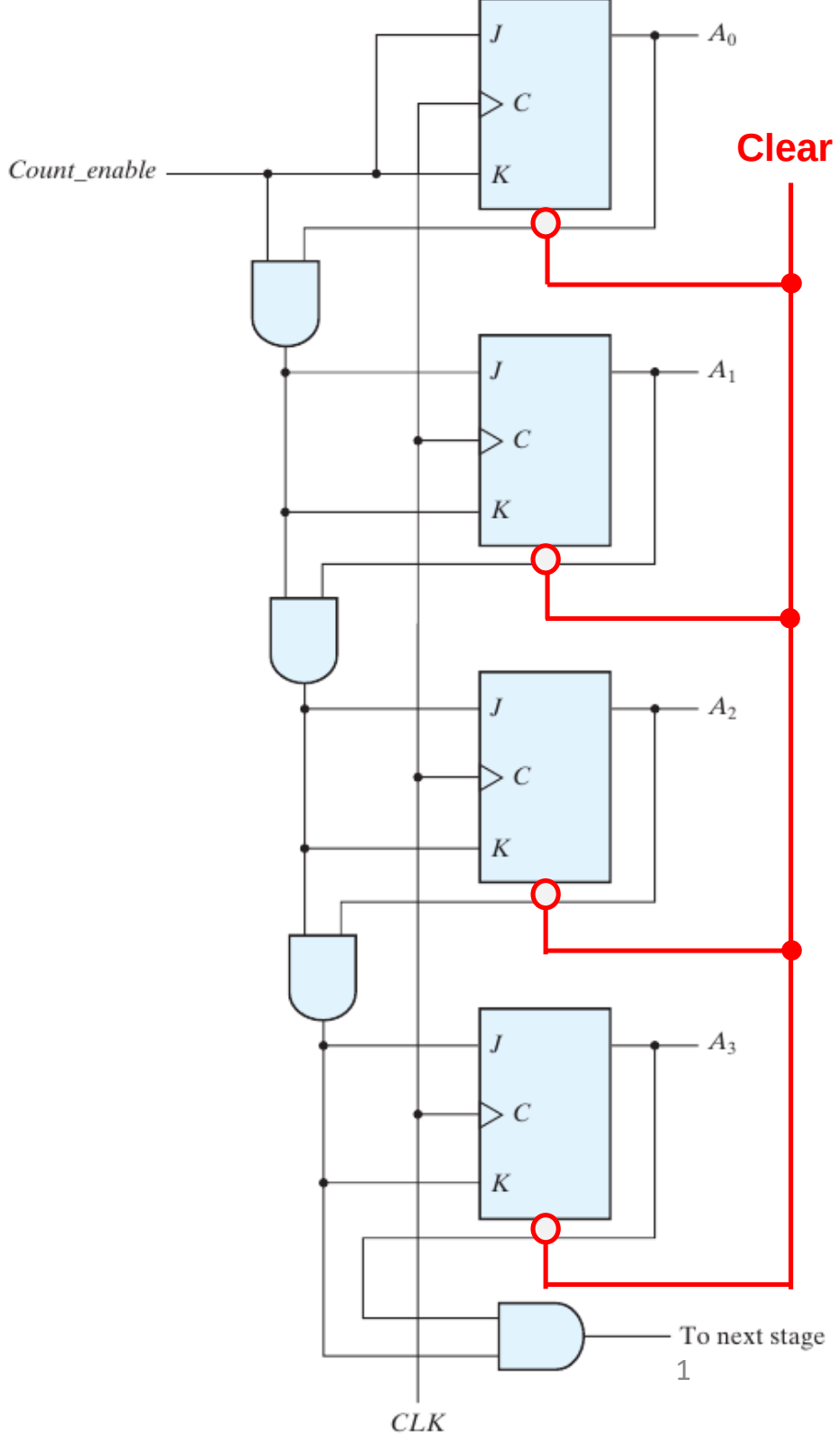
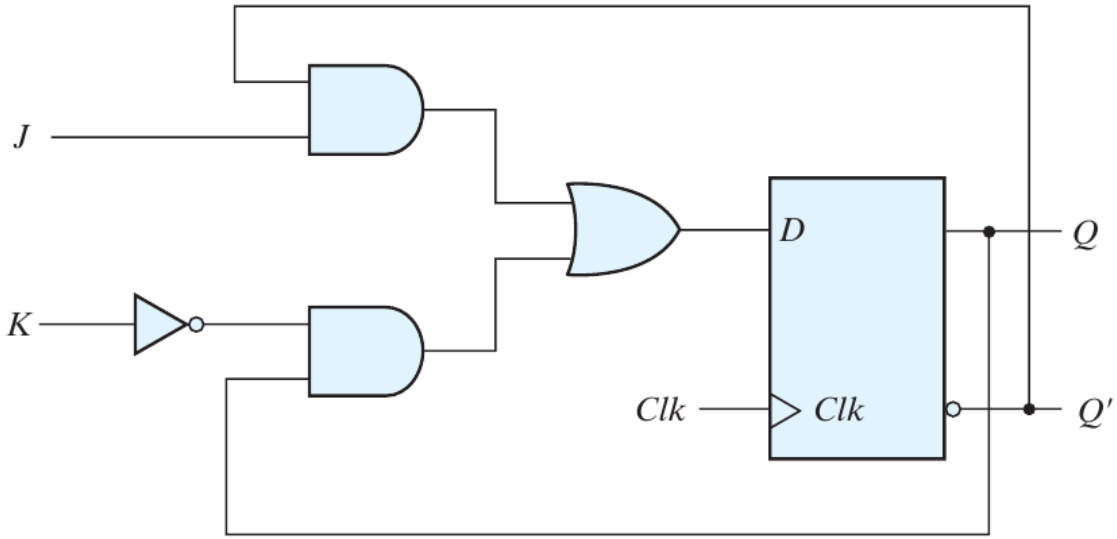
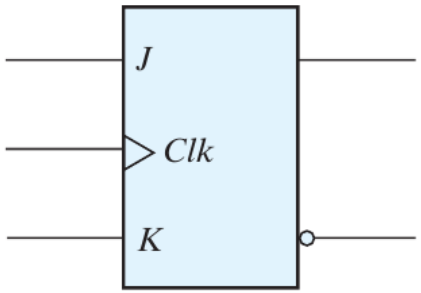
Revisit Synchronous Binary Counter

