THE UNIVERSITY OF TEXAS AT DALLAS

CS4341 Digital Logic & Computer Design

Lecture Notes 16

Omar Hamdy
Assistant Professor
Department of Computer Science

Review: State and Output Encoding

- The state as well as output encodings were chosen randomly and could be selected differently
- Different encoding will result in different circuit design (hence different complexities), but same circuit behavior
- There are two types of encoding:
  - `Binary encoding`: each state or output is represented as a binary number
  - `One-hot-encoding`: each state is represented by its own memory bit.
- `One-hot-encoding` usually means simpler circuit, but more memory bits

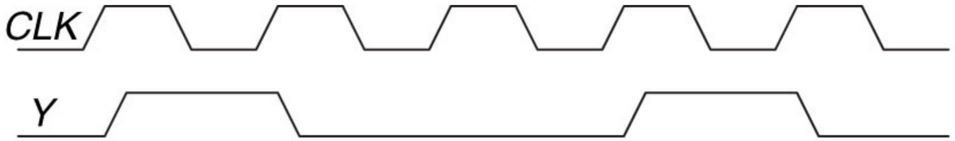
Output	Encoding $L_{1:0}$	
Green	1	1
Yellow	1	0
Red	0	0

Output	Encoding $L_{1:0}$	
Green	1	0
Yellow	1	1
Red	0	1

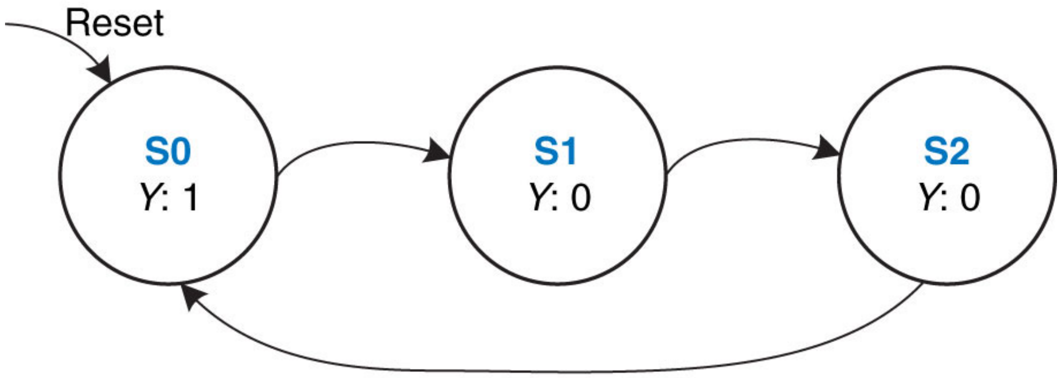
Output	Encoding $L_{1:0}$	
Green	0	1
Yellow	0	0
Red	1	1

Example: Divide-by-N Counter

- This special circuit has no inputs and one output.
- The output Y is High (1) for one clock cycle out of every N



(a)

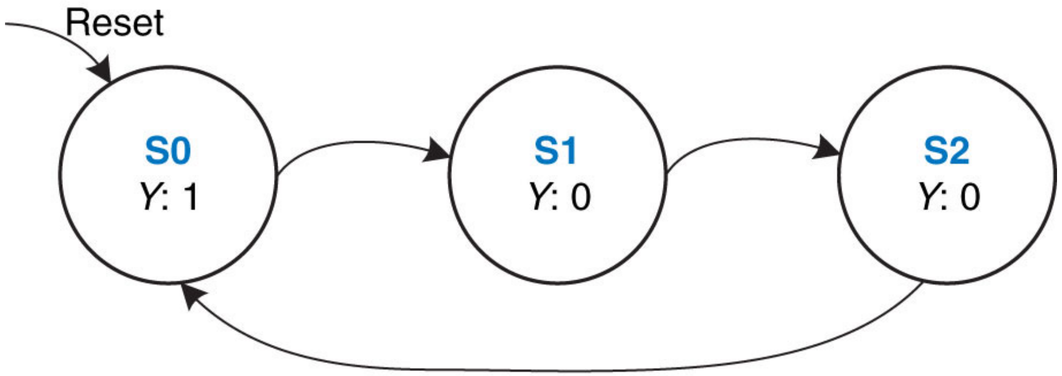


(b)

- Design the circuit using `binary` and `one-hot` encodings.

Example: State Transition & Output Tables

- The following represents the state transitions and the output tables without any encodings .



