DIGITAL SYSTEM I

Lectures:

Professor : Dr A. Karmouch, office CBY A508

Chapter 6

Registers & Counters

Registers

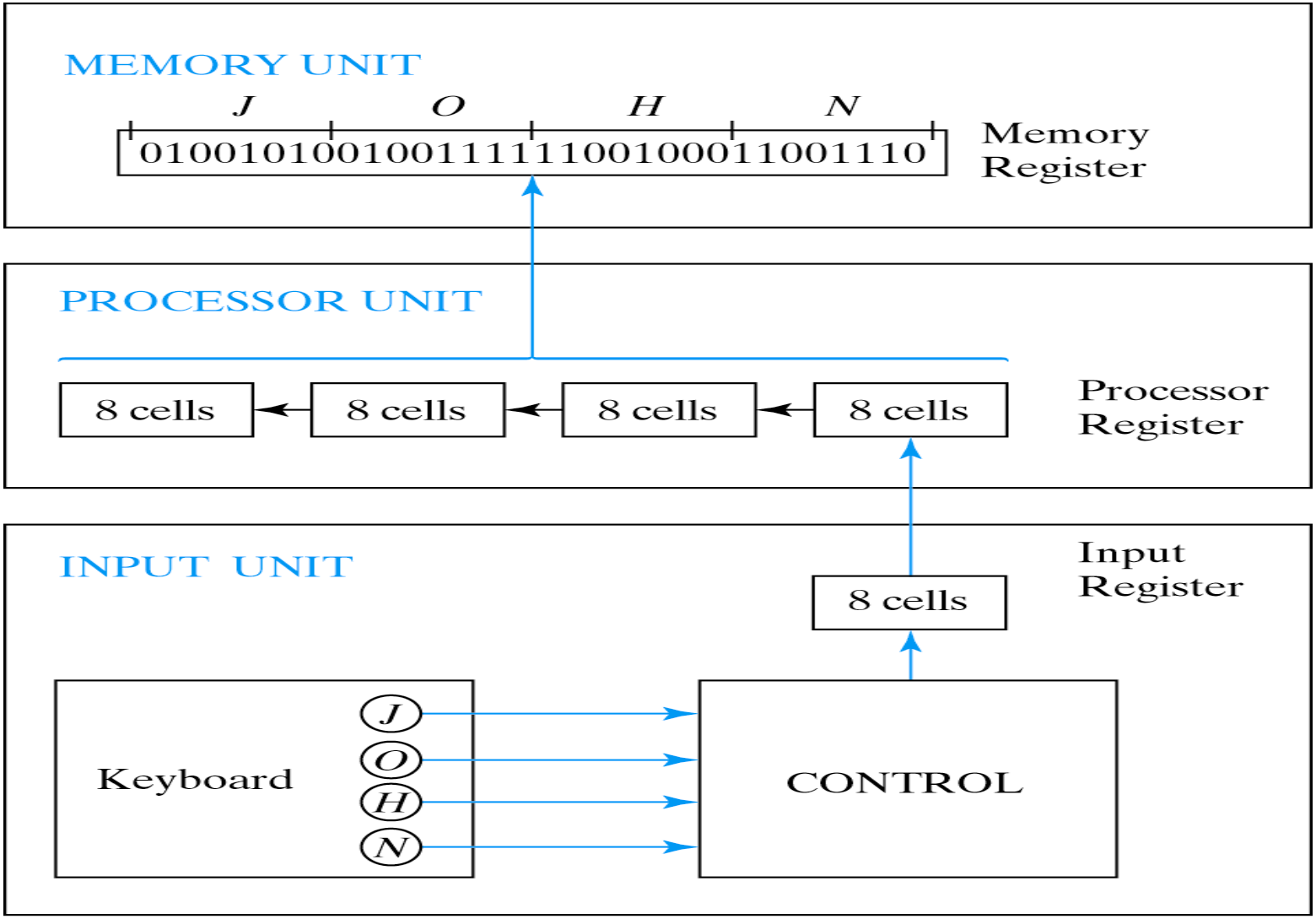


Fig. 1-1 Transfer of information with registers

Registers

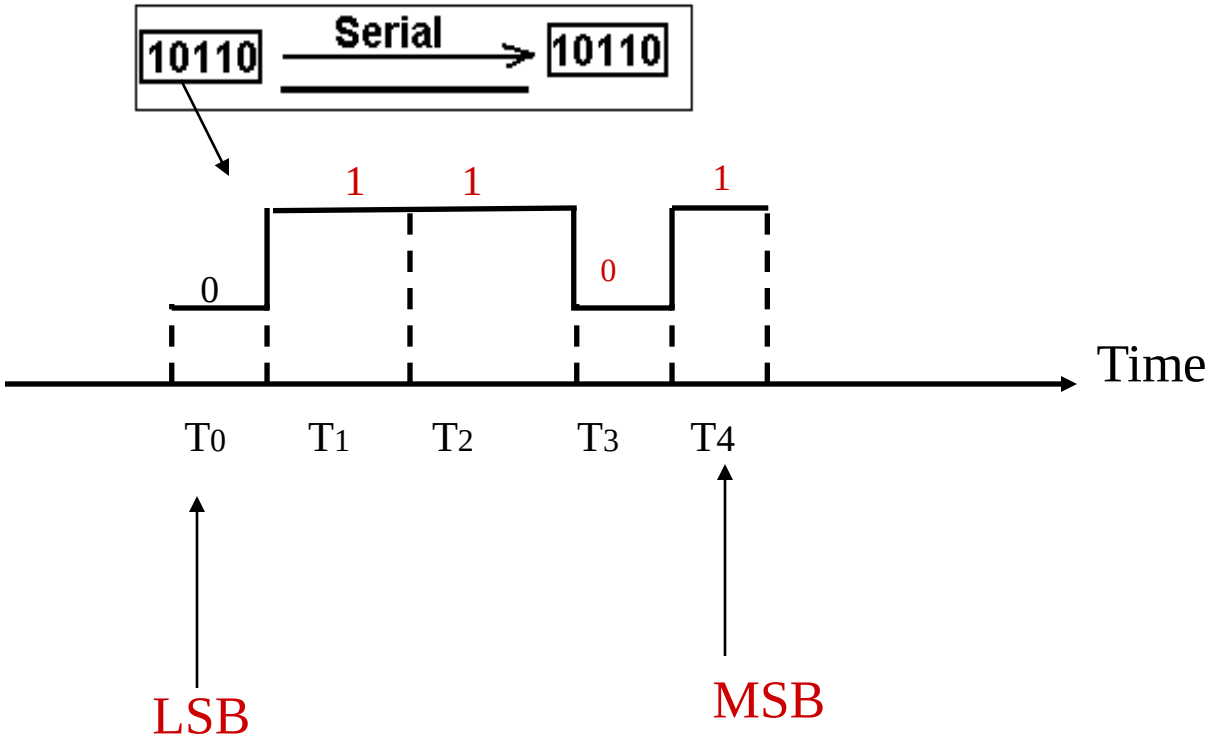
- Multiple flip flops can be combined to form a data ***register***
- **Shift registers** allow data to be transported one bit at a time
- **Registers** also allow for parallel transfer

  →Many bits transferred at the same time
- Basic components of most computers

Parallel versus Serial

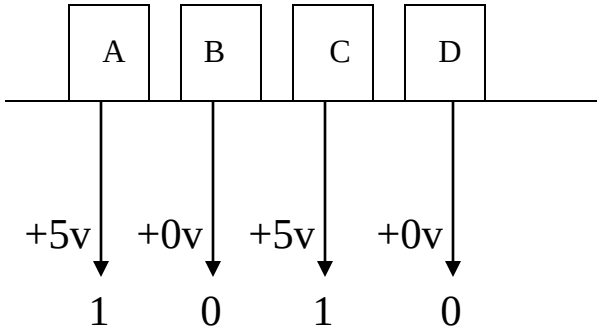
→Serial communications, transfers a binary number as a sequence of binary digits, one after another, through one data line.

- One Circuit is necessary to represent any binary number

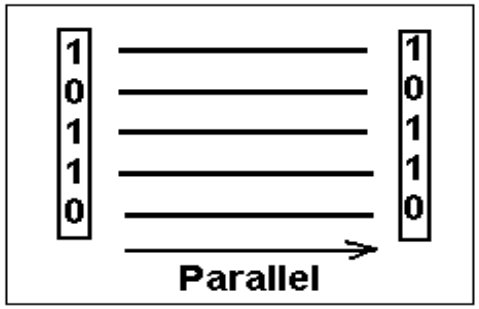


Parallel versus Serial

→ Parallel communications

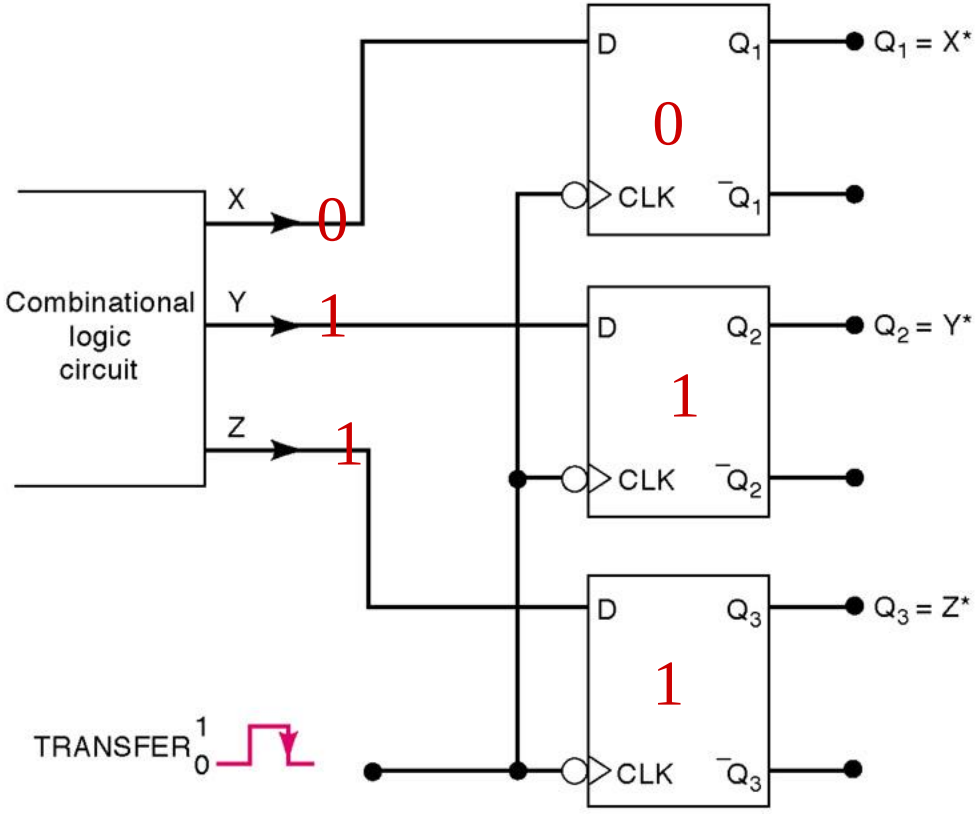


–Transfers a binary number through multiple data lines at the same time.



Parallel Data Transfer

- In this example Flip flops D store outputs from combinational logic that has 3 outputs. 3 flip flops are required
- Multiple flops can store a collection of binary data



Register with Parallel Load

- Register: Group of Flip-Flops
- Ex: D Flip-Flops
- Holds 4 bits of Data
- Loads in Parallel on Clock Transition
- Asynchronous Clear (Reset)

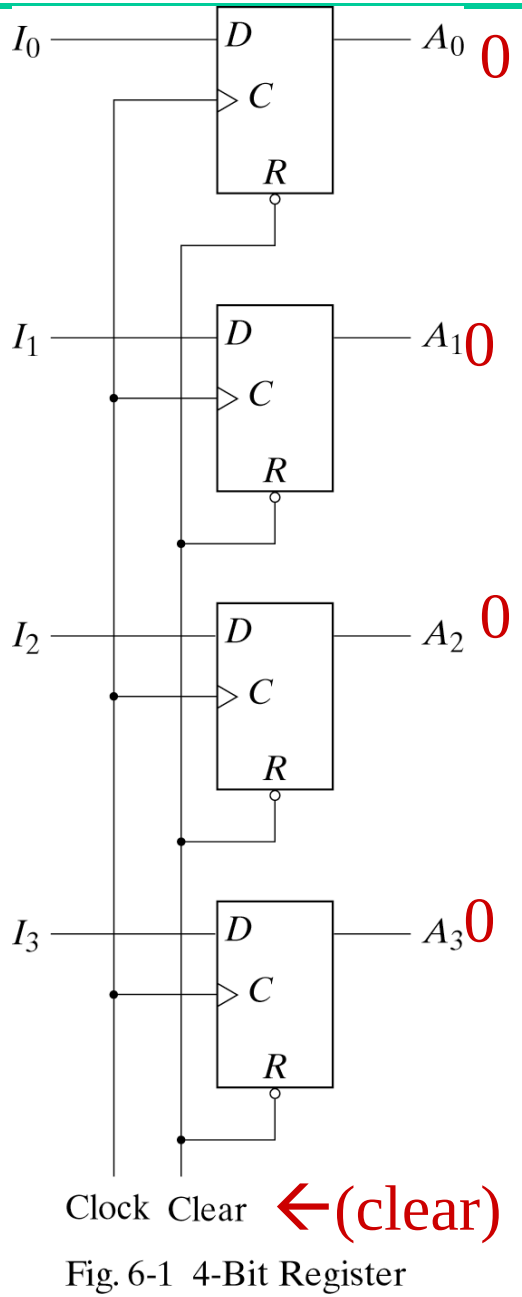


Fig. 6-1 4-Bit Register

Register with Parallel Load

- Register: Group of Flip-Flops
- Ex: D Flip-Flops
- Holds 4 bits of Data
- Loads in Parallel on Clock Transition
- Asynchronous Clear (Reset)

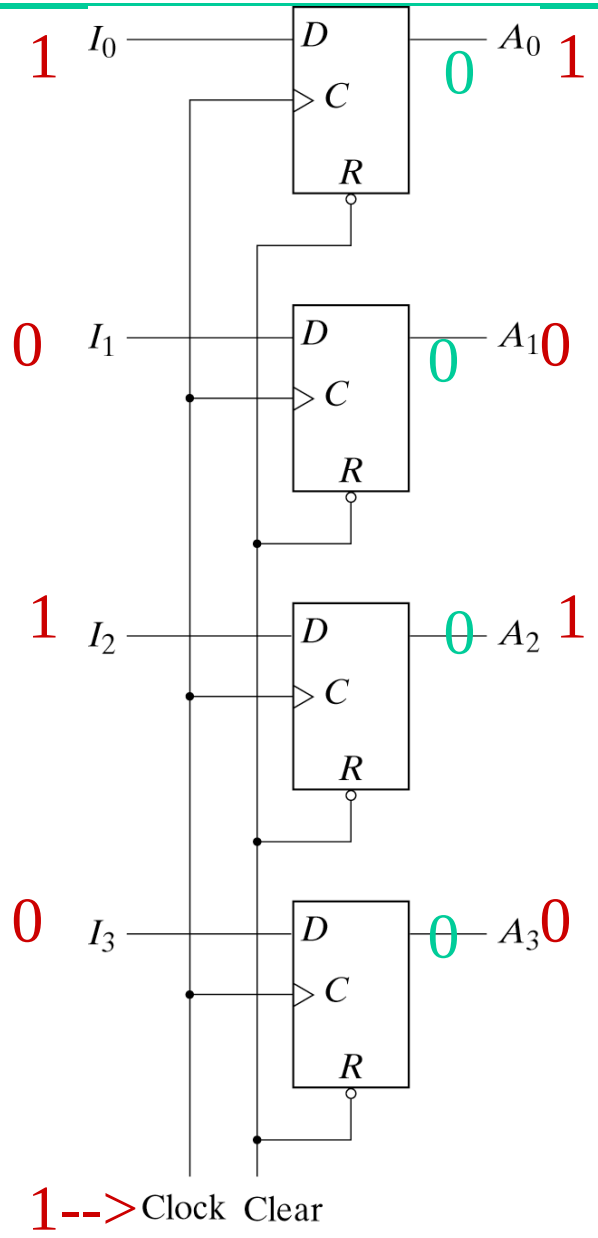
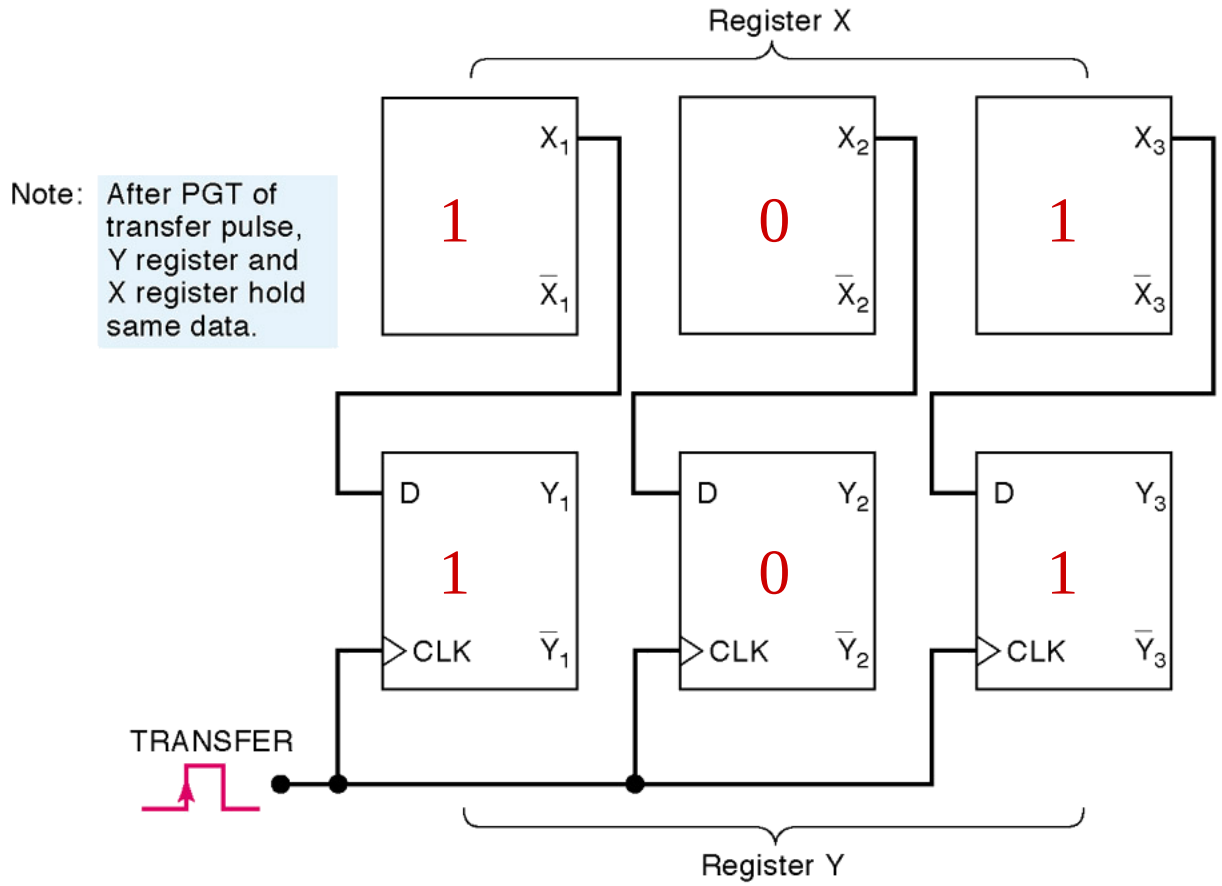


