Analysis of Flip Flop Circuits

Analysis of Sequential Logic Circuits

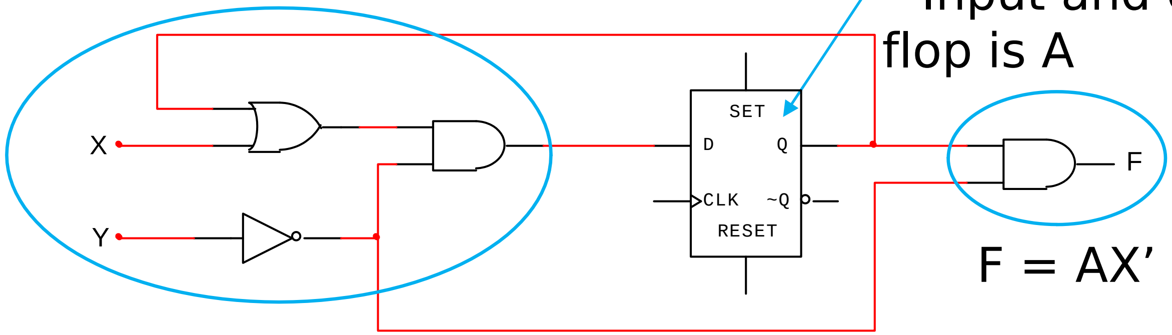
- State equations are similar to Boolean expressions from combinational logic
  - Describe the output and transition logic of circuit
- State table is similar to a truth table
  - Describes state transition and output given combination of inputs
- State diagrams are visual representations of the state table

Circuit to State Equation

* State equation is the Boolean expression for circuit

* Will have multiple equations
  - One for output of circuit (F)
  - One to describe, state, or input to flip flops

