Chapter 6

Registers & Counters

Introduction

- Have analysis and design tools to develop combination and sequential circuits
- Shall explore devices in this Chapter
- Registers
  - Parallel Registers versus Shift Registers
  - The Serial Adder
- Counters
  - Asynchronous (Ripple) Counters versus Synchronous Counters
  - Binary and BCD Counters
  - Johnson and Ring Counters

Registers

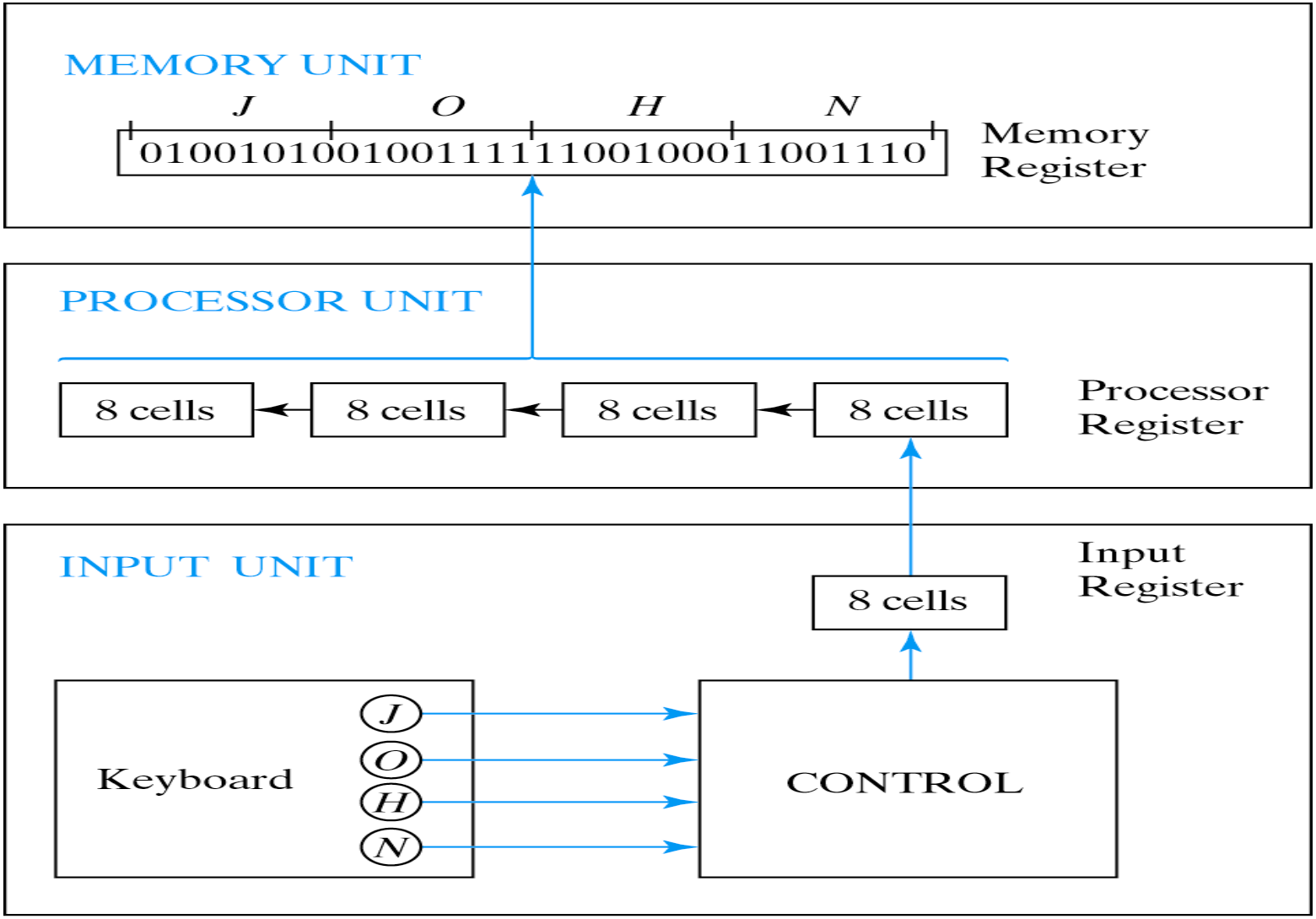


Fig. 1-1 Transfer of information with registers

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

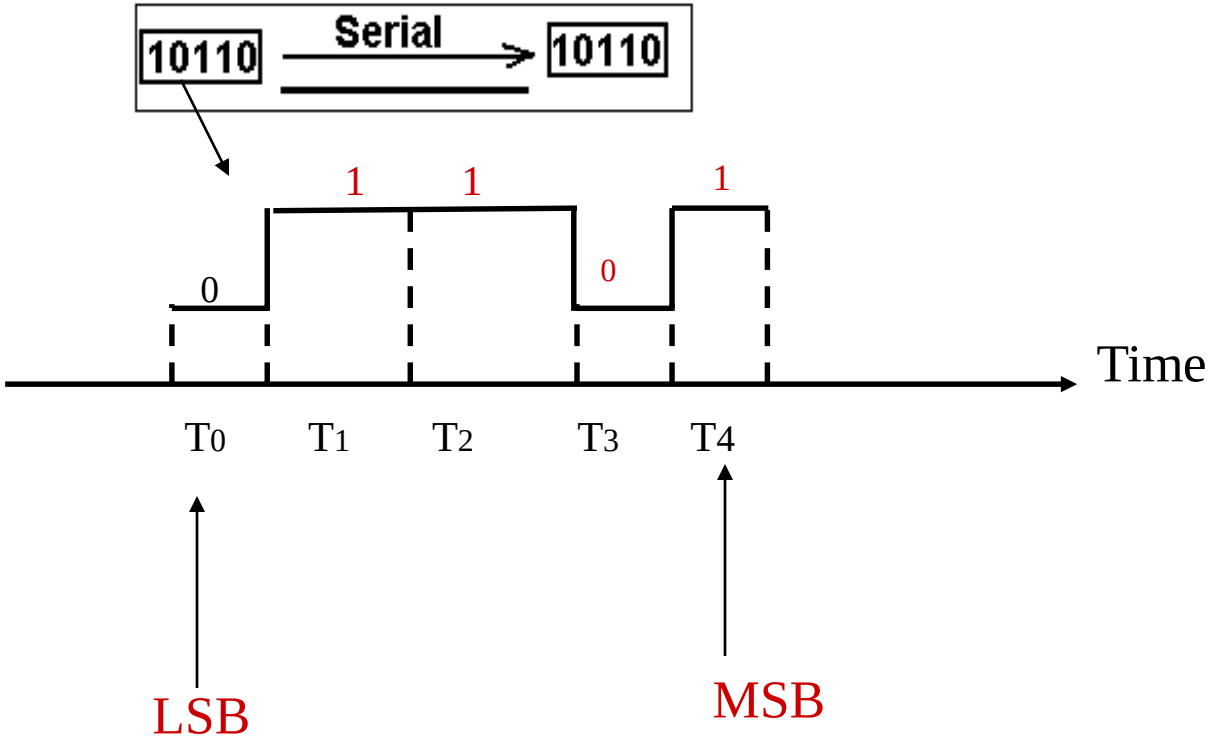
  →Many bits transferred at the same time
- Shift registers can be used with adders to build arithmetic units
- Remember: most digital hardware can be built from combinational logic (and, or, invert) and flip flops

  →Basic components of most computers

Parallel versus Serial

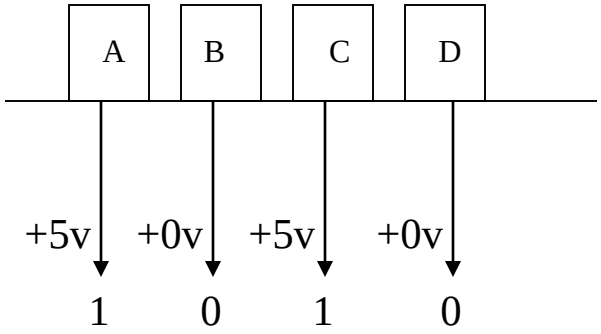
→Serial communications, transfers a binary number as a sequence of binary digits, one after another, through one data line.

- One Circuit is necessary to represent any binary number

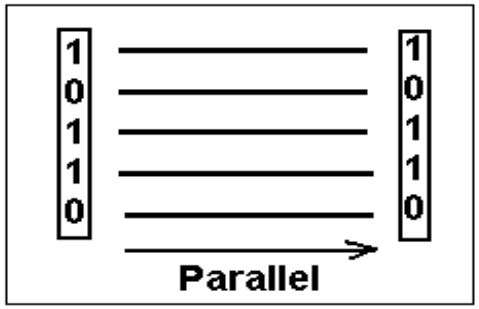


Parallel versus Serial

→ Parallel communications

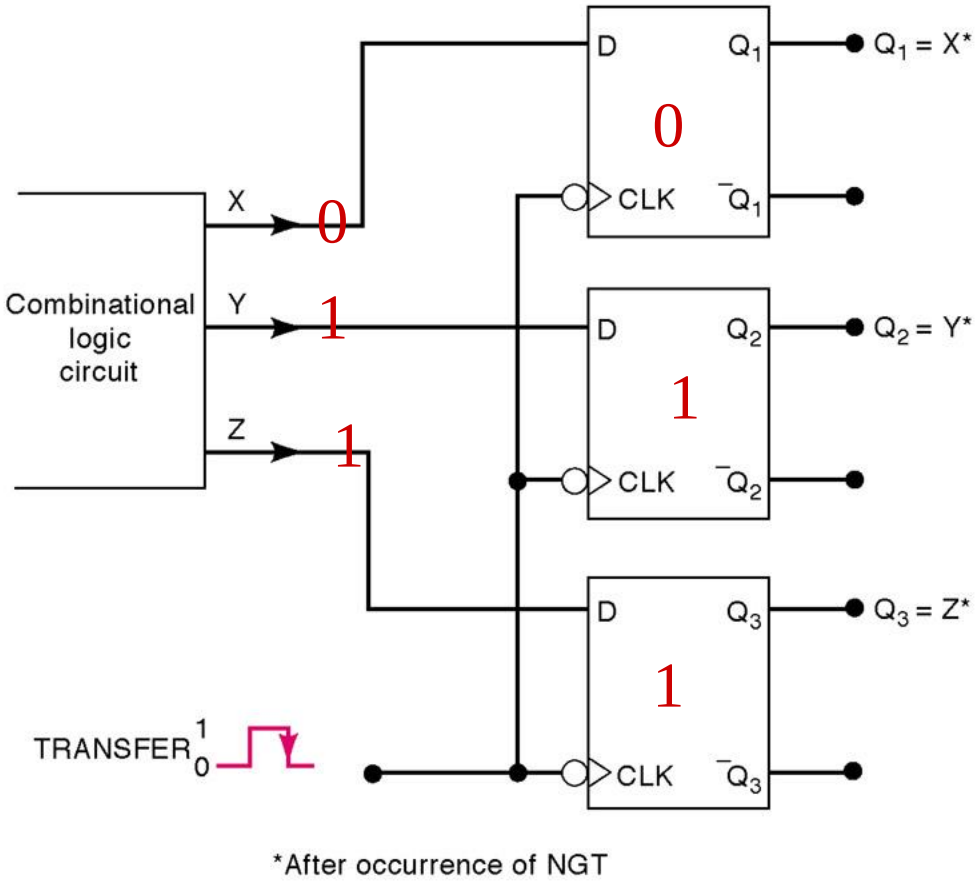


–Transfers a binary number through multiple data lines at the same time.



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)

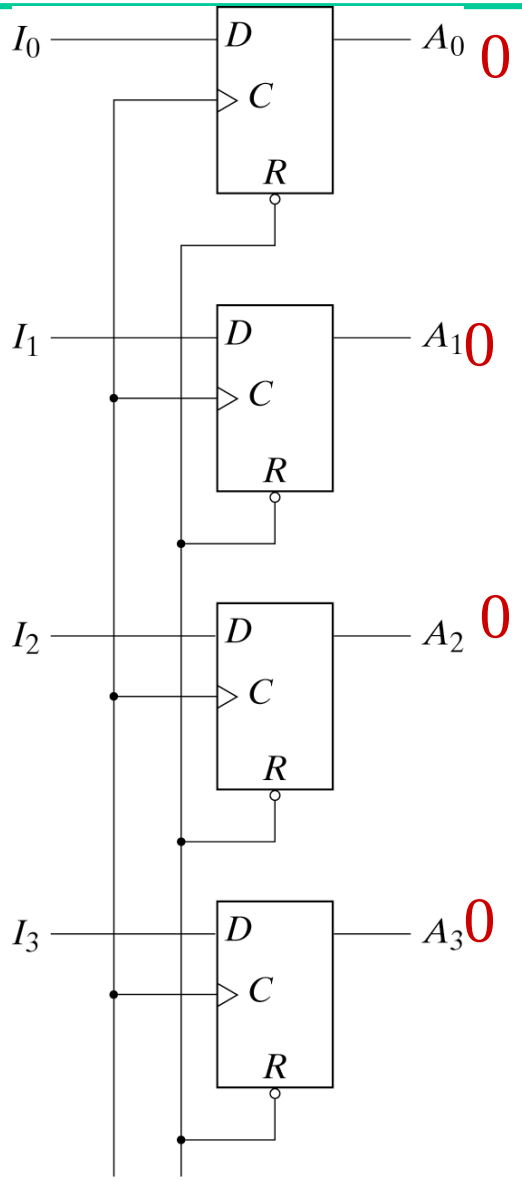


Fig. 6-1 4-Bit Register

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)

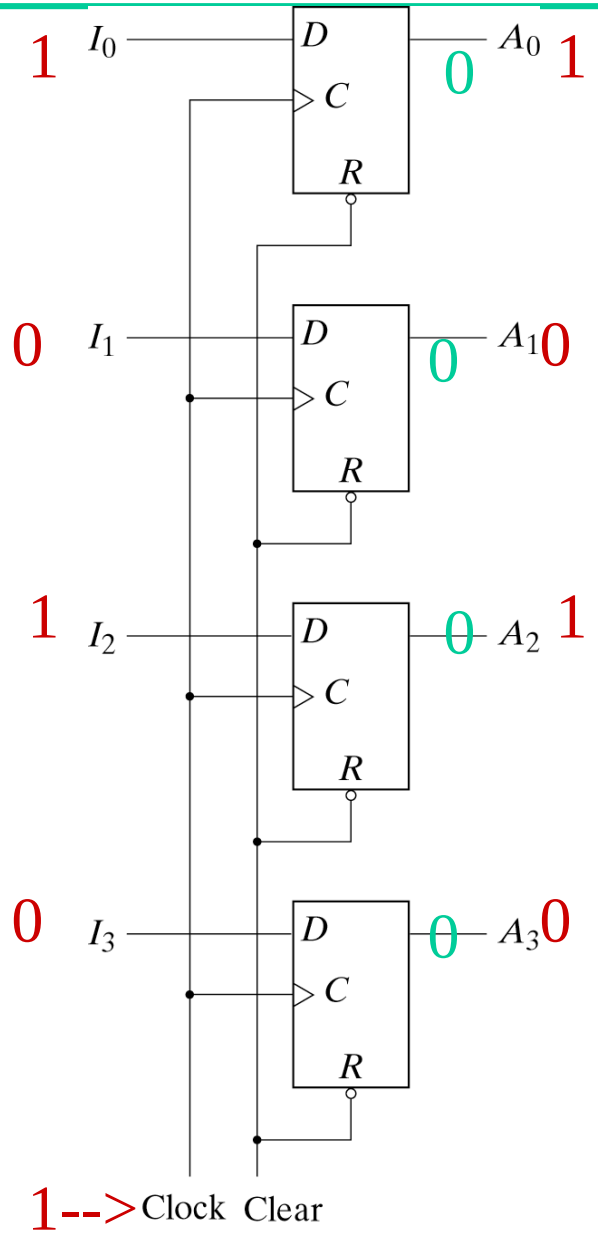
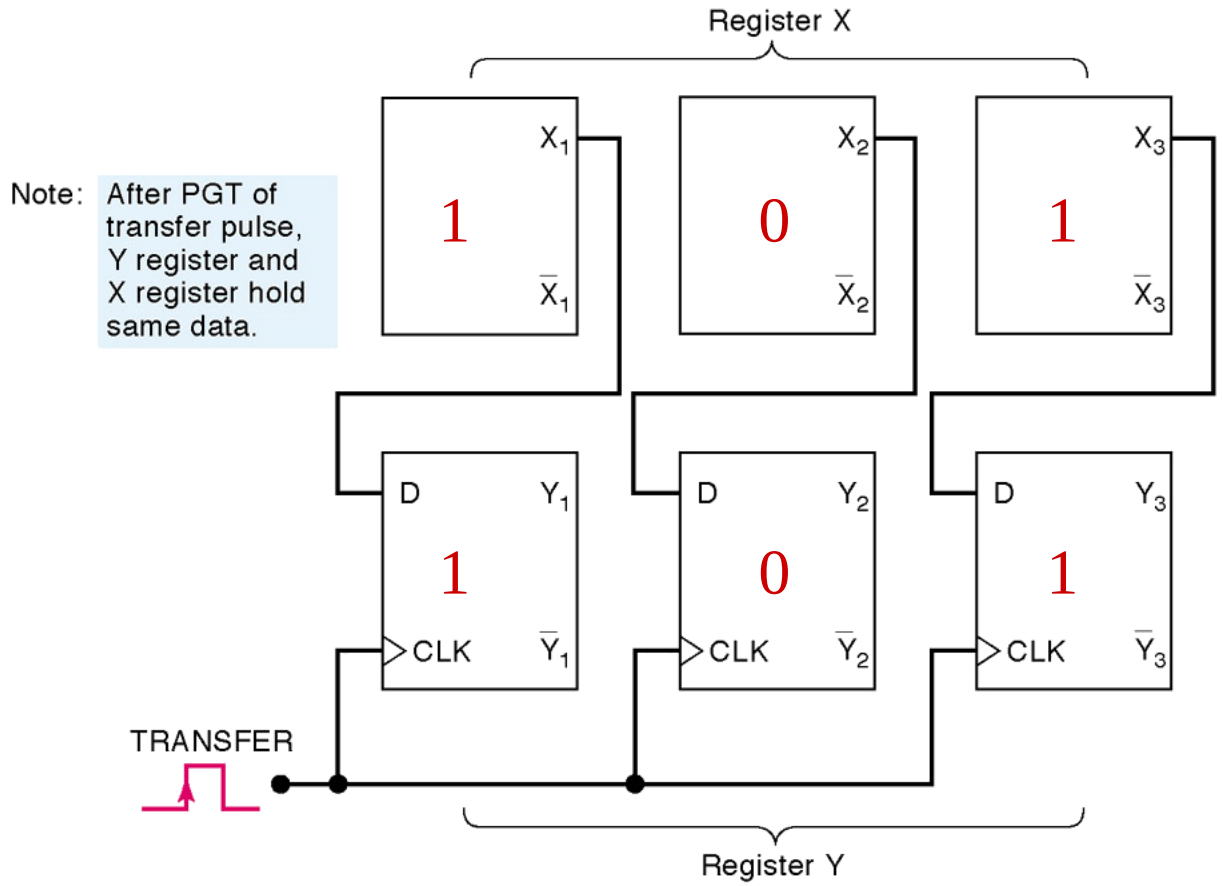


Fig. 6-1 4-Bit Register

Parallel Data Transfer

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



Register with Load Control

- Want to control the loading of the register
  - Do not control clock signal – delays!!
  - Load Control = 1
    - New data loaded on next clock edge
  - Load Control = 0
    - Old data reloaded on next clock edge