Current Status	Next Status
S0	S1
S1	S2
S2	S0

Current Status	Output
S0	1
S1	0
S2	0

Example: Binary and One-hot Encodings

- Binary encoding represents each state as a binary number.
- One-hot encoding represents each state using one memory bit

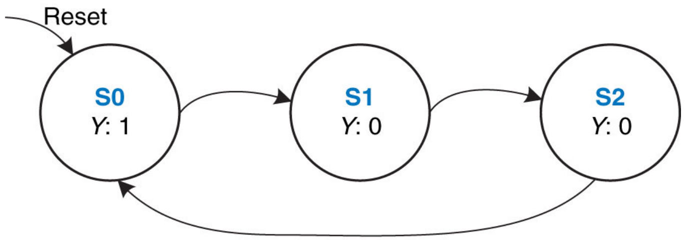
	Binary Encoding	
State	S_1	S_0
S0	0	1
S1	1	0
S2	0	0

	One-Hot Encoding		
State	S_2	S_1	S_0
S0	0	0	1
S1	0	1	0
S2	1	0	0

- Note that one-hot encoding required one additional memory bit

Example: State Transition Truth Tables

	Binary Encoding	
State	S_1	S_0
S0	0	1
S1	1	0
S2	0	0



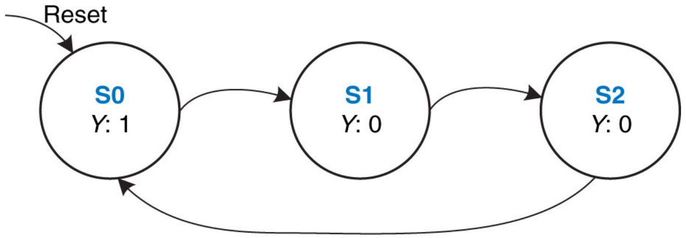
	One-Hot Encoding		
State	S_2	S_1	S_0
S0	0	0	1
S1	0	1	0
S2	1	0	0

Current State		Next State	
S_1	S_0	S'_1	S'_0
0	1	1	0
1	0	0	0
0	0	0	1

Current State			Next State		
S_2	S_1	S_0	S'_2	S'_1	S'_0
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	0	0	1

Example: Output Truth Table

	Binary Encoding	
State	S_1	S_0
S0	0	1
S1	1	0
S2	0	0



	One-Hot Encoding		
State	S_2	S_1	S_0
S0	0	0	1
S1	0	1	0
S2	1	0	0

Current State		Output
S_1	S_0	Y
0	0	0
0	1	1
1	0	0

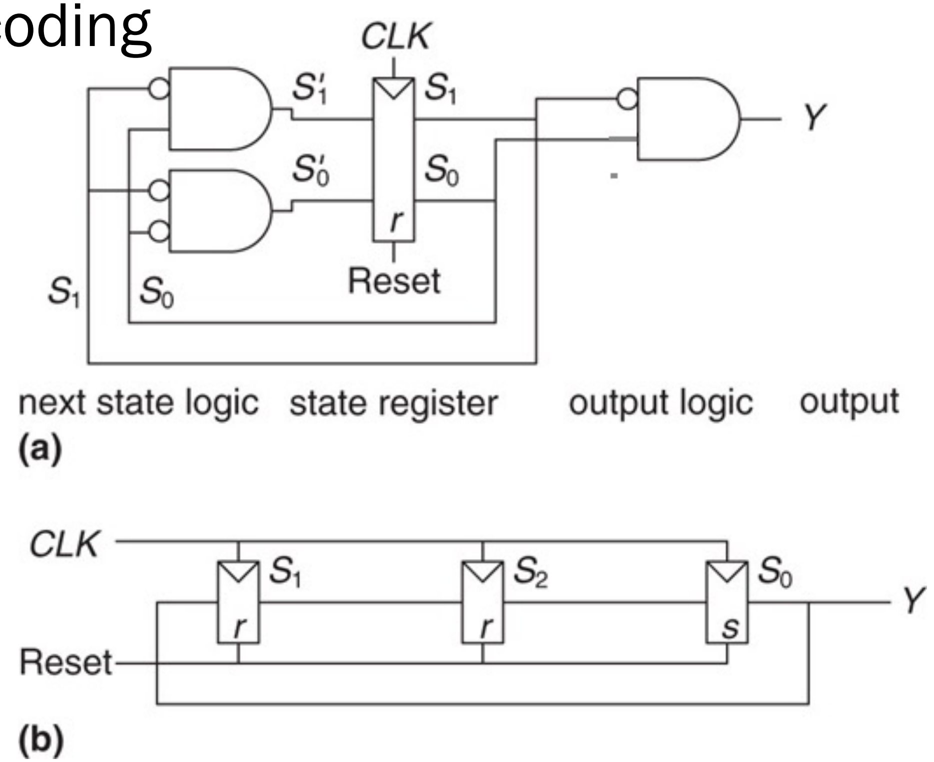
Current State			Output
S_2	S_1	S_0	Y
0	0	1	1
0	1	0	0
1	0	0	0

