SER 232

Computer Systems Fundamentals I

Topics

- Introduction: Finite State Machines

Finite State Machine

- Short: FSM
- A model defining:
  - A **finite** set of **states** that a system can be in
  - The inputs required to **transition** from one state to another (also called **events**)
  - The **output** values each state or alternatively during transitions
- FSM needs memory (a register) to store the current system state
  - FSM circuits are sequential circuits

FSM Example

- Example: Microwave (simplified)
- What states can a microwave be in?
  - E.g.: On, off (door open), off (door closed), (power %, defrosting, different heating types, …)
- What **events** lead to entering or *transitioning* between those states?
  - On: Pressing the **start button** with **door closed**; …
  - Off: Pressing the **stop button**; **opening the door** while on; …
- What “output” does the microwave have for each state?
  - On: Microwaves are generated
  - Off: No microwaves are generated

Moore Machine

- The FSMs used in this course are all Moore FSMs
  - Output only dependent on state
  - Transitions dependent only on inputs

Mealy Machine

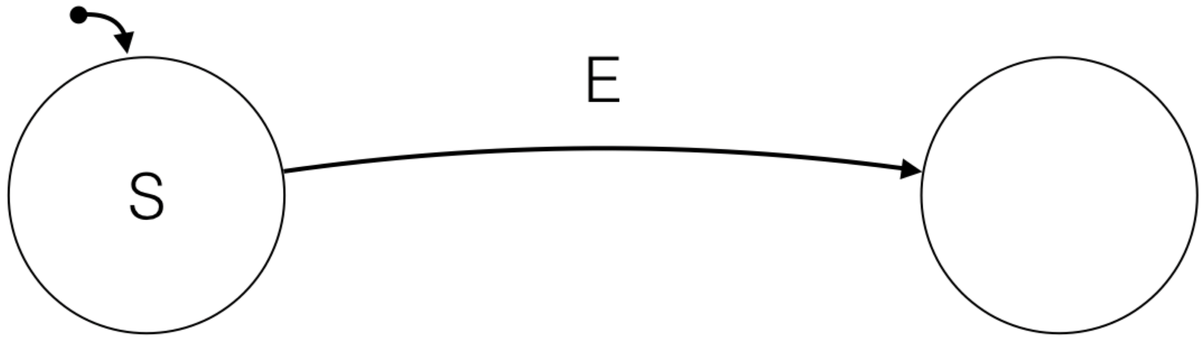
- Output depends on current state AND current input
  - Transitions define both input and output
- We will focus on Moore Machines
  - More information about Moore and Mealy FSMs can be found in *Digital Design* book page 360

Finite State Machine

- Goal: Create FSM as a circuit
- In order to get there, we have to use/do the following:
  1. Create a representation of the system behavior: State diagram
  2. Define binary numbers that each represent a state of the system
  3. Convert state diagram into a truth table
  4. Convert truth table into Boolean equation
  5. Convert Boolean equation into FSM circuit

State Diagram

- FSM can be represented by a state diagram
  - Terminology



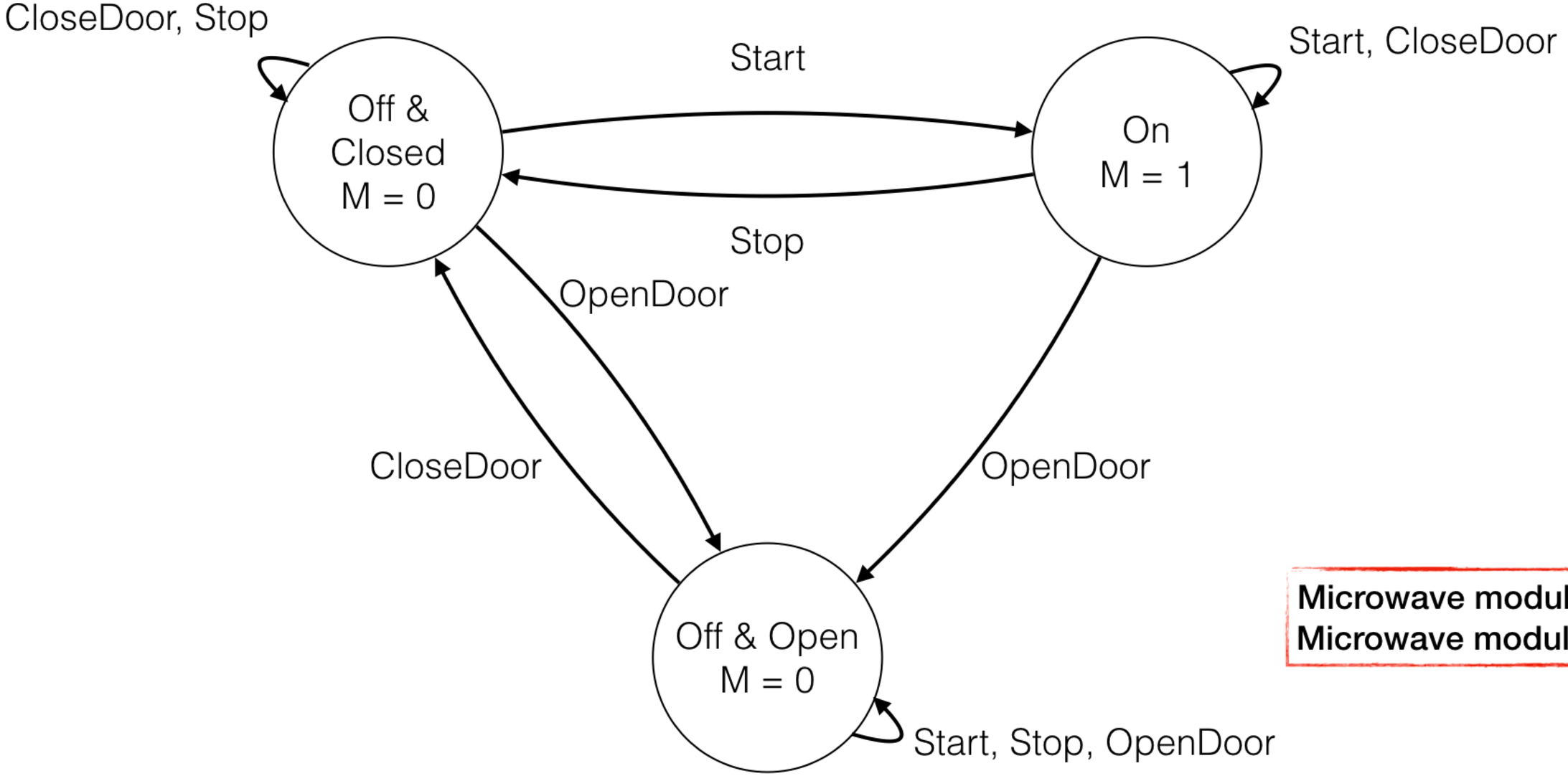
- S: State, ↗ : Transition, E: Event, ↷ : Initial State

FSM Example

- Example: Microwave (simplified)
- What states can a microwave be in?
  - E.g.: On, off (door open), off (door closed), (power %, defrosting, different heating types, …)
- What **events** lead to entering or *transitioning* between those states?
  - On: Pressing the **start button** with **door closed**; …
  - Off: Pressing the **stop button**; **opening the door** while on; …
- What "output" does the microwave have for each state?
  - On: Microwaves are generated
  - Off: No microwaves are generated

Events:
- Start
- Stop
- Close door
- Open door

Microwave FSM Example



Microwave module on: M =1
Microwave module off: M =0

SER 232

Computer Systems Fundamentals I

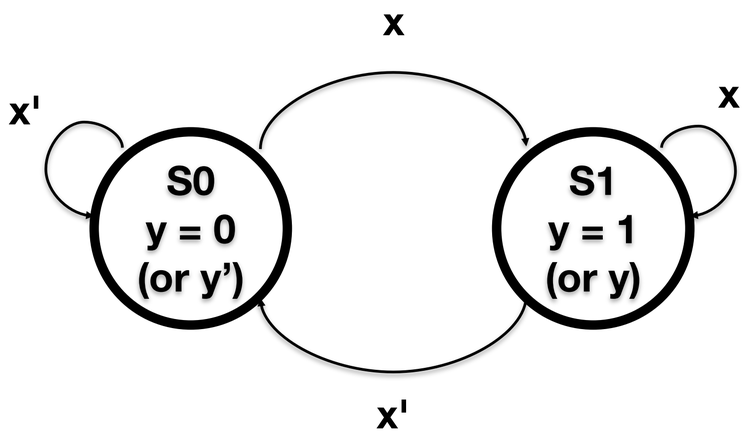
Topics

- Finite State Machines

State Diagram

- A diagram commonly used to show all possible states and the transitions between them

Inputs: x
Outputs: y



Controller Combinational Logic

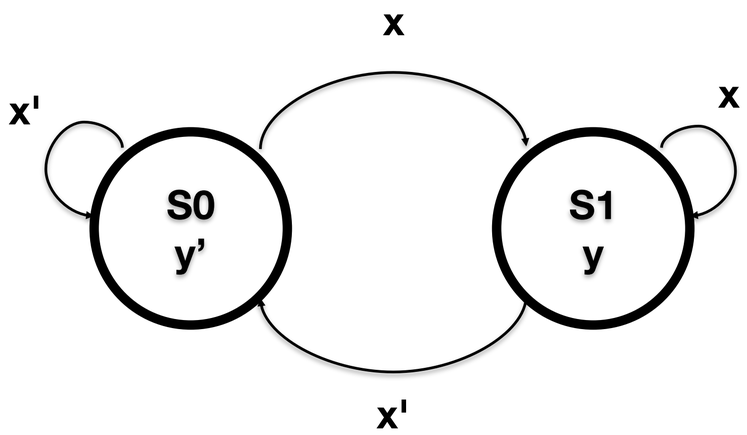
- Truth table
  - Inputs:
    - Current state (from Q output of state register)
    - FSM inputs
  - Outputs:
    - FSM outputs
    - Next state (fed into D of state register)