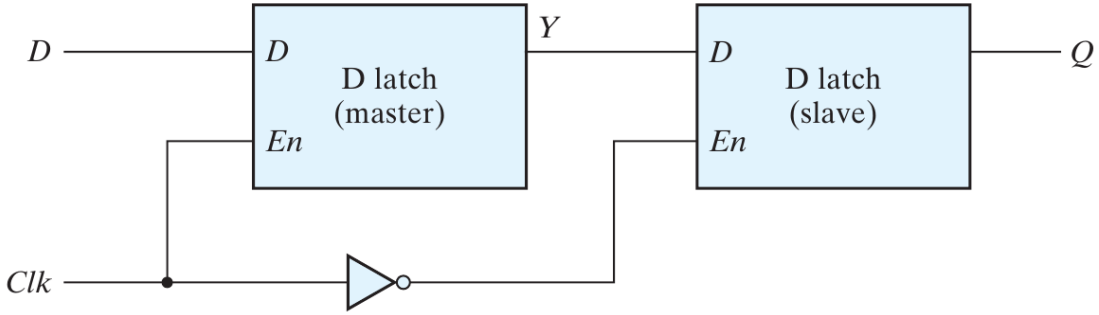
- Can we reset the counter to 0000?

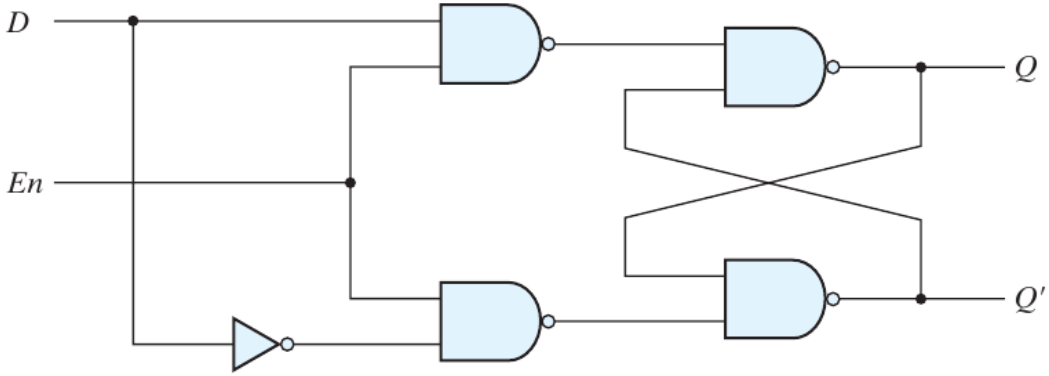


You Remember Resettable FF?





