CSE 219: DIGITAL LOGIC & COMPUTER DESIGN

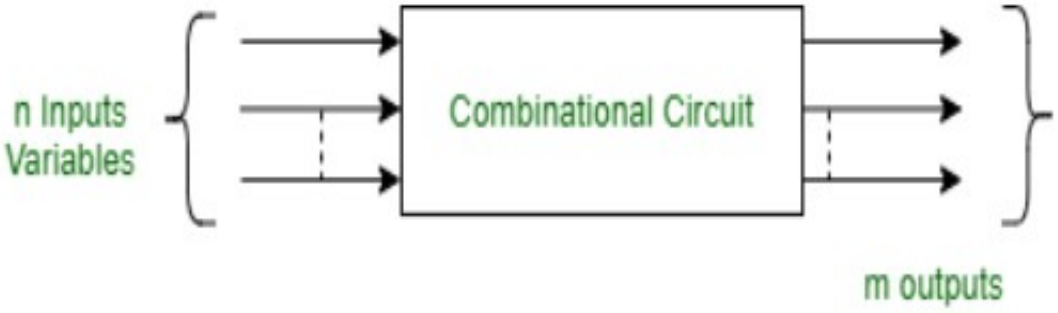
Tamanna Haque Nipa
Assistant Professor
Dept. of CSE
Stamford University Bangladesh

COMBINATIONAL AND SEQUENTIAL CIRCUITS

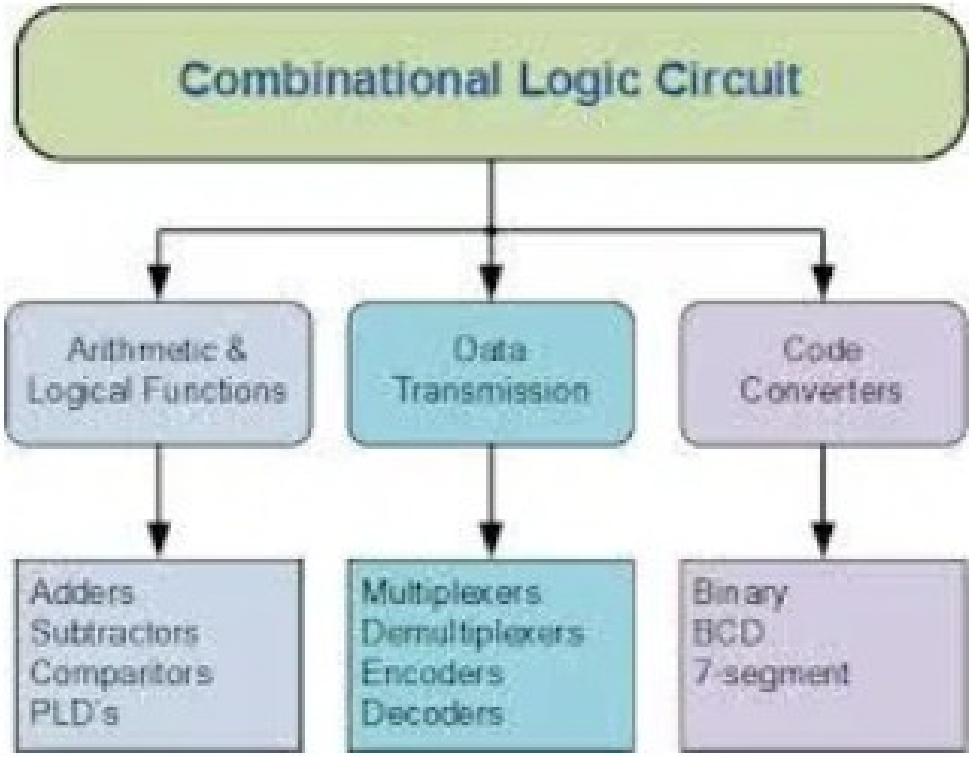
- **Types of Logic Circuits:**
  - **Combinational Circuits:** A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs and they have no memory.
  - **Sequential Circuits:** A sequential circuit consists of logic gates whose outputs at any time are determined from both the present combination of inputs and previous output. That means sequential circuits use memory elements to store the value of previous output.

COMBINATIONAL CIRCUITS

- A combinational circuit consists of input variables and output variables. Since these circuits are not dependent upon previous input to generate any output, so are combinational logic circuits. A combinational circuit can have an n number of inputs and m number of outputs.