* Each flip flop is designated A,B,C,…etc.
* Input and output to flip flop is A

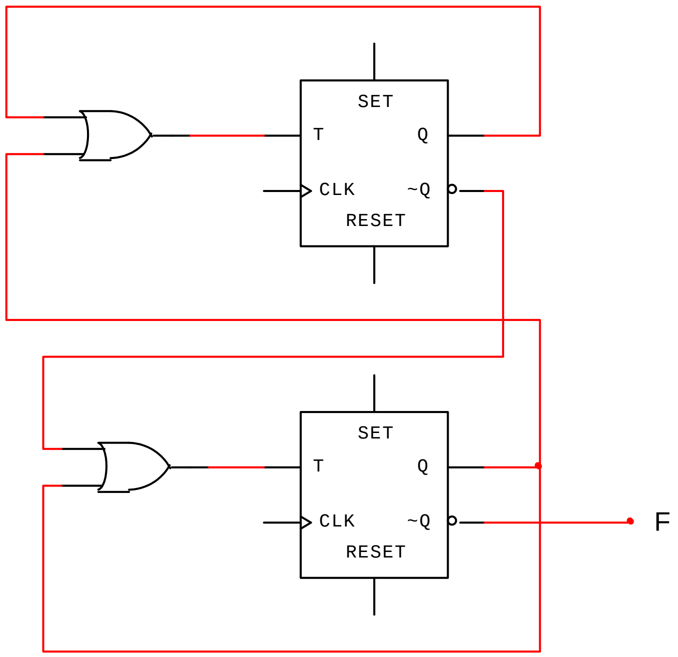


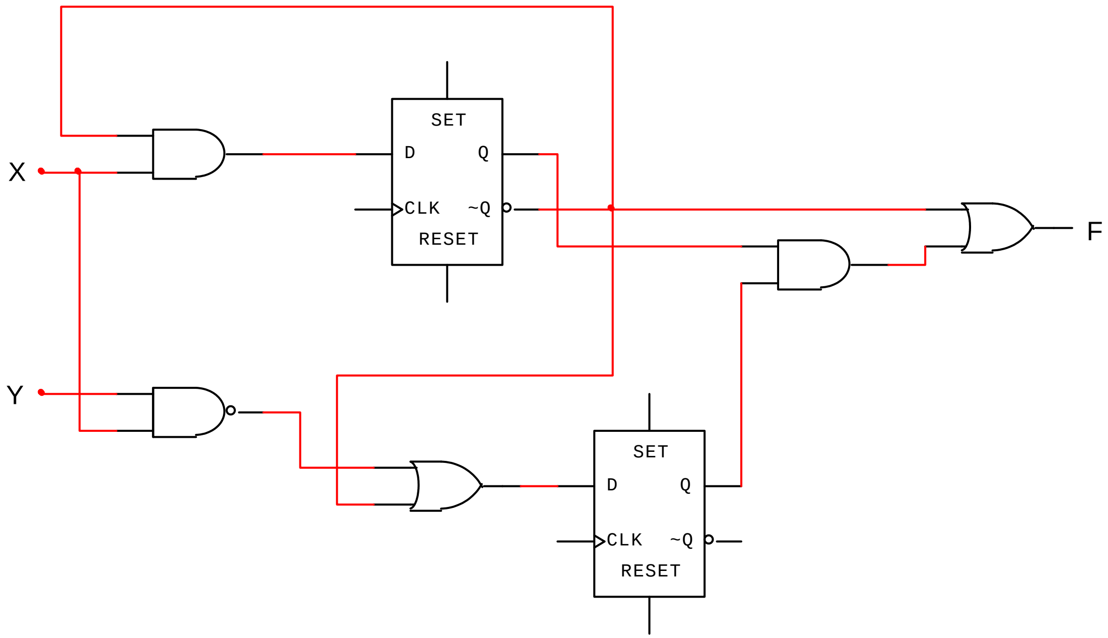
F = AX’

A = (A + X)Y’

Example

What are the state equations for following circuits?





State Table

* Similar to a truth table, it describes all the possible outputs given input combinations
  - Inputs include current output of flip flops (A,B,..etc.) and input of circuit (X,Y,..etc.)
  - Outputs include next output of flip flops (A,B,..etc.) and output of circuit (F,G,..etc.)

Input to circuit

Flip Flops next output

Flip Flops current output

Present State		Input	Next State		Output
A	**B**	**x**	**A**	**B**	**F**
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	1	1
1	1	0	0	1	0
1	1	1	1	1	1

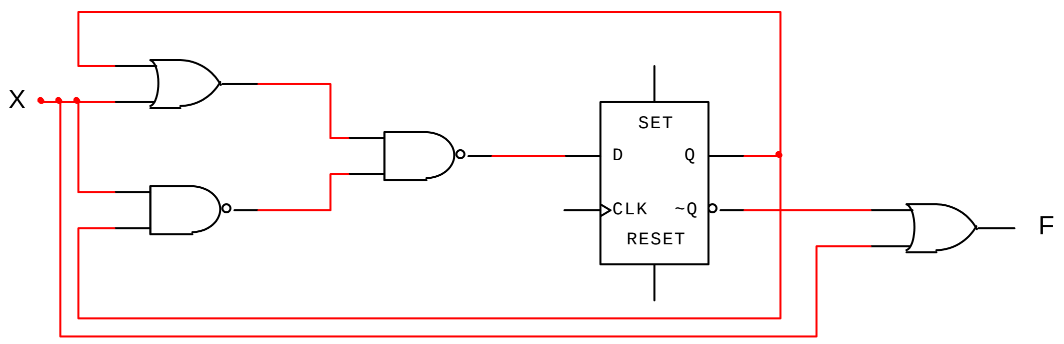
Output of circuit

Input to state table

Output of state table

Example

Create a State Table for the following circuit



Present State	Input	Next State	Output
A	X	A	F
0	0		
0	1		
1	0		
1	1		

State Table to K-map

* Each column of a state table output can be simplified with a K - map

Present State		Input	Next State		Output
A	B	x	A	B	F
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	1	1
1	1	0	0	1	0
1	1	1	1	1	1

