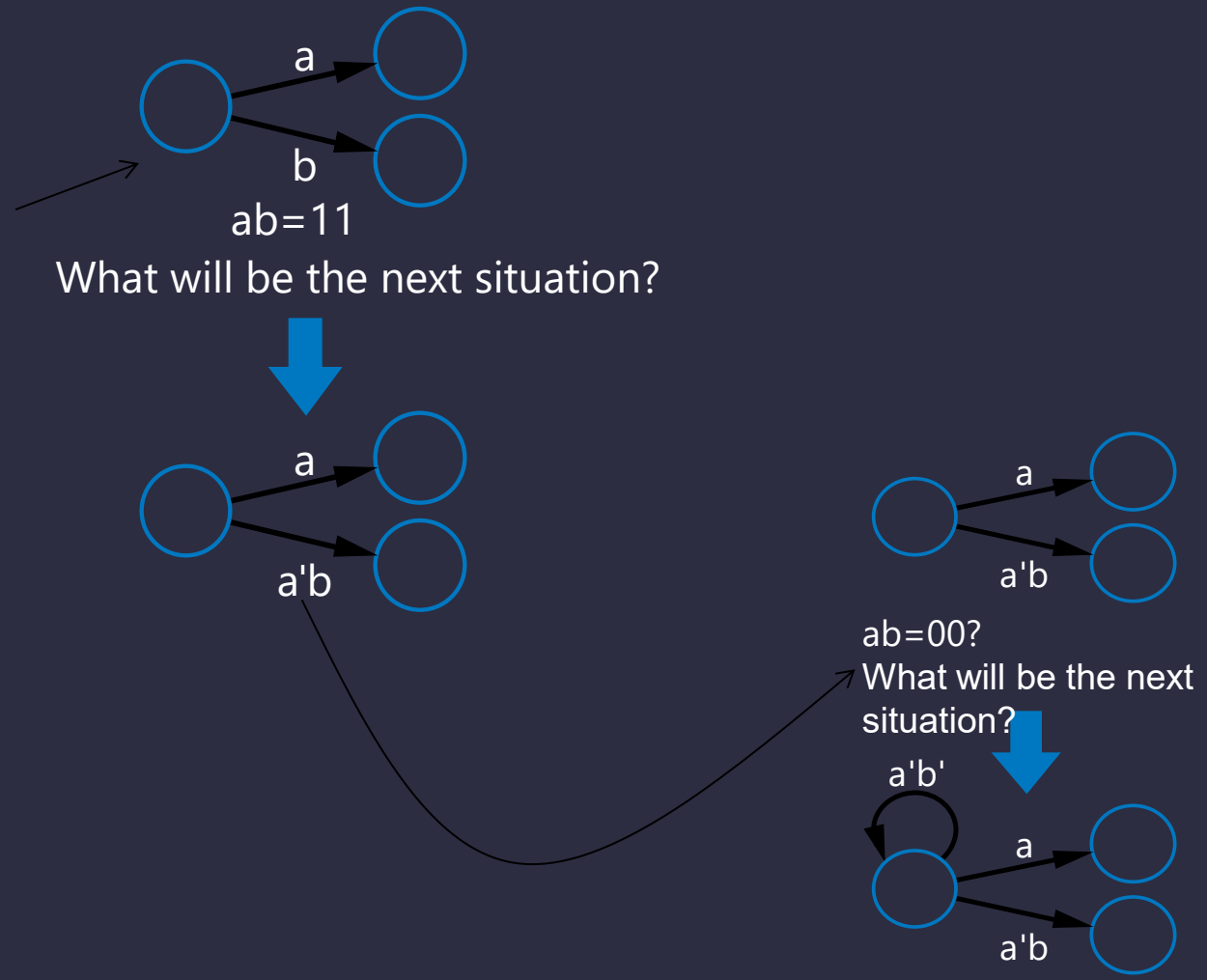
n1 = s1's0bi | s1s0bi
n0 = s1's0'bi
bo = s1's0bi' | s1's0bi = s1s0