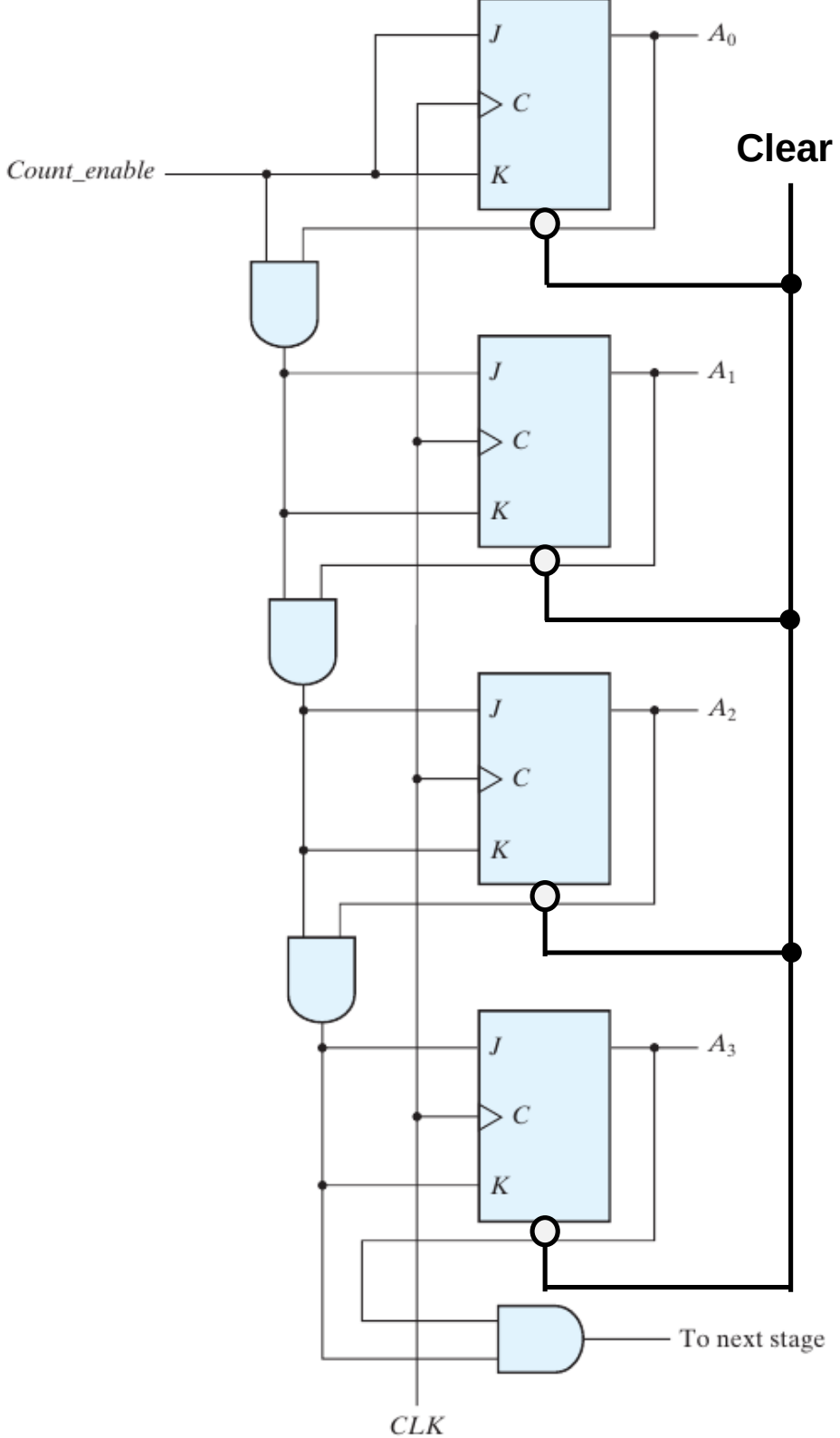




Okay We Have Reset Function Now.

- Can you reset the counter to the arbitrary value, instead of all 0's?

➔ Parallel load function



Chap 6 (5)
Synchronous Counter w/ Parallel Load

Hanwool Jeong

hwjeong@kw.ac.kr

Binary Counter with Parallel Load

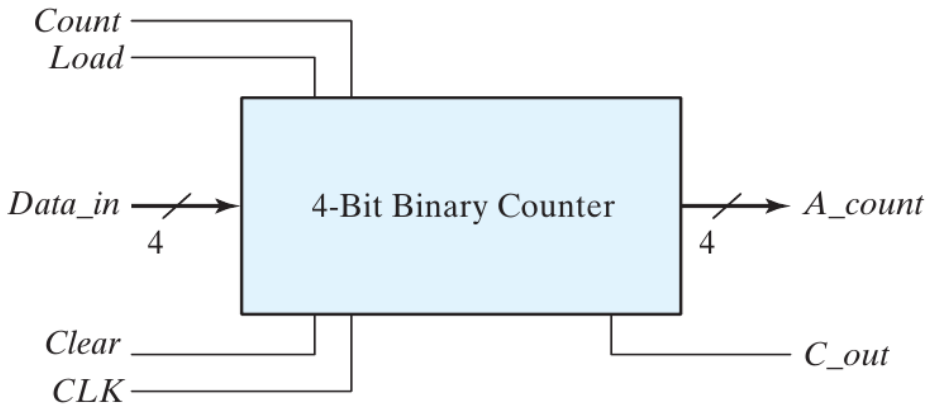
- Counters employed in digital systems quite often require a parallel-load capability for transferring an initial binary number into the counter prior to the count operation.
- Then, we need mode selection signal. How?
- Input pin?
- Output pin?

Function Table for Binary Counter with Parallel Load

- Can you see priority?
- Asynchronous/synchronous signal?

