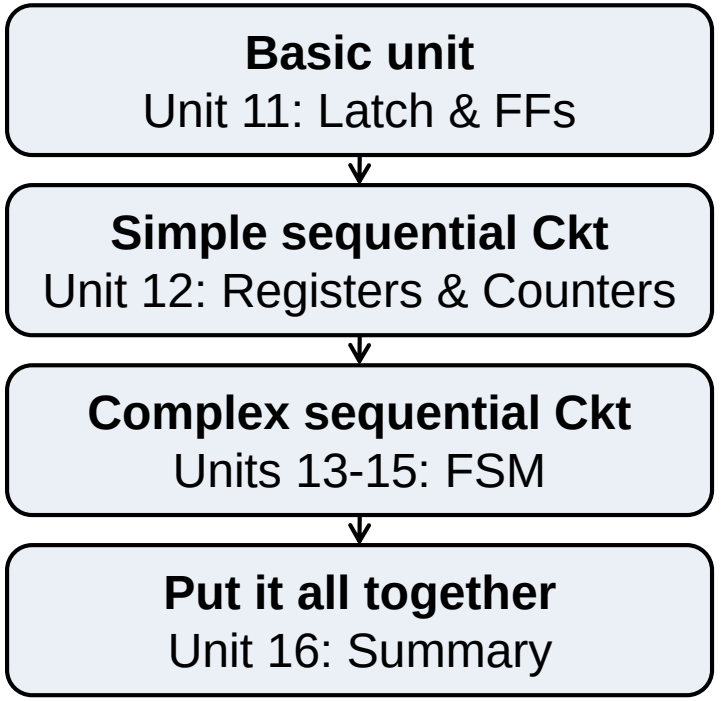
UNIT 12
REGISTERS AND COUNTERS

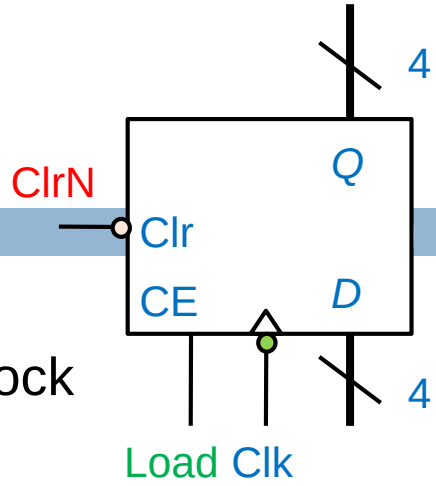
Fall 2021

Registers and Counters

- **Contents**
  - Register and register transfers
    - Operation
    - Shift register
  - Counters
    - Design of binary counters
    - Counters for other sequences
    - Counter design using D FFs
    - Counter design using S-R and J-K
    - Derivation of FF input equations
- **Reading**
  - Unit 12

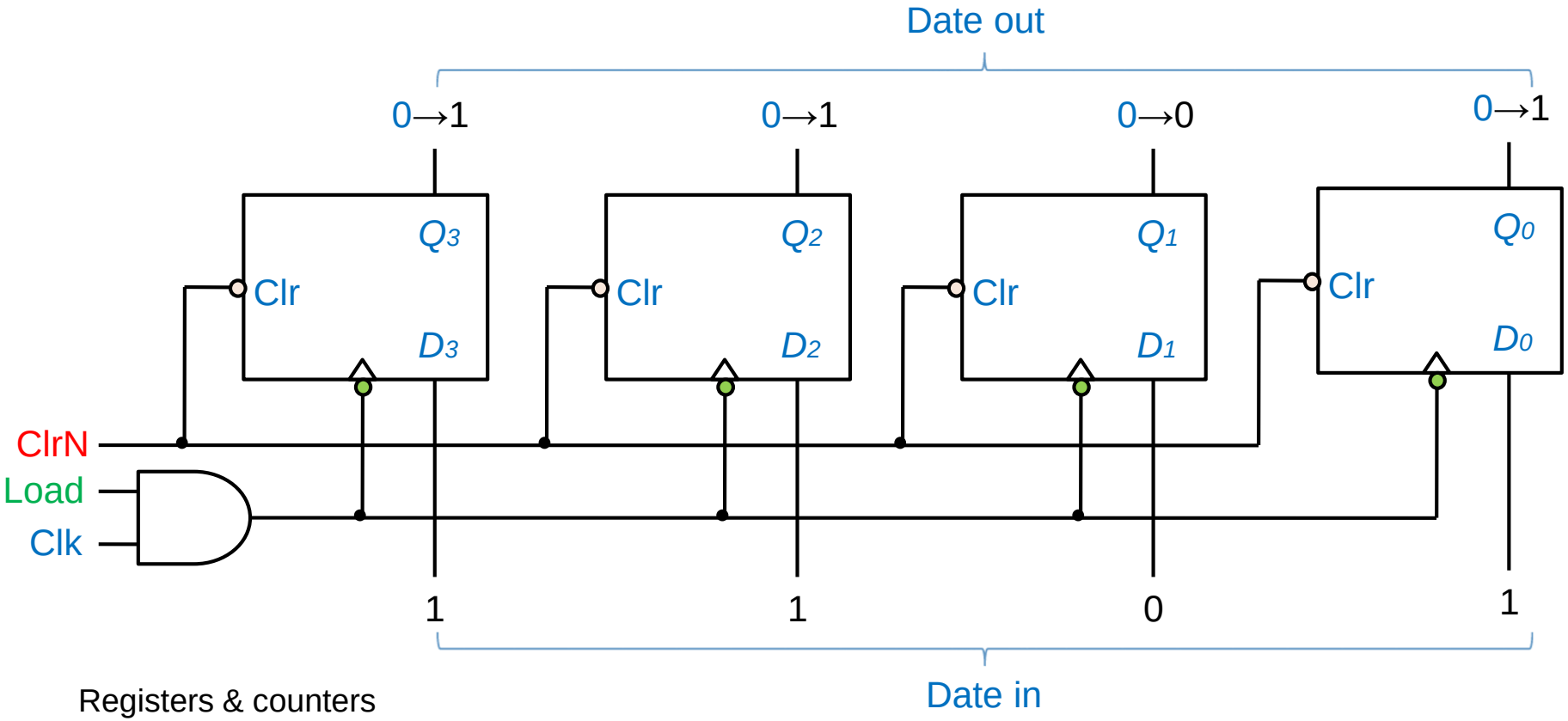


Registers (1/2)



- **A register: a group of D FFs with a common clock**
  - e.g., 4-bit D FF registers with Data, Load, Clear, Clock
    1. Using gated clock

       When Load = 1, load data at *D* to *Q* at Clk falling

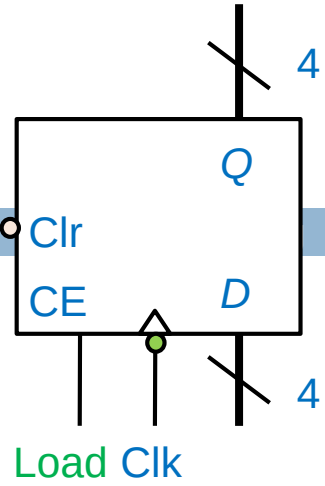


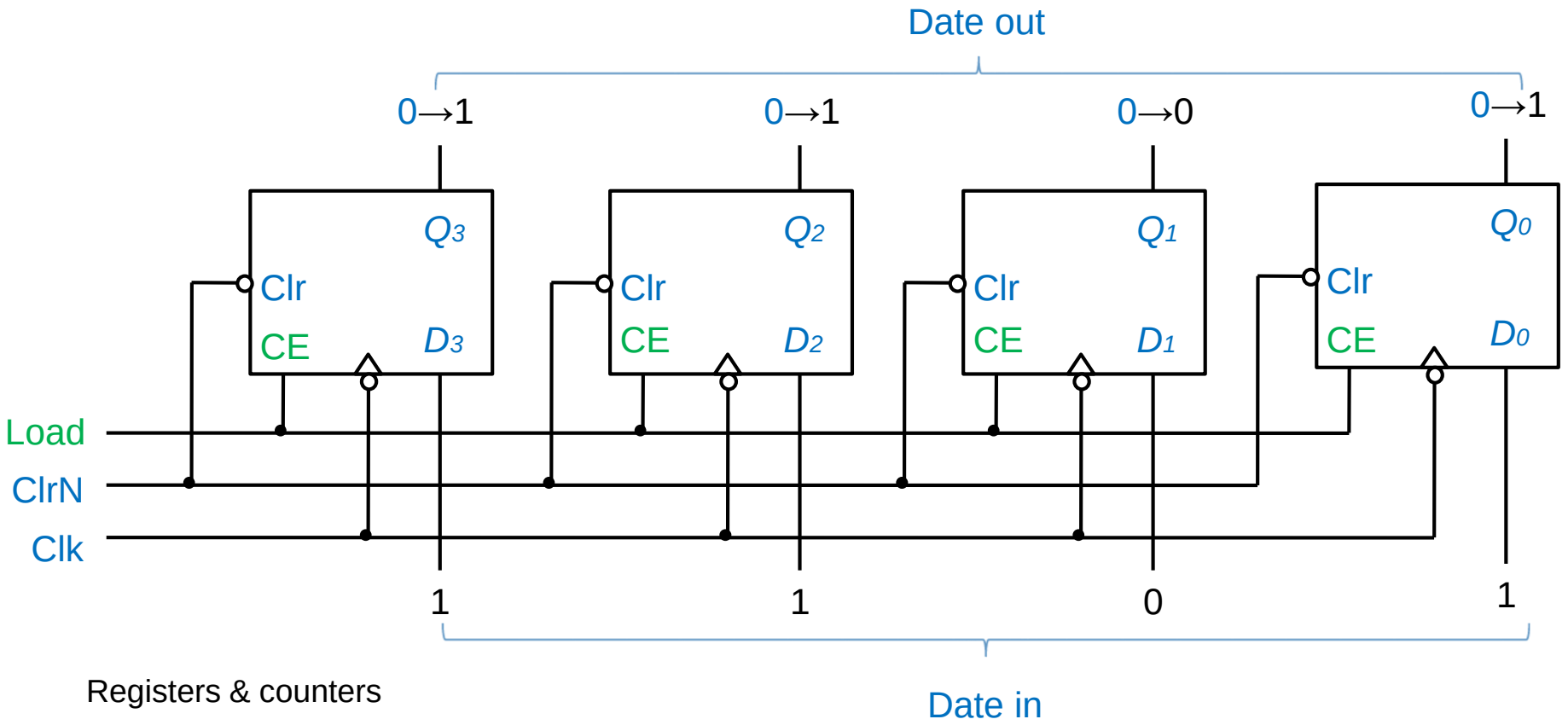
Registers (2/2)

