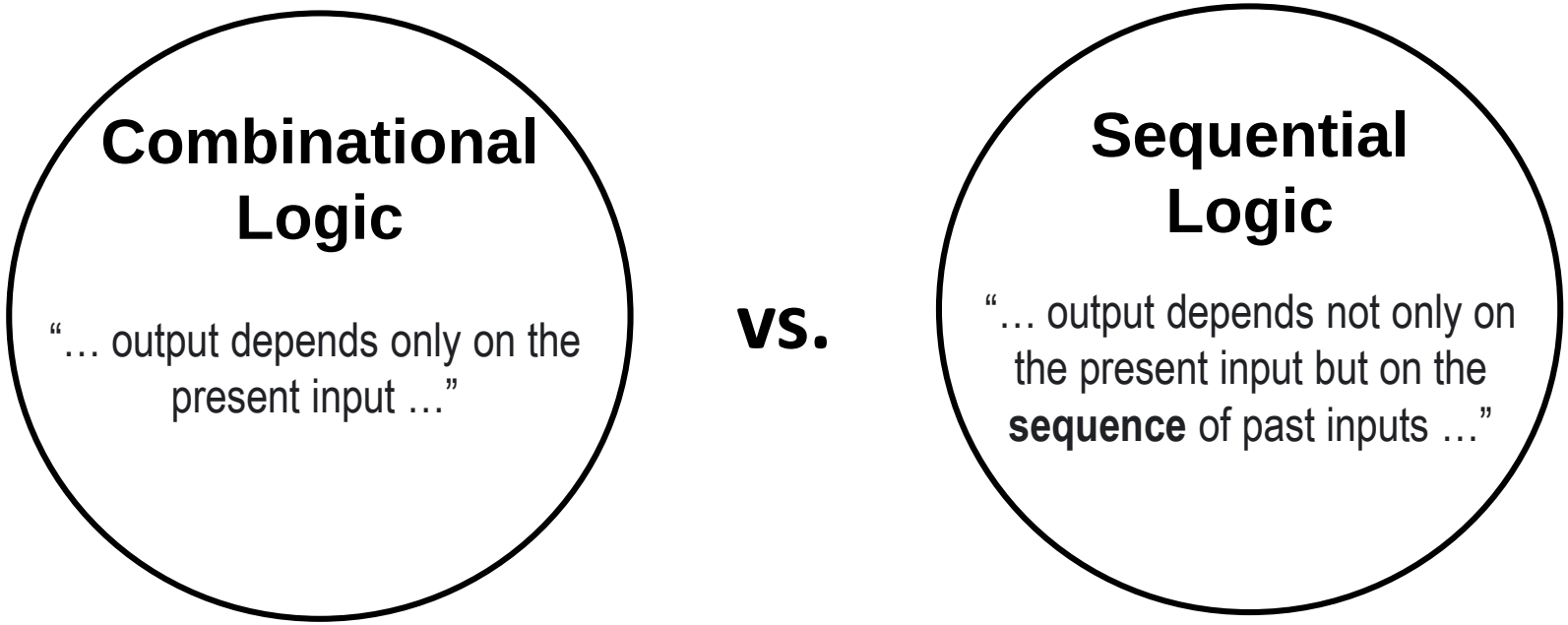
What We Are Studying is

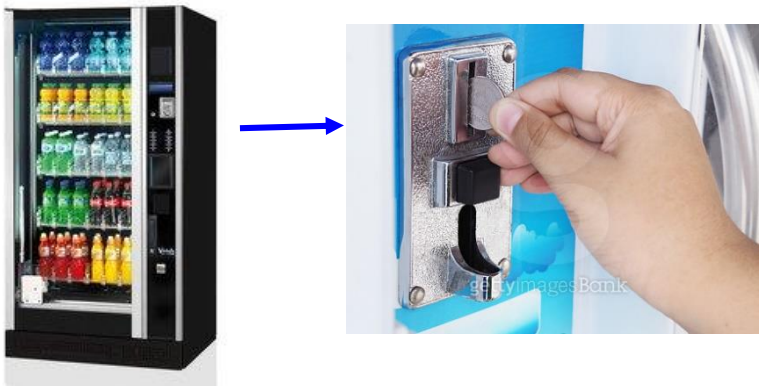


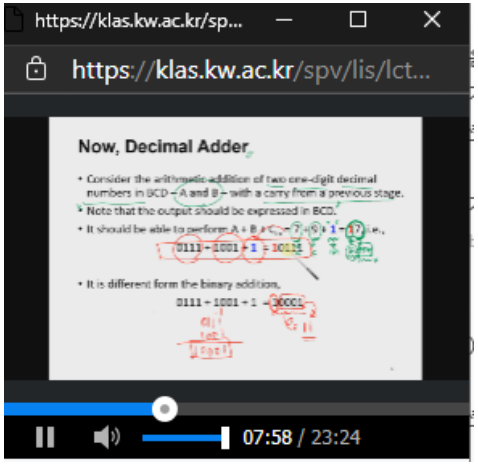
- You should know **why** for everything you learn.