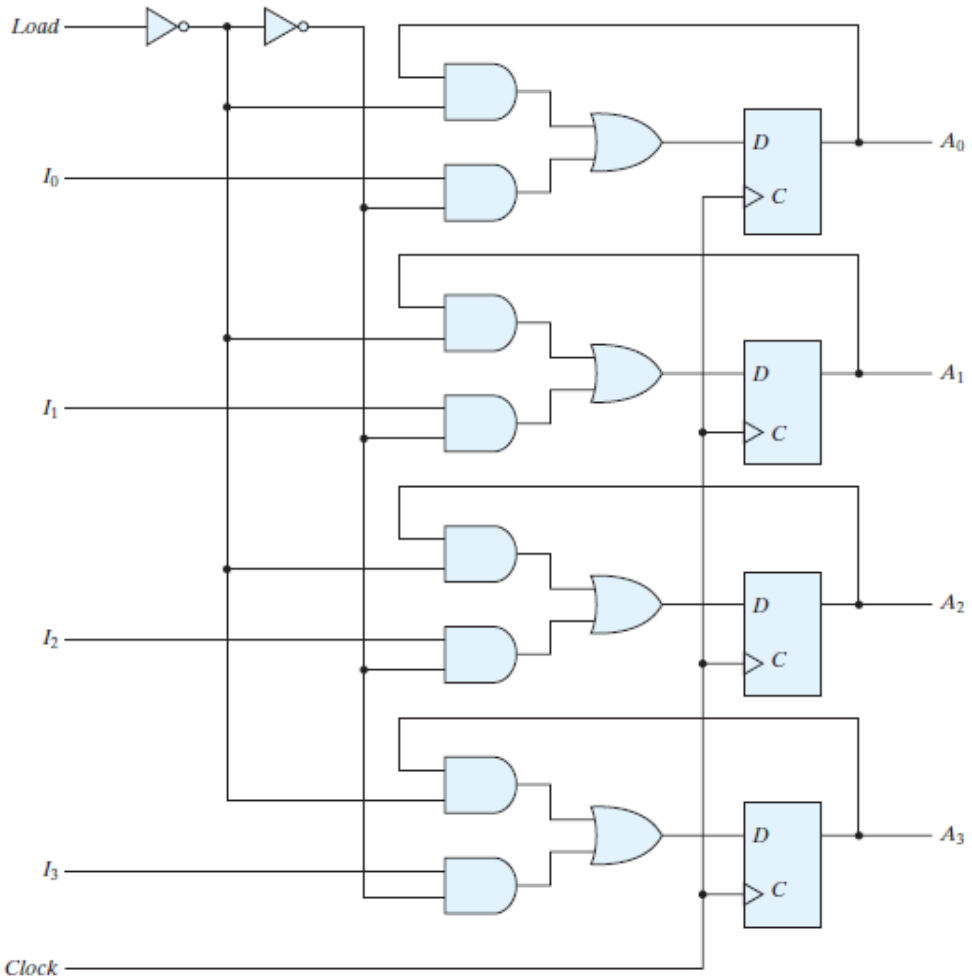
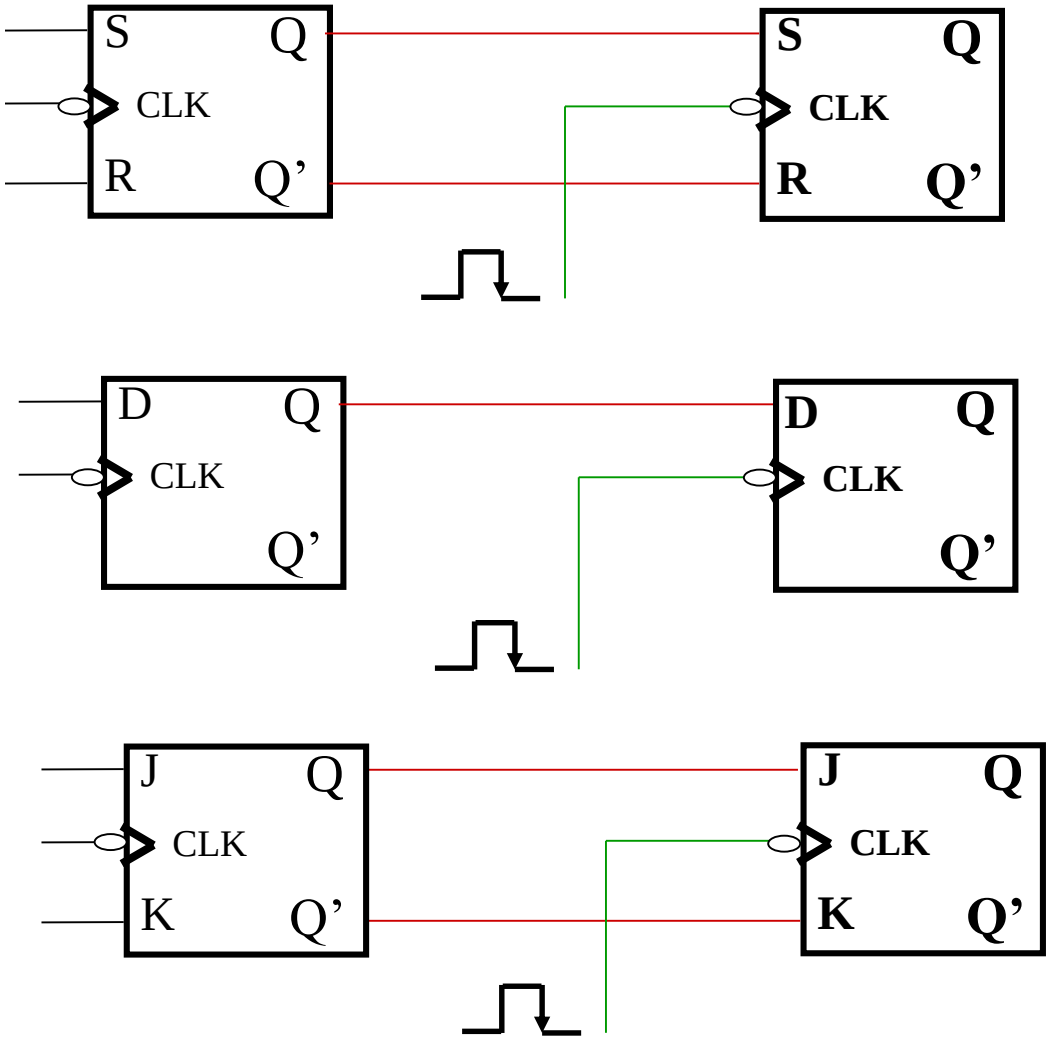


FIGURE 6.2
Four-bit register with parallel load

Serial Transfer



Shift Registers

- Cascade chain of Flip-Flops
- Bits travel on positive edges (in this example)
- Serial in (SI) – Serial out (SO)
- Data is transferred one bit at a time

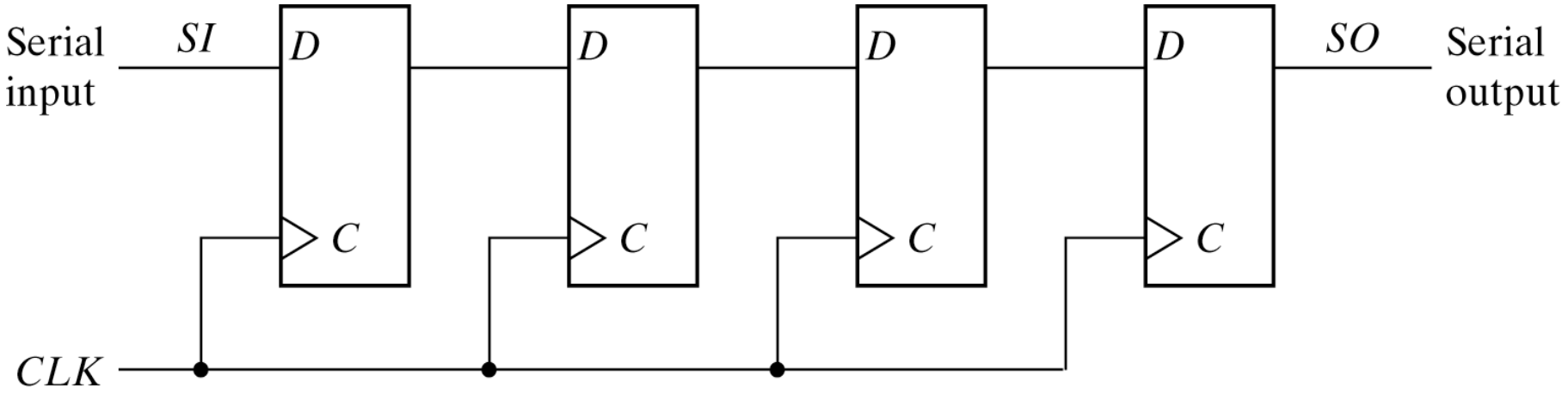
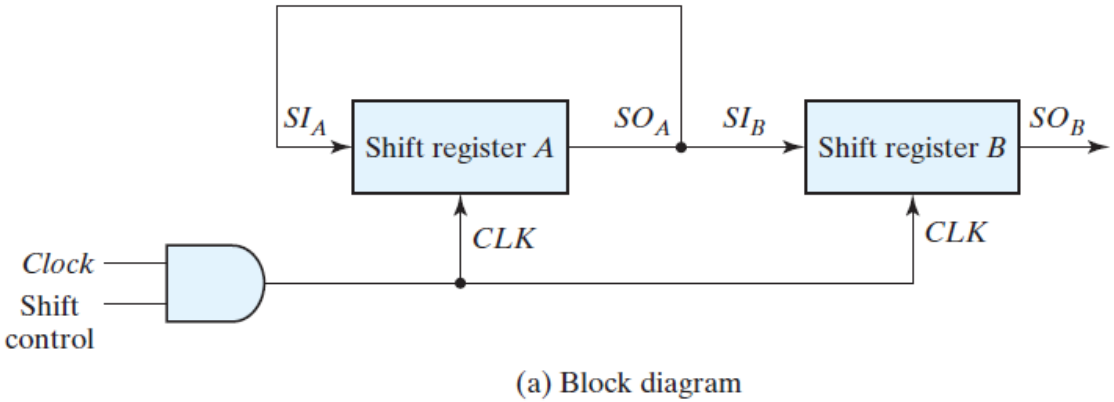


Fig. 6-3 4-Bit Shift Register

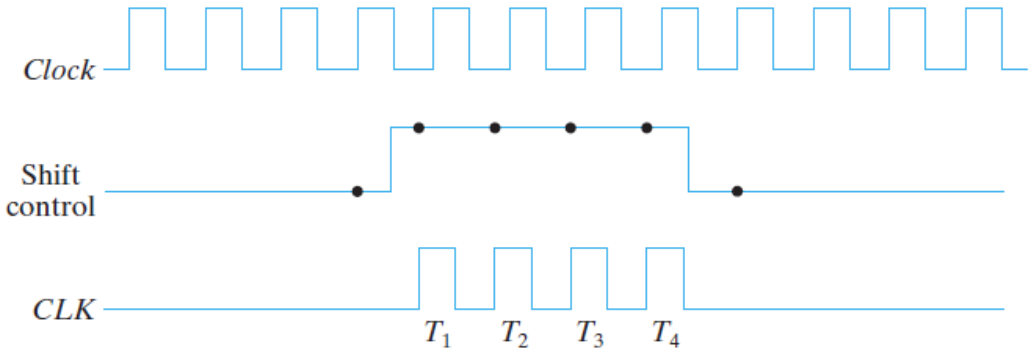
Serial Transfer

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



(a) Block diagram

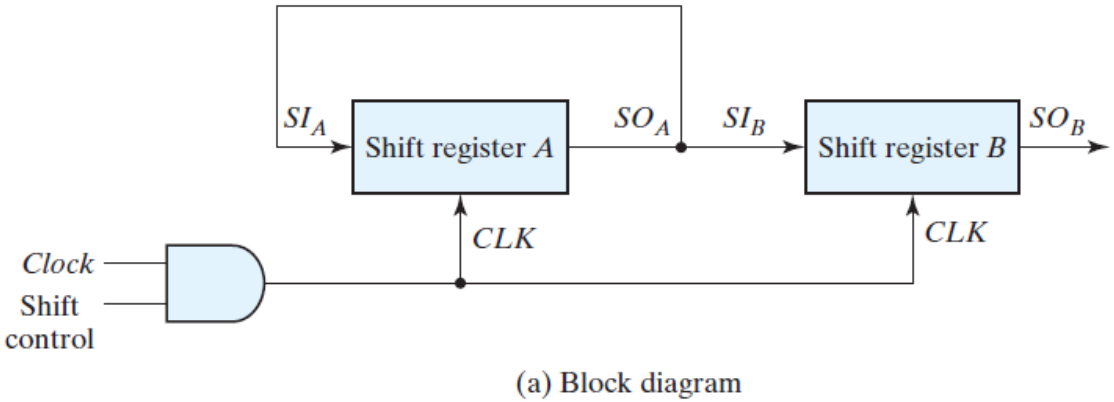
Time	A	B
T0	1011	0011
T1		
T2		
T3		
T4		



(b) Timing diagram

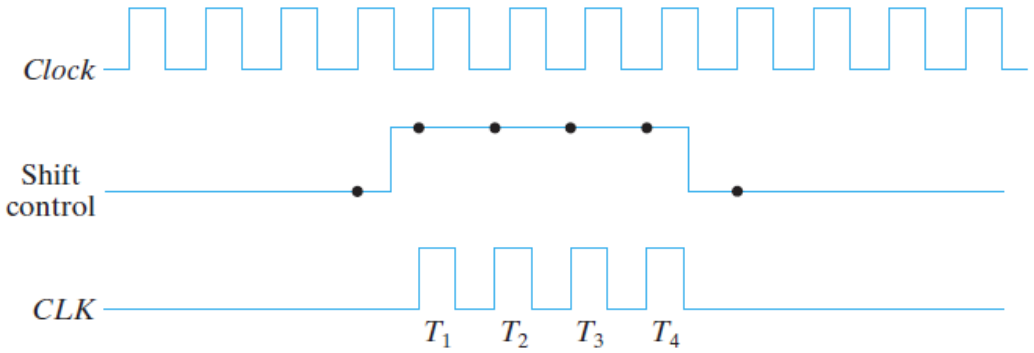
Serial Transfer

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



(a) Block diagram

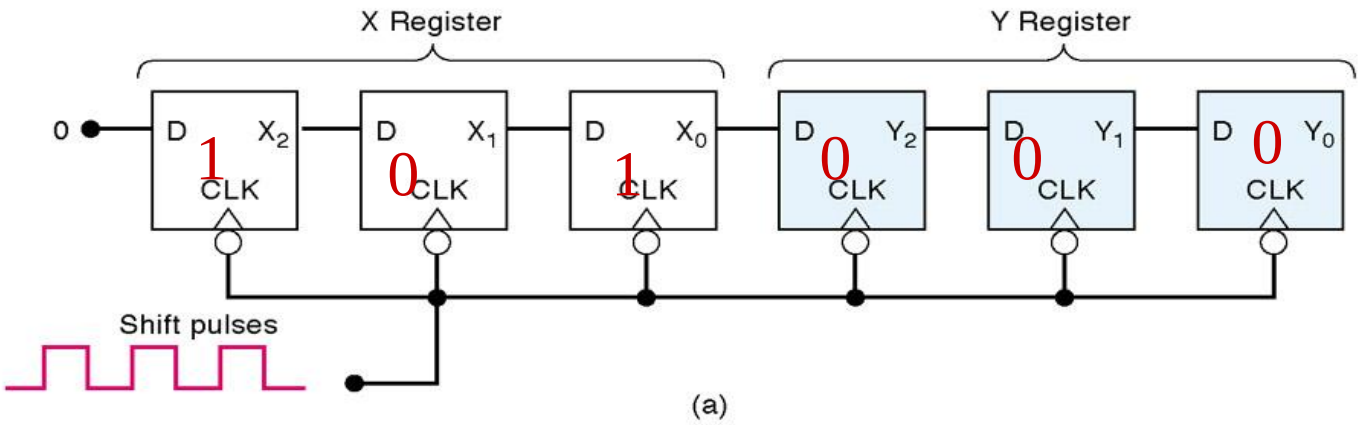
Time	A	B
T0	1011	0011
T1	1101	1001
T2	1110	1100
T3	0111	0110
T4	1011	1011



(b) Timing diagram

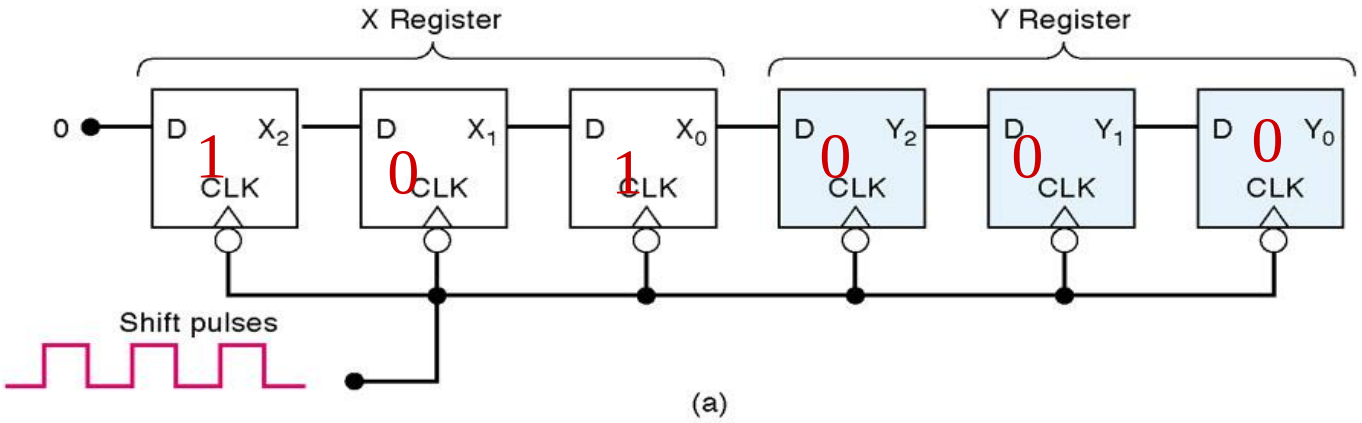
Serial transfer of Data

- Transfer from register X to register Y (Negative clock edges in this example)



Serial transfer of Data

- Transfer from register X to register Y (Negative clock edges in this example)

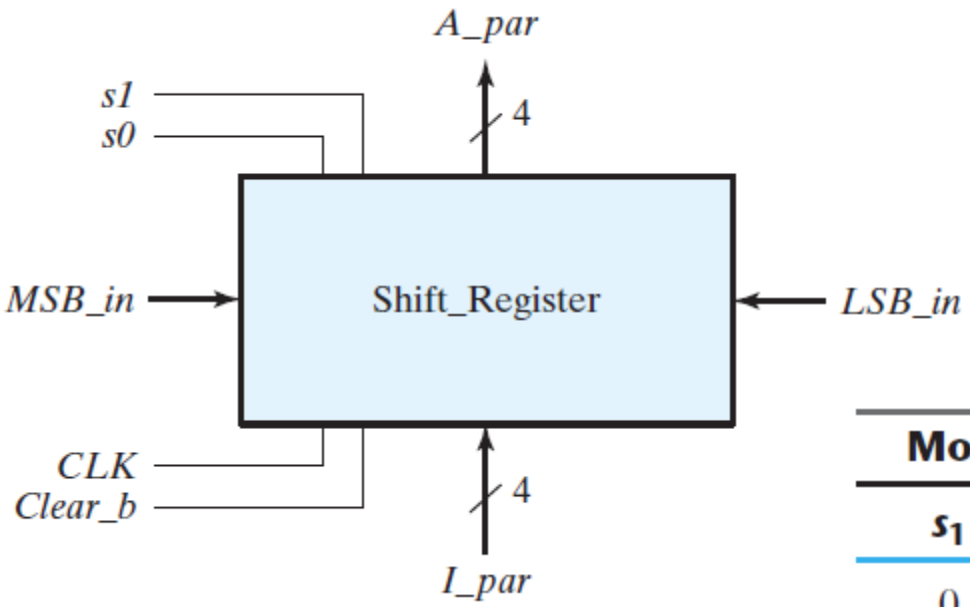


X_2	X_1	X_0	Y_2	Y_1	Y_0	
1	0	1	0	0	0	← Before pulses applied
0	1	0	1	0	0	← After first pulse
0	0	1	0	1	0	← After second pulse
0	0	0	1	0	1	← After third pulse

(b)

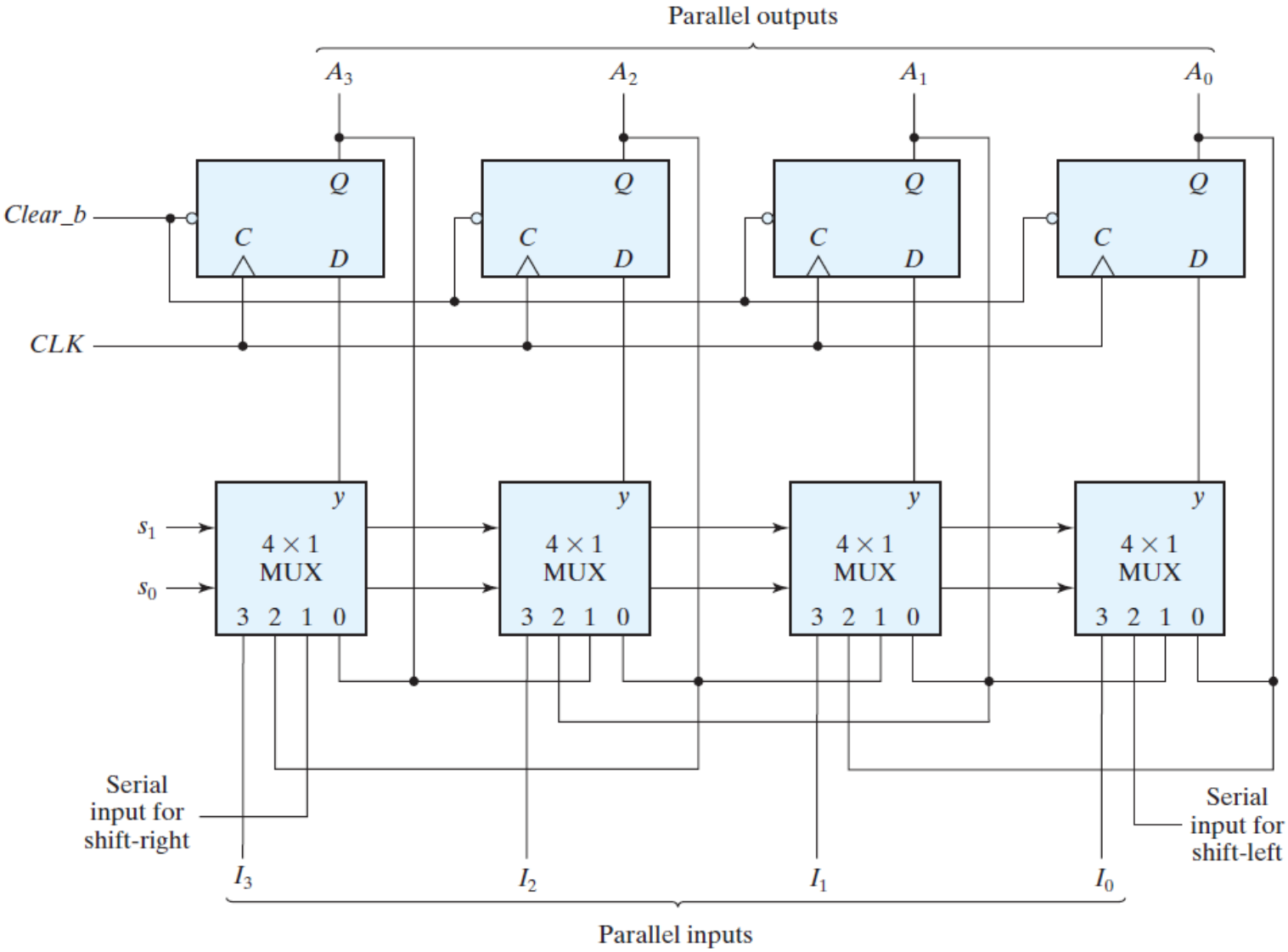
Universal Shift Register

- Allows for shift operations in both directions and parallel load.



