

CDA 3200 Digital Systems

Instructor: Dr. Janusz Zalewski

Developed by: Dr. Dahai Guo

Spring 2012

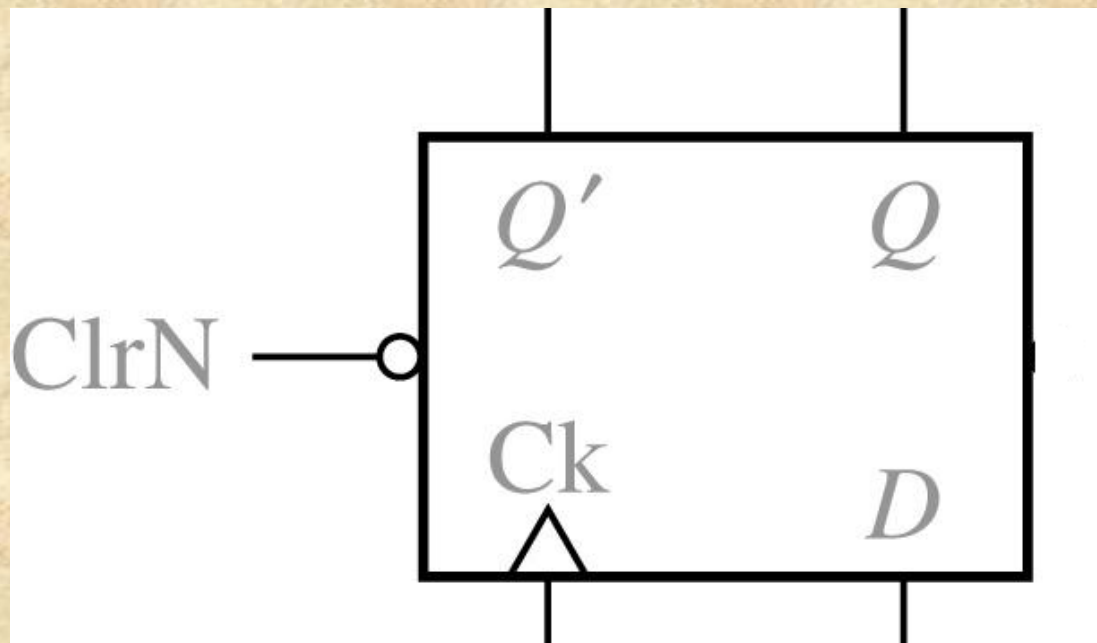
Outline

- Registers and Register Transfers
- Shift Registers
- Design of Binary Counters
- Counters for Other Sequences
- Counter Design Using SR and JK Flip-Flop (FFs)
- Derivation of Flip-Flop (FF) Input Equations

Registers and Register Transfers

(1/8)

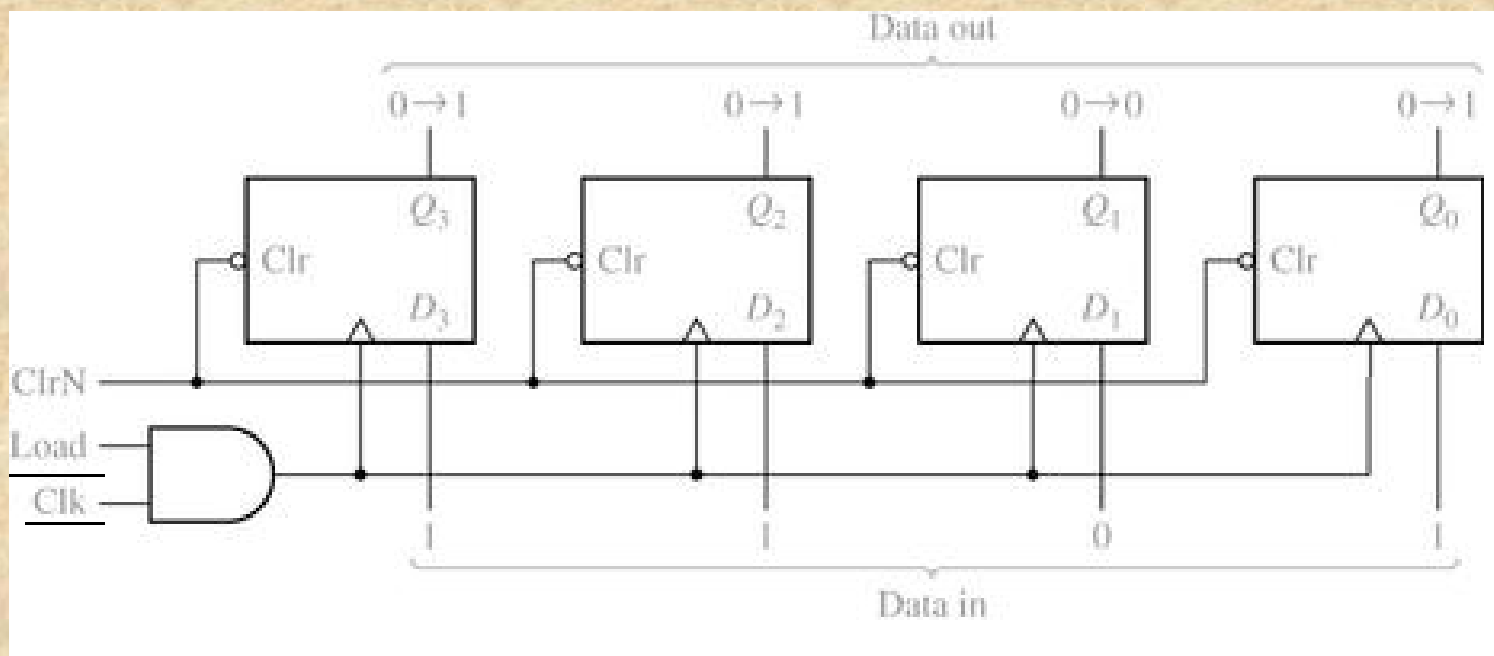
- In a D flip-flop (FF)
 - $Q^+ = D$, triggered on the rising/falling edge
 - ClrN clears Q asynchronously.



Registers and Register Transfers

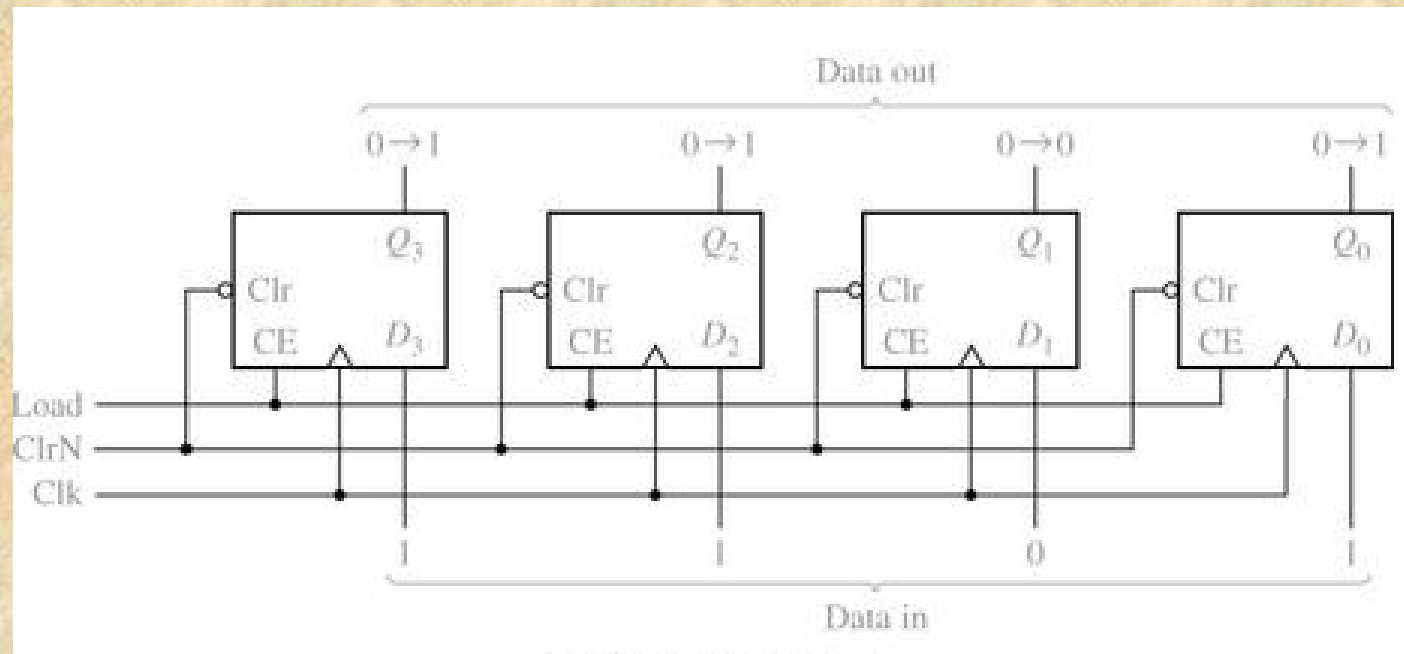
(2/8)