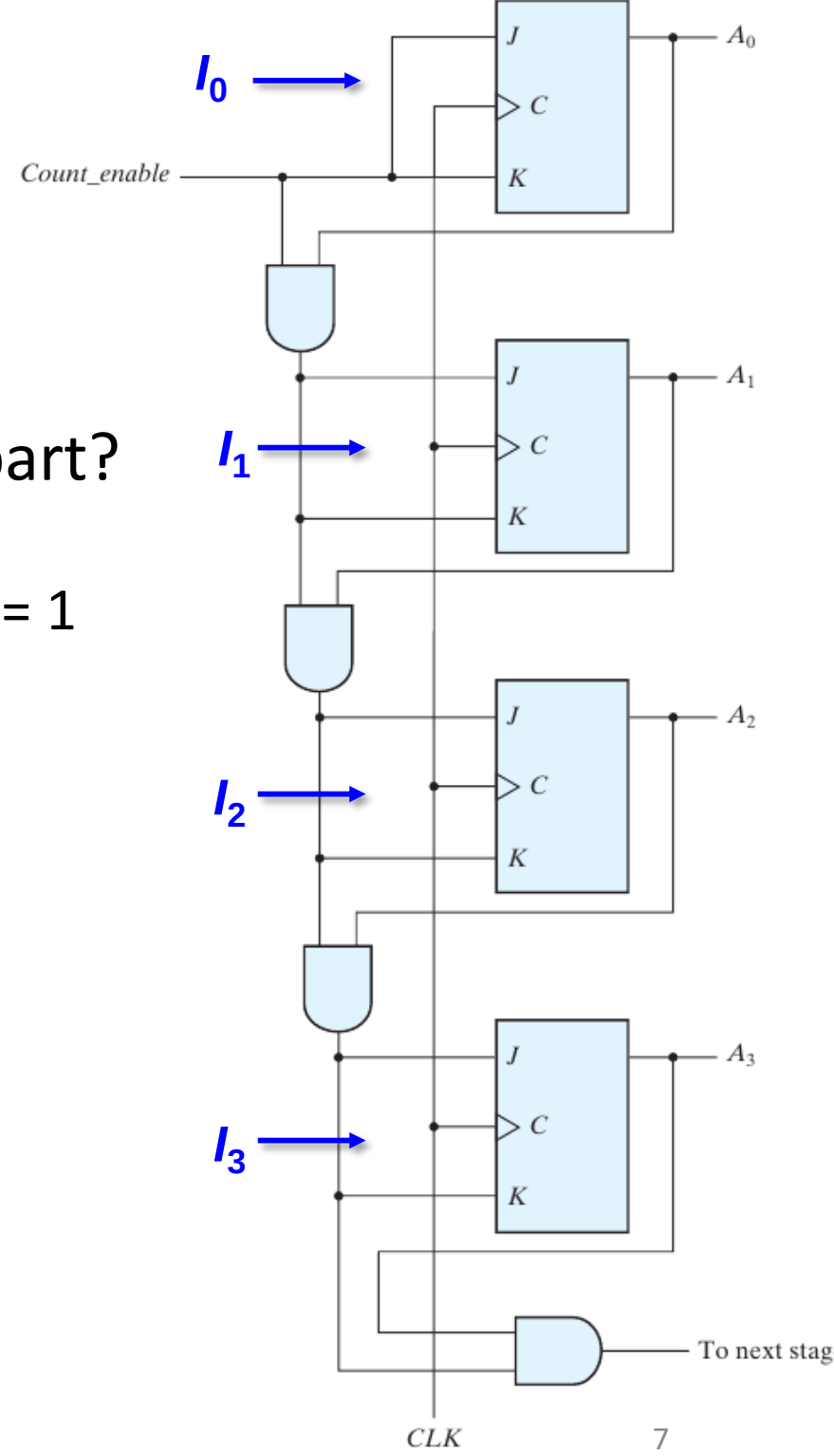
Table 6.6
Function Table for the Counter of Fig. 6.14

Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change



Starting From Binary Counter

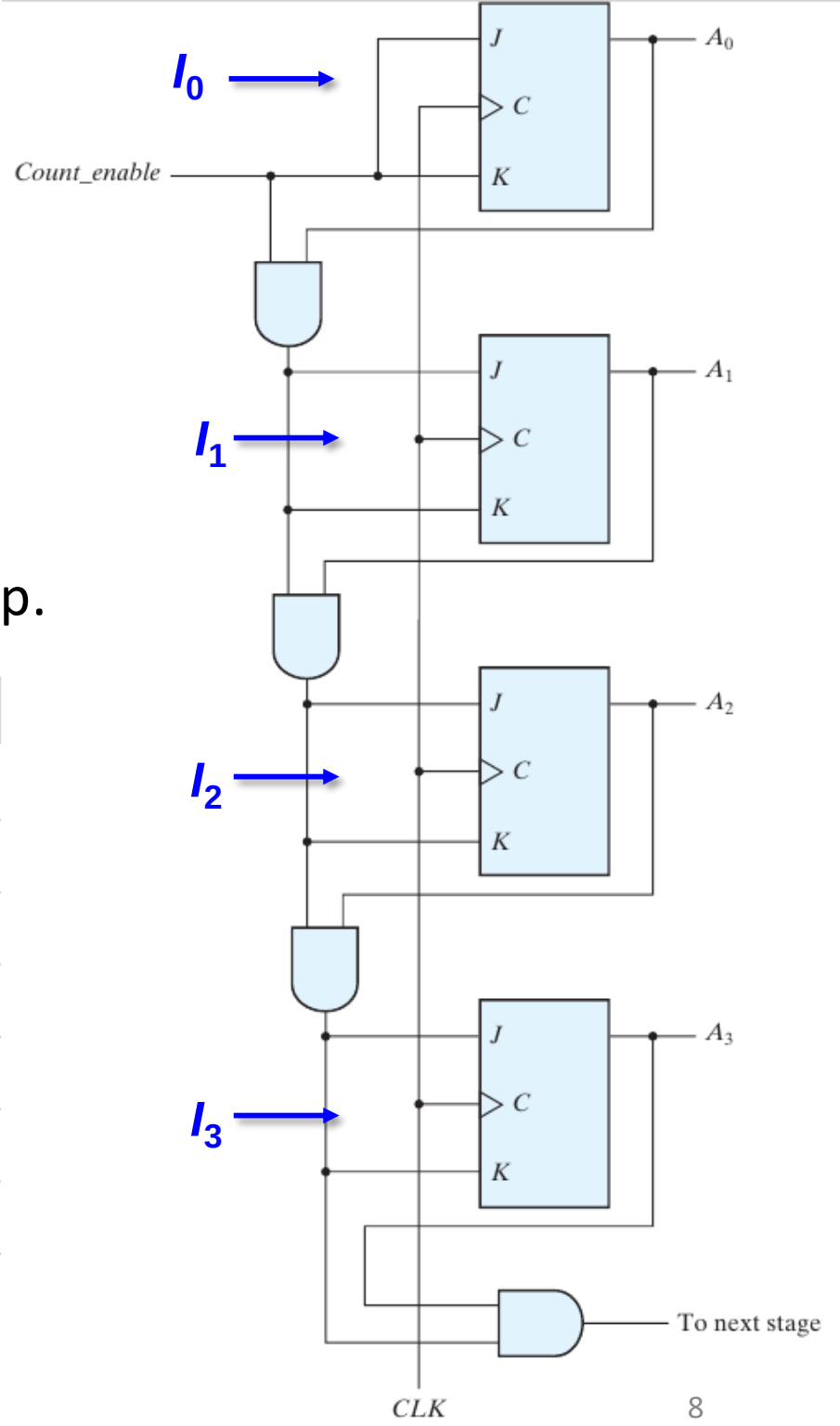
- How should we revise the counter?
- How can we design the mode select part?
  - Load is higher priority input
    ➔ Count should be blocked when Load = 1



J K Input for A_0

- Let's start with A_0
- How can we design J & K?
  - When Load = 1, $I_3I_2I_1I_0$ should be reflected into FFs, rather than count up.

I_0	Load	Count_En	J_0	K_0
0	0	0	**0**	**0**
0	0	1	**1**	**1**
0	1	0	**0**	**1**
0	1	1	**0**	**1**
1	0	0	**0**	**0**
1	0	1	**1**	**1**
1	1	0	**1**	**0**
1	1	1	**1**	**0**



J K Input for A_0

I_0	Load	Count_En	J_0	K_0
0	0	0	**0**	**0**
0	0	1	**1**	**1**
0	1	0	**0**	**1**
0	1	1	**0**	**1**
1	0	0	**0**	**0**
1	0	1	**1**	**1**
1	1	0	**1**	**0**
1	1	1	**1**	**0**

J_0

	00	01	11	10
0	0	1	0	0
1	0	1	1	1

K_0

	00	01	11	10
0	0	1	1	1
1	0	1	0	0

Smarter Way

I_0	Load	Count_En	J_0	K_0
0	0	0	**0**	**0**
0	0	1	**1**	**1**
0	1	0	**0**	**1**
0	1	1	**X**	**X**
1	0	0	**0**	**0**
1	0	1	**1**	**1**
1	1	0	**1**	**0**
1	1	1	**X**	**X**

Load	Count_En	J_0	K_0
0	0	**0**	**0**
0	1	**1**	**1**
1	0	**I_0**	**I_0'**
1	1	**X**	**X**