Step 5: Build Circuit

FSM Design Errors

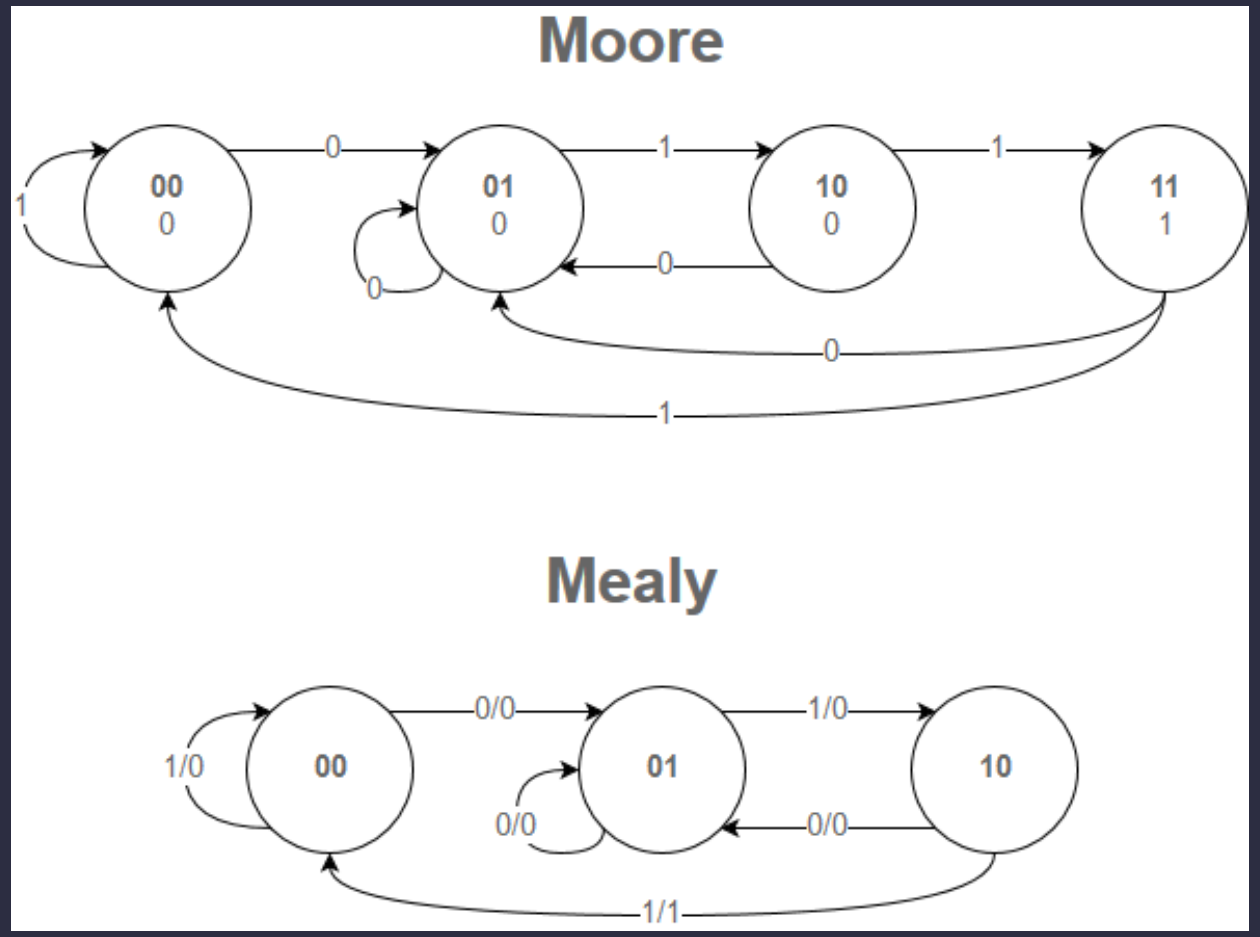
- *In conditions in transitions between states, only one transition condition must be true at a time*



