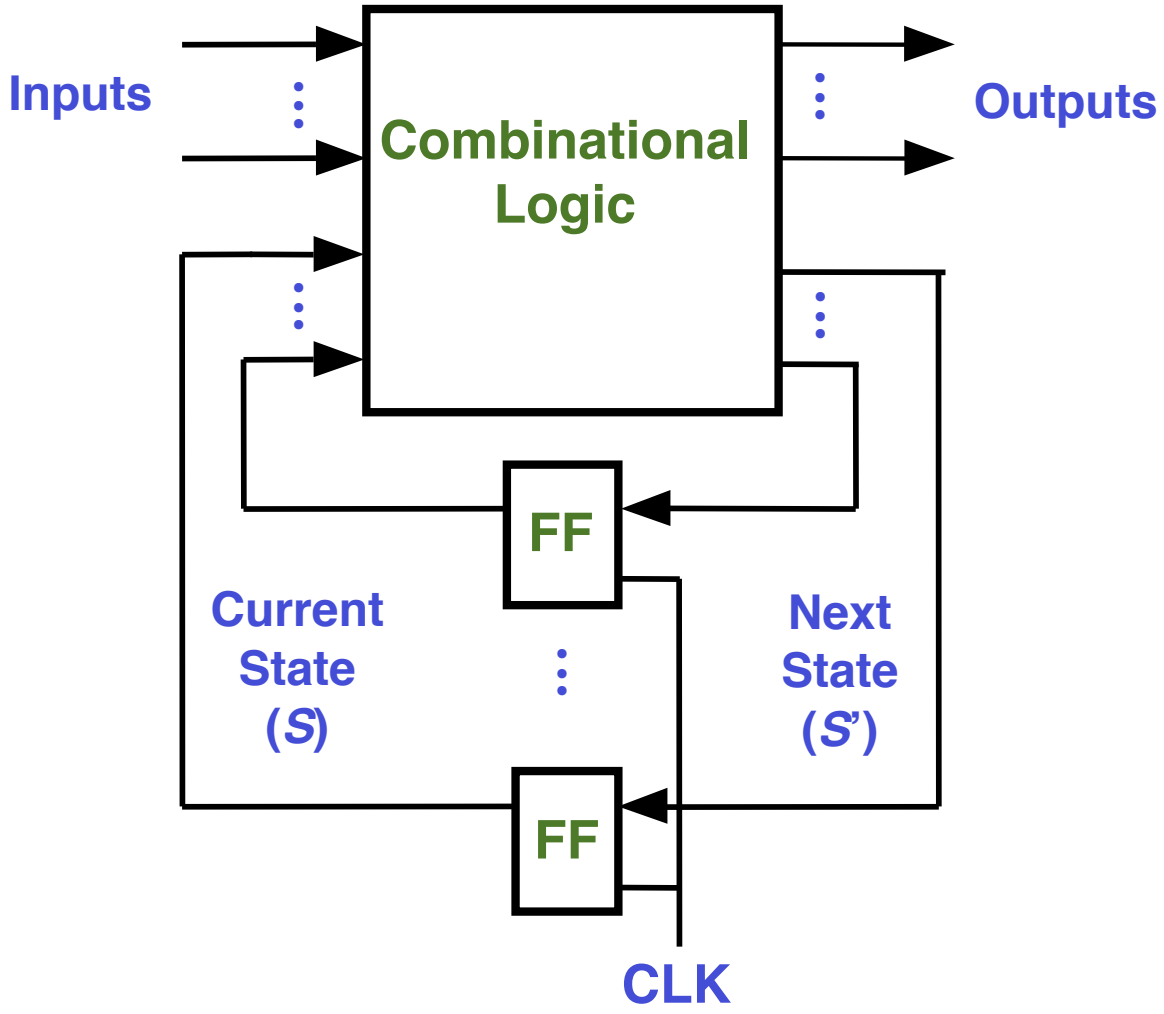
ECE 2300
Digital Logic & Computer Organization