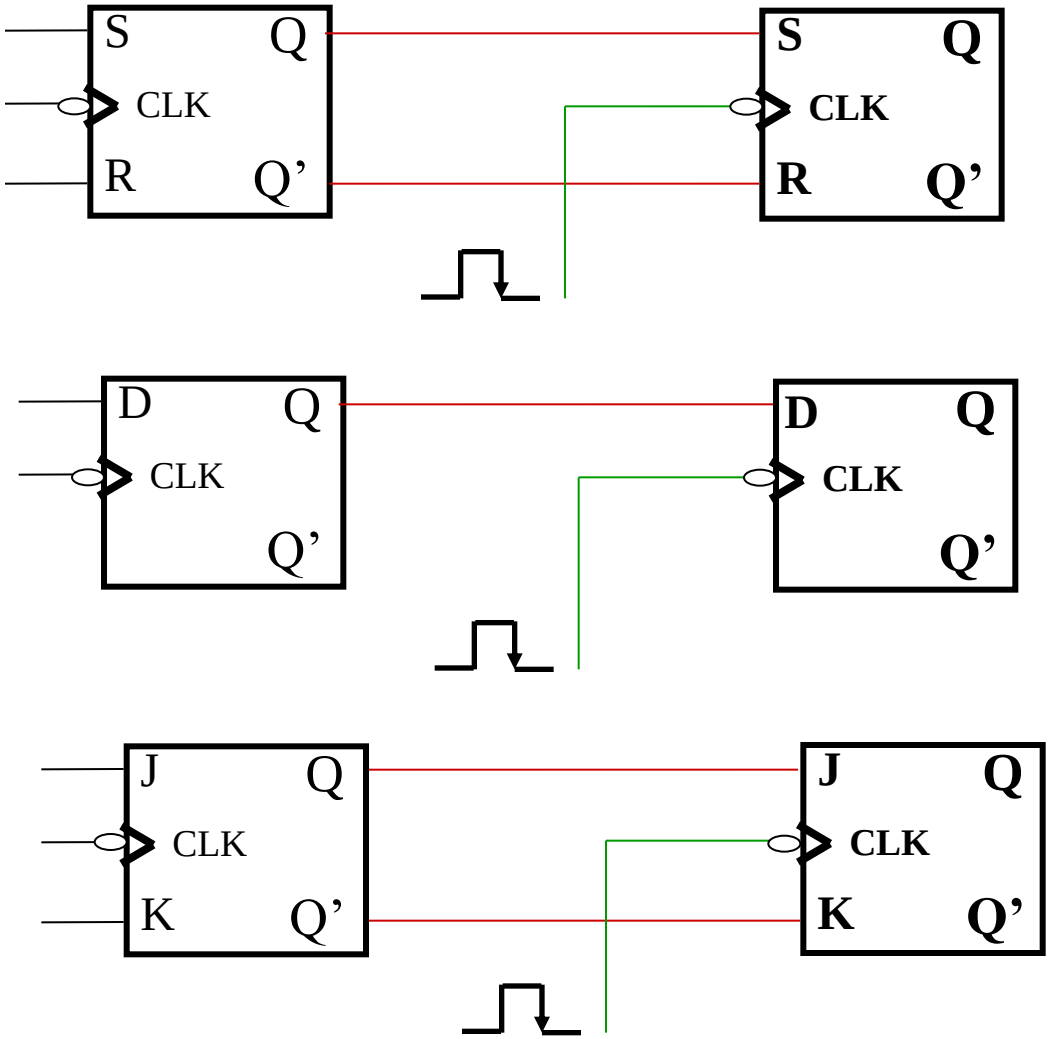
Fig. 6-1 4-Bit Register

Parallel Data Transfer

- All data is transferred on one positive edge
- Data stored into register Y



Serial Transfer



Shift Registers

- Cascade chain of Flip-Flops
- Bits travel on positive edges (in this example)
- Serial in (SI) – Serial out (SO)

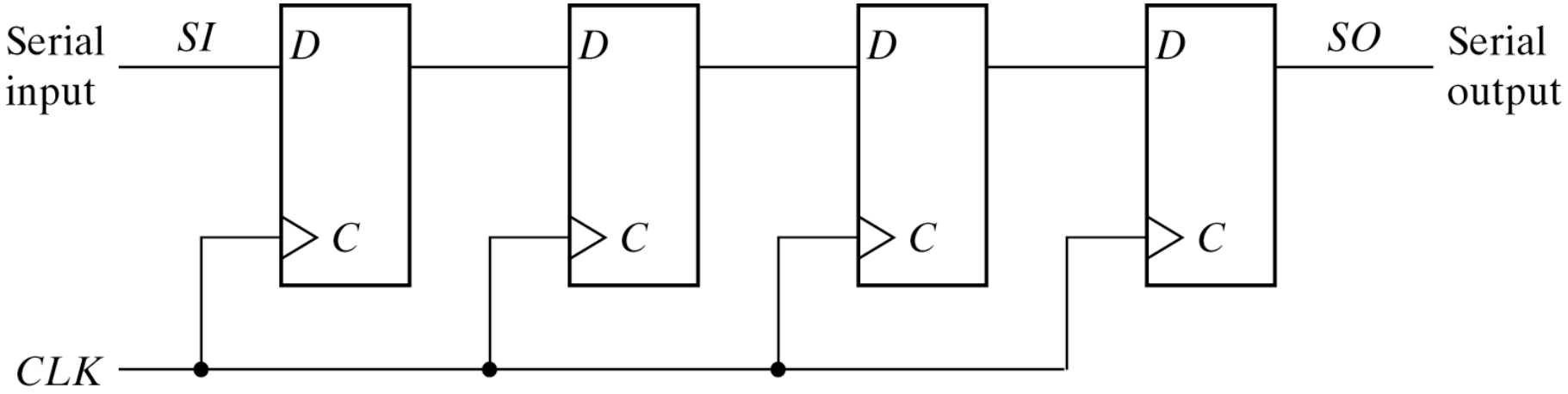
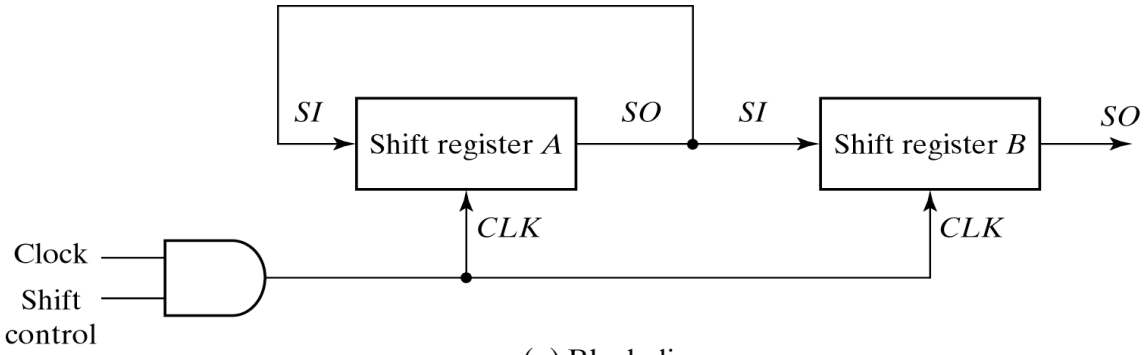


Fig. 6-3 4-Bit Shift Register

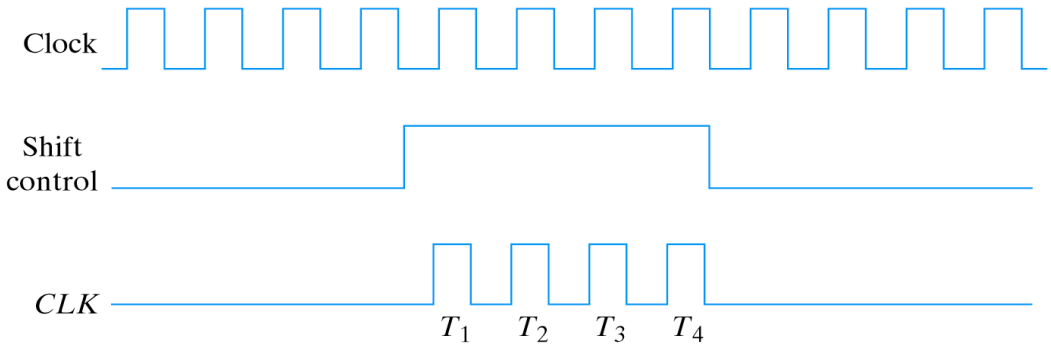
Serial Transfer

- Data is transferred one bit at a time
- Note the data loop back for register A

Time	A	B
T0	1011	0011
T1		
T2		
T3		
T4		



(a) Block diagram



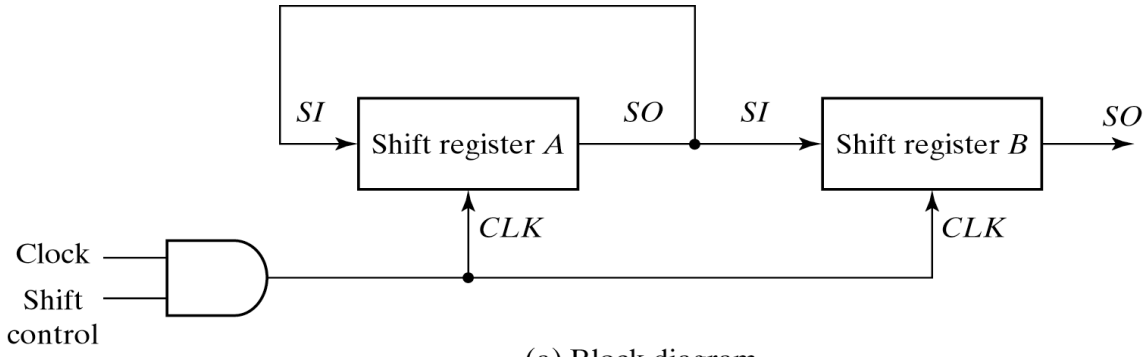
(b) Timing diagram

Fig. 6-4 Serial Transfer from Register *A* to register *B*

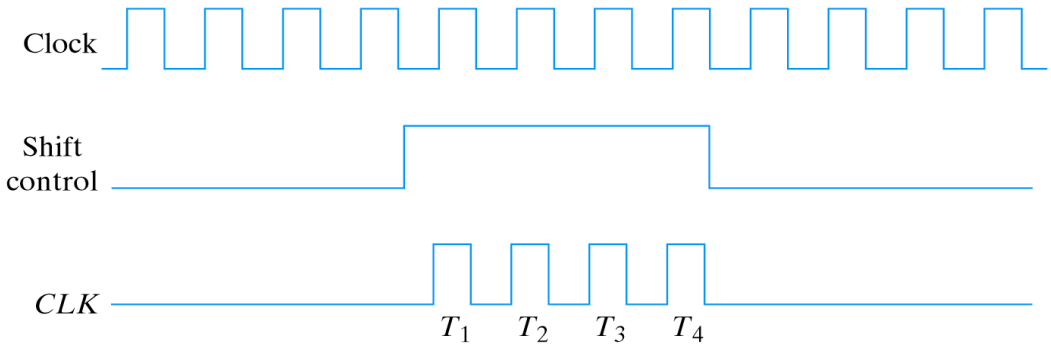
Serial Transfer

- Data is transferred one bit at a time
- Note the data loop back for register A

Time	A	B
T0	1011	0011
T1	1101	1001
T2	1110	1100
T3	0111	0110
T4	1011	1011



(a) Block diagram

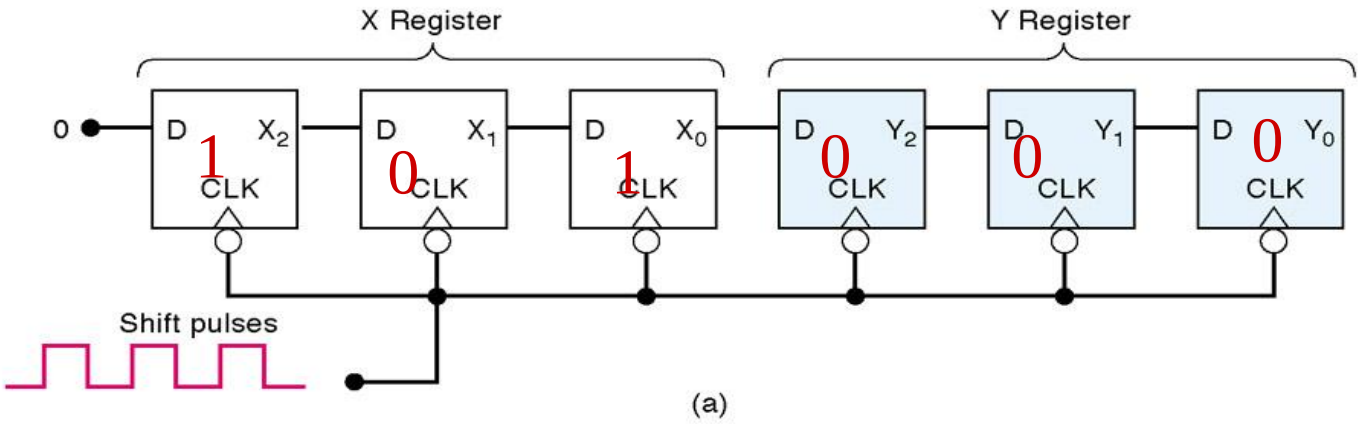


(b) Timing diagram

Fig. 6-4 Serial Transfer from Register *A* to register *B*

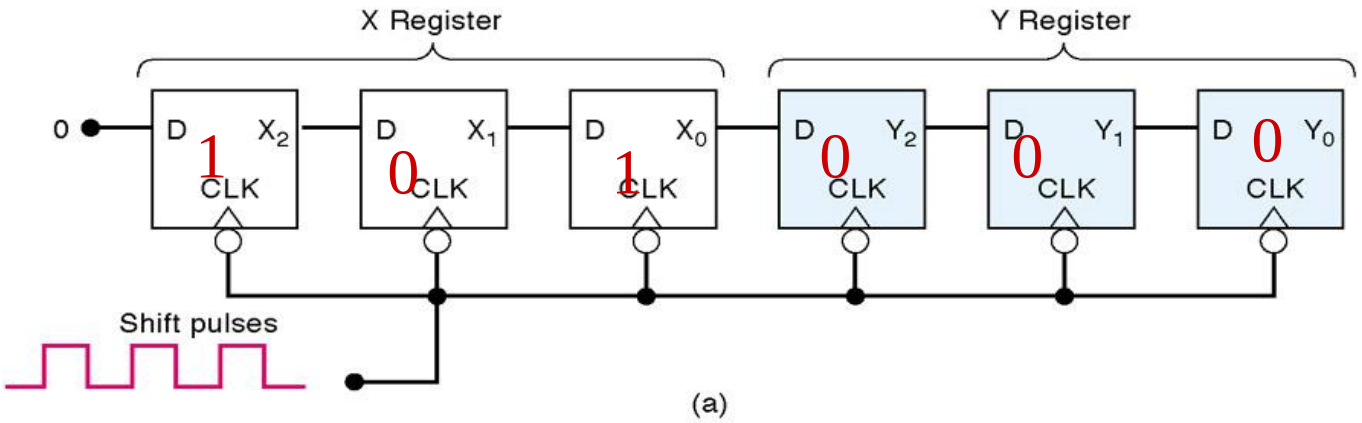
Serial transfer of Data

Transfer from register X to register Y



Serial transfer of Data

Transfer from register X to register Y



X_2	X_1	X_0	Y_2	Y_1	Y_0	
1	0	1	0	0	0	← Before pulses applied
0	1	0	1	0	0	← After first pulse
0	0	1	0	1	0	← After second pulse
0	0	0	1	0	1	← After third pulse

(b)

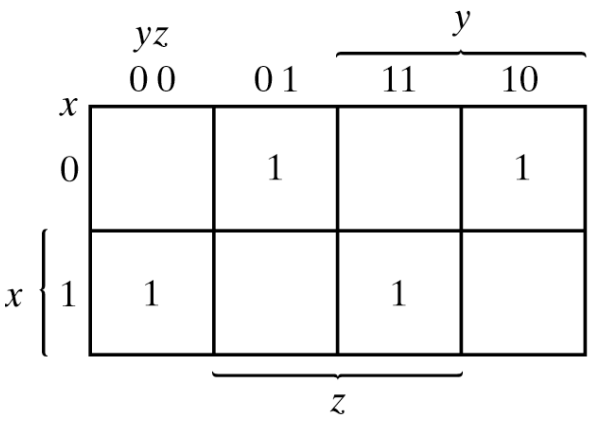
Full Adder (see other implementations in Chap. 2)