Mode Control		
s_1	s_0	Register Operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

Universal Shift Register (continued)



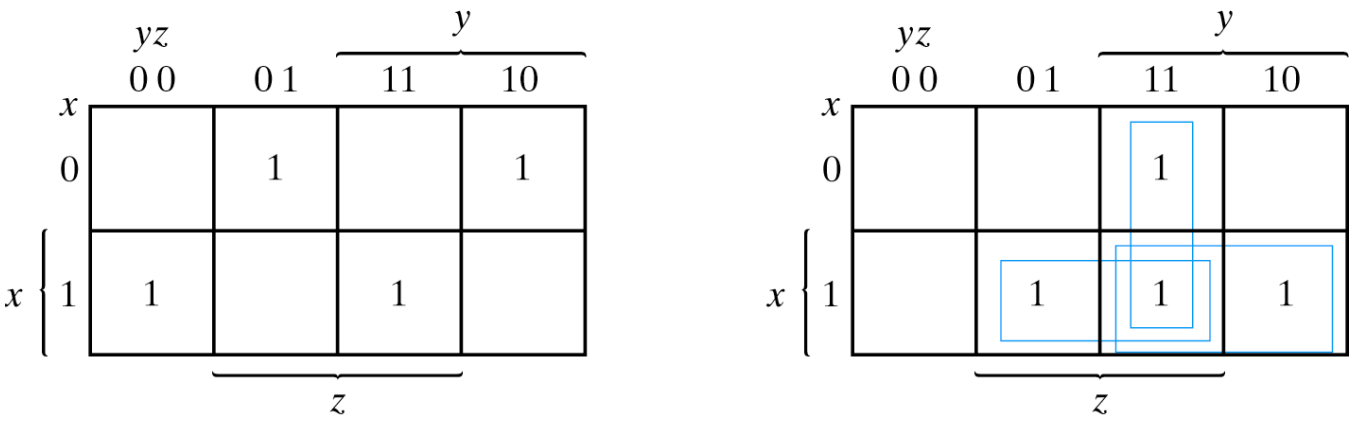
Full Adder (see other implementations in Chap. 2)

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)

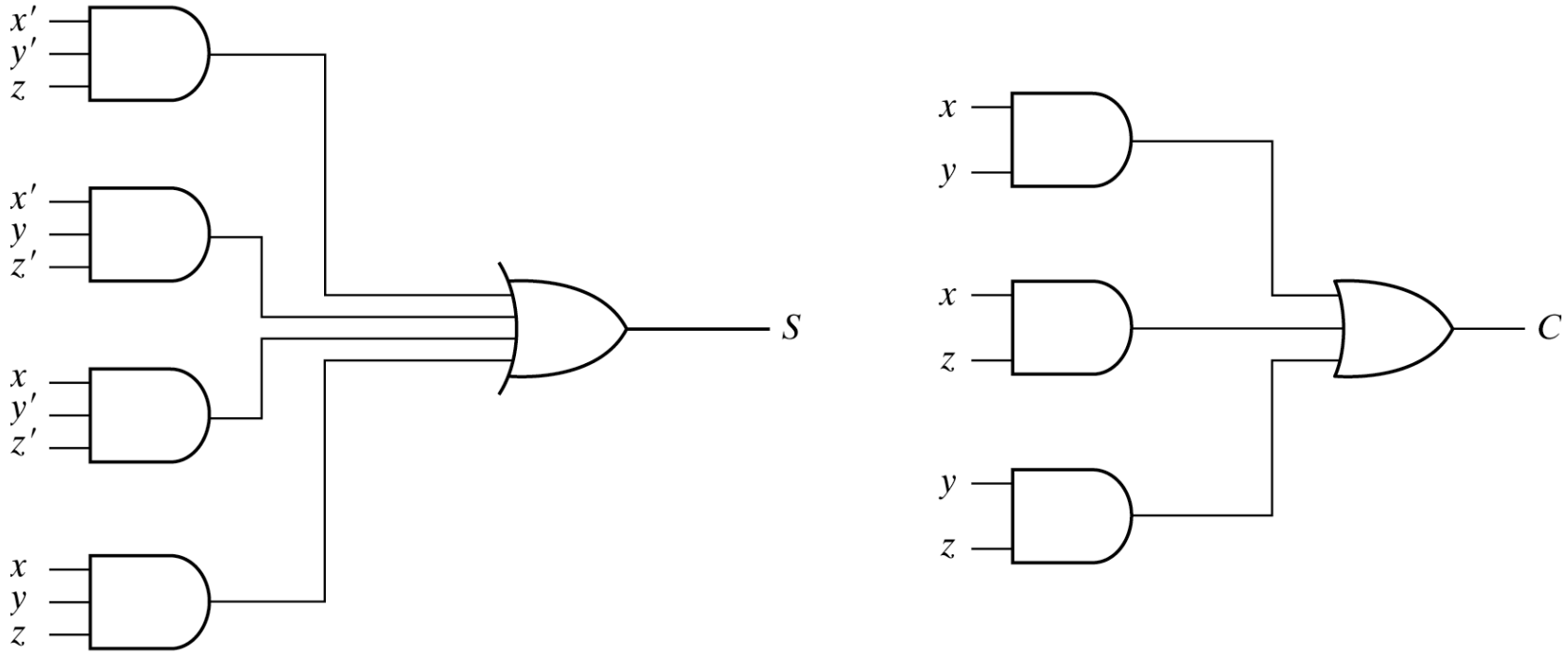


Fig. 4-7 Implementation of Full Adder in Sum of Products

Full Adder (same as in Chap. 2)

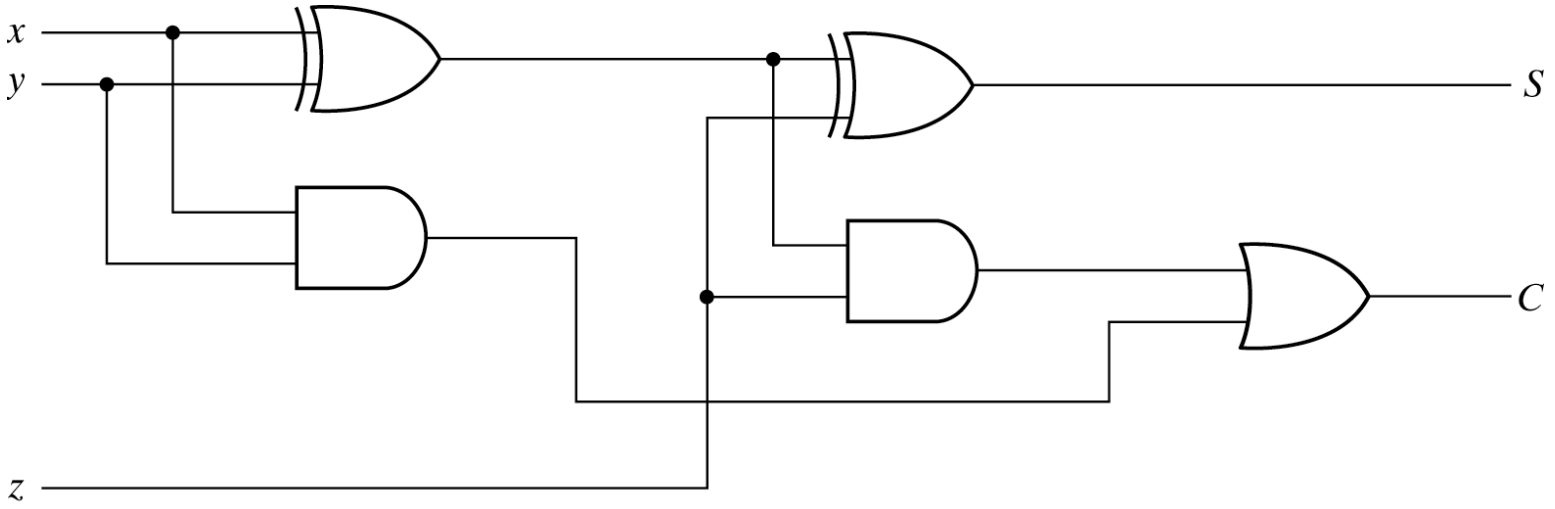


Fig. 4-8 Implementation of Full Adder with Two Half Adders and an OR Gate

Full Adder (see other implementations in Chap. 2)

Truth Table

x	y	C_{in}	C_{out}	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
- Good for multiplexed data

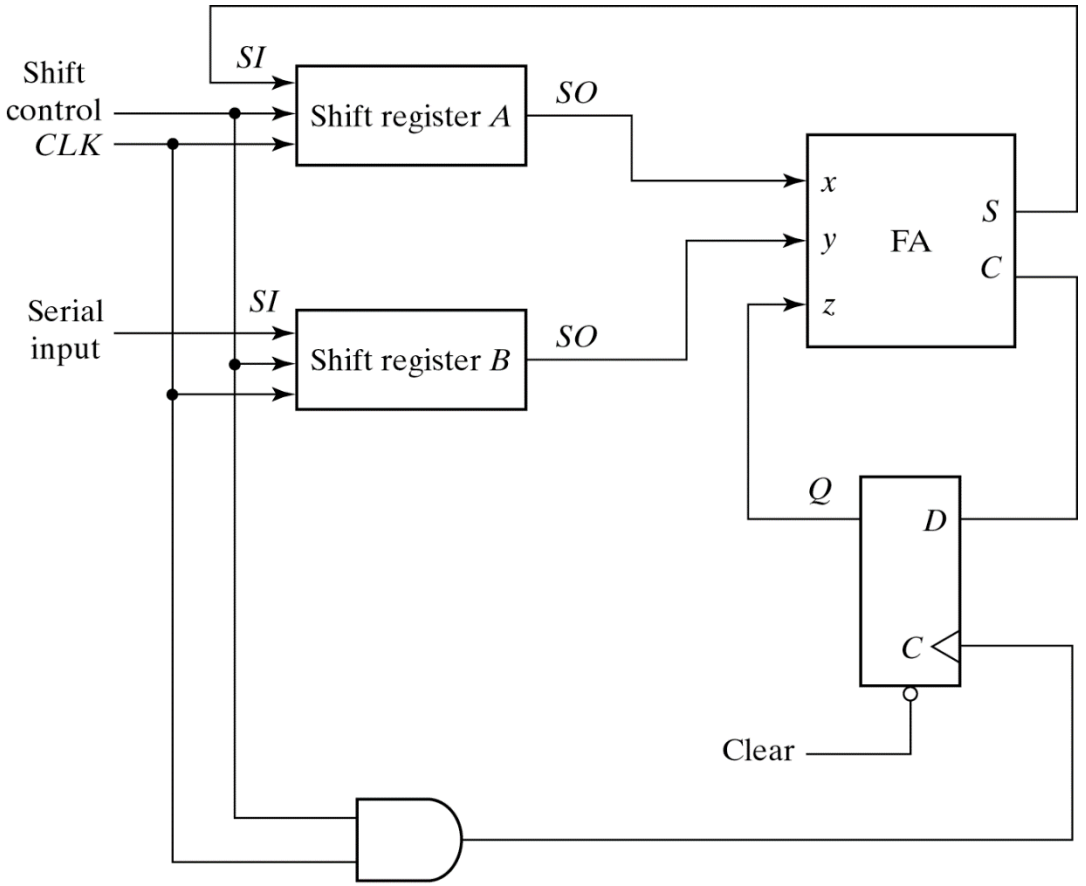


Fig. 6-5 Serial Adder

Serial Addition (D Flip-Flop)

- Only one full adder
- Reused for each bit
- Start with low-order bit addition
- Note that carry (Q) is saved
- Add multiple values.
  - New values placed in shift register B

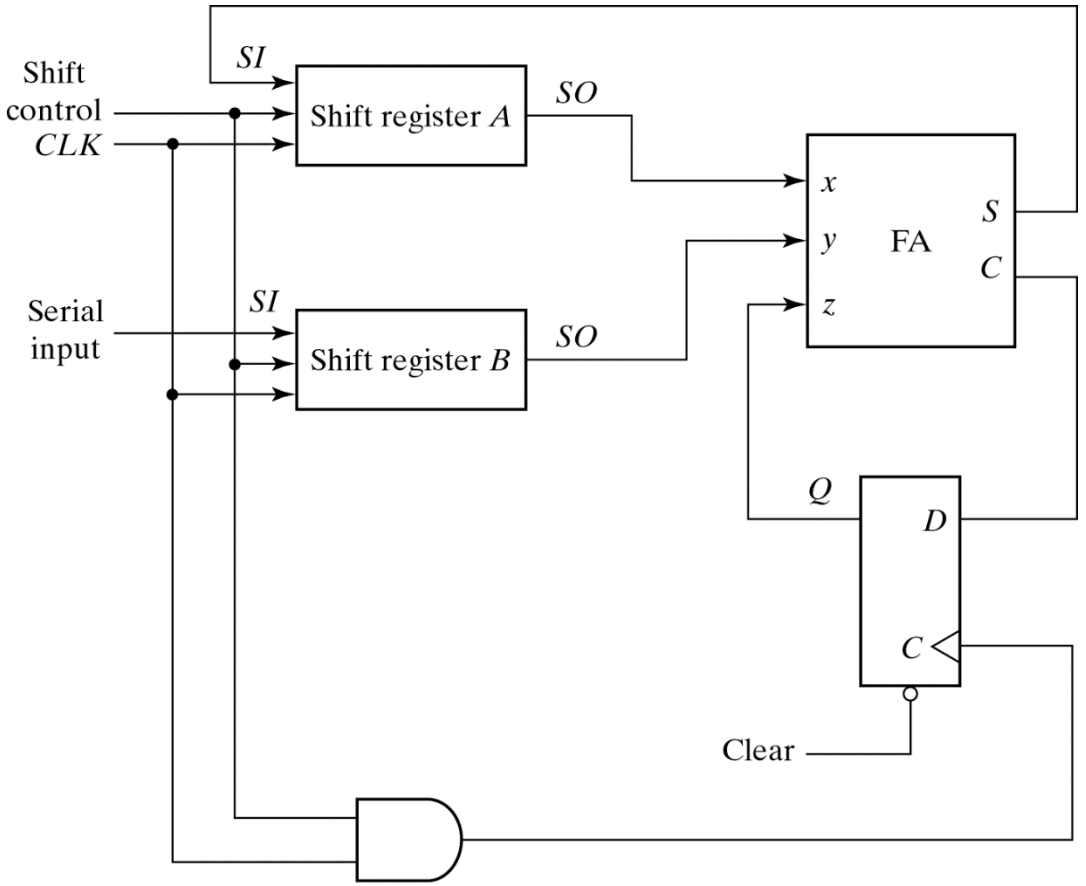


Fig. 6-5 Serial Adder

Serial Addition (D Flip-Flop)

- Shift control used to stop addition
- Generally not a good idea to gate the clock
- Shift register can be arbitrary length
- FA can be built from combin. logic

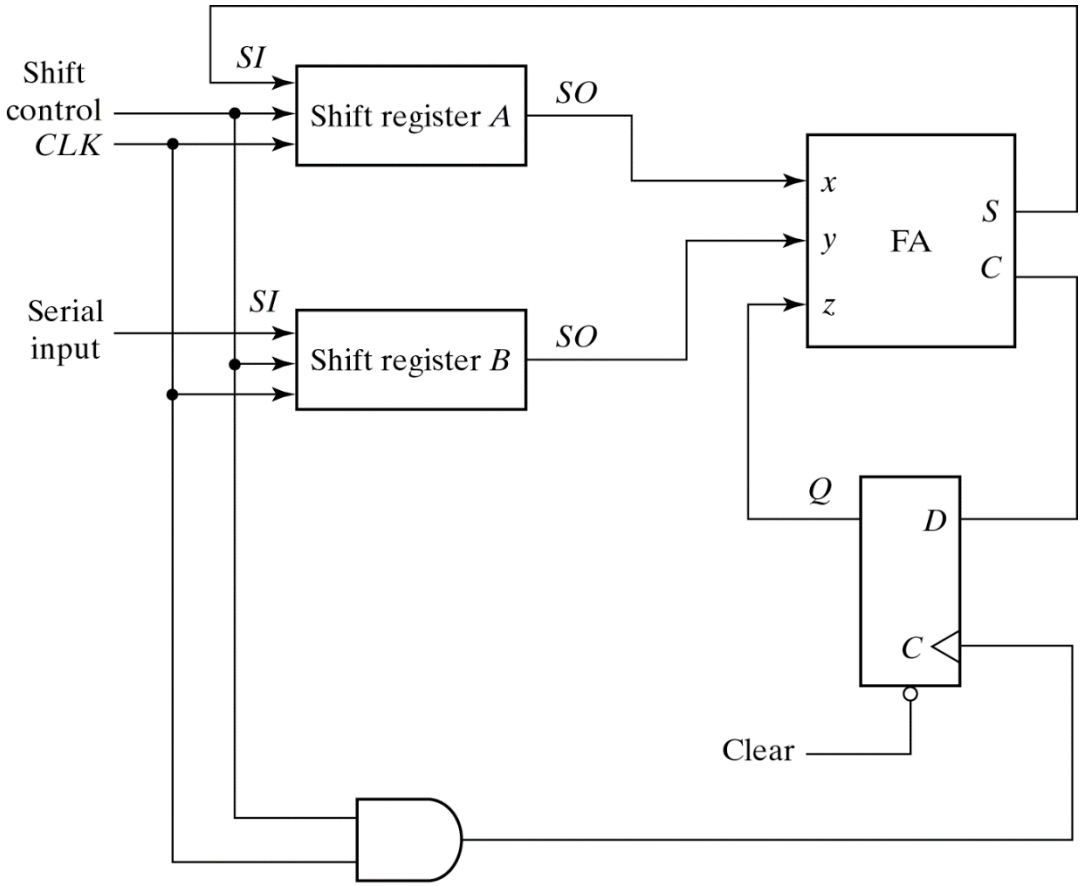


Fig. 6-5 Serial Adder

Serial Adder Operation

- Initialisation
  - Clear registers A and B and carry D FF
  - Shift first number into register B
  - When #1 number in B is shifted through FA,
    - 0's + #1 number = #1 number ⟹ A
    - #2 number ⟹ B
- Addition
  - When #2 number in B is shifted through FA,
    - #2 number + #1 number ⟹ A
    - #3 number ⟹ B
  - Can repeat for adding 3, 4, 5 … numbers

Design Serial Adder with JK FF

- Assumptions
  - Shift registers A and B are available
  - Replace FA and D FF with a JK FF and combinational circuit
  - Want to design synchronous circuit with:
    - 2 inputs: x and y which are the outputs of shift registers A and B respectively
    - Output: S – the sum bit
    - Use JK FF output Q to store the carry C
      - Note that the carry in the original design is the output of the D FF

Designing a JK Serial Adder – State Table

This is the Carry Out

This is the Carry In

Present State	Input	Next State	Output	JK FF Inputs	
Q	*xy*	Q(t+1)	S	*J*	*K*
0	00	0	0	0	x
0	01	0	1	0	x
0	10	0	1	0	x
0	11	1	0	1	x
1	00	0	1	x	1
1	01	1	0	x	0
1	10	1	0	x	0
1	11	1	1	x	0

JK FF Input Equations

$$J = xy$$

Q \ xy	00	01	11	10
0	m_0	m_1	m_3 1	m_2
1	m_4 x	m_5 x	m_7 x	m_6 x

$$K = x'y' = (x+y)'$$

q \ xy	00	01	11	10
0	m_0 x	m_1 x	m_3 x	m_2 x
1	m_4 1	m_5	m_7	m_6

$$S = x'y'Q + xyQ + x'yQ' + xy'Q'$$
$$= (x'y' + xy)Q + (x'y + xy')Q'$$
$$= (x \oplus y)' Q + (x \oplus y)Q' = (x \oplus y) \oplus Q$$