Fall 2016

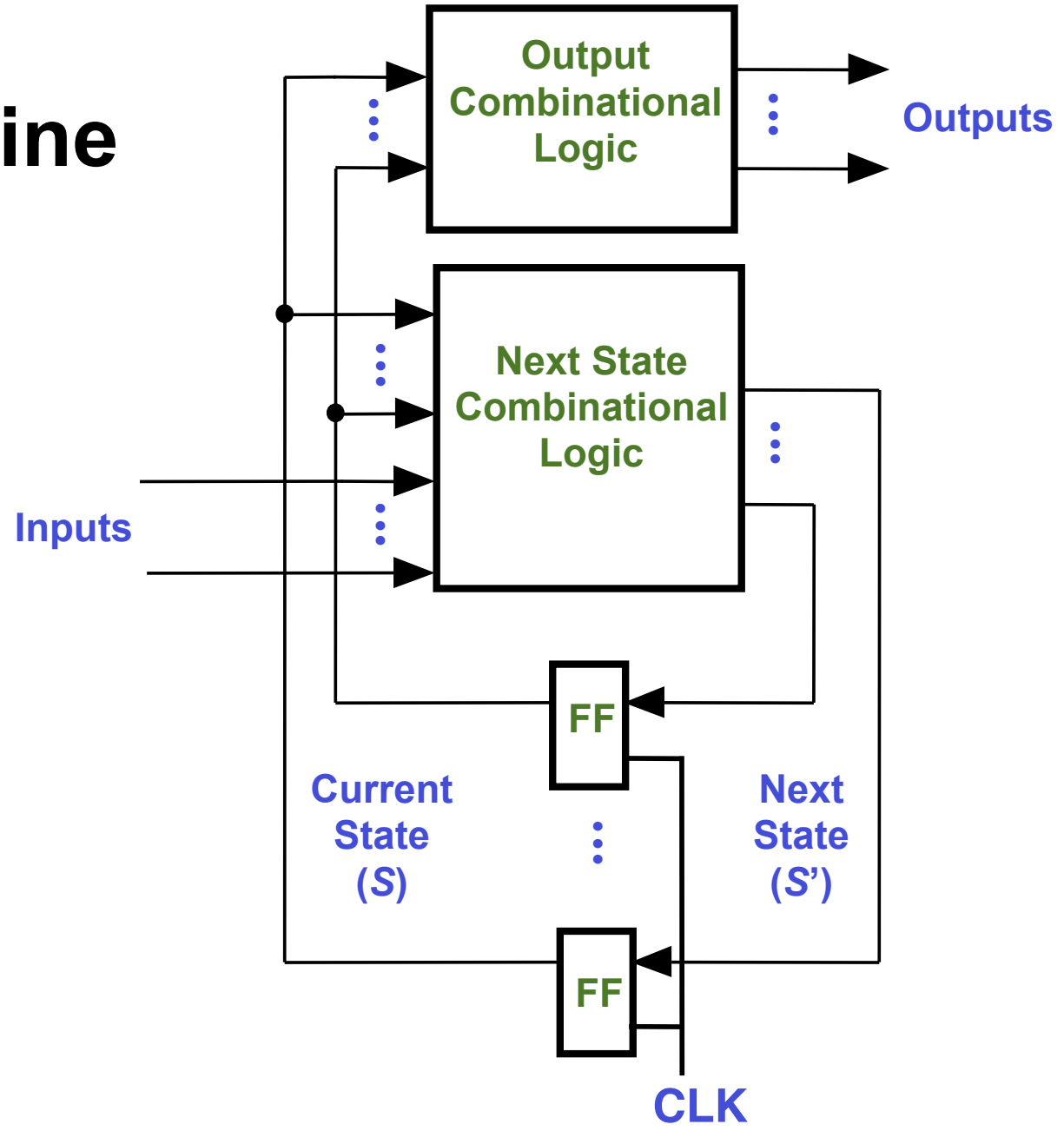
More Finite State Machines

Cornell University

FSM: General Form

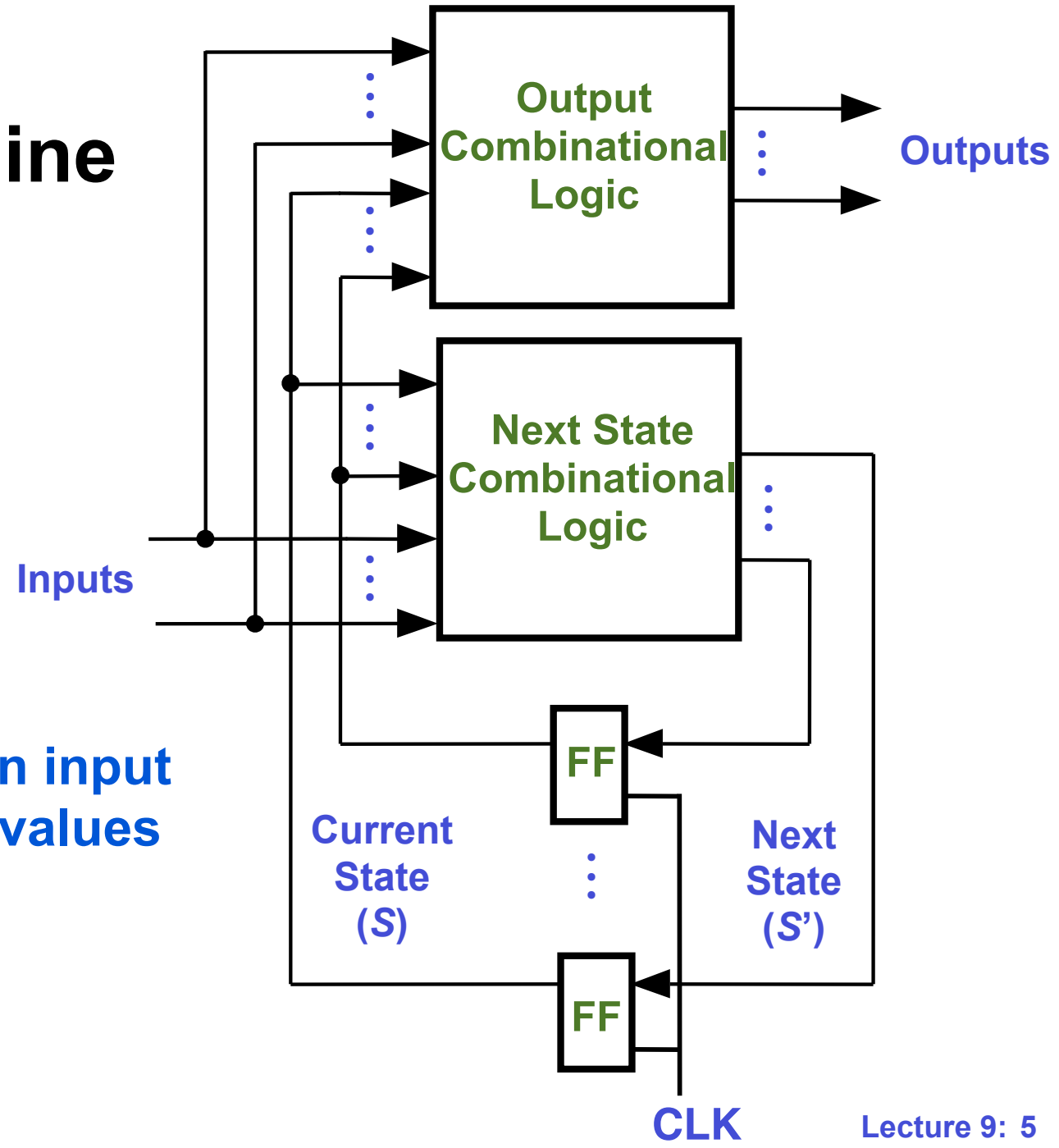


Moore Machine



Outputs depend on current state value

Mealy Machine



Outputs depend on input and current state values

Example FSM: Pattern Detector

- Monitors the input, and outputs a 1 whenever a specified input pattern is detected