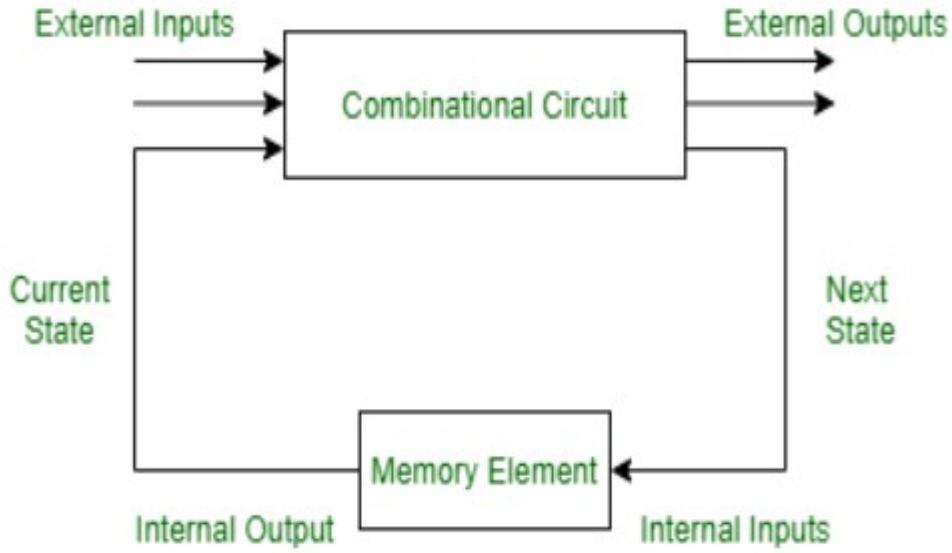


TYPE OF COMBINATIONAL CIRCUIT

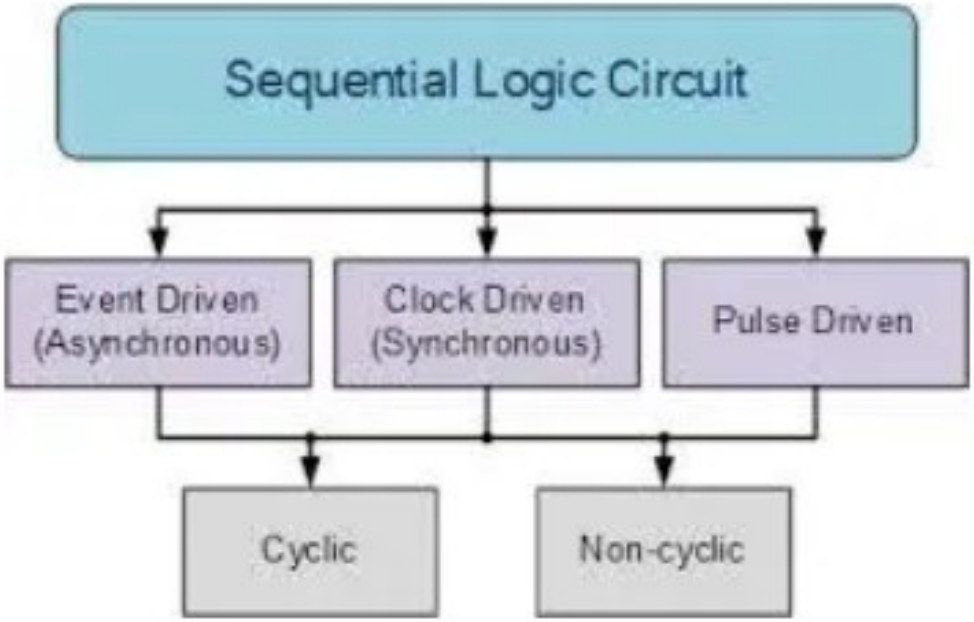


SEQUENTIAL CIRCUITS

- A sequential circuit is specified by a time sequence of inputs, outputs, and internal states. The output of a sequential circuit depends not only the combination of present inputs but also on the previous outputs. Unlike combinational circuits, sequential circuits include memory elements with combinational circuits.