- How do we represent state S0 and S1 in the truth table?
  - We need to encode them to a binary number!
  - 2 states = 1 bit necessary (2^n relation: 2^1 = 2)

State	Encoding
S0	0
S1	1

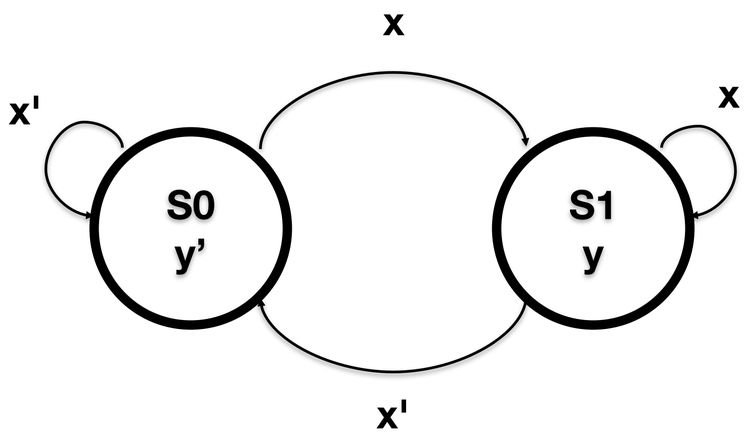
c	x	y	n
0	0		
0	1		
1	0		
1	1		

Inputs: x
Outputs: y



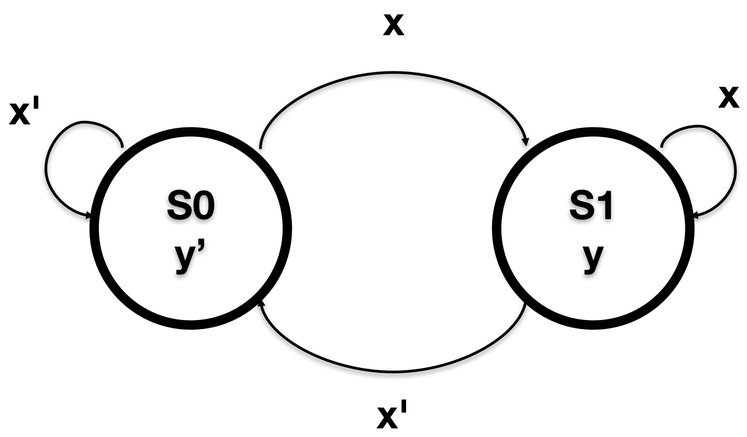
c	x	y	n
0	0	0	
0	1	0	
1	0		
1	1		

Inputs: x
Outputs: y



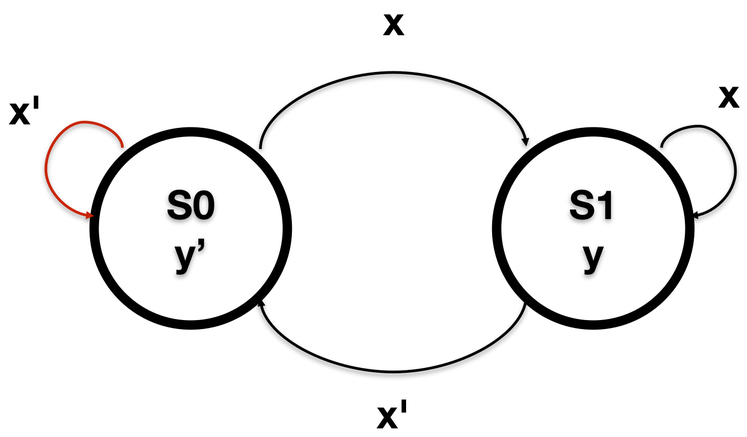
c	x	y	n
0	0	0	
0	1	0	
1	0	1	
1	1	1	

Inputs: x
Outputs: y



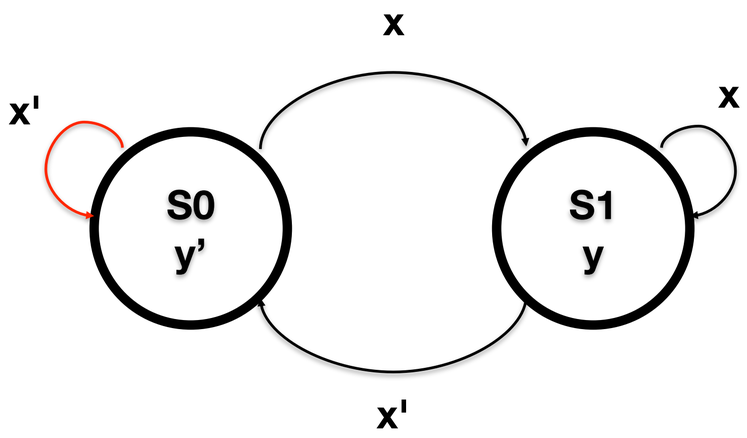
c	x	y	n
0	0	0	
0	1	0	
1	0	1	
1	1	1	

Inputs: x
Outputs: y



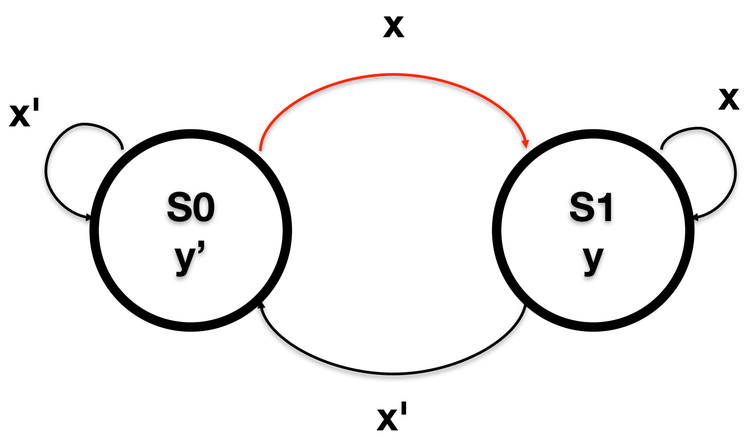
c	x	y	n
0	0	0	0
0	1	0	
1	0	1	
1	1	1	

Inputs: x
Outputs: y



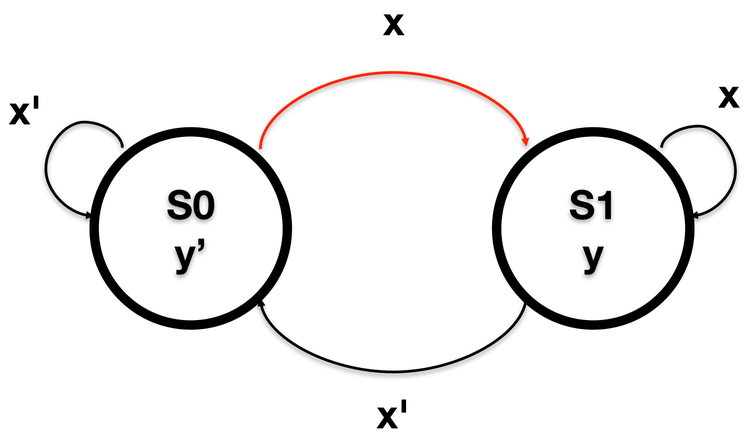
c	x	y	n
0	0	0	0
0	1	0	
1	0	1	
1	1	1	

Inputs: x
Outputs: y



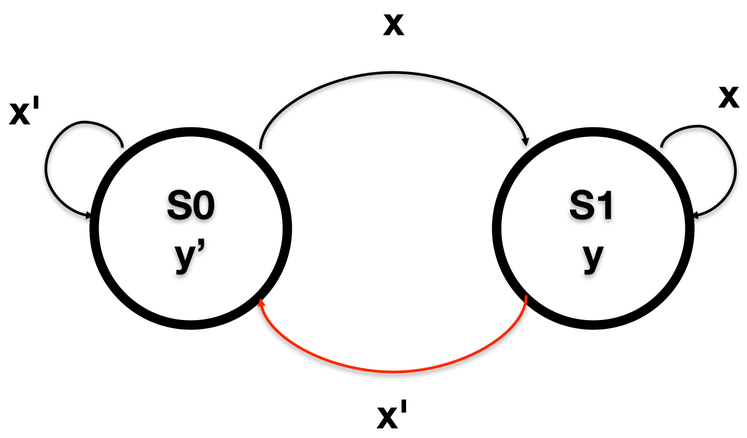
c	x	y	n