Q \ xy	00	01	11	10
0	m_0	m_1 1	m_3	m_2 1
1	m_4 1	m_5	m_7 1	m_6

New Serial Adder

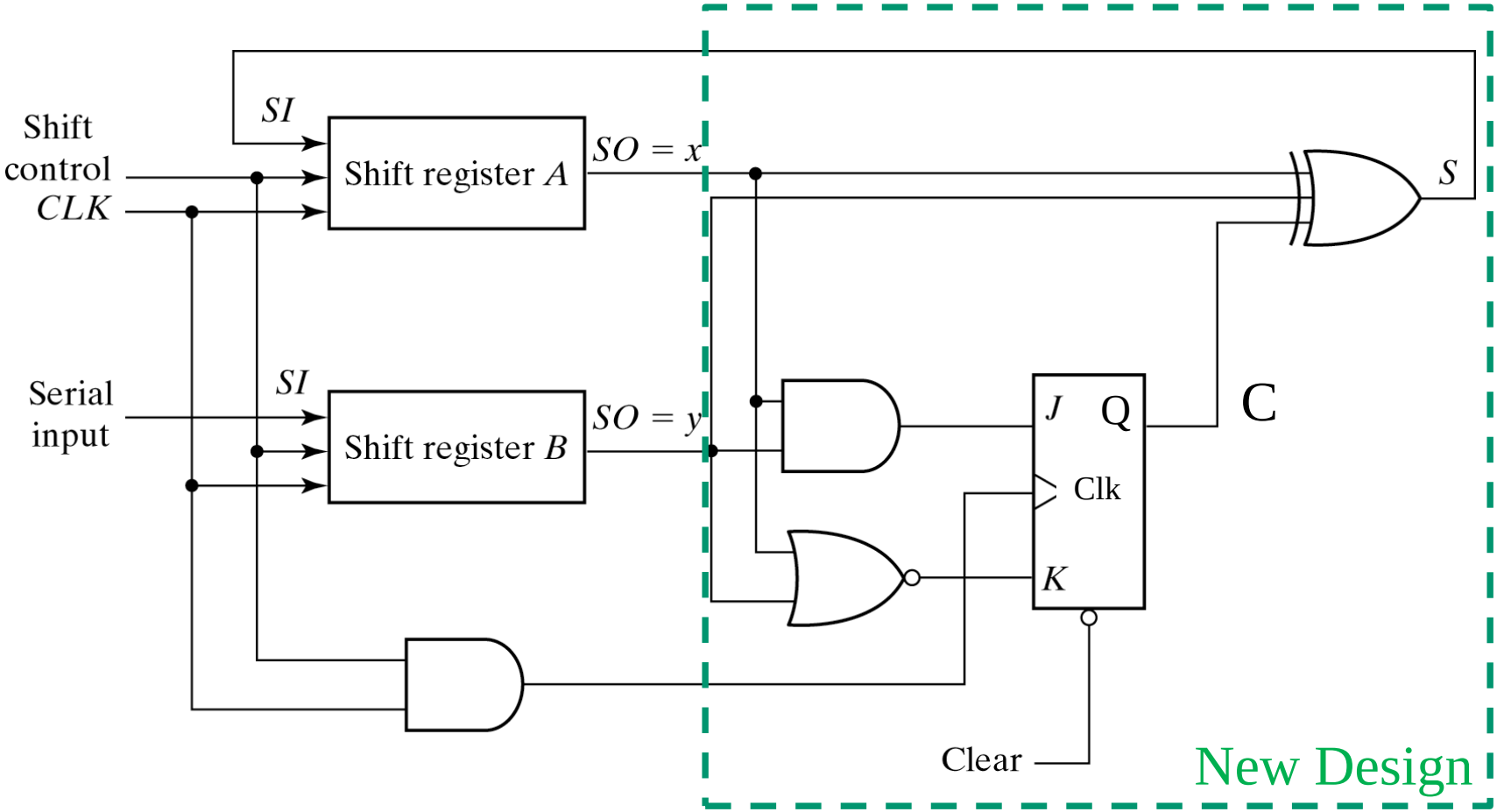


Fig. 6-6 Second form of Serial Adder

Counters

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

→Two main types of counters

1- asynchronous counters: also known as Ripple counters

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

2- 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
- Flip flop is configured to toggle
  - Next state becomes inverse of current state, i.e. Q(t+1)=Q(t)'

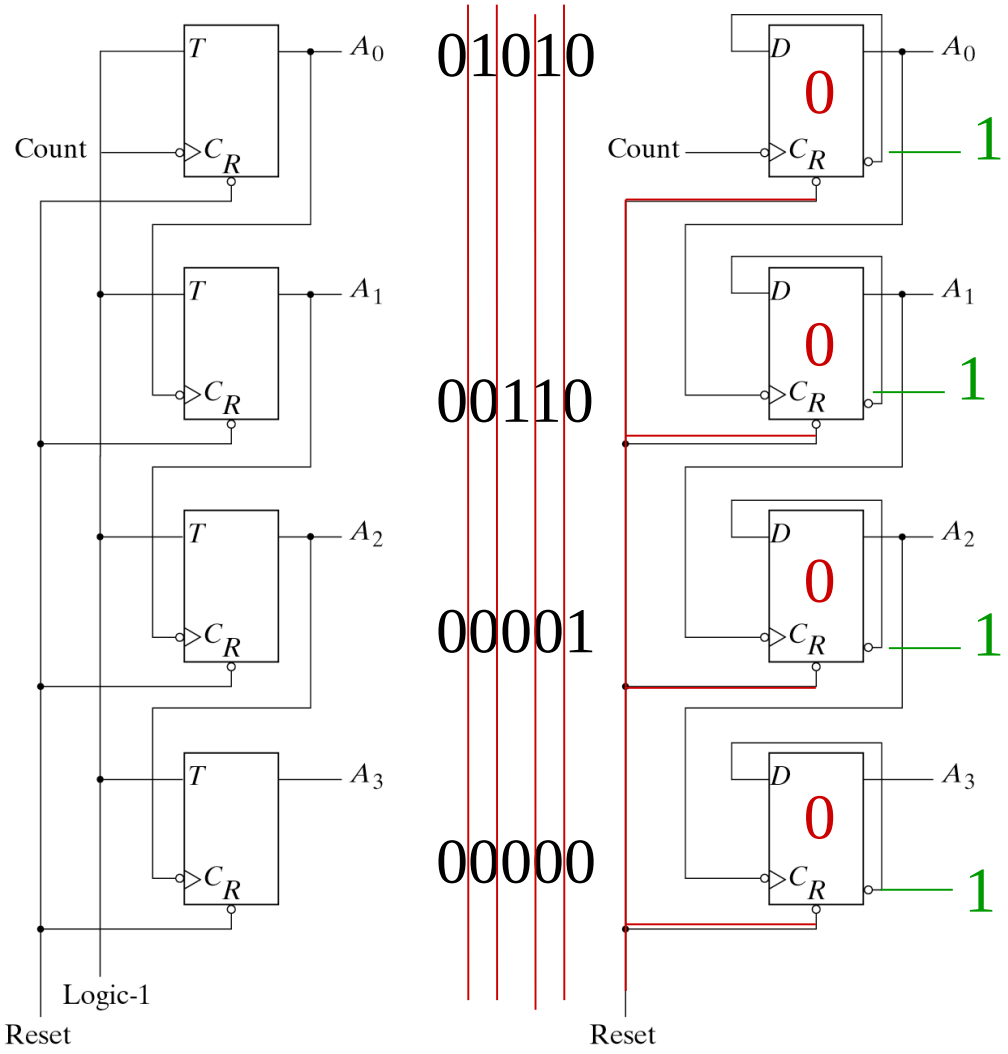
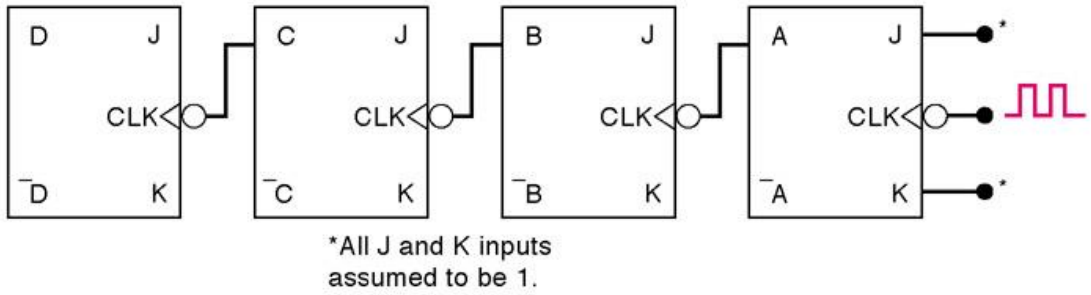
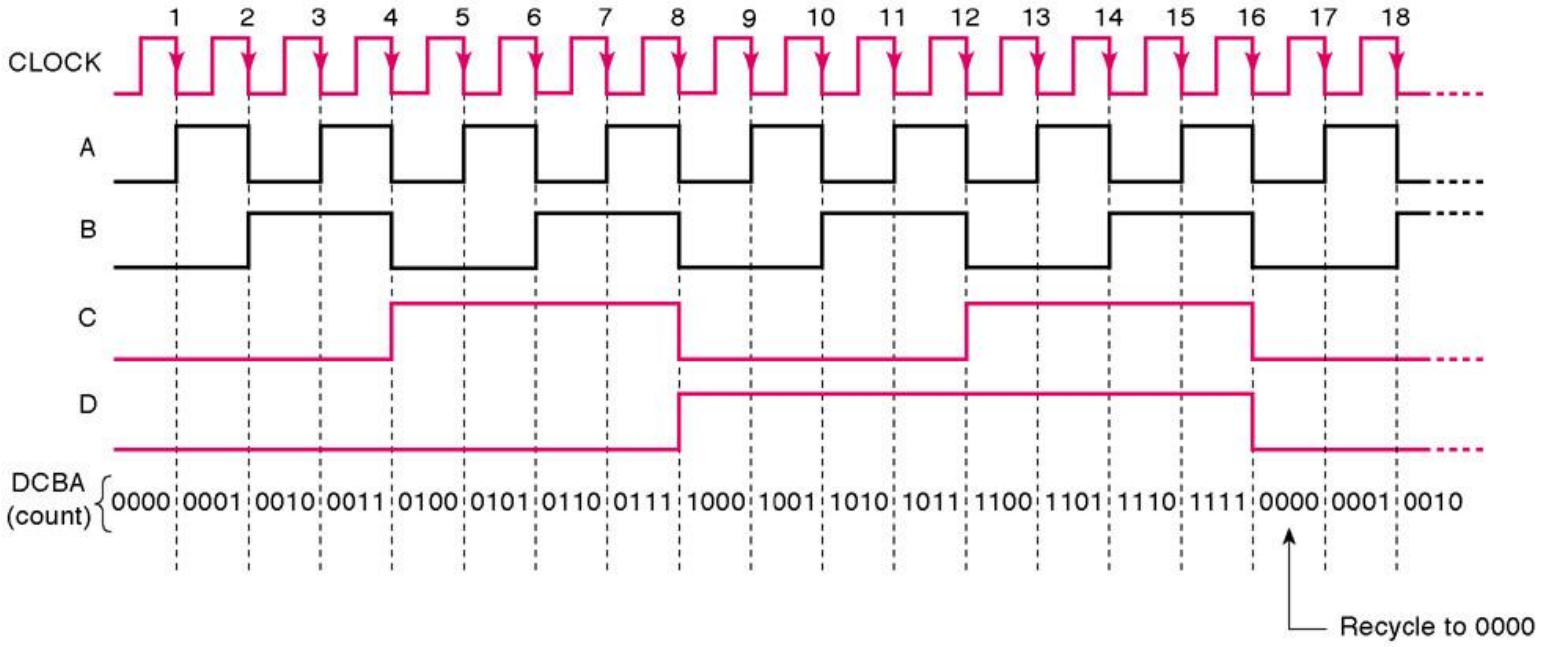


Fig. 6-8 4-Bit Binary Ripple Counter

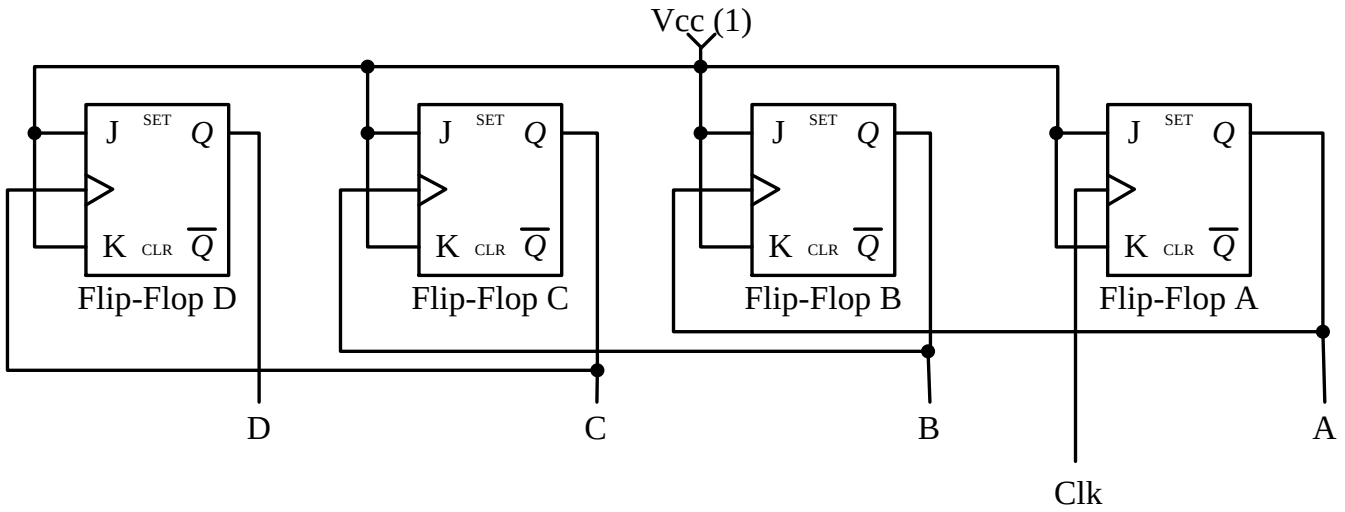
Another Asynchronous Ripple Counter





- Similar to T flop example on previous slide

Another Ripple Counter



- Starting at 0000 for DCBA, how does the counter change for each clock pulse?
- Have a down counter!!!
- Can also achieve with negative transition FFs by connecting Q' outputs to clocks – why?

Asynchronous, BCD Counter

State Diagram

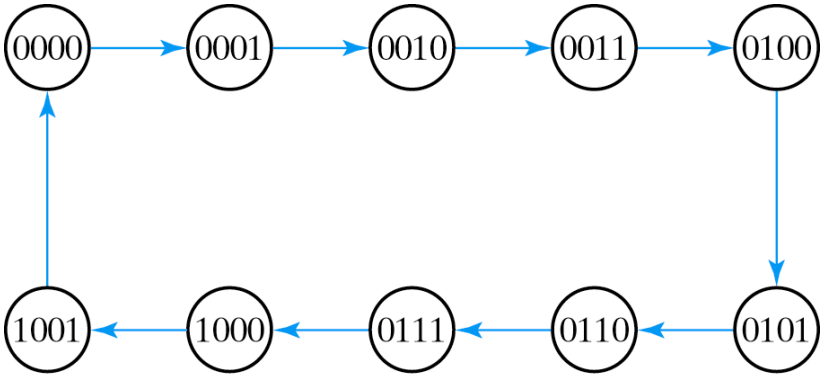


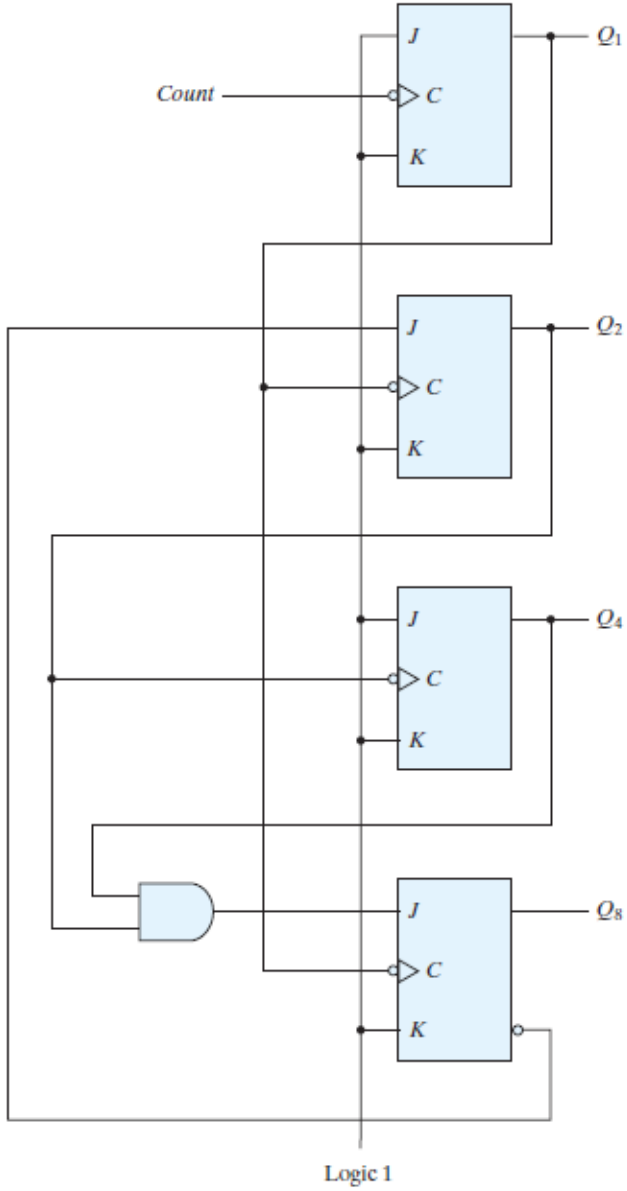
Fig. 6-9 State Diagram of a Decimal BCD-Counter

State Table

Present State	Next State
$Q_8\ Q_4\ Q_2\ Q_1$	$Q_8\ Q_4\ Q_2\ Q_1$
0000	0001
0001	0010
0010	0011
0011	0100
0100	0101
0101	0110
0110	0111
0111	1000
1000	1001
1001	0000

Asynchronous, BCD Counter

- Q_1 toggled by count (clock)
- Q_2 toggled by Q_1 except when Q_8 is 1
  - Q_8' tied to J , Q_2 remains at 0 when Q_8 is 1
- Q_4 toggled by Q_2
- Q_8 clocked by Q_1
  - When Q_4Q_2 equals 11, Q_8 complements at negative transition on Q_1
  - On the next negative transition on Q_1, Q_8 is cleared to 0.



Decimal (BCD) Counter, 3 digits (up to 999)

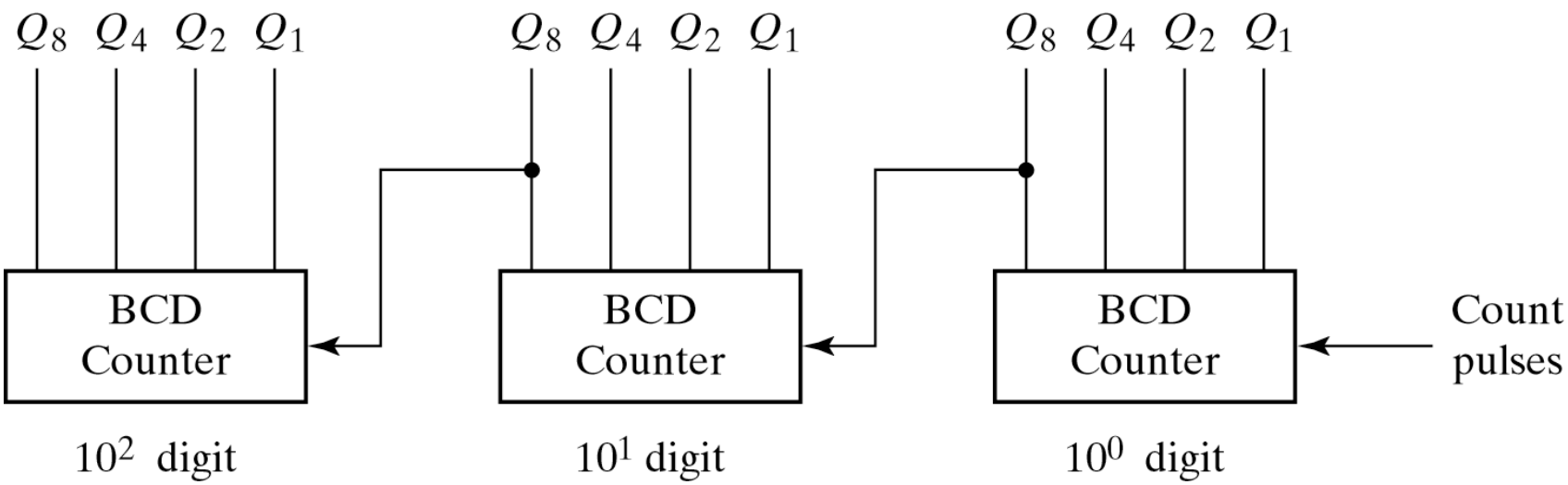


Fig. 6-11 Block Diagram of a Three-Decade Decimal BCD Counter

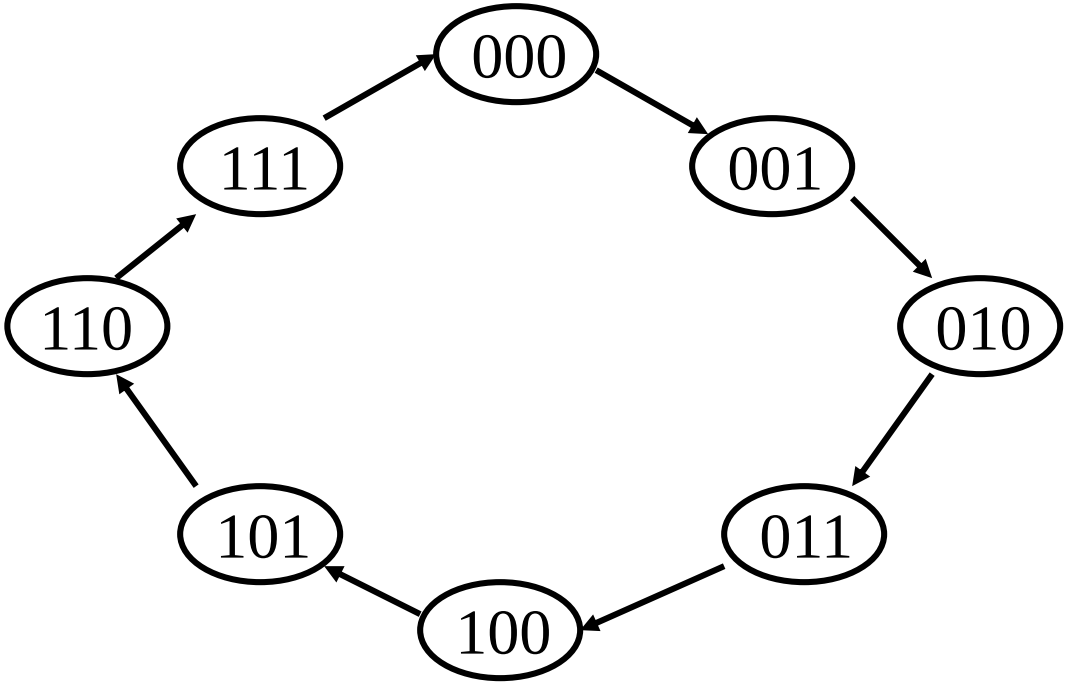
- Examine state table and explain how Q_8 can be used as count signal for next digit?