- Each flip-flop (FF) can store one bit of information. Four of them can form a register of 4 bits



Registers and Register Transfers (3/8)

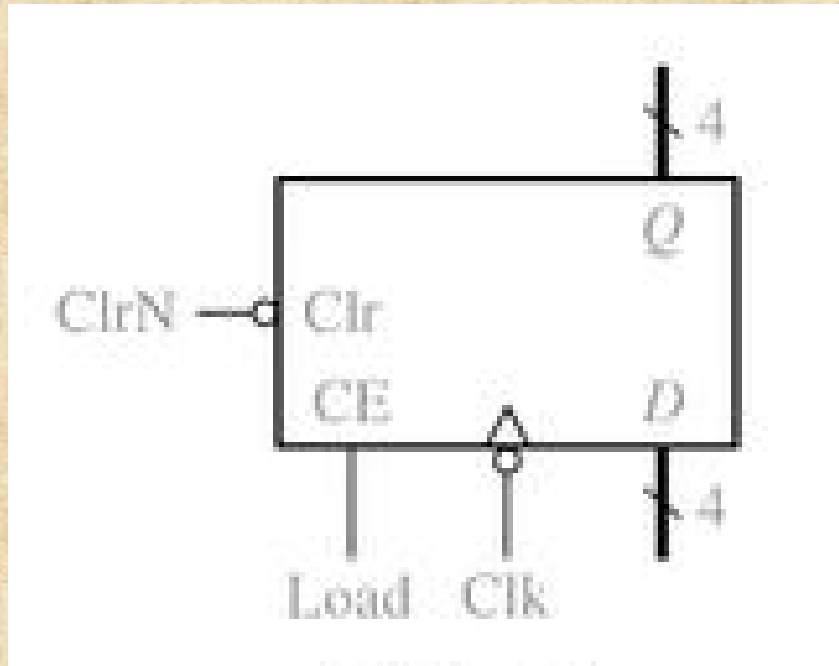
- Alternatively, CE inputs can be used.



Registers and Register Transfers

(4/8)

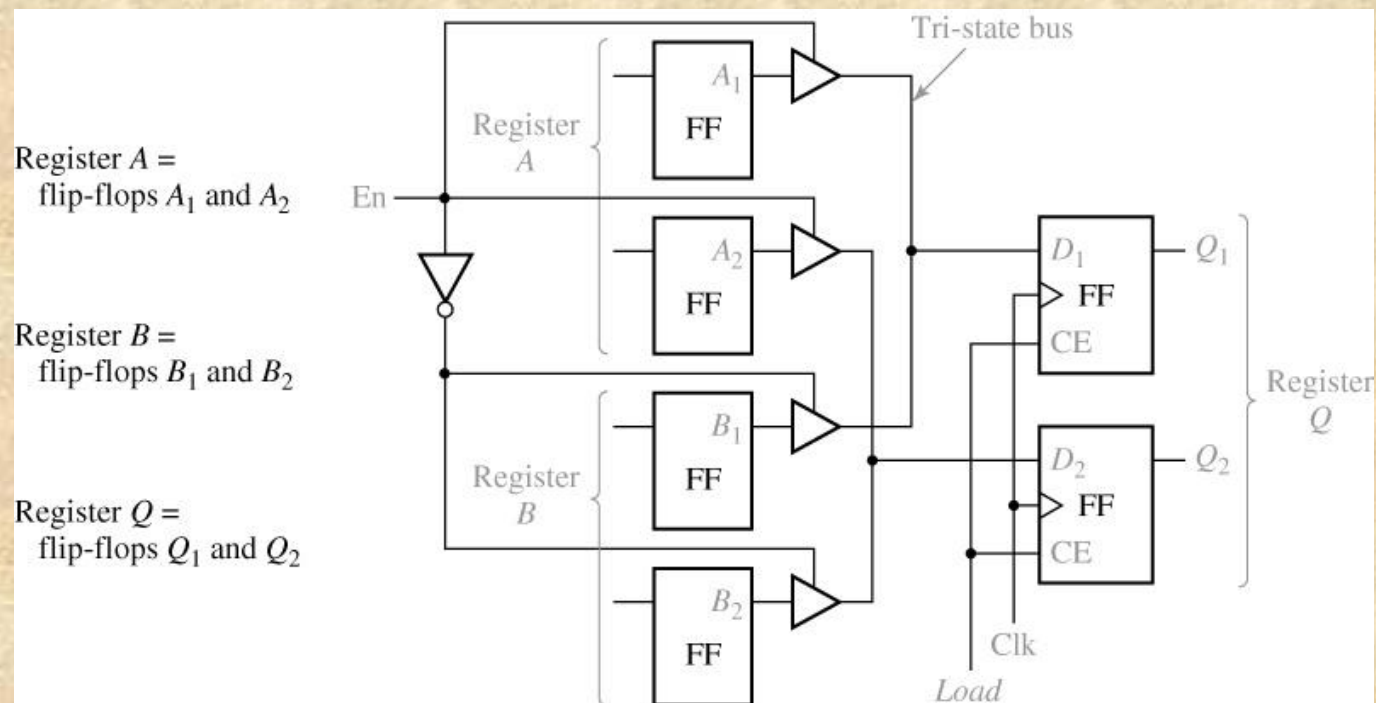
- Bus notation.



Registers and Register Transfers

(5/8)

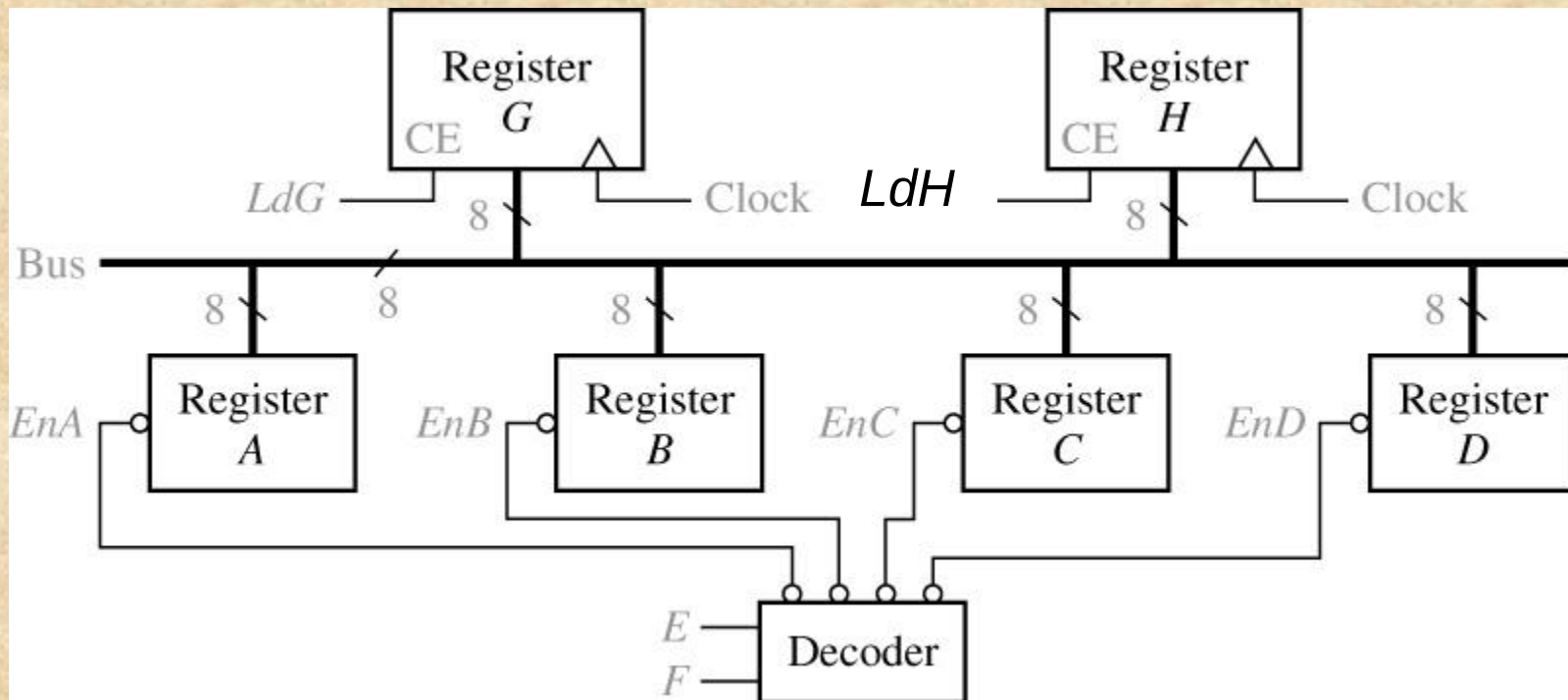
- Example 1: transfer data from one of two registers into a third register.