0	0	0	0
0	1	0	1
1	0	1	
1	1	1	

Inputs: x
Outputs: y



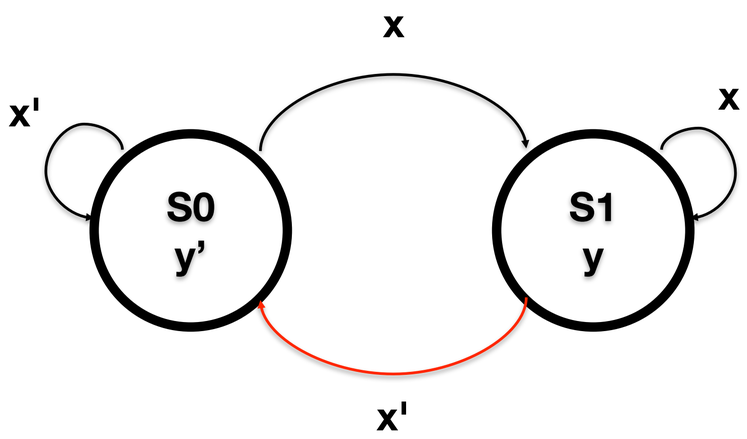
c	x	y	n
0	0	0	0
0	1	0	1
1	0	1	
1	1	1	

Inputs: x
Outputs: y



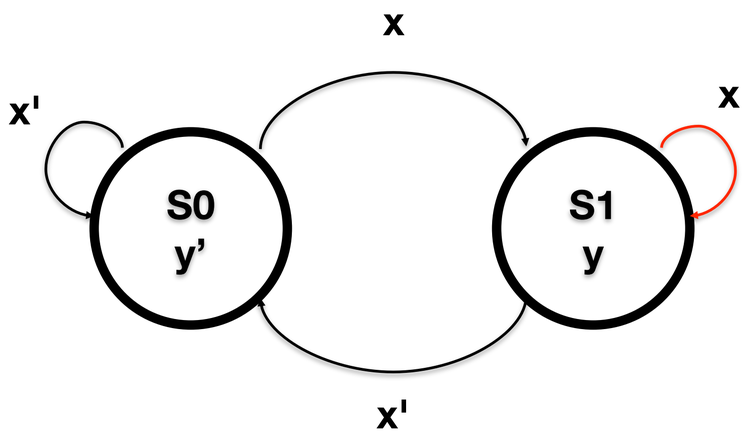
c	x	y	n
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	

Inputs: x
Outputs: y



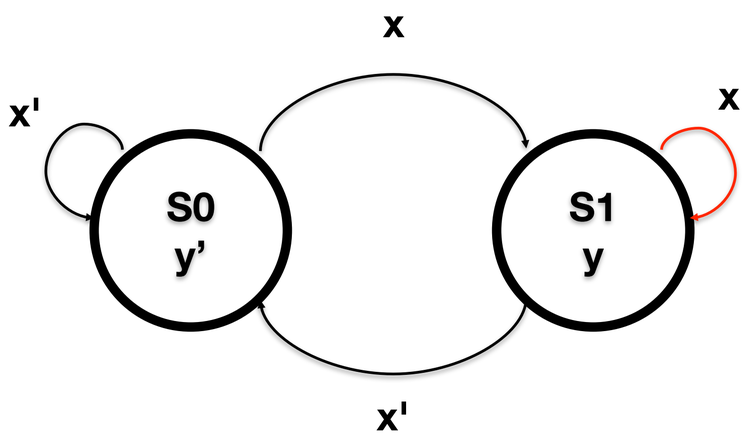
c	x	y	n
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	

Inputs: x
Outputs: y



c	x	y	n
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

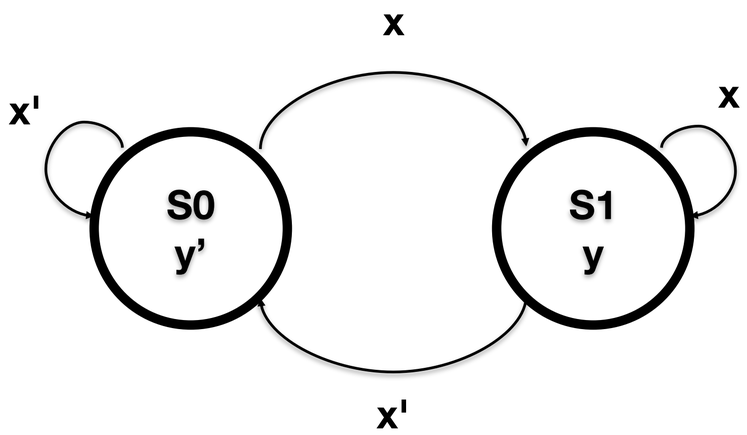
Inputs: x
Outputs: y



c	x	y	n	
0	0	0	0	
0	1	0	1	c’x
1	0	1	0	
1	1	1	1	cx

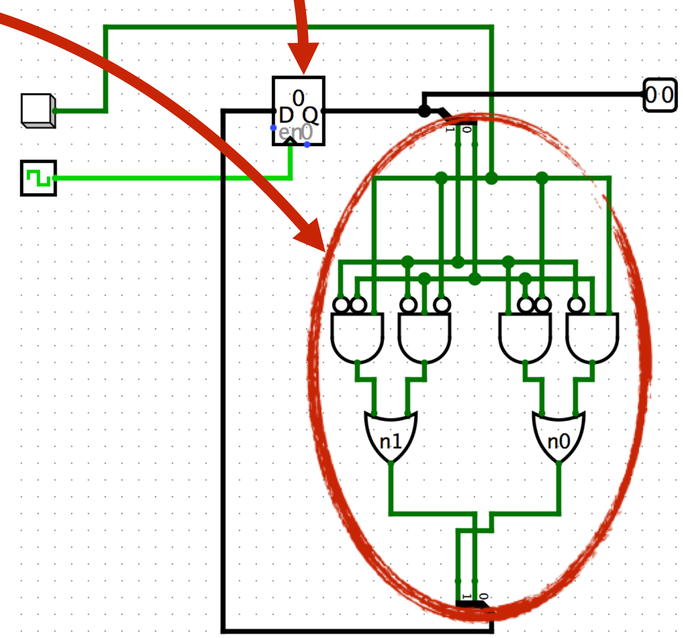
n = c’x + cx

Inputs: x
Outputs: y



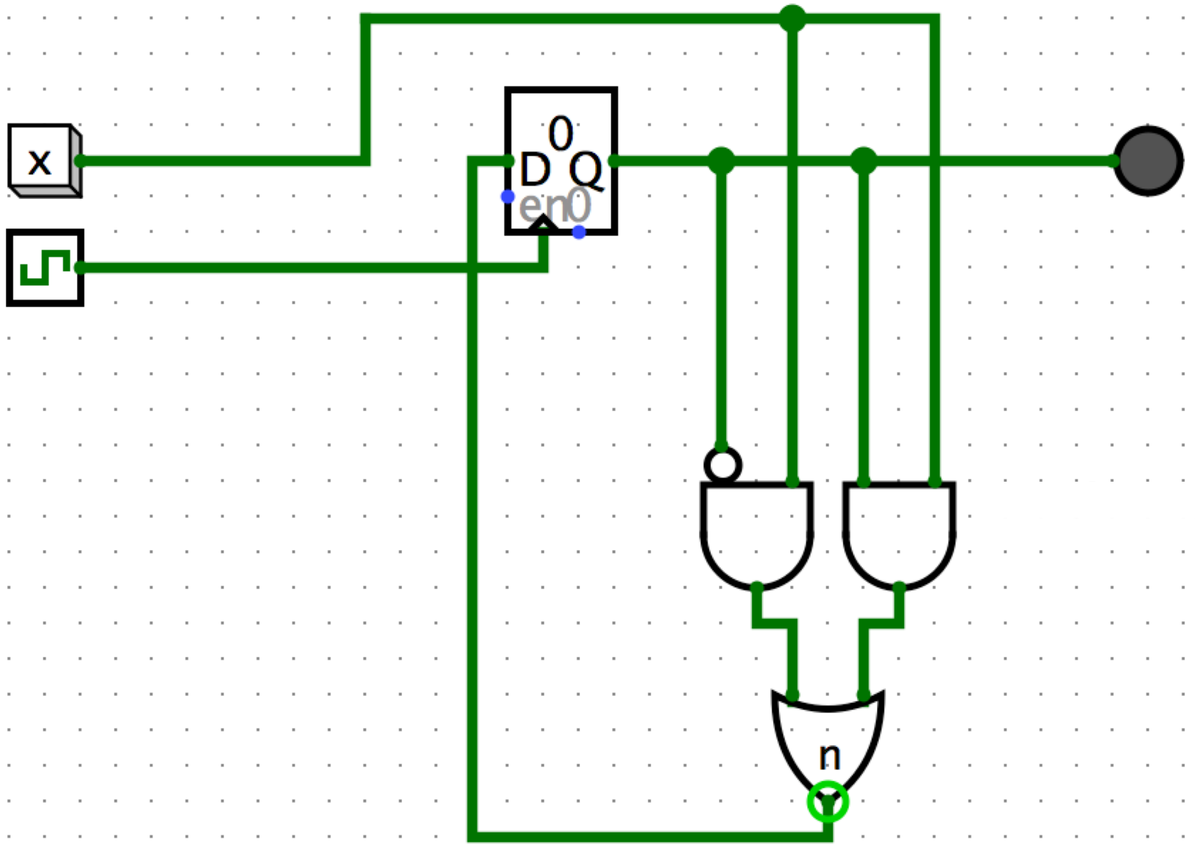
Controller for FSM

- State register
  - Holds current state number
- Combinational Logic
  - Output updates state register