Course

- State Machines
  - State Machines Verilog Representations

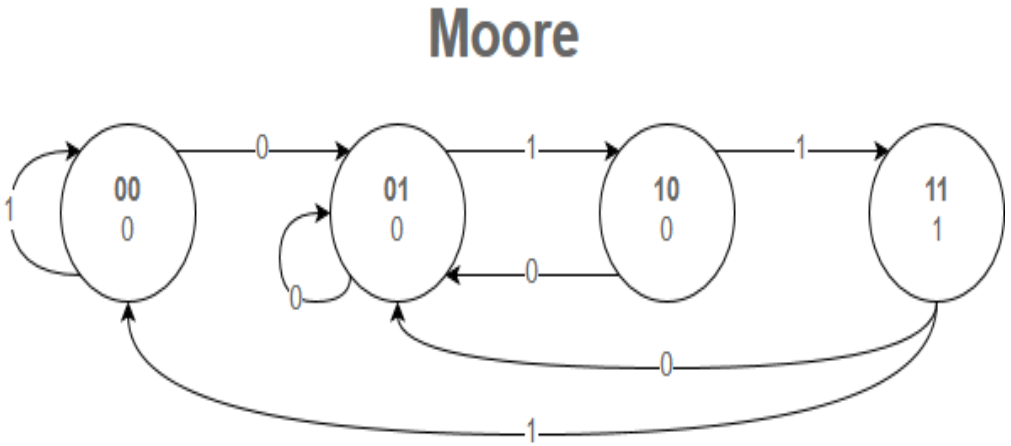
Durum Makinaları



Moore State Machine Examples

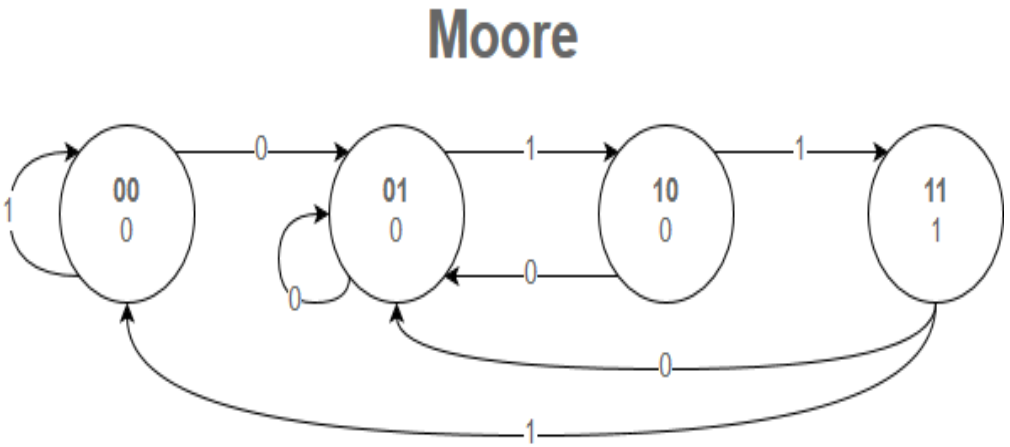
State Machines

```
module stateMachineTest (input clk, input in, output reg out)

reg [1:0] state, stateNext;
reg outNext;

initial begin
        state = 0;
        stateNext = 0;
        out = 0;
        outNext = 0;
end

always@(posedge clk) begin
        state <= stateNext;
        out <= outNext;
end
```



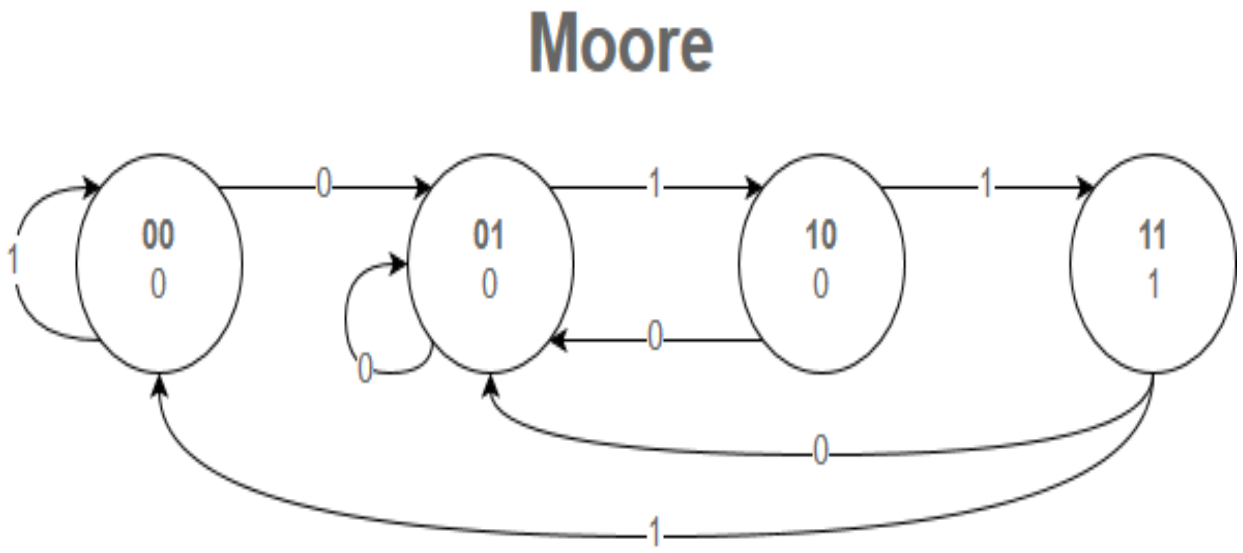
State Machines

```
always@(*) begin
        stateNext = state;
        outNext = out;
        case(state)
                0: begin
                        if(in == 0)begin
                                stateNext = 1;
                                outNext = 0;
                        end
                end
        endcase
end

endmodule
```