Truth Table

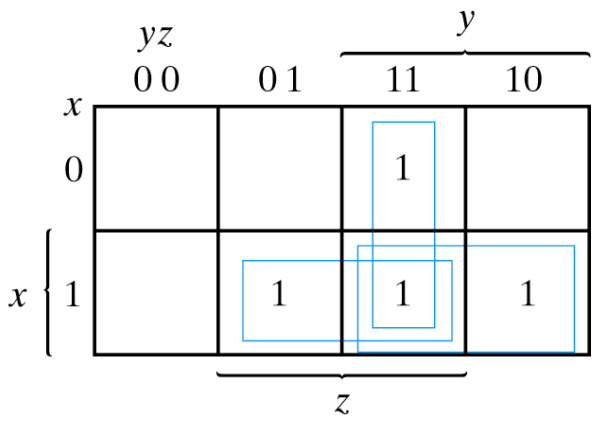
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

→ The Full-adder is combinational circuit

Full Adder (see other implementations in Chap. 2)



$S = x'y'z + x'yz' + xy'z' + xyz$



$C = xy + xz + yz$

Fig. 4-6 Maps for Full Adder

Full Adder (see other implementations in Chap. 2)

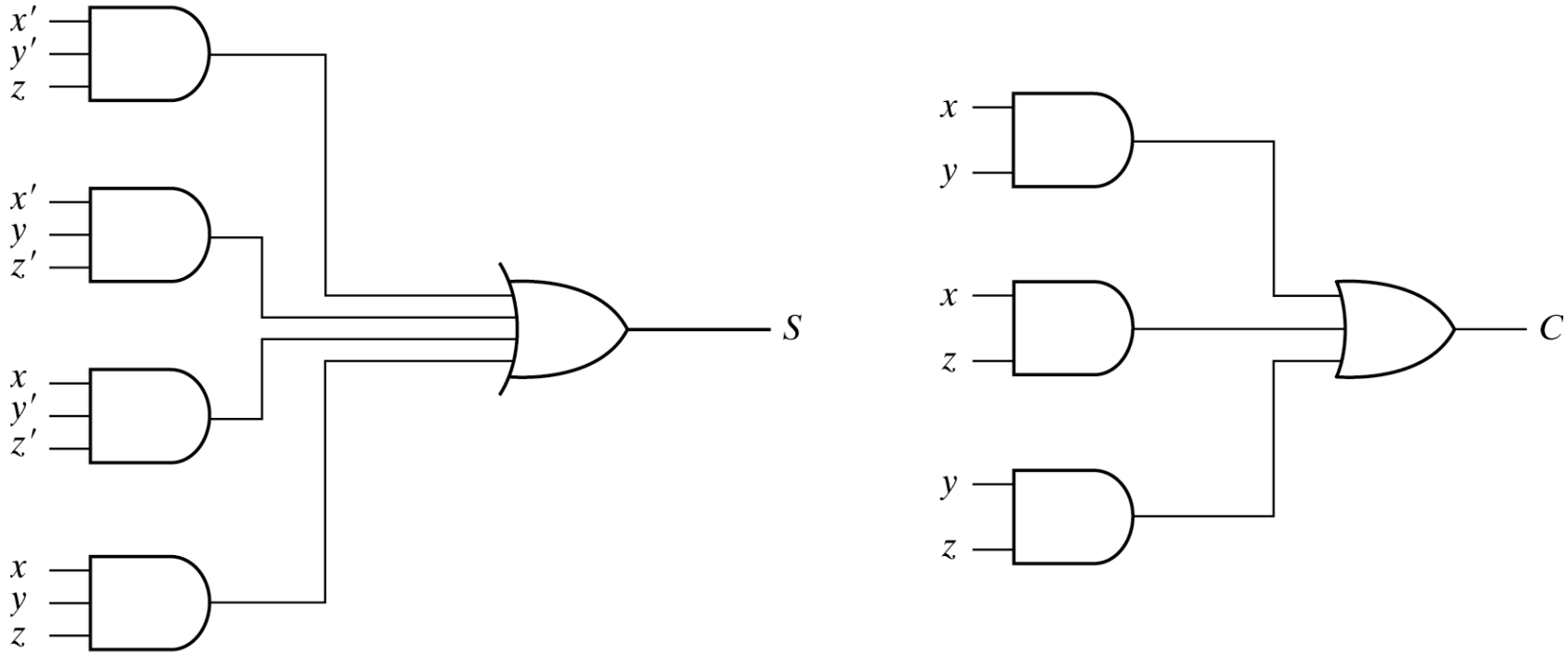


Fig. 4-7 Implementation of Full Adder in Sum of Products

Full Adder (same as in Chap. 2)

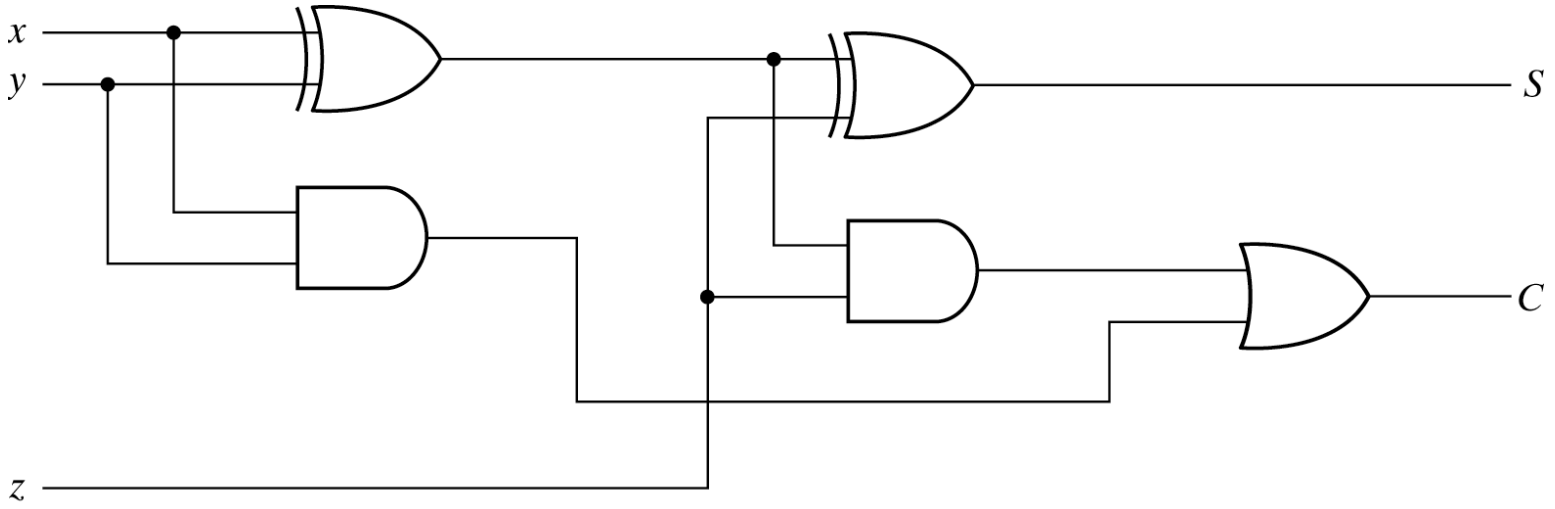


Fig. 4-8 Implementation of Full Adder with Two Half Adders and an OR Gate

Full Adder (see other implementations in Chap. 2)

Truth Table

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

→ The Full-adder is combinational circuit

Serial Addition (D Flip-Flop)

- Slower than parallel
- Low cost
- Share fast hardware on slow data

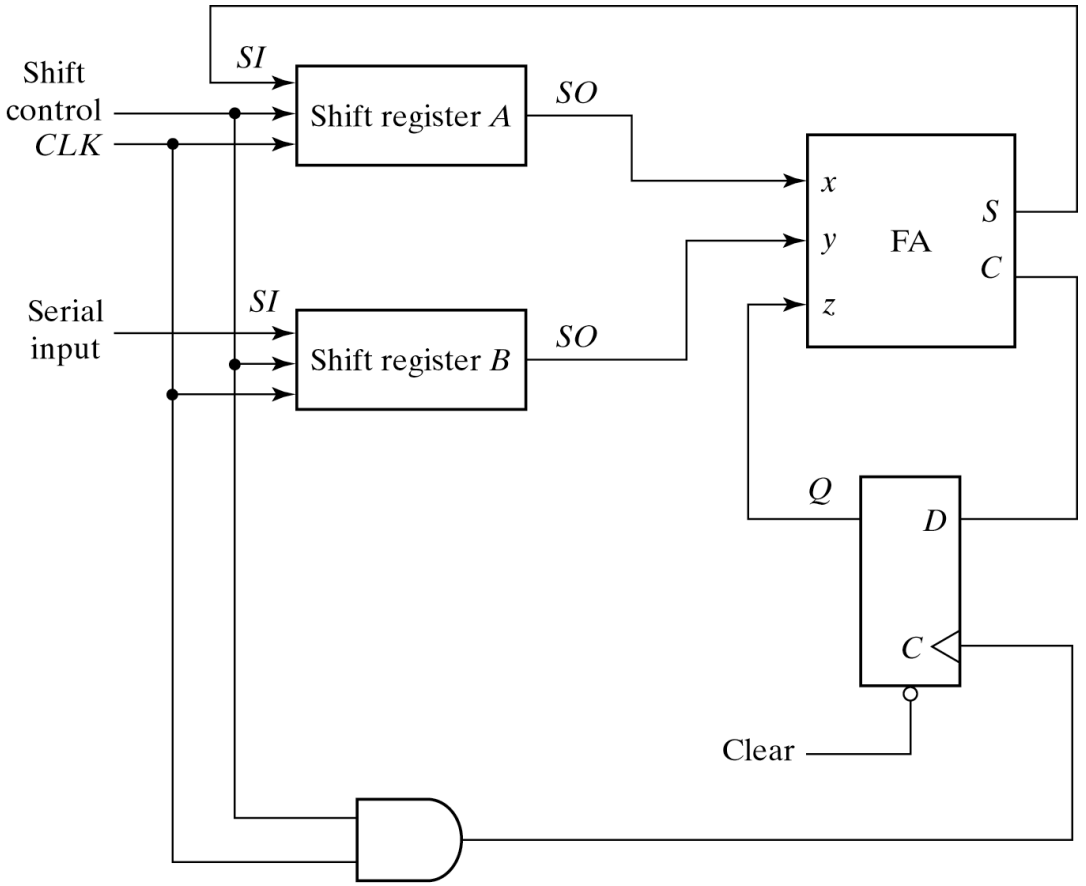


Fig. 6-5 Serial Adder

Transition Tables

J-K Flip-Flop Transition Table

J	K	Q_{n+1}	Function
0	**0**	Q_n	**No change**
0	**1**	**0**	**RESET**
1	**0**	**1**	**SET**
1	**1**	**Q'n**	**complement (also Toggle)**

- we indicate the present and next Q-output. If the flip-flop toggles or holds, we indicate that binary value.
- Also note that we need to determine both the J and K inputs.
- The "X" indicates "Don't Care" states (can be '1' or '0' input).

J-K FLIP-FLOP	
Present Next	Input J K
0 → 0	0 X
0 → 1	1 X
1 → 0	X 1
1 → 1	X 0

Designing a JK Serial Adder

- $J_Q = x \cdot y$
- $K_Q = x' \cdot y' = (x+y)'$
- $S = x \oplus y \oplus Q$

Present State	Inputs		Next State	Output	Flip_Flop Inputs	
Q	x	y	Q	S	J_Q	K_Q
0	0	0	0	0	0	x
0	0	1	0	1	0	x
0	1	0	0	1	0	x
0	1	1	1	0	1	x
1	0	0	0	1	x	1
1	0	1	1	0	x	0
1	1	0	1	0	x	0
1	1	1	1	1	x	0

New Serial Adder

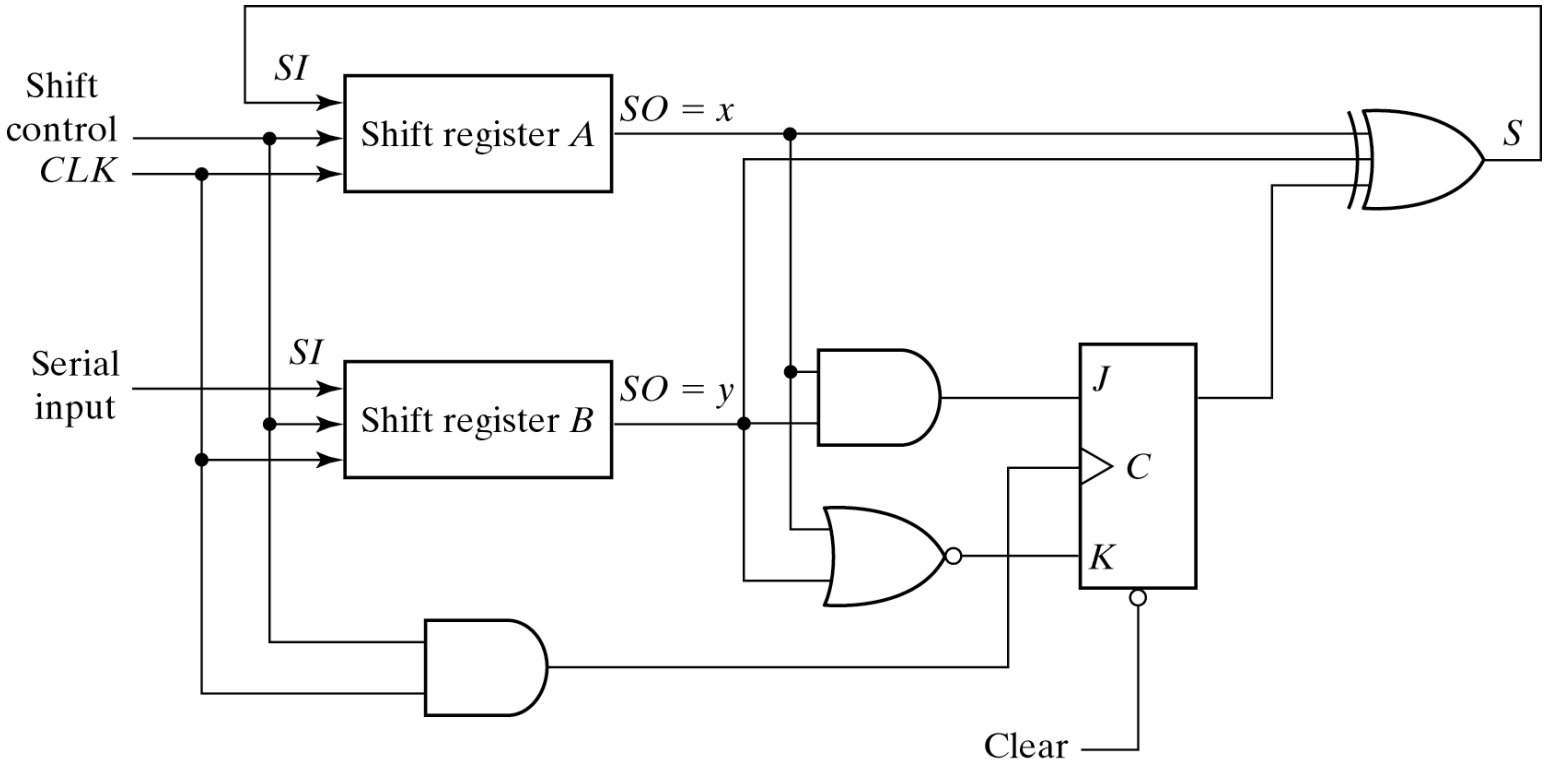


Fig. 6-6 Second form of Serial Adder

Counters

→Counter: A register that goes through a prescribed series of states

→Two main types of counters

1- asynchronous counters: also known as Ripple counter

2- Synchronous counters

→*Ripple counters*

– Flip flop output serves as a clock for triggering connected flip flops

→ *Synchronous counters*

– All flip flops are triggered by a clock signal at the same time

→ Synchronous counters are more widely used

Asynchronous Counters

- Each Flip flop output controls the CLK input of the next Flip flop.
- Flip flop do not change states in exact synchronism with the applied clock pulses.
- *Ripple counter* due to the way the flip flops respond one after another in a kind of rippling effect.