FSM in Logisim

- Creating the circuit in Logisim: n = c'x + cx



From FSM to Circuit

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

Your Turn

- System has 3 states A, B, C
  - If button **x** is pressed (x=1 or x), go to next state (A->B->C->A->…)
  - If button **x** is not pressed (x=0 or x'), stay in current state
  - Output the encoded value of the state

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

Step 1

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

Step 2

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

Step 3

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

Step 4

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

Step 5

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

SER 232

Computer Systems Fundamentals I

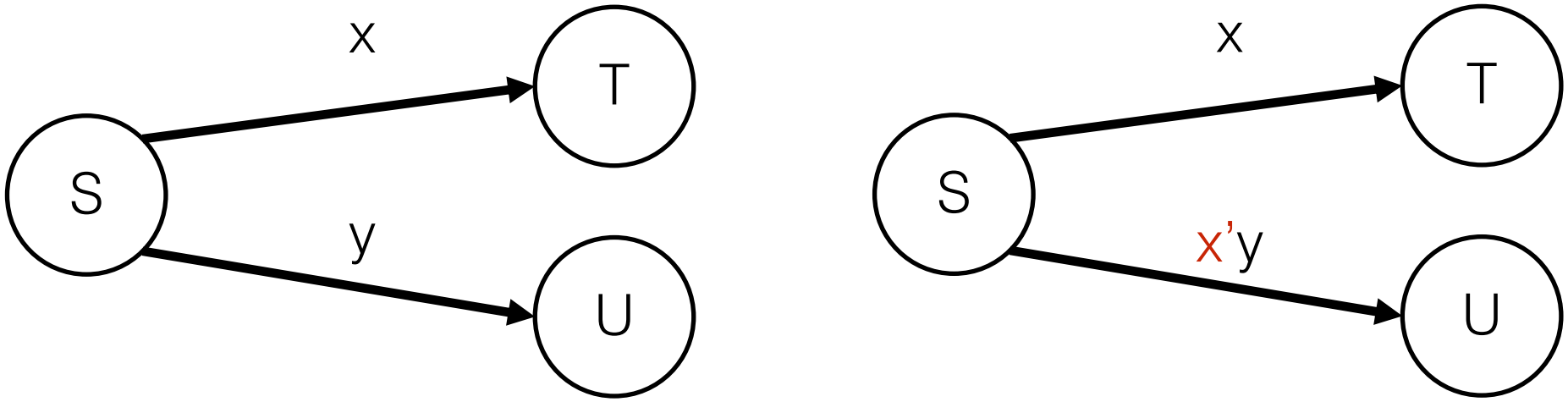
Dr. Heinrichs

Topics

- Finite State Machines, Considerations

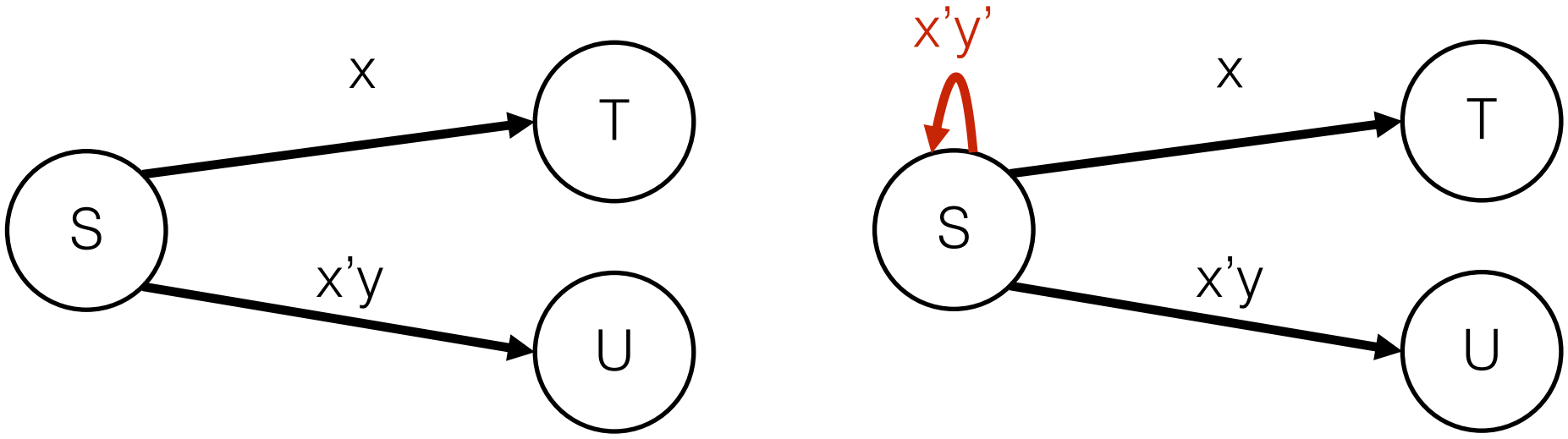
Capturing FSMs

- Non-exclusive transitions
  - *What is the next state for* `xy=11` *?*



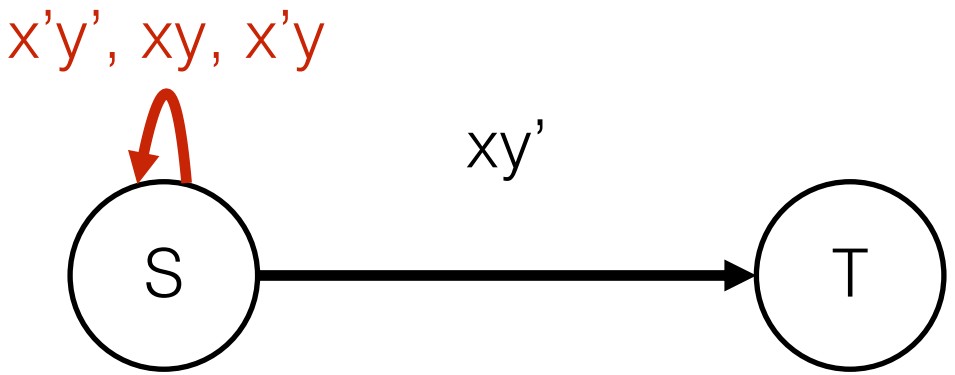
Capturing FSMs

- Incomplete transitions
  - *What is the next state for* `xy=00` *?*



Capturing FSMs

- Only if x=1 and y=0 the transition to T triggers



SER 232

Computer Systems Fundamentals I

Topics

- Finite State Machines, Larger Example

FSM

- 3 states (**A, B, C**); 2 inputs (**x, y**); 1 output (**f**)
  - If input **x = 1**, go to next state (A->B->C->A->…)
  - If input **y = 1**, go to previous state (A->C->B->A->…)
  - If **x = 1** and **y = 1** go to state C
  - If **x = 0** and **y = 0** stay in current state
  - Output **f** will be the following for each state:
    - A: 0b10
    - B: 0b11
    - C: 0b00

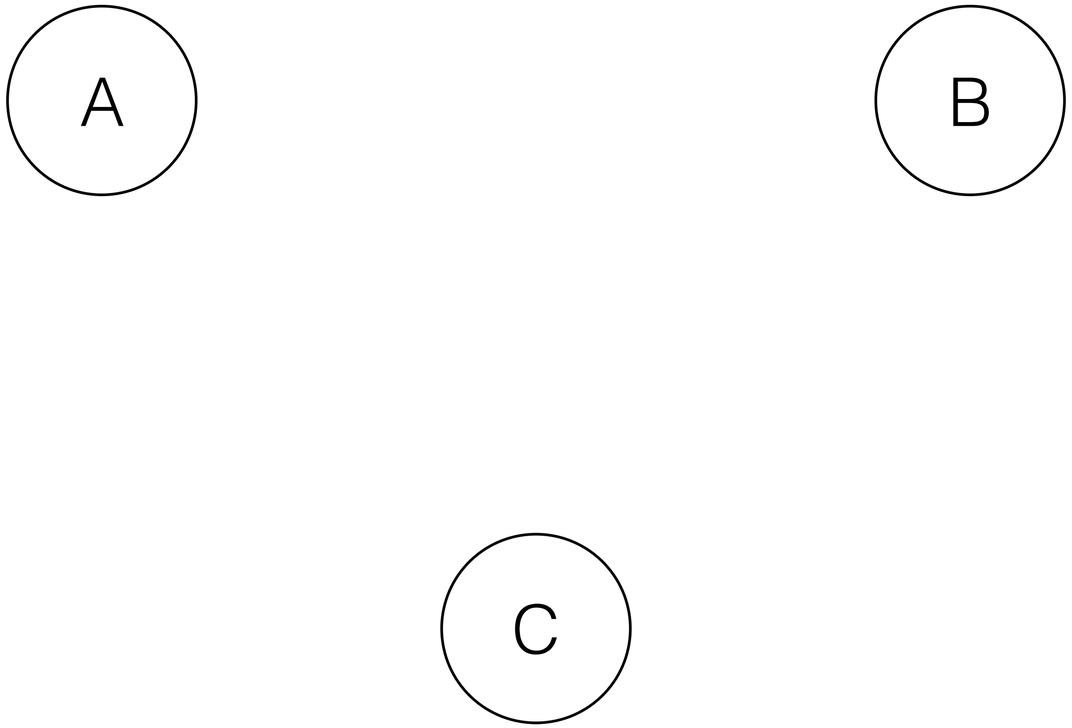
State	Encoded Binary Value
A	00
B	11
C	01