TYPES OF SEQUENTIAL CIRCUIT

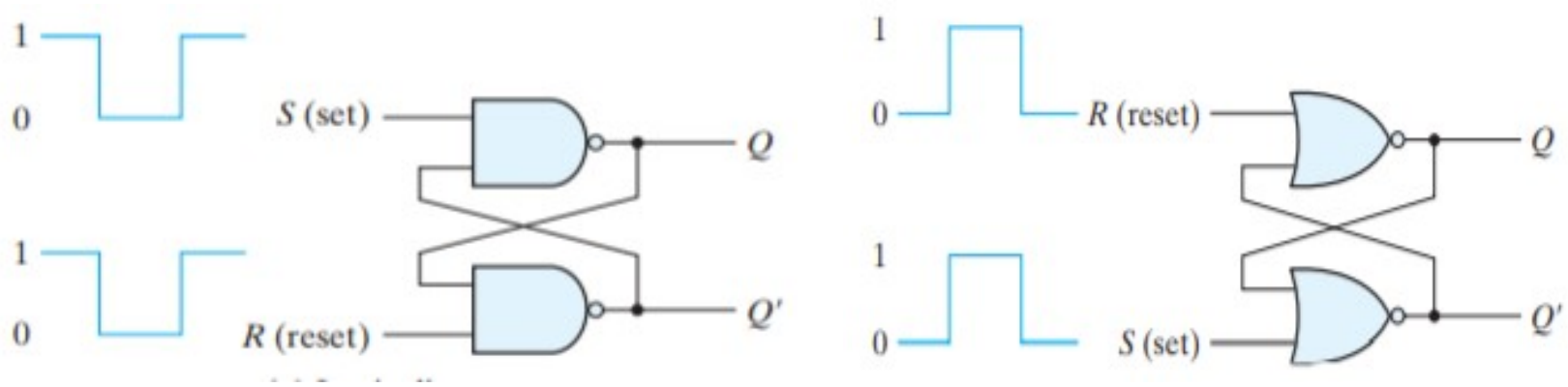


SYNCHRONOUS AND ASYNCHRONOUS CIRCUIT

SYNCHRONOUS CIRCUIT	ASYNCHRONOUS CIRCUIT
All the **State Variable** changes are synchronized with a universal clock signal.	The **State Variables** are not synchronized to change simulteneously and may change at anytime irrespective of each other to achieve the next **Steady Internal State**
Since all the Internal State changes are in the strict control of a master clock source they are less prone to failure or to a race condition and hence are more reliable.	Since there is no such universal clock source, the internal state changes as soon as any of the inputs change and hence are more prone to a race condition.
Timings of the internal state changes are in our control.	The changes in the internal state of an asynchronous circuit are not in our control.

STORAGE ELEMENTS: LATCHES

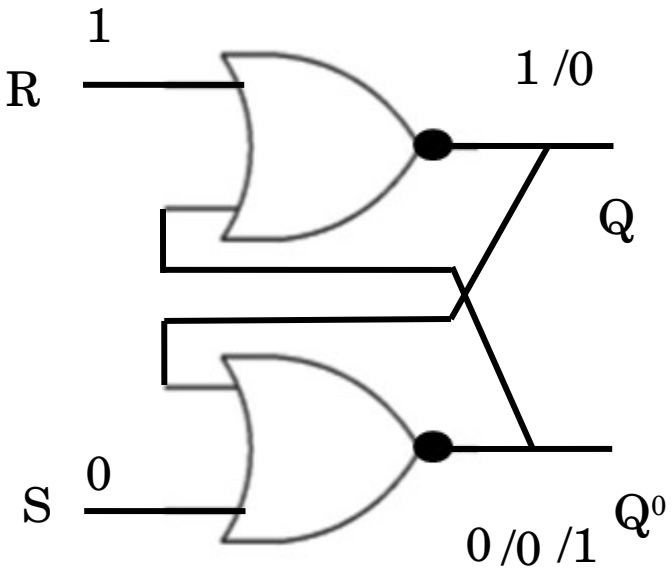
- SR Latch
  - NAND Latch
  - NOR Latch



S-R LATCH (NOR GATE)

A	B	$F=(A+B)^0$
0	0	1
0	1	0
1	0	0
1	1	0

NOR Gate

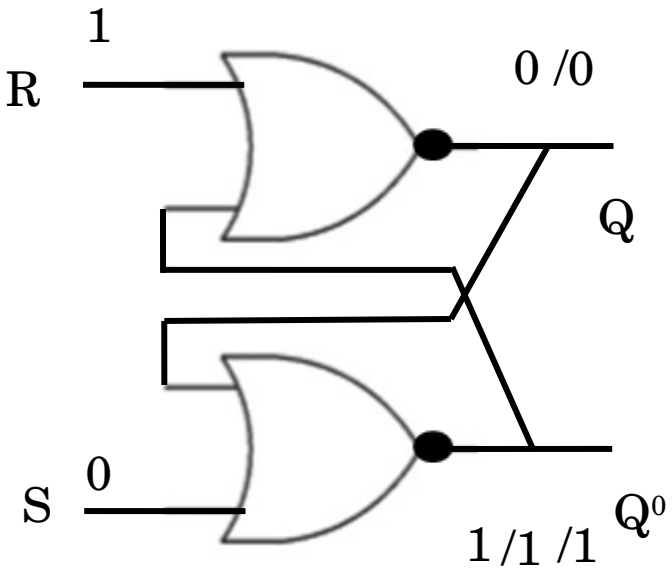


S	R	Q	Q^0
0	0		
0	1	0	1
1	0		
1	1		

S-R LATCH (NOR GATE)

A	B	$F=(A+B)^0$
0	0	1
0	1	0
1	0	0
1	1	0

NOR Gate

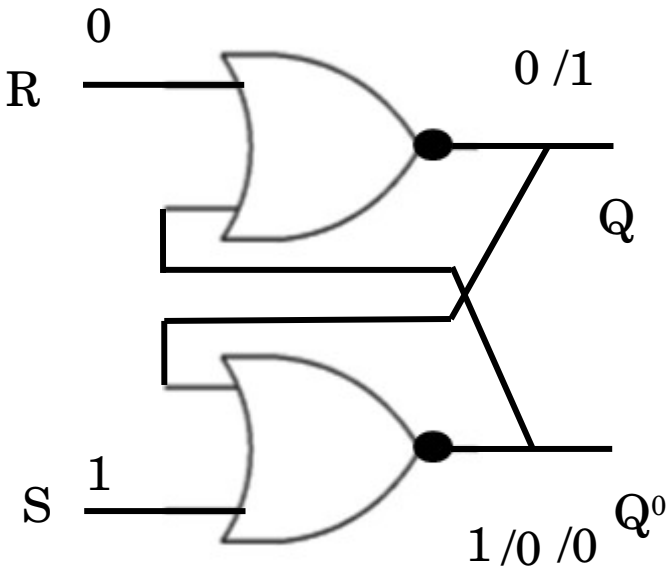


S	R	Q	Q^0
0	0		
0	1	0	1
1	0		
1	1		

S-R LATCH (NOR GATE)