- Example: Output a 1 whenever 111 is detected on the input for 3 consecutive clock cycles
  - Overlapping patterns also detected (1111...)

- Input *In*
- Output *Out*
- *Reset* causes FSM to start in initial state
- *Clock* input not shown (always present)

111 Pattern Detector Timing

FSM Design Procedure

(1) Understand the problem statement and determine inputs and outputs

(2) Identify states and create a *state diagram*

(3) Determine the number of required D FFs

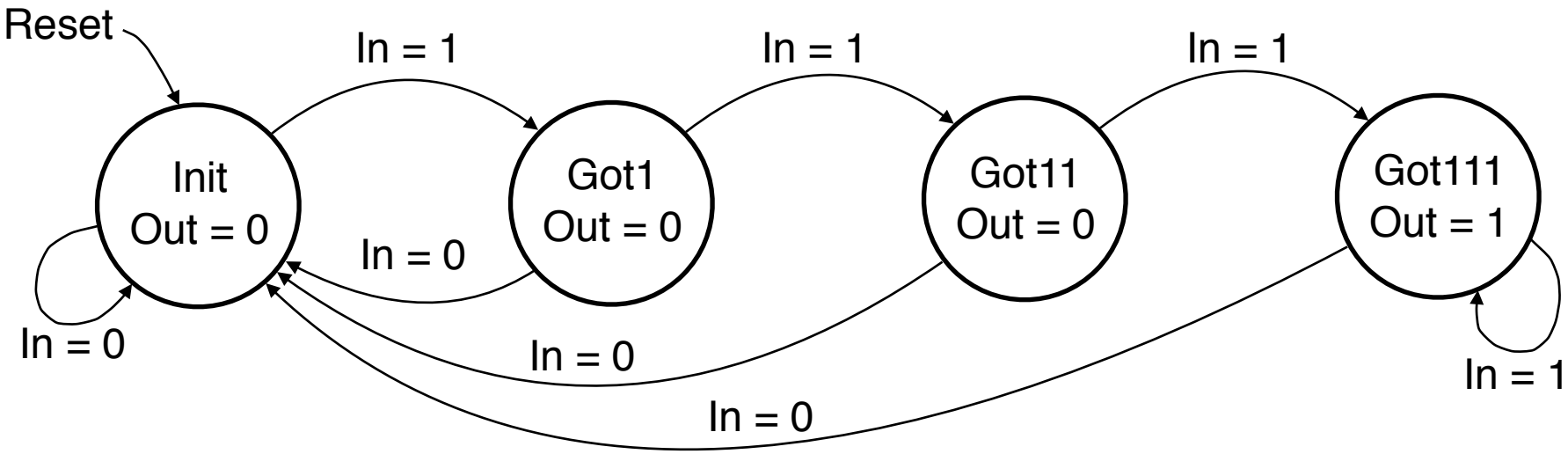
(4) Implement combinational logic for outputs and next state

(5) Simulate the circuit to test its operation

Transition/Output Table

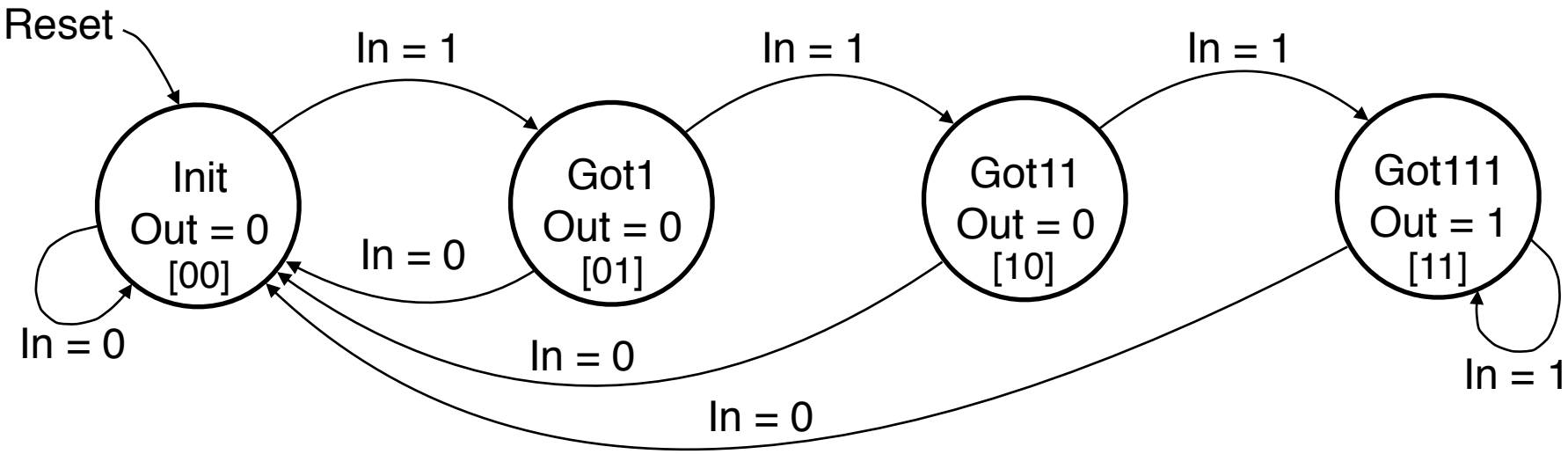
- Shows the next state (*S*') and output values for each combination of current state (*S*) and inputs
- Used to derive the minimized *state transition* (*S*') and *output* Boolean equations
- Version 1: uses descriptive state names
- Version 2: uses state binary encodings
- Text shows different type of T/O table