2. With clock enable

   When Load = 1, load data at *D* to *Q* at Clk falling

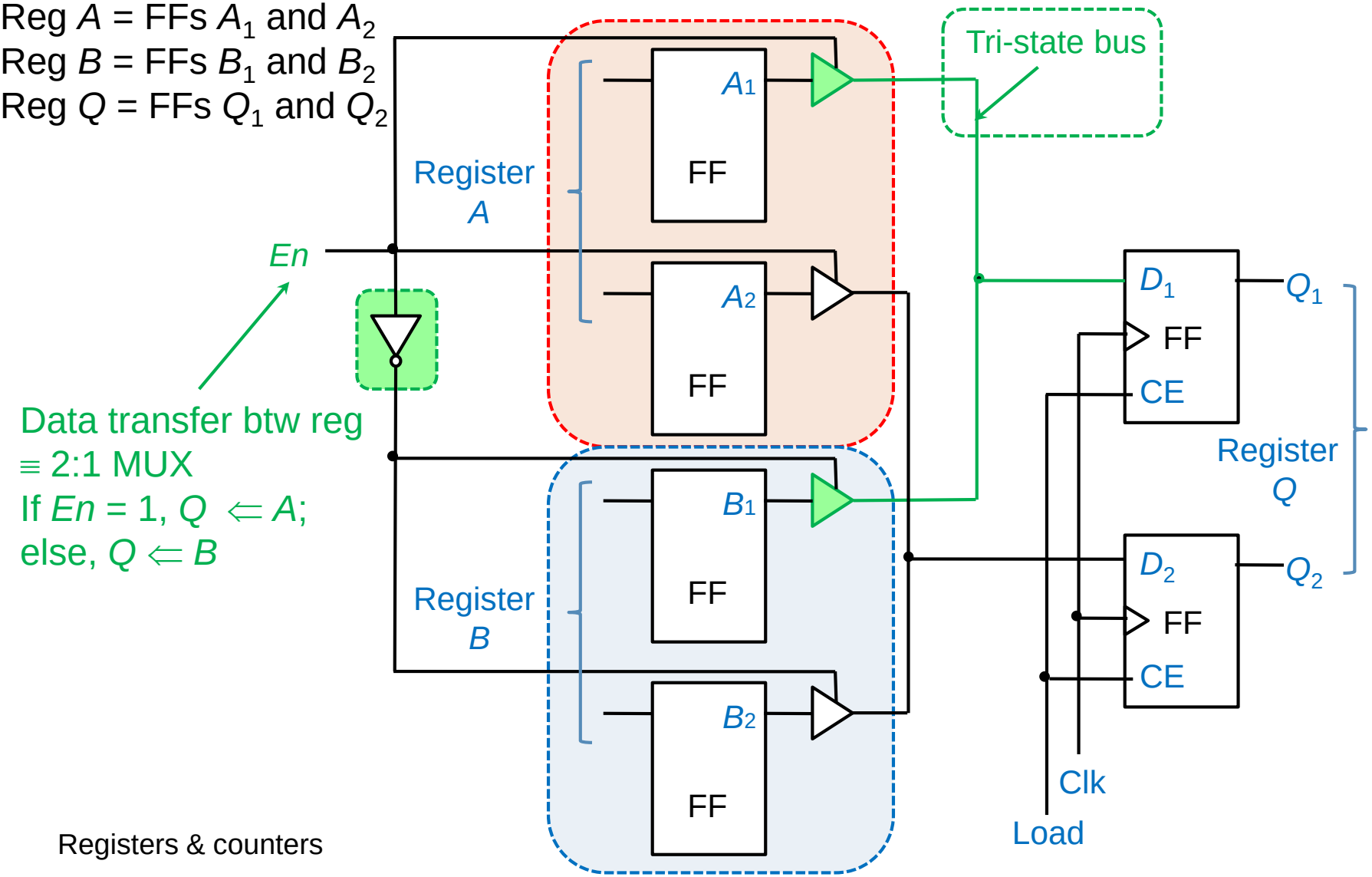
   When ClrN = 0, clear *Q* to 0





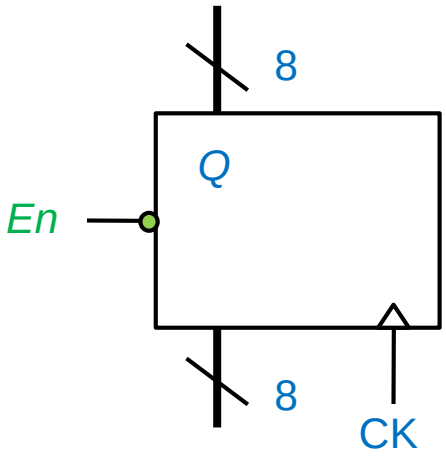
Data Transfer between Registers

Reg A = FFs A_1 and A_2
Reg B = FFs B_1 and B_2
Reg Q = FFs Q_1 and Q_2

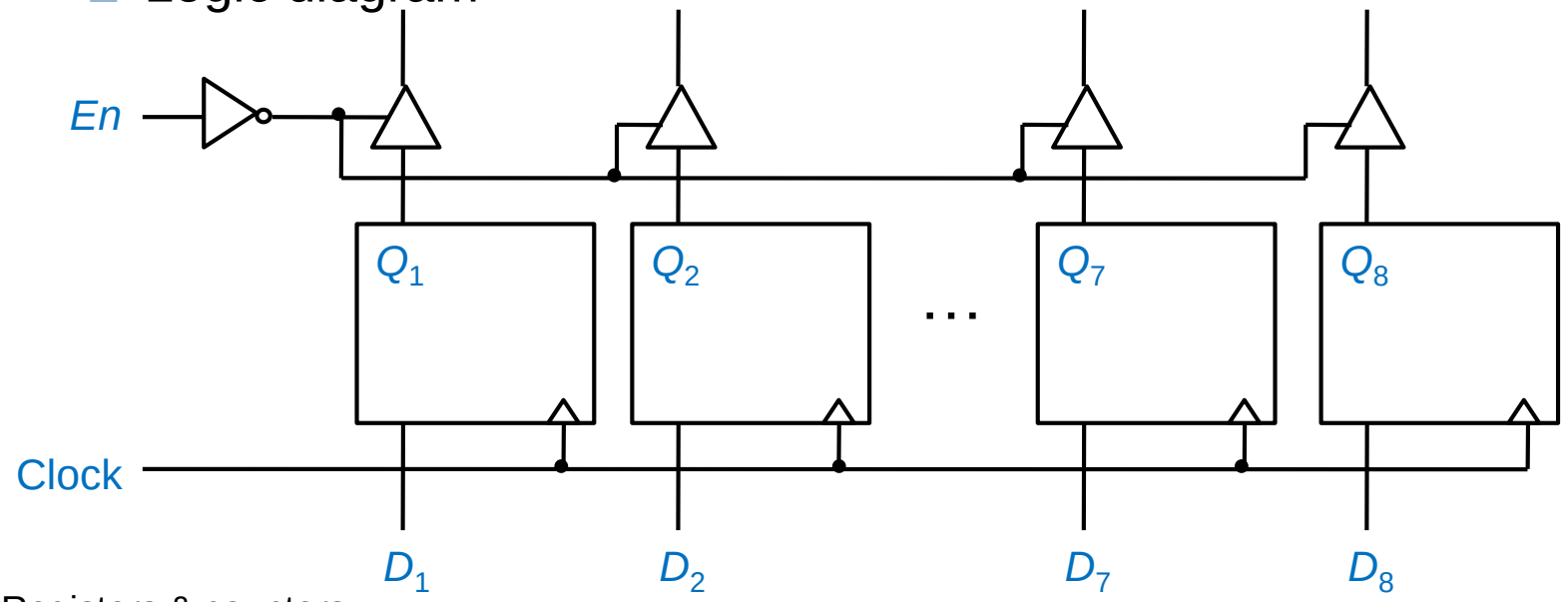


8-Bit Register with Tri-State Output (1/2)

- Symbol

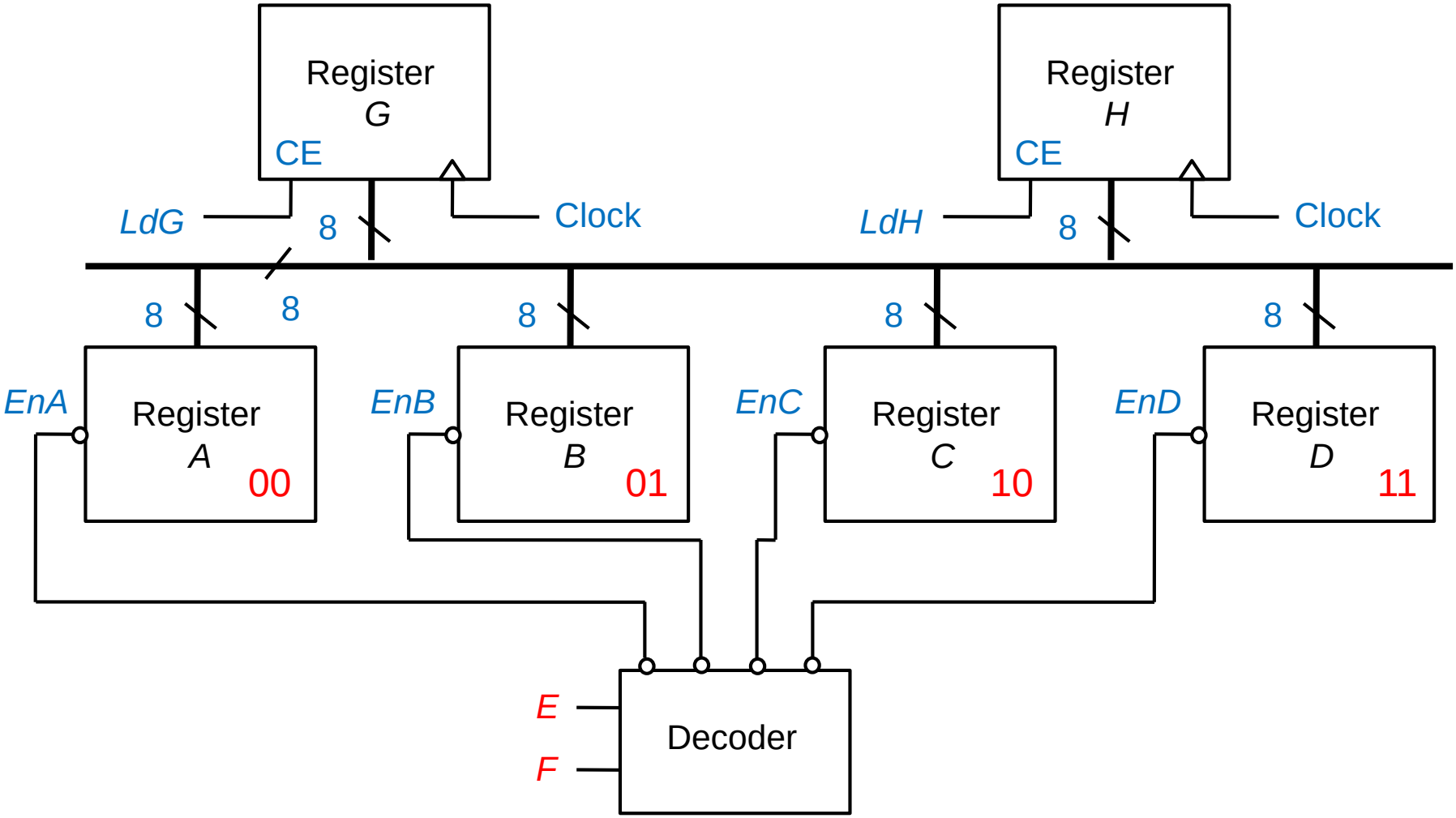


- Logic diagram



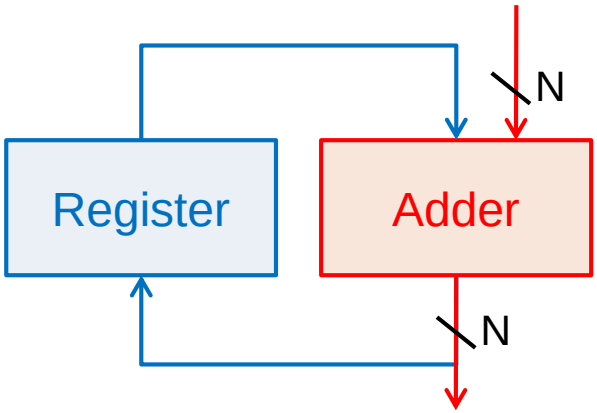
8-Bit Register with Tri-State Output (2/2)

- Data transfer

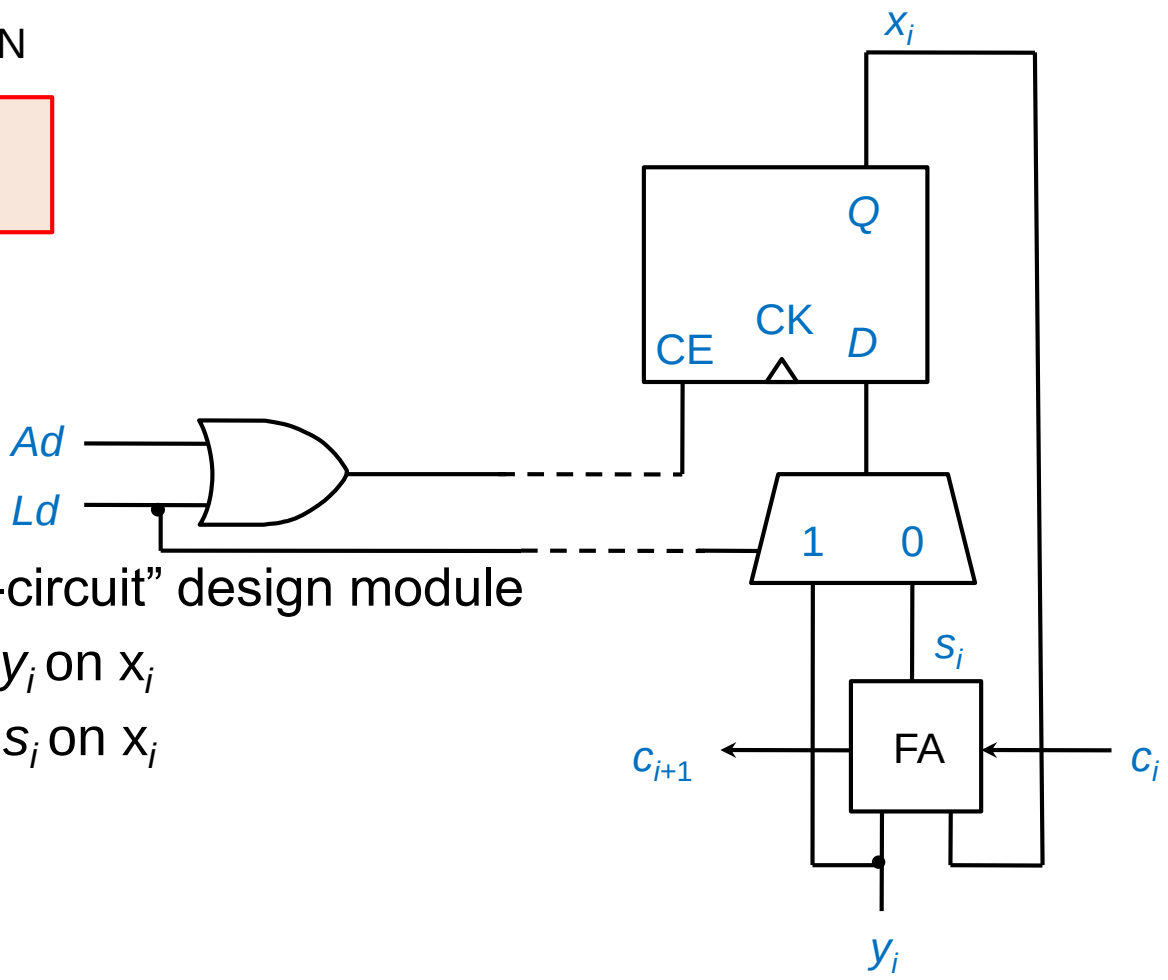


N-Bit Parallel Adder with Accumulator (1/2)

- **Adder with accumulator**

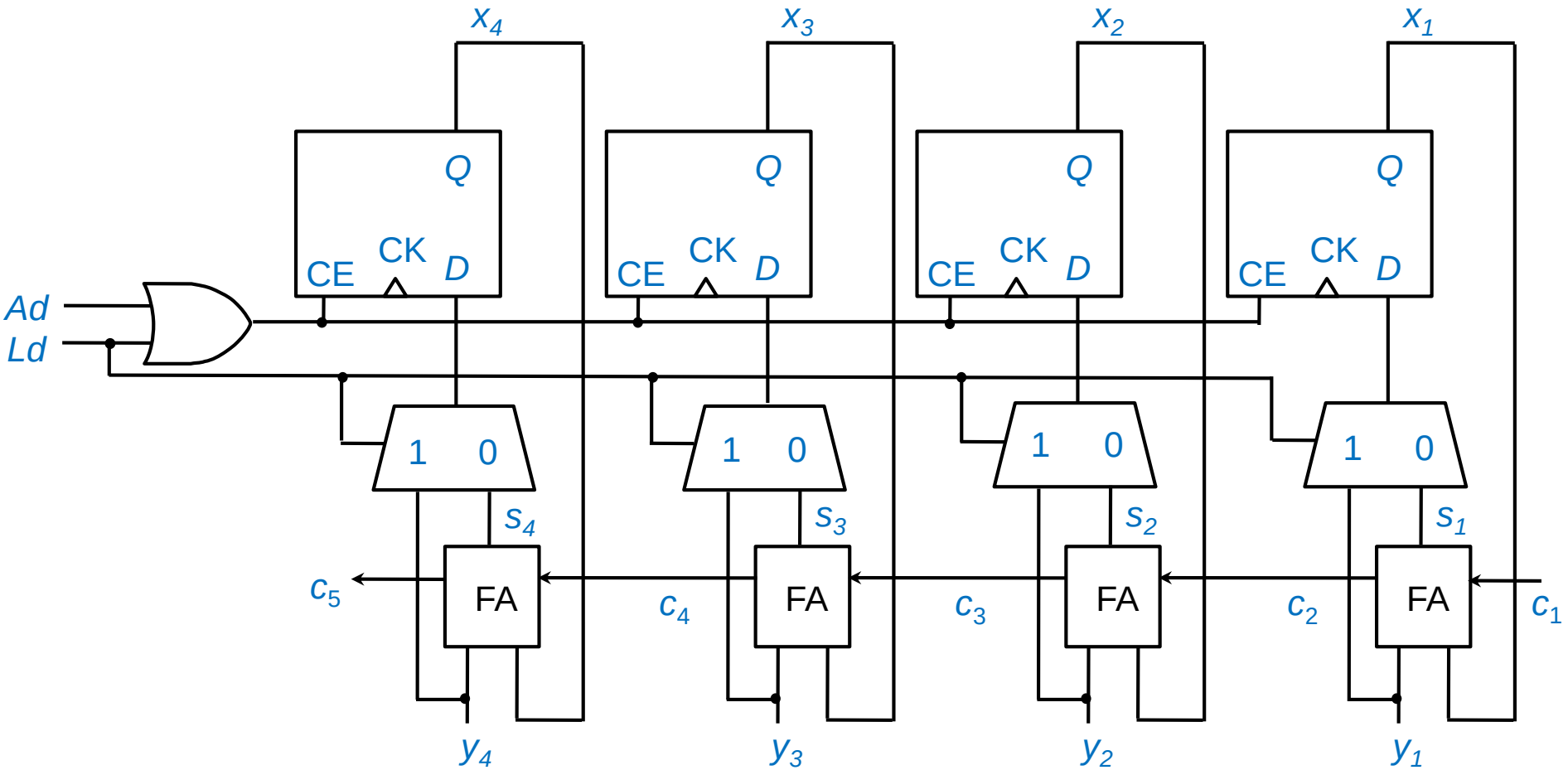


- **Adder with MUX**
  - Modular design
  - Suitable for "sub-circuit" design module
    1. Set Ld = 1; put y_i on x_i
    2. Set Ad = 1; put s_i on x_i

