Digital Circuits

ECS 371

Dr. Prapun Suksompong

prapun@siit.tu.ac.th

Lecture 20

Office Hours:
BKD 3601-7
Monday 9:00-10:30, 1:30-3:30
Tuesday 10:30-11:30

ECS371.PRAPUN.COM

Announcement

- Reading Assignment:
  - Chapter 7: 7-1, 7-2, 7-4
  - Chapter 8: 8-1, 8-2, 8-4
- Potentially no ECS371 lectures next week (24-28 Aug)!
  - Due date for HW7: Aug 26 (Wednesday).
  - Check the course web site regularly for announcement.

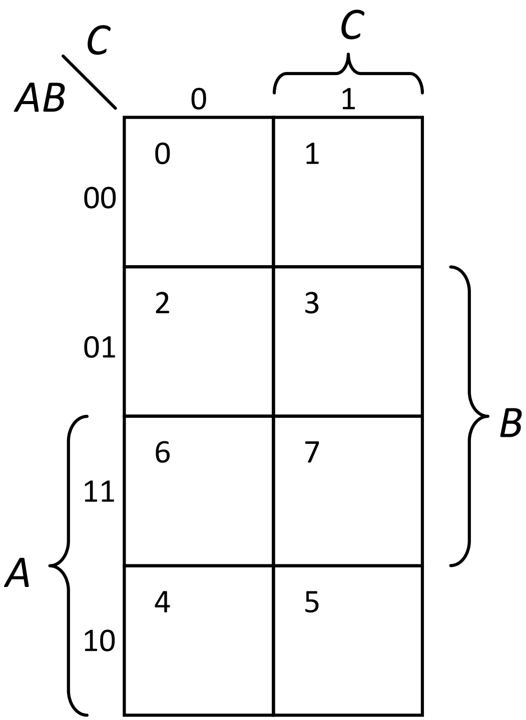
Caution!

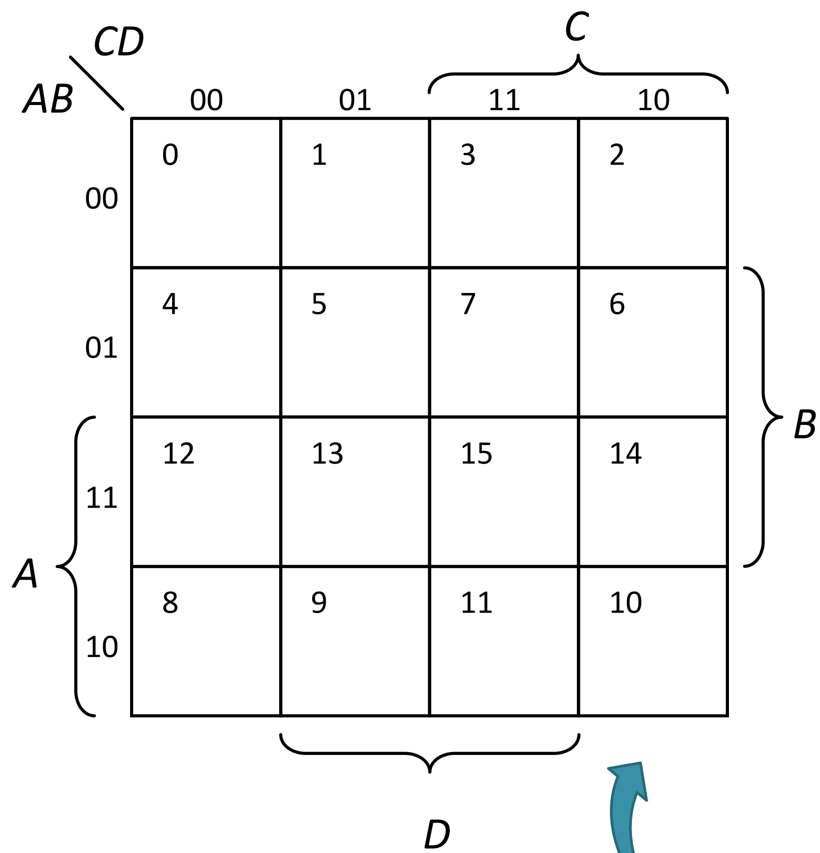
- The analysis and design of synchronous counter is significantly different from the analysis and design of asynchronous counter
- Recall: Suppose we want to count from 0 to 9. For asynchronous counter,
  - we let the number 10 shows up first,
  - we detect it with a simple decoder,
  - then we use the detection to *asynchronously* clear all the FFs back to 0.
  - In other words, the number 10 actually shows up on the output but only for a short time (so short that you many not see it in the real circuit.)