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

Binary Ripple Counter

- Reset signal sets all outputs to 0
- Count signal toggles output of low-order flip flop
- Low-order flip flop provides trigger for adjacent flip flop
- Not all flops change value simultaneously
  - Lower-order flops change first

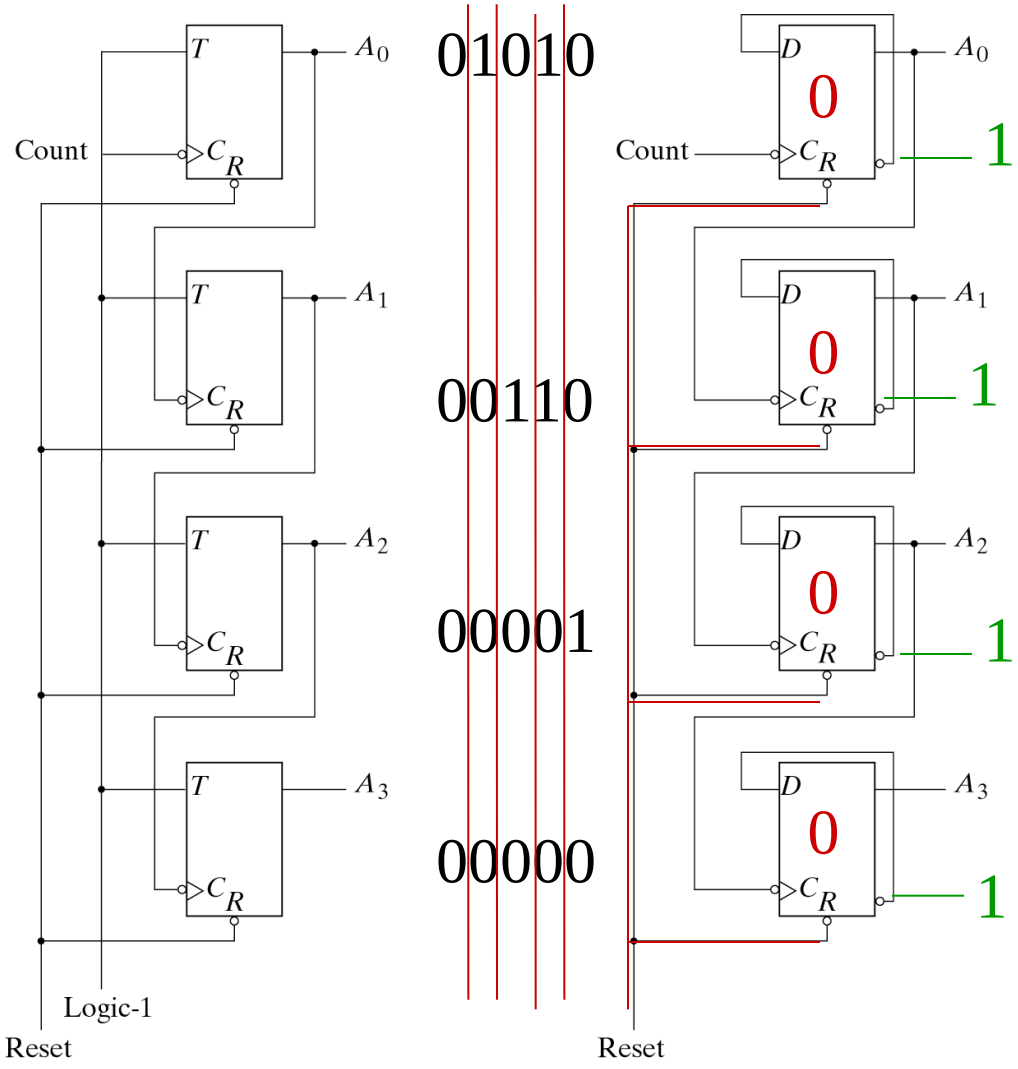
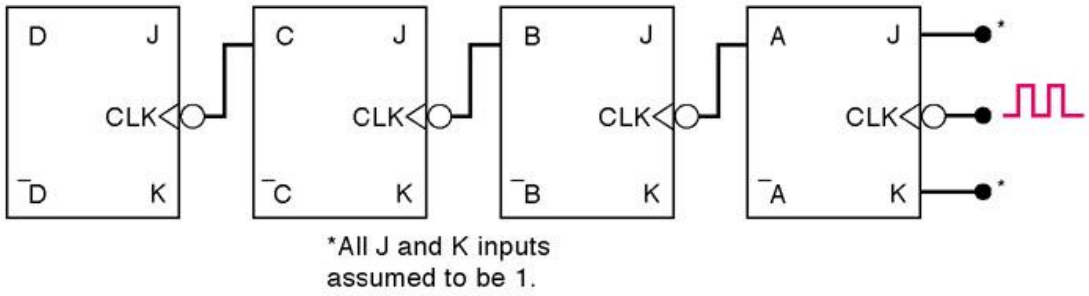
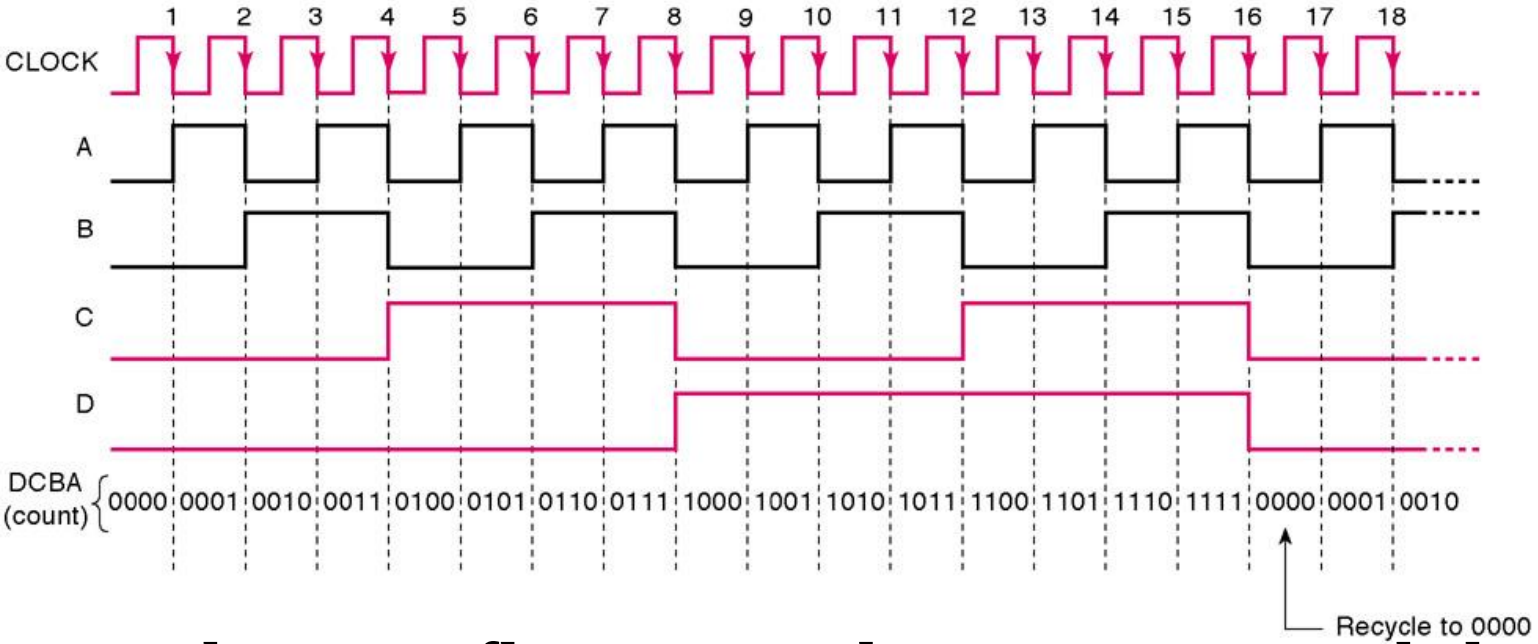


Fig. 6-8 4-Bit Binary Ripple Counter

Another Asynchronous Ripple Counter





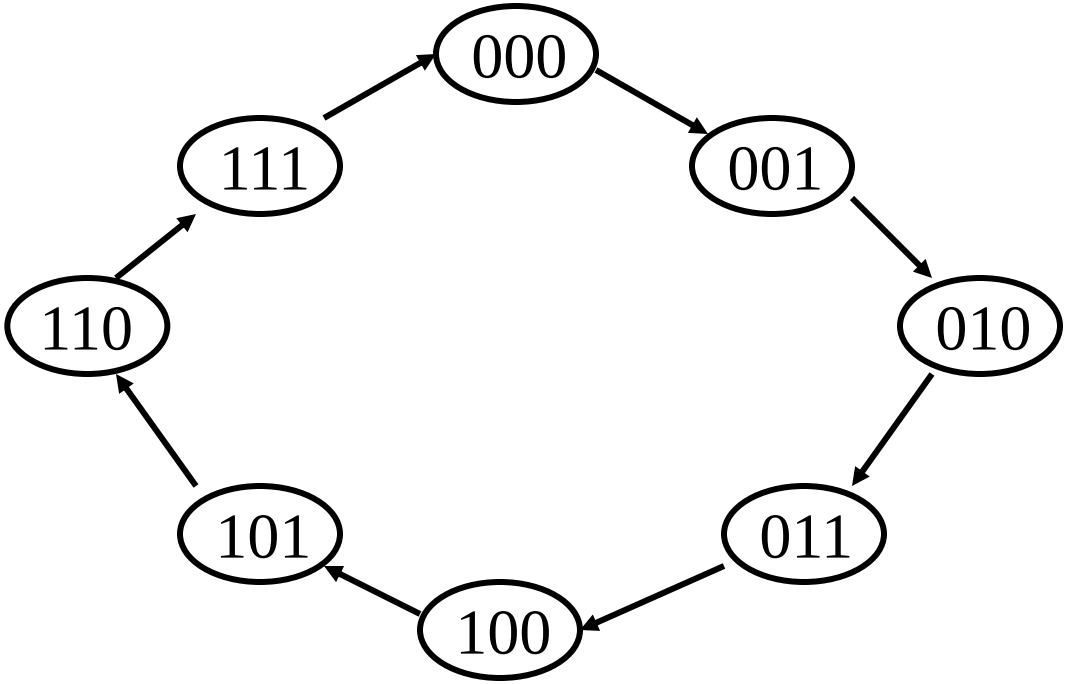
- Similar to T flop example on previous slide

Binary Counters

Counters produce a fixed sequence of binary digits, or states.

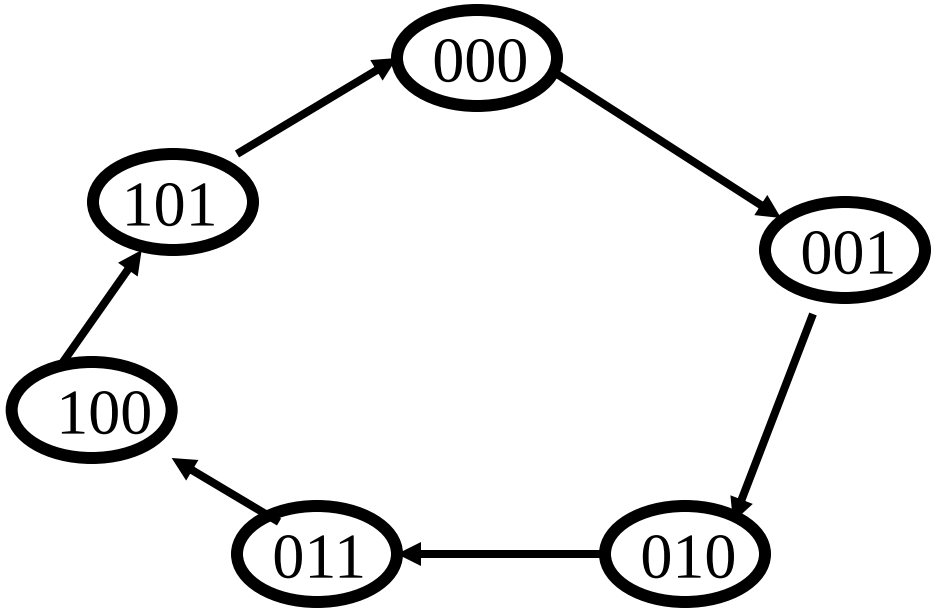
Counters are normally designed to recycle or restart the sequence.

Example:



Binary Counters: Modulus

- The number of output states is called the MODULUS, or MOD.
  - For example, a mod-6 counter has 6 unique output states → what's the max MOD ?



Counter Sequences

→*Full Sequence Count*

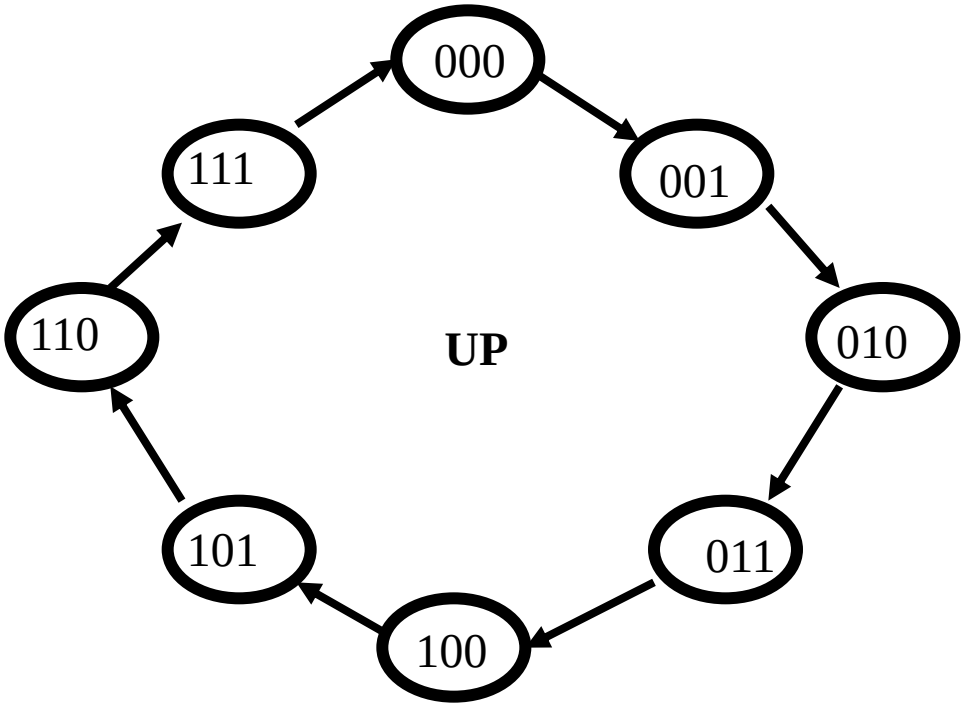
refers to a natural count that includes all possible binary numbers. It's modulus is the same as its maximum modulus.

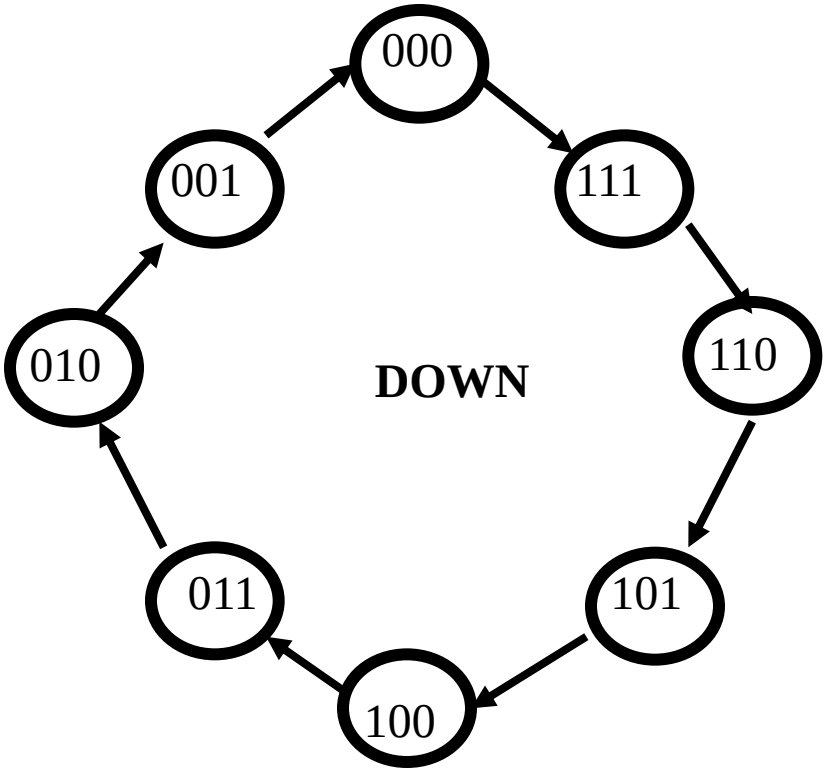
→A Truncated Sequence Count

when the modulus is less than its maximum, or where less than all possible binary numbers are used.

Binary Counters: Up/Down

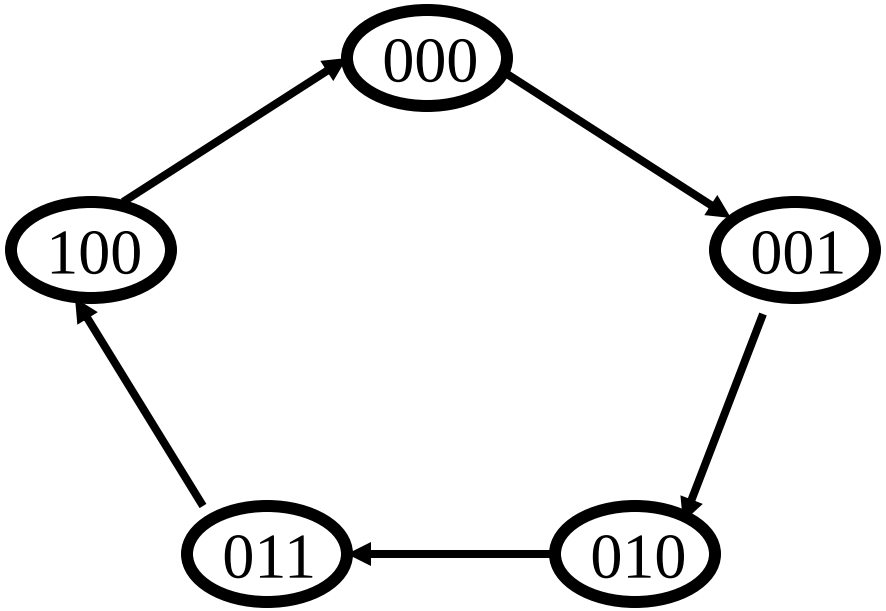
A sequential count refers to a natural numerical count.The sequence can be Up or Down:





Counter State Diagram

- Counter states are sequential, where an output state will follow another in a sequence.
- State Diagrams are used to present the sequence of these states.



State Diagram

Counter State Table

- A State Table is another means of presenting state sequences.
- As with the state diagram, the state table will help determine the next output based on the present output state.

State Table

Current			Next		
Q_c	Q_b	Q_a	Q_c	Q_b	Q_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	0	0	0

Design Procedure

1- Obtain from a (given) state table,

(i) the state diagram and

(ii) transition table

2- Provide the required input to each of the flip-flops by utilizing the outputs of any of the other flip-flops.