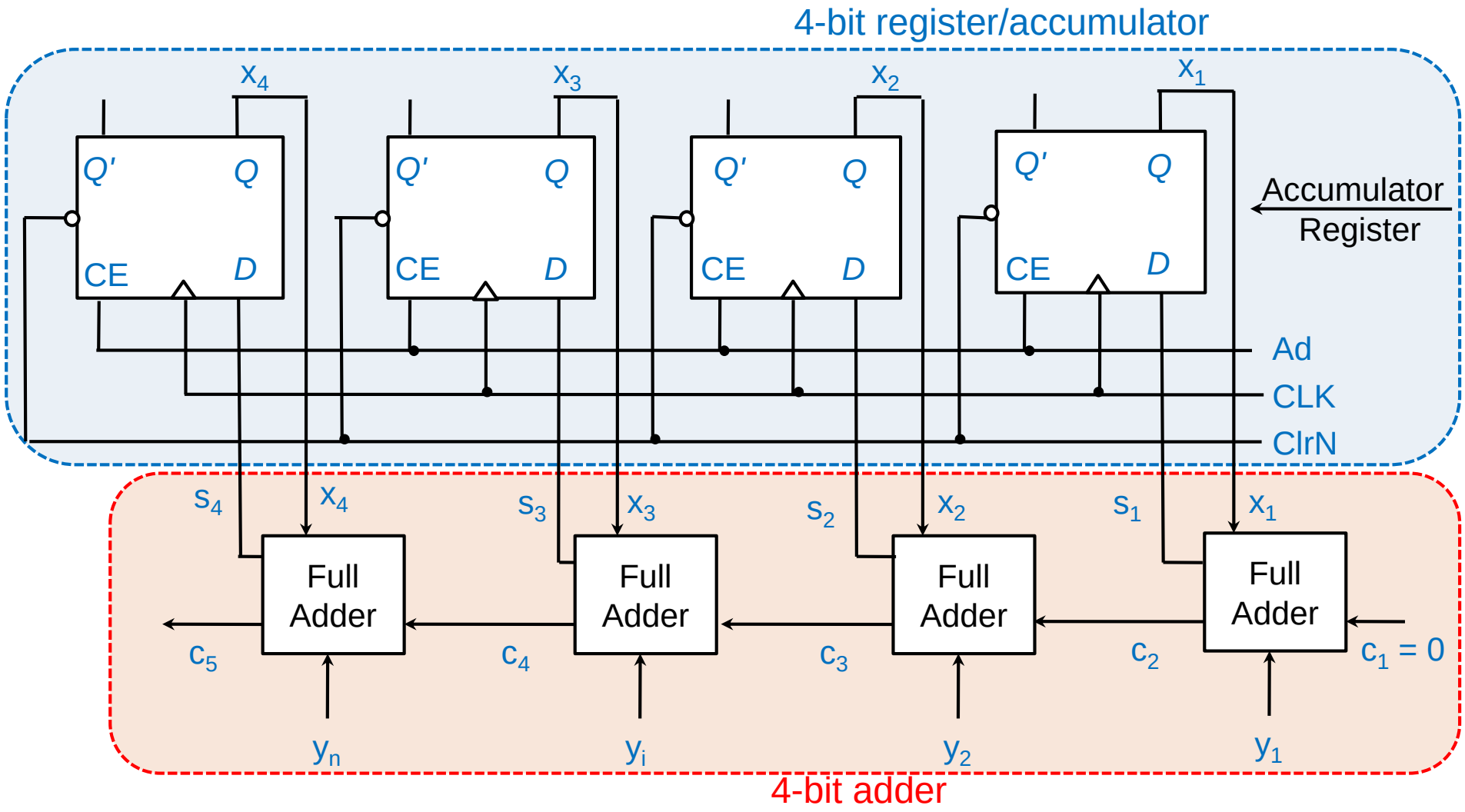


N-Bit Parallel Adder with Accumulator (2/2)

- **Down to bit-level**



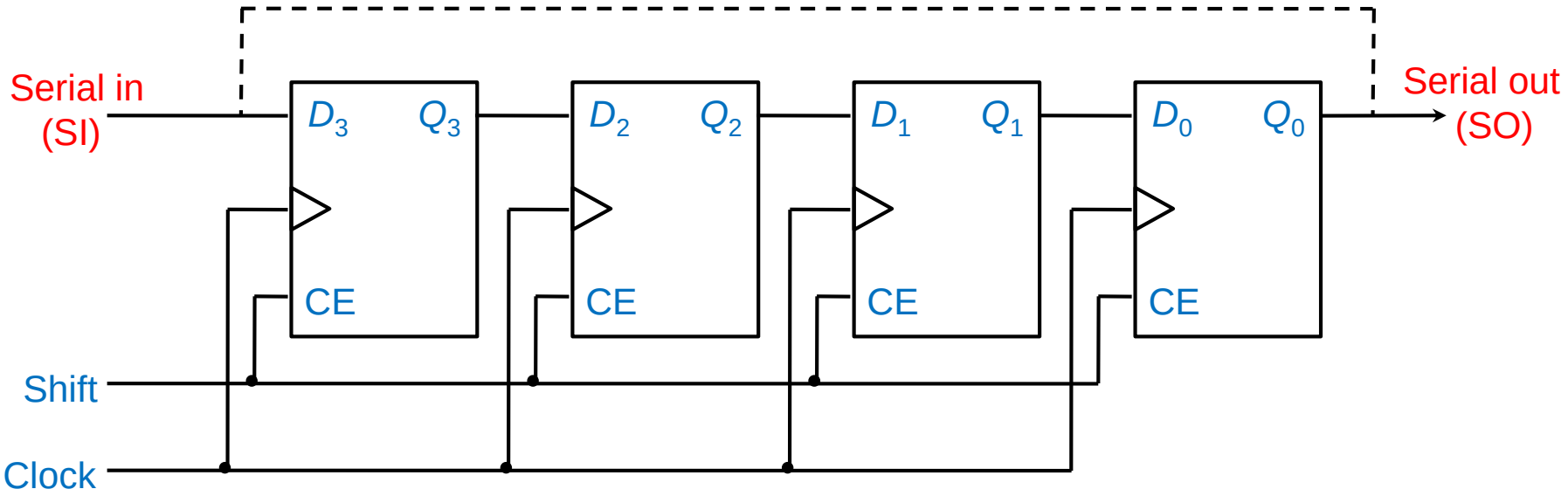
Adder with Resettable Accumulator



Shift Registers

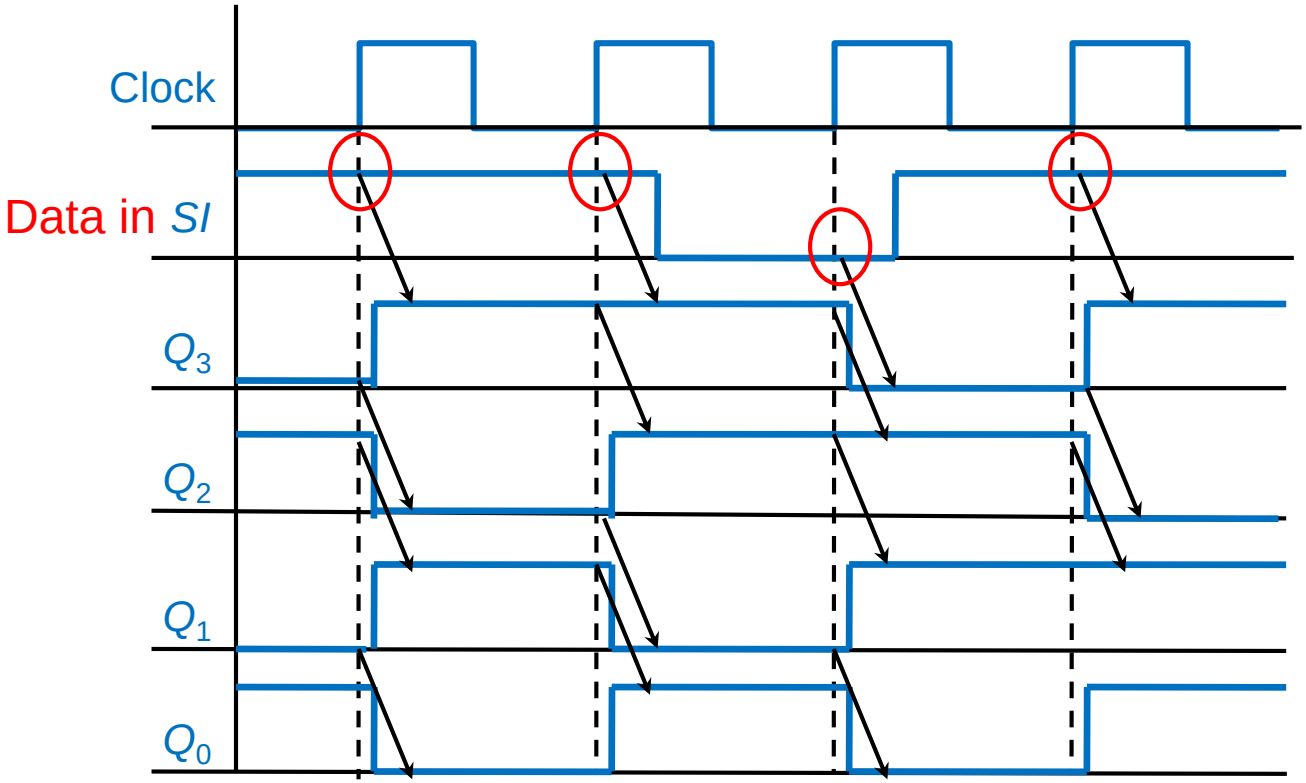
Shift Registers (1/2)

- **A shift register: a group of FFs where binary data can be stored and shifted left or right when a shift signal is applied**
  - e.g., 4-bit right-shift register



Shift Registers (2/2)

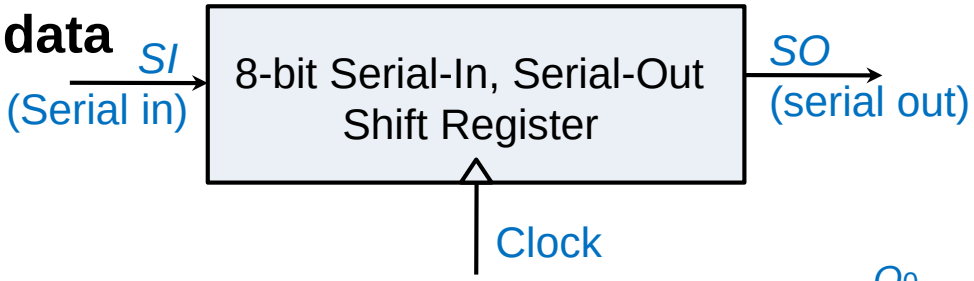
- Timing diagram of a 4-bit right-shift register

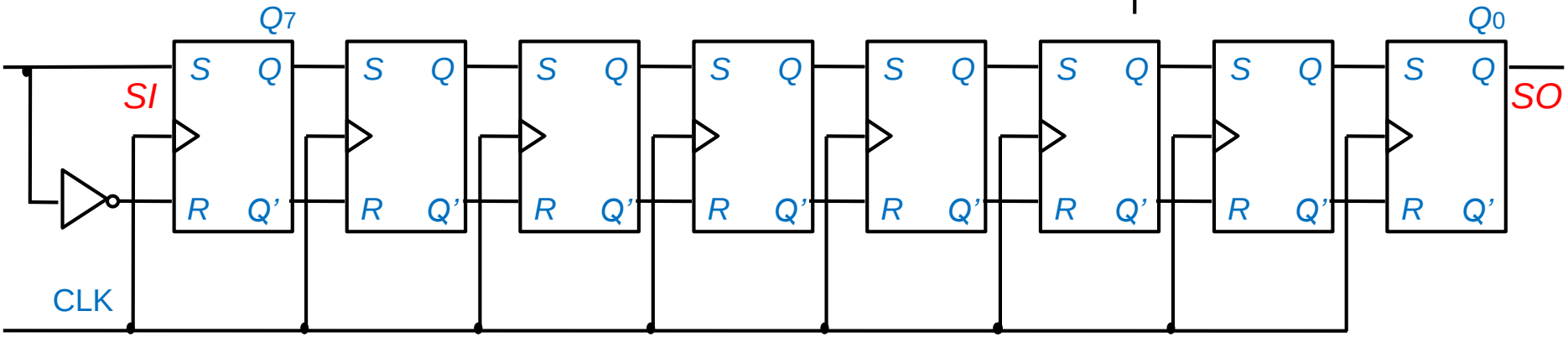


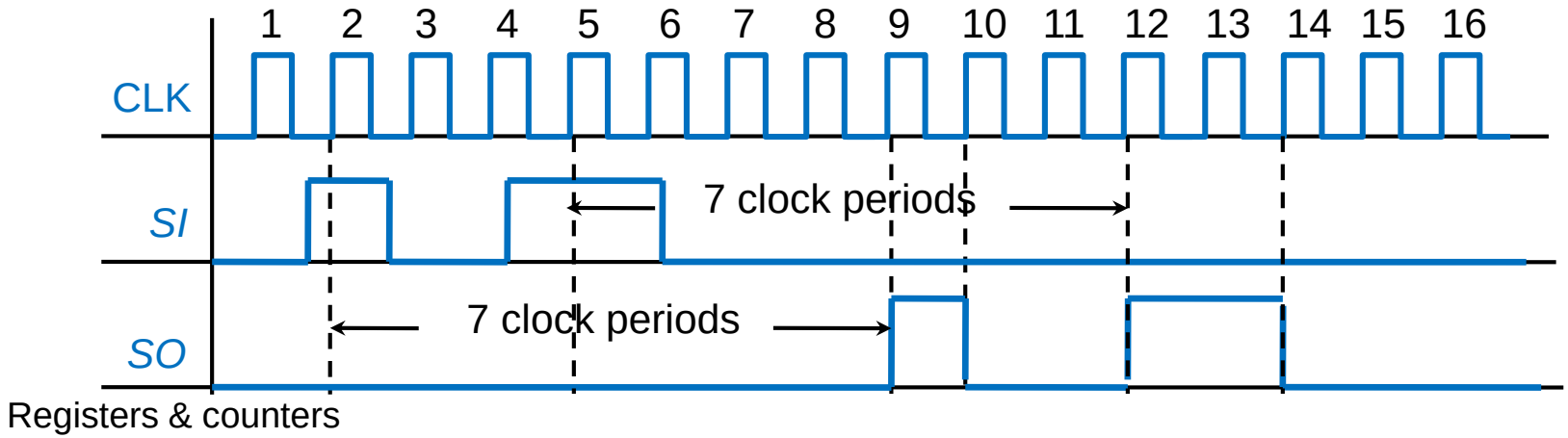
Initial, $Q_3Q_2Q_1Q_0$=0101
SI = 1, 1, 0, 1
$\Rightarrow$ $Q_3Q_2Q_1Q_0$
0101
1010
1101
0110
1011