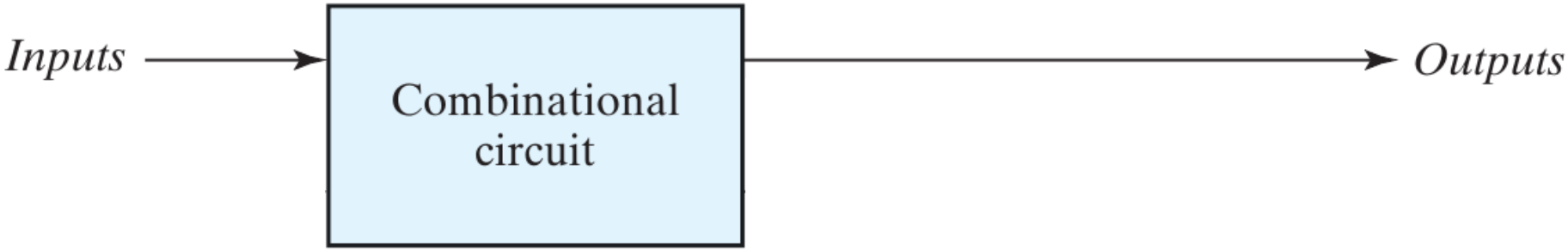
➔**Why** do we need sequential logic additional to combinational logic?

➔**Why** do we separate two concepts?

“Feel” the Difference b/w Combinational Logic vs. Sequential Logic



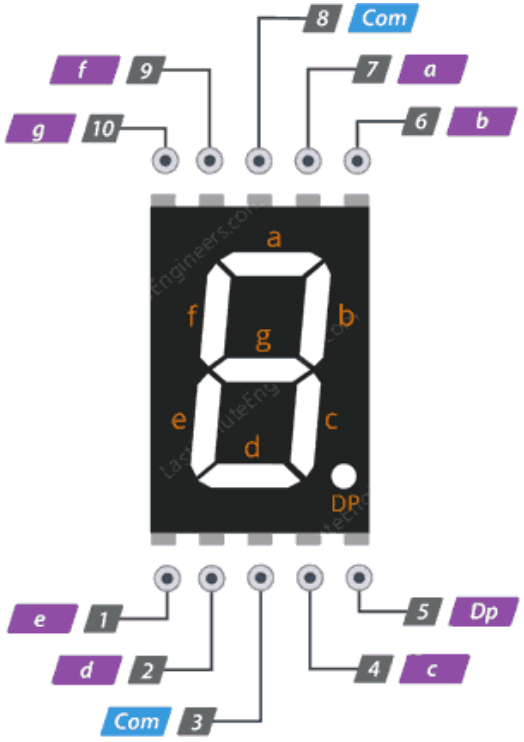




Feedback?

Ultimate Goal of Studying Synchronous Sequential Logic

We want to design a circuit of the target operation !





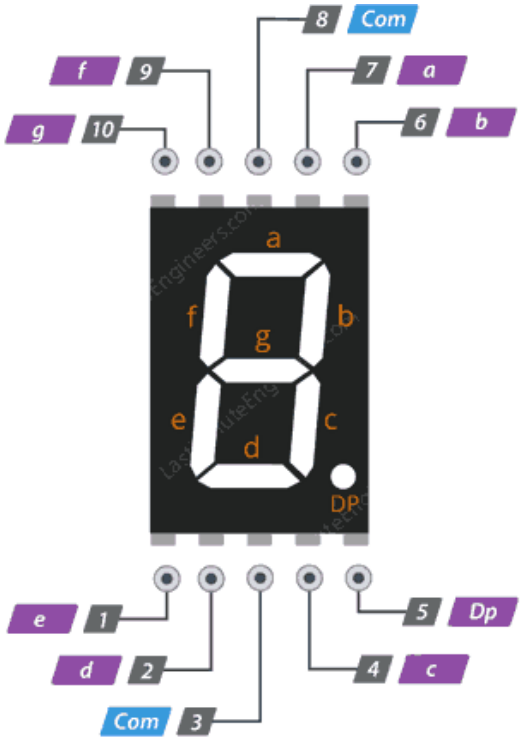
In Combinational Logic

- We have the useful “tools” for designing circuits
  - Truth table
  - K-map …
- What do we have for sequential logic?

What We Learned

- Why do we need and what is synchronous sequential logic
- Components composing of synchronous sequential logic
- Designing of a latch
- Designing of D Flip flop
- Setup time & hold time in FF
- JK FF and T FF