Example: Next State and Output Logic

Current State		Next State	
S_1	S_0	S'_1	S'_0
0	1	1	0
1	0	0	0
0	0	0	1

Current State		Output
S_1	S_0	Y
0	0	0
0	1	1
1	0	0

- Using `binary` encoding
  - Next State Logic:
    - $S'_1 = \overline{S_1} S_0$
    - $S'_0 = \overline{S_1}\,\overline{S_0}$
  - Output Logic:
    - $Y = \overline{S_1} S_0$



Current State			Next State		
S_2	S_1	S_0	S'_2	S'_1	S'_0
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	0	0	1

Current State			Output
S_2	S_1	S_0	Y
0	0	1	1
0	1	0	0
1	0	0	0

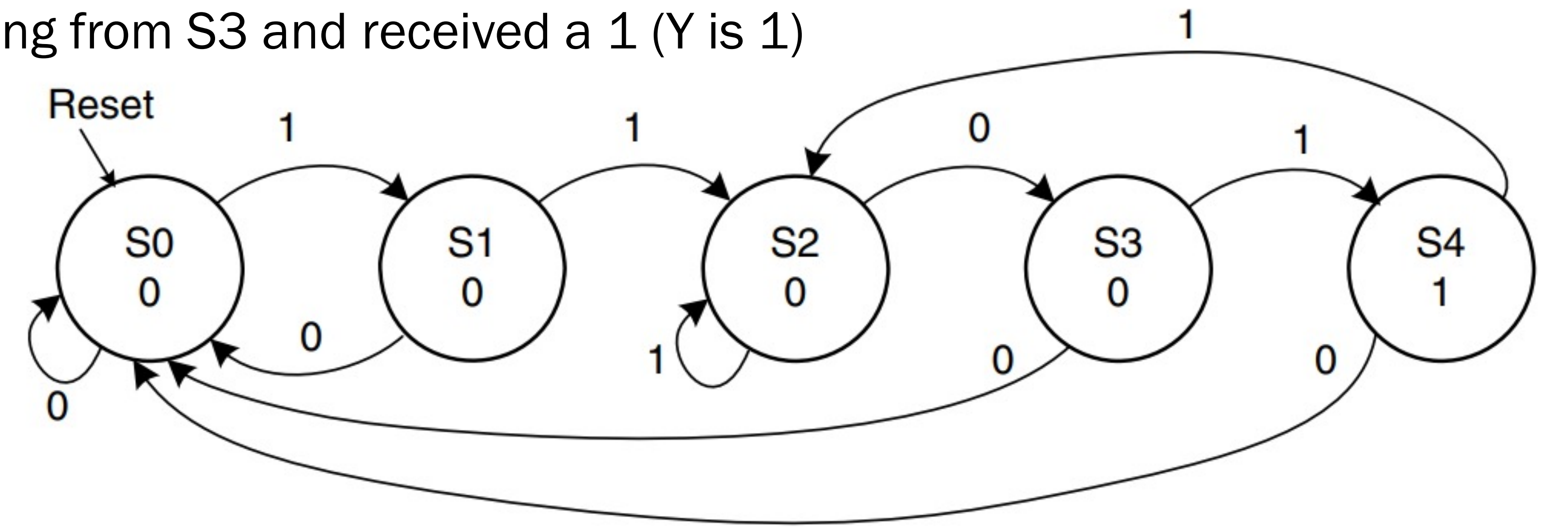
- Using `one-hot` encoding
  - Next State Logic:
    - $S'_2 = S_1$
    - $S'_1 = S_0$
    - $S'_0 = S_2$
  - Output Logic:
    - $Y = S_0$

Example: Pattern Detector

- Design an FSM (both Moore and Mealy) that can detect an input pattern. The FSM has the following details:
  - The FSM attempts to detect the input pattern 1101
  - Has 1 input A and one output Y
  - If a 1101 input pattern is detected, Y produces 1. Otherwise, Y is 0
  - Assume a reset state of input 0