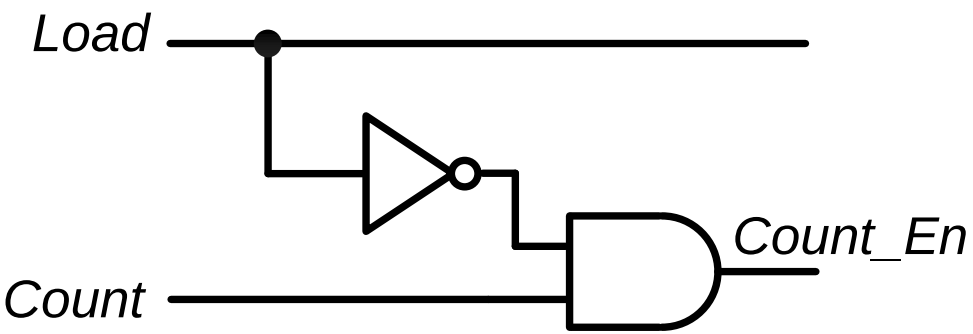
J_0	0	1
0	0	1
1	**I_0**	X

K_0	0	1
0	0	1
1	**I_0'**	X

J_0	00	01	11	10
0	0	1	**X**	0
1	0	1	**X**	1

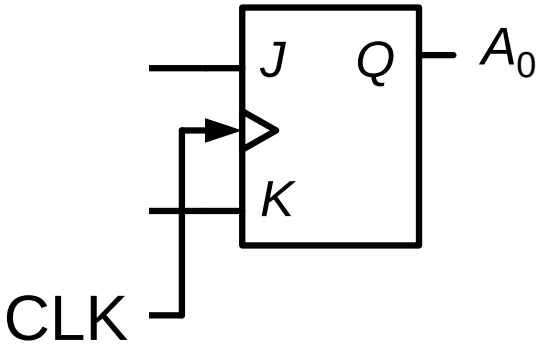
K_0	00	01	11	10
0	0	1	**X**	1
1	0	1	**X**	0