Binary Counters

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

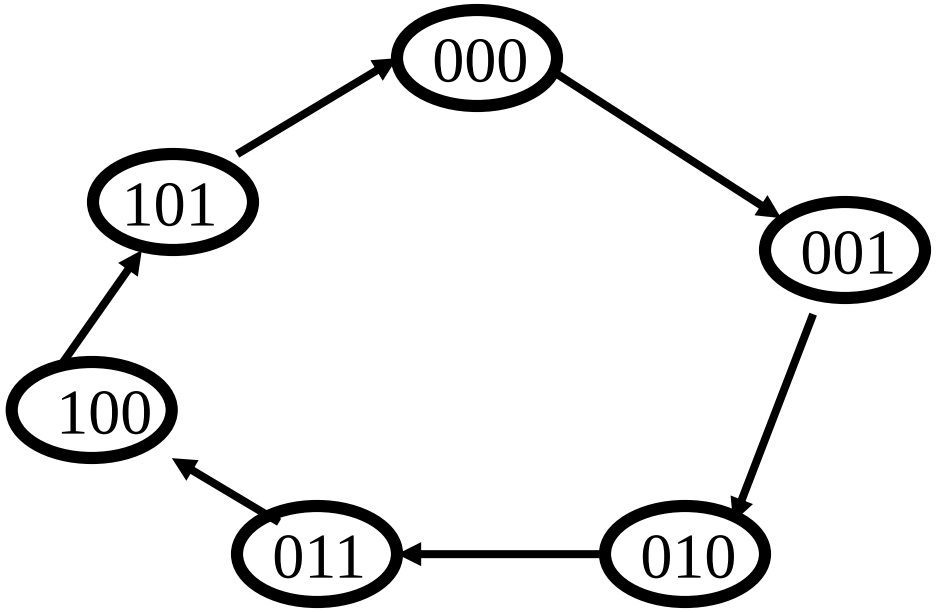
Counters are normally designed to recycle or restart the sequence.

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*

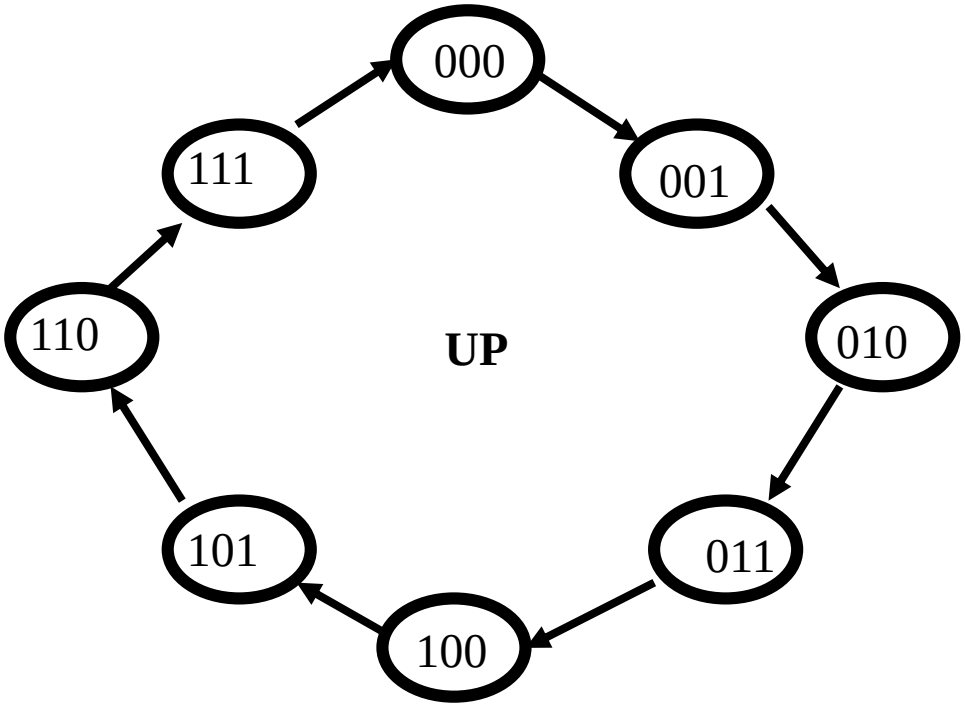
refers to a natural count that includes all possible binary numbers. It's modulus is the same as its maximum modulus.

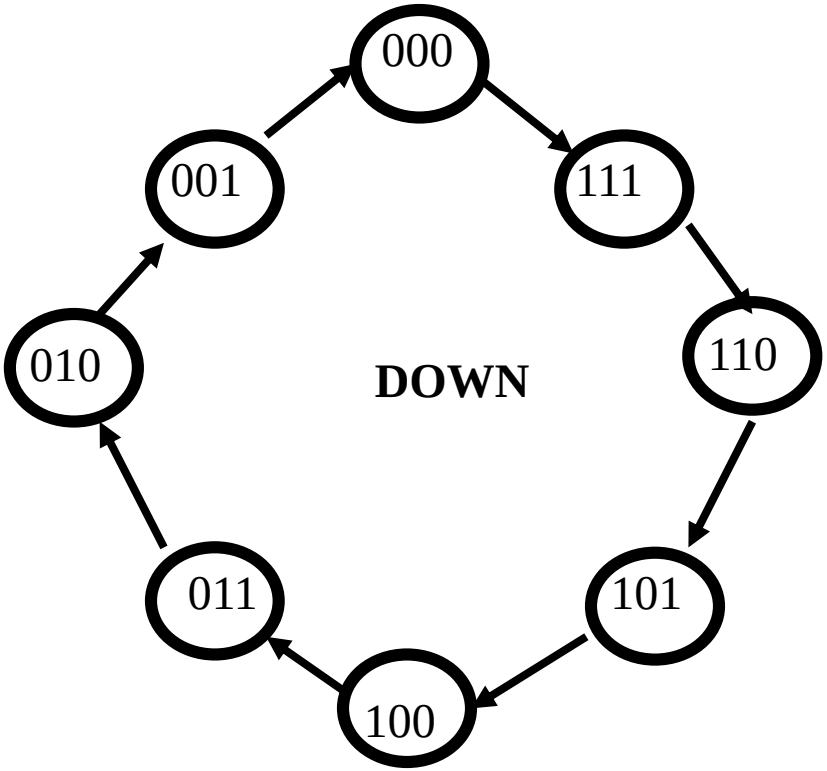
→A Truncated Sequence Count

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

Binary Counters: Up/Down

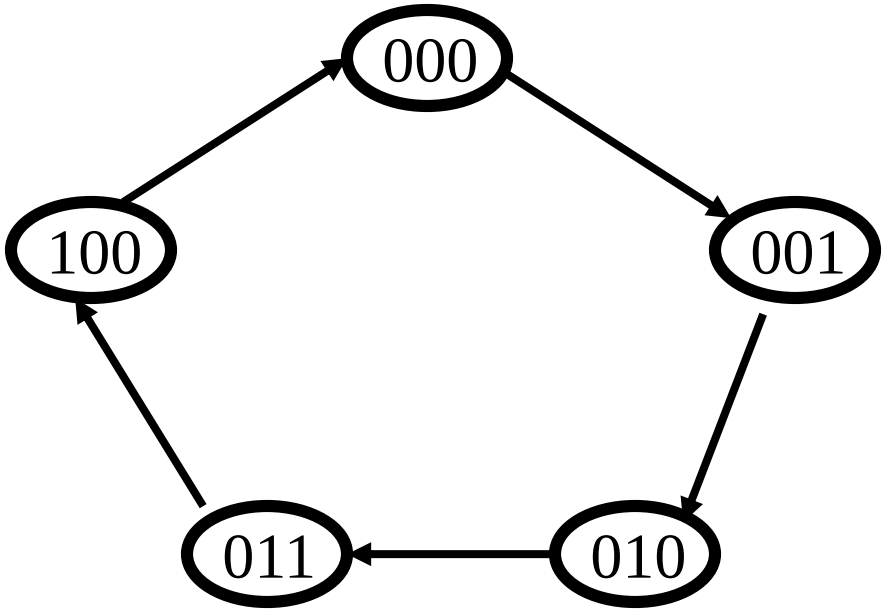
A sequential count refers to a natural numerical count.The sequence can be Up or Down:





Counter State Diagram

- Counter states are sequential, where an output state of flip-flops 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 of flip-flops based on the present output state of flip-flops.

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. Excitation (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 of flip-flops from outputs of flip-flops)
4. Use K-Maps to record and simplify the available outputs to create the inputs.
5. Implement the Circuit (i.e. Diagram)

Excitation (Transition) Tables

- Presented in Chapter 5

- 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.**

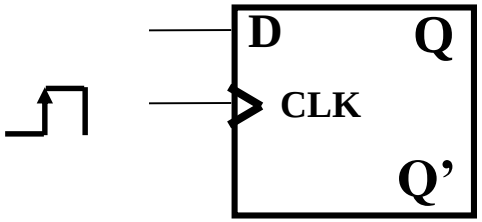
Building an Excitation (Transition) Table

- Building an Excitation Table is done using the Characteristic Function Tables of the Flip-Flops.

- Where the Characteristic Function Table indicates “Q”, “Q’” or “No Change”, we indicate the binary value.

D Flip flop

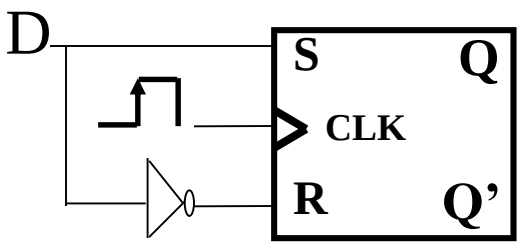
- Output changes only on the clock transition



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'

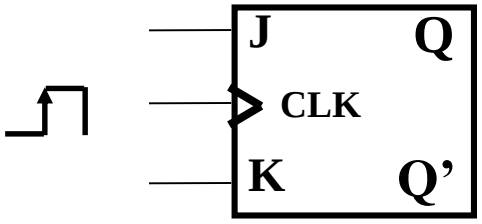
Excitation (Transition) Table: D Flip Flop

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 Characteristic Table

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 $Q_{n+1} = 1$

JK Excitation (Transition) Table

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 Excitation (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

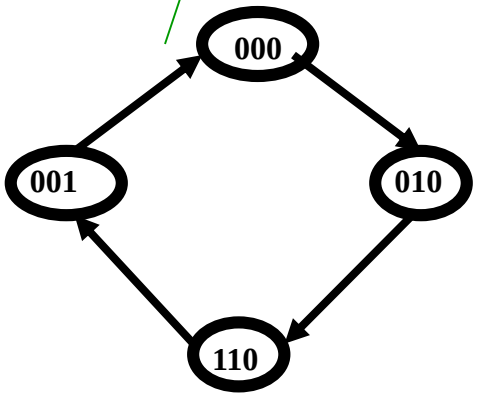
Toggle or Reset

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- Excitation (Transition) Table for the Flip-Flop we intend to use (here D ff)
    - Present and Next outputs
    - Necessary inputs to create Next outputs

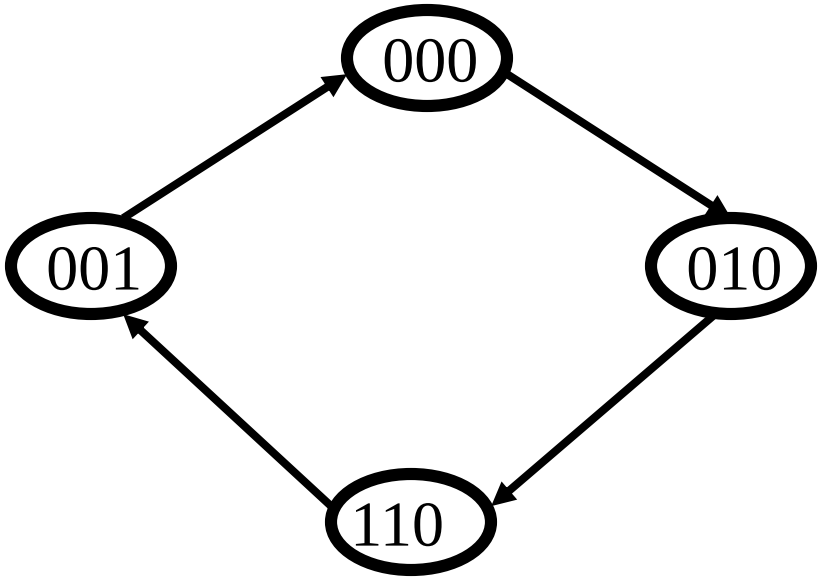


Present	Next
$Q_C Q_B Q_A$	$Q_C Q_B Q_A$
0 0 0	0 1 0
0 1 0	1 1 0
1 1 0	0 0 1
0 0 1	0 0 0

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

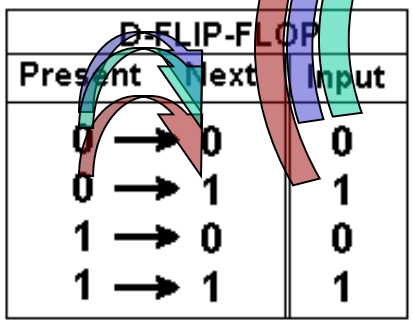
Transition Table

Building the K-Map: Step 3

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

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

Present	Next	FF Inputs
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			FF 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 FF Input:

Present			Next			FF 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

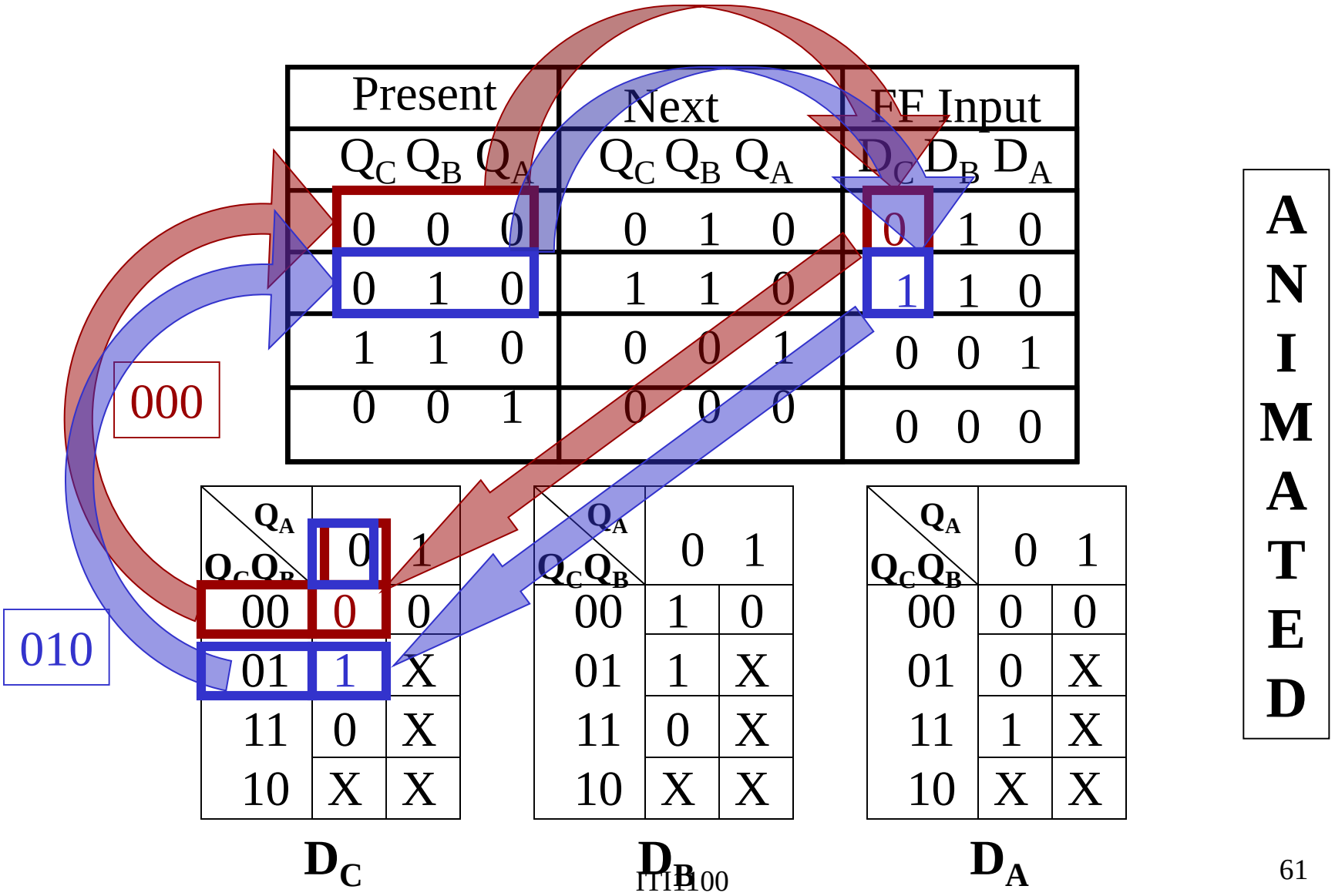
D_C

$Q_C Q_B$ \ 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
FF 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 FF 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

$D_B = Q'_C Q'_A$

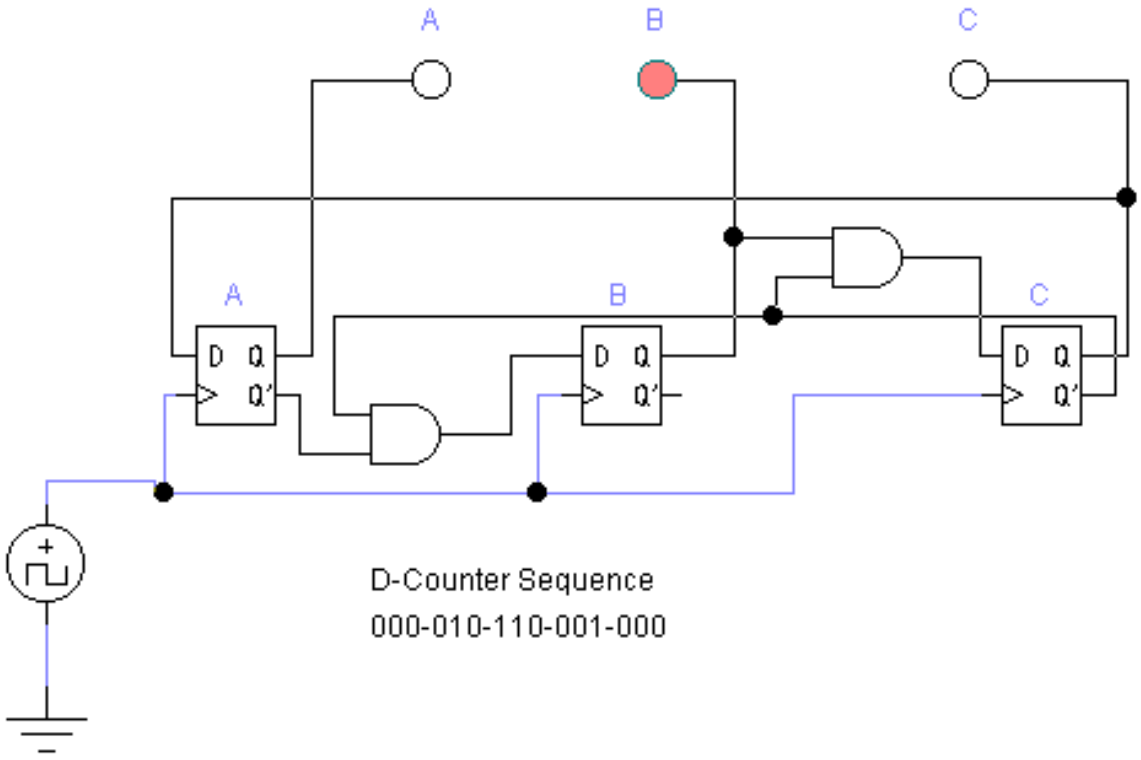
$Q_C Q_B$ \ Q_A	0	1
00	0	0
01	0	X
11	1	X
10	X	X

D_A

$D_A = Q_C$

Implementation Diagram

- Draw the Circuit Diagram
- Verify its operations.