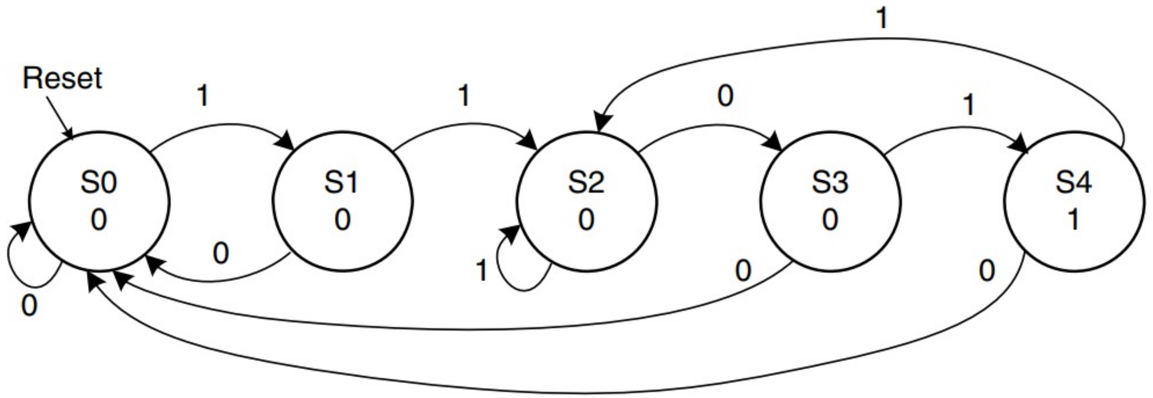
Pattern Detector – State Transitioning Diagram

- In Moore FSM, each state describes the output value Y inside.
- What are the possible input patterns?
  - S0 (Reset): Input 0, and Y is 0
  - S1: Coming from S0 and received a 1 (Y is 0)
  - S2: Coming from S1 and received a 1 (Y is 0)
  - S3: Coming from S2 and received a 0 (Y is 0)
  - S4: Coming from S3 and received a 1 (Y is 1)



Pattern Detector: State Transition Truth Table

- For Moore FSM, we need 3 memory bits to represent the 5 states



Current State S	A	Next State S'
S0	0	S0
S0	1	S1
S1	0	S0
S1	1	S2
S2	0	S3
S2	1	S2
S3	0	S0
S3	1	S4
S4	0	S0
S4	1	S2

Pattern Detector: State Transition Truth Table

- Formally representing the Moore FSM with state encoding

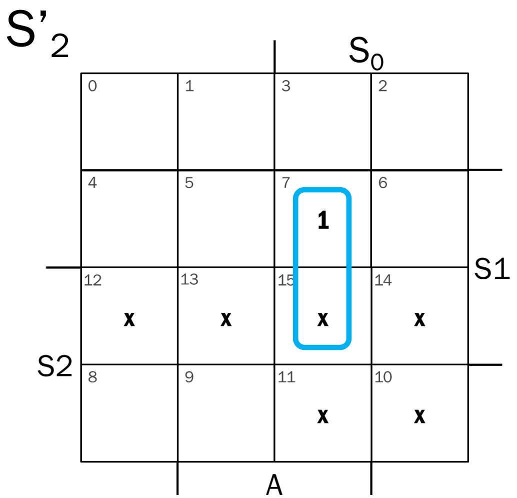
State	Encoding $S_{2:0}$		
S0	0	0	0
S1	0	0	1
S2	0	1	0
S3	0	1	1
S4	1	0	0

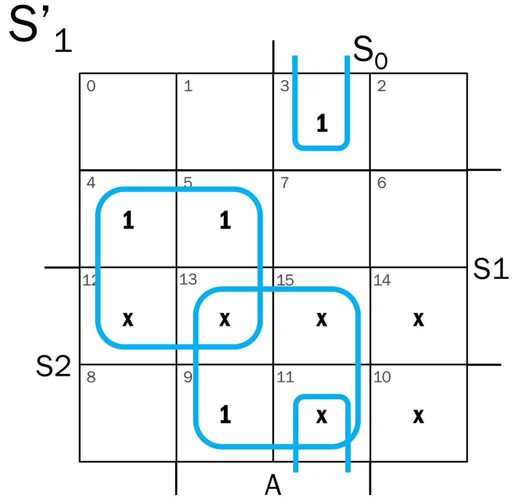
Current State S				Next State S'		
S2	S1	S0	A	S'2	S'1	S'0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	0
0	1	1	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	0