Revisit Button-Up
Seven Segment Machine

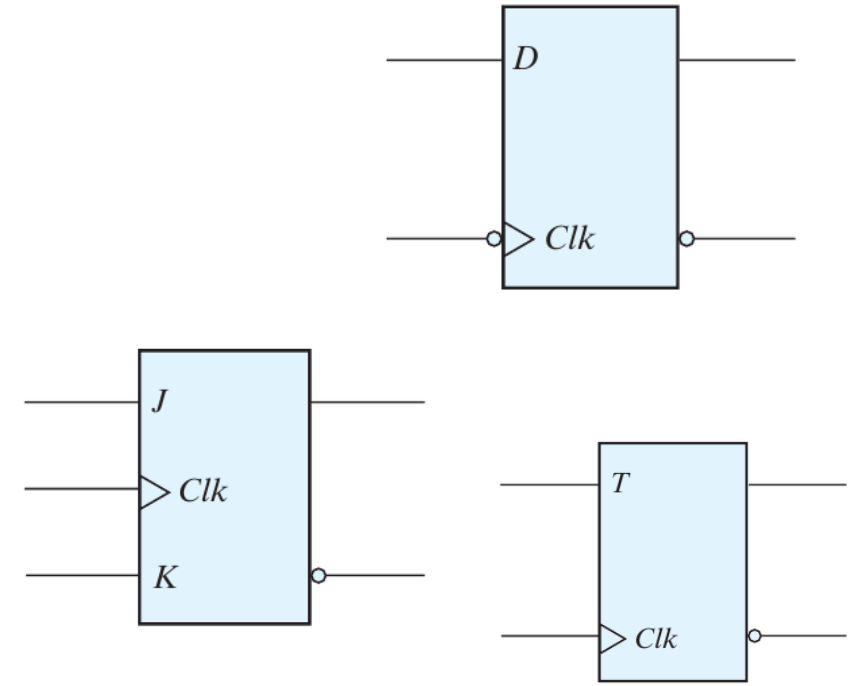




X	a	b	c	d	e	f	g	DP

Circuit Components We Have

- Logic gates
- Flip-flops



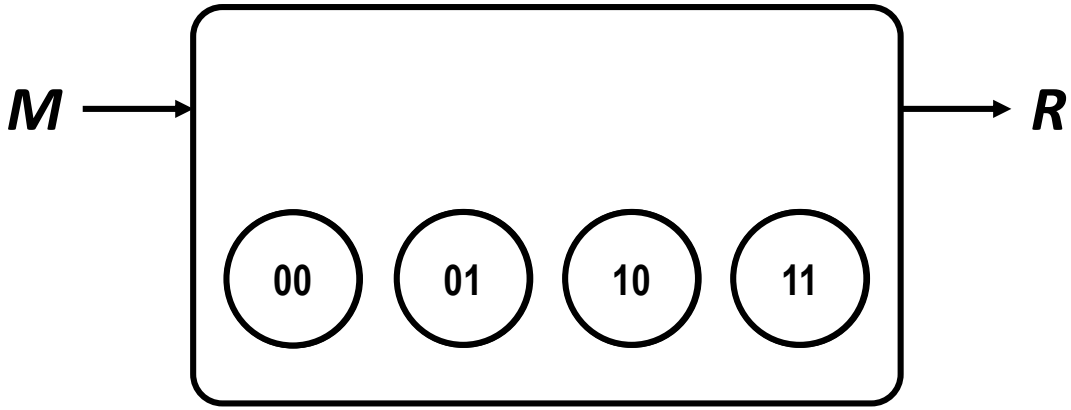
Chap 5
Synchronous Sequential Logic (2)

Hanwool Jeong

hwjeong@kw.ac.kr

What Do We Need?
Let's Simplify the Target Application

- Image a vending machine that gives you a "**RealDew**" drink every time you insert 4000 wons.
- You can only insert 1000 won bill.
- Can you specify input/output/**state** of a circuit?
  - Don't forget that they should be expressed in **binary codes**



Key Difference From Combinational Logic is..

- We cannot decide the output only based on the present input
- That is, in terms of truth table,

M	R

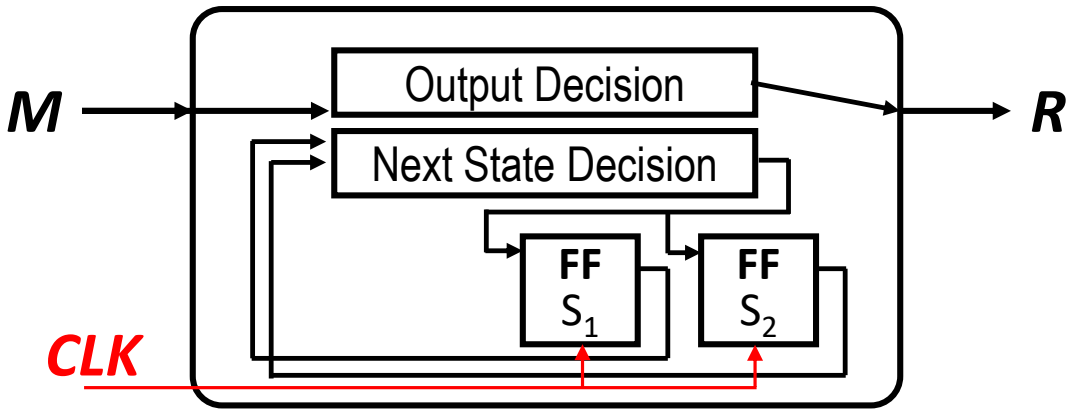
M	S_1	S_0	R

M	Present		Next		R
	S_1	S_0	S_1	S_0	

Revisit the RealDew Example with Roles of Flip-Flop

- **States** can be stored and updated using **Flip-Flop**
- Based on the input and states, the output and next state are determined.
- Note that FF updates at every rising edge of clock (Why?)