From FSM to Circuit

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

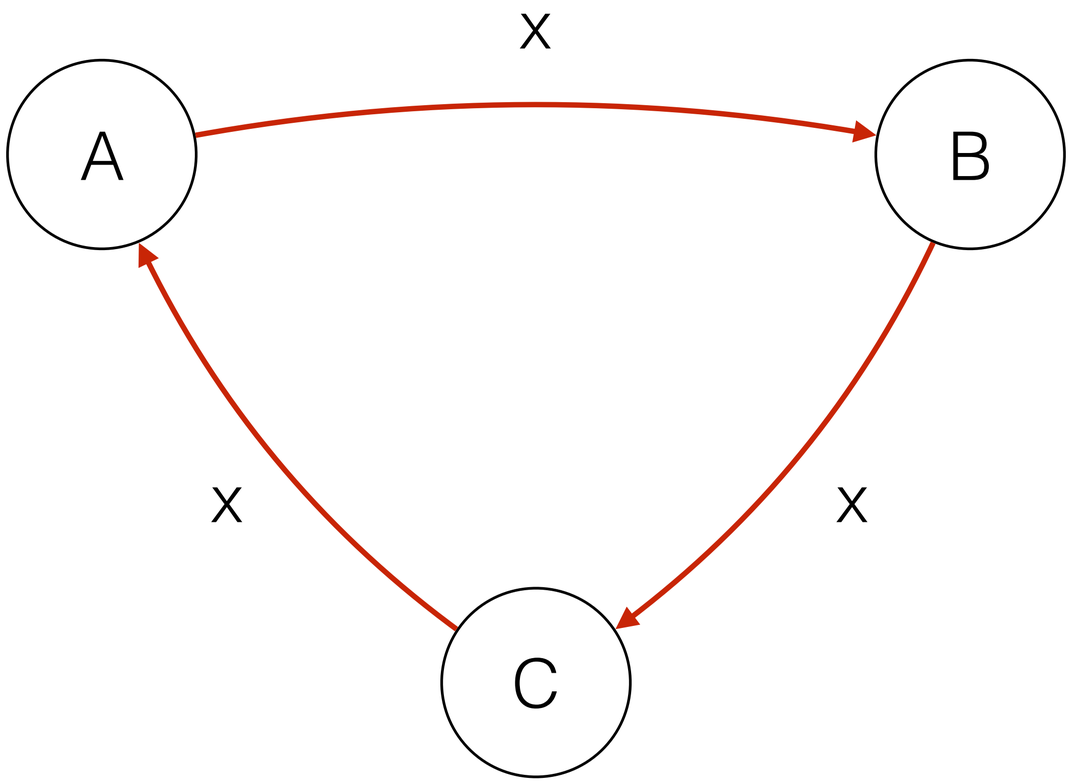
FSM

- 3 states A, B, C



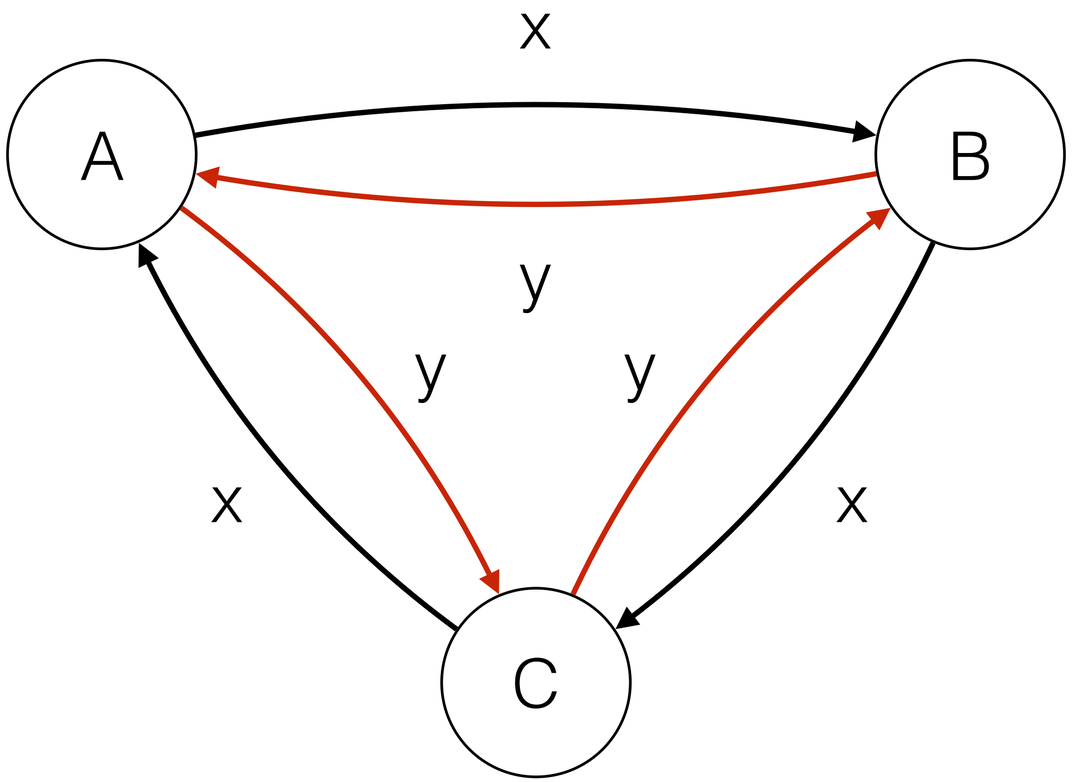
FSM

- If input **x = 1**, go to next state (A->B->C->A->…)



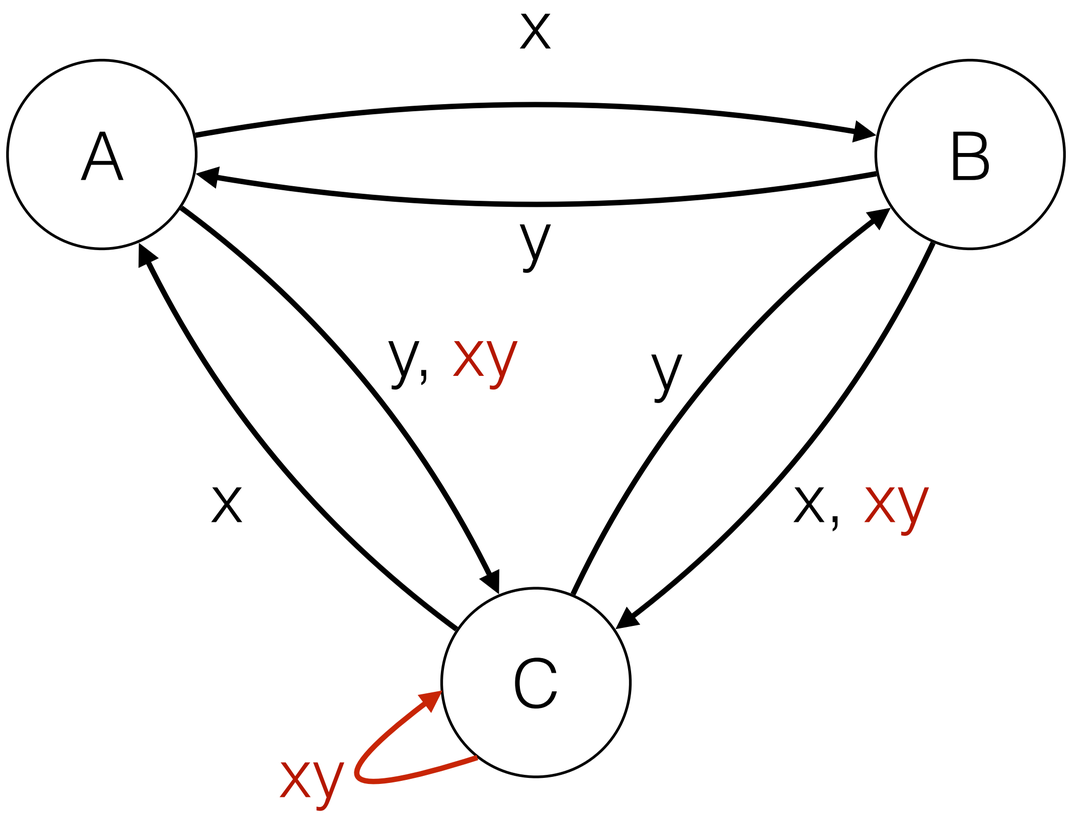
FSM

- If input **y = 1**, go to previous state (A->C->B->A->…)