Moore Transition/Output Table 1



Current State (*S*)	Next State (*S'*)		Out
	In = 0	In = 1	
Init	Init	Got1	0
Got1	Init	Got11	0
Got11	Init	Got111	0
Got111	Init	Got111	1

Moore Transition/Output Table 2

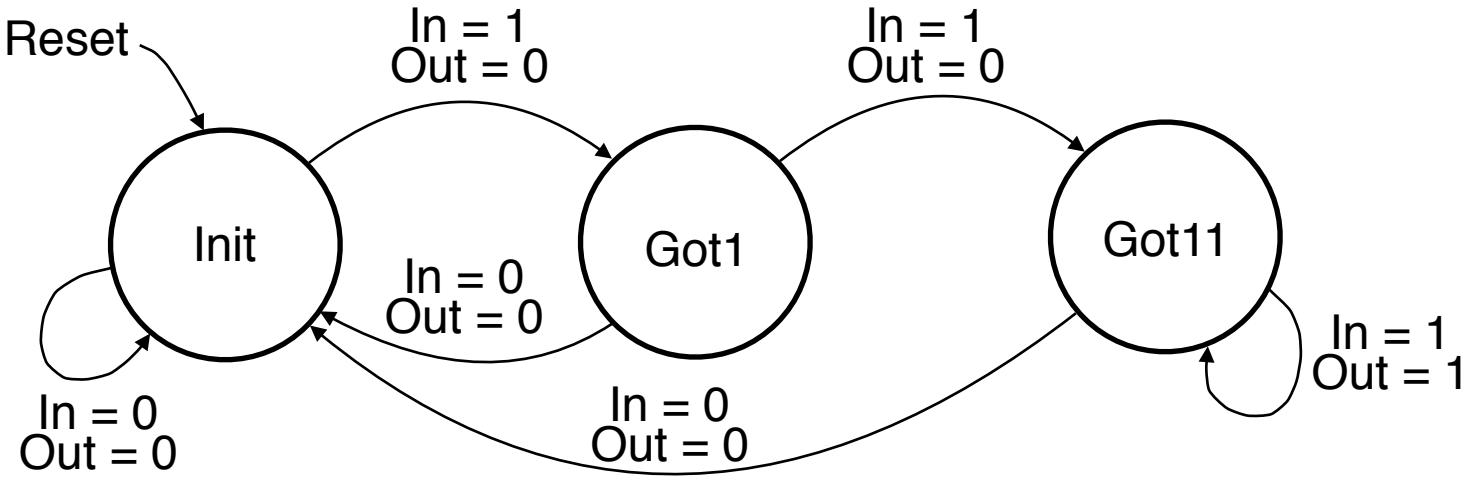


S_1 S_0	S_1' S_0'		Out
	In = 0	In = 1	
0 0	0 0	0 1	0
0 1	0 0	1 0	0
1 0	0 0	1 1	0
1 1	0 0	1 1	1

Minimized Equations for *S'* and *Out*

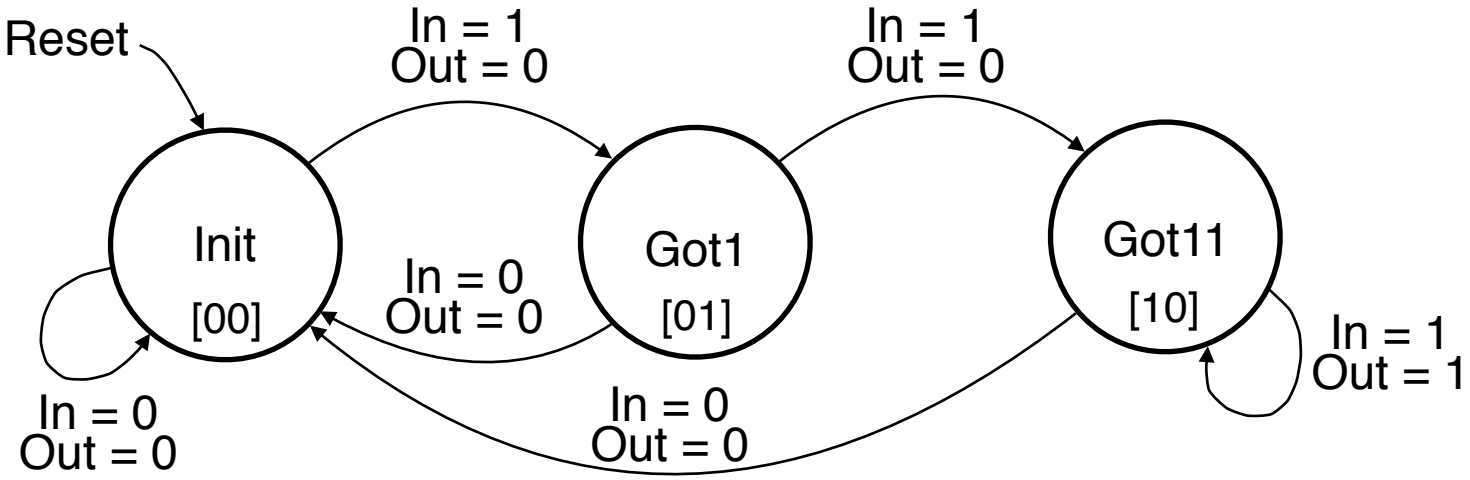
S_1 S_0	S_1' S_0'		Out
	In = 0	In = 1	
0 0	0 0	0 1	0
0 1	0 0	1 0	0
1 0	0 0	1 1	0
1 1	0 0	1 1	1