Synchronous Counters (2)

- We won't use the same technique (of asynchronously zeroing all FFs) in synchronous design.
  - Reason: It is asynchronous!
- We want to change the value of the states/outputs of the FFs only at the rising edge of the clock.
- In other words, if we let the number 10 shows up, it will be there until the next rising edge of the clock.
- So, we need to come up with a new technique.
  - Tabular analysis.

Recall: Karnaugh Map





Row #	A	B	C
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

The small number inside each cell is the corresponding row number in the truth table, assuming that the truth table inputs are labeled alphabetically from left to right (e.g. A, B, C) and the rows are numbered in binary counting order.

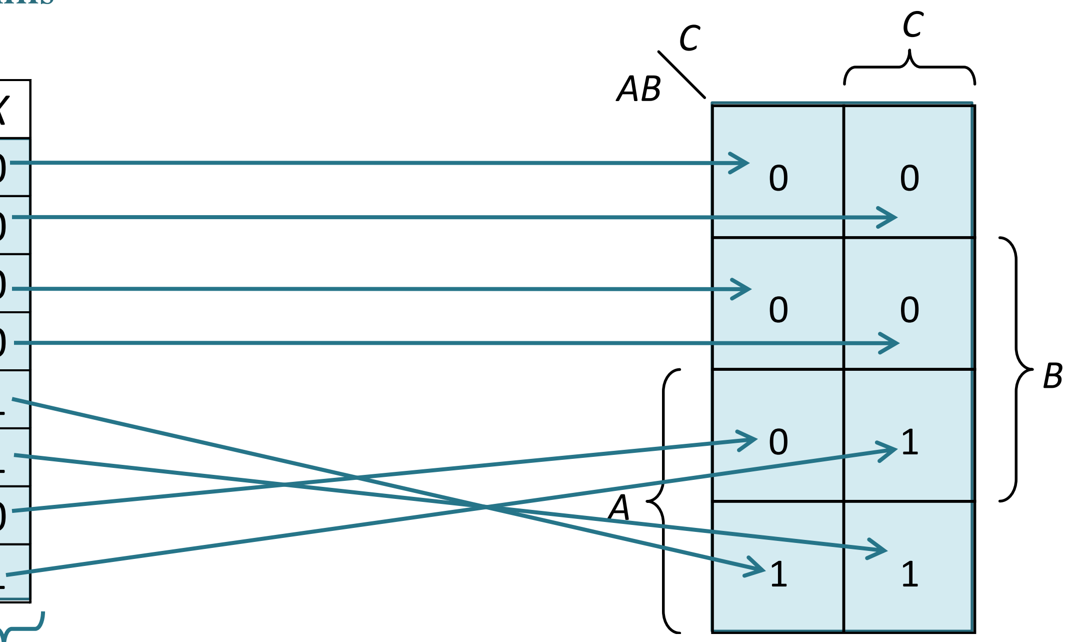
Row #	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Recall: Truth Table v.s. K-map