A	B	$F=(A+B)^0$
0	0	1
0	1	0
1	0	0
1	1	0

NOR Gate

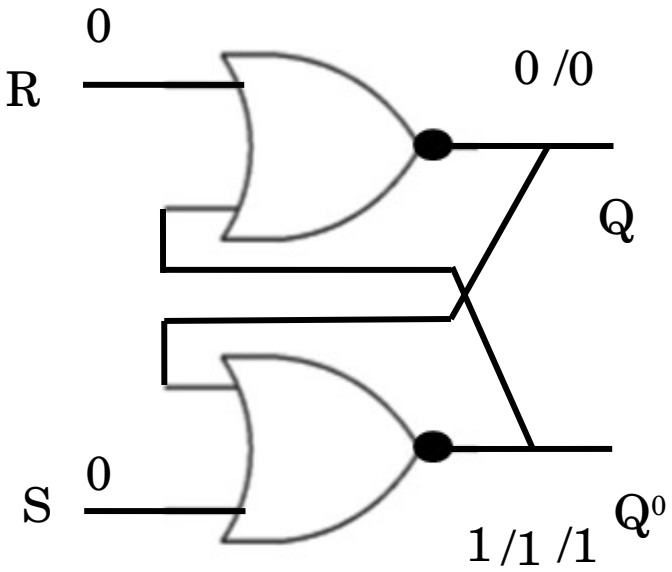


S	R	Q	Q^0
0	0		
0	1	0	1
1	0	1	0
1	1		

S-R LATCH (NOR GATE)

A	B	$F=(A+B)^0$
0	0	1
0	1	0
1	0	0
1	1	0

NOR Gate

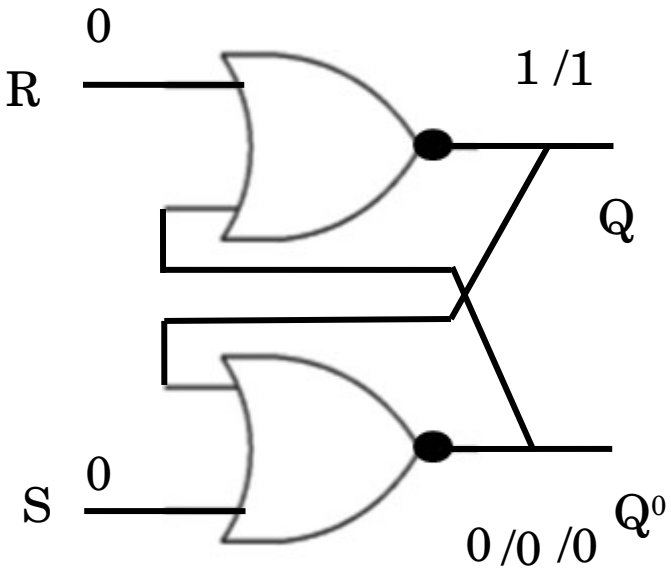


S	R	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1		

S-R LATCH (NOR GATE)

A	B	$F=(A+B)^0$
0	0	1
0	1	0
1	0	0
1	1	0

NOR Gate

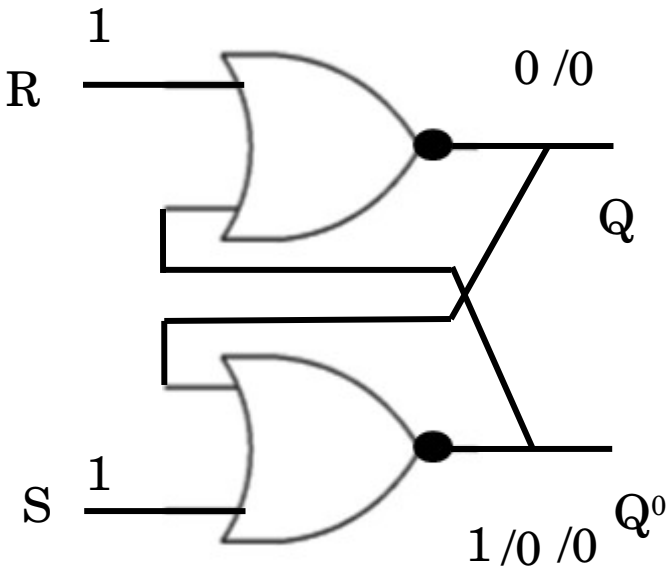


S	R	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1		

S-R LATCH (NOR GATE)