- $En=1$, what will be stored in Q?
- $En=0$, what will be stored in Q?

Registers and Register Transfers (6/8)

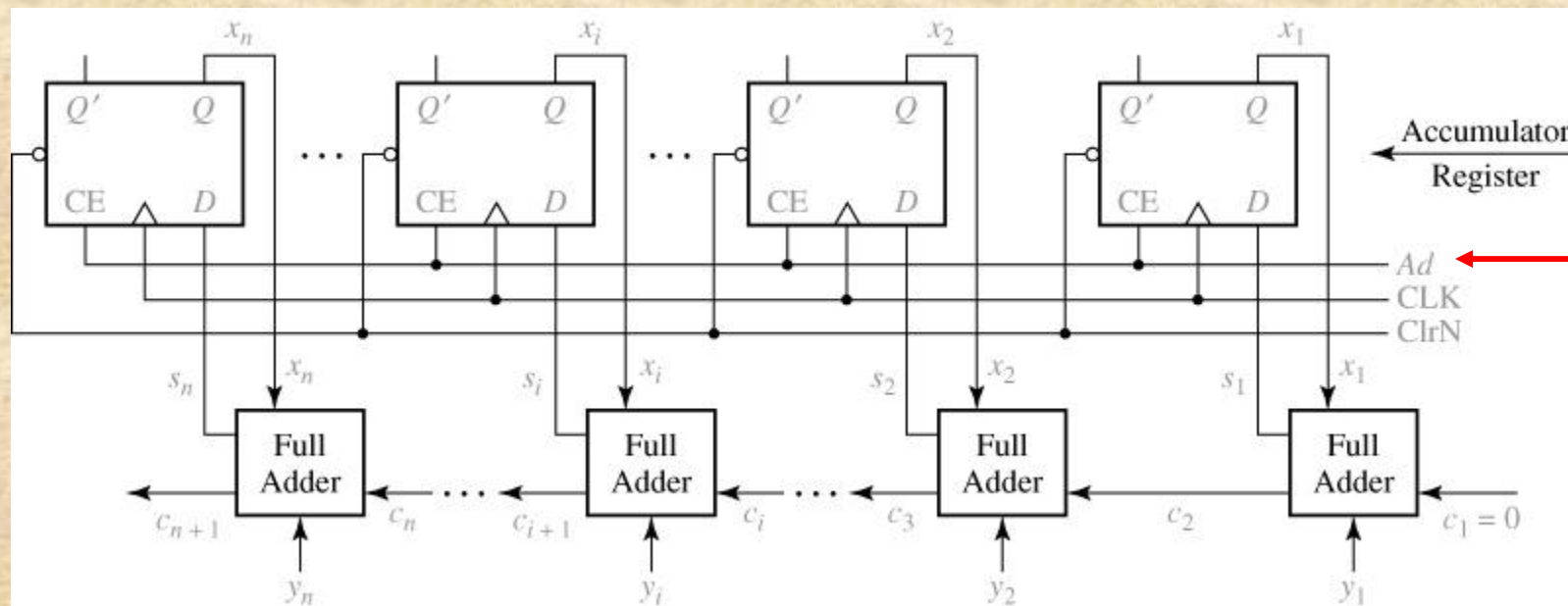
- Example 2: using a decoder in selecting register



- What happens when $E, F, LdG, LdH=1101$?
- What happens when $E, F, LdG, LdH=0011$?

Registers and Register Transfers (7/8)

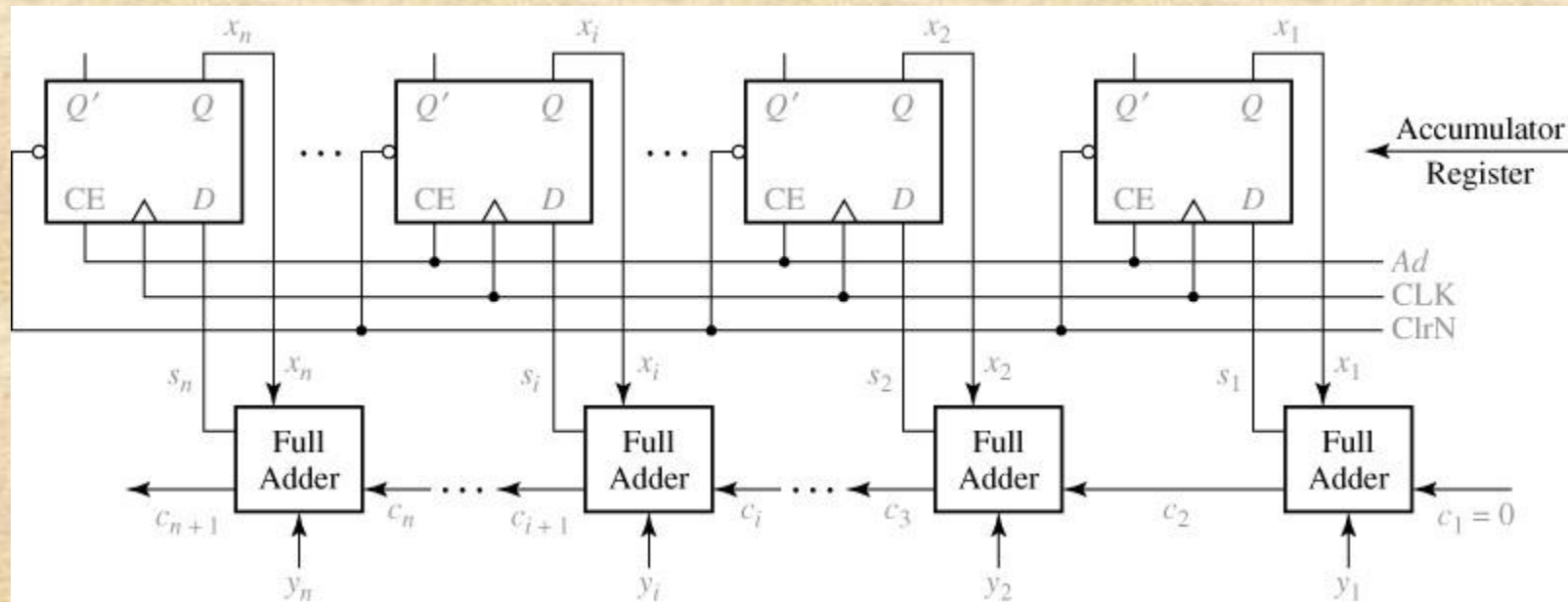
- Accumulator



- Only when $Ad=1$, the number X in the accumulator will be replaced with the sum of X and Y.

Registers and Register Transfers (8/8)

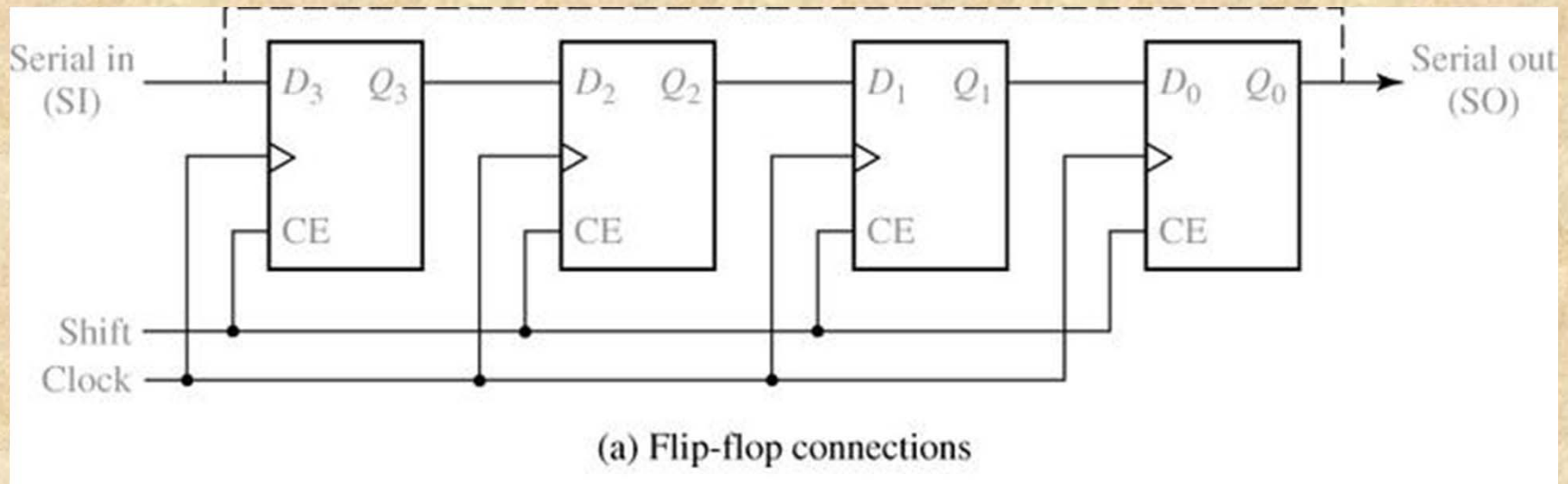
- Accumulator



- How to implement:

