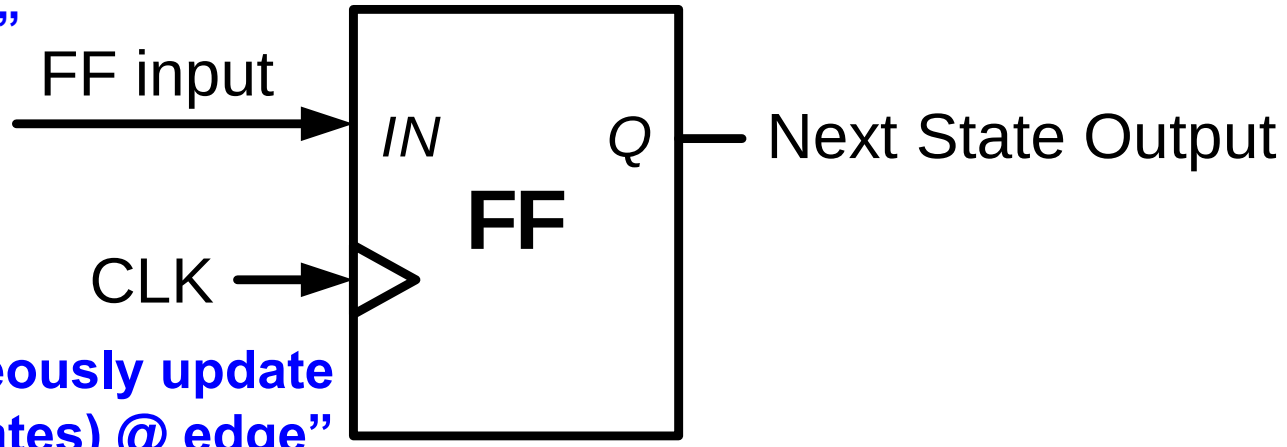
Basic Approach/Thought in "Synchronous" Sequential Logic

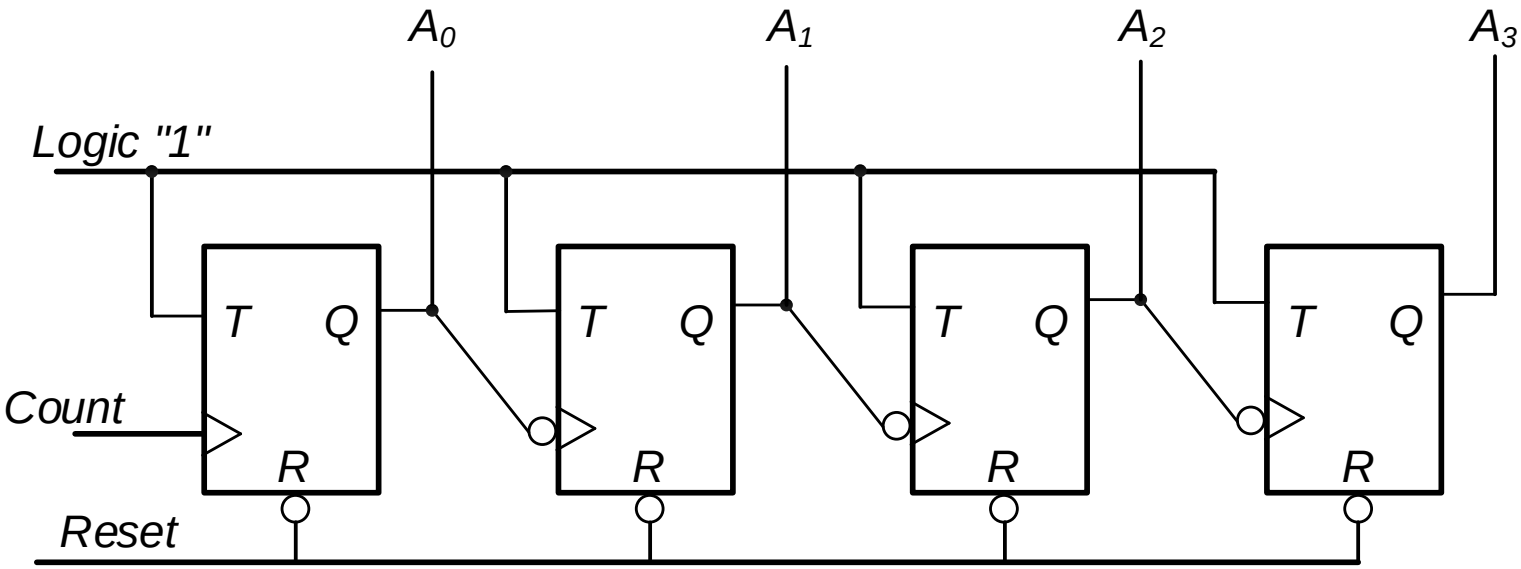
Circuit Design Focus : FF input generation

"Oh, then, I will prepare the FF input to generate intended "next state" outputs @ CLK edge"



"I will simultaneously update the outputs (states) @ edge"