3- Use external gates, to detect specific, usable sequences of flip-flop outputs to create the necessary next input levels. (How to create inputs from outputs)

4- Use K-Maps to record and simplify the available outputs to create the inputs.

5- Implement the Circuit (i.e. Diagram)

Transition Tables

- By using K-Maps, we no longer need to utilize the toggling mode of a flip-flop (as in asynchronous). This allows us to use any edge triggered flip-flop to create the non-sequential counter.

- We need to know how the different flip-flops respond to various input states, based on their **current state.**

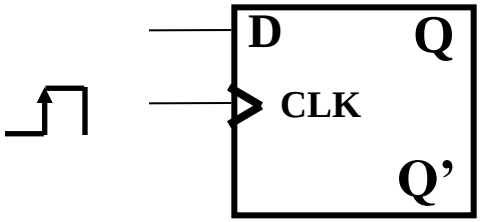
Building a Transition Table

- Building a Transition Table is done using the Function Tables of the Flip-Flops.

- Where the Function Table indicates "Q", "Q'" or "No Change", we indicate the binary value.

D Flip flop

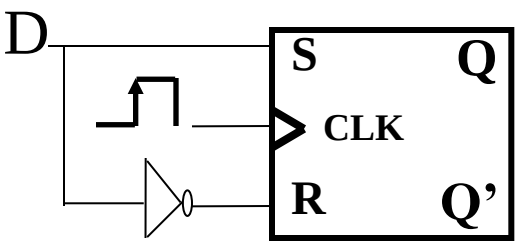
- Output changes only on the clock transition



Symbol

D	Q_{n+1}
0	0
1	**1**

D	CLK	Q	Q'
0	↑	0	1
1	↑	1	0
X	0	Q_0	Q_0'



D Flip flop can be implemented using SR

S = D

R =D'

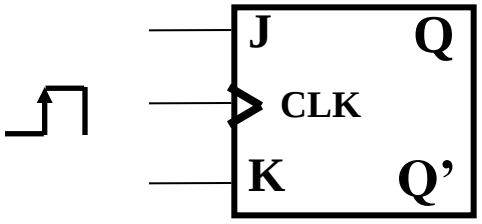
Transition Table: Example with D Flip Flop

D-FLIP-FLOP		
Present	Next	Input
0	0	0
0	1	1
1	0	0
1	1	1

- we indicate the present and the next states of the outputs.
- The input states indicates the required D input to produce the next state.

JK Flip flop

Symbol



J	K	Q_{n+1}	Function
0	0	Q_n	No change
0	1	0	RESET
1	0	1	SET
1	1	Q'_n	complement (also Toggle)

Same as SR except for

K=J= 1 the JK flip flop will

Output the opposite state of Q_n.

- Q_n = state *before* positive edge
- Q_{n+1} = state *after* positive edge

If $Q_n = 1$ then $Q_{n+1} = 0$

If $Q_n = 0$ then $Q_{n+1} = 1$

Transition Tables

J-K Flip-Flop Transition Table

- we indicate the present and next Q-output. If the flip-flop toggles or holds, we indicate that binary value.
- Also note that we need to determine both the J and K inputs.
- The "X" indicates "Don't Care" states (can be '1' or '0' input).

J-K FLIP-FLOP	
Present Next	Input J K
0 → 0	0 X
0 → 1	1 X
1 → 0	X 1
1 → 1	X 0

J-K Flip Flop Transition Table

J-K FLIP-FLOP			
Present	Next	Input J	Input K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

States

Hold or Reset

Toggle or Set

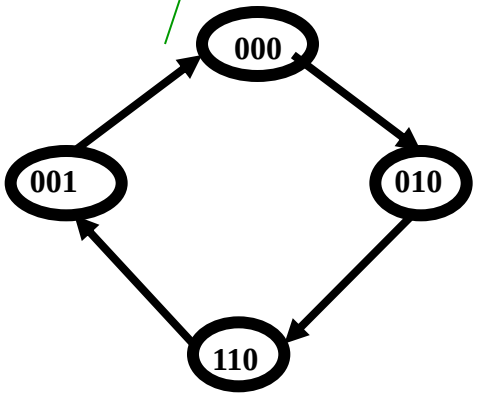
Toggle or Reset

Hold or Set

→ **each output of the J-K Flip-Flop is the result of 2 possible states.**

Building a K-Map with D Flip Flop

- To build a K-Map, we first need:
  1- State Diagram of the output sequence
  2- State Table of the output sequence
  3- Transition Table for the Flip-Flop we intend to use (here D ff)
    - Present and Next outputs
    - Necessary inputs to create Next outputs

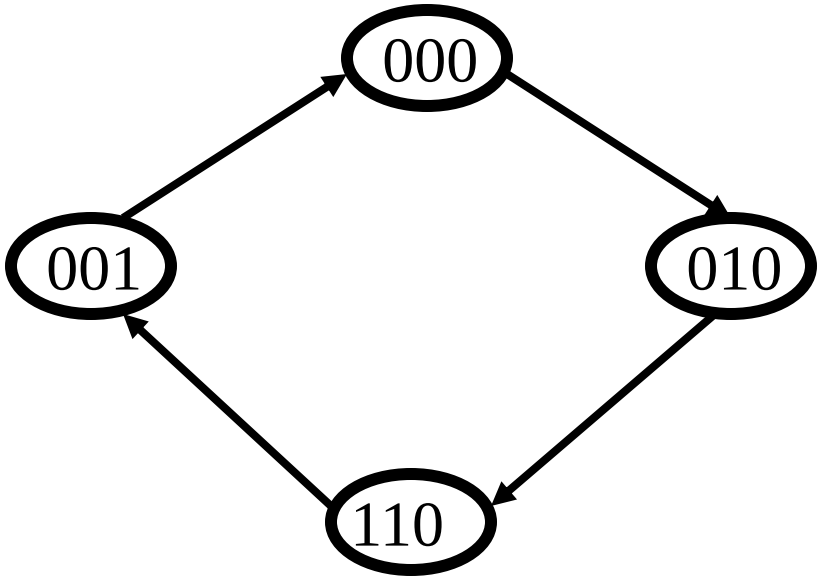


Present $Q_C Q_B Q_A$	Next $Q_C Q_B Q_A$
0 0 0	0 1 0
0 1 0	1 1 0
1 1 0	0 0 1
0 0 1	0 0 0

D-FLIP-FLOP		
Present	Next	Input
0	0	0
0	1	1
1	0	0
1	1	1

Building a K-Map for D Flip Flop: Step 1

- Draw the State Table and State Diagram of the output sequence.



State Diagram

Present $Q_C Q_B Q_A$	Next $Q_C Q_B Q_A$
0 0 0	0 1 0
0 1 0	1 1 0
1 1 0	0 0 1
0 0 1	0 0 0

State Table

Building the K-Map : Step 2

- Determine the type of Flip-Flop and its Transition Table. For our example, we use the D Flip-Flop:

D-FLIP-FLOP	
Present → Next	Input
0 → 0	0
0 → 1	1
1 → 0	0
1 → 1	1

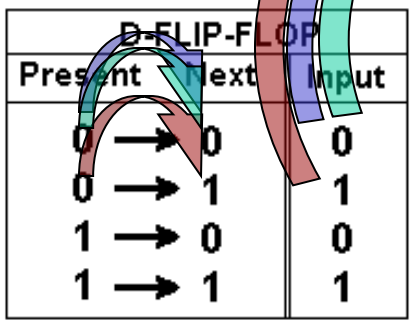
Transition Table

Building the K-Map: Step 3

- **Build the State Table with the inputs indicated:**

D-FLIP-FLOP		
Present	Next	Input
0	0	0
0	1	1
1	0	0
1	1	1

Present	Next	Input
Q_C Q_B Q_A	Q_C Q_B Q_A	D_C D_B D_A
0 0 0	0 1 0	0 1 0
0 1 0	1 1 0	1 1 0
1 1 0	0 0 1	0 0 1
0 0 1	0 0 0	0 0 0



Present			Next			Input		
Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	D_C	D_B	D_A
0	0	0	0	1	0	0	1	0
0	1	0	1	1	0	1	1	0
1	1	0	0	0	1	0	0	1
0	0	1	0	0	0	0	0	0

Animated

Building the K-Map: Step 4

Build a K-Map for each output:

Present			Next			Input		
Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	D_C	D_B	D_A
0	0	0	0	1	0	0	1	0
0	1	0	1	1	0	1	1	0
1	1	0	0	0	1	0	0	1
0	0	1	0	0	0	0	0	0

$Q_C Q_B$ \ Q_A	0	1
00	0	0
01	1	X
11	0	X
10	X	X

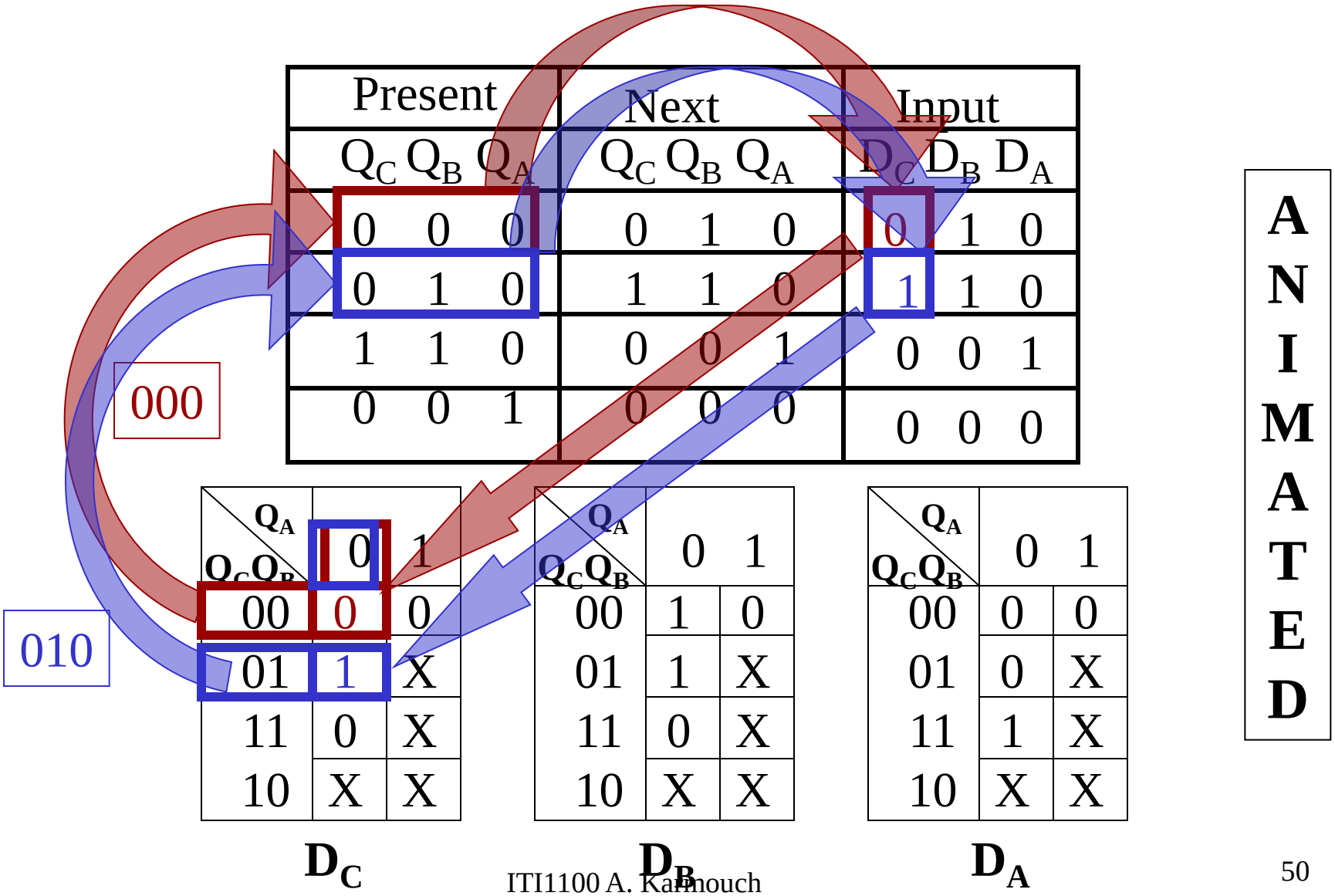
D_C

$Q_C Q_B$ \ Q_A	0	1
00	1	0
01	1	X
11	0	X
10	X	X

D_B

$Q_C Q_B$ \ Q_A	0	1
00	0	0
01	0	X
11	1	X
10	X	X

D_A


Present
Next
Input
Q_C Q_B Q_A
Q_C Q_B Q_A
D_C D_B D_A
0 0 0
0 1 0
0 1 0
1 1 0
1 1 0
1 1 0
0 0 1
0 0 1
0 0 1
0 0 1
0 0 0
0 0 0
000
010
Q_A
$Q_C Q_B$
0 1
00 0 0
01 1 X
11 0 X
10 X X
D_C
Q_A
$Q_C Q_B$
0 1
00 1 0
01 1 X
11 0 X
10 X X
D_B
Q_A
$Q_C Q_B$
0 1
00 0 0
01 0 X
11 1 X
10 X X
D_A

ANIMATED

Building the K-Map: Step 5

determine the simplified SOP for the inputs.

$Q_C Q_B$ \ Q_A	0	1
00	0	0
01	1	X
11	0	X
10	X	X

D_C

$D_C = Q'_C Q_B$

$Q_C Q_B$ \ Q_A	0	1
00	1	0
01	1	X
11	0	X
10	X	X

D_B

$D_B = Q'_C Q'_A$

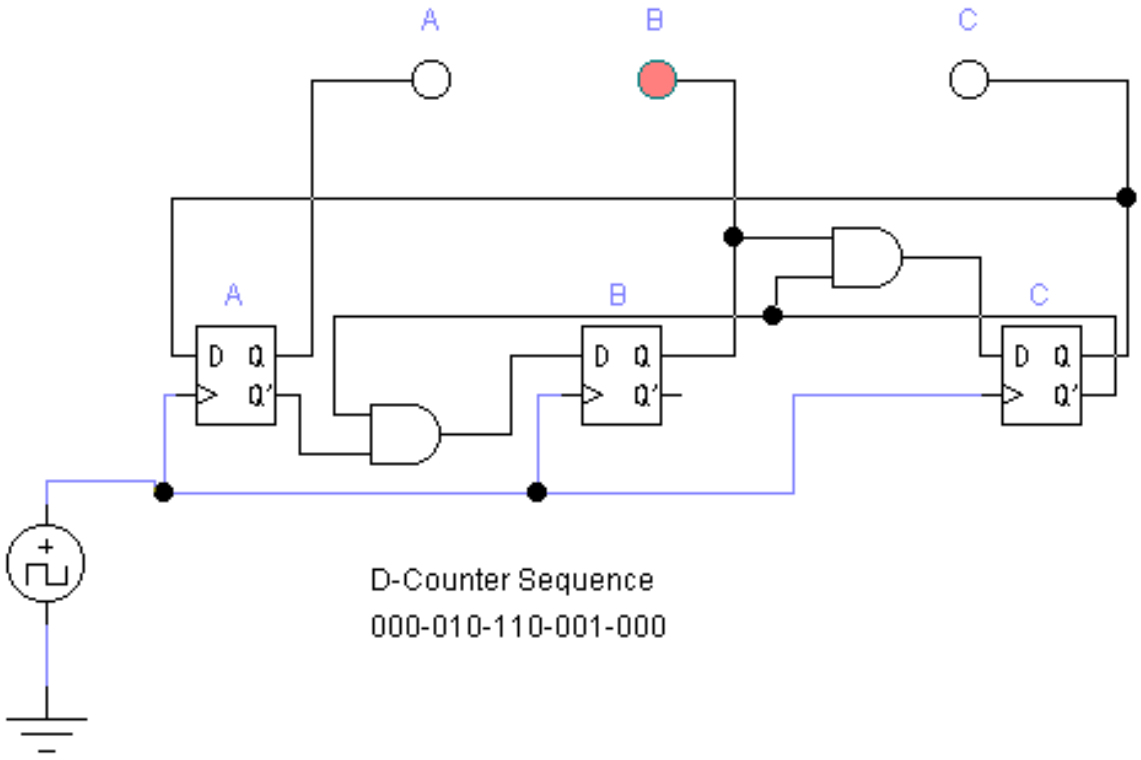
$Q_C Q_B$ \ Q_A	0	1
00	0	0
01	0	X
11	1	X
10	X	X

D_A

$D_A = Q_C$

Implementation Diagram

- Draw the Circuit Diagram
- Verify its operations.