- `mov A, 1101` ; save 1101 to the accumulator
- `add A, 0011` ; add 0011 to the accumulator

Shift Registers (1/6)



$$Q^+ = D$$

Initial States:

$$Q_3 = 0$$

$$Q_2 = 1$$

$$Q_1 = 0$$

$$Q_0 = 1$$

Shift=1 enables the clock and assume SI=1



Rising Edge

$$Q_3^+ = D_3 = SI$$

$$Q_2^+ = D_2 = Q_3$$

$$Q_1^+ = D_1 = Q_2$$

$$Q_0^+ = D_0 = Q_1$$

$$Q_3^+ = 1$$

$$Q_2^+ = 0$$

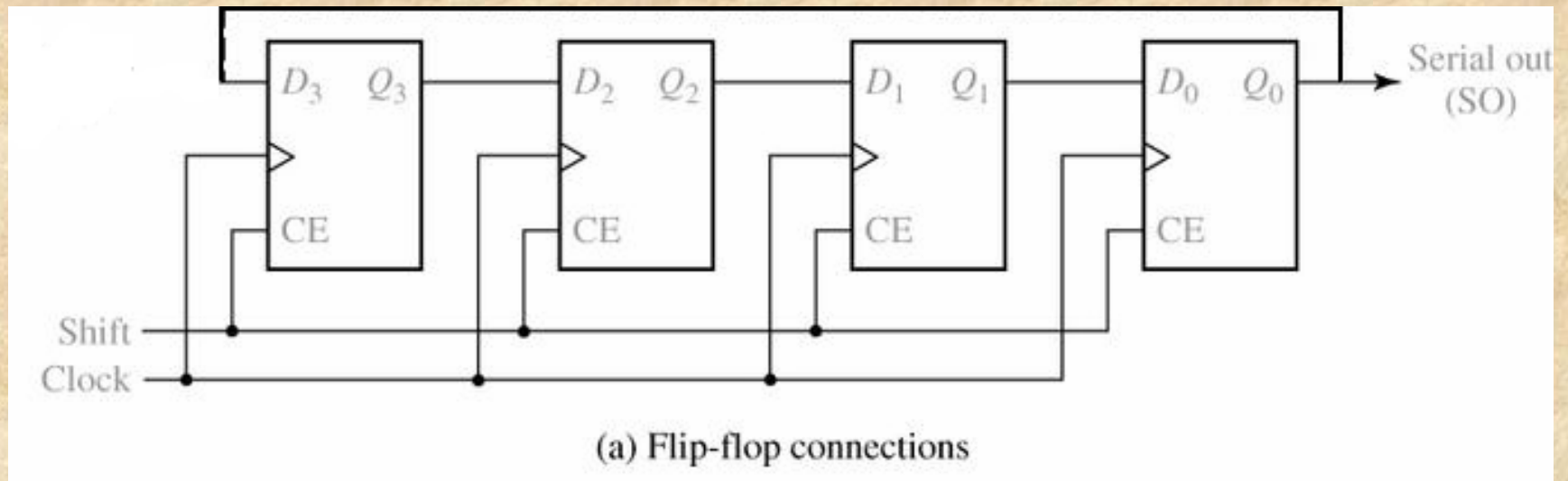
$$Q_1^+ = 1$$

$$Q_0^+ = 0$$

SI is shifted in.

Q₀ is shifted out.

Shift Registers (2/6)



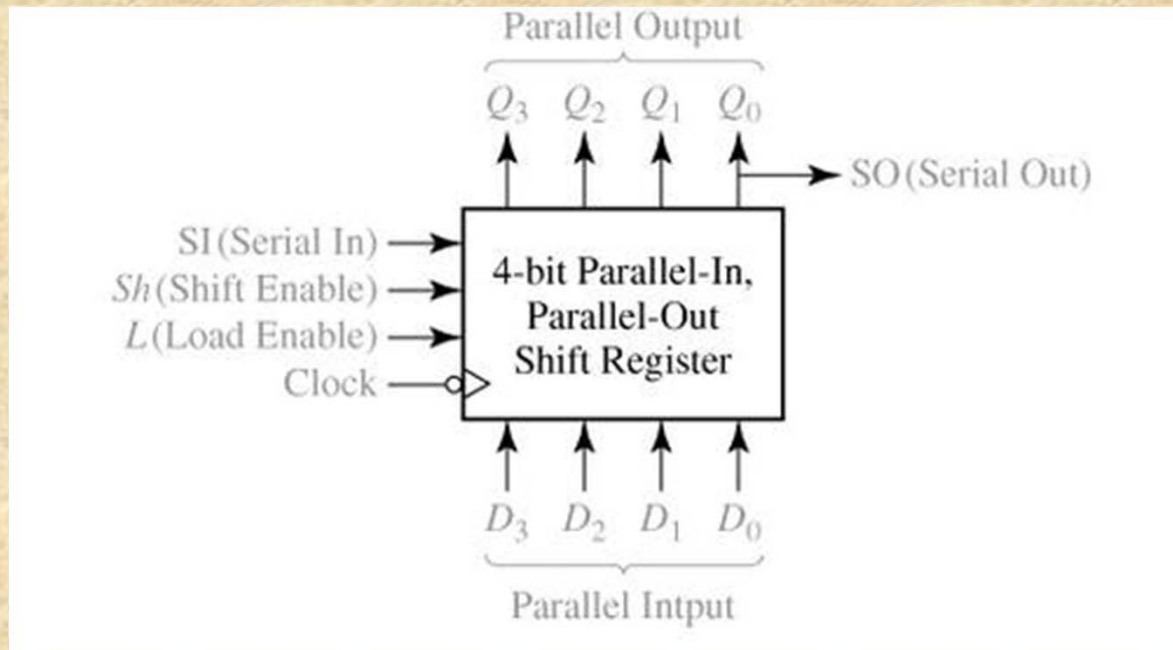
- Before the rising edge: $Q_3 Q_2 Q_1 Q_0 = 1101$
- What are they after the rising edge?

Shift Registers (3/6)

- Can you design a four-bit shift/rotate register? Two external control signals are *shift* and *rotate*.
 - shift&rotate can be 00/01/10

Shift Registers (4/6)

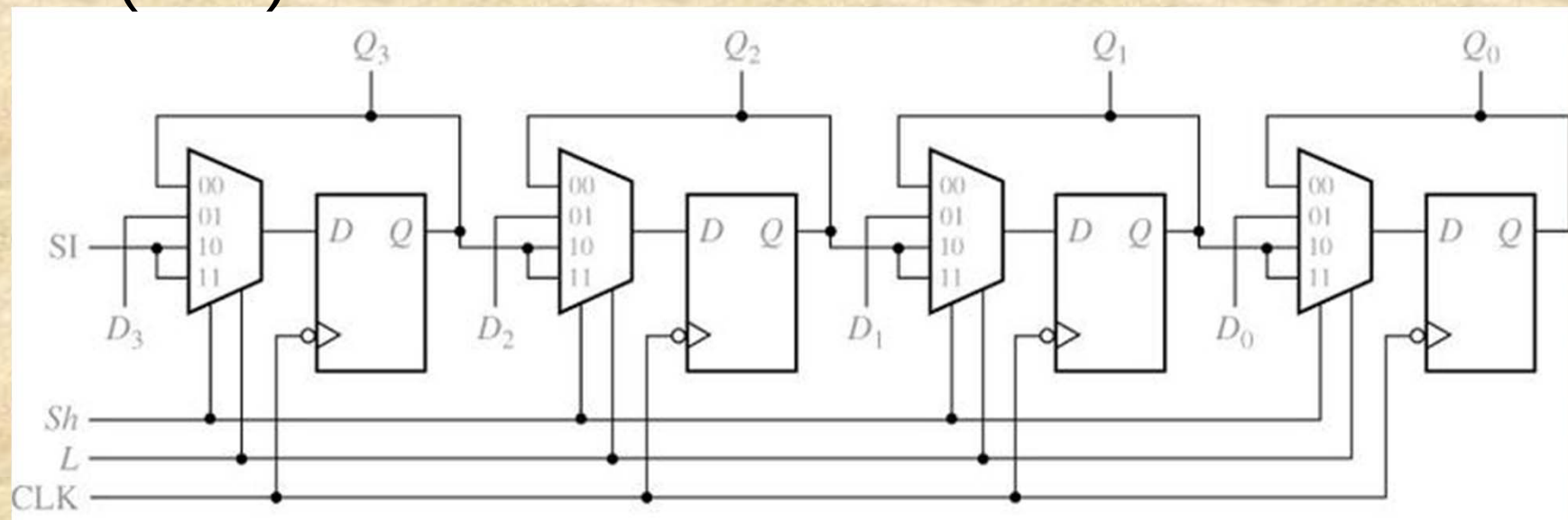
- 4-bit parallel-in, parallel-out shift register