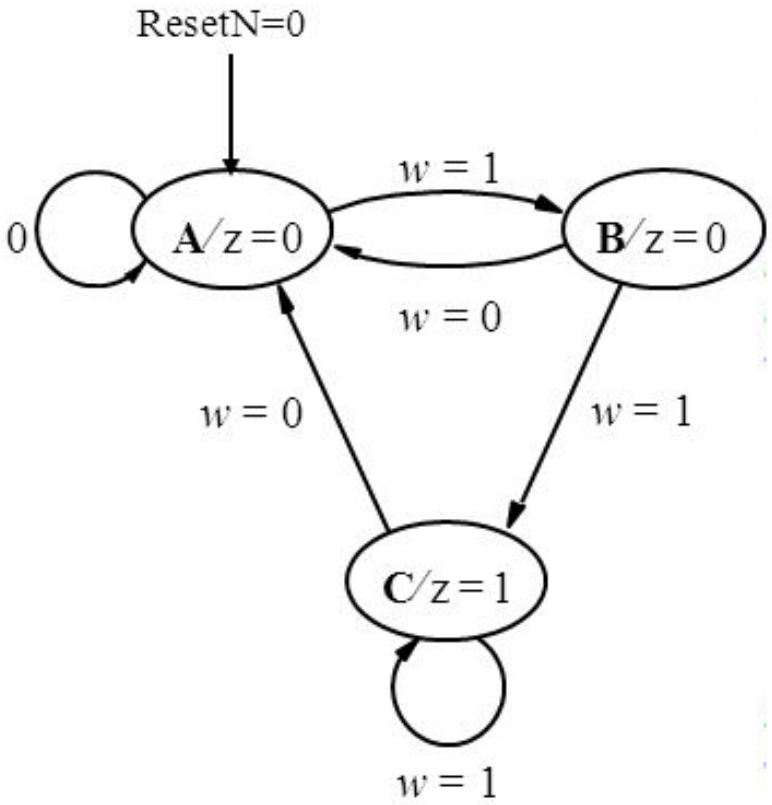
State Machines

```
1: begin
        if(in == 1)begin
                stateNext = 2;
                outNext = 0;
        end
end
2: begin
        if(in == 1)begin
                stateNext = 3;
                outNext = 1;
        end else begin
                stateNext = 1;
                outNext = 0;
        end
end
3: begin
        if(in == 0)begin
                stateNext = 1;
                outNext = 0;
        end else begin
                stateNext = 0;
                outNext = 0;
        end
end
```