M	Present		Next		R
	S_1	S_0	S_1	S_0	



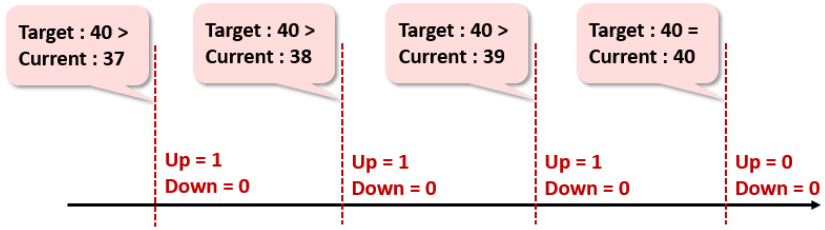
Why??

- See slide 4 and 23 of the previous lecture note

What Do We Need to Further Study?

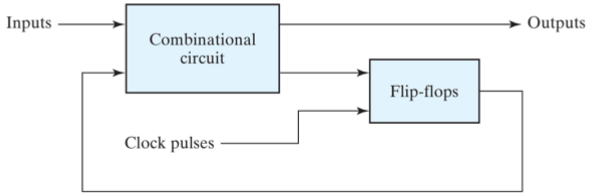
① How we **store** & update information

② Operating based on **discrete times**



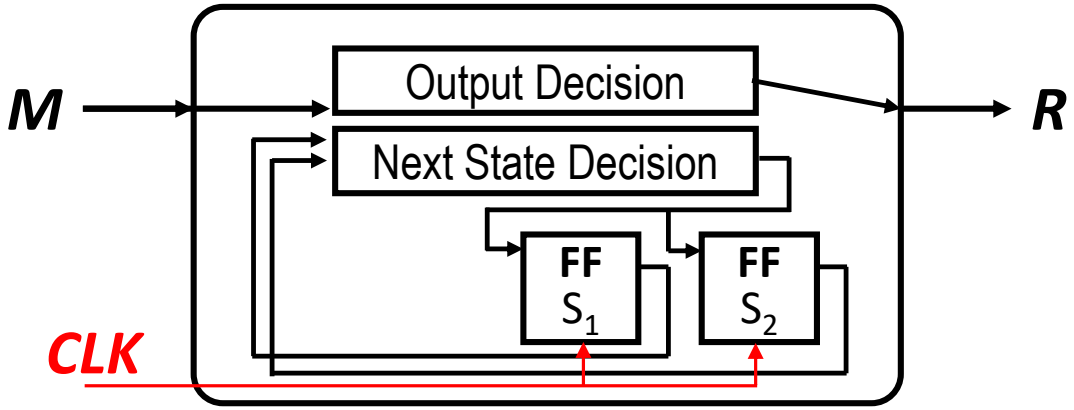
Revisit Synchronous Sequential Circuit; We Cannot Use Latch For State Storage

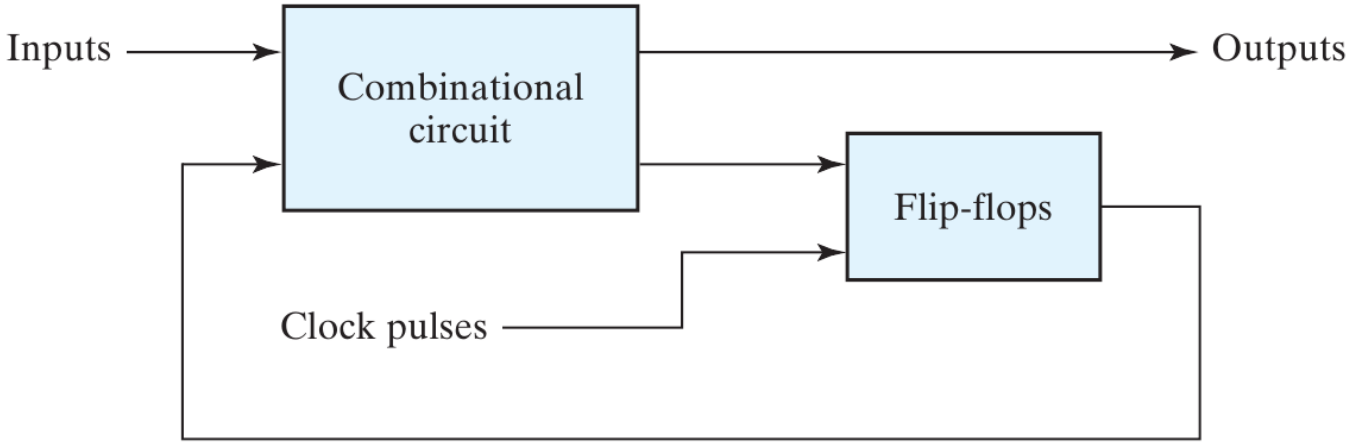
- The latch cannot be used for state storage below. Why?
  - Suppose that latch input changes while the clock pulse is still at 1.
  - The latches will respond to new values and update a new state.
  - The result is an unpredictable situation, since the state of the latches may keep changing for as long as the clock is high
- Thus, the output of a latch cannot be applied directly or thru combinational logic to the input of the same or another latch when all the latches are triggered by a common clock source.