N-bit Serial-In Serial-Out Shift Registers

- **Take (*n*-1) cycles to output data**

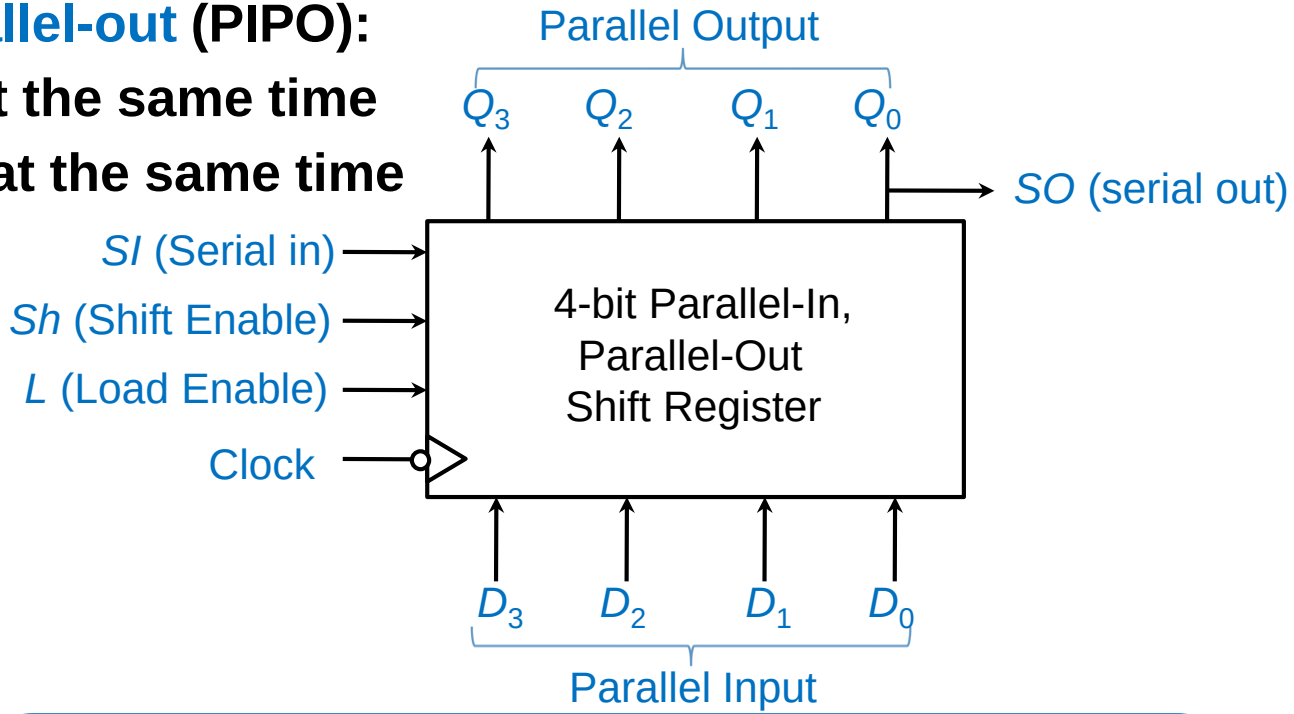






Parallel-In Parallel-Out Right Shift Register

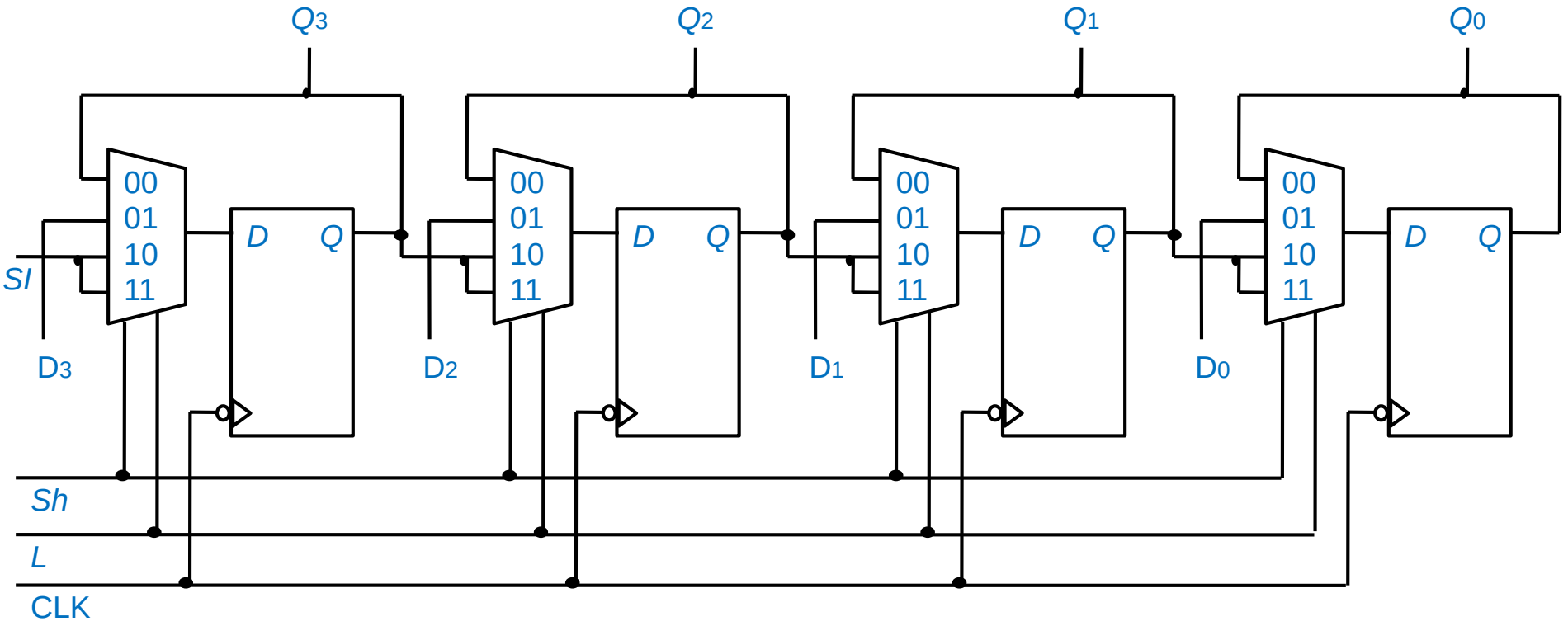
- **Parallel-in parallel-out (PIPO):**
  Load all data at the same time
  Read out data at the same time



Inputs		Next state				Action
Sh(Shift)	*L*(Load)	Q_3^+	Q_2^+	Q_1^+	Q_0^+	
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

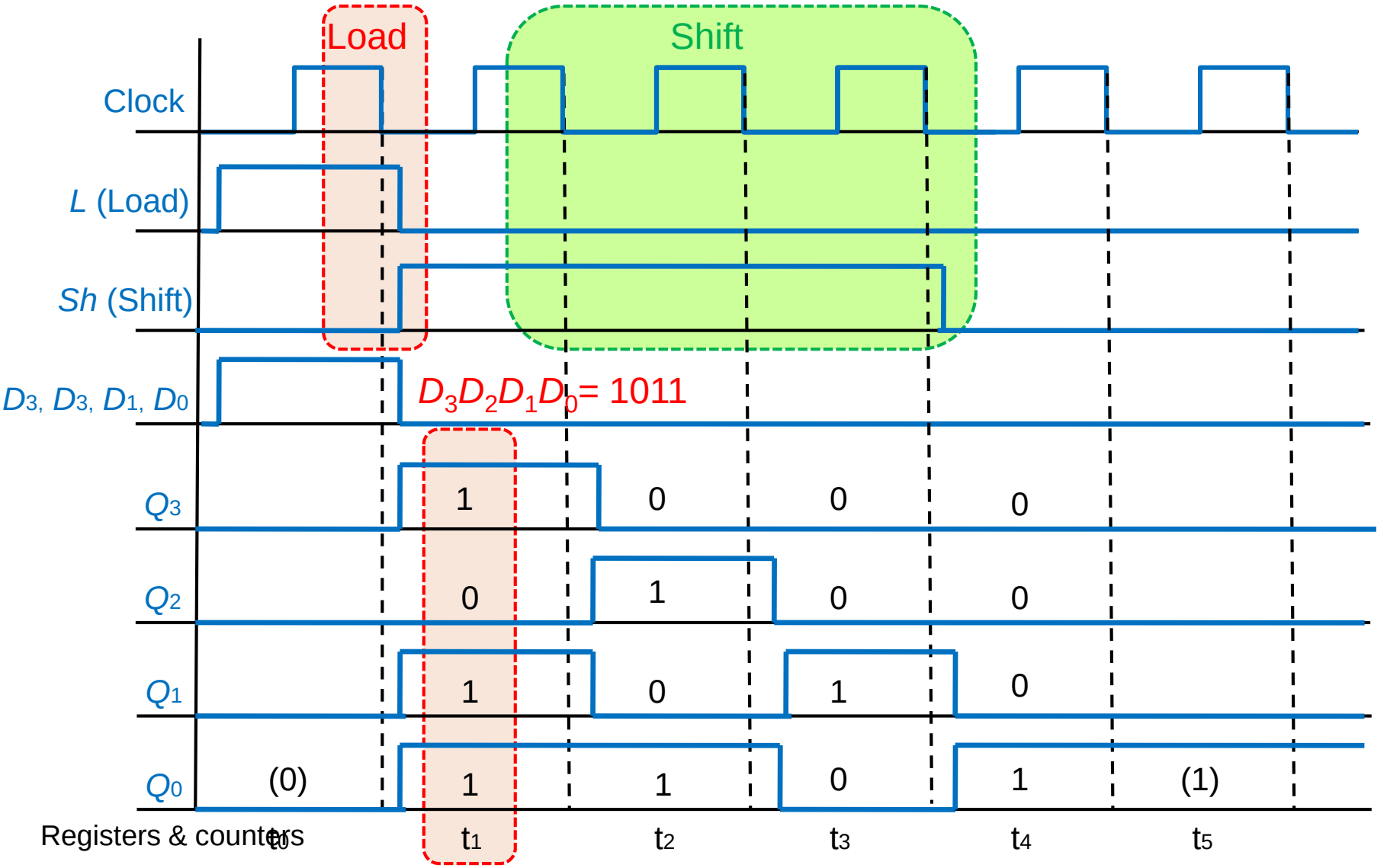
PIPO Shift Register Implementation (1/2)

- **Implement using FFs and MUXes**



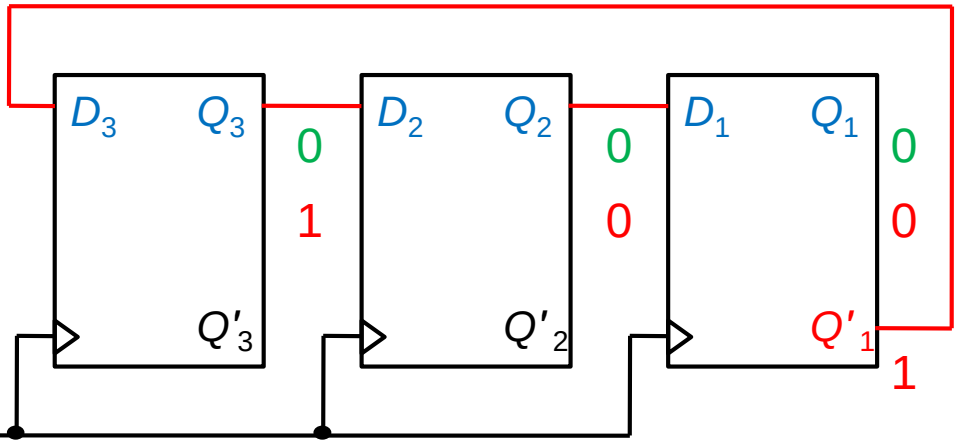
PIPO (2/2)

Sh(Shift)	*L*(Load)	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

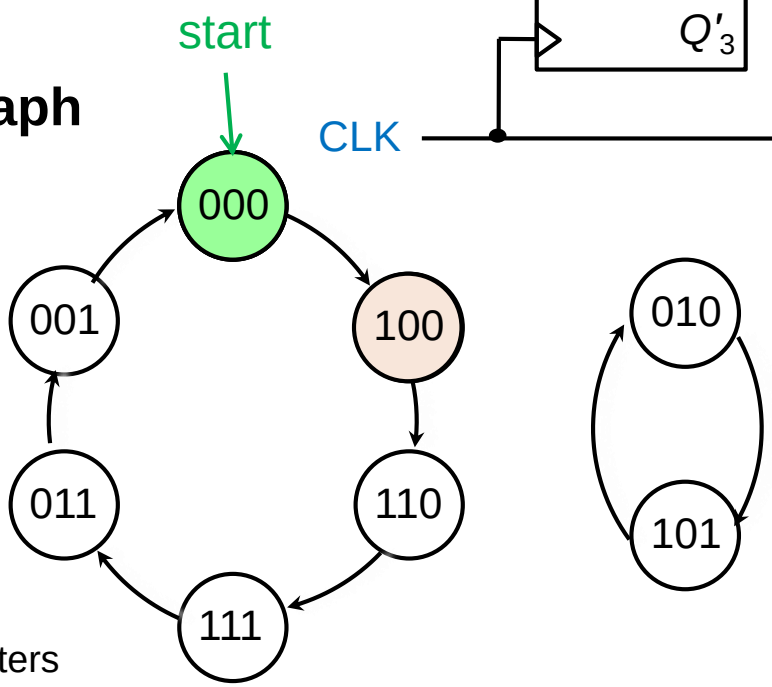


Shift Register with Inverted Feedback

- **Johnson counter: a shift register with inverted feedback**
  - Counter: a circuit that cycles through a fixed sequence of states



- **State graph**

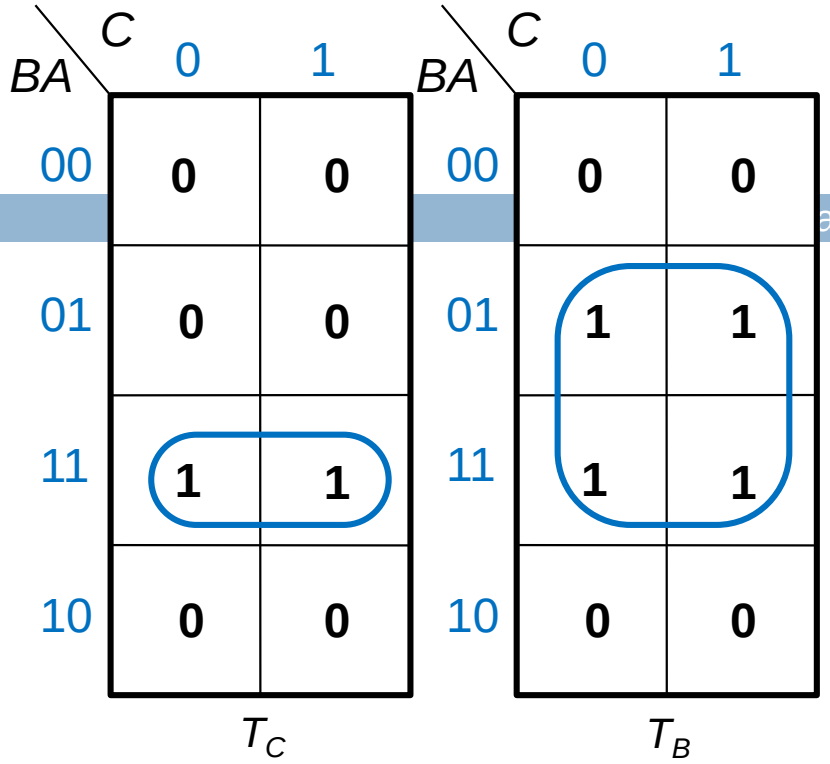


Binary Counters

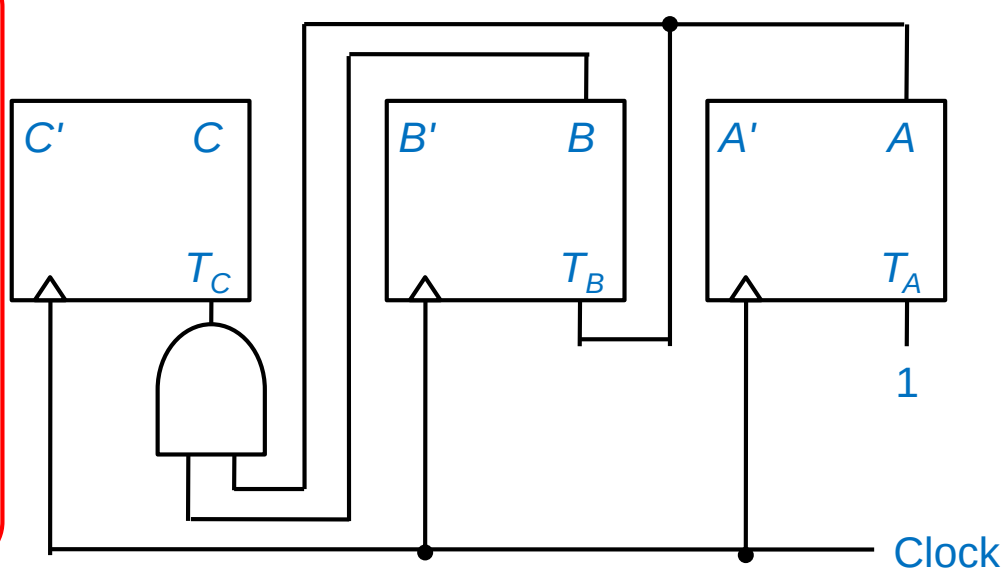
Synchronous counters discussed

Counting 0—7 (1/3)

- **A synchronous counter: FFs are synchronized by a common clock**
  - e.g., count 0~7
    1. T FF?