Mealy Transition/Output Table 1



Current State (*S*)	Next State (*S*'), Out	
	In = 0	In = 1
Init	Init, 0	Got1, 0
Got1	Init, 0	Got11, 0
Got11	Init, 0	Got11, 1

Mealy Transition/Output Table 2



S_1 S_0	S_1' S_0', Out	
	In = 0	In = 1
0 0	0 0, 0	0 1, 0
0 1	0 0, 0	1 0, 0
1 0	0 0, 0	1 0, 1

Minimized Equations for *S'* and *Out*

S_1 S_0	S_1' S_0', Out	
	In = 0	In = 1
0 0	0 0, 0	0 1, 0
0 1	0 0, 0	1 0, 0
1 0	0 0, 0	1 0, 1

FSMs in Verilog

<module statement>
<input and output declarations>

<reg declarations>

<parameter or typedef statement>

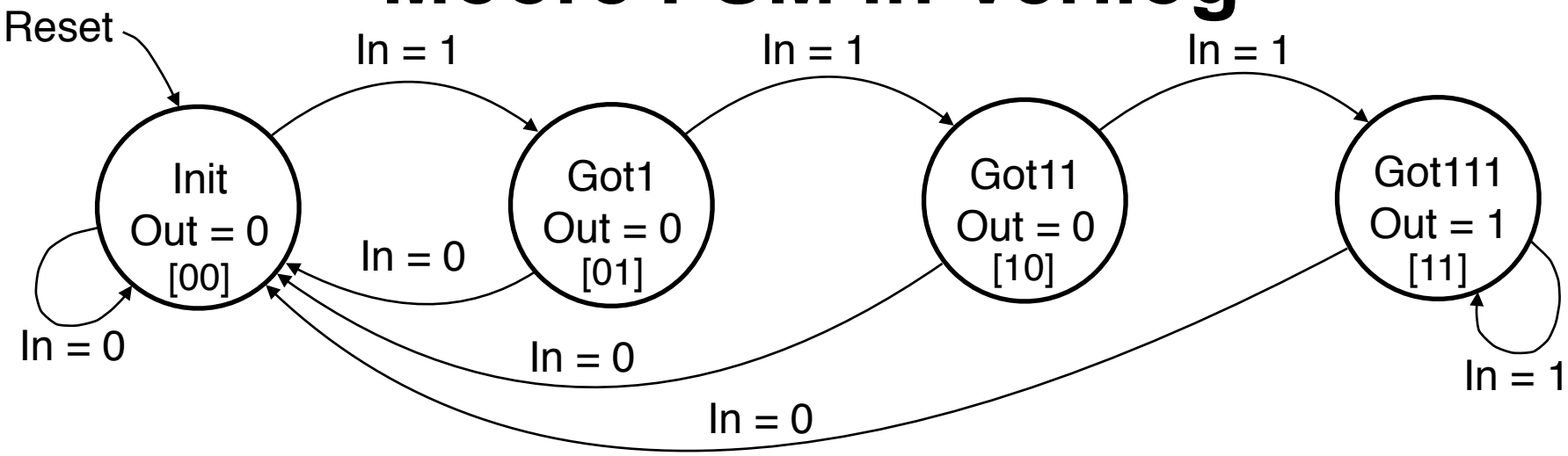
<*always* block for next state>

<*always* block for output>

<*always* block for state D FFs>

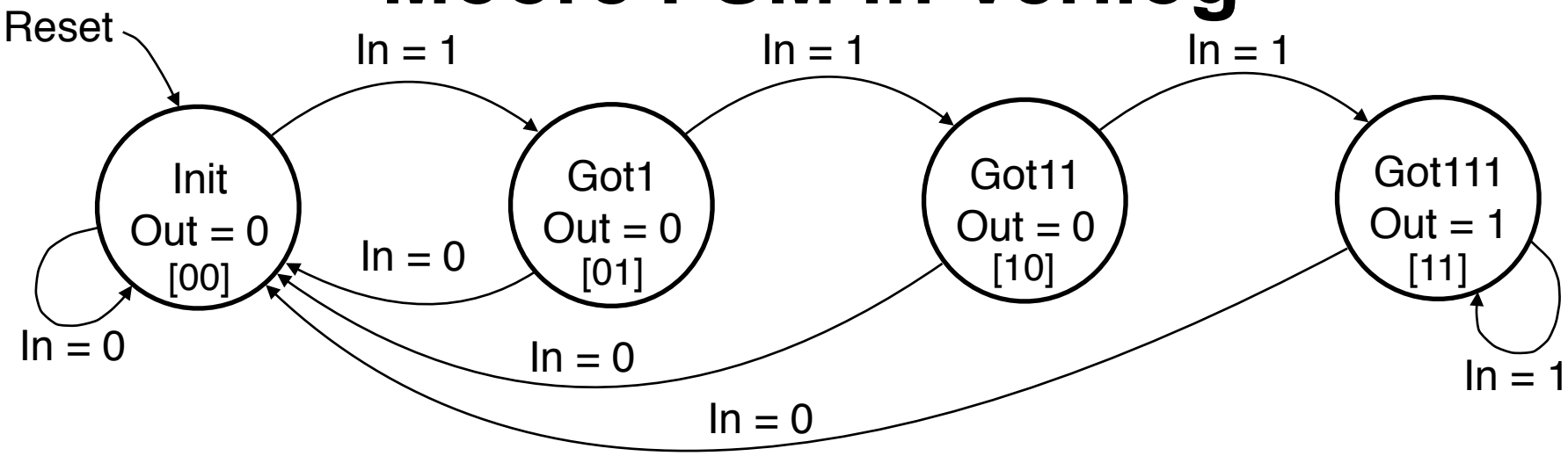
endmodule

Moore FSM in Verilog




```
module PatDectMoore (Clk, In, Reset, Out);
input Clk, In, Reset;