Input Columns

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Output Column

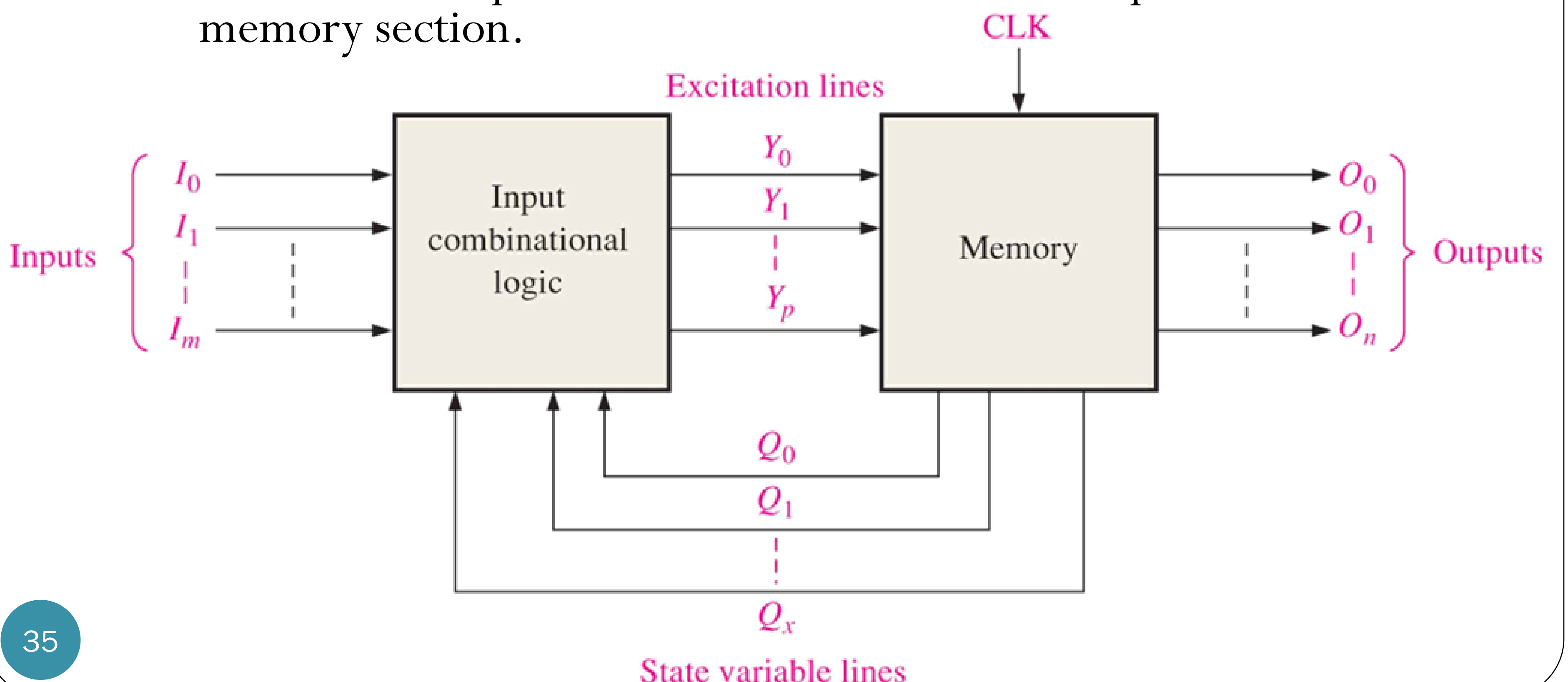


Counter Design Technique

- We want to design counters with arbitrary sequences
- We will discuss in detail a commonly used technique for designing synchronous counters using J-K flip-flops or D flip-flops.
- The design of the counters basically involves designing a suitable combinational logic circuit that takes its inputs from the normal and complemented outputs of the FFs used and decodes the different states of the counter to generate the correct logic states for the inputs of the FFs such as J, K, D, etc.
- We will start by learning sequential circuit design techniques.