Present State			Next State			FF Inputs		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



Counting 0—7 (2/3)

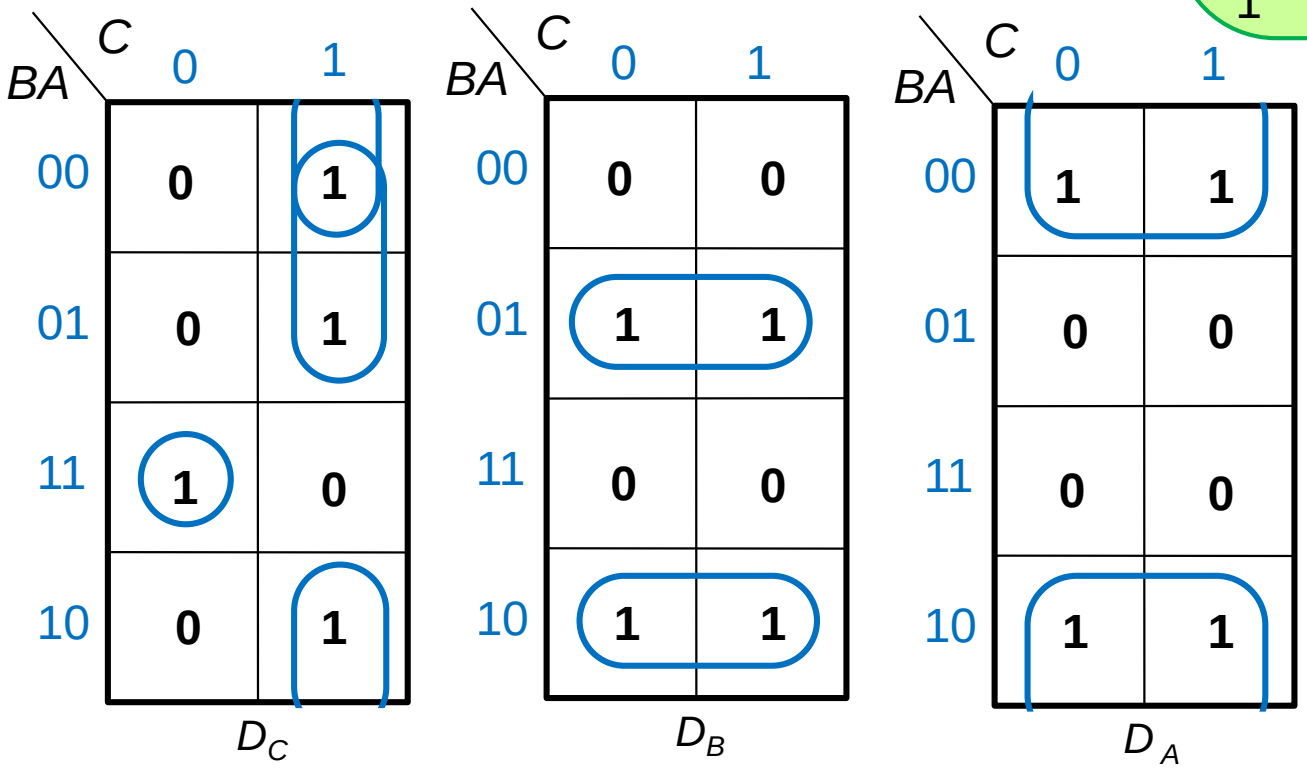
Present State			Next State		
C	B	A	C^+	B^+	A^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

2. D FF?

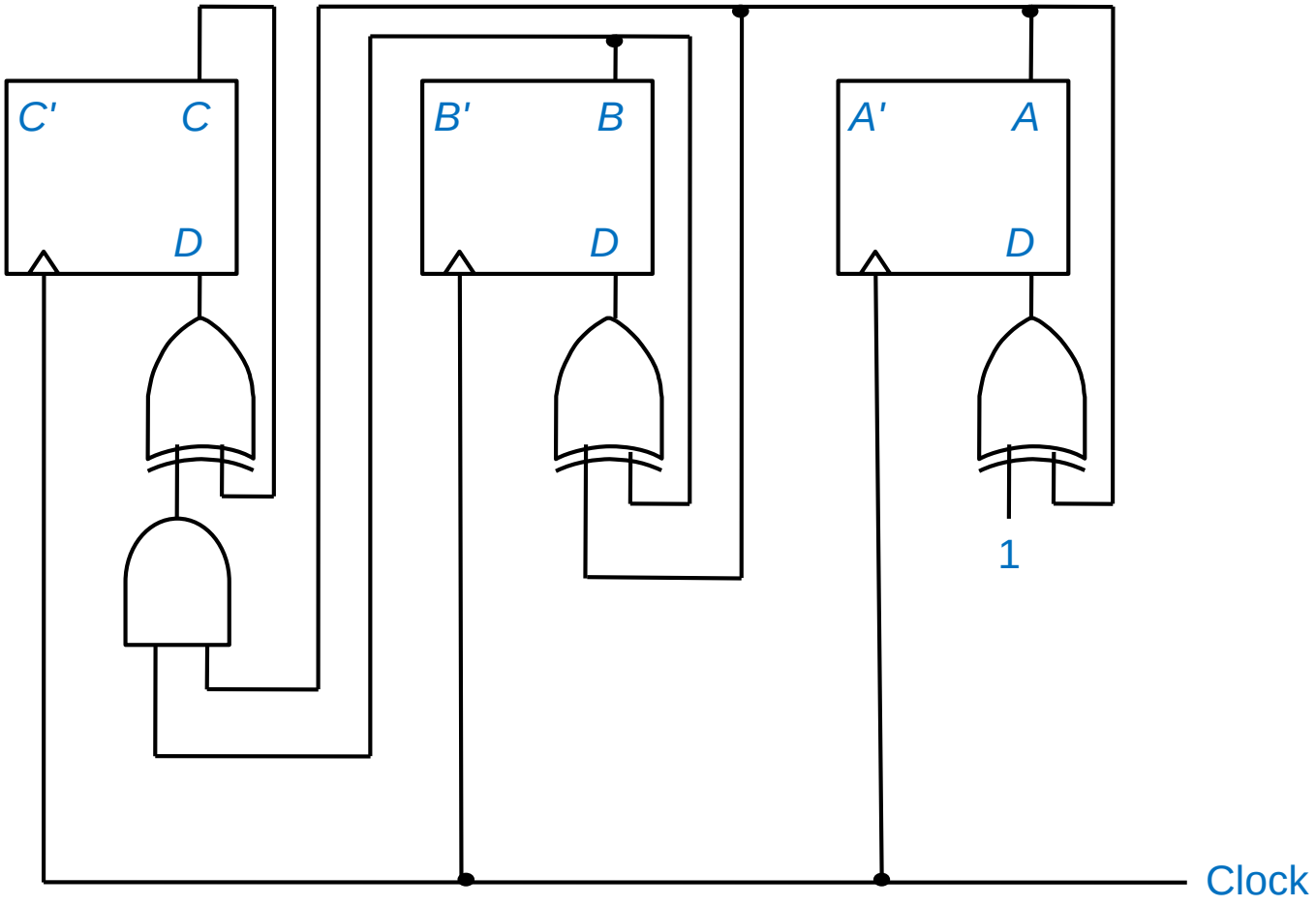
$D_A = A^+ = A'$

$D_B = B^+ = BA' + B'A = B \oplus A$

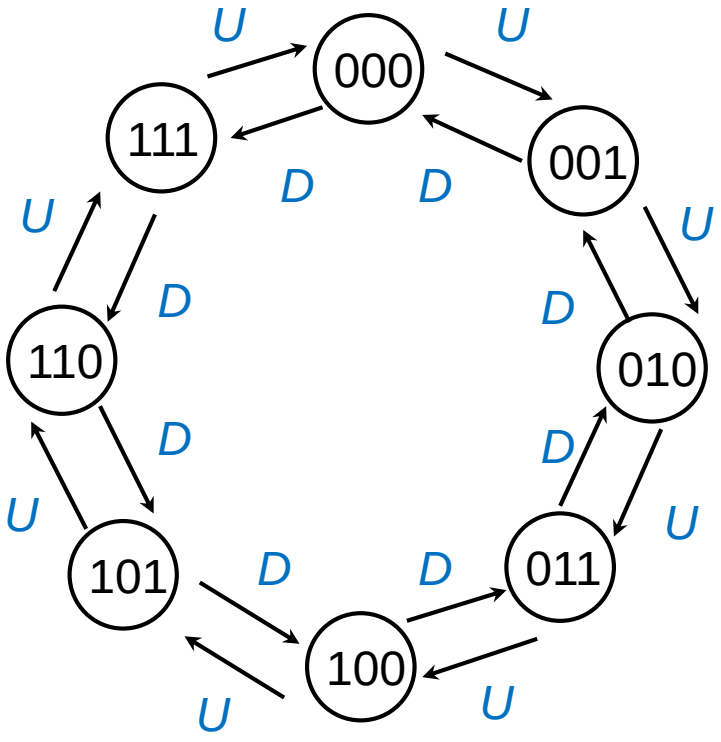
$D_C = C^+ = C'BA + CB' + CA' = C'BA + C(BA)' = C \oplus BA$



Counting 0—7 (3/3)



Up-Down Counter



CBA	$C^+ B^+ A^+$	
	U	D
000	001	111
001	010	000
010	011	001
011	100	010
100	101	011
101	110	100
110	111	101
111	000	110

pp.20

Do not allow:
$U = D = 1$
Up counter:
$U = 1; D = 0$
Down counter:
$U = 0; D = 1$

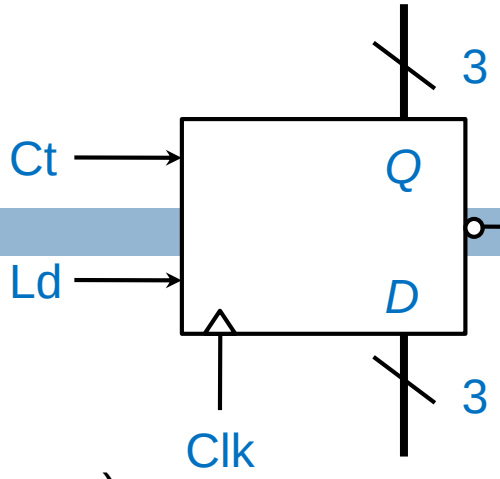
$D_A = A^+ = A \oplus (U + D) = A'$ (A changes state every clock cycle)

$D_B = B^+ = B \oplus (UA + DA') = B \oplus A'$ (B changes state when $A = 0$)

$D_C = C^+ = C \oplus (UBA + DB'A') = C \oplus B'A'$ (C changes state when $B = A = 0$)

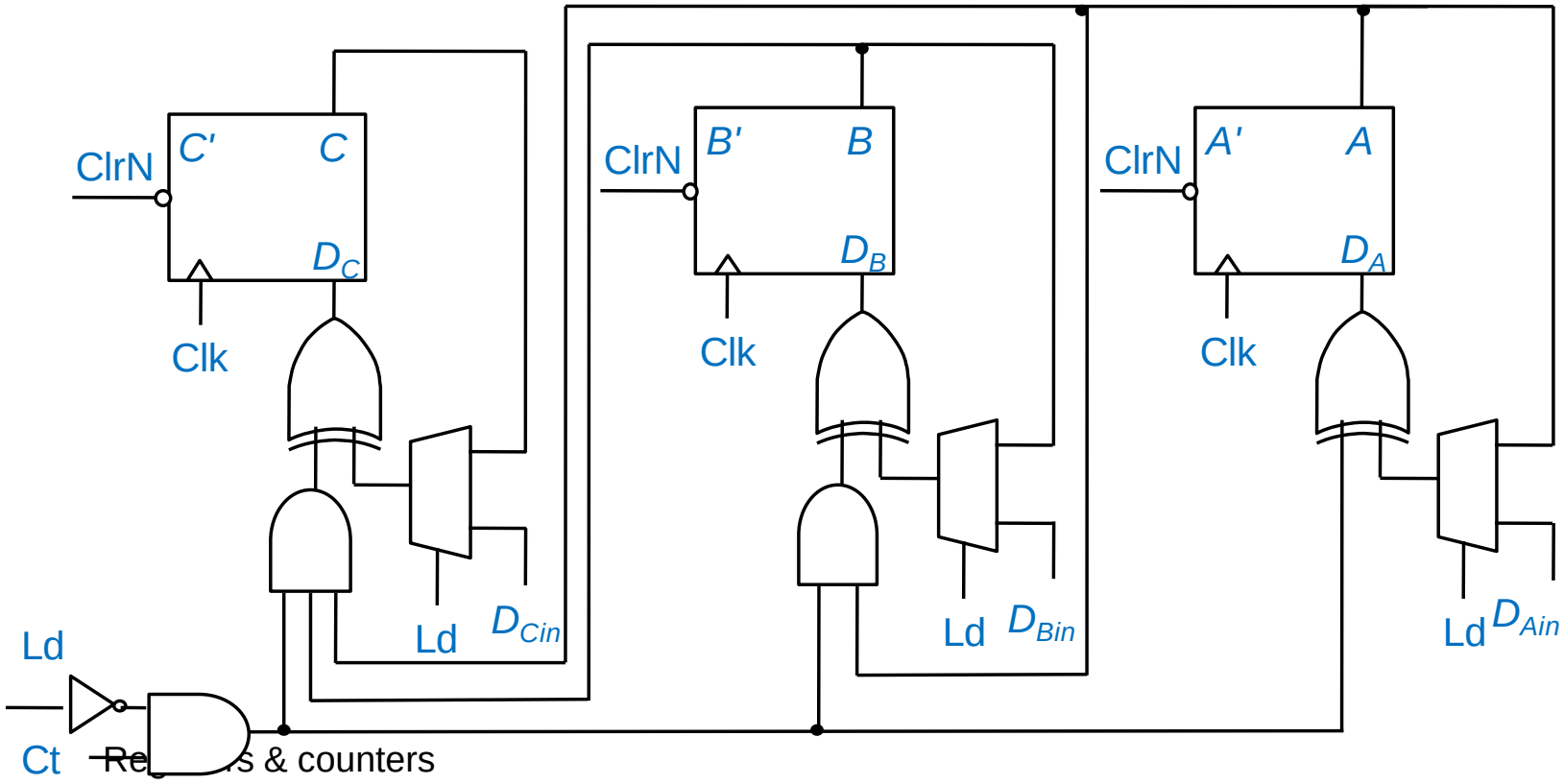
Check Figure 12-18 in the text book for the detailed implementation