23

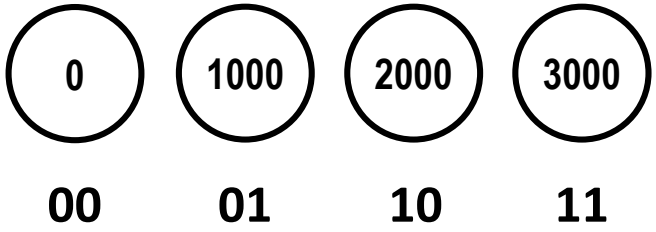
Now You May Better Understand …





Let's Start to Design Circuit

- State mapping



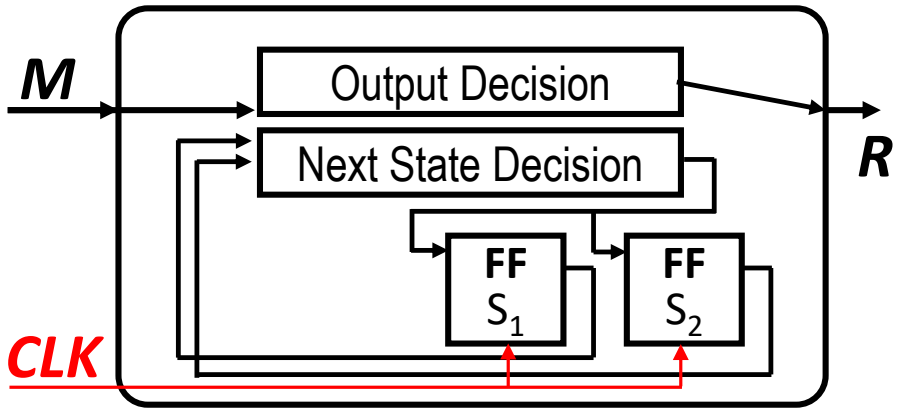
- Establishing truth table

M	Present		Next		R
	S_1	S_0	S_1	S_0	
0	0	**0**	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	0	**1**

State Table (=State Transition Table)

M	Present		Next		**R**
	S_1	S_0	S_1	S_0	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	0	**1**

Output Equation & State Equation



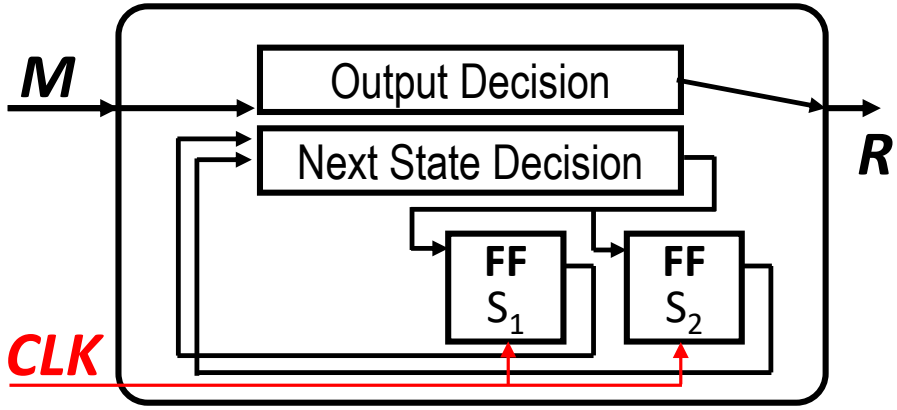
- $R = MS_1S_0$
- $S_0(t+1) = f(M,S_1,S_0) = M'S_0 + MS_0'$
- $S_1(t+1) = f(M,S_1,S_0) = M'S_1 + S_1S_0' + MS_1'S_0$

