

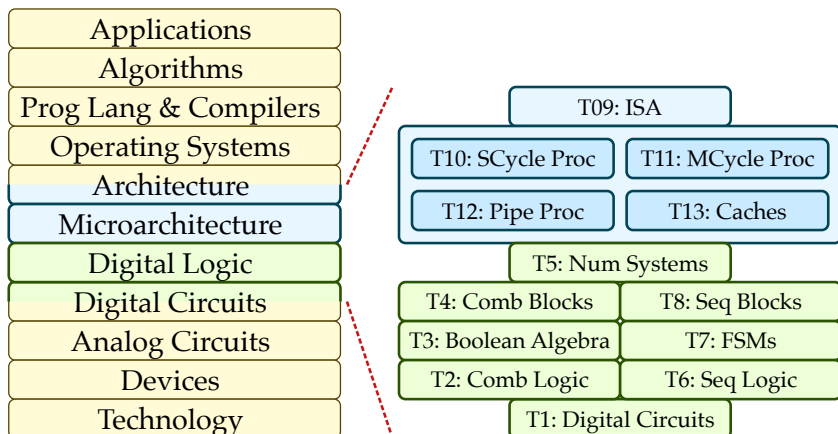
ECE 2300 Digital Logic and Computer Organization Fall 2026

Topic 7: Finite State Machines

School of Electrical and Computer Engineering
Cornell University

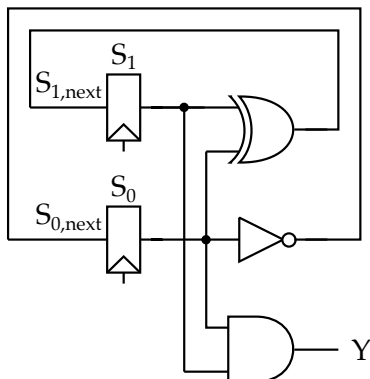
revision: 2026-10-08-01-44

1	From Logic Gates to Finite-State Machines	3
2	Finite-State Machines	5
2.1.	FSM Abstraction Triangle	6
2.2.	Moore FSMs	7
2.3.	Mealy FSMs	10
2.4.	FSM State Encoding	12
2.5.	Designing FSMs	13
2.6.	Factoring FSMs	15
3	Verilog Modeling	19
3.1.	Gate-Level Modeling of FSMs	19
3.2.	RTL Modeling of FSMs	23



Copyright © 2026 Christopher Batten. All rights reserved. This handout was prepared by Prof. Christopher Batten at Cornell University for ECE 2300 / ENGRD 2300 Digital Logic and Computer Organization. Download and use of this handout is permitted for individual educational non-commercial purposes only. Redistribution either in part or in whole via both commercial or non-commercial means requires written permission.

1. From Logic Gates to Finite-State Machines



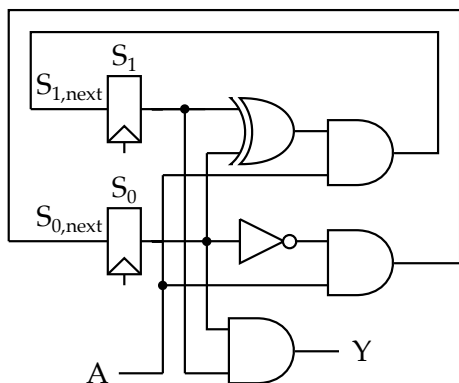
Truth Table

S_1	S_0	$S_{1,next}$	$S_{0,next}$	Y
0	0	0	1	0
0	1	1	0	0
1	0	1	1	0
1	1	0	0	1

Implicit-Clock Simulation Table

Assume all flip-flops are reset to zero.

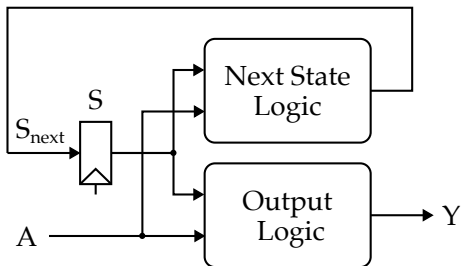
cycle	S_1	S_0	$S_{1,next}$	$S_{0,next}$	Y
0	0	0	0	1	0
1	0	1	1	0	0
2	1	0	1	1	0
3	1	1	0	0	1
4	0	0	0	1	0
5	0	1	1	0	0
6	1	0	1	1	0
7	1	1	0	0	1