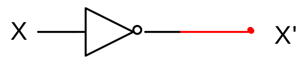
3 K – maps in total

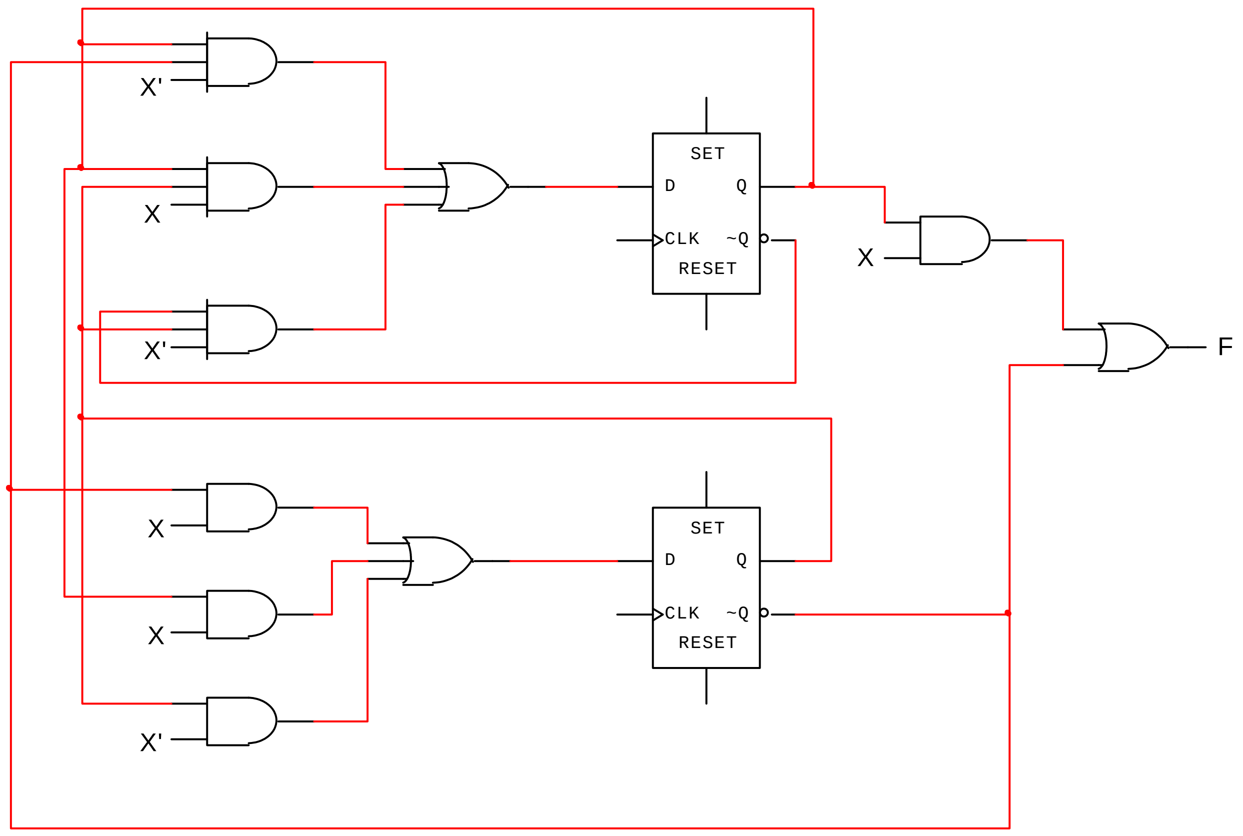
Example

Use K – Map to draw the logic circuit from the state table

Present State		Input	Next State		Output
A	**B**	**x**	**A**	**B**	**F**
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	1	1
1	1	0	0	1	0
1	1	1	1	1	1