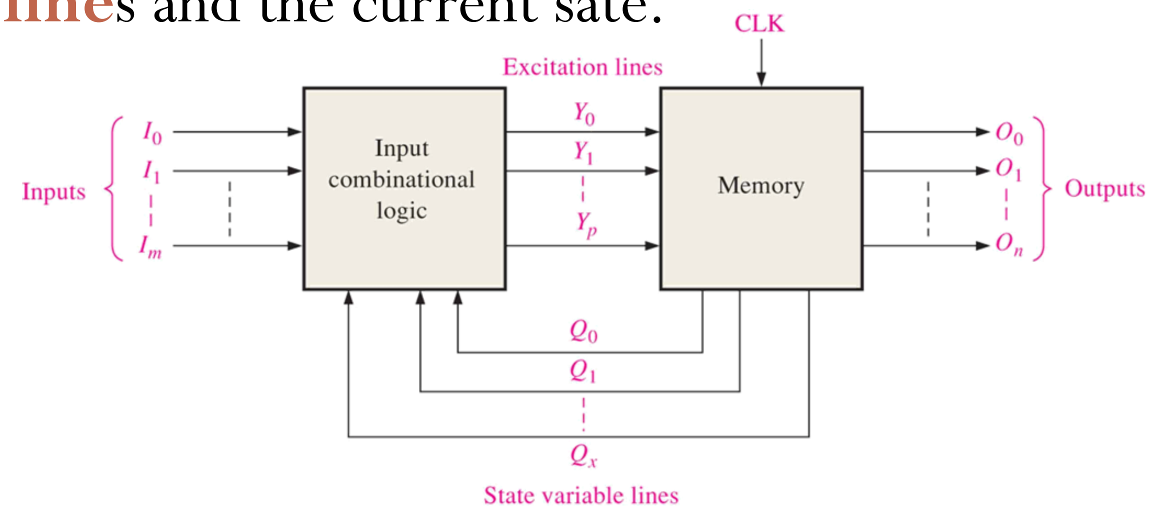
Sequential Circuits

- Also known as **state machine**.
- A general sequential circuit consists of a combinational logic section and a memory section (FFs).
- In a clocked sequential circuit, there is a clock input to the memory section.



Sequential Circuits

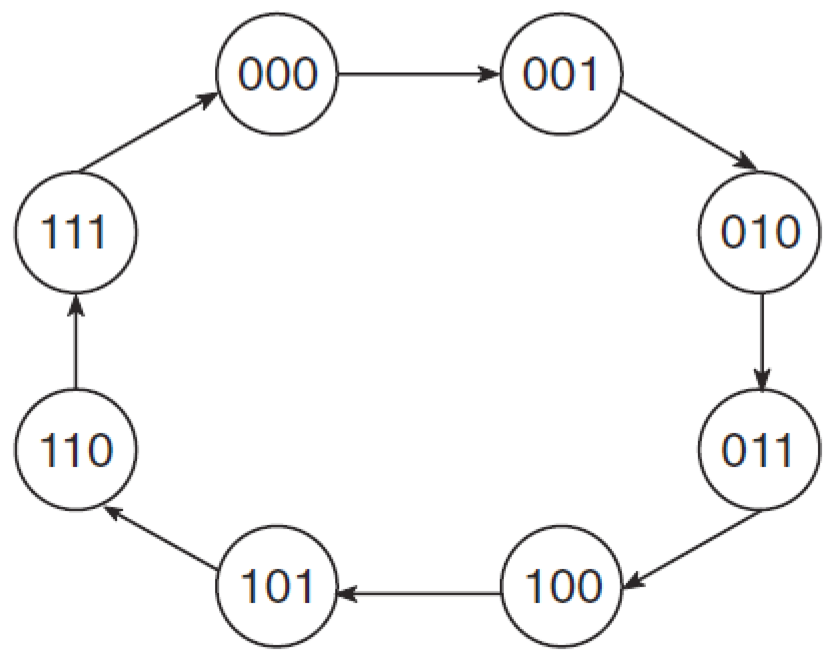
- At any given time, the memory is in a state called the **present/current state**.
  - The present state is represented by the **state variable**s (Q_0, Q_1, Q_2, Q_3,…)
- It will advance to the **next state** on the clock pulse.
- The next state is determined by the conditions on the **excitation line**s and the current sate.