Truth Table

S_1	S_0	A	$S_{1,next}$	$S_{0,next}$	Y
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

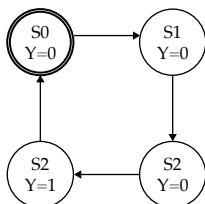
2. Finite-State Machines



- *Finite state machines* (FSM) are abstractions for sequential gate-level networks
 - Finite number of inputs (A)
 - Finite number of outputs (Y)
 - Finite number of states (S)
 - Designated reset state
 - FSM next-state equation ($S_{next}(S, A)$)
 - FSM output equation ($Y(S)$ or $Y(S, A)$)
- Two kinds of FSMs
 - Moore FSMs: Outputs only depend on current state ($Y(S)$)
 - Mealy FSMs: Outputs depend on current state and inputs ($Y(S, A)$)
- FSMs can be represented using:
 - FSM diagrams
 - FSM next state and output tables (i.e., truth tables)
 - FSM next state and output equations

2.1. FSM Abstraction Triangle

Diagrams



Next State & Output Tables

S_1	S_0	$S_{1,next}$	$S_{0,next}$	Y
0	0	0	1	0
0	1	1	0	0
1	0	1	1	0
1	1	0	0	1

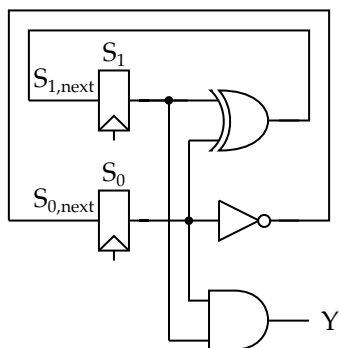
Next State & Output Equations

$$S_{0,next} = \overline{S_1} \overline{S_0} + S_1 \overline{S_0}$$

$$S_{1,next} = \overline{S_1} S_0 + S_1 \overline{S_0}$$

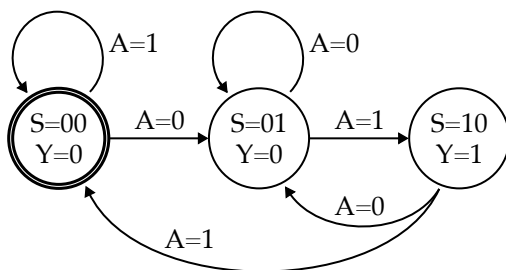
$$Y = S_1 S_0$$

Gate-Level Networks



2.2. Moore FSMs

FSM Diagram



FSM Next State and Output Table

S_1	S_0	A	$S_{1,next}$	$S_{0,next}$	Y
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			

FSM Implicit-Clock Simulation Table

cycle	S	A	S_{next}	Y
0	00	0	01	0
1		0		
2		1		
3		0		
4		0		
5		0		
6		1		
7		1		
8		0		
9		1		
10		0		

$S_1 S_0$

A \	00	01	11	10
0				
1				

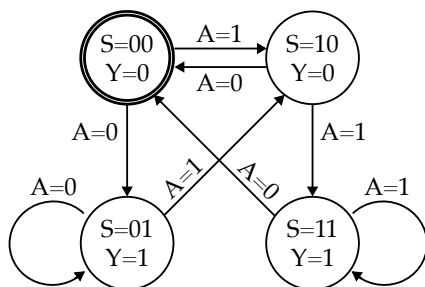
$S_1 S_0$

A \	00	01	11	10
0				
1				

$S_1 S_0$

A \	00	01	11	10
0				
1				

Implement the following FSM at the gate level.



What does this FSM do?