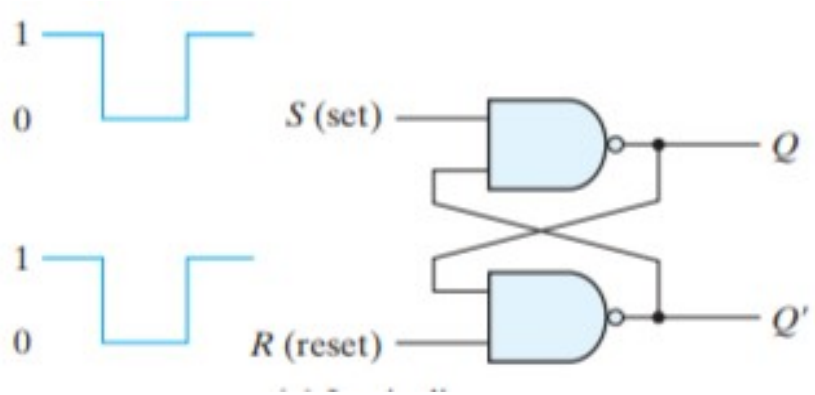
A	B	$F=(A+B)^0$
0	0	1
0	1	0
1	0	0
1	1	0

NOR Gate

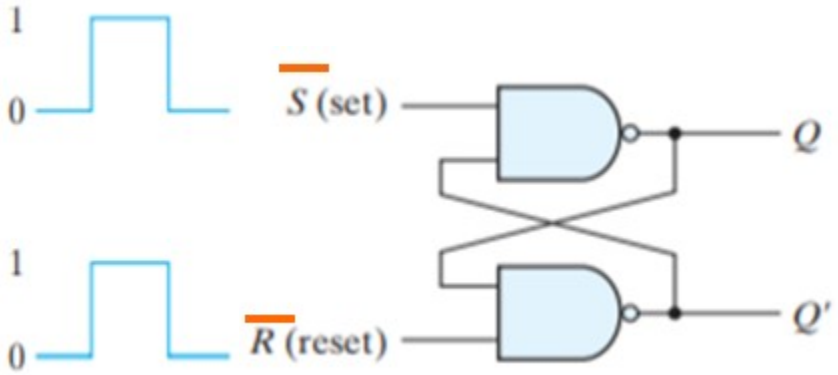


S	R	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	0	0

S-R LATCH (NAND GATE)



S	R	Q	Q^0
0	0	1	1
0	1	1	0
1	0	0	1
1	1	NC	NC

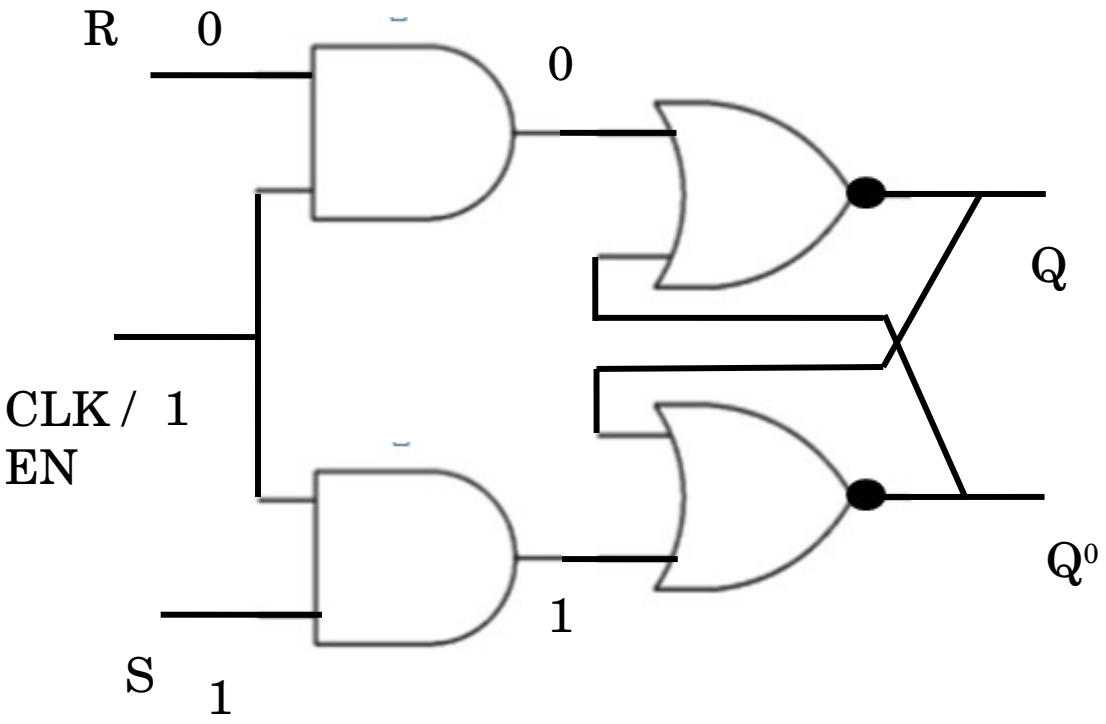


S	R	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	0	0

S-R FLIP FLOP (NOR GATE)