Synchronous Binary Counter Examples

- Examples of binary counter with JK flip Flop

  1- Without missing States (full sequence)

  2- With missing States (truncated sequence)

Binary Counter with JK Flip-Flops

Excitation (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

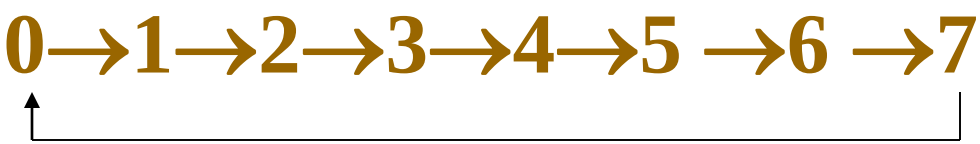
(b) Excitation (Transition) table for JK flip-flops

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

Full Sequence Binary Counter with JK Flip-Flops

- 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 to get JK FF input equations
- Build the circuit for the counter

Full Sequence Binary Counter with JK Flip-Flops



- 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

Full Sequence Binary Counter with JK Flip-Flops

State Table for The Binary Counter Example

Present State	Next State	JK FF Inputs					
ABC	ABC	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

Full Sequence Binary Counter with JK Flip-Flops

Use K-maps to simplify expressions for JK

$J_A = BC$

A \ BC	00	01	11	10
0	m_0	m_1	m_3 1	m_2
1	m_4 x	m_5 x	m_7 x	m_6

$K_A = BC$

A \ BC	00	01	11	10
0	m_0 x	m_1 x	m_3 x	m_2 x
1	m_4	m_5	m_7 1	m_6

$J_B = C$

A \ BC	00	01	11	10
0	m_0	m_1 1	m_3 x	m_2 x
1	m_4	m_5 1	m_7 x	m_6 x

$K_B = C$

A \ BC	00	01	11	10
0	m_0 x	m_1 x	m_3 1	m_2
1	m_4 x	m_5 x	m_7 1	m_6

$J_C = 1$

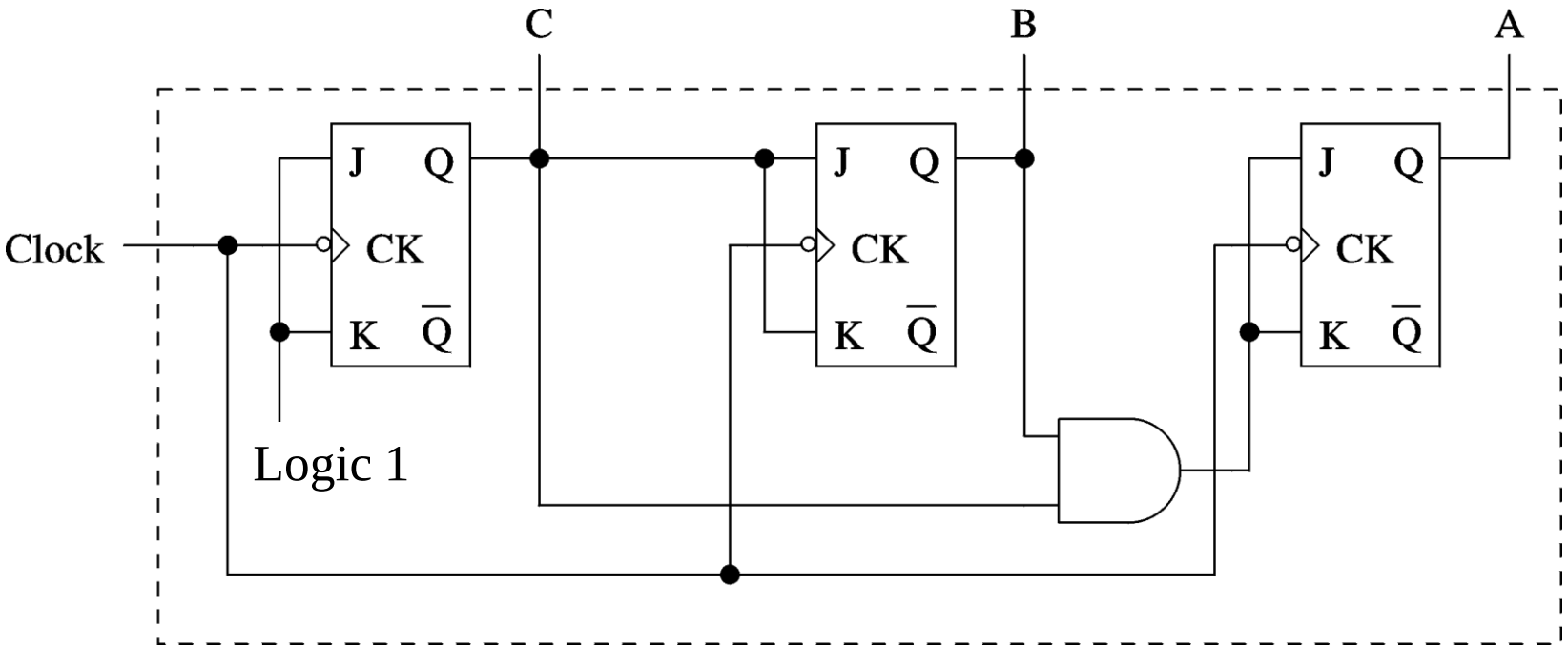
A \ BC	00	01	11	10
0	m_0 1	m_1 x	m_3 x	m_2 1
1	m_4 1	m_5 x	m_7 x	m_6 1

$K_C = 1$

A \ BC	00	01	11	10
0	m_0 x	m_1 1	m_3 1	m_2 x
1	m_4 x	m_5 1	m_7 1	m_6 x

Binary counter with JK Flip-Flops

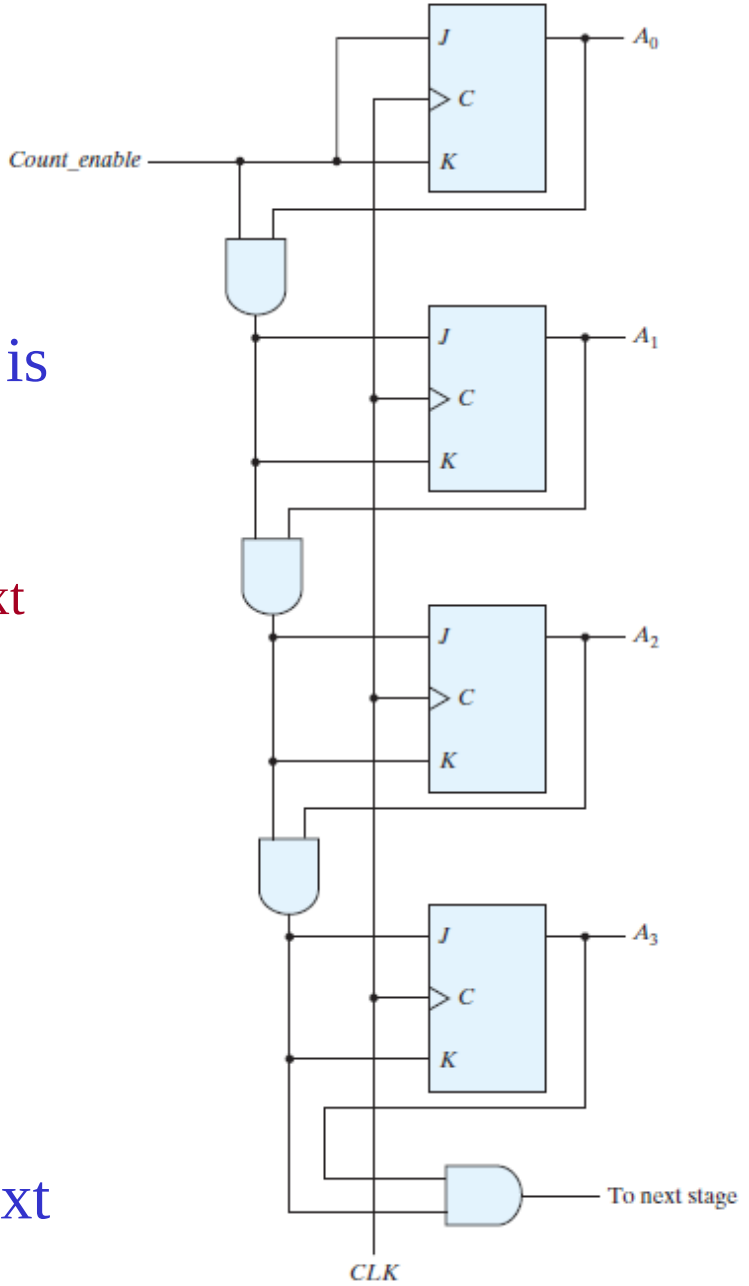
- Final circuit for the binary counter



Does it matter if +ve or –ve transition clock used?

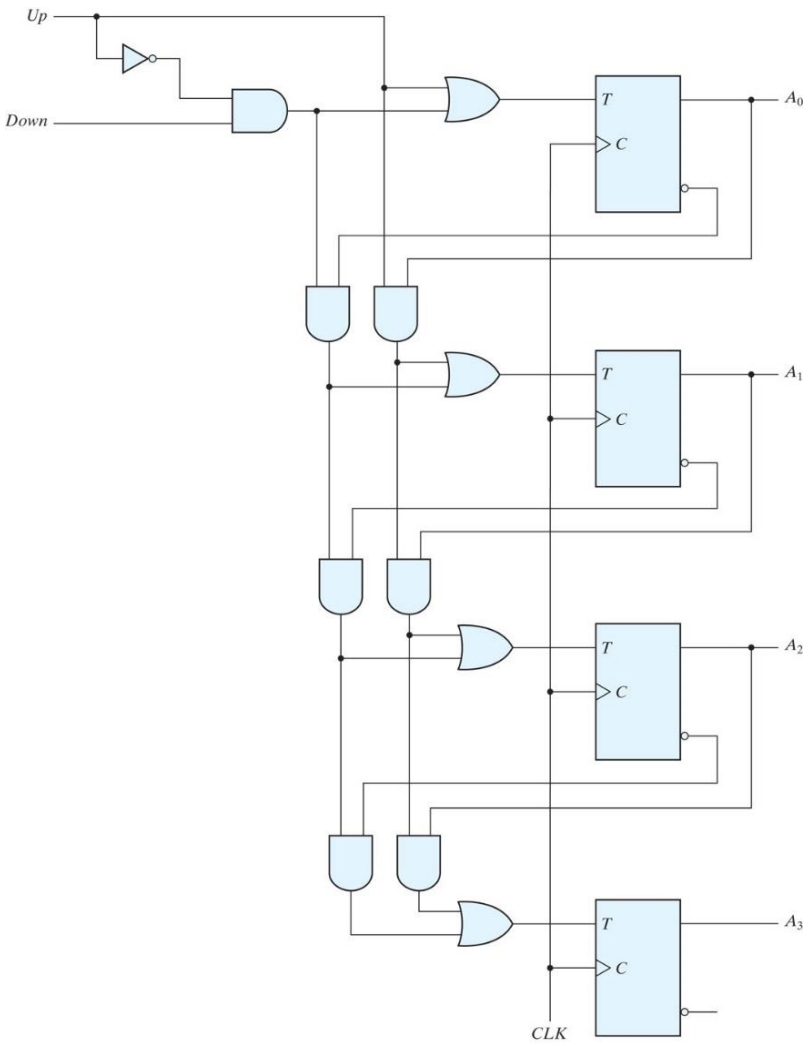
4- bit Synchronous Binary Counter with Enable Input

- Pattern for count UP binary counter: The LSB flip-flop is complemented with every pulse. Any other flip-flop is complemented when all bits in lower significant positions are 1
  - present state $A_3A_2A_1A_0 = 0011$, the next state is 0100
- Note the use of Count_enable input
  - Count_enable = 0, all JK inputs =0, and counter state unchanged
  - Count_enable = 1, all JK flip-flops are enabled, and counter is enabled
- Cascading consists of attaching FF output of MSB to count_enable of next stage.



Up-Down Binary Counter

- Countdown binary counter counts in reverse order, 1111 to 0000, back to 1111.
- Pattern for DOWN binary counter: The LSB flip-flop is complemented with every pulse. Any other flip-flop is complemented when all bits in lower significant positions are 0
  - present state $A_3A_2A_1A_0 =$ 0100, the next state is 0011
- Note the use of up and down control inputs
  - When up=1, counter counts up
  - When up=0, down=1, counter counts down

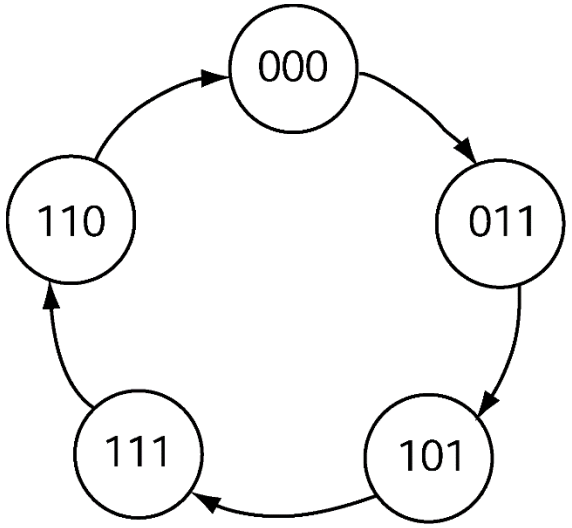


Binary Counter with JK FF's (missing states)

→ 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)

State Table For The Binary Counter Example

Present State	Next State	JK FF Inputs					
ABC	ABC	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	x	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)

Use K-maps to simplify expressions for JK inputs

$J_A = B$

A \ BC	00	01	11	10
0	m_0	m_1 x	m_3 1	m_2 x
1	m_4 x	m_5 x	m_7 x	m_6 x

$K_A = C'$

A \ BC	00	01	11	10
0	m_0 x	m_1 x	m_3 x	m_2 x
1	m_4 x	m_5	m_7	m_6 1

$J_B = 1$

A \ BC	00	01	11	10
0	m_0 1	m_1 x	m_3 x	m_2 x
1	m_4 x	m_5 1	m_7 x	m_6 x

$K_B = A' + C'$

A \ BC	00	01	11	10
0	m_0 x	m_1 x	m_3 1	m_2 x
1	m_4 x	m_5 x	m_7	m_6 1

$J_C = A'$

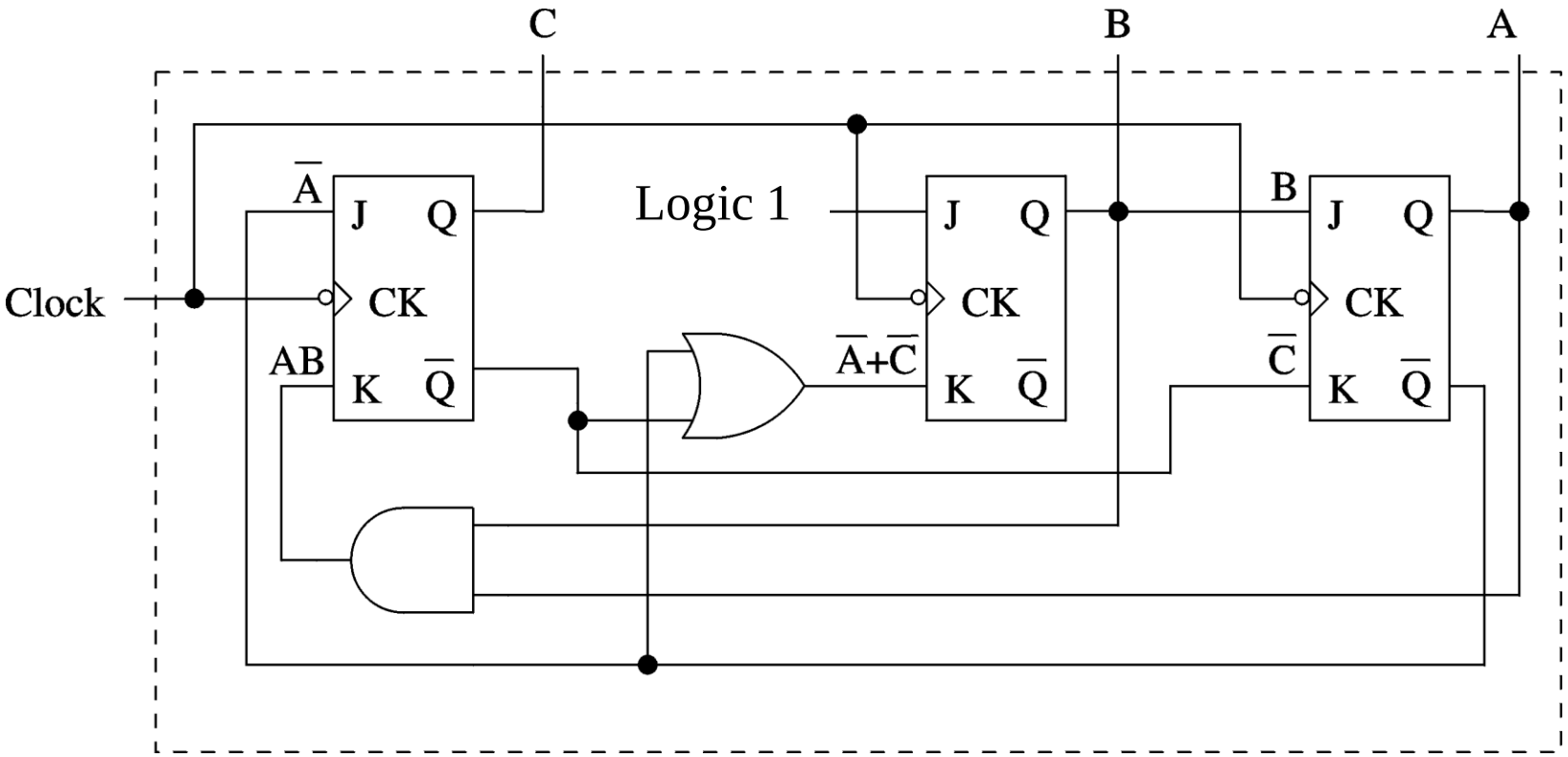
A \ BC	00	01	11	10
0	m_0 1	m_1 x	m_3 x	m_2 x
1	m_4 x	m_5 x	m_7 x	m_6

$K_C = AB$

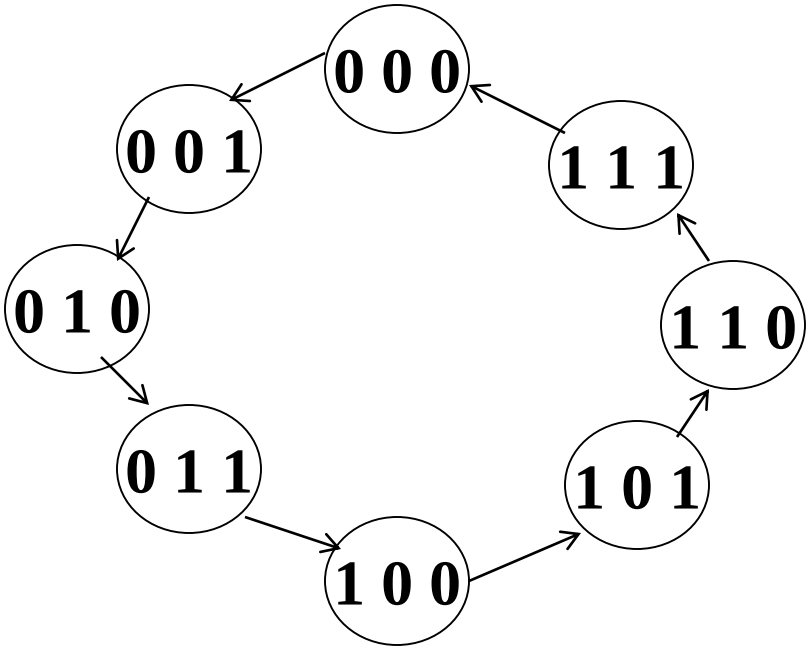
A \ BC	00	01	11	10
0	m_0 x	m_1 x	m_3	m_2 x
1	m_4 x	m_5	m_7 1	m_6 x

Binary counter with JK flip flop (missing states)

Final circuit



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

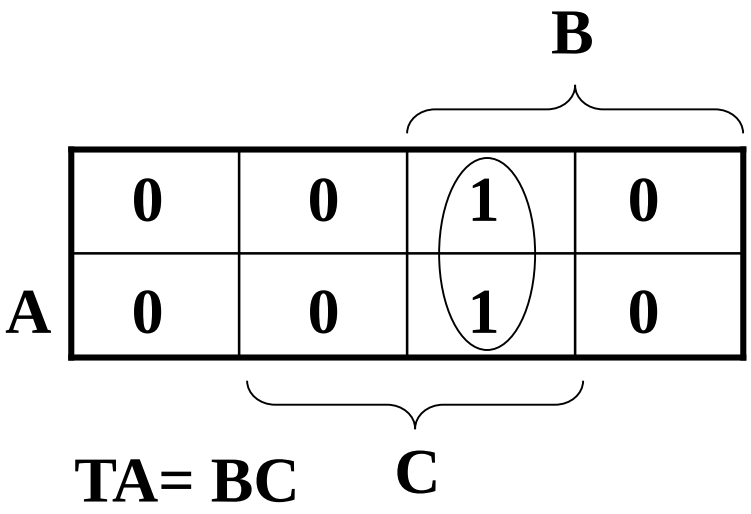


Example: - Design a 3 - bit binary counter using T flip - flops.