Example



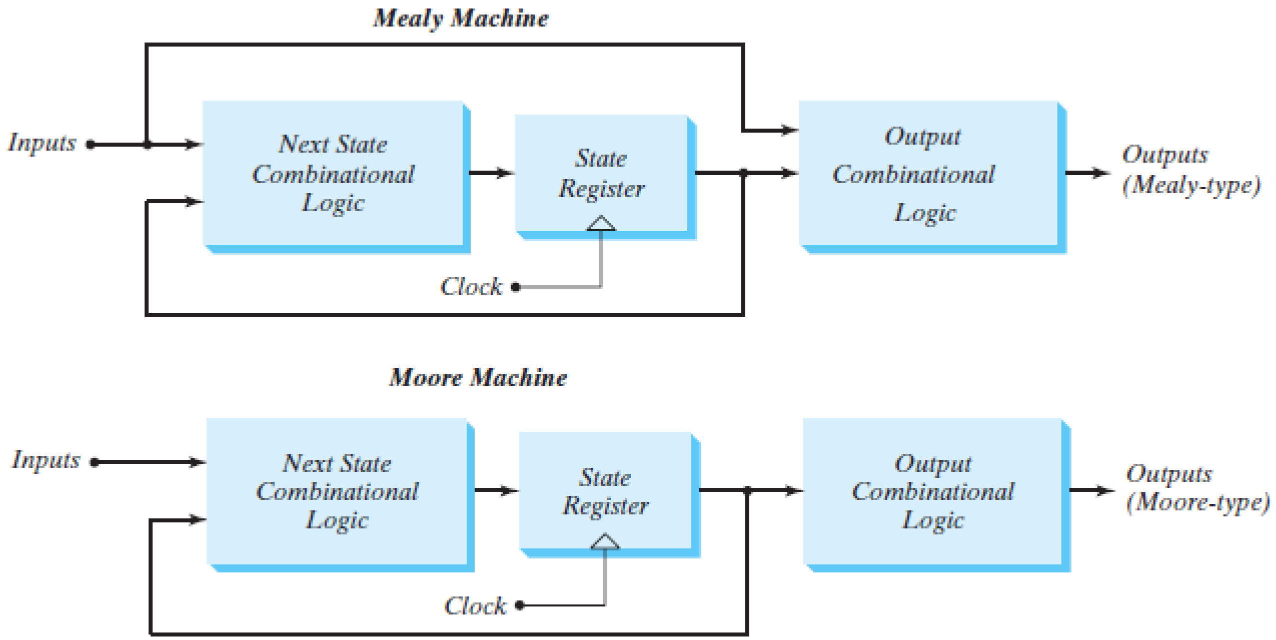


Finite State Machine

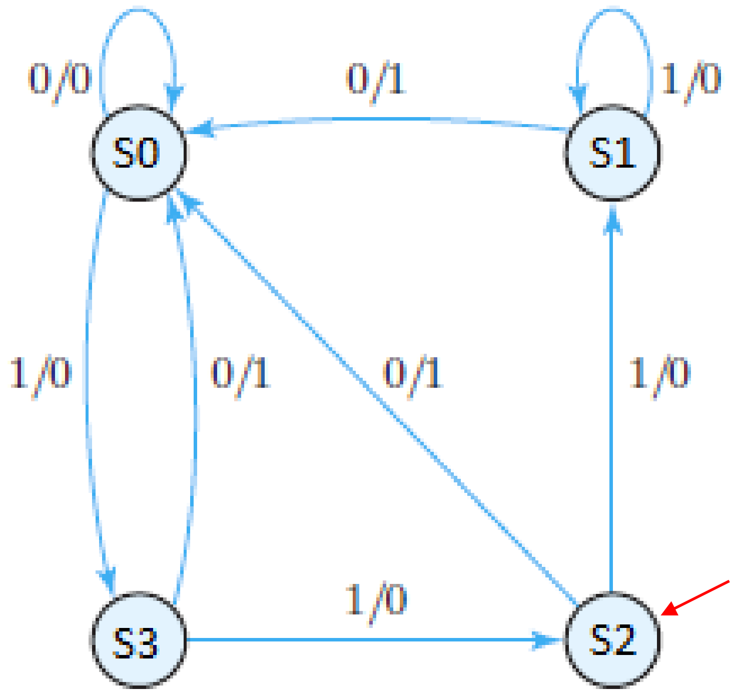
* Diagram showing all states, how to transition from state to state, and output at each state

* **Mealy** – Output depends on current state and input. Output changes immediately