Revisit Ripple Counter w/ T-FF



Chap 6 (3)
Synchronous Counter
w/ Binary Counter Example

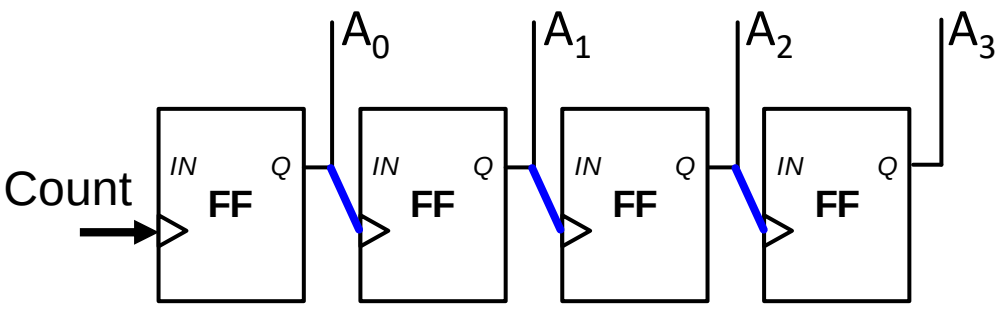
Hanwool Jeong

hwjeong@kw.ac.kr

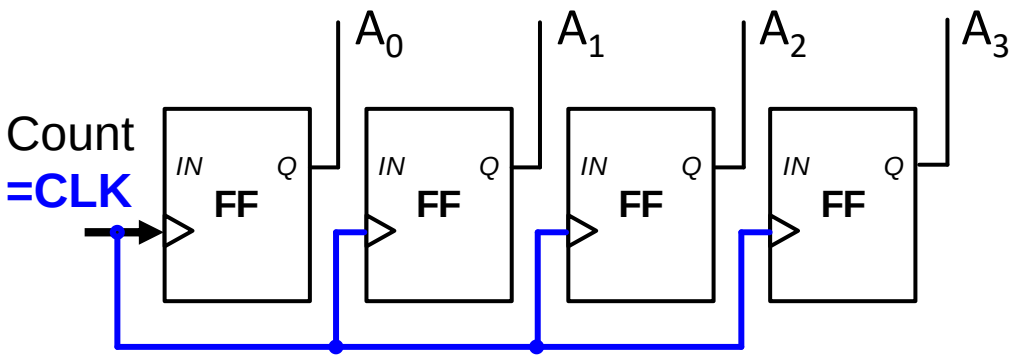
Key Difference b/w Ripple & Synchronous Counter

- Synchronous counters are different from ripple counters in that clock pulses are applied to the inputs of all flip-flops.
- A common clock triggers all flip-flops simultaneously, rather than one at a time in succession as in a ripple counter.

Ripple



Synchronous : Counts "CLK"



Binary Count Sequence

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

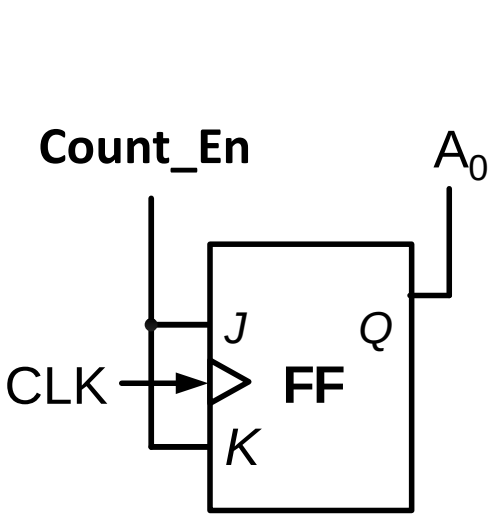
- Which FF (D-, JK-,T-) float in your mind?
- Although we can use any type,
- They toggle, right? **We can use JK- or T-**
  - (J, K) = (1,1) : Toggle
  - (J, K) = (0,0) : Hold
- We can start with how JK FF should be designed for A_0 and A_1
- Can we make counter operate only when we want?

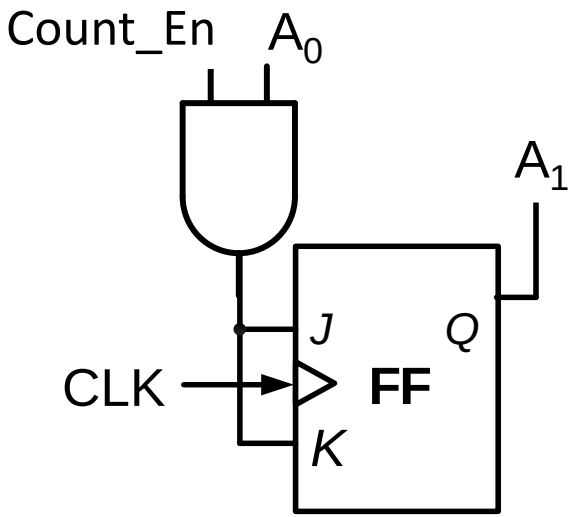
➔ Use **Count_En** signal!