- How can you relate

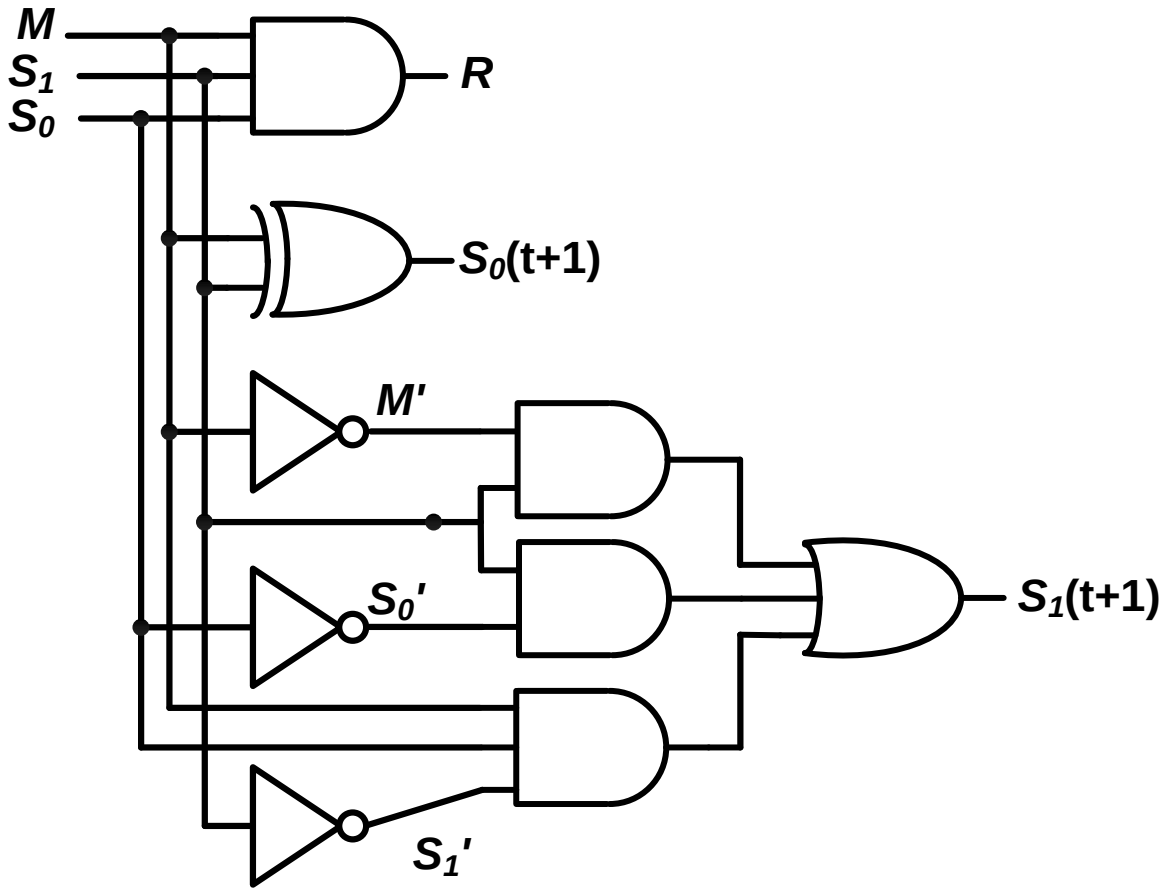
  FF input vs. **Next state?**

M	Present		Next		R
	S_1	S_0	S_1	S_0	
0	0	**0**	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	0	**1**

Next State vs. FF Input?



- $R = MS_1S_0$
- $S_0(t+1) = f(M,S_1,S_0) = M'S_0 + MS_0'$
- $S_1(t+1) = f(M,S_1,S_0) = M'S_1 + S_1S_0' + MS_1'S_0$



Depends on FF!
You remember Characteristic Equation?