* **Moore** changes on next clocking e

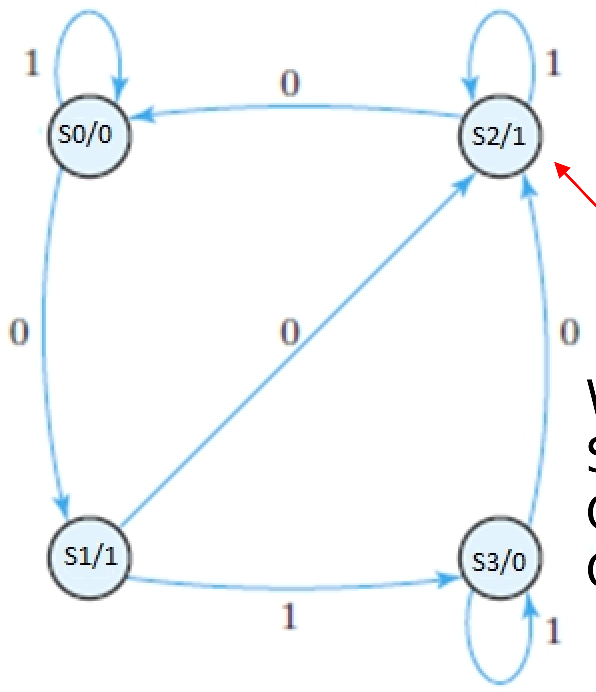


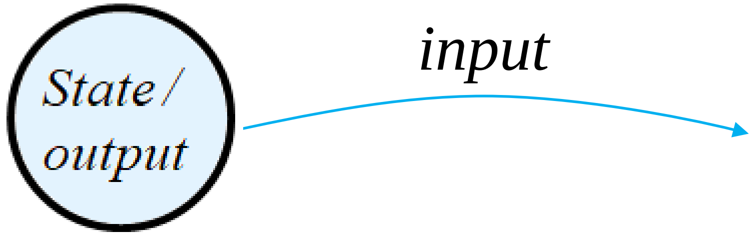
Mealy State Diagram



When in state 2, transition to state 1 if input is 1, output becomes 0. If input is 0, transition to state 0, output becomes 1. Get to state 2 from state 3 if input is 1, output becomes 0.

Moore State Diagram





When in state 2, output is 1. Transition to state 0 if input is 0.
Stay in state 2 if input is 1, output is 1.
Get to state 2 from state 1 if input is 0, output is 1.
Get to state 2 from state 3 if input is 0, output is 1.

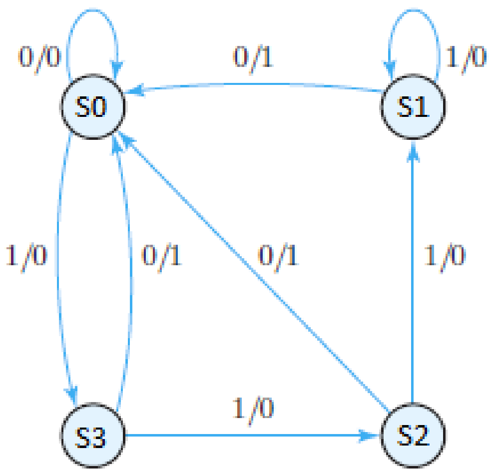
Encode States

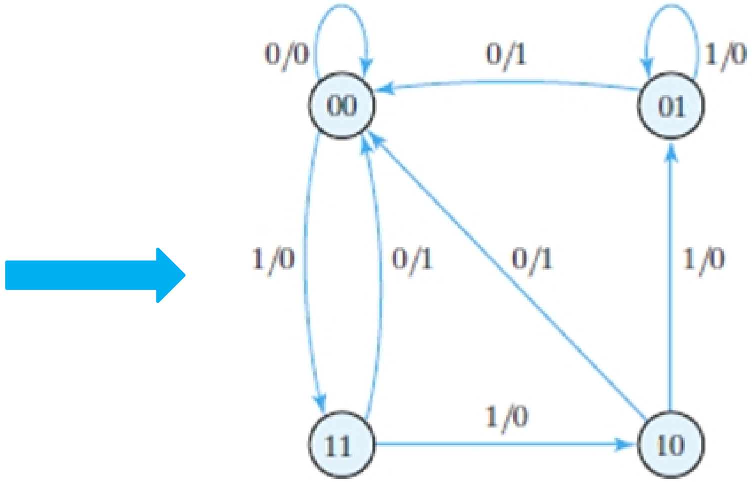
* States are encoded as binary values to design FSM into a logic circuit