Binary Counter Examples (will NOT be on the website)

- Examples of binary counter with JK flip Flop

1- Without missing States

2- With missing States

Binary counter with JK flip flop

Transition Table for JK FF

(a) JK flip-flop truth table

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

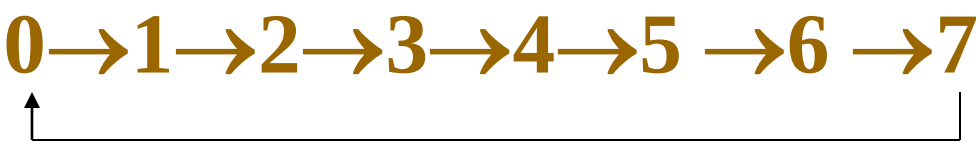
(b) Transition table for JK flip-flops

Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Binary counter with JK flip flop

- Build state diagram
- Build the state table that consists of
  - Current state output
  - Next state output
  - JK inputs for each flip-flop
- Use K-maps to simplify expressions for JK inputs
- Build the circuit for the counter

Binary counter with JK flip flop



- 3-bit binary counter
- 3 JK flip-flops are needed
- Current state and next state outputs are 3 bits each
- 3 pairs of JK inputs

Binary counter with JK flip flop

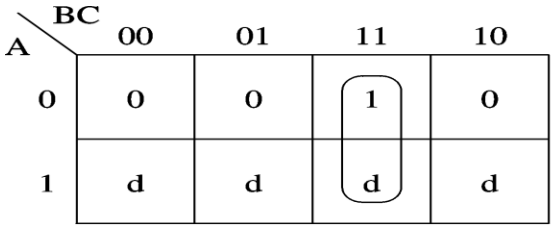
State table for the binary counter example

Present state			Next state			JK flip-flop inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

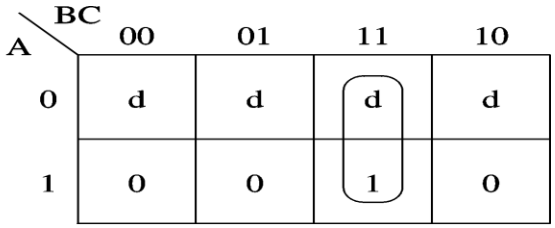
Binary counter with JK flip flop

Use K-maps to simplify expressions for JK inputs

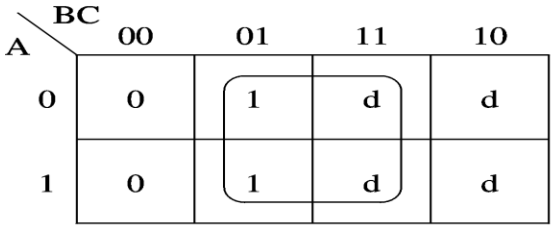
X= d



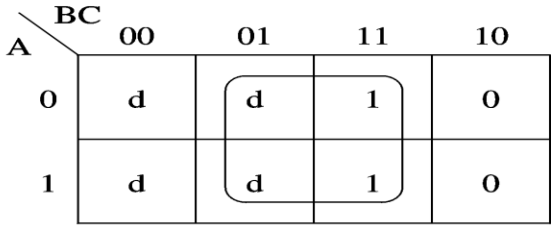
$J_A = B\,C$



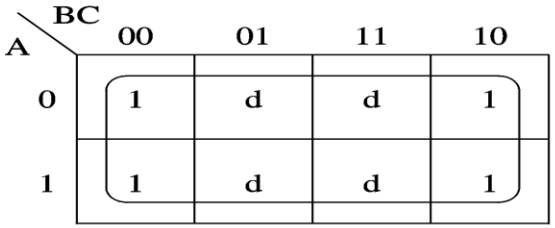
$K_A = B\,C$



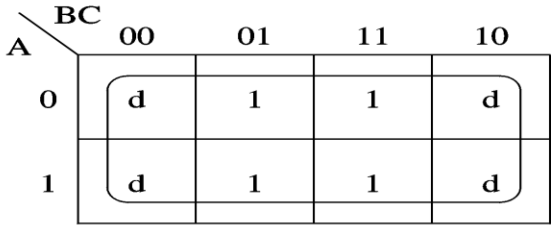
$J_B = C$



$K_B = C$



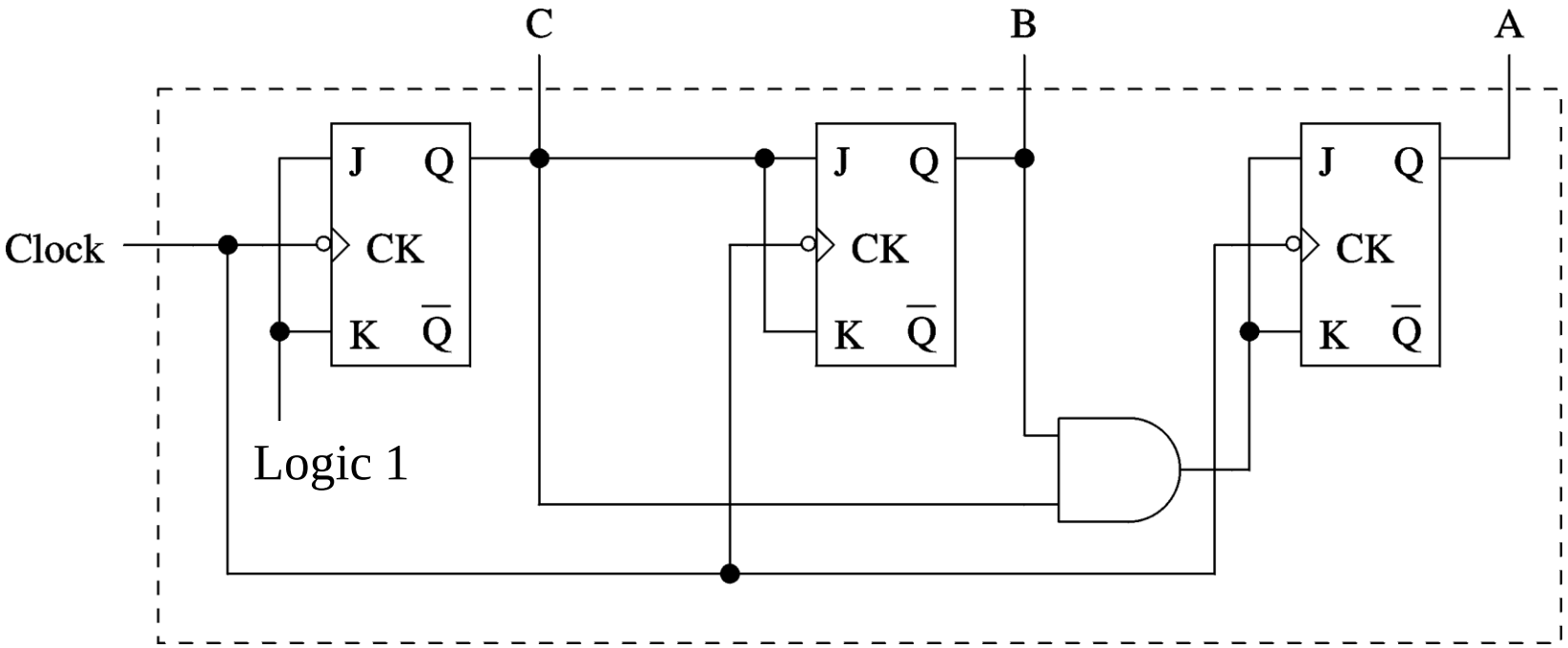
$J_C = 1$



$K_C = 1$

Binary counter with JK flip flop

- Final circuit for the binary counter

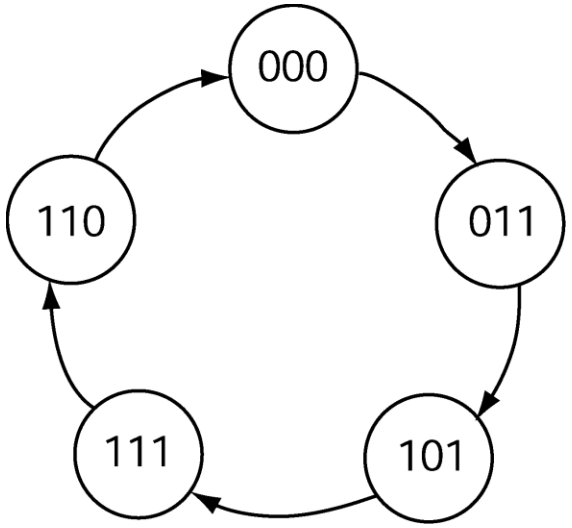


Binary counter with JK flip flop (missing states)

→ Example with missing states

0→3→5→7→6→0

- Same design process as before
- One significant change
  - Missing states
    - » 1, 2, and 4
    - » Use don't cares for these states



Binary counter with JK flip flop (missing states)

Sate table for the counter example

Present state			Next state			JK flip-flop inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	1	1	0	X	1	X	1	X
0	0	1	X	X	X	X	X	X	X	X	X
0	1	0	X	X	X	X	X	d	X	X	X
0	1	1	1	0	1	1	X	X	1	X	0
1	0	0	X	X	X	X	X	X	X	X	X
1	0	1	1	1	1	X	0	1	X	X	0
1	1	0	0	0	0	X	1	X	1	0	X
1	1	1	1	1	0	X	0	X	0	X	1

Binary counter with JK flip flop (missing states)

K-maps to simplify JK input expressions

X=d

A \ BC	00	01	11	10
0	0	d	1	d
1	d	d	d	d

$J_A = B$

A \ BC	00	01	11	10
0	d	d	d	d
1	d	0	0	1

$K_A = \overline{C}$

A \ BC	00	01	11	10
0	1	d	d	d
1	d	1	d	d

$J_B = 1$

A \ BC	00	01	11	10
0	d	d	1	d
1	d	d	0	1

$K_B = \overline{A} + \overline{C}$

A \ BC	00	01	11	10
0	1	d	d	d
1	d	d	d	0

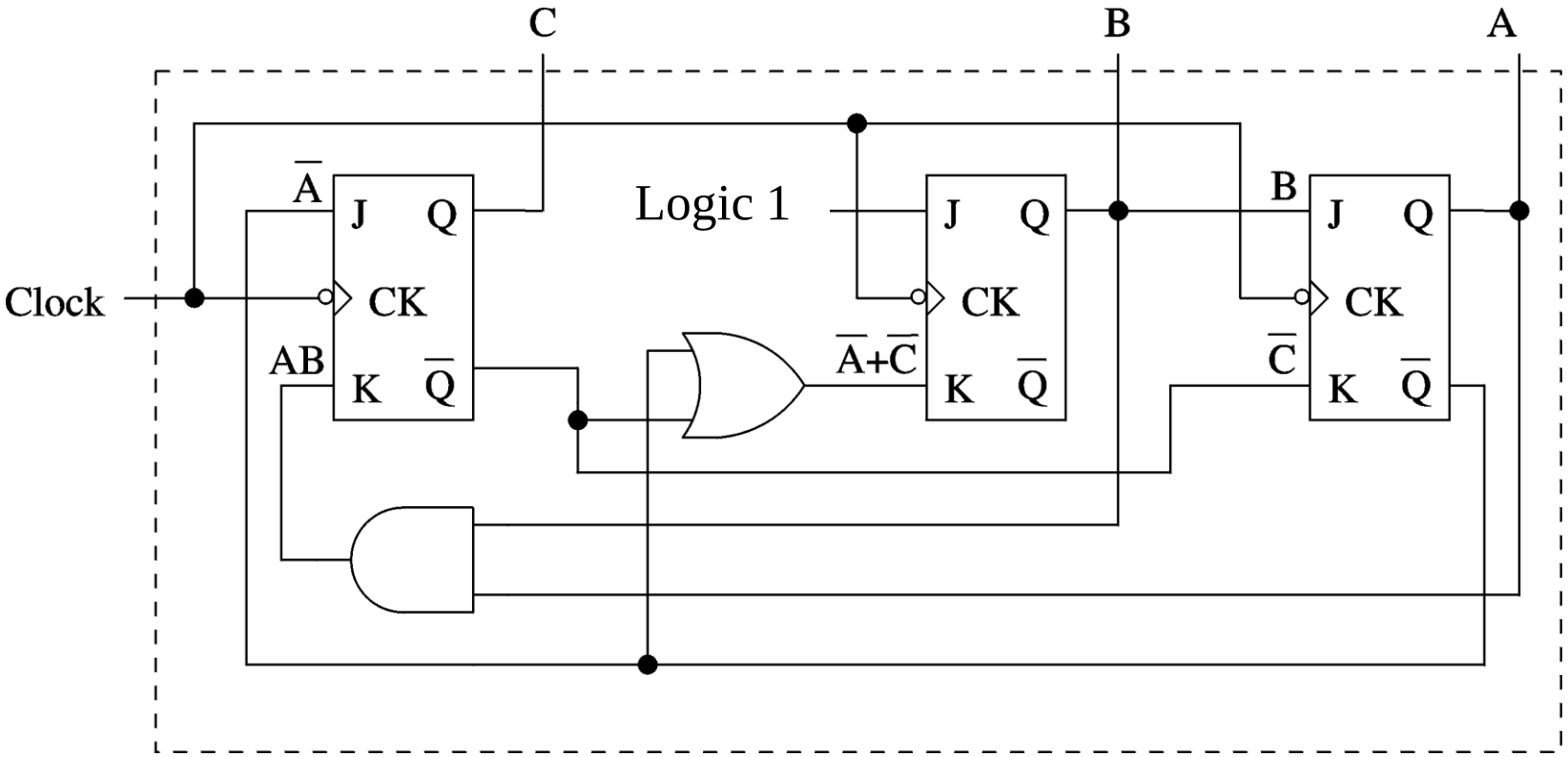
$J_C = \overline{A}$

A \ BC	00	01	11	10
0	d	d	0	d
1	d	0	1	d

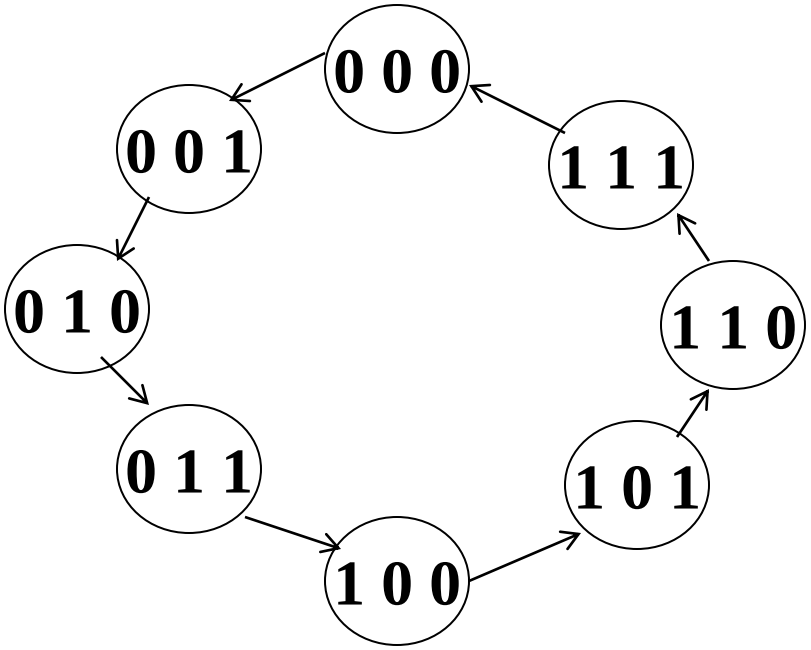
$K_C = A\,B$

Binary counter with JK flip flop (missing states)

Final circuit



Example: - Design a 3 - bit binary counter using T flip - flops

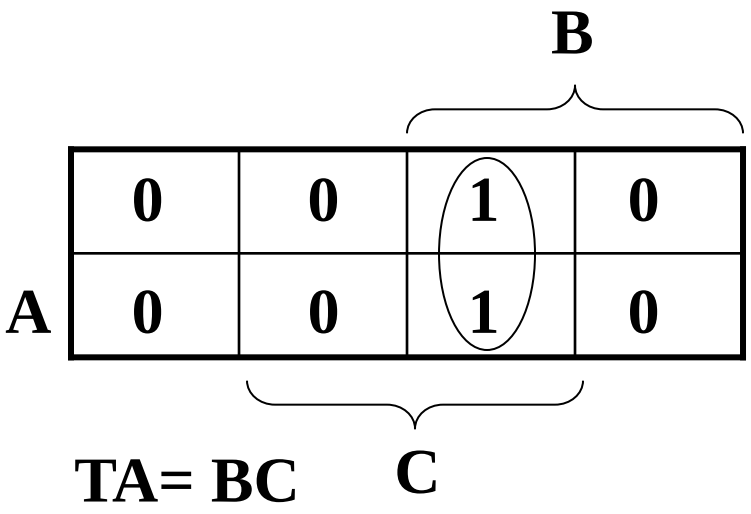


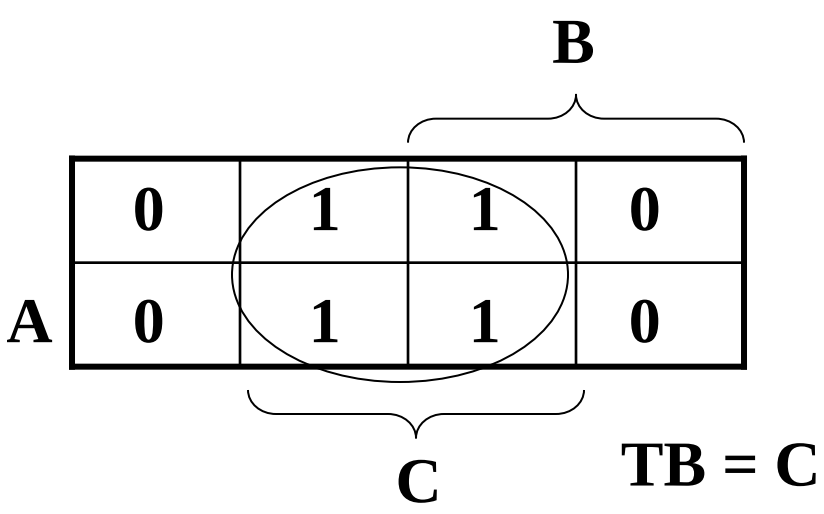
Example: - Design a 3 - bit binary counter using T flip - flops.

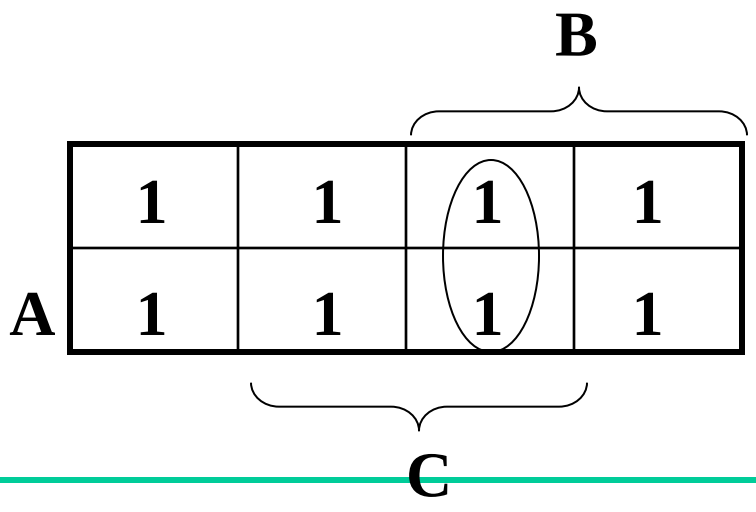
Present State A B C	Next State A B C	Flip - Flop Inputs T A	T B	T C
0 0 0	0 0 1	0	0	1
0 0 1	0 1 0	0	1	1
0 1 0	0 1 1	0	0	1
0 1 1	1 0 0	1	1	1
1 0 0	1 0 1	0	0	1
1 0 1	1 1 0	0	1	1
1 1 0	1 1 1	0	0	1
1 1 1	0 0 0	1	1	1

Example: - Design a 3 - bit binary counter using T flip - flops.