S	R	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	I*	I*

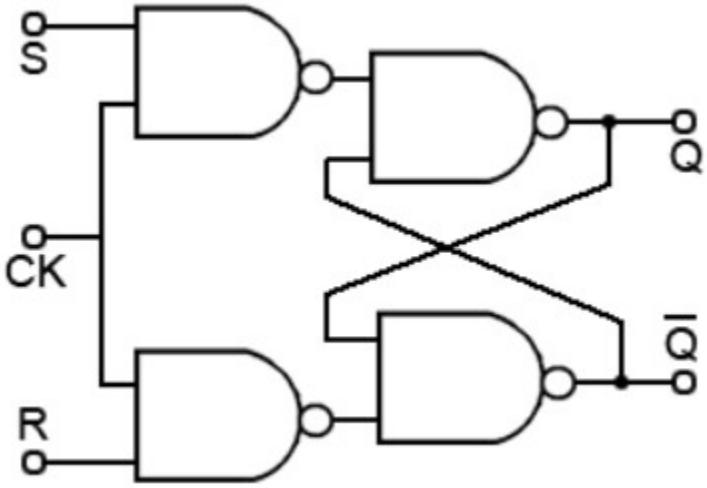
I* =Indeterminate



S-R FLIP FLOP (NAND GATE)