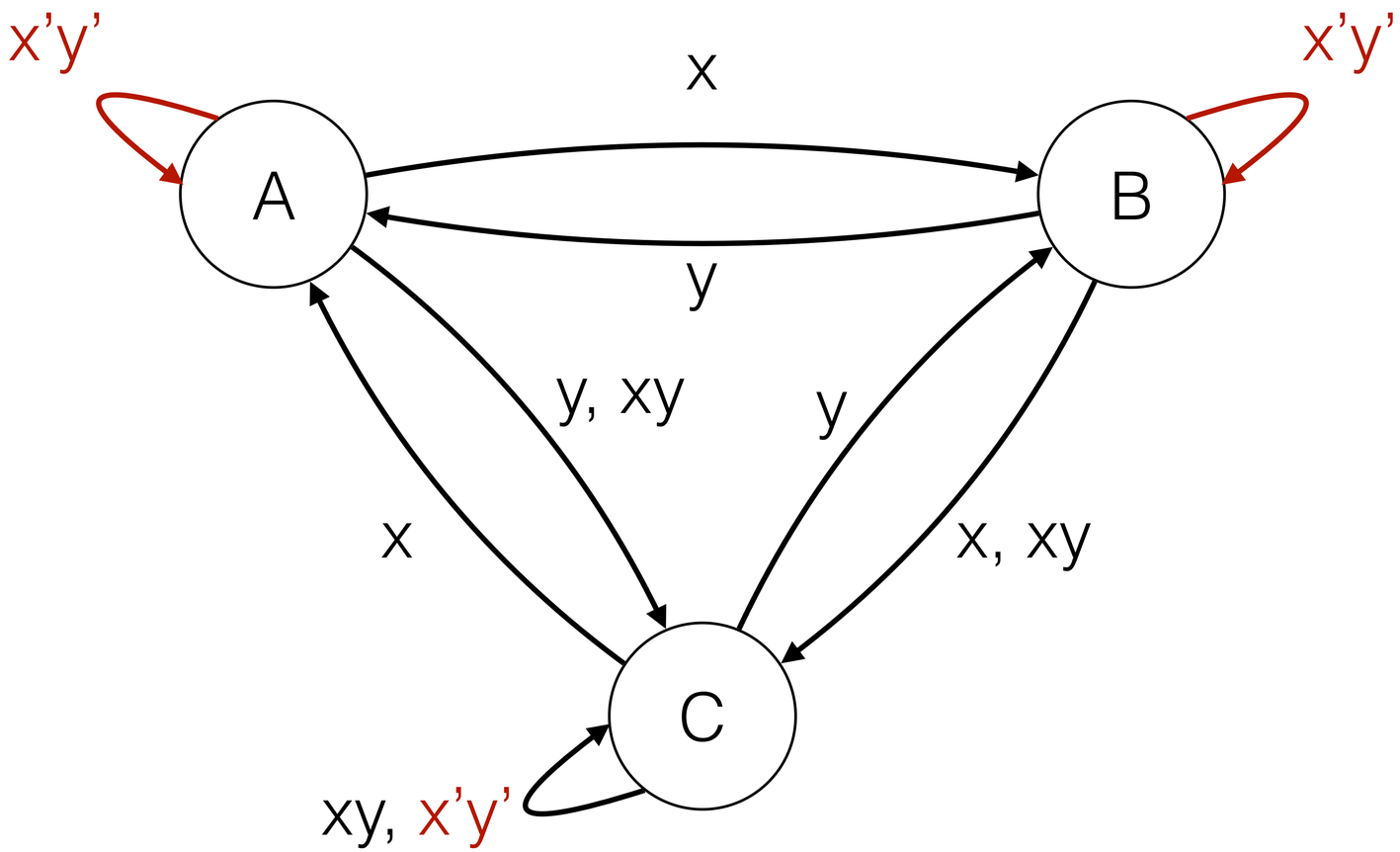
FSM

- If **x = 1** and **y = 1** go to state C

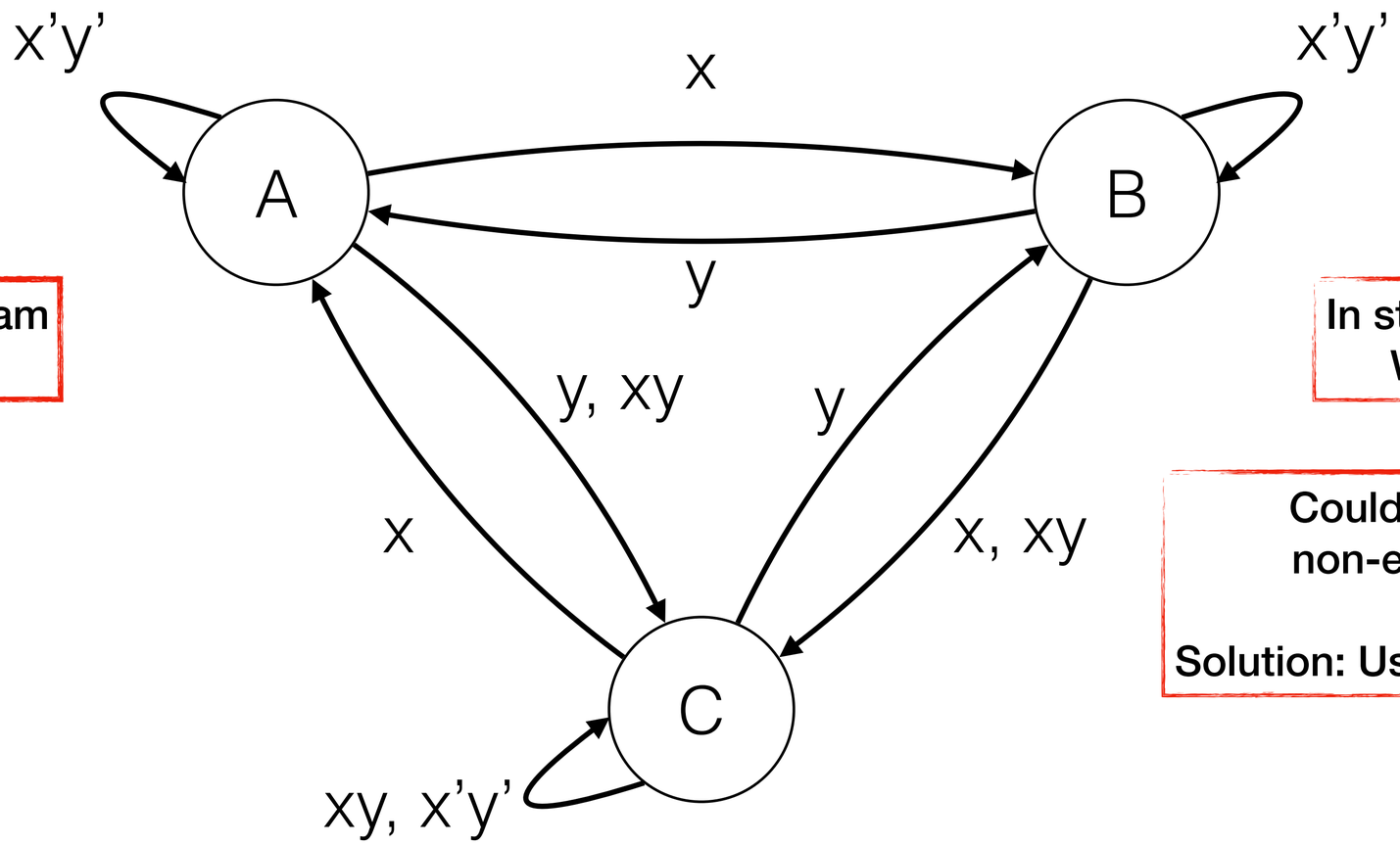


FSM

- If **x = 0** and **y = 0** stay in current state



FSM



Does this state diagram have any issues?

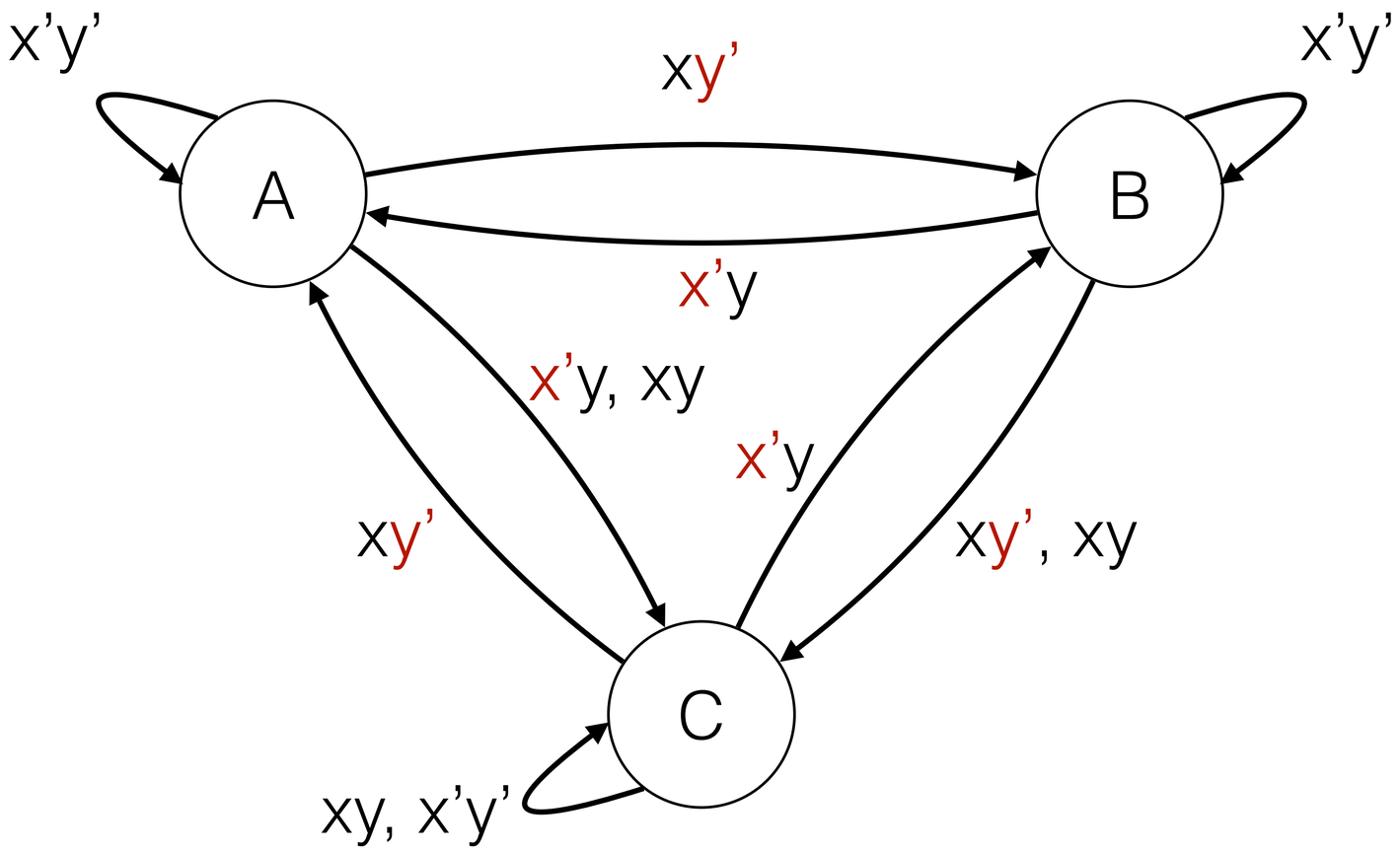
In state A with input x=1, y=1: What is the next state?