Loadable Counter



Ld: Load
Ct: Count
ClrN: Clear

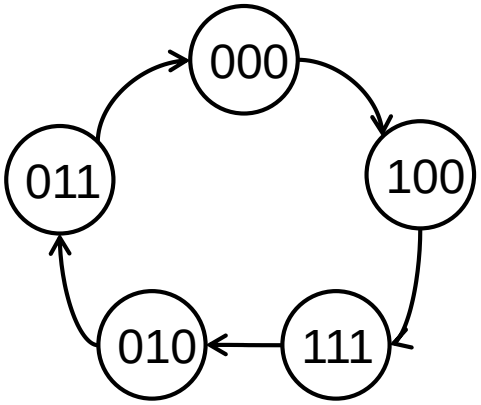
ClrN	Ld	Ct	C^+	B^+	A^+	
0	X	X	0	0	0	
1	1	X	D_C	D_B	D_A	(load)
1	0	0	C	B	A	(no change)
1	0	1	Present state +1			



Counters for Other Sequence

State Diagram of a Counter

- **What if the sequence is not in straight binary order?**



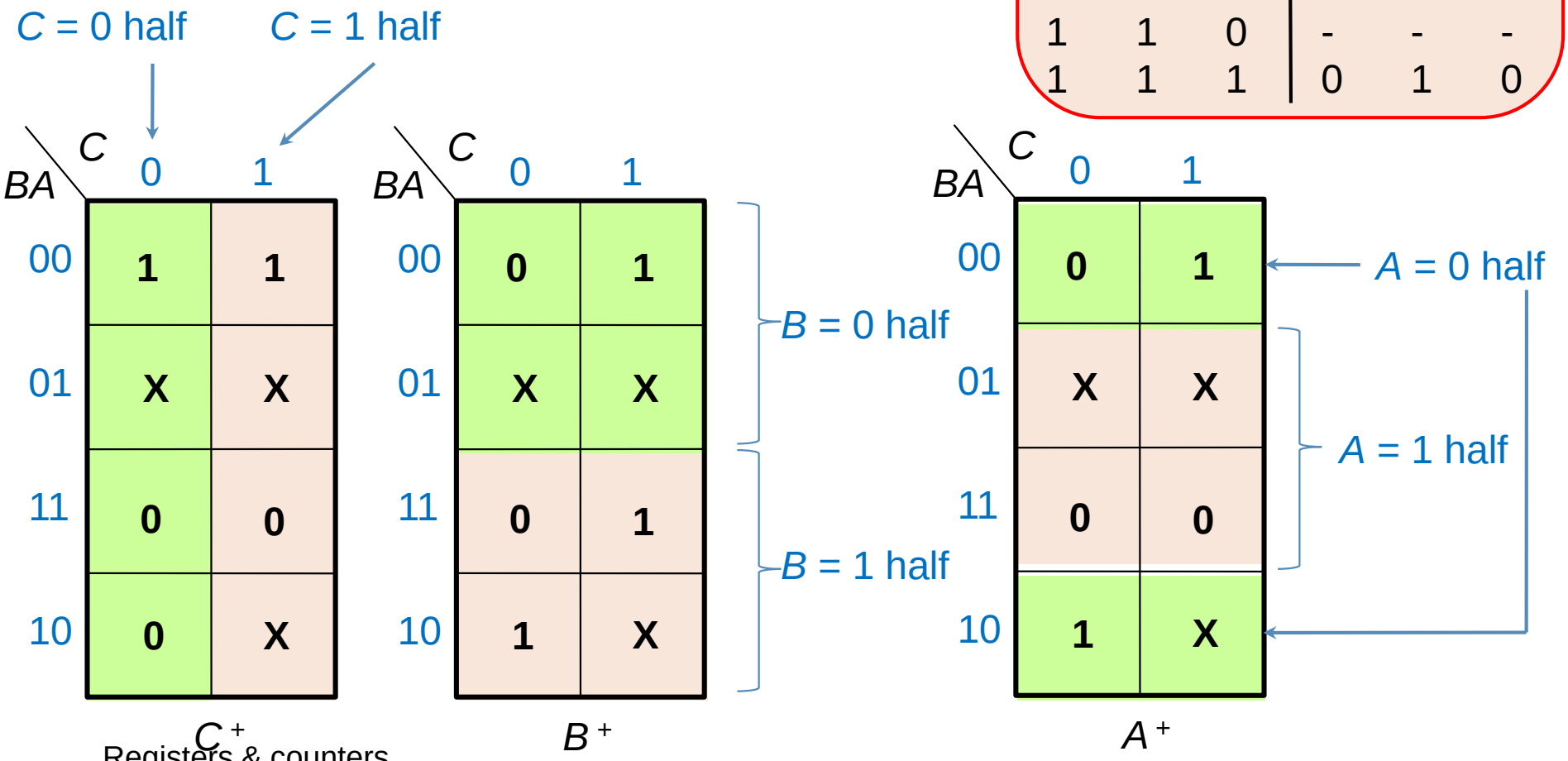
C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	-	-	-
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	-	-	-
1	1	0	-	-	-
1	1	1	0	1	0

Unspecified ⇒ don't care

K-Map Derivation (1/2)

- **Next states**

C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	-	-	-
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	-	-	-
1	1	0	-	-	-
1	1	1	0	1	0



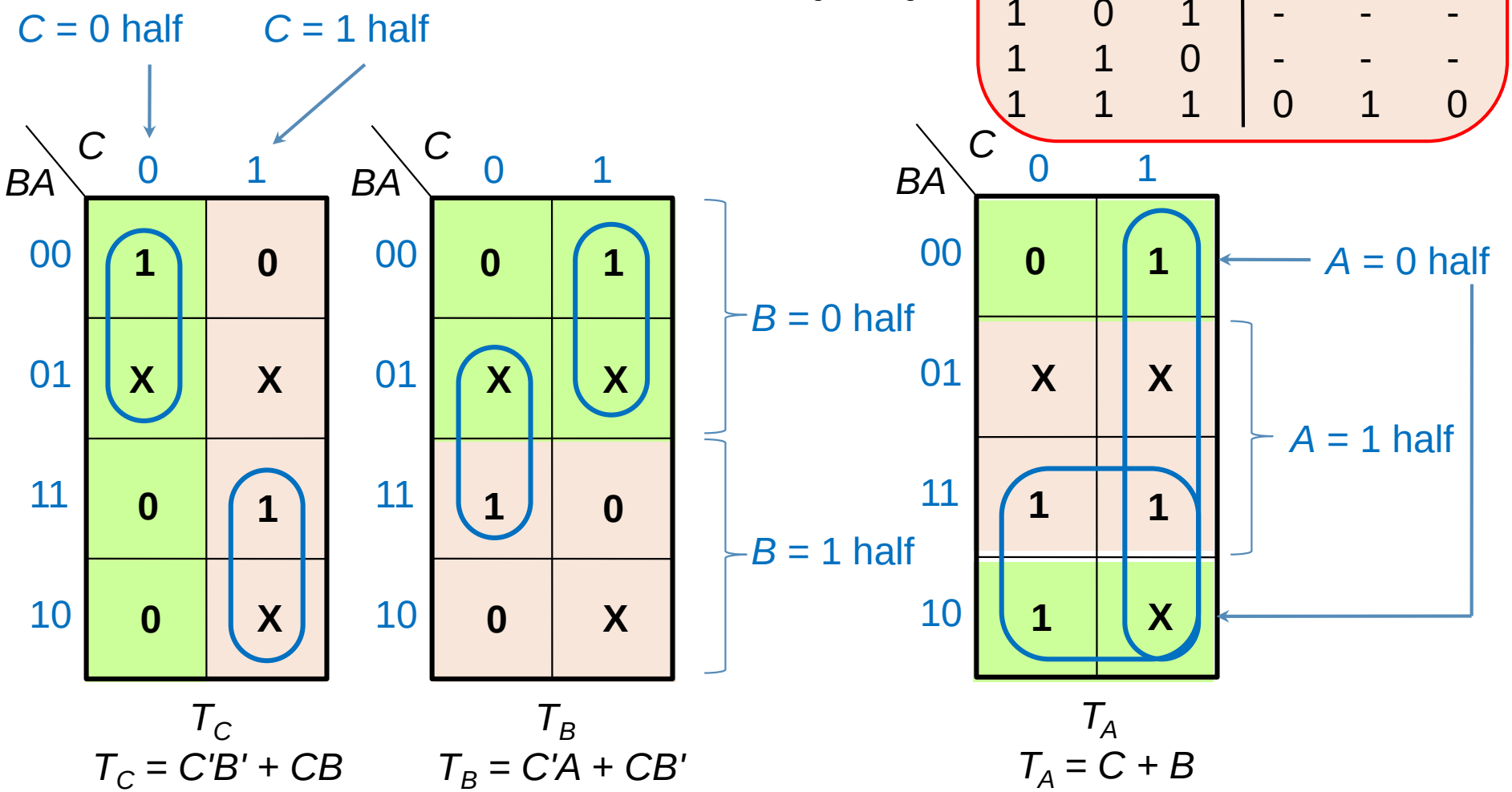
K-Map (2/2)

- **T inputs: $T = Q \oplus Q^+$**

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

$T = Q^+ \oplus Q$

C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	-	-	-
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	-	-	-
1	1	0	-	-	-
1	1	1	0	1	0