* Make each state a binary number

* Previous examples had 4 states, so 2 bits are used to represent each state. Each bit is 1 flip flop

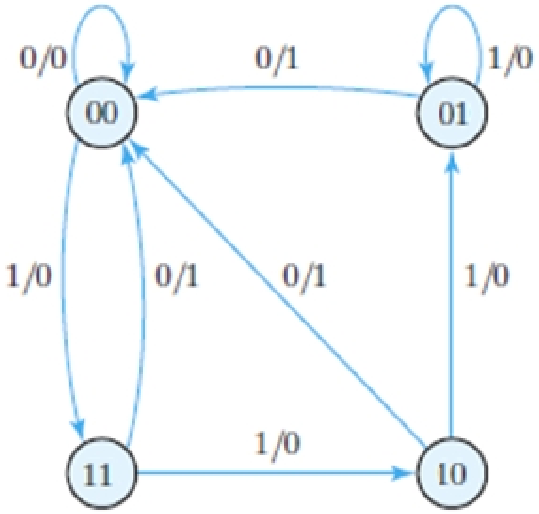
- S0 -> 00
- S1 -> 01
- S2 -> 10
- S3 -> 11





Creating State Table

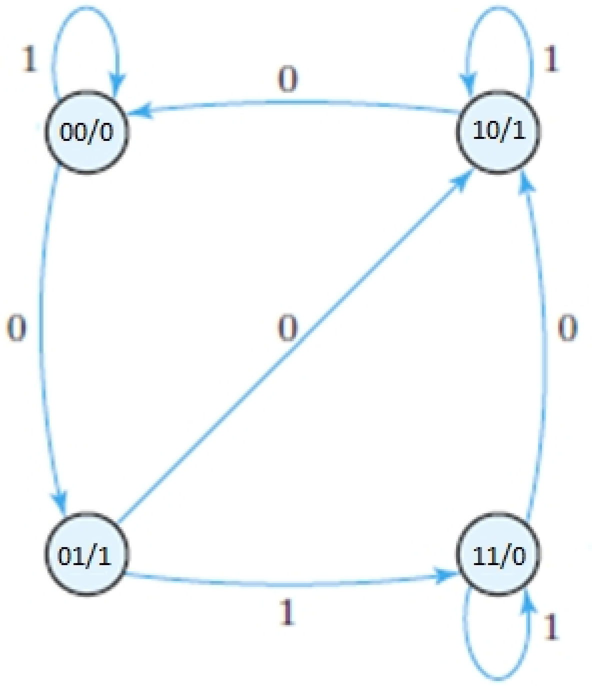
* Use encoded state values and input to create table for next state and output. A and B are a single bit from state encoding.



	Present State		Input	Next State		Output
	A	**B**	***x***	**A**	**B**	***y***
S0	0	0	0	0	0	0
	0	0	1	1	1	0
S1	0	1	0	0	0	1
	0	1	1	0	1	0
S2	1	0	0	0	0	1
	1	0	1	0	1	0
S3	1	1	0	0	0	1
	1	1	1	1	0	0

Mealy FSM

Moore State Table



Present State		Input	Next State		Output
A	**B**	**x**	**A**	**B**	**y**
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	1	1	0

Output is dependent on present state

K – Map State Table

Inputs on K-Map

K-Map outputs (each column will need a K-Map)

Present State		Input	Next State		Output
A	**B**	***x***	**A**	**B**	***y***
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	0	0	1
1	1	1	1	0	0