output Out;

reg Out;
reg [1:0] Scurr, Snext;

parameter [1:0] Init = 2'b00,
                Got1 = 2'b01,
                Got11 = 2'b10,
                Got111 = 2'b11;
```

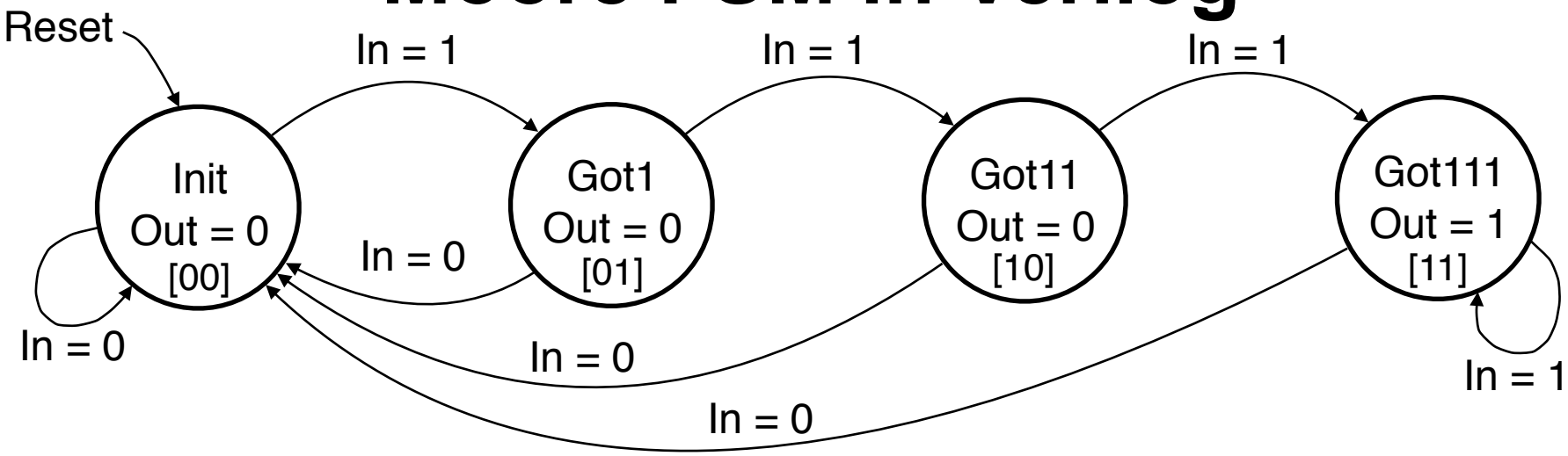
Moore FSM in Verilog




```
always @ (In, Scurr)
begin
  case (Scurr)
    Init: if (In == 1) Snext = Got1; else Snext = Init;
    Got1: if (In == 1) Snext = Got11; else Snext = Init;
    Got11: if (In == 1) Snext = Got111; else Snext = Init;
    Got111: if (In == 1) Snext = Got111; else Snext = Init;
    default: Snext = Init;
  endcase
end
```

next state comb logic

Moore FSM in Verilog

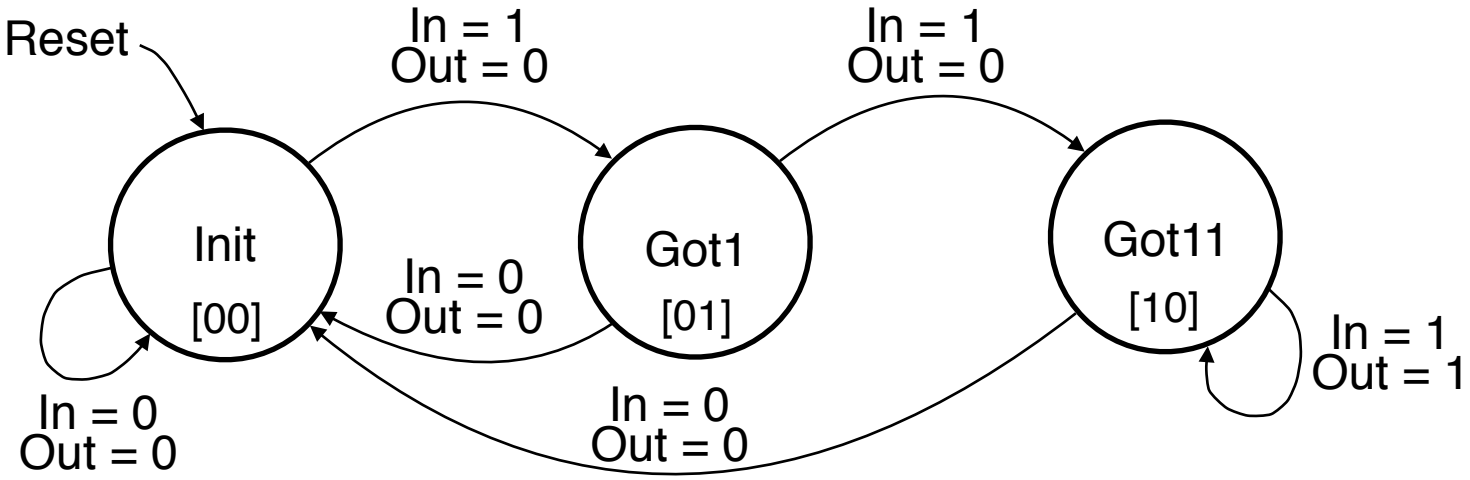



```
always @ (Scurr)
  if (Scurr == Got111) Out = 1; else Out = 0;

always @ (posedge Clk, posedge Reset)
  if (Reset == 1) Scurr <= Init; else Scurr <= Snext;

endmodule
```