T_C

$T_C = C'B' + CB$

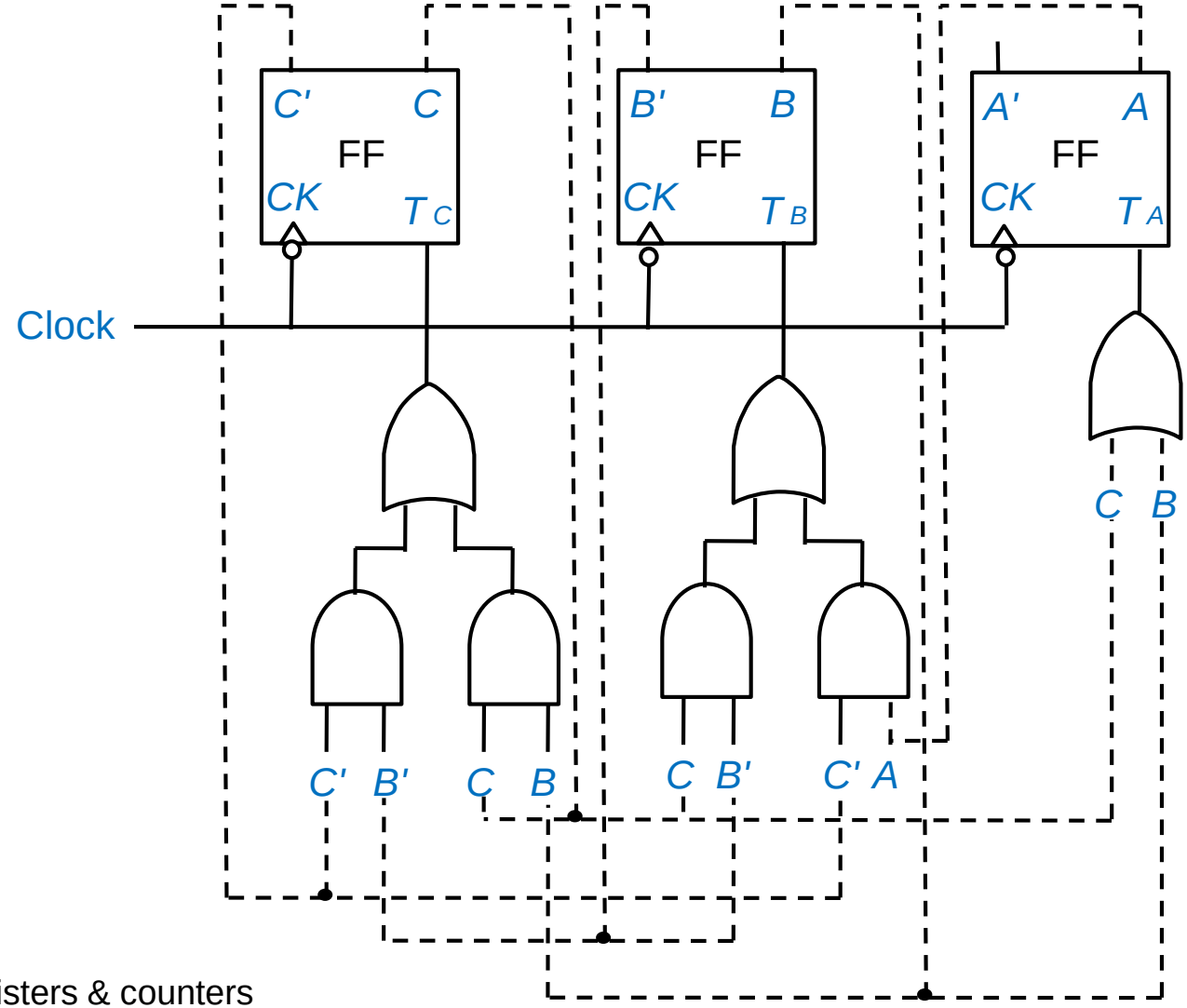
T_B

$T_B = C'A + CB'$

T_A

$T_A = C + B$

Logic



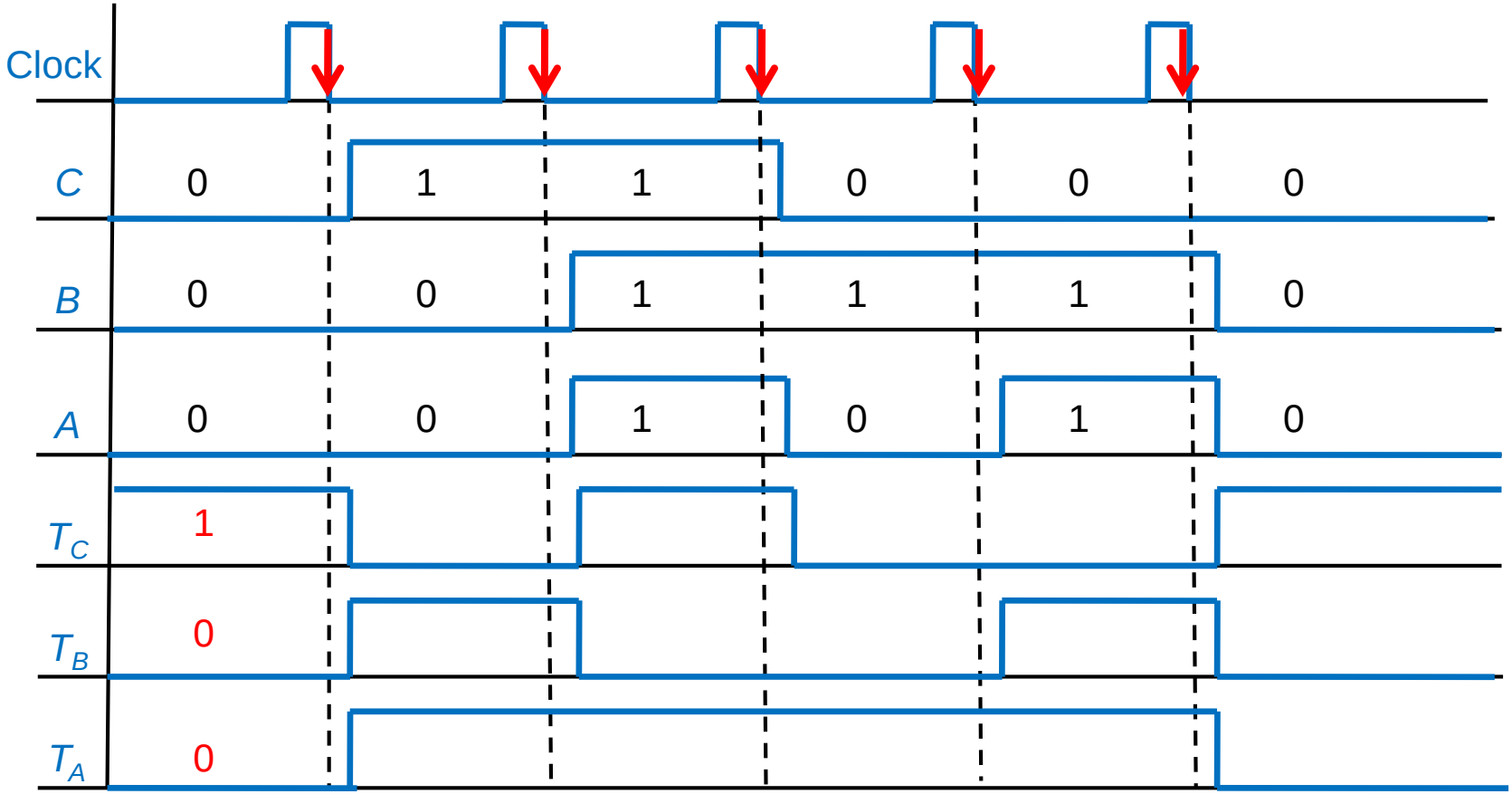
Timing Diagram

- **Negative-edge triggered counter**

$T_C = C'B' + CB$

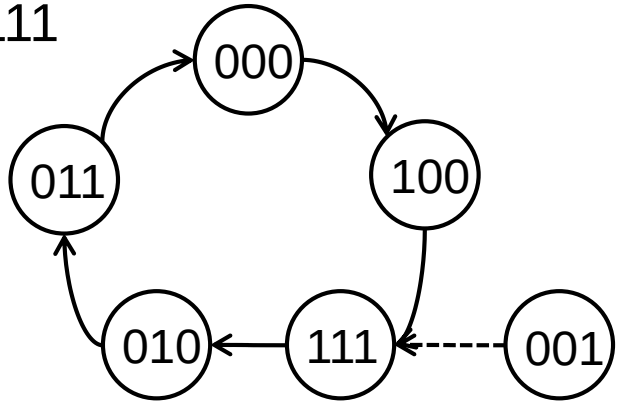
$T_B = C'A + CB'$

$T_A = C + B$



Don’t Care States

- **If FFs are initially set to *CBA*=001**
  - Tracking signals through the network shows that $T_C=T_B=1$, so the state changes to 111



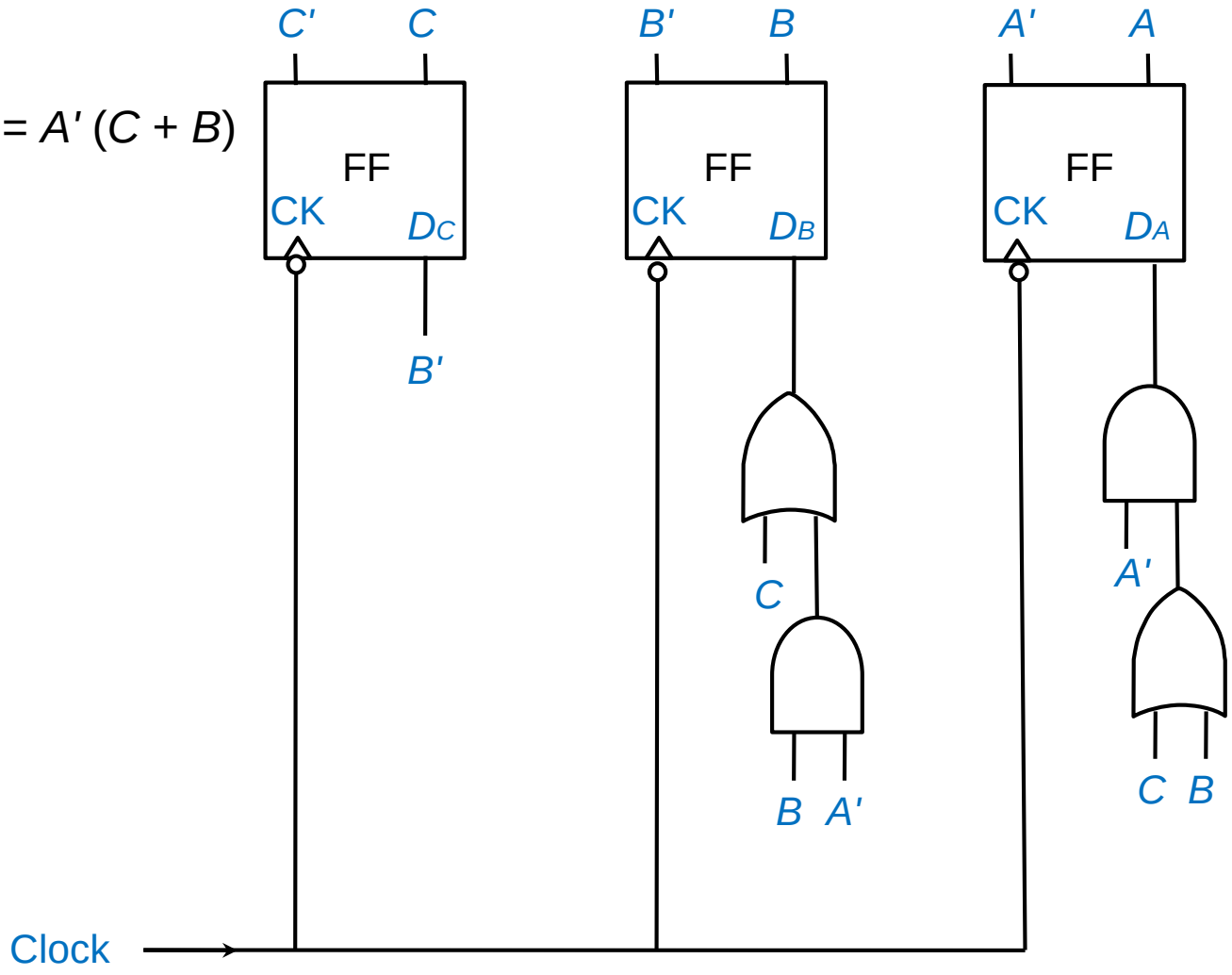
- **When the power is turned on, the initial states of all FFs are unpredictable!!**
  - Don’t care states should be checked to make sure that they eventually lead into the main counting sequence
  - Or use power-up reset

Using D FFs Instead

$D_A = A^+ = CA' + BA' = A'(C + B)$

$D_B = B^+ = C + BA'$

$D_C = C^+ = B'$

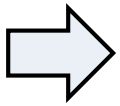


Counter Design Using S-R & J-K FFs

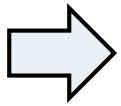
Recap S-R FFs

- **Q: What is the relation between *S*, *R* and *Q*, Q^+?**

S	*R*	*Q*	Q^+	Operation
0	0	0	0	Unchanged
0	0	1	1	
0	1	0	0	Reset *Q* to 0
0	1	1	0	
1	0	0	1	Set *Q* to 1
1	0	1	1	
1	1	0	-	inputs not allowed
1	1	1	-	



Q	Q^+	*S*	*R*
0	0	0	0
		0	1
0	1	1	0
1	0	0	1
1	1	0	0
		1	0



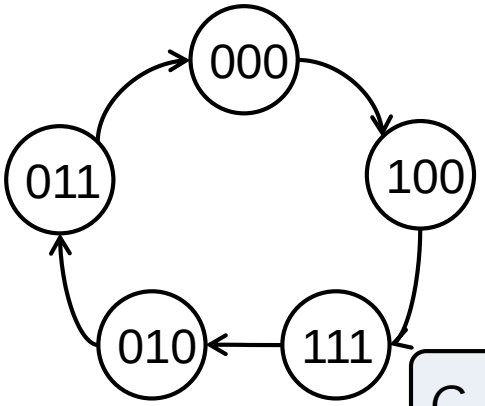
Q	Q^+	*S*	*R*
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Excitation Table

- **However, actually, we do it reversely**
  - ~~*S*, *R* ⇒ *Q*, Q^+~~
  - *S*, *R* ⇐ *Q*, Q^+

Using S-R FFs (1/3)

- **Derive S-R FF input maps from the excitation table**

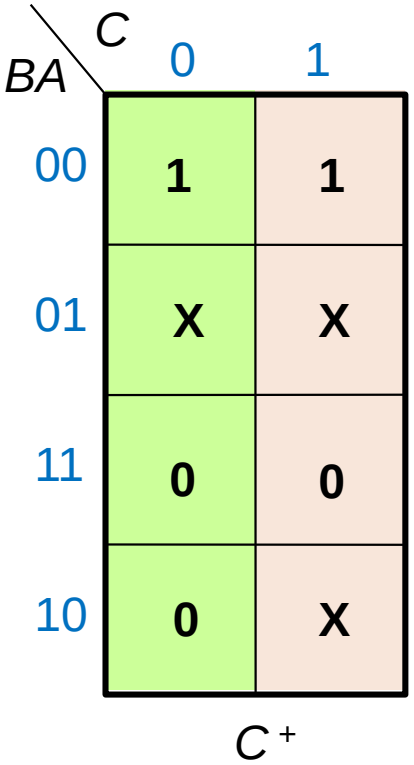


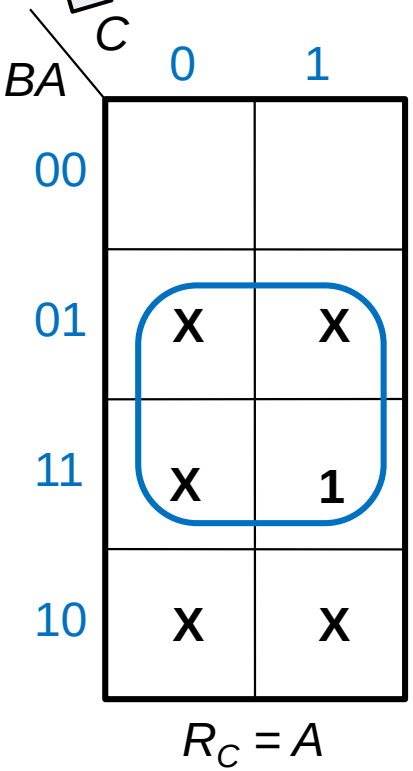
Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

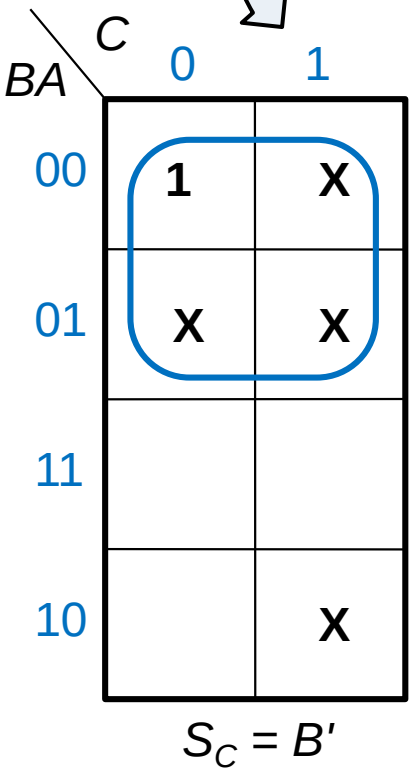
C	B	A	C^+	B^+	A^+	S_C	R_C	S_B	R_B	S_A	R_A
0	0	0	1	0	0	1	0	0	X	0	X
0	0	1	-	-	-	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	0
0	1	1	0	0	0	0	X	0	1	0	1
1	0	0	1	1	1	X	0	1	0	1	0
1	0	1	-	-	-	X	X	X	X	X	X
1	1	0	-	-	-	X	X	X	X	X	X
1	1	1	0	1	0	0	1	X	0	0	1

Using S-R FFs (2/3)

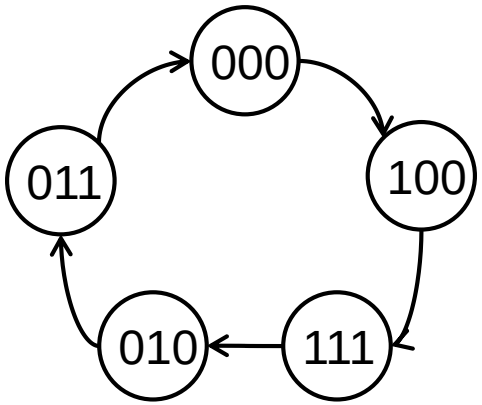
- **Derive S-R FF inputs from next state maps (faster)**







Q	Q+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0



Using S-R FFs (3/3)

BA \ C	0	1
00	1	1
01	X	X
11	0	0
10	0	X

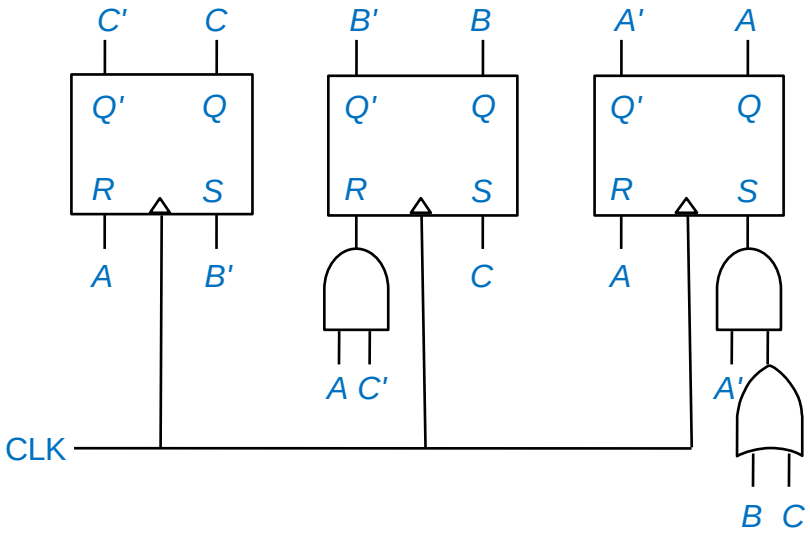
C^+

BA \ C	0	1
00	0	1
01	X	X
11	0	1
10	1	X

B^+

BA \ C	0	1
00	0	1
01	X	X
11	0	0
10	1	X

A^+



BA \ C	0	1
00		
01	X	X
11	X	1
10	X	X

$R_C = A$

BA \ C	0	1
00	1	X
01	X	X
11		
10		X

$S_C = B'$

BA \ C	0	1
00	X	
01	X	X
11	1	
10		X

$R_B = C'A$

BA \ C	0	1
00		1
01	X	X
11		X
10	X	X

$S_B = C$

BA \ C	0	1
00	X	
01	X	X
11	1	1
10		X

$R_A = A$

BA \ C	0
00	1
01	X
11	
10	1 X

$S_A = CA' + BA'$
$= A'(C+B)$

Using J-K FFs (1/2)