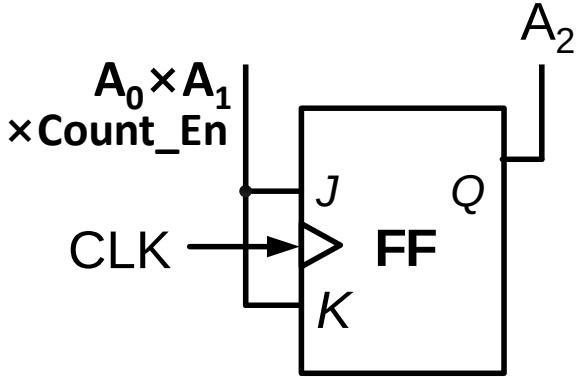
Binary Counter Design Using JK-FF

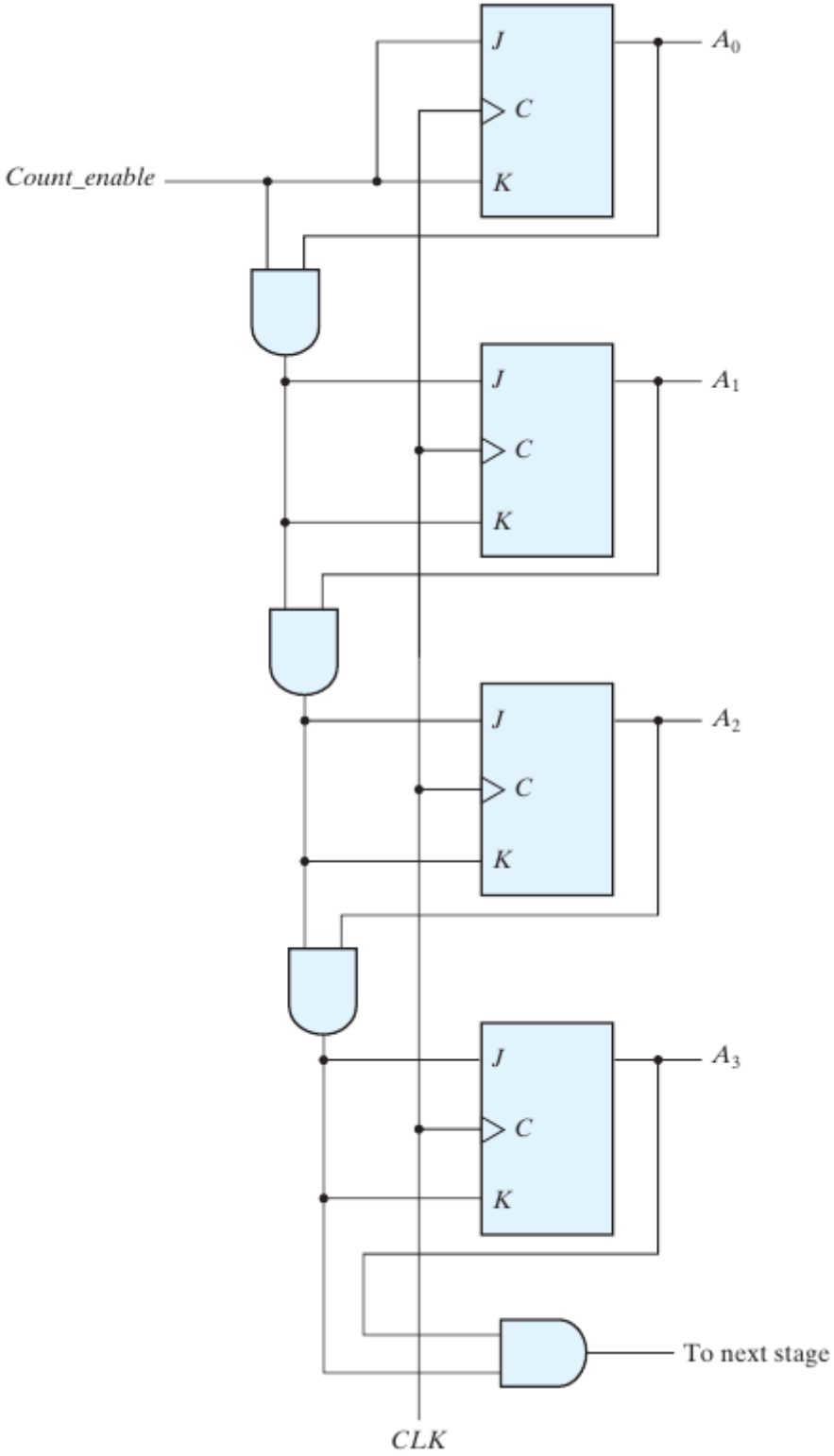
- Think how outputs should be:
  - A_0 : toggled @ every clock edge
  - A_1 : toggled @ clock edge w/ present A_0=1
  - A_2 : toggled @ clock edge w/ present A_1A_0=11

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0








J
C
K
A_0
Count_enable
J
C
K
A_1
J
C
K
A_2
J
C
K
A_3
To next stage
CLK

Checkpoints

- ✓Concept of synchronous counter
- ✓Design procedure for synchronous binary counter
- ✓Coming up next : Up/down counter & BCD counter