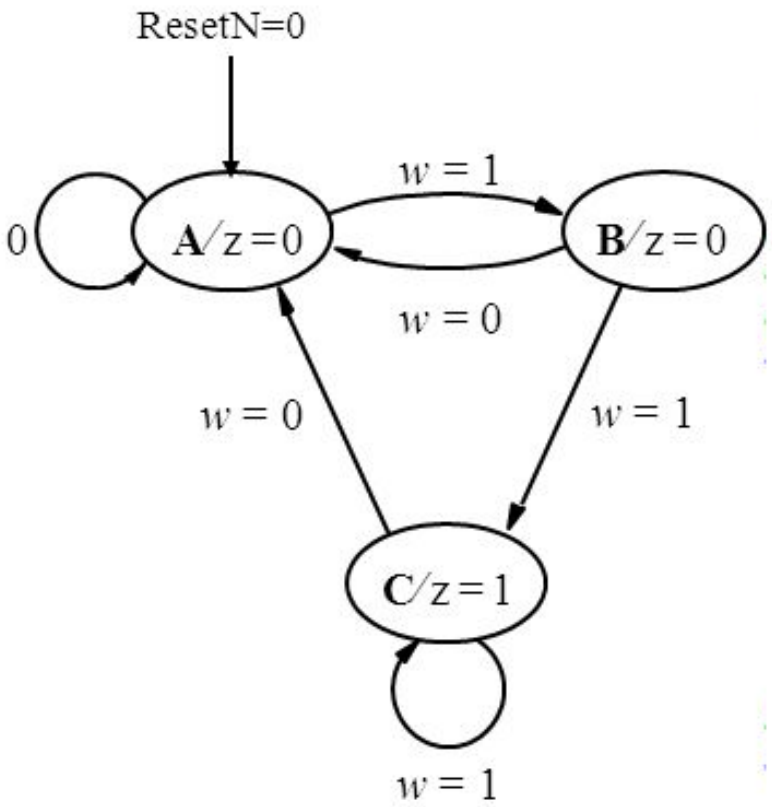
State Machines

```
module mooreMachine2 (input clk, input ResetN, input w, output reg z)

reg [1:0] state, stateNext;
reg zNext;

initial begin
        state = 0;
        stateNext = 0;
        z = 0;
        zNext = 0;
end

always@(posedge clk) begin
        state <= stateNext;
        z <= zNext;
end
```



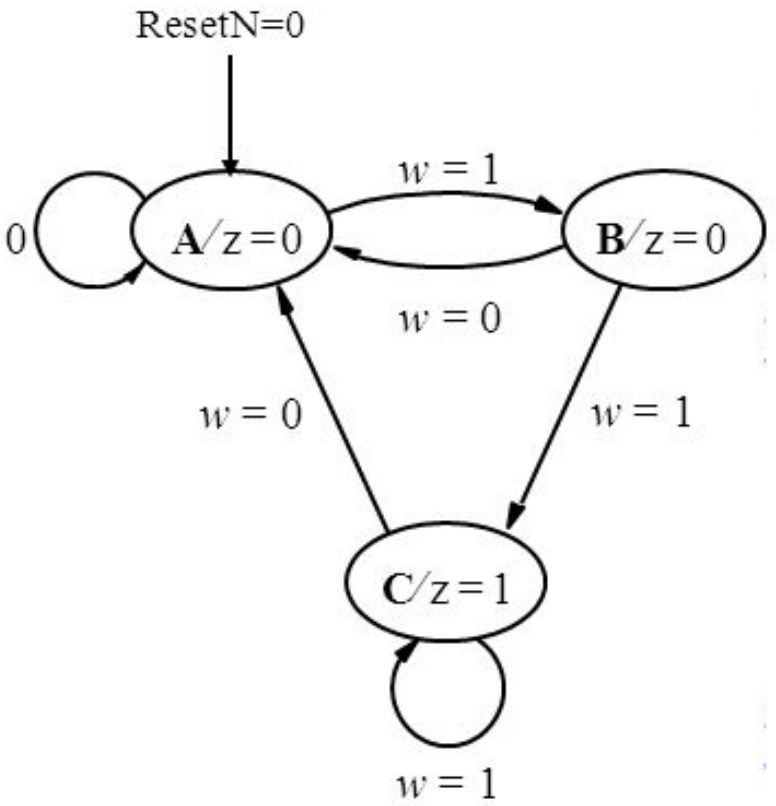
State Machines

```
always@(*) begin
        stateNext = state;
        zNext = z;
        if(ResetN) begin
                stateNext = 0;
                zNext = 0;
        end else begin
                case(state)
                        0: begin
                                if(w == 1)begin
                                        stateNext = 1;
                                        zNext = 0;
                                end
                        end
                endcase
        end
end
endmodule
```