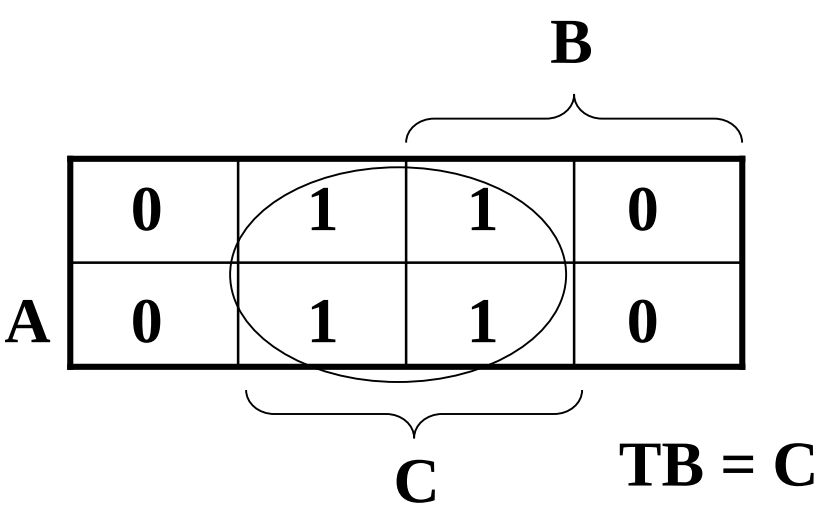
Present State A B C	Next State A B C	Flip - Flop Inputs T A	T B	T C
0 0 0	0 0 1	0	0	1
0 0 1	0 1 0	0	1	1
0 1 0	0 1 1	0	0	1
0 1 1	1 0 0	1	1	1
1 0 0	1 0 1	0	0	1
1 0 1	1 1 0	0	1	1
1 1 0	1 1 1	0	0	1
1 1 1	0 0 0	1	1	1

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

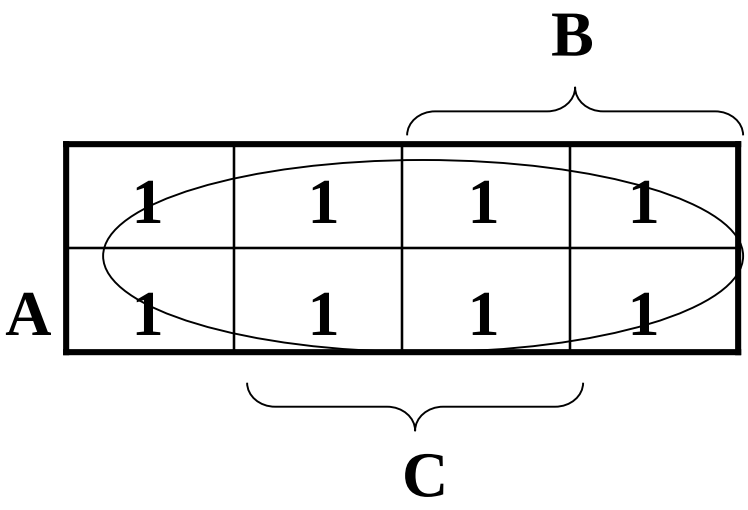
- Kmap



TA= BC

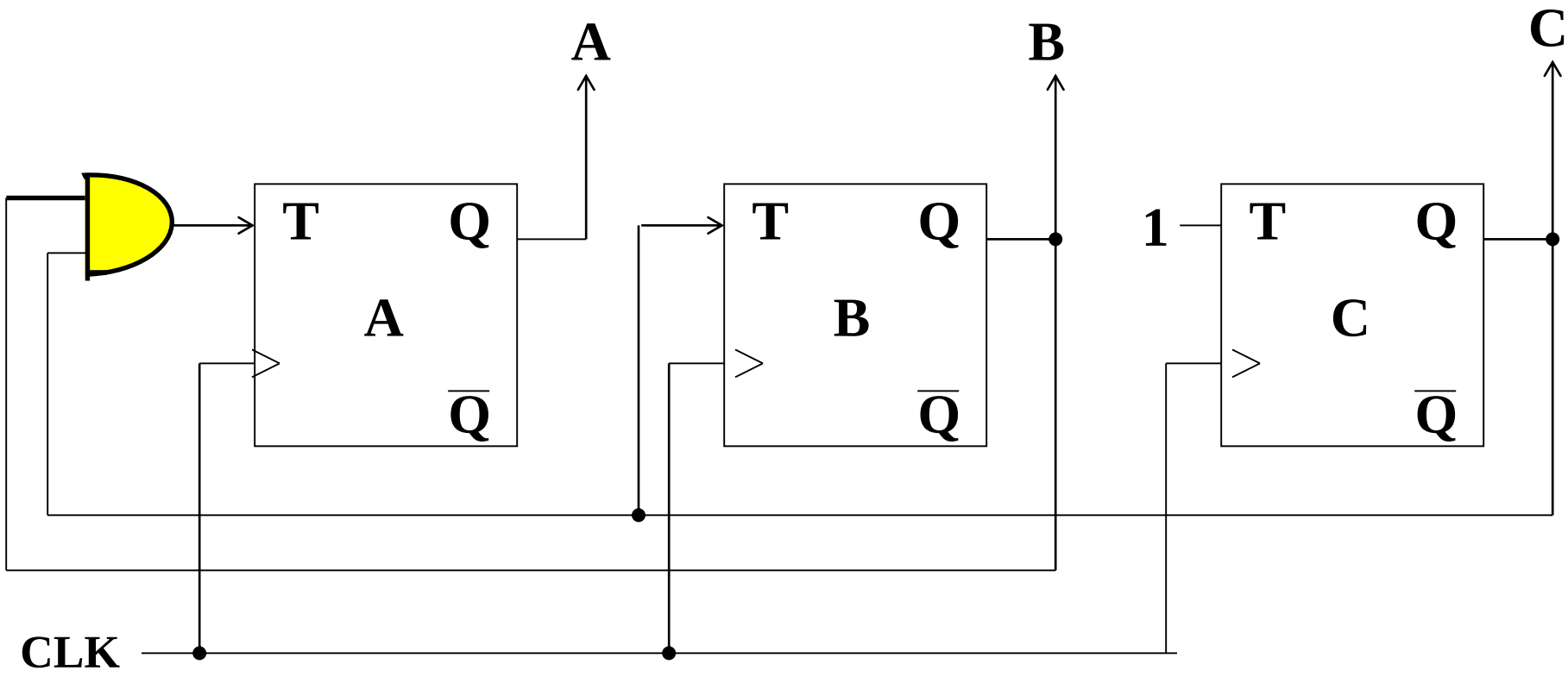


TB = C



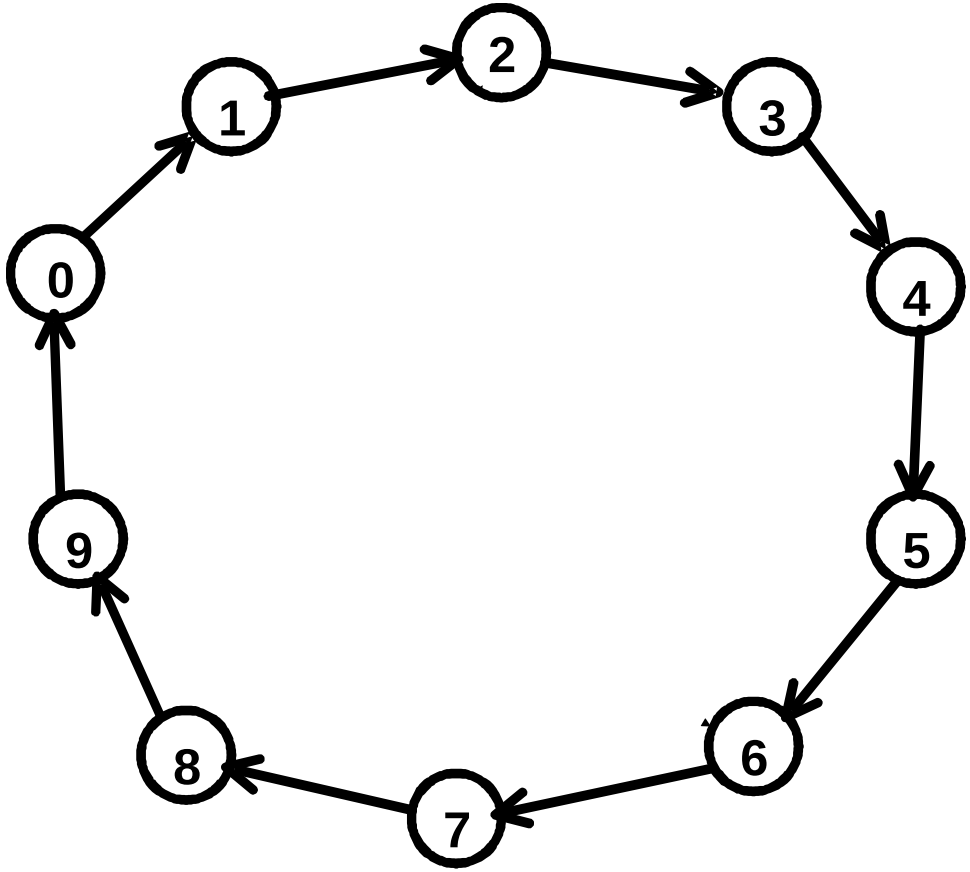
TC = 1

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

- We can use the sequential logic model to design a synchronous BCD counter with T FF's. Below is the State Table.

Don't care states have been left out (from 1010 to 1111).

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 FF input functions

$T_8 = Q_8Q_1 + Q_4Q_2Q_1$

Q_8Q_4 \ Q_2Q_1	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7 1	m_6
11	m_{12} x	m_{13} x	m_{15} x	m_{14} x
10	m_8	m_9 1	m_{11} x	m_{10} x

$T_4 = Q_2Q_1$

Q_8Q_4 \ Q_2Q_1	00	01	11	10
00	m_0	m_1	m_3 1	m_2
01	m_4	m_5	m_7 1	m_6
11	m_{12} x	m_{13} x	m_{15} x	m_{14} x
10	m_8	m_9	m_{11} x	m_{10} x

$T_2 = Q_8'Q_1$

Q_8Q_4 \ Q_2Q_1	00	01	11	10
00	m_0	m_1 1	m_3 1	m_2
01	m_4	m_5 1	m_7 1	m_6
11	m_{12} x	m_{13} x	m_{15} x	m_{14} x
10	m_8	m_9	m_{11} x	m_{10} x

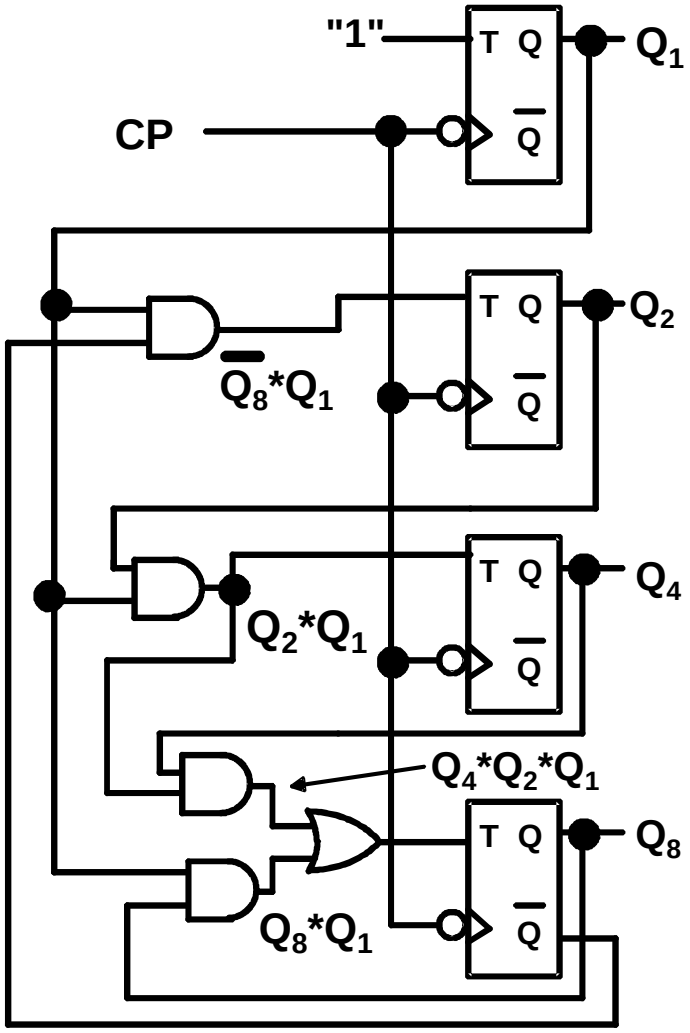
$T_1 = 1$

Q_8Q_4 \ Q_2Q_1	00	01	11	10
00	m_0 1	m_1 1	m_3 1	m_2 1
01	m_4 1	m_5 1	m_7 1	m_6 1
11	m_{12} x	m_{13} x	m_{15} x	m_{14} x
10	m_8 1	m_9 1	m_{11} x	m_{10} x

Note: <u>Don't Care</u> states are included.

Synchronous BCD (Continued)

- **The minimized circuit:**



Designing a Synchronous BCD Counter

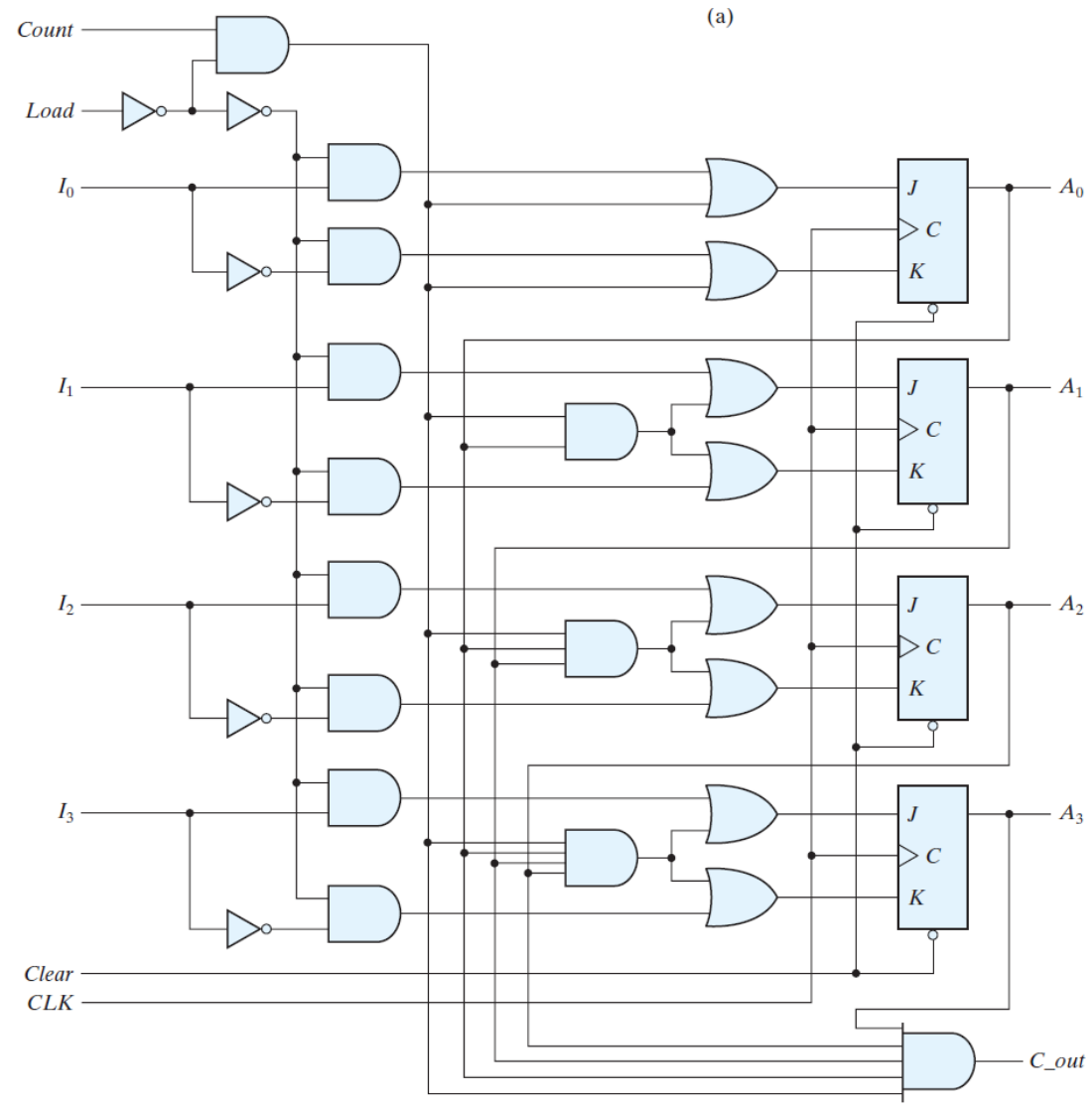
Present State				Next State				Output	Next State			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	1	0	0	1	1	0	0	0	0	1	1
0	1	0	1	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0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

Can serve as the count enable of next stage.

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

,

Counter with Parallel load



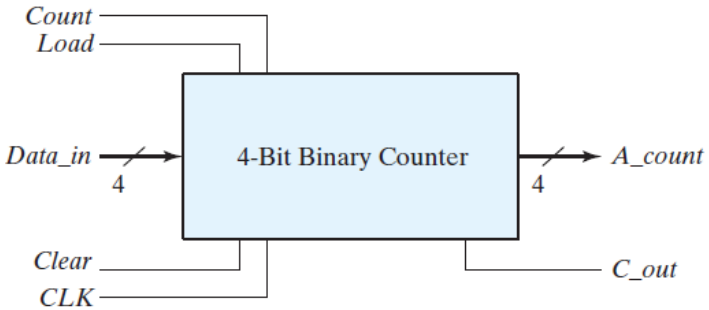


Table 6.6
Function Table for the Counter of Fig. 6.14

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

Counter with Parallel load: FF Input Equations

$$J_0 = LI_0 + L'C \qquad J_1 = LI_1 + L'CA_0 \qquad J_2 = LI_2 + L'CA_0A_1 \qquad J_3 = LI_3 + L'CA_0A_1A_2$$
$$K_0 = LI_0' + L'C \qquad K_1 = LI_1' + L'CA_0 \qquad K_2 = LI_2' + L'CA_0A_1 \qquad K_3 = LI_3' + L'CA_0A_1A_2$$
$$C_{out} = L'CA_0A_1A_2A_3$$

- With L = 1, C = X, parallel load

$$J_0 = I_0 \qquad J_1 = I_1 \qquad J_2 = I_2 \qquad J_3 = I_3$$
$$K_0 = I_0' \qquad K_1 = I_1' \qquad K_2 = I_2' \qquad K_3 = I_3'$$
$$C_{out} = 0$$

- With L = 0, C = 1, count

$$J_0 = 1 \qquad J_1 = A_0 \qquad J_2 = A_0A_1 \qquad J_3 = A_0A_1A_2$$
$$K_0 = 1 \qquad K_1 = A_0 \qquad K_2 = A_0A_1 \qquad K_3 = A_0A_1A_2$$
$$C_{out} = A_0A_1A_2A_3$$

- With L = 0, C = 0, no change, disabled

$$J_0 = 0 \qquad J_1 = 0 \qquad J_2 = 0 \qquad J_3 = 0$$
$$K_0 = 0 \qquad K_1 = 0 \qquad K_2 = 0 \qquad K_3 = 0$$
$$C_{out} = 0$$

- Note: L = Load, C = Count

More BCD Counters (with Parallel Load)

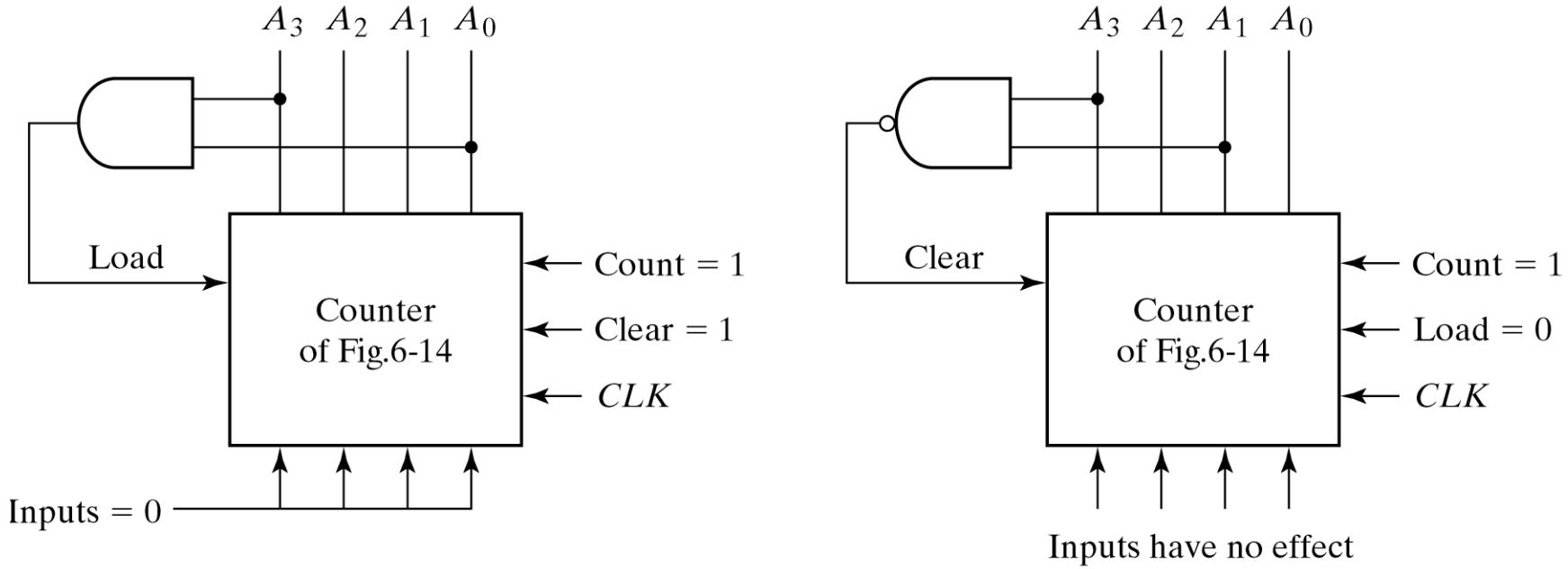


Fig. 6-15 Two ways to Achieve a BCD Counter Using a Counter with Parallel Load

When A_3A_0=1 (i.e. at value 1001), then load in 0000 at next clock transition

When A_3A_1=1 (i.e. at value 1010), Clear becomes low and counter clear to 0000 immediately

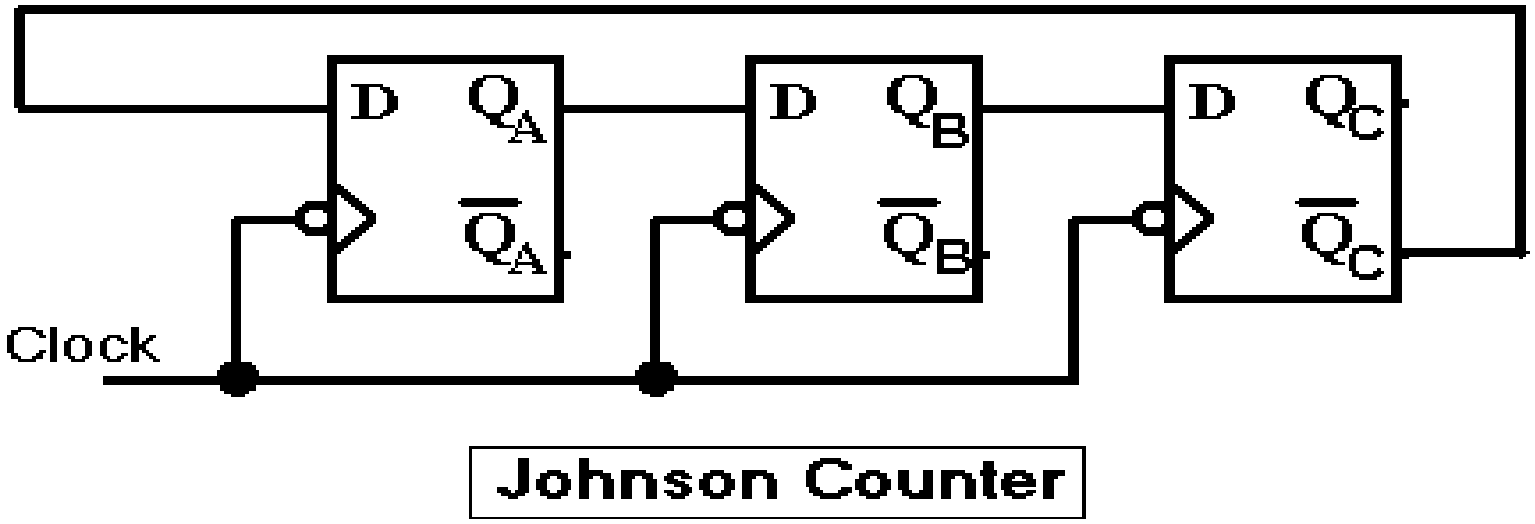
- Not recommended, why?