output comb logic

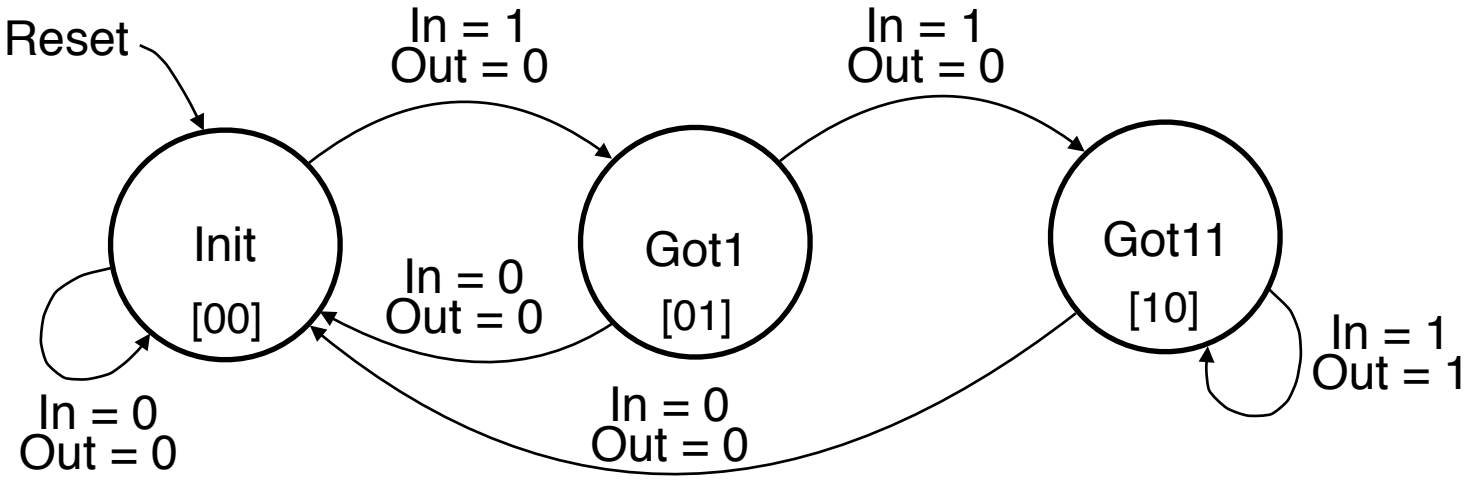
clock + async reset state D FFs

Mealy FSM in Verilog




```
module PatDectMealy (Clk, In, Reset, Out);
input Clk, In, Reset;
output Out;

reg Out;
reg [1:0] Scurr, Snext;

parameter [1:0] Init = 2'b00,
                Got1 = 2'b01,
                Got11 = 2'b10;
```

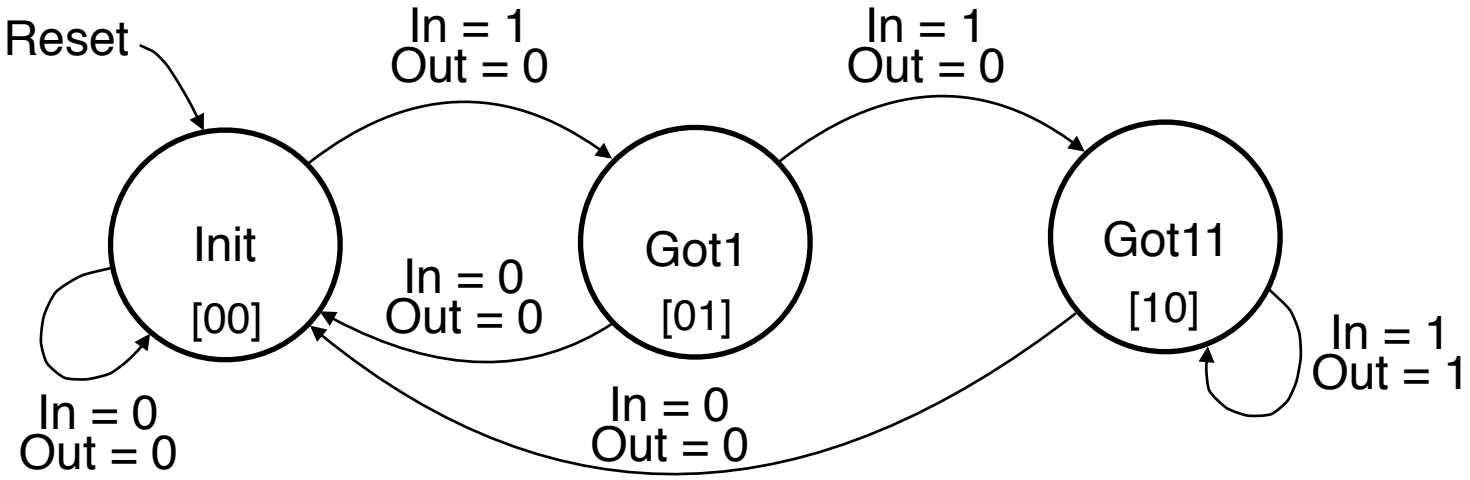
Mealy FSM in Verilog




```
always @ (In, Scurr)
begin
  case (Scurr)
    Init: if (In == 1) Snext = Got1; else Snext = Init;
    Got1: if (In == 1) Snext = Got11; else Snext = Init;
    Got11: if (In == 1) Snext = Got11; else Snext = Init;
    default: Snext = Init;
  endcase
end
```