- Kmap

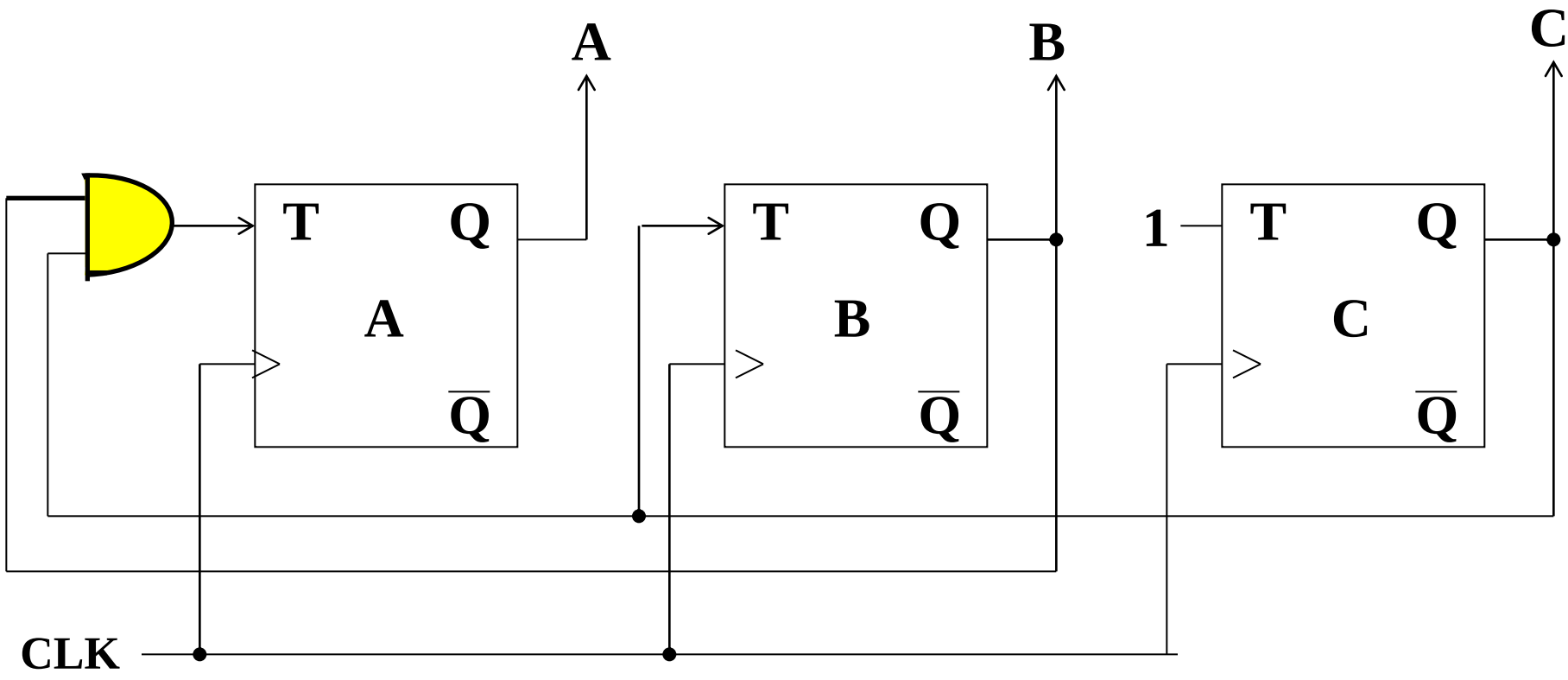






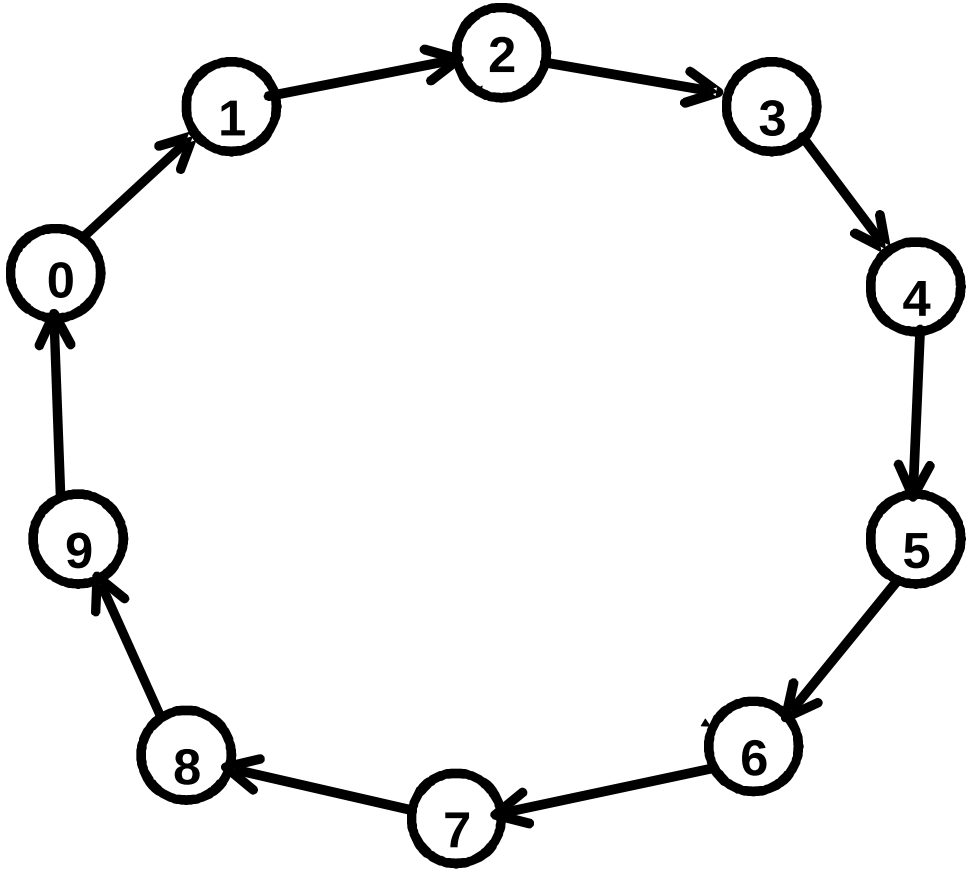
TC = 1

Example: - Design a 3 - bit binary counter using T flip - flops.



Design: Synchronous BCD

- Design a BCD counter using T Flip Flop



Design: Synchronous BCD

- State table

Current State Q_8 Q_4 Q_2 Q_1	Next State Q_8 Q_4 Q_2 Q_1	T-FF inputs T_8 T_4 T_2 T_1
0 0 0 0	0 0 0 1	
0 0 0 1	0 0 1 0	
0 0 1 0	0 0 1 1	
0 0 1 1	0 1 0 0	
0 1 0 0	0 1 0 1	
0 1 0 1	0 1 1 0	
0 1 1 0	0 1 1 1	
0 1 1 1	1 0 0 0	
1 0 0 0	1 0 0 1	
1 0 0 1	0 0 0 0	

Design: Synchronous BCD

- **We can use the sequential logic model to design a synchronous BCD counter with T flip-flops. Below is the State Table.**

- **Don't care states have been left out here.**

Current State Q_8 Q_4 Q_2 Q_1	Next State Q_8 Q_4 Q_2 Q_1	T-FF inputs T_8 T_4 T_2 T_1
0 0 0 0	0 0 0 1	0 0 0 1
0 0 0 1	0 0 1 0	0 0 1 1
0 0 1 0	0 0 1 1	0 0 0 1
0 0 1 1	0 1 0 0	0 1 1 1
0 1 0 0	0 1 0 1	0 0 0 1
0 1 0 1	0 1 1 0	0 0 1 1
0 1 1 0	0 1 1 1	0 0 0 1
0 1 1 1	1 0 0 0	1 1 1 1
1 0 0 0	1 0 0 1	0 0 0 1
1 0 0 1	0 0 0 0	1 0 0 1

Synchronous BCD (Continued)

- **Use K-Maps to minimize the next state function:**

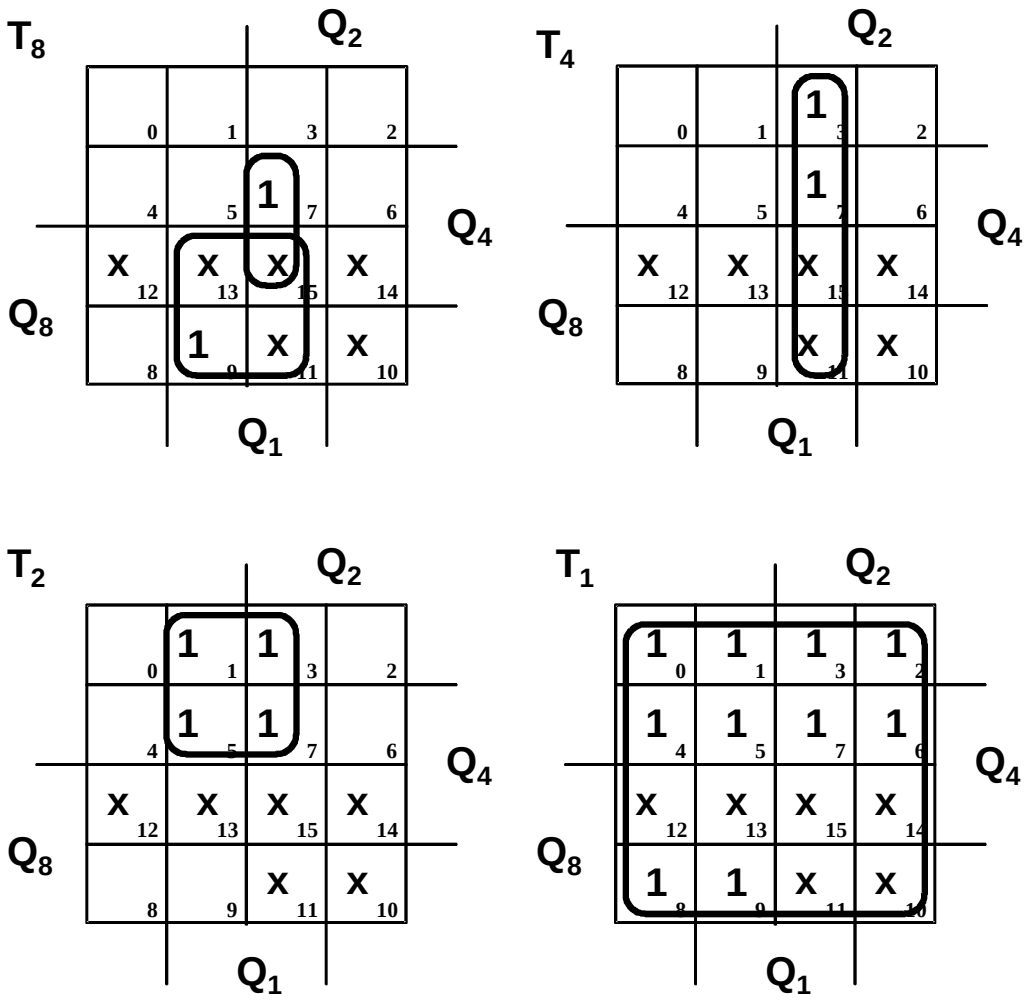
$T_8 = Q_8 \bullet Q_1 + Q_4 \bullet Q_2 \bullet Q_1$

$T_4 = Q_2 \bullet Q_1$

$T_2 = Q_8' \bullet Q_1$

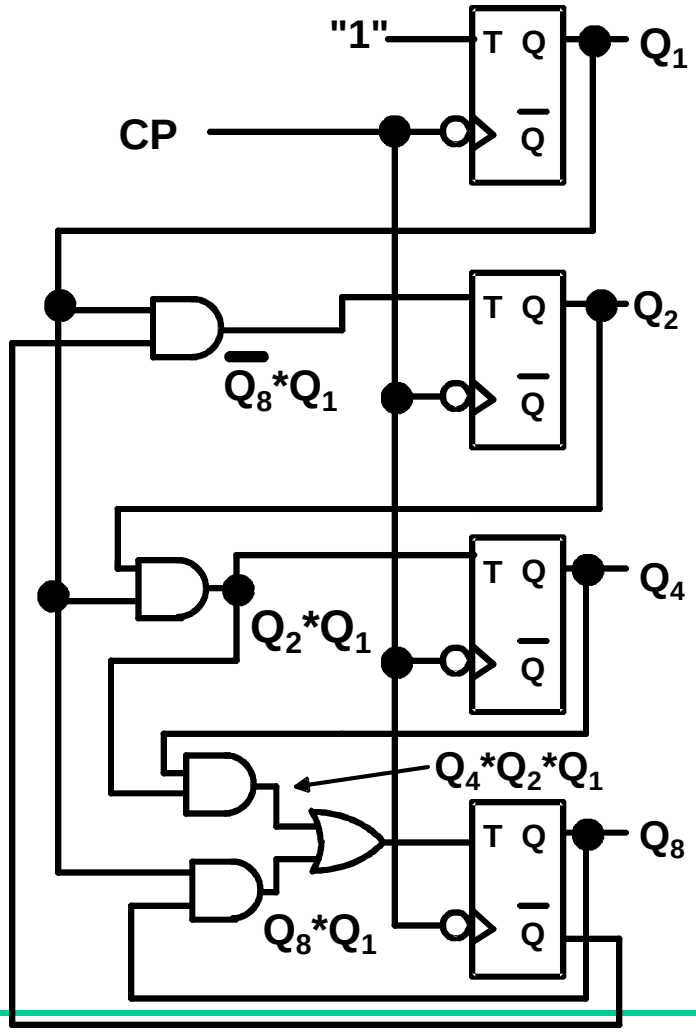
$T_1 = \text{"1"}$

Note: <u>Don't Cares</u> are included here.



Synchronous BCD (Continued)

- **The minimized circuit:**



Designing a Synchronous BCD Counter

Present State				Next State				Output	Next State			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	1	0	0	1	1	0	0	0	0	1	1
0	1	0	1	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1.	1	1
1	0	0	0	1	0.	0	1	0	0	0	0	1
1	0.	0	1	0	0	0	0	1	1	0	0	1

$TQ_1 = 1$, $TQ_2 = Q'_8Q_1$, $TQ_4 = Q_2Q_1$, $TQ_8 = Q_8Q_1 + Q_4Q_2Q_1$ $y = Q_8Q_1$

,

Asynchrone, BCD Counter

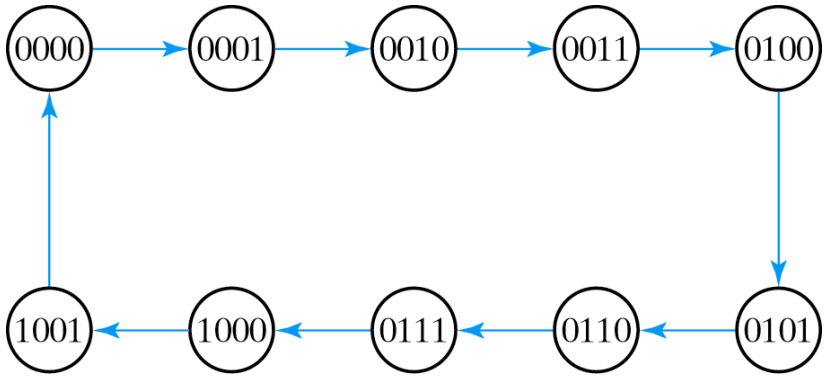
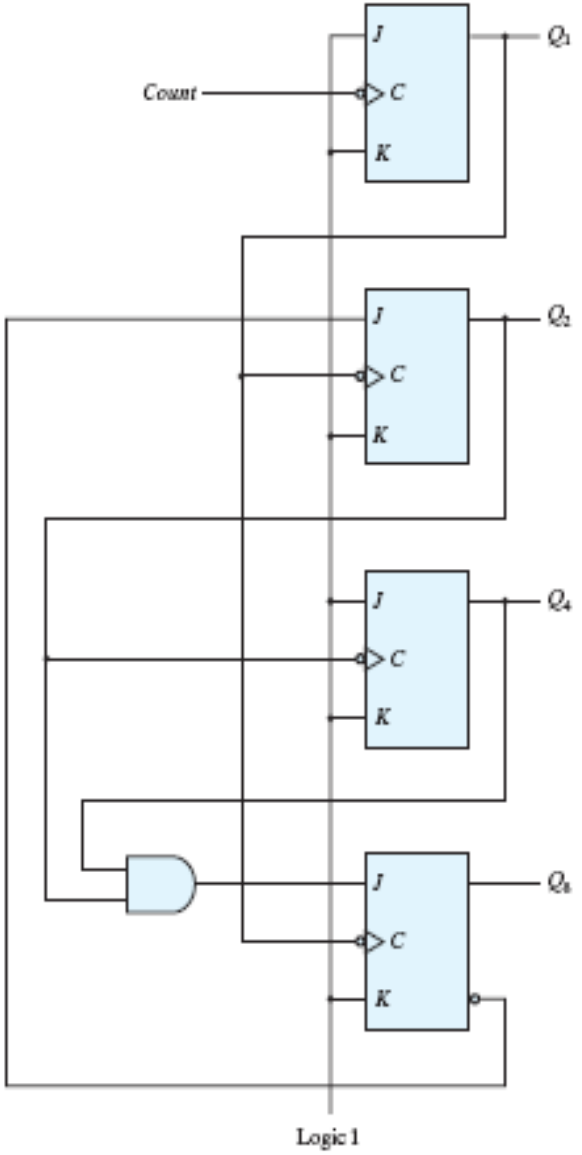