S	R	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	I*	I*

I* =Indeterminate

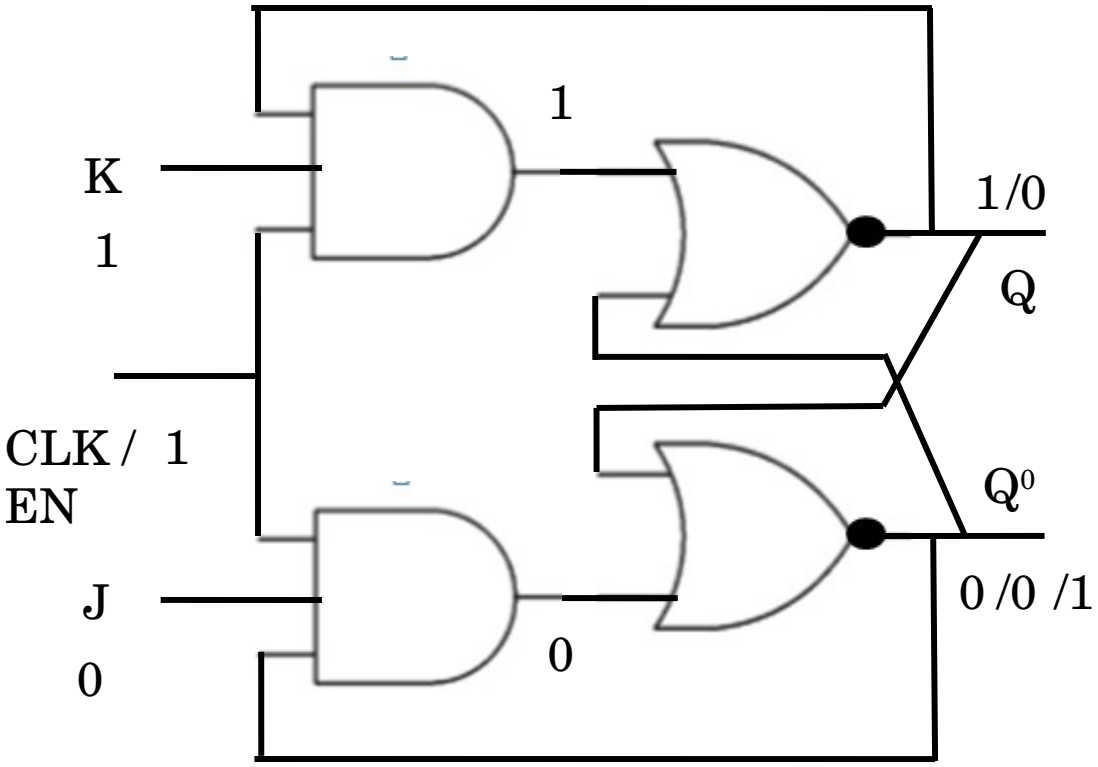


J-K FLIP FLOP

J	K	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	T*	T*

I* =Toggle

A	B	$F=(A+B)^0$
0	0	1
0	1	0
1	0	0
1	1	0



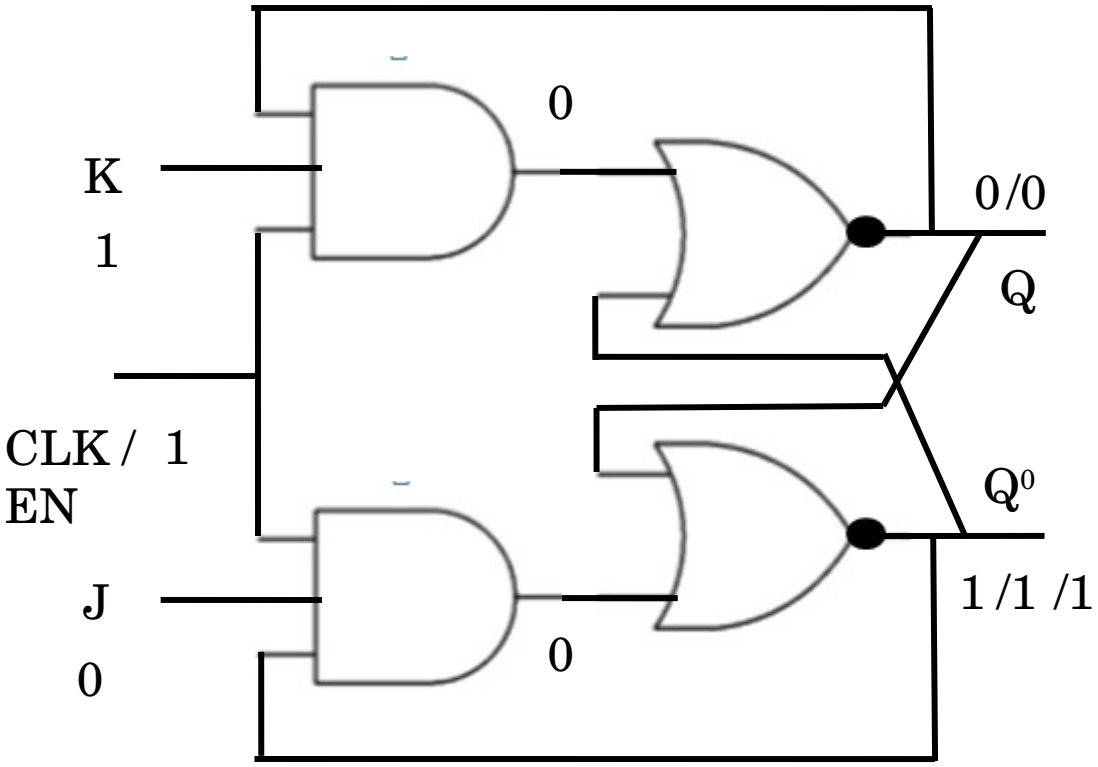
NOR Gate

J-K FLIP FLOP(CONT…)

J	K	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	T*	T*

I* =Toggle

A	B	$F=(A+B)^0$
0	0	1
0	1	0
1	0	0
1	1	0



NOR Gate

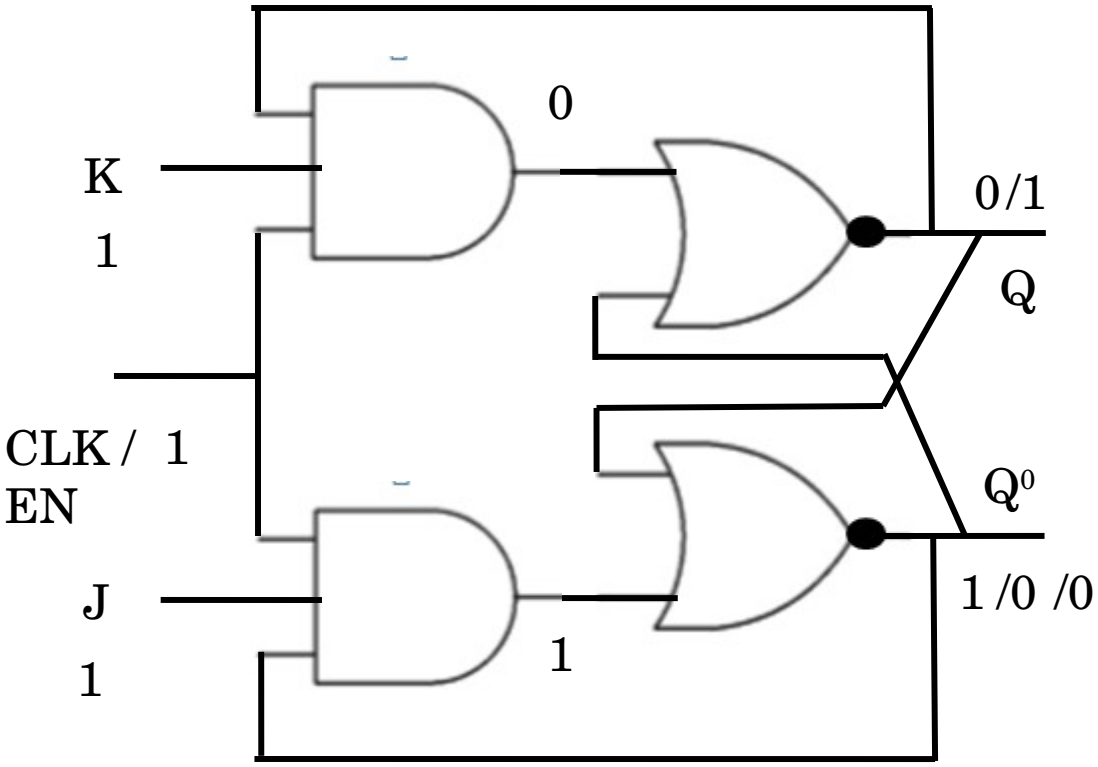
J-K FLIP FLOP(CONT…)