State Transition Diagram

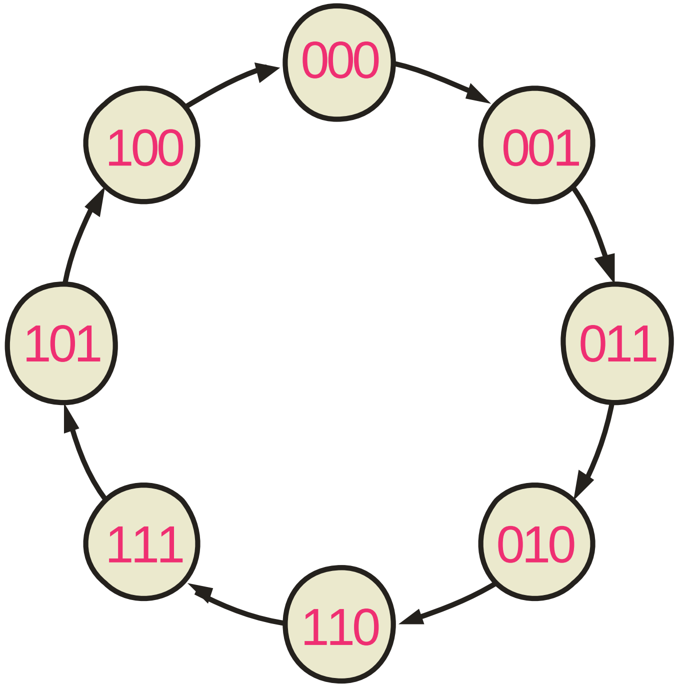
- The state transition diagram is a graphical representation of different states of a given sequential circuit and the sequence in which these states occur in response to a clock input.
- Different states are represented by circles, and the arrows joining them indicate the sequence in which different states occur.

Ex. State transition diagram for a MOD-8 binary counter.



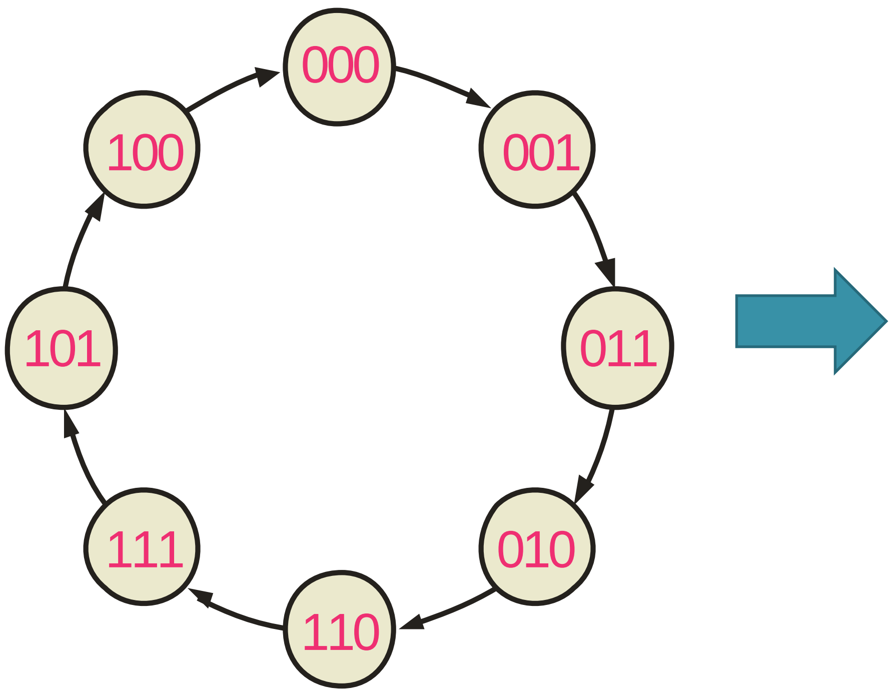
Step 1: State Diagram

- Specify the counter sequence and draw a state diagram.
- As an example, here is a state diagram for a 3-bit Gray code counter.