K - Map

Present state and input from state table

A \ Bx	00	01	11	10
0		1		
1			1	

A

A \ Bx	00	01	11	10
0		1	1	
1		1		

B

Next state from state table

A \ Bx	00	01	11	10
0				1
1	1			1

y

output from state table

K - Map

A \ Bx	00	01	11	10
0		1		
1			1	

A

A \ Bx	00	01	11	10
0		1	1	
1		1		

B

A \ Bx	00	01	11	10
0				1
1	1			1

y

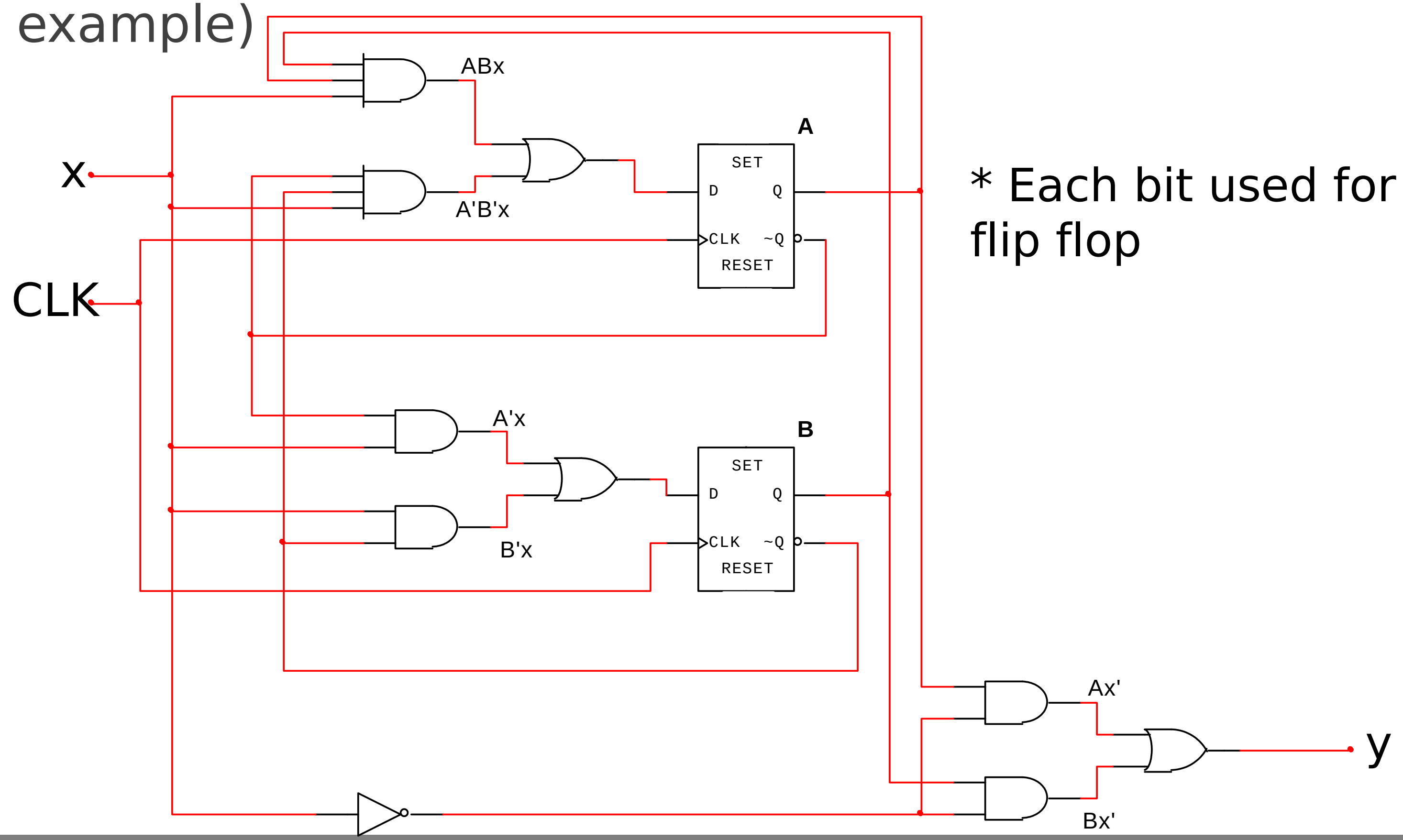
A = A’B’x + ABx

B = B’x + A’x

y = Bx’ + Ax’

Build Logic Circuit

* A and B are flip flop output and fed back into flip flop input. (Using D Flip Flop in example)



* Each bit used for state encoding is a flip flop

What if State Doesn’t Exist?

* Mark states that don’t exist as “don’t care”, or X on a state table

* Same applies if a transition does not exist as well

* Use the X when creating k-map to try and simplify circuit

Example

* Given the following state diagram create the state table and logic circuit