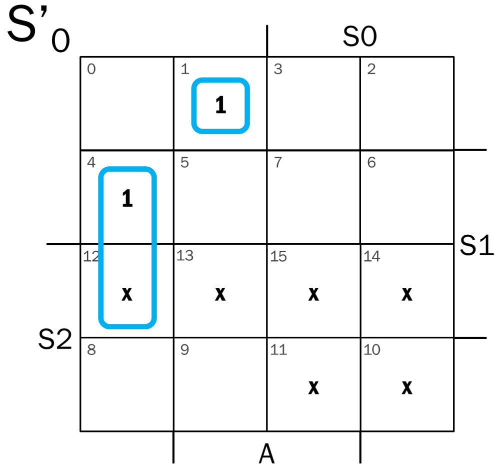
Pattern Detector: State Transition Expression

- Simplify next state logic using K-Map. Consider all the don't care next states

Current State S				Next State S'		
S2	S1	S0	A	S'2	S'1	S'0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	0
0	1	1	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	0
1	0	1	0	x	x	X
1	0	1	1	x	x	X
1	1	0	0	x	x	X
1	1	0	1	x	x	X
1	1	1	0	x	x	x
1	1	1	1	x	x	x







$$S'_2 = S_1 S_0 A$$

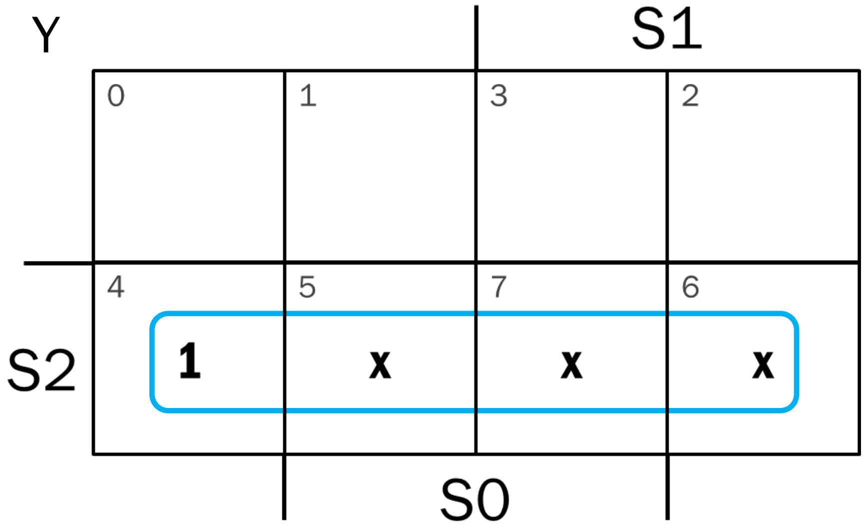
$$S'_1 = S_1\bar{S}_0 + S_2 A + \bar{S}_1 S_0 A$$

$$S'_0 = S_1\bar{S}_0\bar{A} + \bar{S}_2\bar{S}_1\bar{S}_0 A$$

Pattern Detector: Output Truth Table

- The Moore FSM output truth table and simplification K-map are as follows.

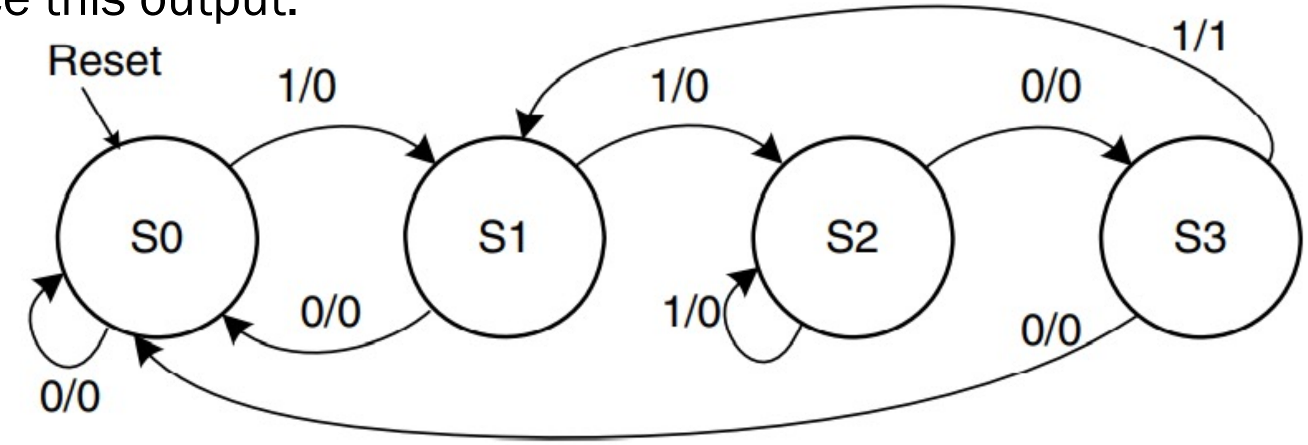
S2	S1	S0	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	x
1	1	0	x
1	1	1	x