Step 2: Next-State Table

- List each state of the counter (current state) along with the corresponding next state.



Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0

Step 3: FF Excitation Inputs

- Find the J and K inputs required for the transitions in the Next-State Table

Current State			*Next State*			*FF_2*		*FF_1*		*FF_0*	
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1						
0	0	1	0	1	1						
0	1	1	0	1	0						
0	1	0	1	1	0						
1	1	0	1	1	1						
1	1	1	1	0	1						
1	0	1	1	0	0						
1	0	0	0	0	0						

Step 3: FF Excitation Inputs

- Find the J and K inputs required for the transitions in the Next-State Table

Rearranged to our familiar form

Current State			*Next State*			FF_2		FF_1		FF_0	
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1						
0	0	1	0	1	1						
0	1	0	1	1	0						
0	1	1	0	1	0						
1	0	0	0	0	0						
1	0	1	1	0	0						
1	1	0	1	1	1						
1	1	1	1	0	1						

We will first develop a transition table showing the FF inputs required for each transition.

FF Excitation/Transition Table

The excitation table lists the present state, the desired next state and the flip-flop inputs (J, K, D, etc.) required to achieve that.

Excitation table of a J-K flip-flop.

Present state (Q_n)	Next state (Q_{n+1})	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Excitation table of a D flip-flop.

Present state (Q_n)	Next state (Q_{n+1})	D
0	0	0
0	1	1
1	0	0
1	1	1

An X indicates a “don't care” (the input can be either a 1 or a 0).

The transition table is always the same for a given type of flip-flop.

Step 3: Find Excitation Inputs to FFs

For the current state 000, Q_0 goes from a present state of 0 to a next state of 1. To make this happen, J_0 must be a 1 and you don't care what K_0 is ($J_0 = 1$, $K_0 = X$),

Output Transitions Q_N	Q_{N+1}	Flip-Flop Inputs J	K
0 →	0	0	X
0 →	1	1	X
1 →	0	X	1
1 →	1	X	0

Current State			Next State			FF_2		FF_1		FF_0	
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1					1	X
0	0	1	0	1	1					X	0
0	1	0	1	1	0					0	X
0	1	1	0	1	0					X	1
1	0	0	0	0	0					0	X
1	0	1	1	0	0					X	1
1	1	0	1	1	1					1	X
1	1	1	1	0	1					X	0