State Machines

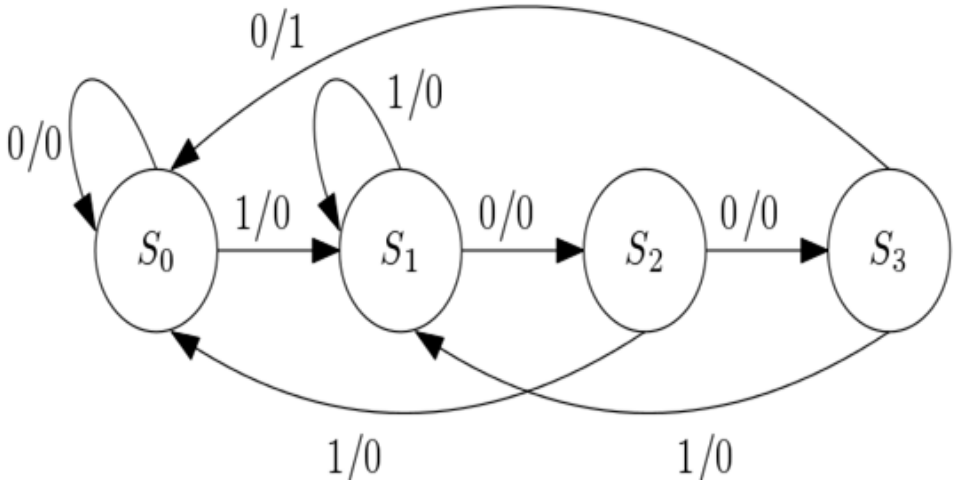
```
1: begin
        if(w == 0)begin
                stateNext = 0;
                zNext = 0;
        end else begin
                stateNext = 2;
                zNext = 1;
        end
end
2: begin
        if(w == 0)begin
                stateNext = 0;
                zNext = 0;
        end
end
```



Mealy State Machine Examples

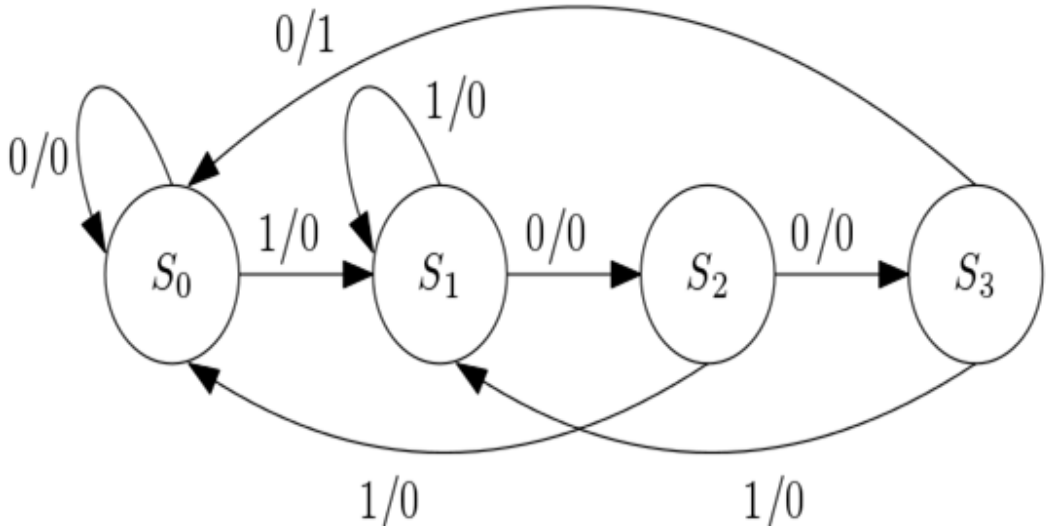
State Machines

```
module mooreMachine1 (input clk, input in, output reg out)

reg [1:0] state, stateNext;

initial begin
        state = 0;
        stateNext = 0;
        out = 0;
end

always@(posedge clk) begin
        state <= stateNext;
end
```



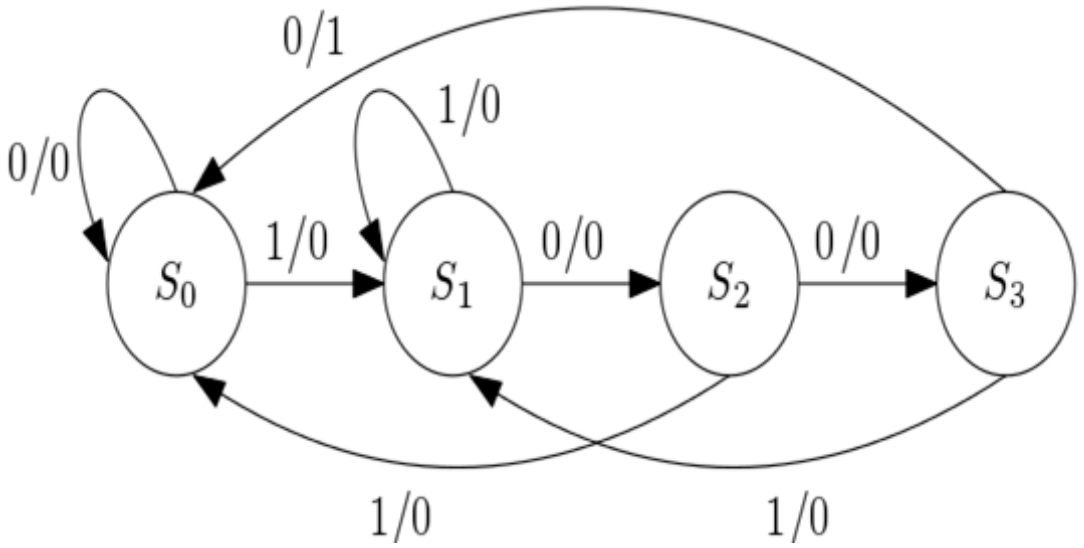
State Machines

```
always@(*) begin
        stateNext = state;
        out = 0;
        case(state)
                0: begin
                        if(in == 0)begin
                                out = 0;
                        end else begin
                                out = 1;
                                stateNext = 1;
                        end
                end
        endcase
end
```