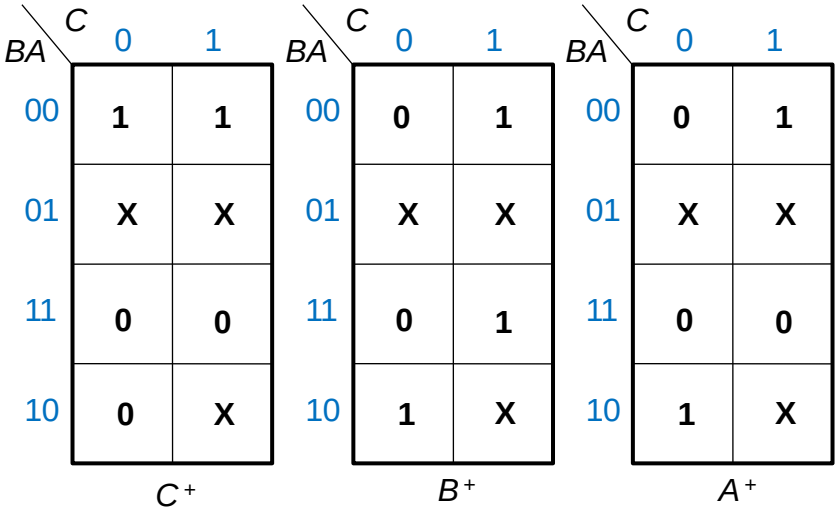
J	K	Q	Q^+	Operation
0	0	0	0	Unchanged
0	0	1	1	
0	1	0	0	Reset Q to 0
0	1	1	0	
1	0	0	1	Set Q to 1
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

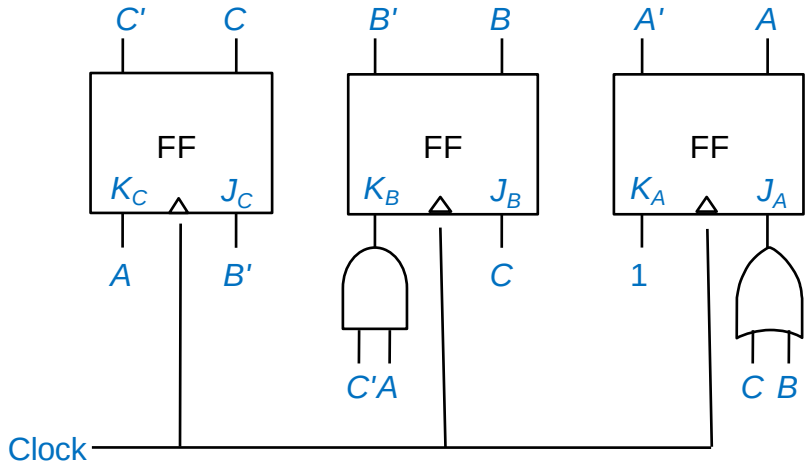
Q	Q^+	J	K
0	0	0	0
		0	1
0	1	1	0
		1	1
1	0	0	1
		1	1
1	1	0	0
		1	0

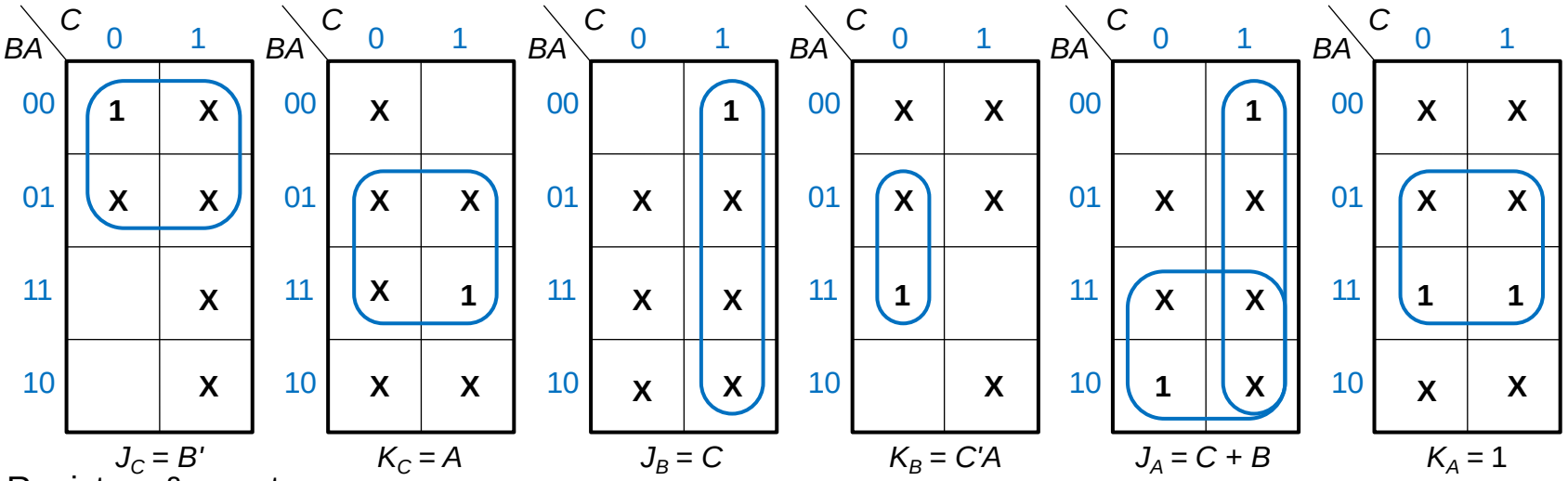
Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

C	B	A	C^+	B^+	A^+	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	-	-	-	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	0	0	X	X	1	X	1
1	0	0	1	1	1	X	0	1	X	1	X
1	0	1	-	-	-	X	X	X	X	X	X
1	1	0	-	-	-	X	X	X	X	X	X
1	1	1	0	1	0	X	1	X	0	X	1

Using J-K FFs (2/2)







Derivation of FF Input Equations

Summary

Derivation of Flip-Flop Input Equations

- **Determine the FF input equations from the next-state equations using K-maps**
  - Always copy X's from next state maps onto input maps first
  - Fill in the remaining squares with Q=0 and Q=1, separately

Type of F/F	Input	Q=0		Q=1		Rules for forming input map from next state map	
		$Q^+=0$	$Q^+=1$	$Q^+=0$	$Q^+=1$	Q=0 Half of Map	Q=1 Half of Map
D	D	0	1	0	1	No change	No change
T	T	0	1	1	0	No change	Complement
S-R	S	0	1	0	x	No change	Replace 1's with x's
	R	x	0	1	0	Replace 0's with x's Replace 1's with 0's	Complement
J-K	J	0	1	x	x	No change	Fill in with x's
	K	x	x	1	0	Fill in with x's	Complement

Important Tables

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

D FF

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

Toggle FF

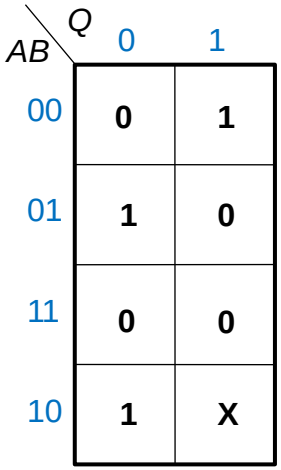
Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

SR FF

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

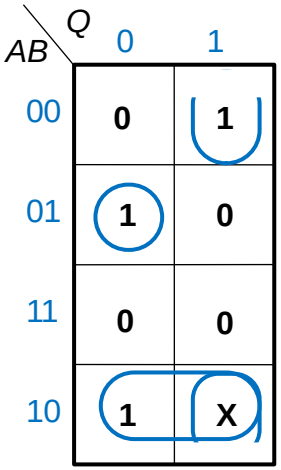
JK FF

Example



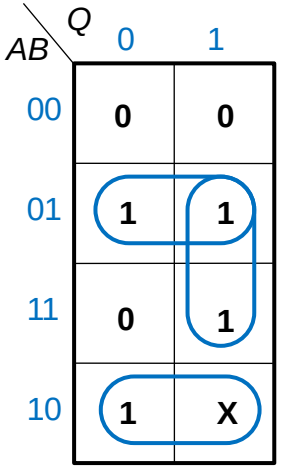
Q^+

Next-state map



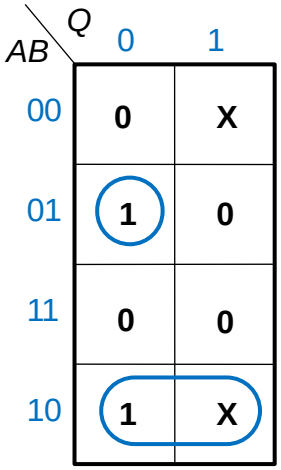
$D = Q'A'B + QB' + AB'$

D input map

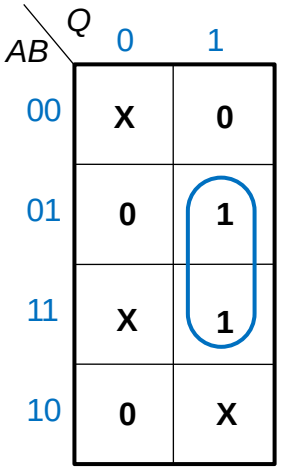


$T = A'B + AB' + QB$

T input map

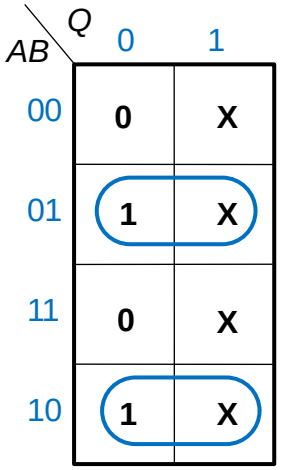


$S = AB' + Q'A'B$

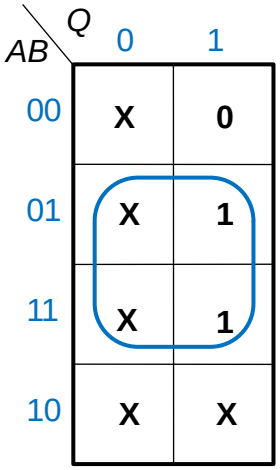


$R = QB$

S-R input maps



$J = A'B + AB'$

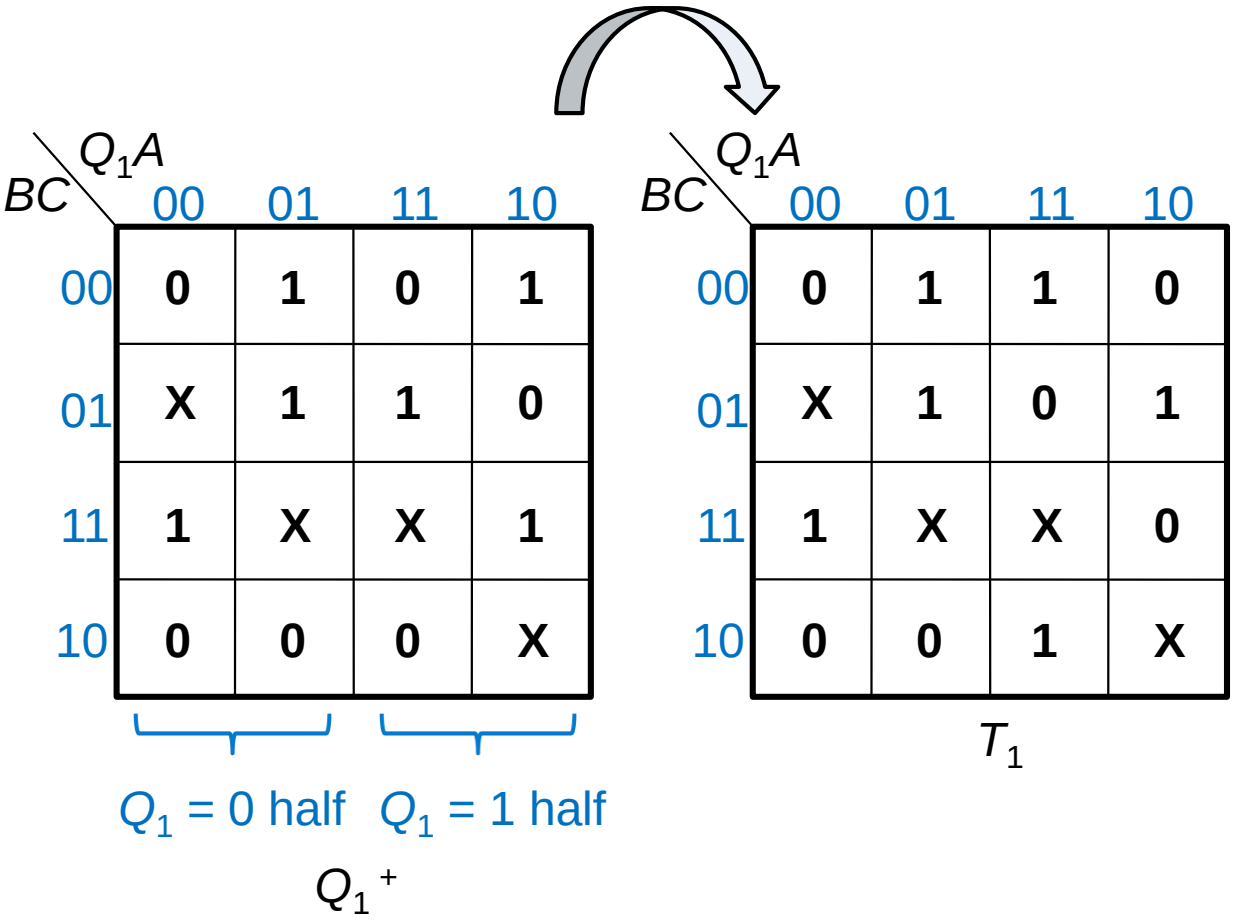


$K = B$

J-K input maps

Example: 4-Variable Maps (1/3)

- **4-variable maps (A, B, C, Q_i)) using 3 different type FFs**



Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

Toggle

Example (2/3)

Q_2=0 half

Q_2=1 half

CQ_2 \ AB	00	01	11	10
00	1	X	1	0
01	0	0	X	1
11	1	0	X	1
10	X	0	0	1

Q_2^+

CQ_2 \ AB	00	01	11	10
00	1	X	1	0
01	0	0	X	X
11	X	0	X	X
10	X	0	0	1

S_2

CQ_2 \ AB	00	01	11	10
00	0	X	0	X
01	1	1	X	0
11	0	1	X	0
10	X	X	X	0

R_2

Q Q^+	S R
0 0	0 X
0 1	1 0
1 0	0 1
1 1	X 0

Set-Reset

Example (3/3)

Q_3C \ AB	00	01	11	10
00 (Q_3=0 half)	0	0	1	X
01 (Q_3=0 half)	0	1	X	1
11 (Q_3=1 half)	X	X	0	0
10 (Q_3=1 half)	1	1	1	0

Q_3^+

Q_3C \ AB	00	01	11	10
00	0	0	1	X
01	0	1	X	1
11	X	X	X	X
10	X	X	X	X

$J_3 = A + BC$

Q_3C \ AB	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	1	1
10	0	0	0	1

$K_3 = C + AB'$

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Jump-Clear