Sh	L	Q_3^+	Q_2^+	Q_1^+	Q_0^+	action
0	0	Q_3	Q_2	Q_1	Q_0	no change
0	1	D_3	D_2	D_1	D_0	load
1	X	SI	Q_3	Q_2	Q_1	right shift

Shift Registers (5/6)

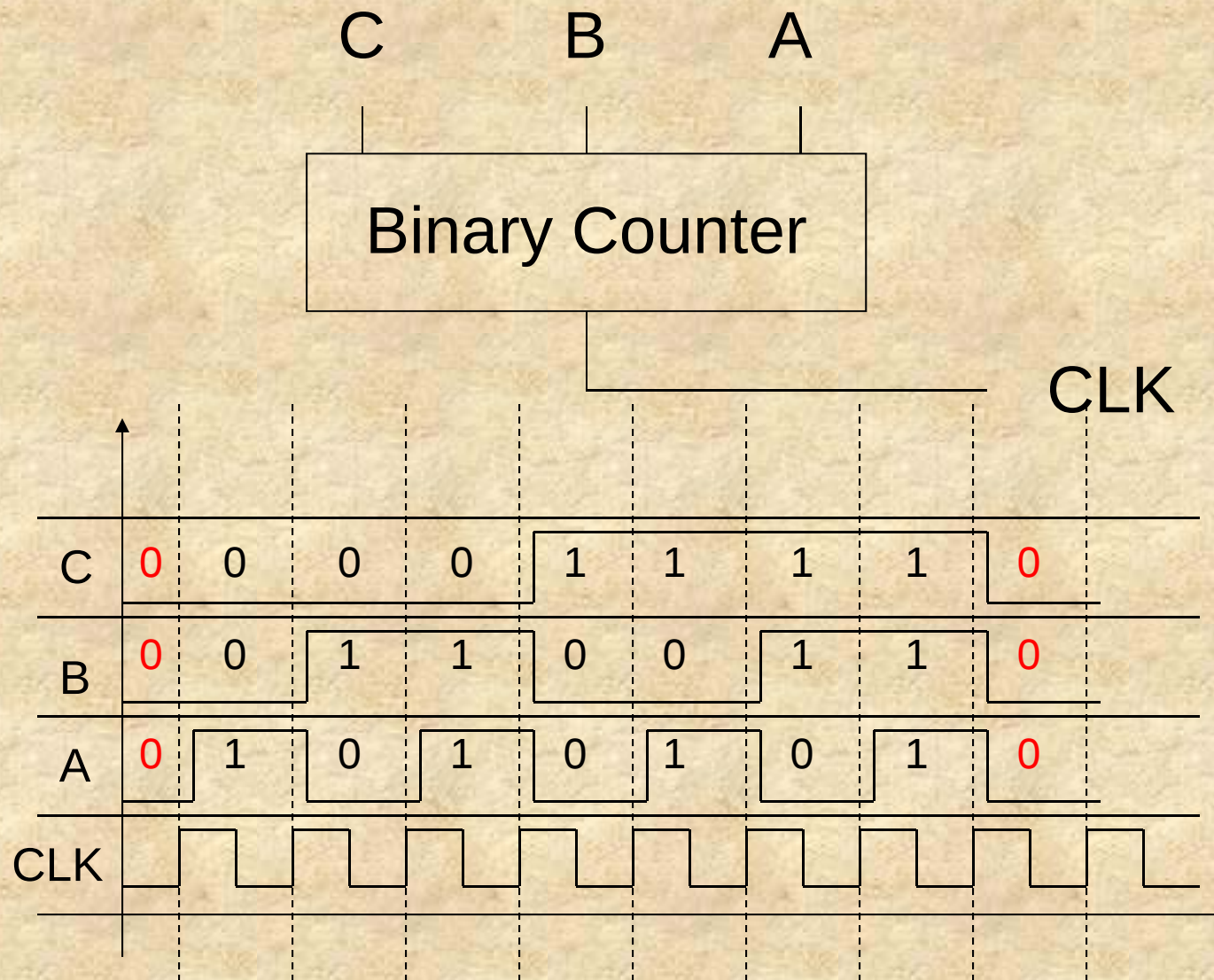
- 4-bit parallel-in, parallel-out shift register (cont)



Shift Registers (6/6)

- How can we design a bi-directional shift register with two external control signals: **S_0** and **S_1** ?
 - $S_0S_1=00$ nothing happens
 - $S_0S_1=01$ shift right
 - $S_0S_1=10$ shift left
 - $S_0S_1=11$ load new contents

Design of Binary Counters (1/11)



Design of Binary Counters (2/11)

- Design based on T flip-flops (FFs) $Q^+ = T \text{ xor } Q$

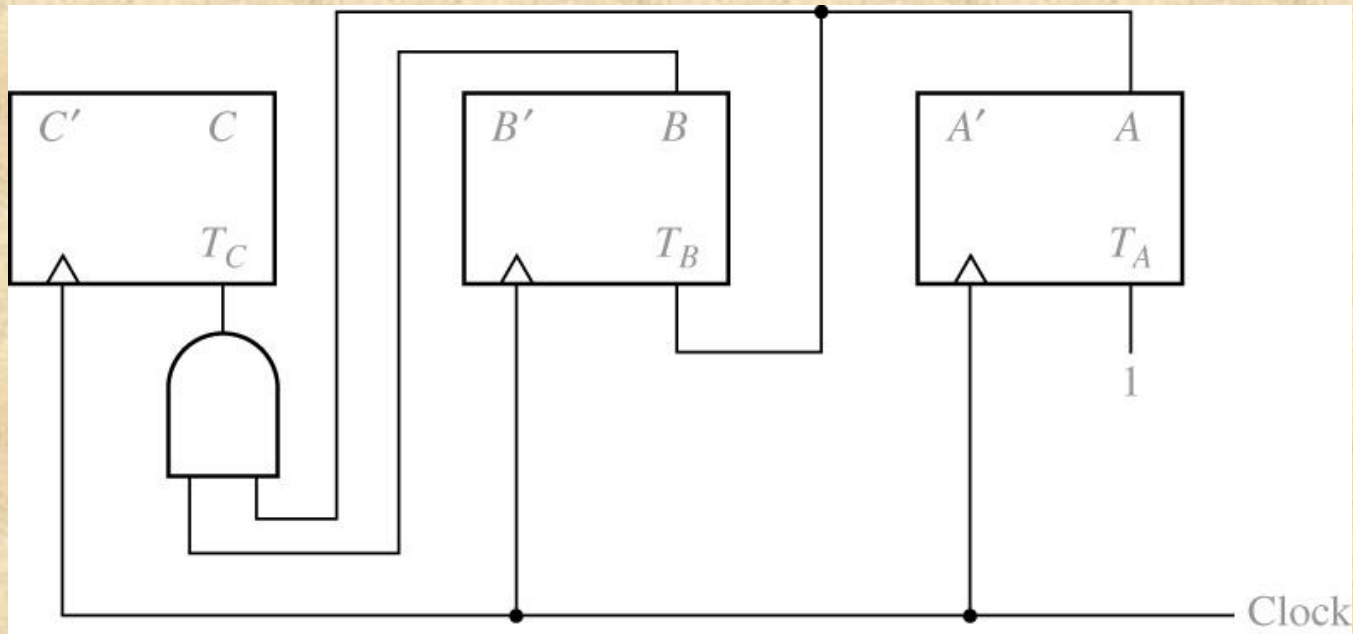
– Truth table

CBA	C^+	B^+	A^+
000	0	0	1
001	0	1	0
010	0	1	1
011	1	0	0
100	1	0	1
101	1	1	0
110	1	1	1
111	0	0	0

- A changes state anyway ($A^+ = A'$).
 - $T_A = 1$
- $B^+ = A \text{ xor } B$
 - $T_B = A$
- $C^+ = AB \text{ xor } C$
 - $T_C = AB$

Design of Binary Counters (3/11)

- Design based on T flip-flops (FFs)



3 bit counter

Design of Binary Counters (4/11)

- How can we design based on D flip-flops (FFs)?