Johnson Counter

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

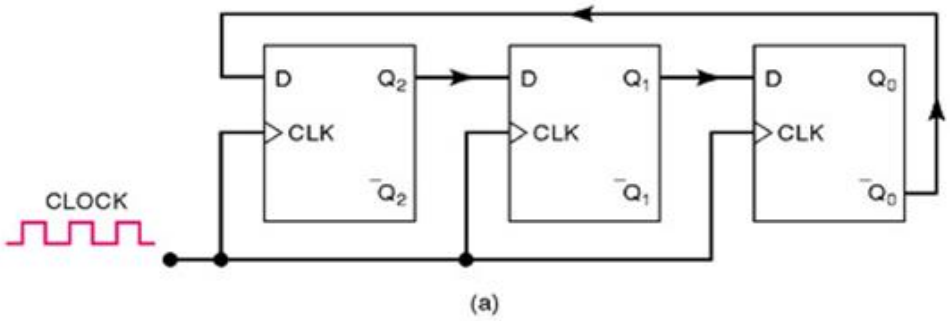
000, 100, 110,111,011,001, 000, 100, 110, ….

Initial state

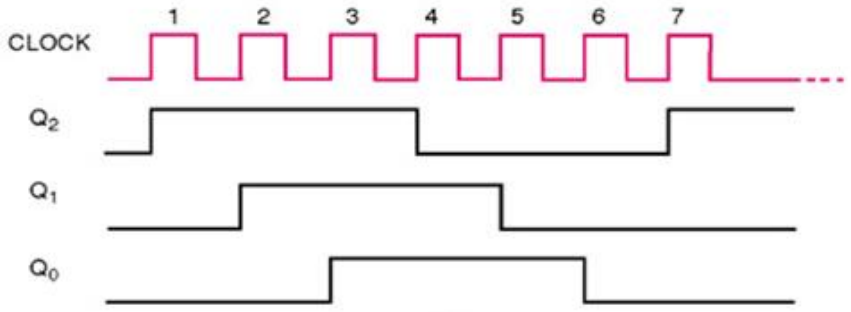


Johnson Counter

Johnson Counter



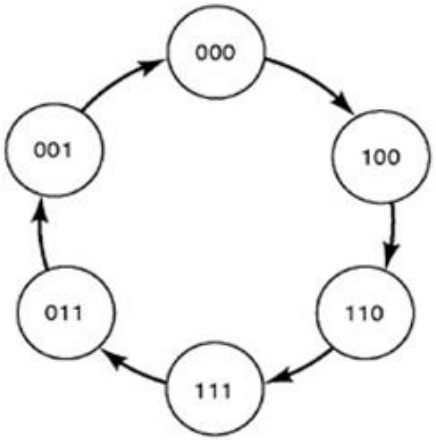
(a)



(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)

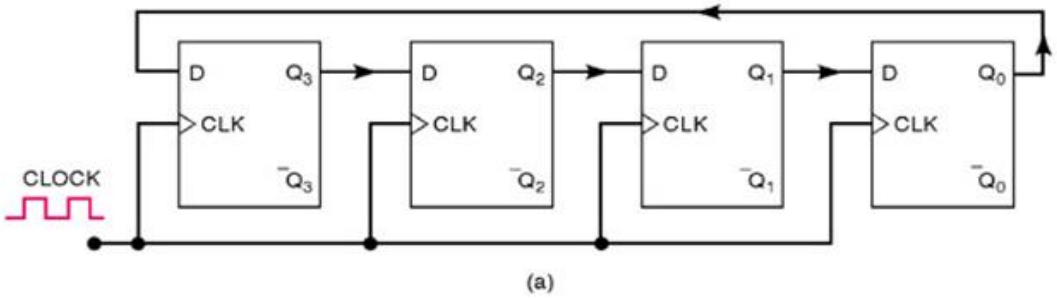
- The modulus of the counter is the number of FF output states
- The modulus of a k-bit Johnson counter is 2k

Ring Counter

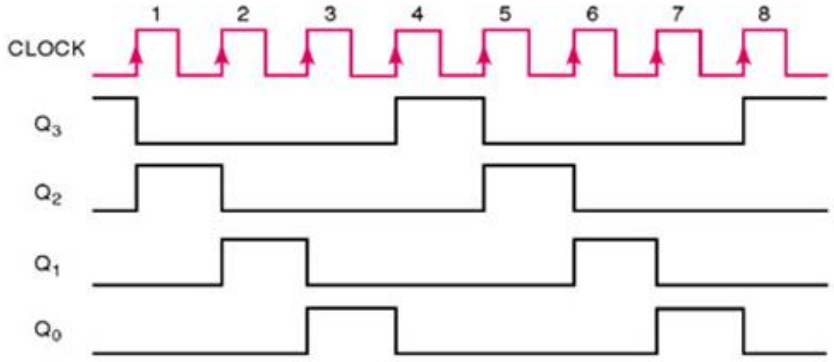
- A ring counter takes the serial output of the last Flip-Flop of a shift register and provides it to the serial input of the first Flip-Flop.

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

Ring Counter



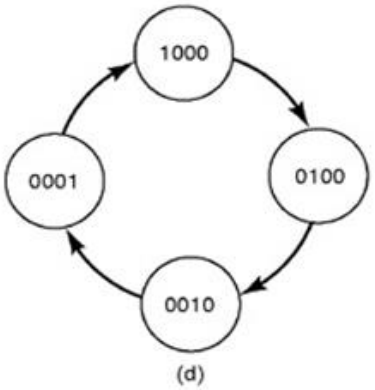
(a)



(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
.	.	.	.	.
.	.	.	.	.

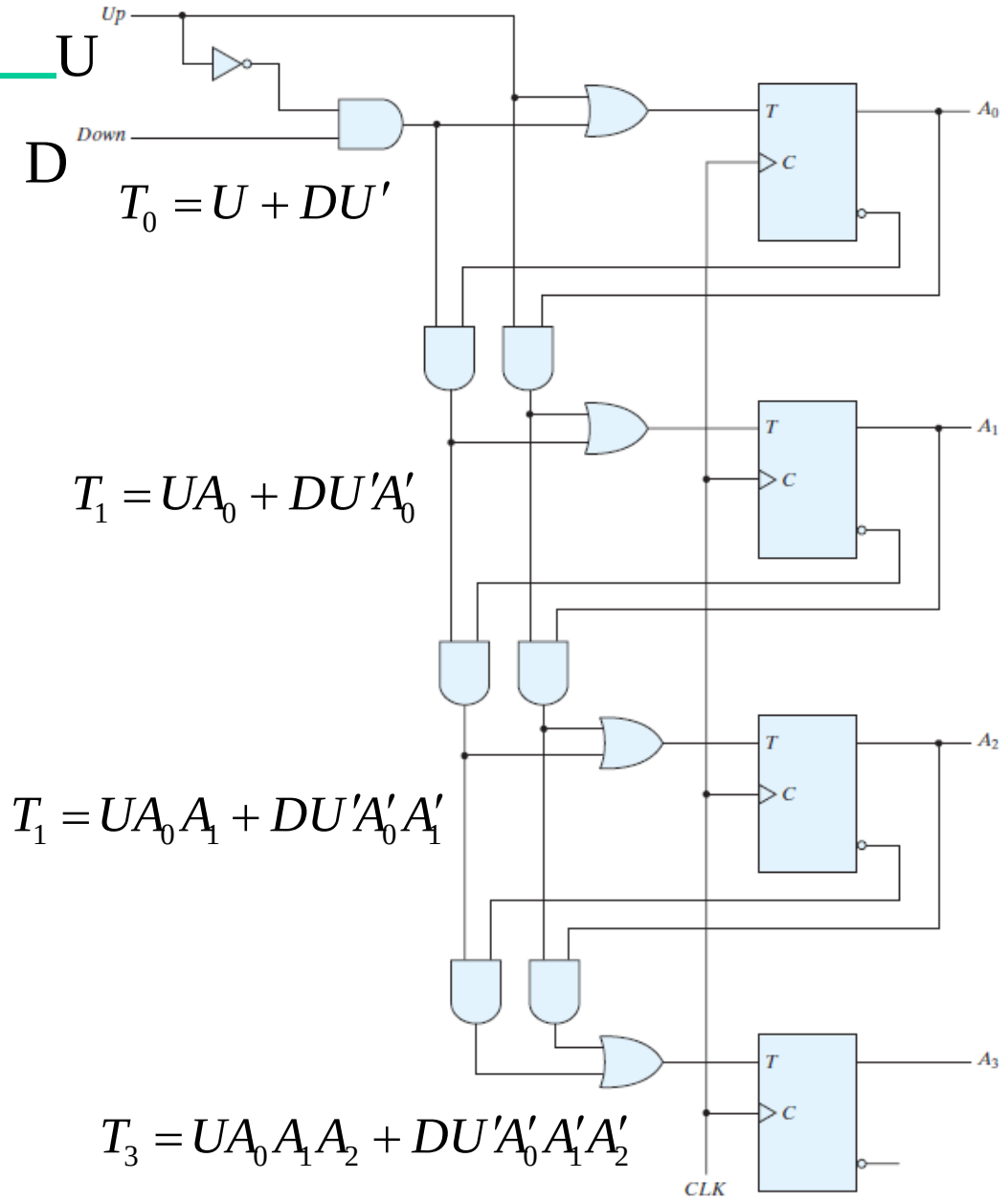
(c)



(d)

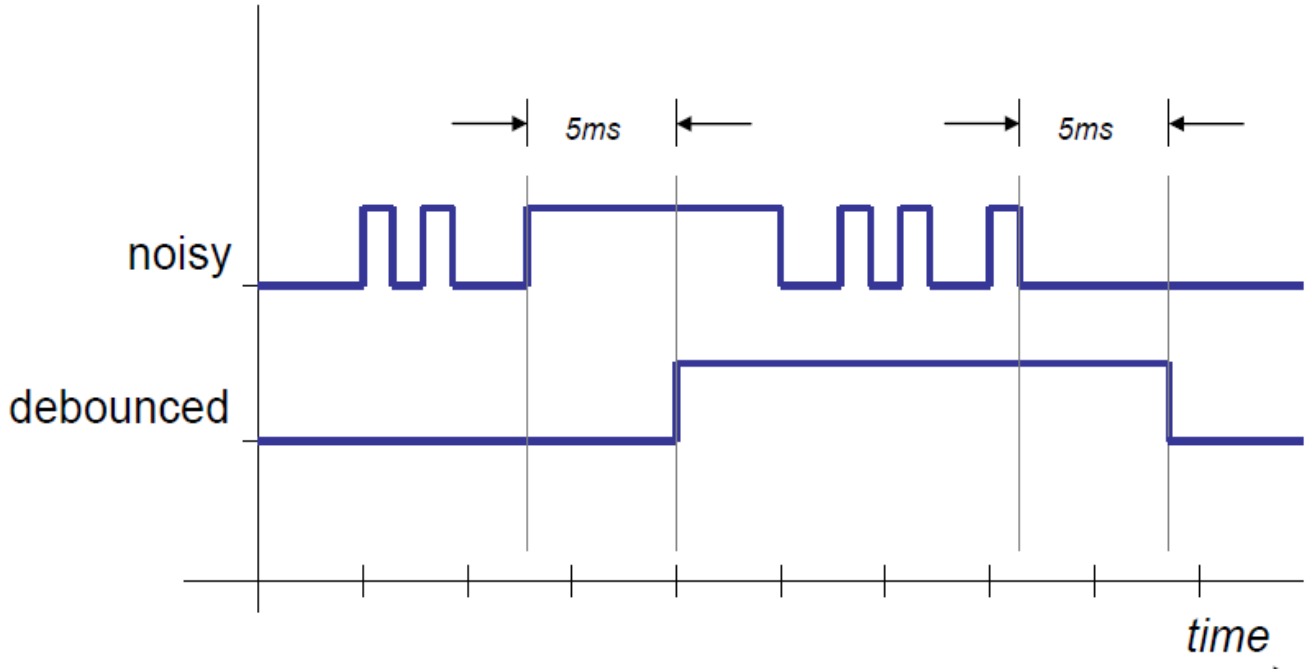
Up Down Counter

- U = 0, D = 0: $T_0 = 0$
  $T_1 = 0$
  $T_2 = 0$
  $T_3 = 0$
- U = 1, D = 0: $T_0 = 1$
  $T_1 = A_0$
  $T_2 = A_0 A_1$
  $T_3 = A_0 A_1 A_2$
- U = 0, D = 1: $T_0 = 1$
  $T_1 = A_0'$
  $T_2 = A_0' A_1'$
  $T_3 = A_0' A_1' A_2'$



Debouncer

- Recall that mechanical switch bounces when its position is changed (or push button pushed/released
- Debouncing consists of removing the bounces from the noisy signal as shown



Debouncer

- Since button when bouncing, FF1 and FF2 become different
  - Exclusive OR gate produces 1 and clears the counter
- When the bouncing has settled: FF1 = FF2 = button
  - Counter increases until C_{out} = 1, enables FF3 to capture button input
  - Select number of bits (N) such that count takes 10 ms (18 bits for a clock of 40 ns/25 MHz)

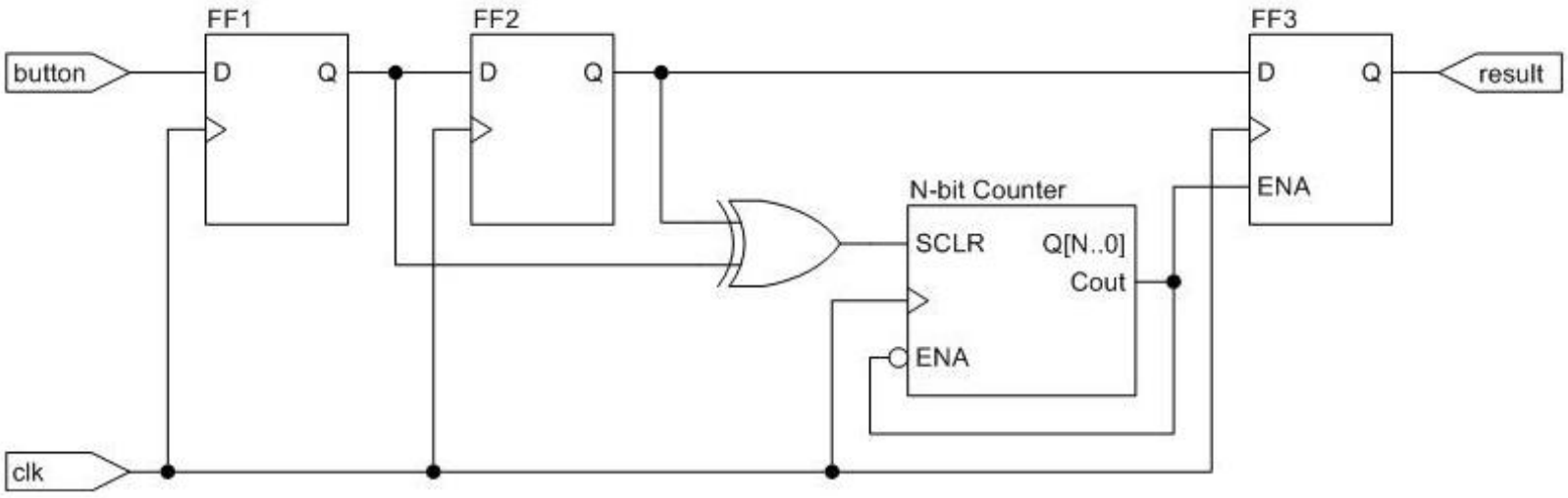
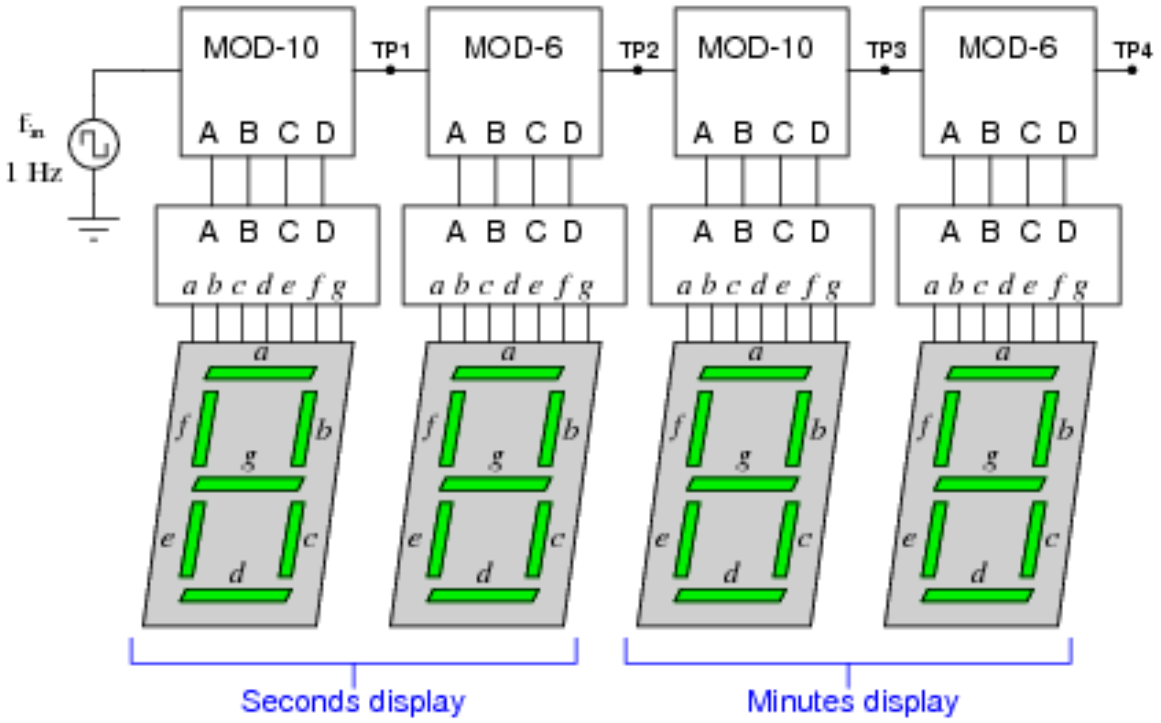


Figure 1. Debounce Circuit

Application of BCD counters



BCD Counters

7-segment decoders

7-segment displays (digits reversed)

Two counters are decade counters (Mod-10) while the other two have a modulus of 6. Why?

Sequential Logic Circuits: Examples

Note:

In addition to the examples presented here other examples were discussed in the class:

-Asynchronous counters (4 examples are discussed in the class)

-Synchronous counters (3 examples are 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)