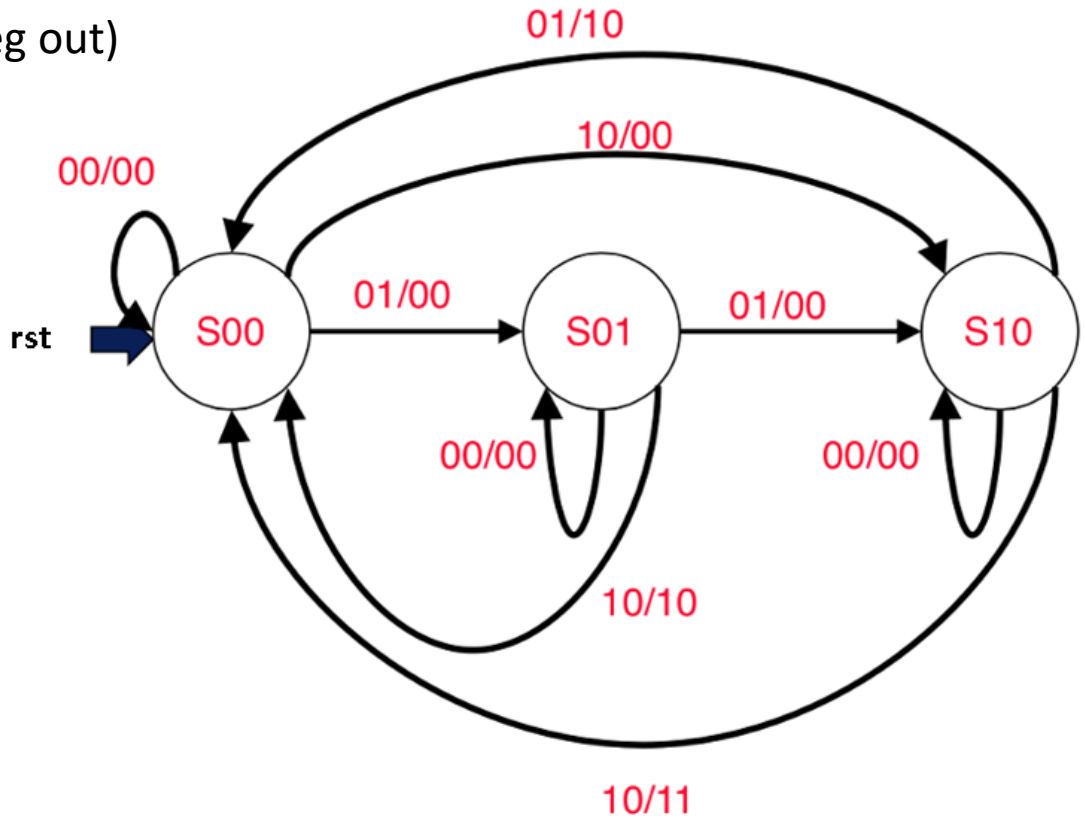
State Machines

```
1: begin
        if(in == 0)begin
                out = 0;
                stateNext = 2;
        end else begin
                out = 0;
        end
end
2: begin
        if(in == 0)begin
                out = 0;
                stateNext = 3;
        end else begin
                out = 0;
                stateNext = 0;
        end
end
3: begin
        if(in == 0)begin
                out = 1;
                stateNext = 0;
        end else begin
                out = 0;
                stateNext = 1;
        end
end
```