So Far,



$J_0 = I_0 \times L + C$

$K_0 = I_0' \times L + C$

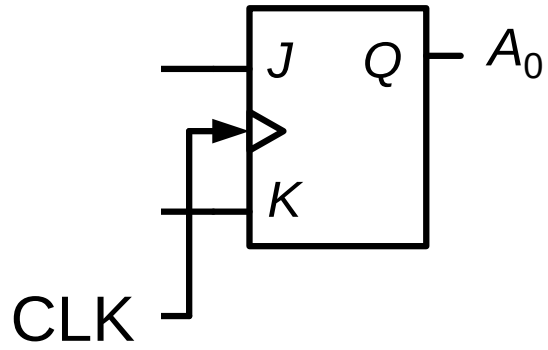


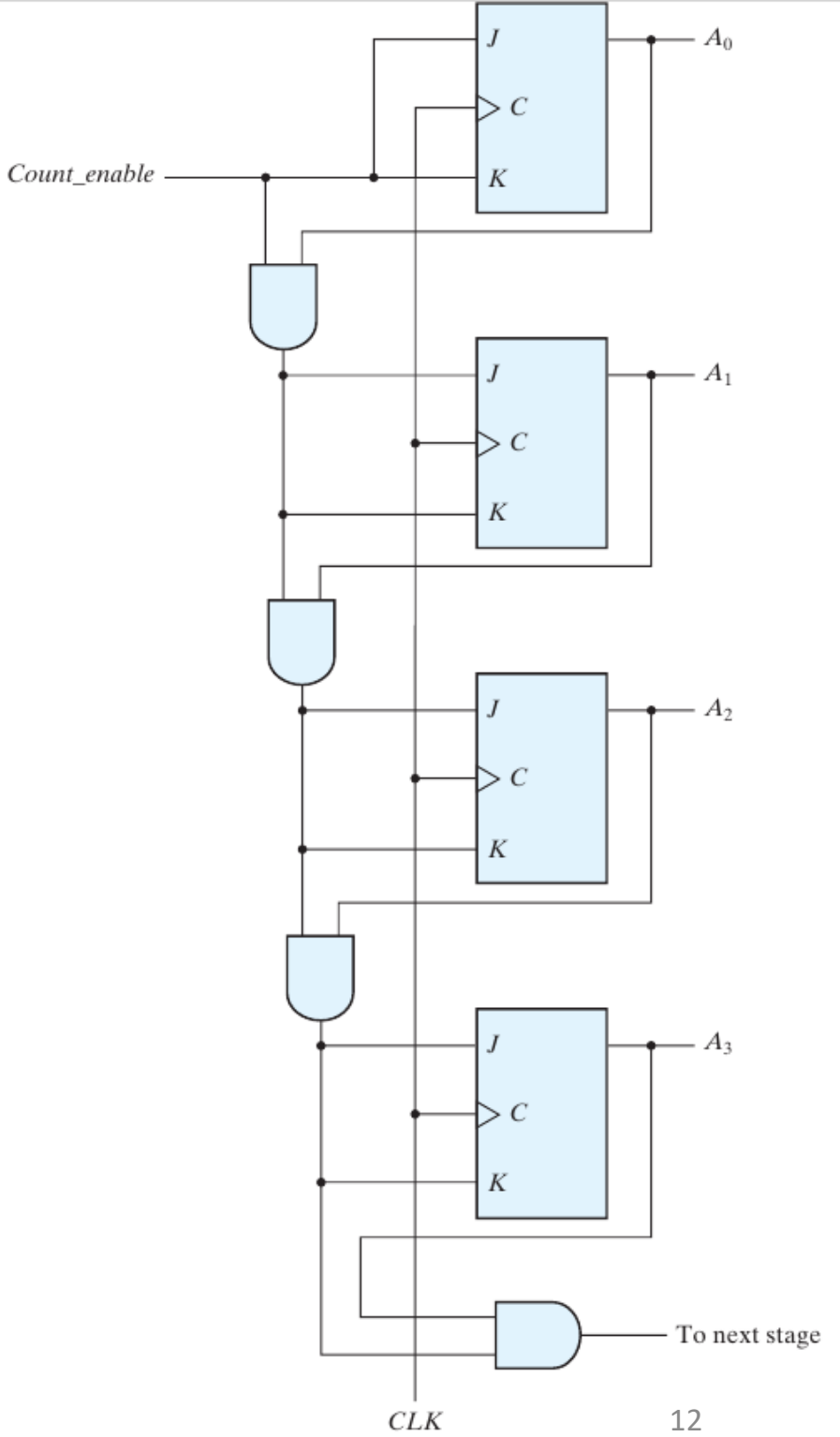
J K Input for A_1

- How about A_1?

Load	Count_En	J_1	K_1
0	0	**0**	**0**
0	1	**A_0**	**A_0**
1	0	**I_1**	**I_1'**
1	1	**X**	**X**

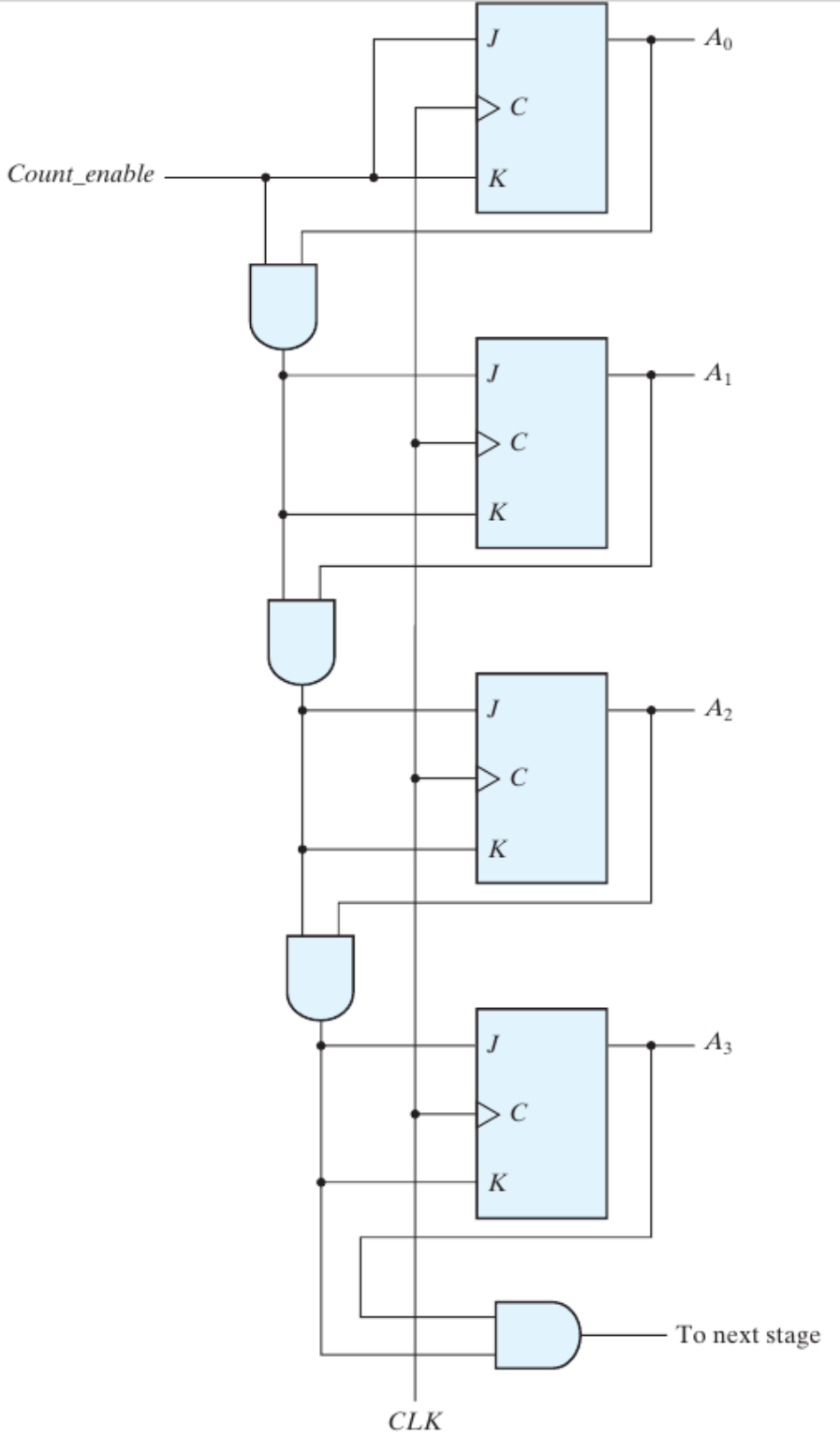
$$J_1 = I_1 \times L + CA_0$$
$$K_1 = I_1' \times L + CA_0$$