- D Flip-flop

$$Q(t+1) = D$$

- JK Flip-flop

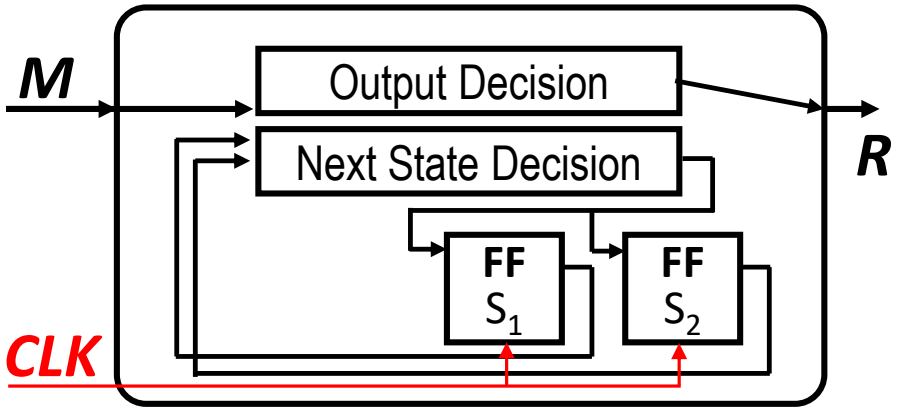
$$Q(t+1) = JQ' + K'Q$$

- T flip Flop

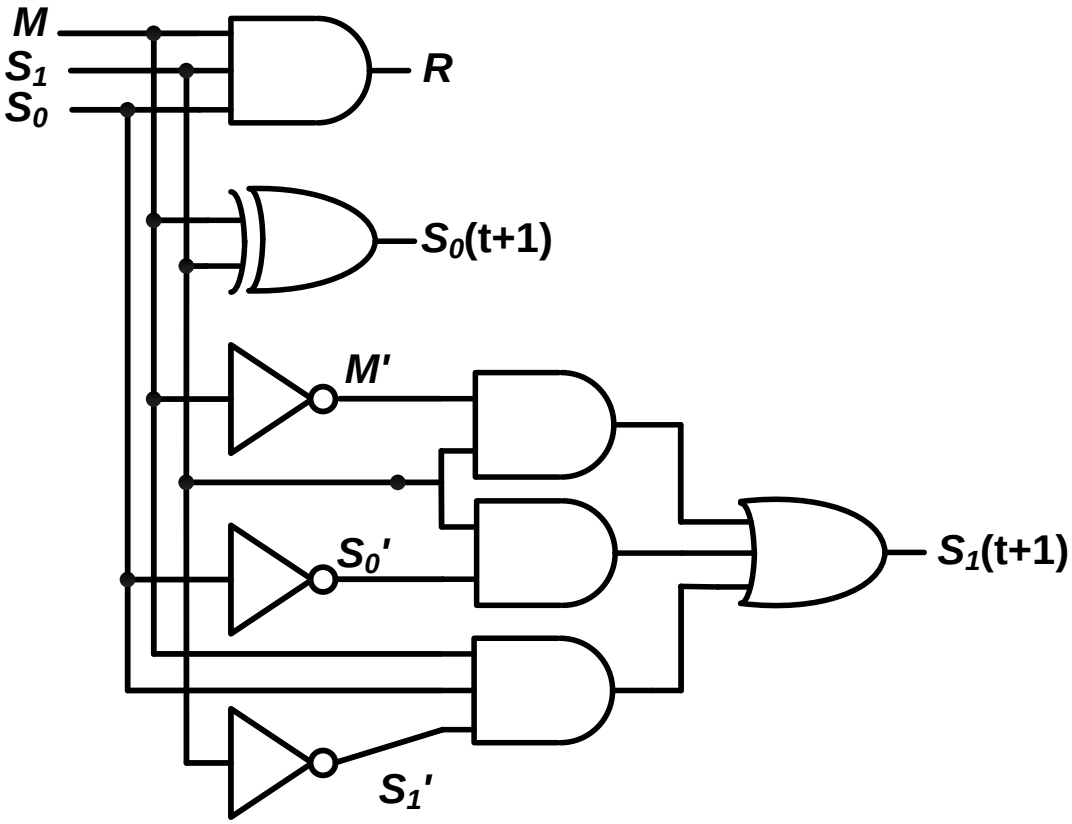
$$Q(t+1) = T \oplus Q = TQ' + T'Q$$

We also call these equations as the **state equations** of FF

Using D-FF is Very Simple



- Next state = FF input



- In general FF, this is not the case! **What do we need?**

Next State vs. FF Input; Excitation Table JK FF Example

- Focus on the nature of JK-FF
- Then, you can relate next state vs. FF input
  - Why do we see don't care (X) in the table

JK Flip-Flop			
J	***K***	***Q*(*t* + 1)**	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

***Q*(*t*)**	***Q*(*t* = 1)**	***J***	***K***
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

FF Input Decision Using Excitation Table

$Q(t)$	$Q(t+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

FF input table

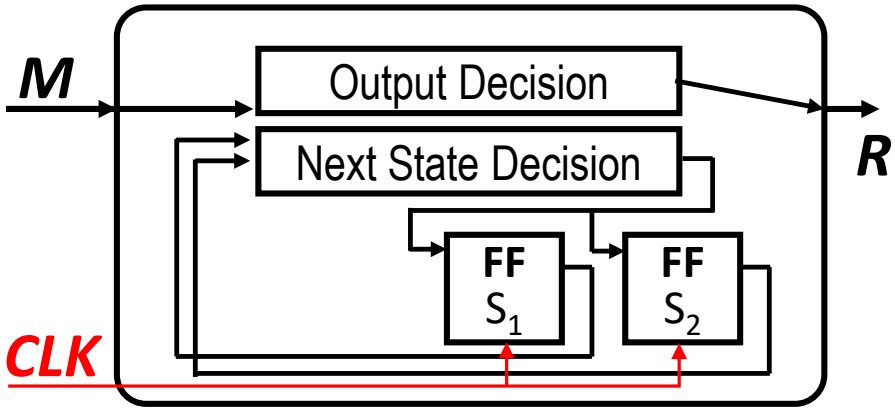
M	Present		Next		R	J_1	K_1	J_0	K_0
	S_1	S_0	S_1	S_0					
0	0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	0	X	X	0
0	1	0	1	0	0	X	0	0	X
0	1	1	1	1	0	X	0	X	0
1	0	0	0	1	0	0	X	1	X
1	0	1	1	0	0	1	X	X	1
1	1	0	1	1	0	X	0	1	X
1	1	1	0	0	**1**	X	1	X	1