- $Q^+ = D$

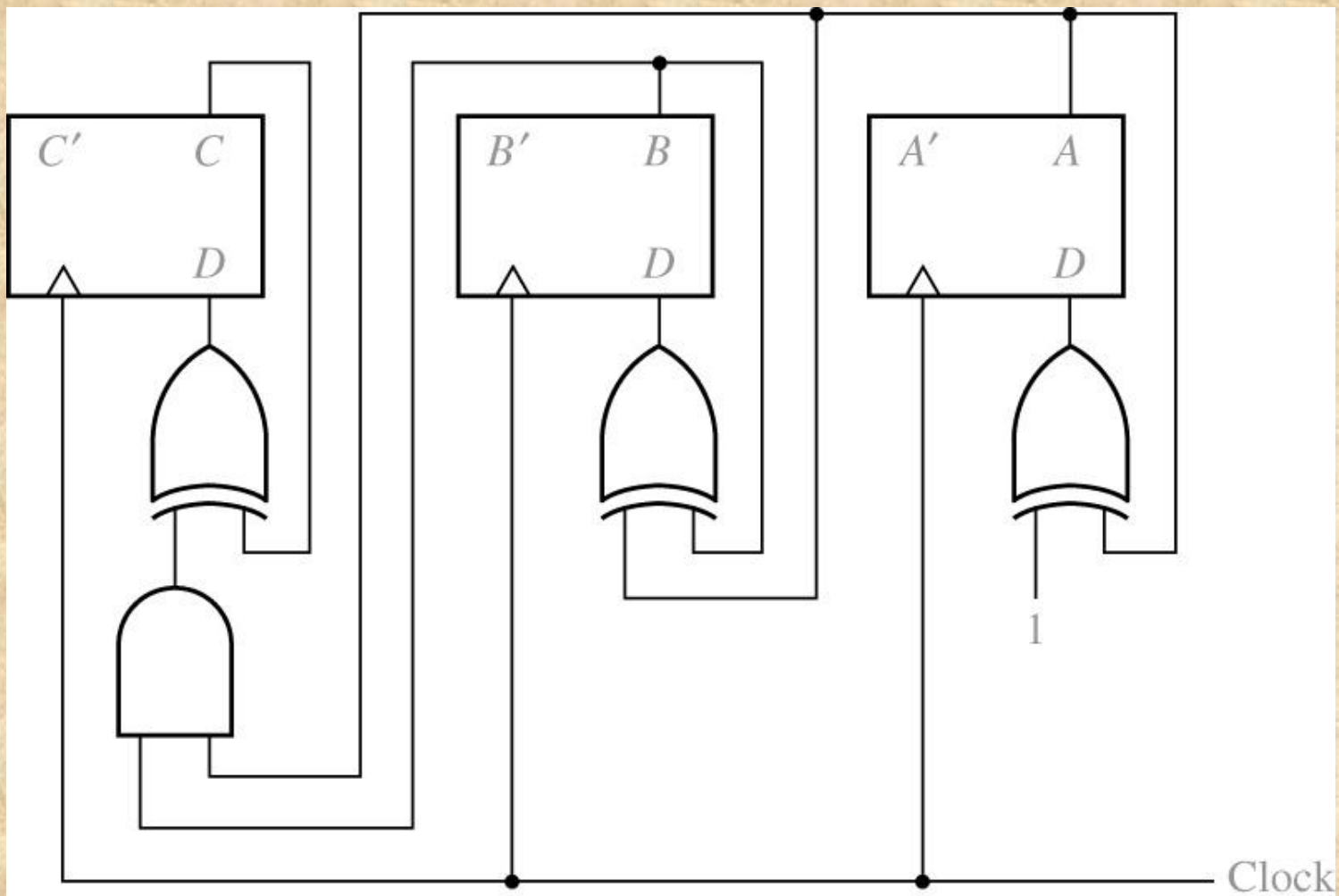
- $A^+ = A'$ $\rightarrow D_A = A' = A \text{ xor } 1$

- $B^+ = A \text{ xor } B$ $\rightarrow D_B = A \text{ xor } B$

- $C^+ = AB \text{ xor } C$ $\rightarrow D_C = AB \text{ xor } C$

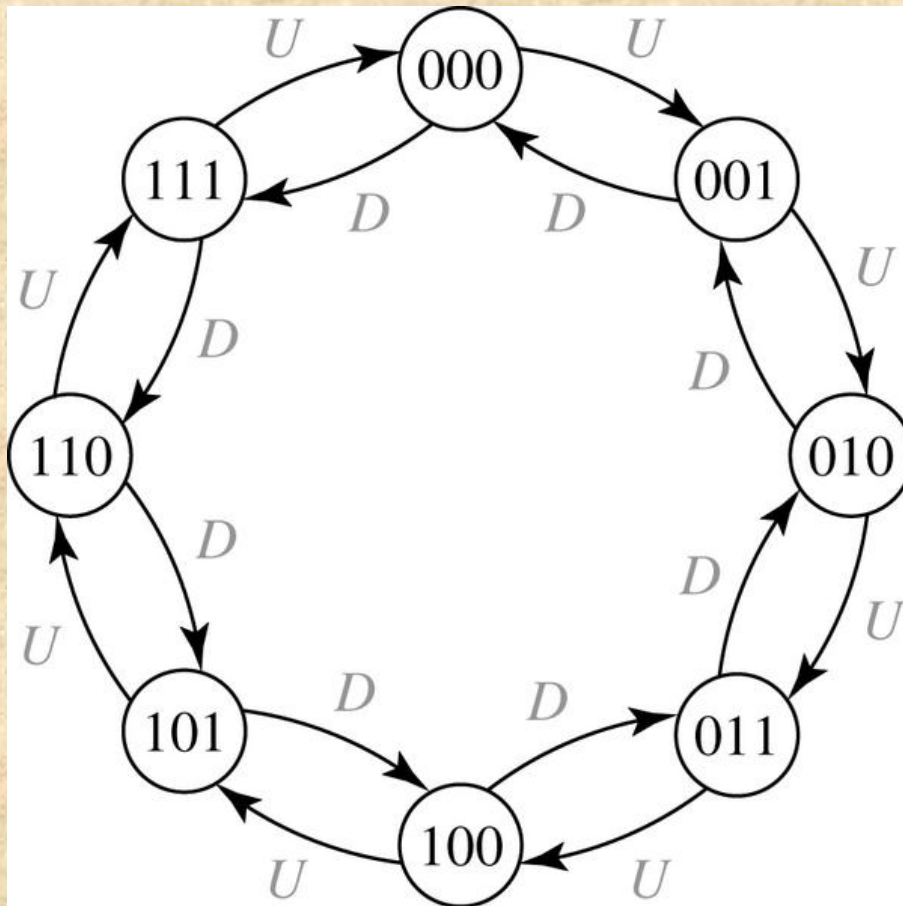
- Note $A' = A \text{ xor } 1$ and $A = A \text{ xor } 0$

Design of Binary Counters (5/11)



Design of Binary Counters (6/11)

- Up-down binary counter



Two controls signals: U and D

Design of Binary Counters (7/11)

- Process of designing up-down counter
 - Number of control signals: two (up and down)
 - Number of possible control modes
 - UD=10 counting up
 - UD=01 counting down
 - UD=00 no change
 - UD=11 restricted

Design of Binary Counters (8/11)

- Process of designing up-down counter (cont)
 - For each control mode, draw truth tables and decide the logic expressions
 - $UD=00$ $A^+ = \underline{A \text{ xor } 0}$, $B^+ = \underline{B \text{ xor } 0}$, $C^+ = \underline{C \text{ xor } 0}$
 - $UD=10$ $A^+ = \underline{A \text{ xor } 1}$, $B^+ = \underline{B \text{ xor } A}$, $C^+ = \underline{C \text{ xor } AB}$
 - $UD=01$ $A^+ = \underline{A \text{ xor } 1}$, $B^+ = \underline{B \text{ xor } A'}$, $C^+ = \underline{C \text{ xor } A'B'}$
 - Put them together. For example
 - $B^+ = \underline{U'D'}(B \text{ xor } 0) +$
 - $\underline{U'D}(B \text{ xor } A) +$
 - $\underline{UD'}(B \text{ xor } A)$

Can be realized using multiplexers.