next state comb logic

Mealy FSM in Verilog




```
always @ (Scurr, In)
  if ((Scurr == Got11) && (In == 1)) Out = 1; else Out = 0;

always @ (posedge Clk, posedge Reset)
  if (Reset == 1) Scurr <= Init; else Scurr <= Snext;

endmodule
```

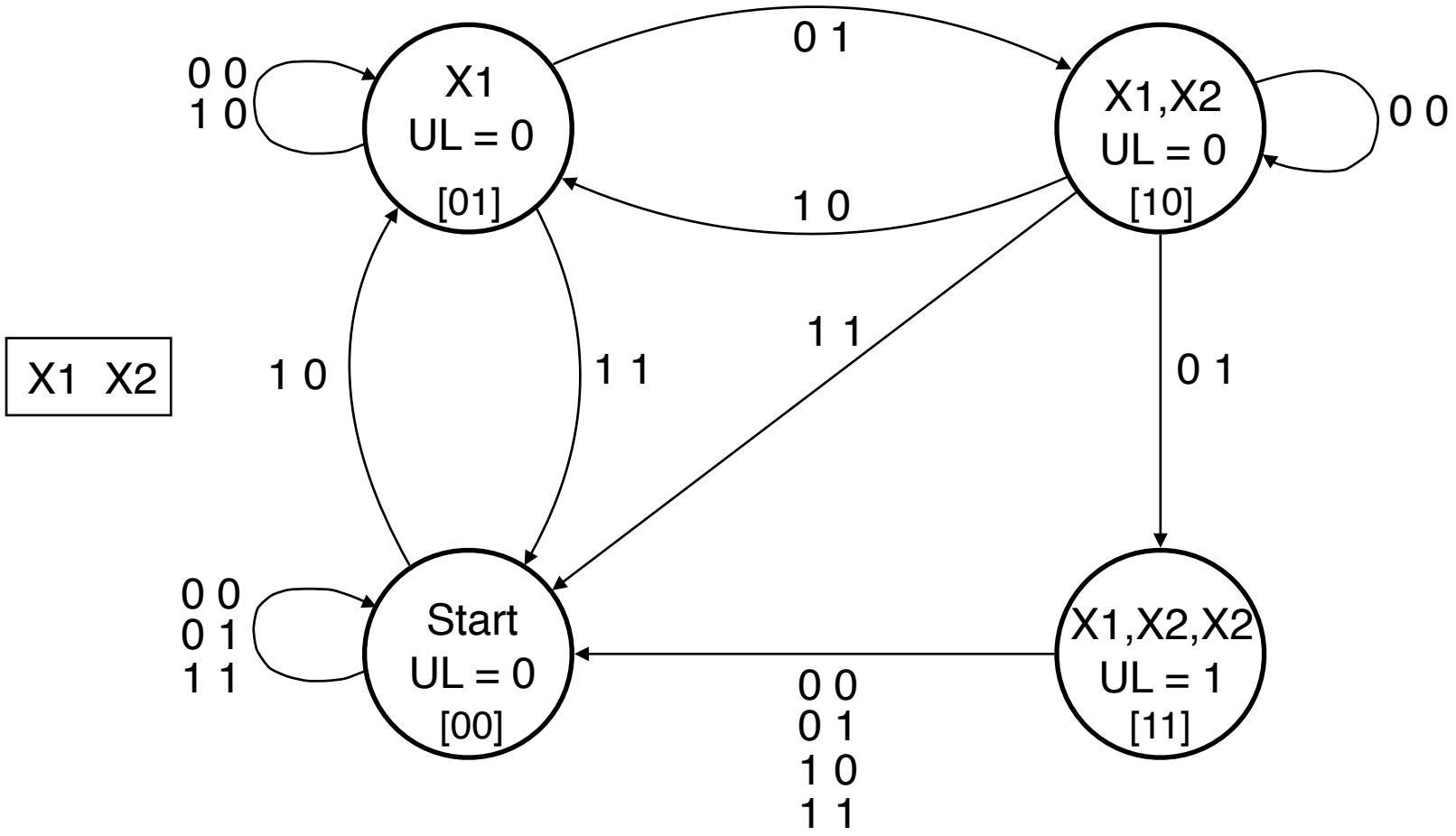
output comb logic

clock + async reset
state D FFs

Example FSM: Pushbutton Lock

- Two pushbutton inputs, X1 and X2
- One output, UL ("Unlock")
- UL = 1 when X1 is pushed, followed by X2 being pushed twice (X1, X2, X2)
- Represent X1 and X2 as two bit input
  - 00: neither button pushed
  - 01: X2 pushed
  - 10: X1 pushed
  - 11: both pushed simultaneously

Pushbutton Lock: Moore State Diagram



Transition/Output Tables

Minimized Equations for *S'* and *UL*

Next State *always* Block

Output *always* Block

State FFs *always* Block

Next Time

Factoring FSMs

Analyzing FSMs