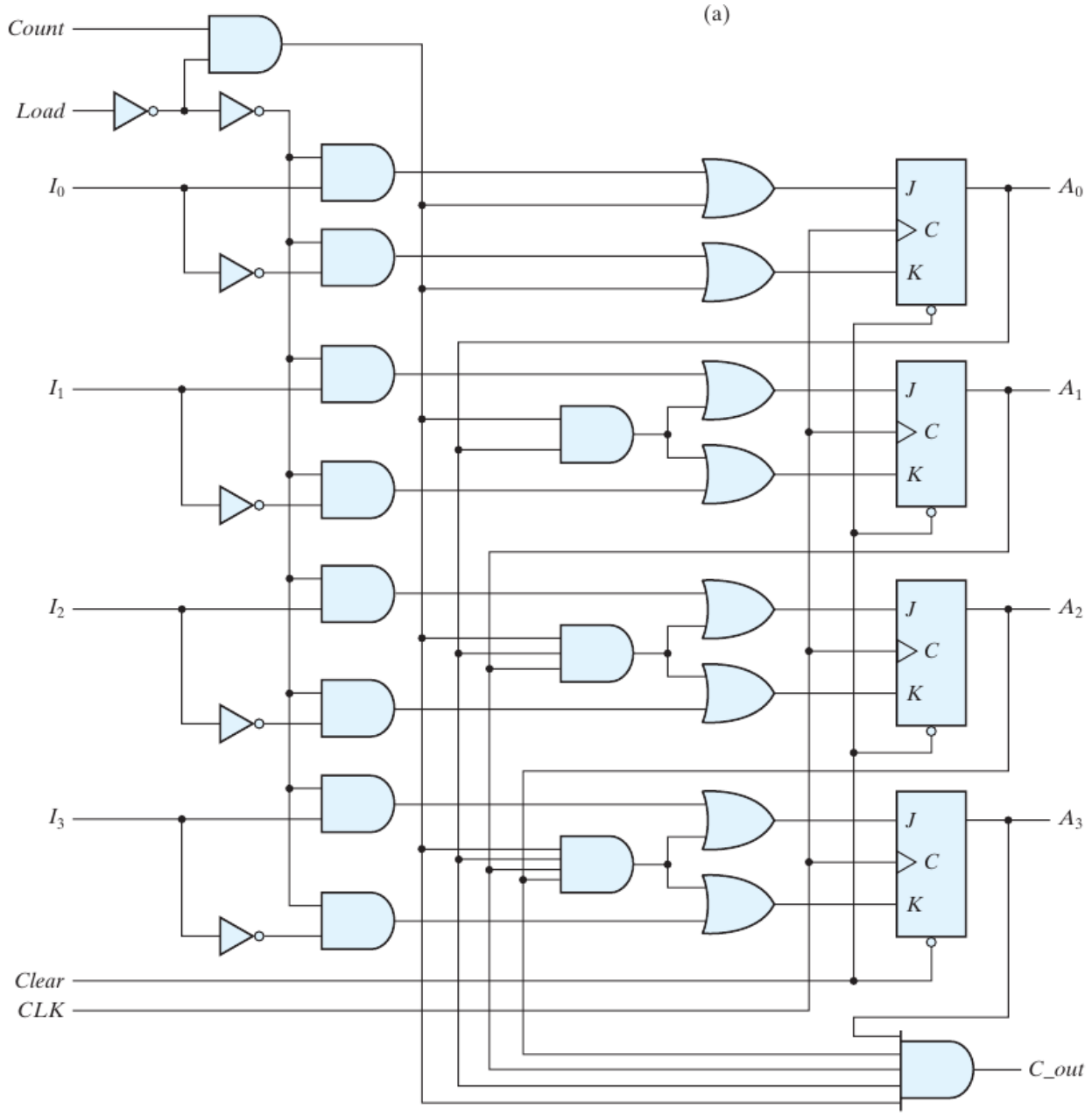


J K Input for A_2

Load	Count_En	J_2	K_2
0	0	**0**	**0**
0	1		
1	0	I_2	I_2'
1	1	**X**	**X**



4-bit Binary Counter w/ Parallel Load



Checkpoints

- ✓Parallel load function implementation
- ✓Handling priority signals
- ✓Smarter way for truth table
- ✓Coming-up next : Various counters
  - Counter w/ unused states
  - Ring counter
  - Johnson counter