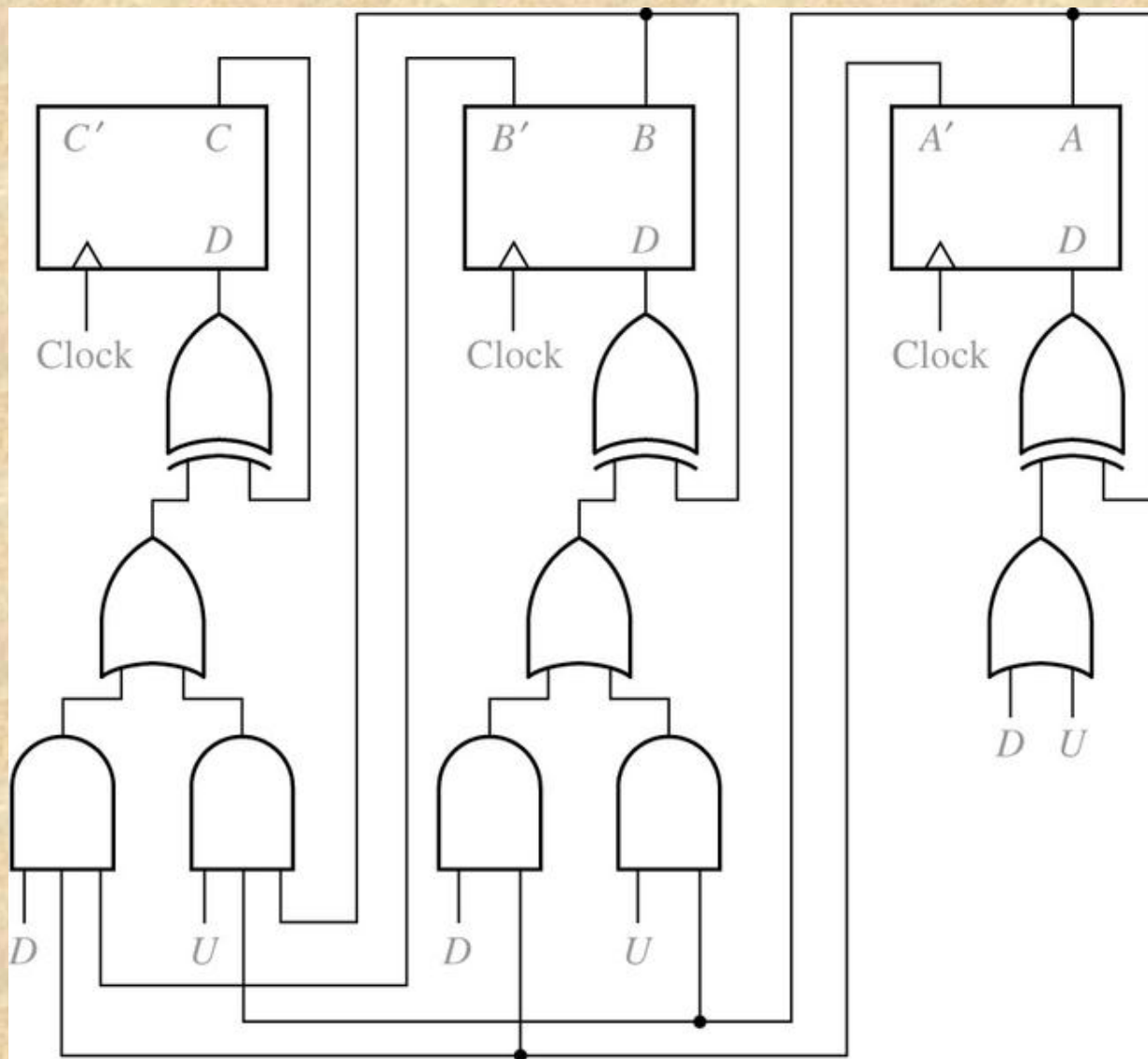
Design of Binary Counters (9/11)

- Process of designing up-down counter (cont)
 - In the textbook, these expressions are written as
 - $A^+ = A \text{ xor } (U+D)$
 - $B^+ = B \text{ xor } (UA+DA')$
 - $C^+ = C \text{ xor } (UBA+DB'A')$

Design of Binary Counters (10/11)

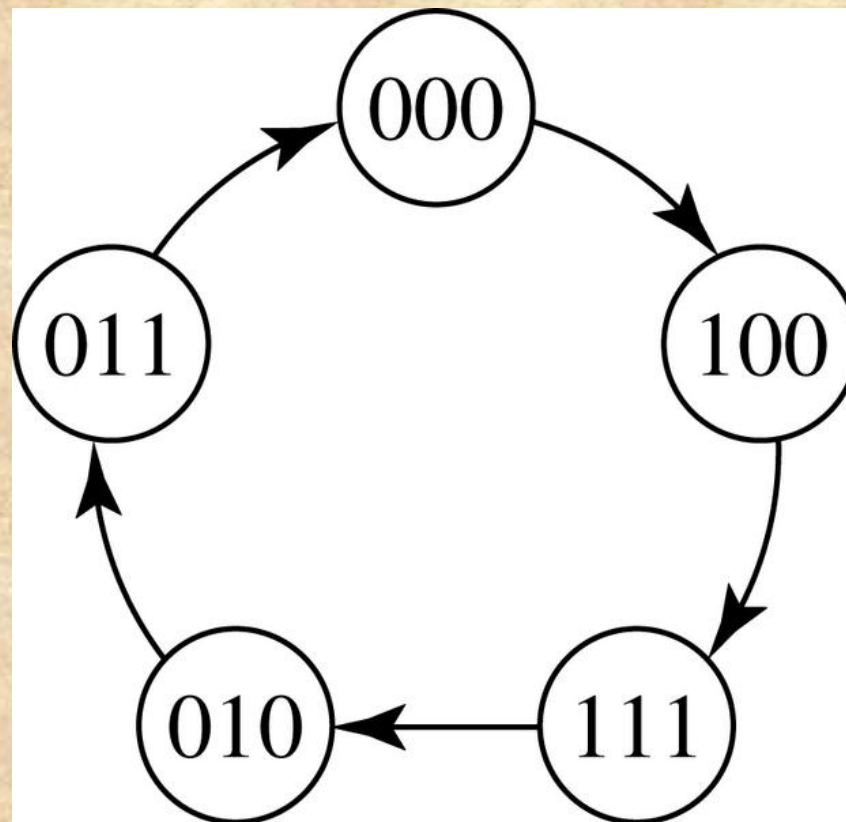
- Process of designing up-down counter (cont)
 - Because $Q^+ = D$ in D flip-flop (FFs),
 - $D_A = A^+ = A \text{ xor } (U+D)$
 - $D_B = B^+ = B \text{ xor } (UA+DA')$
 - $D_C = C^+ = C \text{ xor } (UBA+DB'A')$

Design of Binary Counters (11/11)



Counters for Other Sequences (1/5)

- In some applications, the sequence of states of a counter is not in straight binary order.

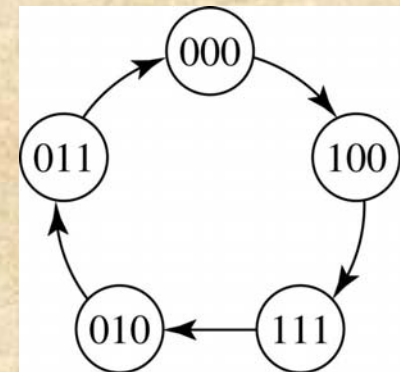


Counters for Other Sequences (2/5)

- The method is very similar to the one for designing a binary counter.
 1. A truth table that shows the relationship between current states and next states.
 2. Decide the expressions for T, JK, SR, or D depending on which kind of flip-flop (FF) you are using.

Counters for Other Sequences (3/5)

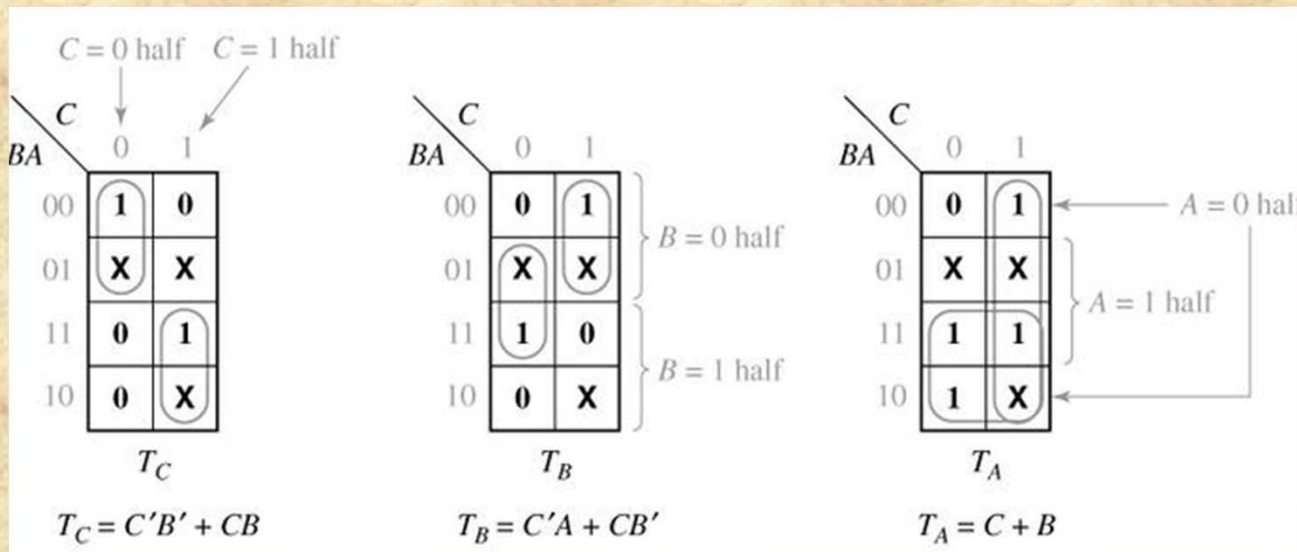
Inputs						Outputs		
C	B	A	C ⁺	B ⁺	A ⁺	T _C	T _B	T _A
0	0	0	1	0	0	1	0	0
0	0	1	-	-	-	X	X	X
0	1	0	0	1	1	0	0	1
0	1	1	0	0	0	0	1	1
1	0	0	1	1	1	0	1	1
1	0	1	-	-	-	X	X	X
1	1	0	-	-	-	X	X	X
1	1	1	0	1	0	1	0	1



Say we are
using T flip-
flop (FF).