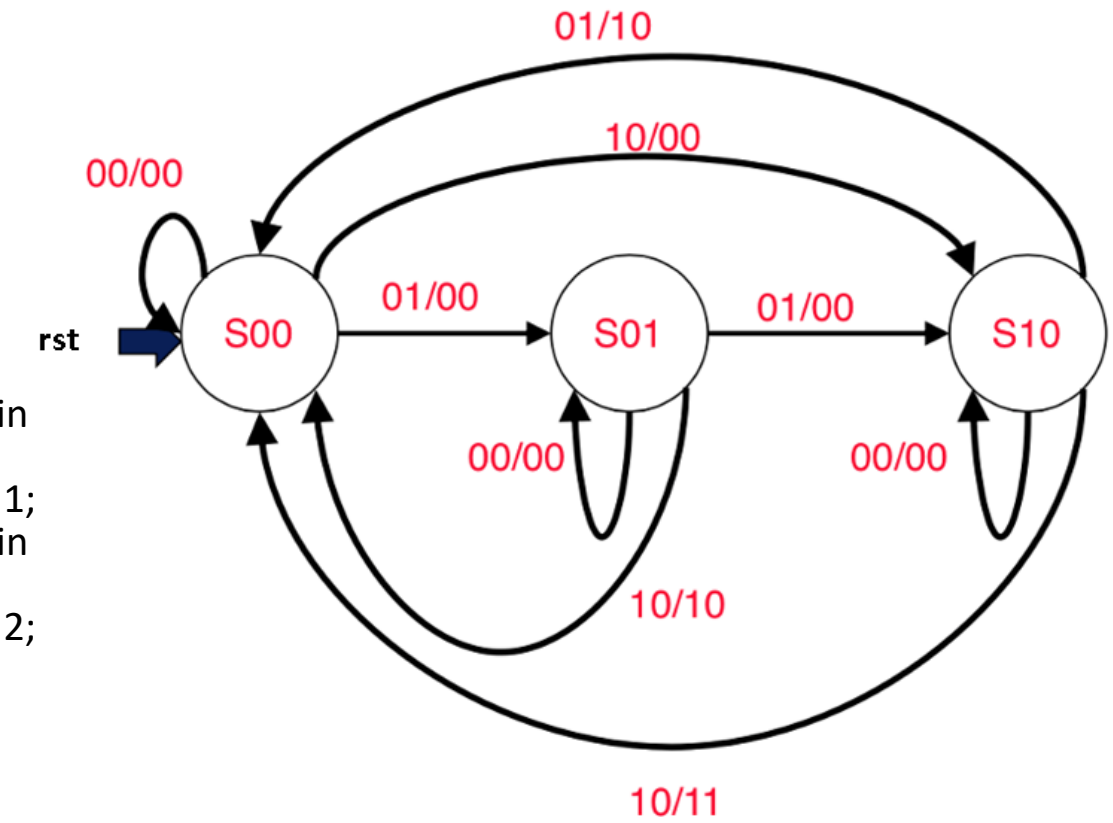
State Machines

```
module mooreMachine2 (input clk, input in, output reg out)

reg [1:0] state, stateNext;

initial begin
        state = 0;
        stateNext = 0;
        out = 0;
end

always@(posedge clk) begin
        state <= stateNext;
end
```



State Machines

```
always@(*) begin
        stateNext = state;
        out = 0;
        if(rst) begin
                stateNext = 0;
                out = 0;
        end else begin
                case(state)
                        0: begin
                                if(in == 0)begin
                                        out = 0;
                                end else if(in == 1) begin
                                        out = 0;
                                        stateNext = 1;
                                end else if(in == 2) begin
                                        out = 0;
                                        stateNext = 2;
                                end
                        end             endcase
        end
end

endmodule
```



State Machines

```
1: begin
            if(in == 0)begin
                        out = 0;
            end else if(in == 1) begin
                        out = 0;
                        stateNext = 2;
            end else if(in == 2) begin
                        out = 2;
                        stateNext = 2;
            end
end
2: begin
            if(in == 0)begin
                        out = 0;
            end else if(in == 1) begin
                        out = 2;
                        stateNext = 0;
            end else if(in == 2) begin
                        out = 3;
                        stateNext = 0;
            end
end
```