$$Y = S_2$$

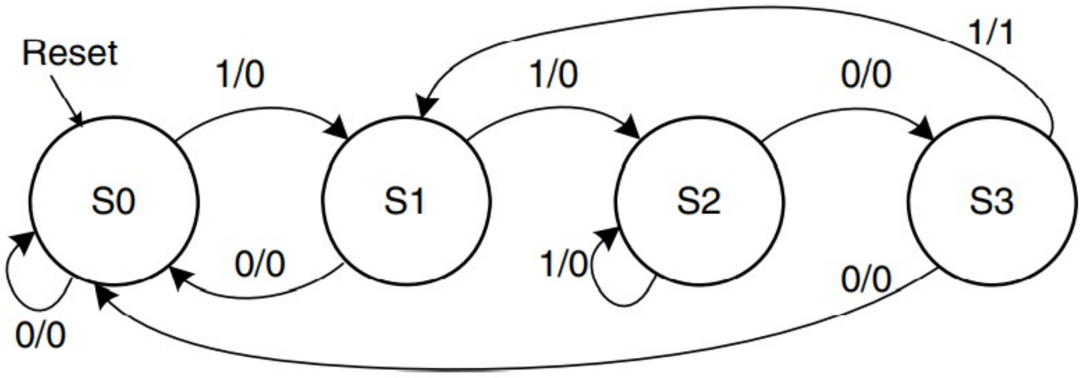
Pattern Detector – State Transitioning Diagram

- In Mealy FSM, the output is determined at the transitioning from each state and based on the input value.
  - S0 (Reset): Input 0, and Y is 0
  - When input 1 is received, state transitions to S1 and output is 0
  - When input 1 is received, state transitions to S2 and output is 0
  - When input 0 is received, state transitions to S3 and output is 0
  - When input 1 is received, state transitions to **S1** and output is 1 (no need for a new state just to produce this output.



Pattern Detector: State Transition Truth Table

- For Mealy FSM, we need 2 memory bits to represent the 4 states



Current State S	A	Next State S'
S0	0	S0
S0	1	S1
S1	0	S0
S1	1	S2
S2	0	S3
S2	1	S2
S3	0	S0
S3	1	S1

Pattern Detector: Truth Table

- Formally representing the Mealy FSM with state encoding

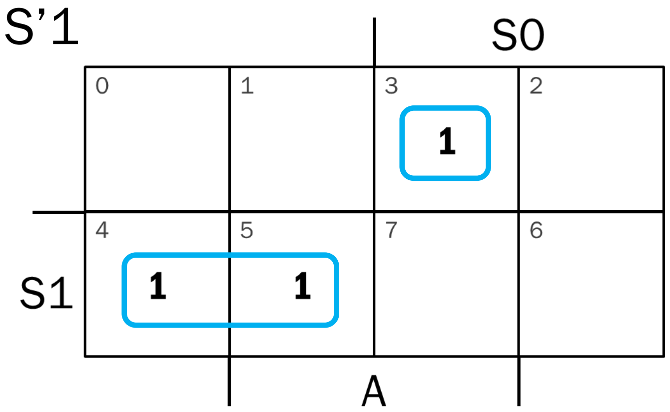
State	Encoding $S_{1:0}$	
S0	0	0
S1	0	1
S2	1	0
S3	1	1

Current State S			Next State S’		
S1	S0	A	S’1	S’0	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1

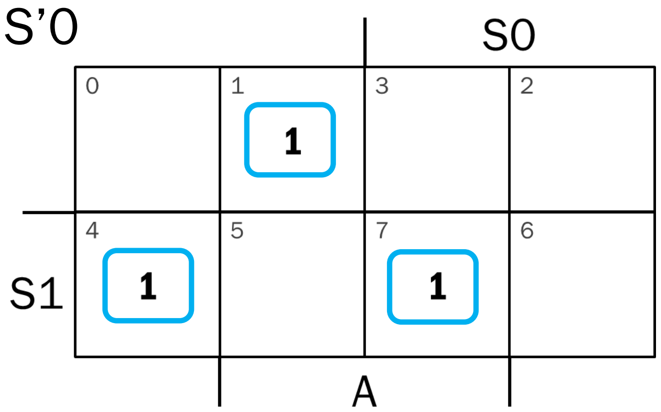
Pattern Detector: Simplified Expressions

- Simplify next state & output logic using K-Map.