Could be B or C. We have non-exclusive transitions.

Solution: Use minterm for each event!

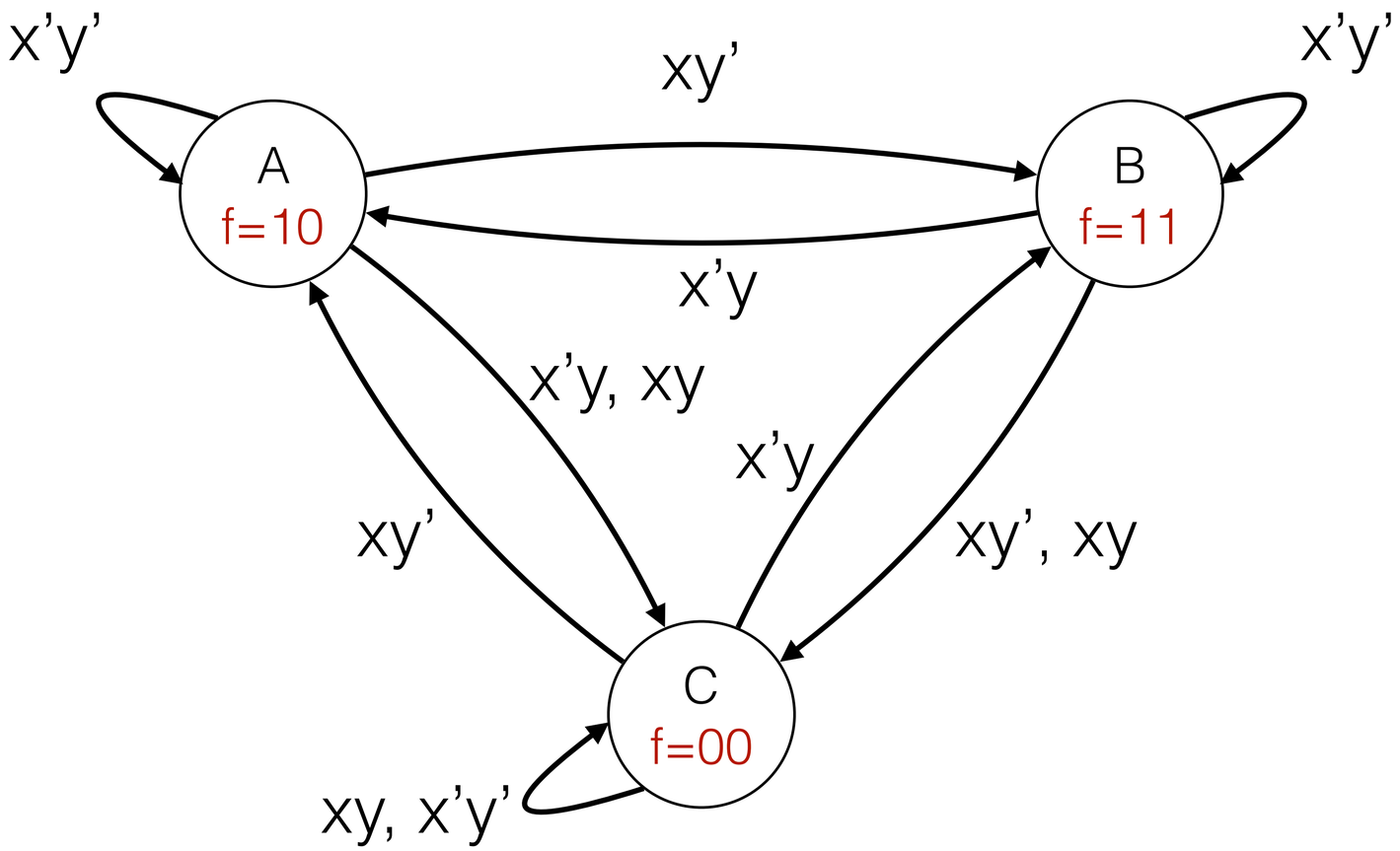
FSM

- Minterm for each event to ensure complete transitions and avoid non-exclusive transitions!