Counters for Other Sequences (4/5)

- Next we need to develop the expressions for T_A , T_B , and T_C .



Note the variables are A, B, C

Counters for Other Sequences (5/5)

- How to design based on D flip-flop (FFs)?

C	B	A	C ⁺	B ⁺	A ⁺	D _C	D _B	D _A
0	0	0	1	0	0	1	0	0
0	0	1	-	-	-	-	-	-
0	1	0	0	1	1	0	1	1
0	1	1	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1
1	0	1	-	-	-	-	-	-
1	1	0	-	-	-	-	-	-
1	1	1	0	1	0	0	1	0

Counter Design Using S-R and J-K Flip-Flops (FF) (1/7)

- Steps for designing counters
 - Draw truth table
 - C B A C⁺ B⁺ A⁺ ← *Problem specification*
 - Decide the required values for flip-flop (FF) inputs
 - Given current and next states, decide the required inputs.
 - Decide the expressions for the inputs to FFs
 - $T = F(C, B, A)$
 - $D = F(C, B, A)$
 - Etc.
- Minterms → Karnaugh maps*

Counter Design Using S-R and J-K Flip-Flop (FFs) (2/7)

- The easiest one
 - $Q^+ = D \rightarrow Q^+$
- T flip-flop (FF)
 - $Q^+ = Q \text{ xor } T \rightarrow T = Q \text{ xor } Q^+$

Counter Design Using S-R and J-K Flip-Flop (FFs) (3/7)

- SR (NOR based)

$\overline{Q}$	Q^+	S	R		Q	Q^+	S	R
0	0	0	0		0	0	0	X
0	1	1	0		1	0	0	1
1	0	0	1		1	1	X	0
1	1	0	0					
		1	0					

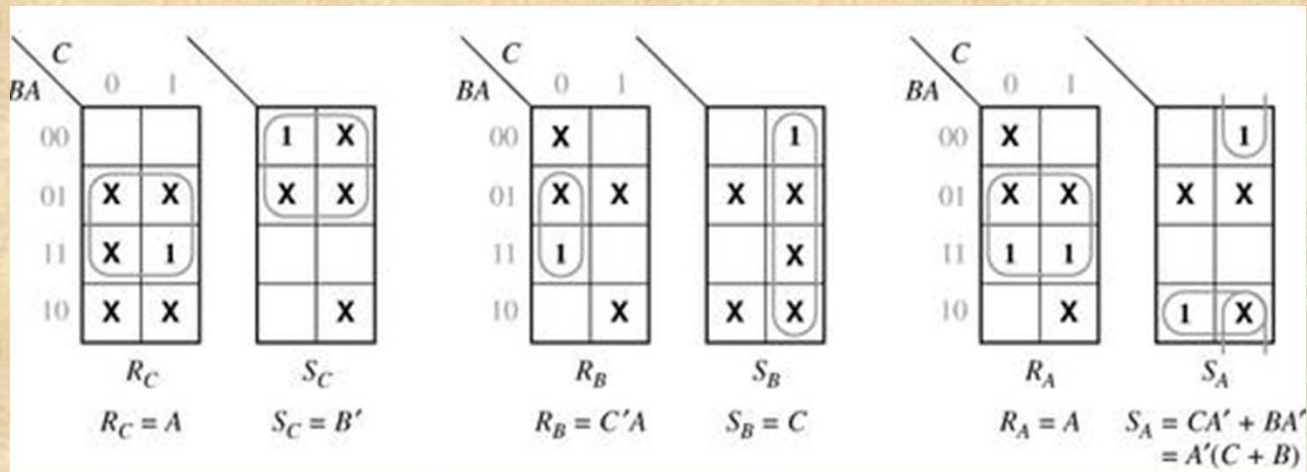
Counter Design Using S-R and J-K Flip-Flop (FFs) (4/7)

- Counter based on SR

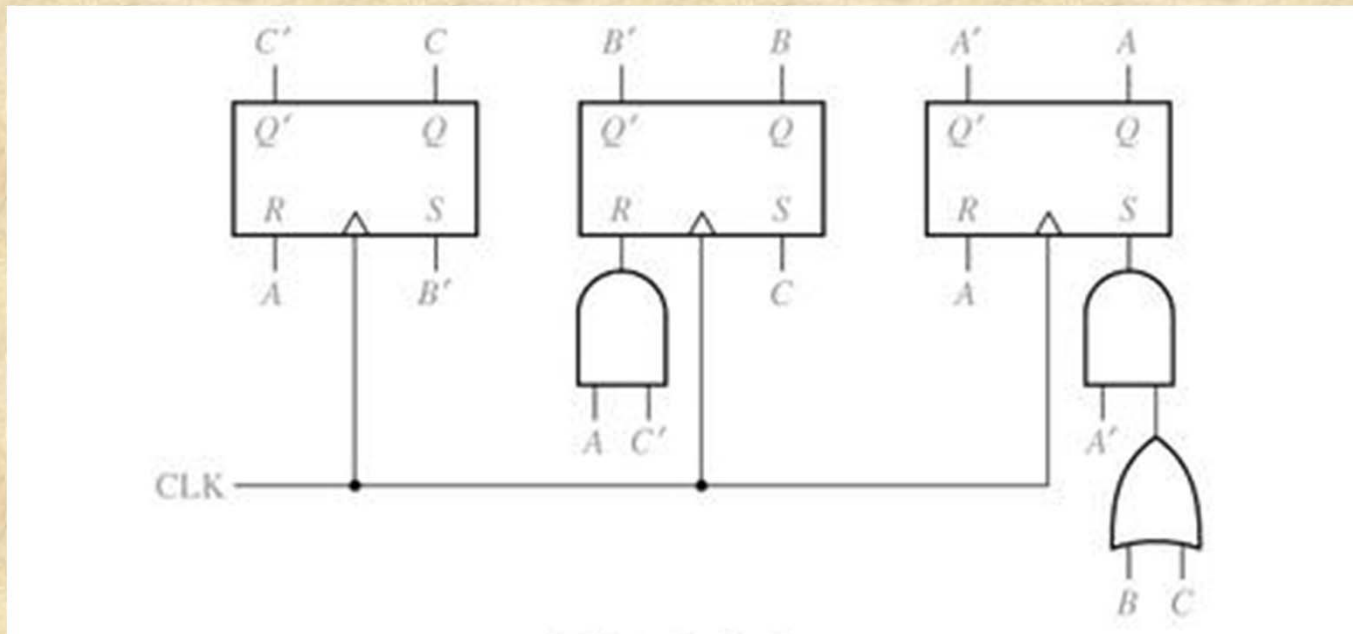
C	B	A	C ⁺	B ⁺	A ⁺	S _C	R _C	S _B	R _B	S _A	R _A
0	0	0	1	0	0	1	0	0	X	0	X
0	0	1	-	-	-	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	0
0	1	1	0	0	0	0	X	0	1	0	1
1	0	0	1	1	1	X	0	1	0	1	0
1	0	1	-	-	-	X	X	X	X	X	X
1	1	0	-	-	-	X	X	X	X	X	X
1	1	1	0	1	0	0	1	X	0	0	1

Counter Design Using S-R and J-K Flip-Flop (FFs) (5/7)

- Use Karnaugh maps to simplify. (Do not forget the do-not-care terms)

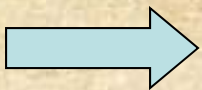


Counter Design Using S-R and J-K Flip-Flop (FFs) (6/7)



Counter Design Using S-R and J-K Flip-Flop (FFs) (7/7)

- JK

$- \underline{Q}$	Q^+	J	K		Q	Q^+	J	K
$- 0$	0	0	0		0	0	0	X
$-$		0	1		0	1	1	X
$- 0$	1	1	0		1	0	X	1
$-$		1	1		1	1	X	0
$- 1$	0	0	1					
$-$		1	1					
$- 1$	1	0	0					
$-$		1	0					