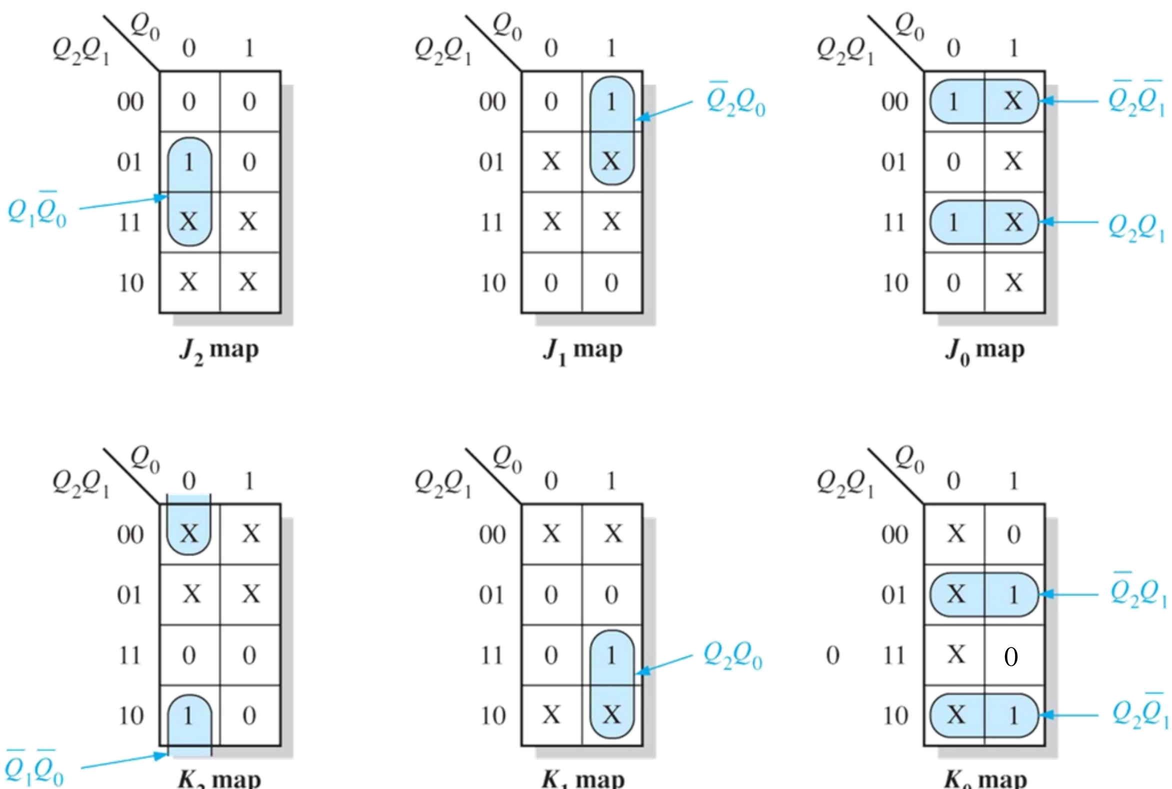
Step 3: Find Excitation Inputs to FFs

Output Transitions Q_N	Q_{N+1}	Flip-Flop Inputs J	K
0 →	0	0	X
0 →	1	1	X
1 →	0	X	1
1 →	1	X	0

Current State			Next State			FF_2		FF_1		FF_0	
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	1	0	X	1	X	X	0
0	1	0	1	1	0	1	X	X	0	0	X
0	1	1	0	1	0	0	X	X	0	X	1
1	0	0	0	0	0	X	1	0	X	0	X
1	0	1	1	0	0	X	0	0	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	1	0	1	X	0	X	1	X	0

Step 4: Karnaugh Maps

- Karnaugh maps can be used to determine the logic required for the J and K inputs of each flip-flop.



FF_0	
J_0	K_0
1	X
X	0
0	X
X	1
0	X
X	1
1	X
X	0

There is a Karnaugh map for each input of each FF.

Step 5: Logic Expressions for FF Inputs

- From the Karnaugh maps, we group the cells to generate the logic expression for each FF input:

$$J_0 = Q_2 Q_1 + \overline{Q_2} \cdot \overline{Q_1} = \overline{Q_2 \oplus Q_1}$$

$$K_0 = Q_2 \overline{Q_1} + \overline{Q_2} Q_1 = Q_2 \oplus Q_1$$

$$J_1 = \overline{Q_2} Q_0$$

$$K_1 = Q_2 Q_0$$

$$J_2 = Q_1 \overline{Q_0}$$

$$K_2 = \overline{Q_1} \cdot \overline{Q_0}$$

$J_0 = \overline{Q_2 \oplus Q_1}$

$K_0 = Q_2 \oplus Q_1$

$J_1 = \overline{Q_2} Q_0$

$K_1 = Q_2 Q_0$

$J_2 = Q_1 \overline{Q_0}$

$K_2 = \overline{Q_1} \cdot \overline{Q_0}$

Step 6: Counter Implementation

- Implement the expressions with combinational logic, and combine with the FFs to create the counter.