Current State S			Next State S’		
S1	S0	A	S’1	S’0	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1



$$S'_1 = S_1\bar{S}_0 + \bar{S}_1 S_0 A$$



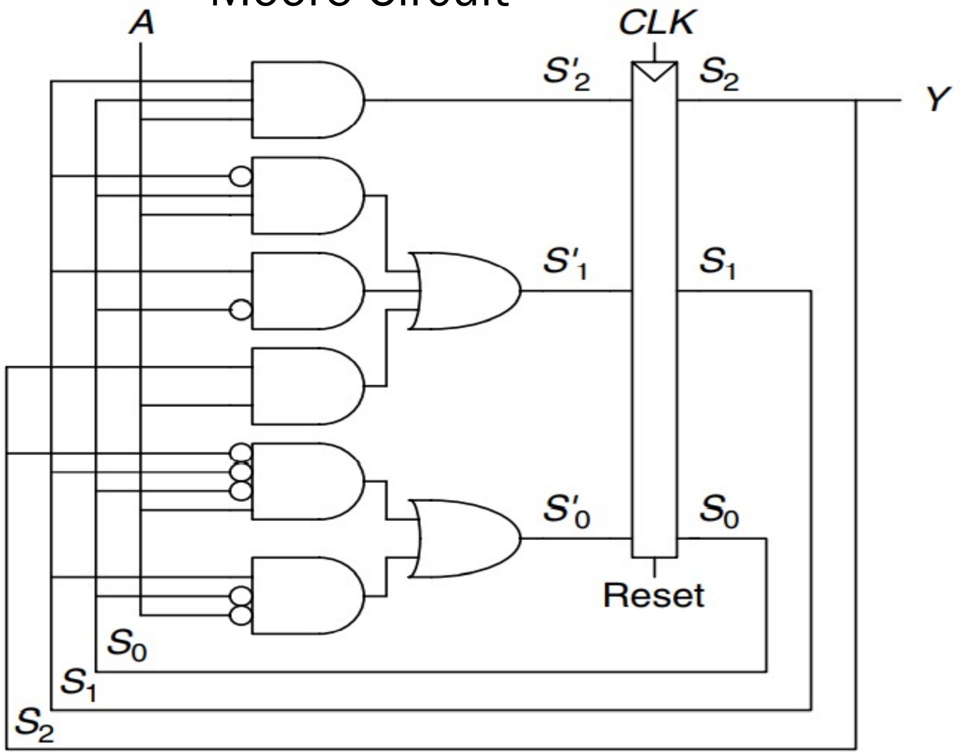
$$S'_0 = S_1\bar{S}_0\bar{A} + \bar{S}_1\bar{S}_0 A + S_1 S_0 A$$