FSM Next State and Output Table

S_1	S_0	A	$S_{1,next}$	$S_{0,next}$	Y
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

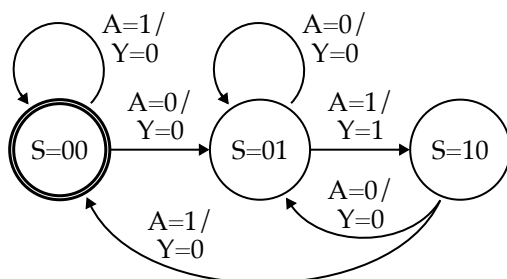
$A \backslash S_1 S_0$	00	01	11	10
0				
1				

$A \backslash S_1 S_0$	00	01	11	10
0				
1				

$A \backslash S_1 S_0$	00	01	11	10
0				
1				

2.3. Mealy FSMs

FSM Diagram



FSM Next State and Output Table

S_1	S_0	A	$S_{1,next}$	$S_{0,next}$	Y
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			

FSM Implicit-Clock
Simulation Table

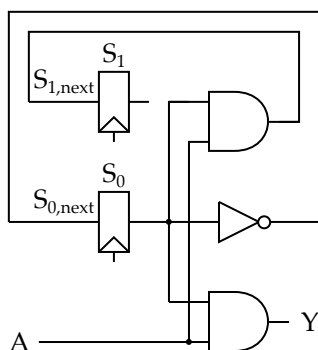
cycle	S	A	S_{next}	Y
0	00	0	01	0
1		0		
2		1		
3		0		
4		0		
5		0		
6		1		
7		1		
8		0		
9		1		
10		0		

FSM Next State and Output Equations

		S_1S_0			
		00	01	11	10
A	0				
	1				

		S_1S_0			
		00	01	11	10
A	0				
	1				

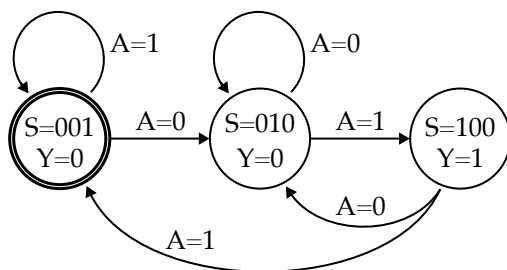
		S_1S_0			
		00	01	11	10
A	0				
	1				

FSM Sequential
Gate-Level Network

Optimized FSM Diagram

2.4. FSM State Encoding

- Often there is freedom in choosing how to encode the states
- Different encodings result in different FSM next state and output equations (and thus produce different gate-level implementations)
- Two common FSM encodings
 - *Binary encoding* uses a binary to encode each state
 - *One-hot encoding* uses a separate bit to encode each state



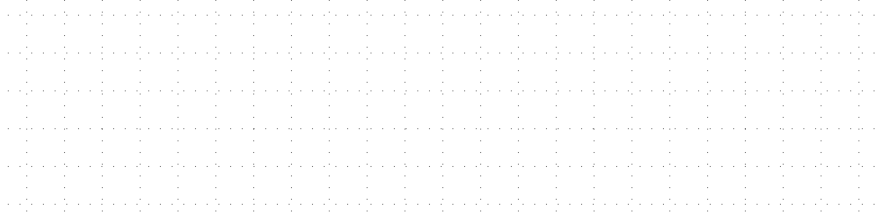
S_2	S_1	S_0	A	$S_{1,next}$	$S_{1,next}$	$S_{0,next}$	Y
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
1	0	0	0				
1	0	0	1				

2.5. Designing FSMs

- Step 1. Understand problem statement, determine inputs/outputs
- Step 2. Choose Moore vs. Mealy, identify states, create state diagram
- Step 3. Determine state encoding
- Step 4. Create FSM next state and output tables
- Step 5. Determine FSM next state and output equations

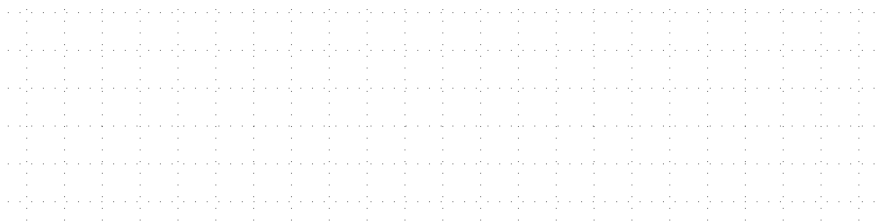
Design an FSM with one input (A) and two outputs (Y, Z) which monitors the input and sets Y to one when it has seen two or more consecutive ones and sets Z to one when it has seen three or more consecutive ones.

Step 1. Understand problem statement, determine inputs/outputs



Step 2. Choose Moore vs. Mealy, identify states, create state diagram



Step 3. Determine state encoding**Step 4. Create FSM next state and output tables****Step 3. Determine FSM next state and output equations**