FSM

- Output **f** will be the following for each state: A: 0b10, B: 0b11, C: 0b00

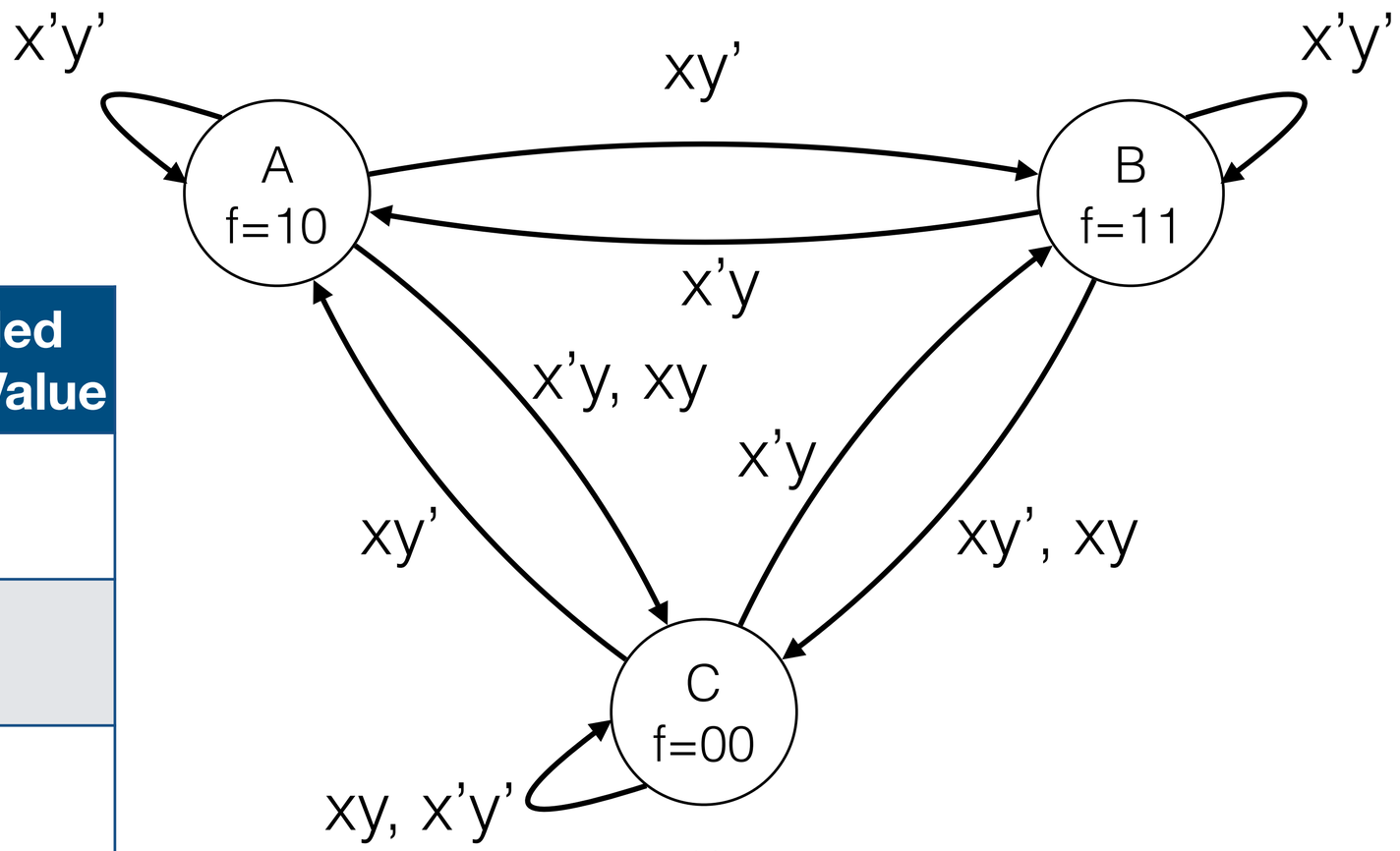


From FSM to Circuit

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

FSM

- Encoded values were given

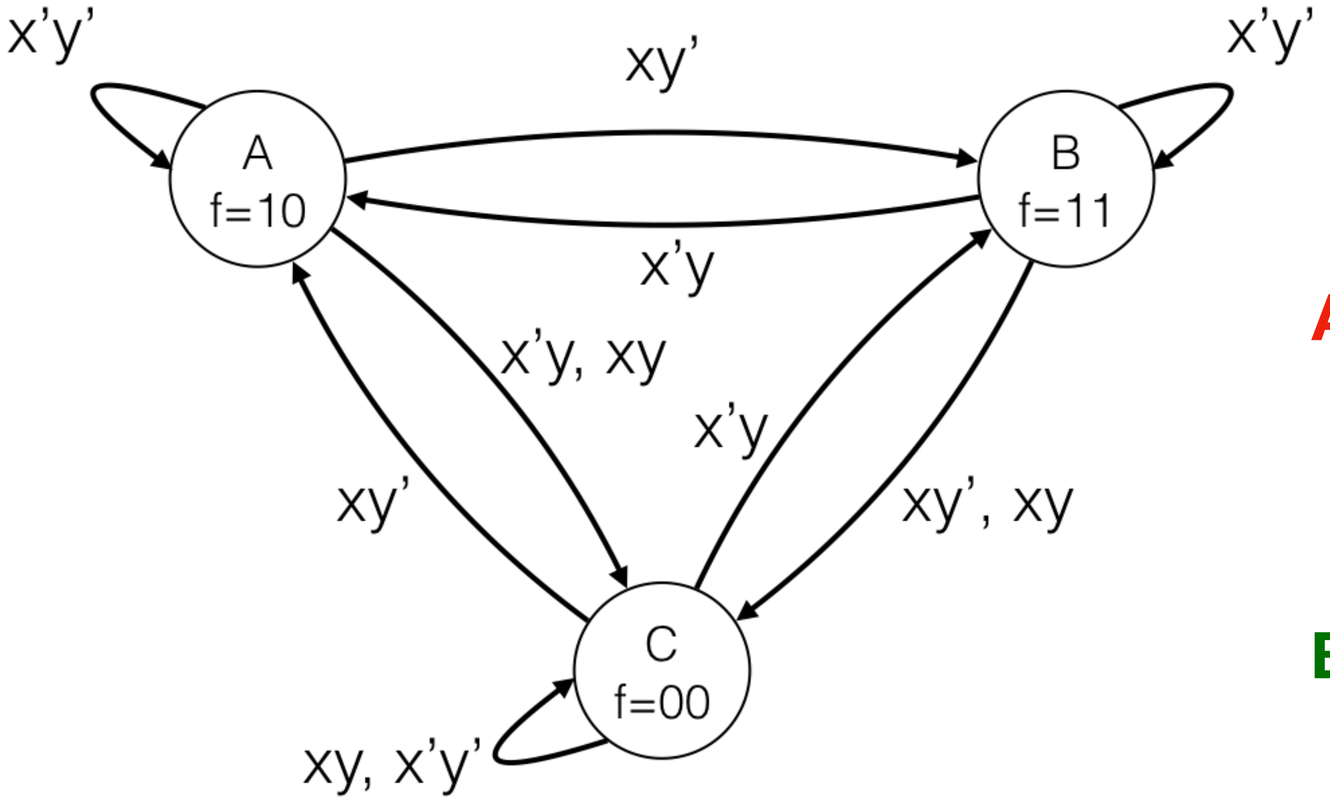


State	Encoded Binary Value
A	00
B	11
C	01

From FSM to Circuit

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

FSM



State	Encoded Binary Value
A	00
B	11
C	01

x	y	c1	c0	f1	f0	n1	n0
0	0	0	0	1	0	0	0
0	1	0	0	1	0	0	1
1	0	0	0	1	0	1	1
1	1	0	0	1	0	0	1
0	0	1	1	1	1	1	1
0	1	1	1	1	1	0	0
1	0	1	1	1	1	0	1
1	1	1	1	1	1	0	1
0	0	0	1	0	0	0	1
0	1	0	1	0	0	1	1
1	0	0	1	0	0	0	0
1	1	0	1	0	0	0	1

A

B

C

From FSM to Circuit

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

FSM

x	y	c1	c0	f1	f0	n1	n0
0	0	0	0	1	0	0	0
0	1	0	0	1	0	0	1
1	0	0	0	1	0	1	1
1	1	0	0	1	0	0	1
0	0	1	1	1	1	1	1
0	1	1	1	1	1	0	0
1	0	1	1	1	1	0	1
1	1	1	1	1	1	0	1
0	0	0	1	0	0	0	1
0	1	0	1	0	0	1	1
1	0	0	1	0	0	0	0
1	1	0	1	0	0	0	1

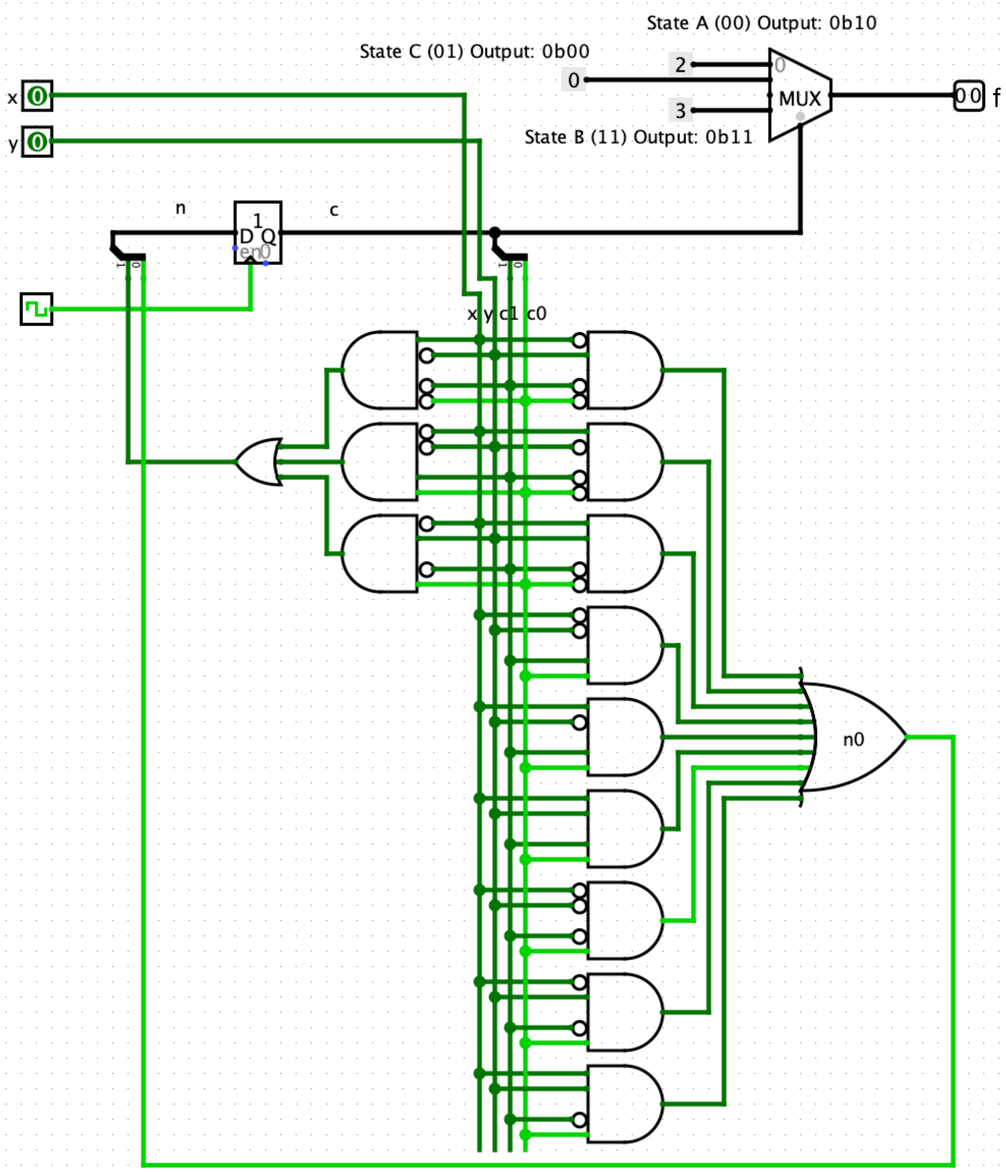
n1 = xy'c1'c0' + x'y'c1c0 + x'yc1'c0

n0 = x'yc1'c0' + xy'c1'c0' + xyc1'c0' +
x'y'c1c0 + xy'c1c0 + xyc1c0 +
x'y'c1'c0 + x'yc1'c0 + xyc1'c0

From FSM to Circuit

1. Create state diagram for FSM
2. Encode states
3. Create truth table (with inputs, current state, outputs and next state)
4. Derive Boolean equation (for all outputs and next state from truth table)
5. Create FSM circuit (from Boolean equations)

FSM


State A (00) Output: 0b10
State C (01) Output: 0b00
State B (11) Output: 0b11
2
0
3
MUX
00 f
x
y
n
c
1
D Q
en
x y c1 c0
n0