$$Y = S_1 S_0 A$$

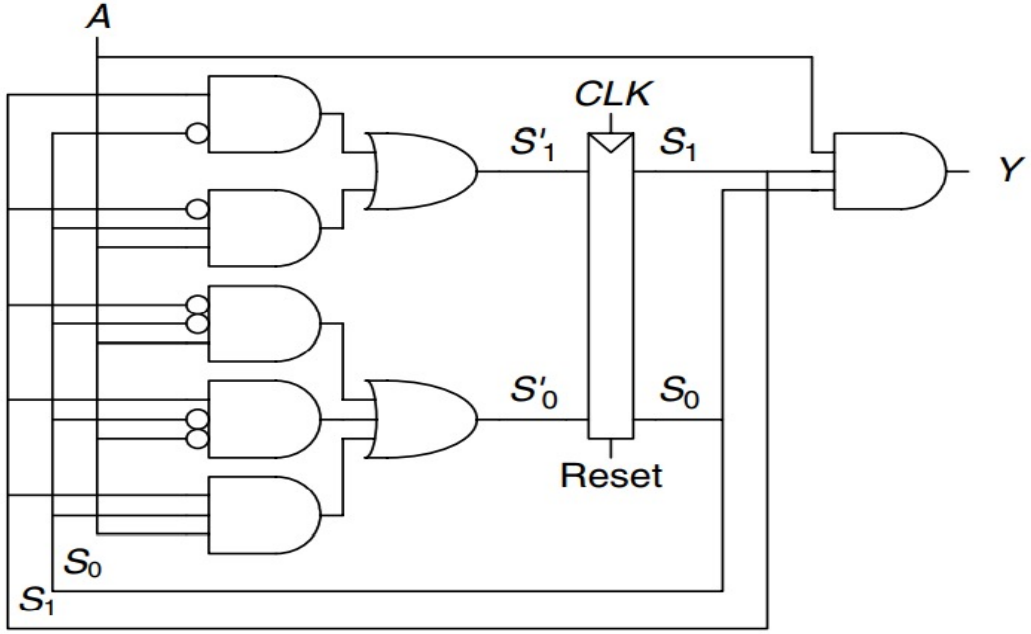
Pattern Detector: Circuit Design

- Using the next state and output simplified expressions for Moor and Mealy, we get the following two designs

Moore Circuit

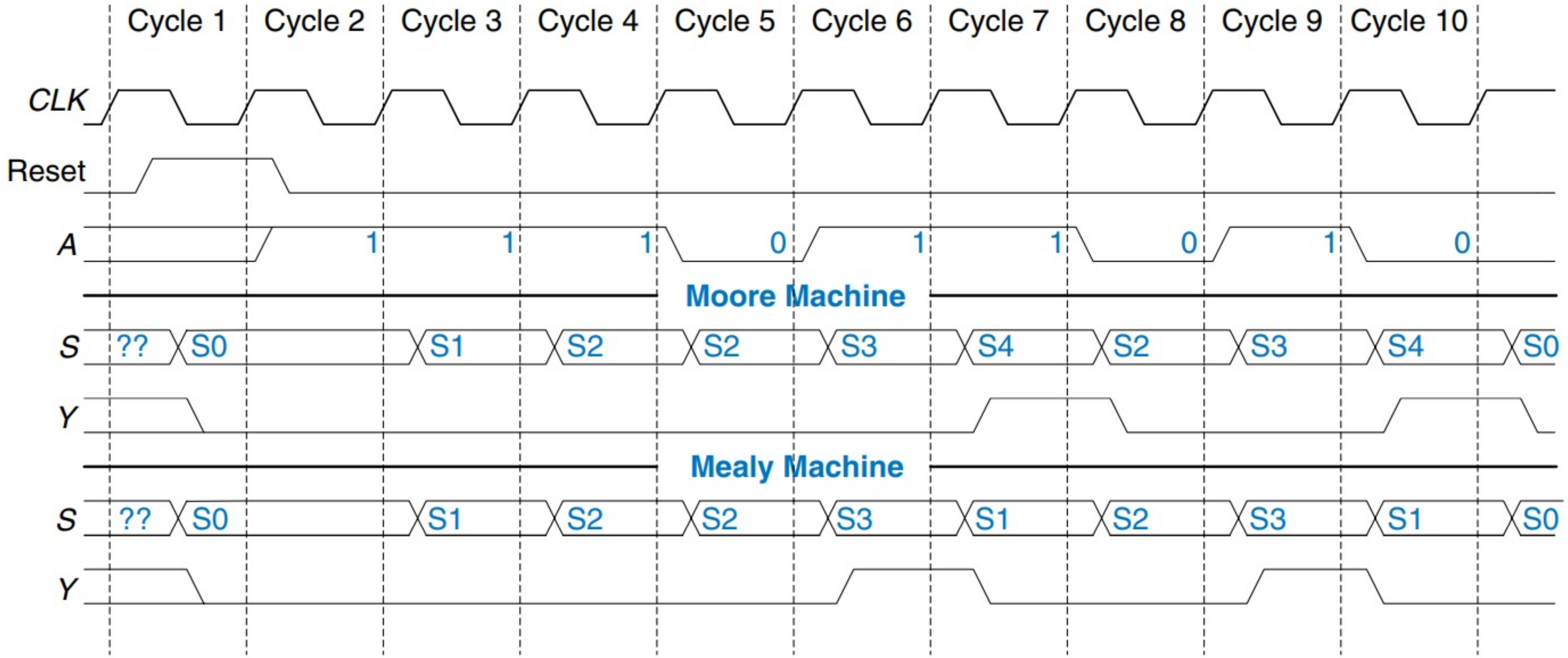


Mealy Circuit



Pattern Detector: Waveform Analysis

- The following is how the waveform for both Moore and Mealy implementations. Note how the output for Mealy machine is shifted



To Do List

- Review lecture notes
- Signup for EDA Playground account
- Continue working on assignment 2