Fig. 6-9 State Diagram of a Decimal BCD-Counter



Decimal (BCD) Counter

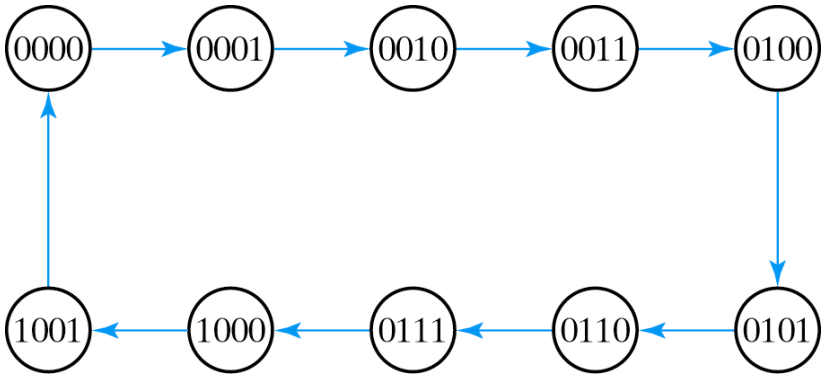


Fig. 6-9 State Diagram of a Decimal BCD-Counter

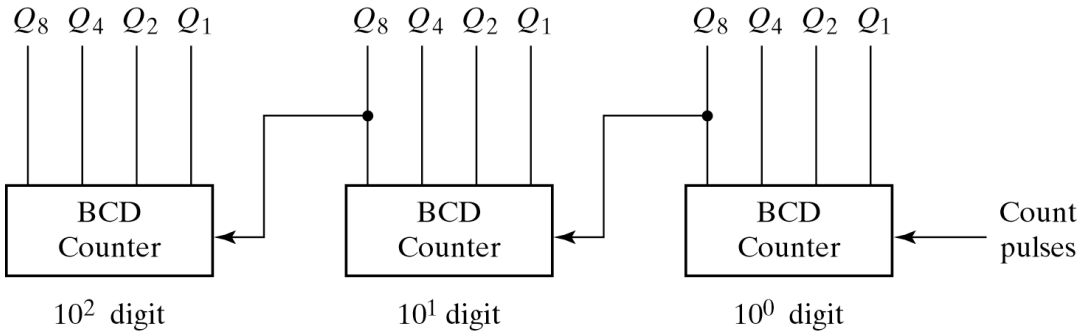
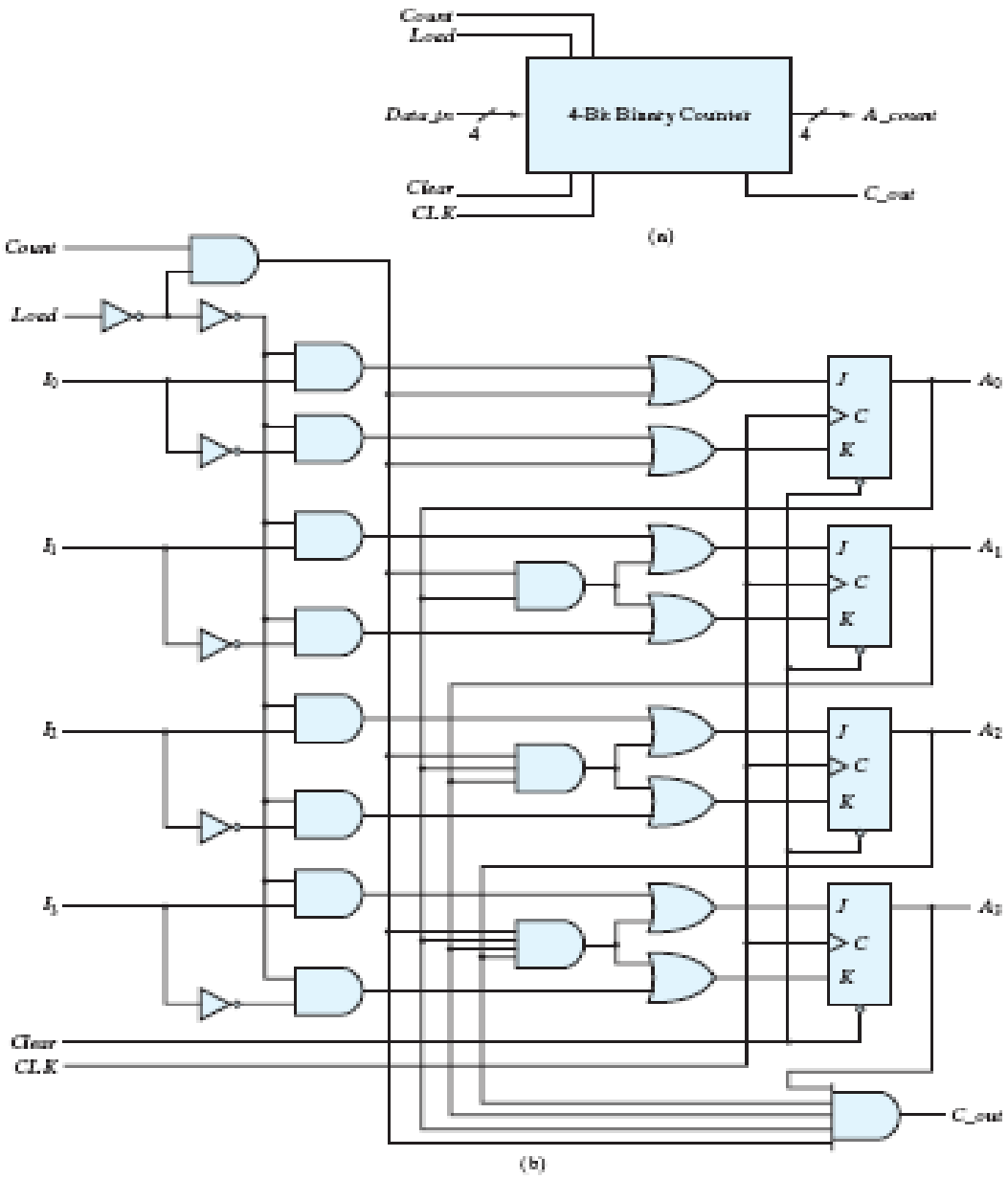


Fig. 6-11 Block Diagram of a Three-Decade Decimal BCD Counter

Counter with Parallel load



More BCD Counters (with Parallel Load)

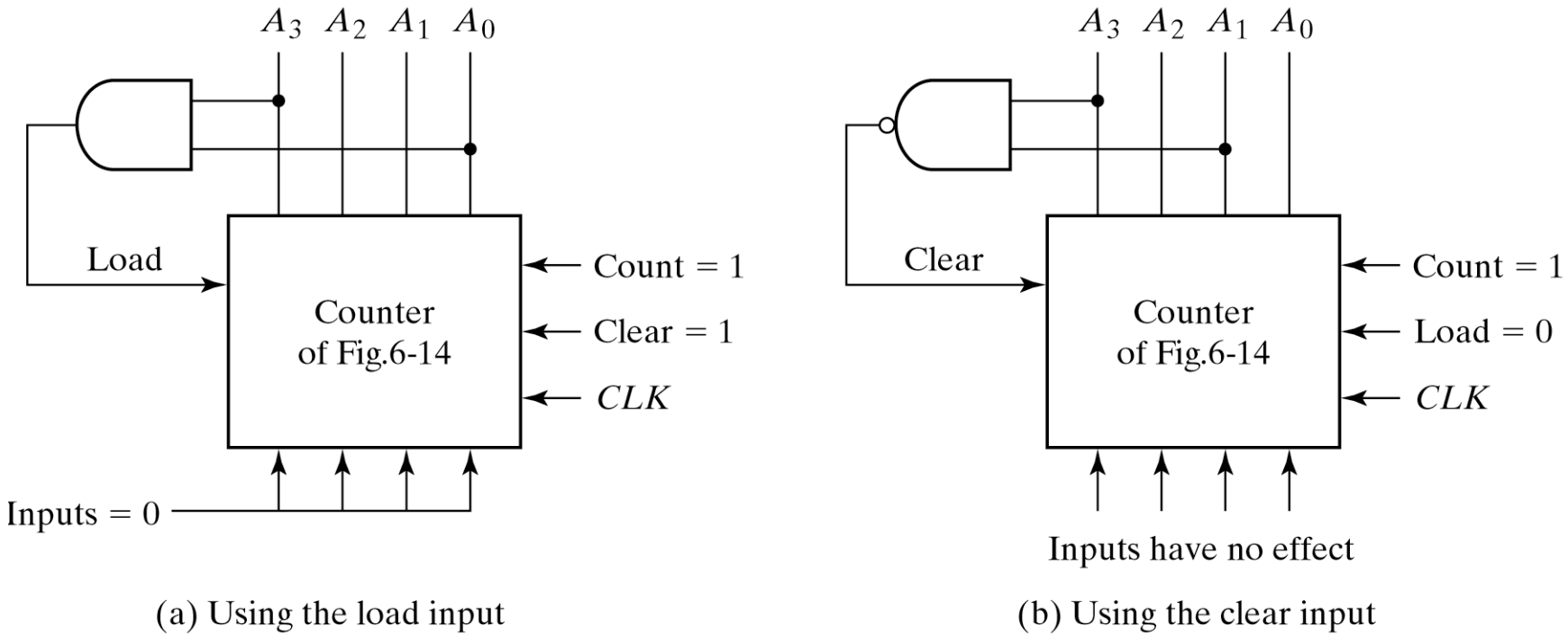


Fig. 6-15 Two ways to Achieve a BCD Counter Using a Counter with Parallel Load

Sequential Logic Circuits: Examples

(**Note:** in addition to the examples presented here other examples were discussed in the class)

-Asynchronous counters (3 examples were discussed in the class)

-Synchronous counters (2 examples were discussed in the class)

- Steps for building synchronous counters (comprehensive examples with animation is used here to show the steps with the D flip Flops)

- Ring counter

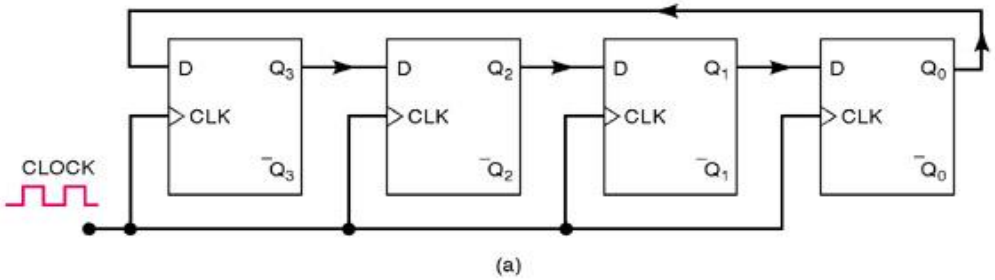
- Johnson counter

Ring Counter

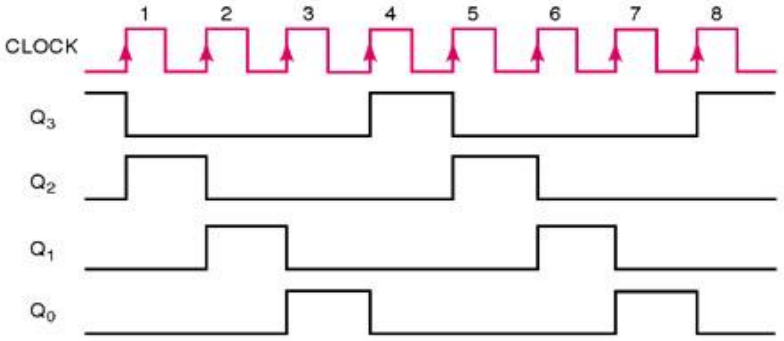
- A ring counter takes the serial output of the last Flip-Flop of a shift register and provides it to the serial input of the first Flip-Flop.

- This is also known as a re-circulating shift register.

Shift-Register Counters: Ring Counter



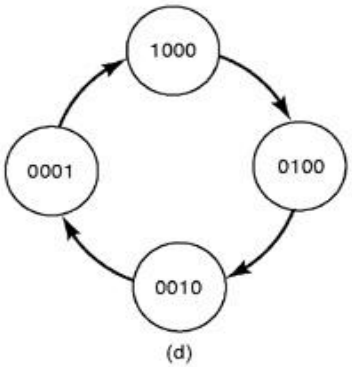
(a)



(b)

Q_3	Q_2	Q_1	Q_0	CLOCK pulse
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7
.	.	.	.	.
.	.	.	.	.

(c)



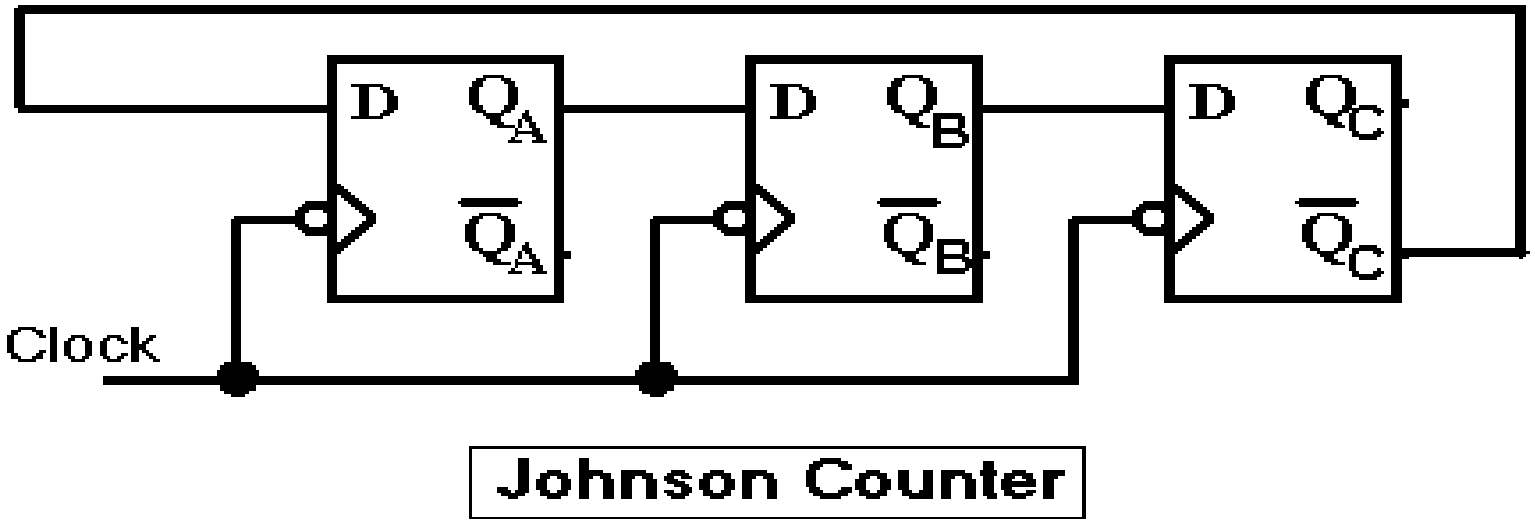
(d)

Johnson Counter

- A Johnson Counter re-circulates the last flip-flop Q' (inverted) output back to the input of the first Flip-Flop.

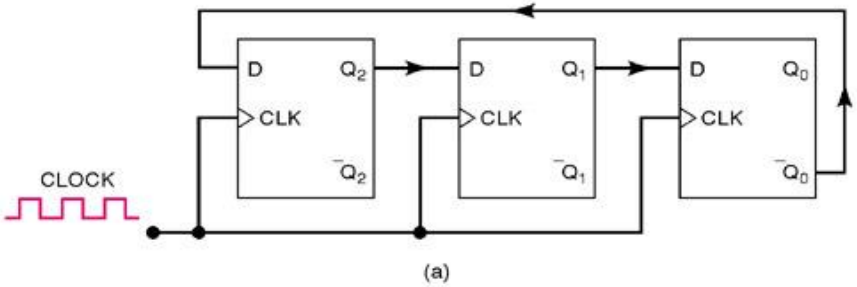
000, 100, 110,111,011,001, 000, 100, 110, ….

Initial state

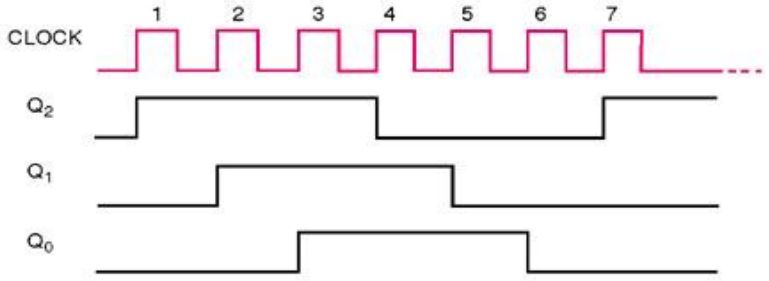


Johnson Counter

Johnson Counter



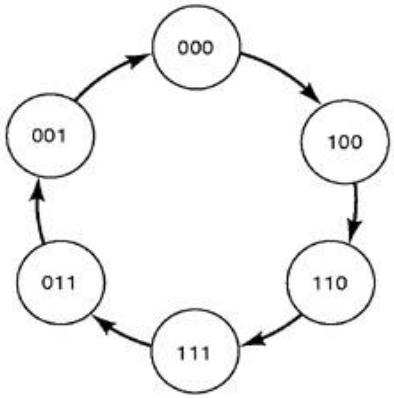
(a)



(b)

Q_2	Q_1	Q_0	CLOCK pulse
0	0	0	0
1	0	0	1
1	1	0	2
1	1	1	3
0	1	1	4
0	0	1	5
0	0	0	6
1	0	0	7
1	1	0	8
.	.	.	.
.	.	.	.
.	.	.	.

(c)



(d)