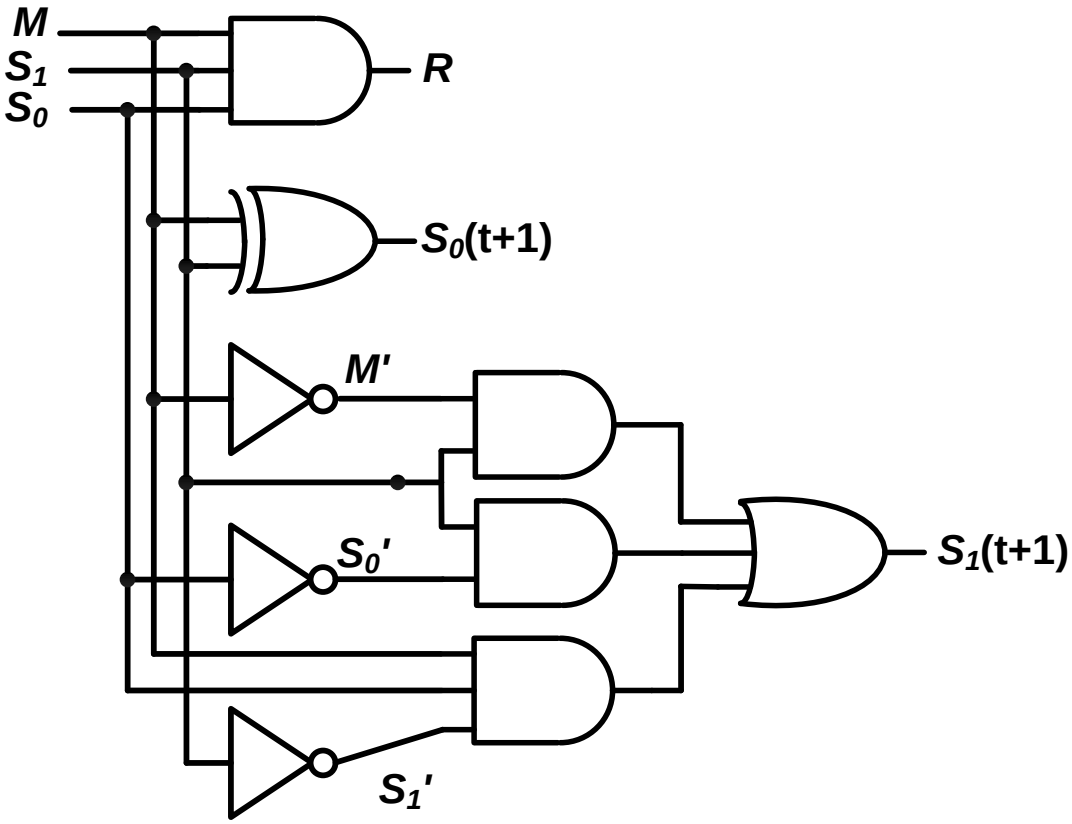
You Can Decide FF Input Equation

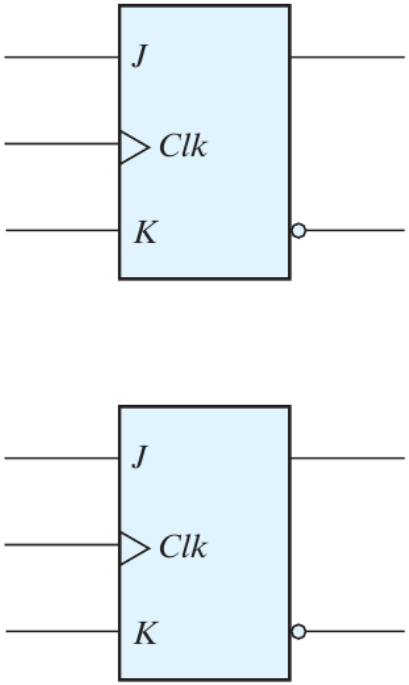
- J_1 =
- K_1 =
- J_0 =
- K_0 =

M	Present		Next		R	J_1	K_1	J_0	K_0
	S_1	S_0	S_1	S_0					
0	0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	0	X	X	0
0	1	0	1	0	0	X	0	0	X
0	1	1	1	1	0	X	0	X	0
1	0	0	0	1	0	0	X	1	X
1	0	1	1	0	0	1	X	X	1
1	1	0	1	1	0	X	0	1	X
1	1	1	0	0	**1**	X	1	X	1

Then, You Can Achieve Circuit!







Excitation Table for T Flip Flop

- You can apply T-FF by yourself to the previous example!

T Flip-Flop

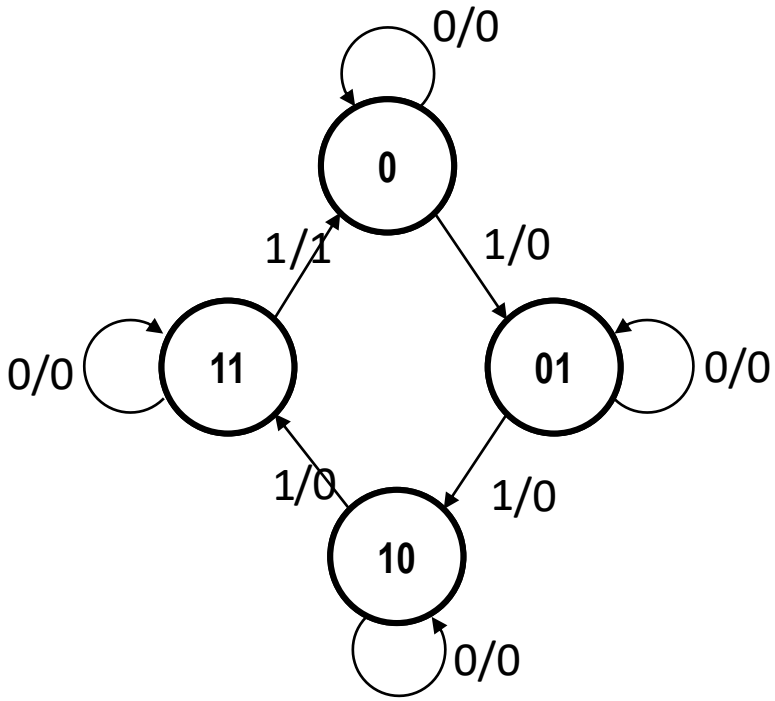
T	Q(t + 1)	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

Q(t)	Q(t = 1)	T
0	0	0
0	1	1
1	0	1
1	1	0

State Diagram; More Intuitive Way to Express Circuit Operation



- Or, you just start from the word description of
  - Image a vending machine that gives you a "**RealDew**" drink every time you insert 4000 wons.
  - You can only insert 1000 won bill.



M	Present		Next		R
	S_1	S_0	S_1	S_0	
0	0	**0**	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	0	**1**

What We Have Learned

- How to design a sequential circuit
- **State (transition) table**
- Current input & state ➔ Output & Next state
  - Here, FFs are used to store/update states
  - So, we need next state vs. FF input relationship
- **FF input table & FF input equations** can be obtained using **FF excitation table**
- Then you can finalize the circuit design
- **State diagram** is more intuitive way to express circuit operation and better starting point of circuit design!