J	K	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	T*	T*

I* =Toggle

A	B	$F=(A+B)^0$
0	0	1
0	1	0
1	0	0
1	1	0

NOR Gate



Toggle Condition

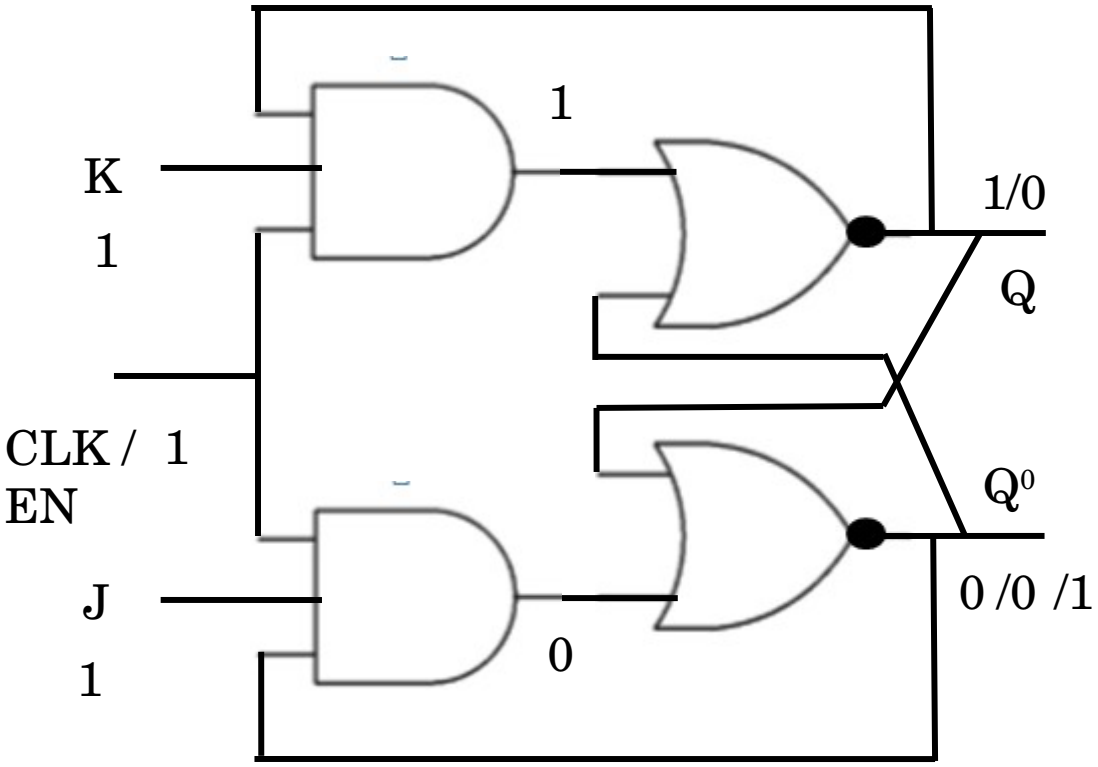
J-K FLIP FLOP(CONT…)

J	K	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	T*	T*

I* =Toggle

A	B	$F=(A+B)^0$
0	0	1
0	1	0
1	0	0
1	1	0

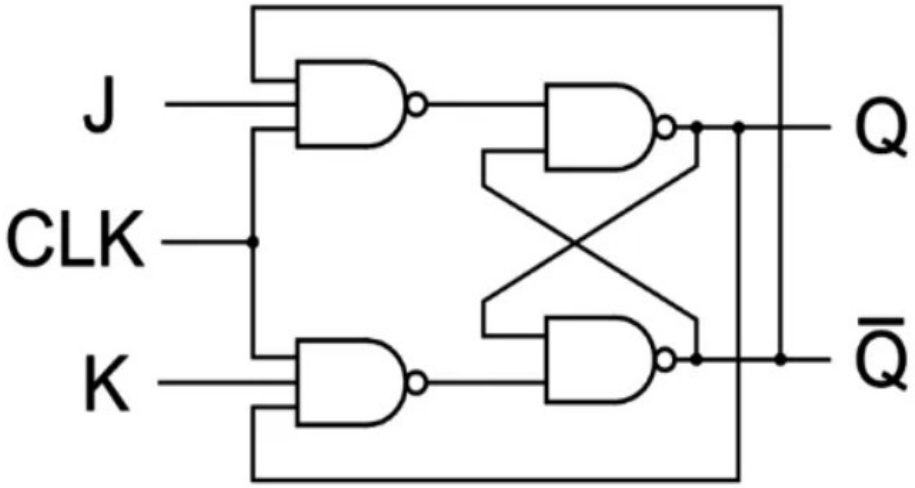
NOR Gate



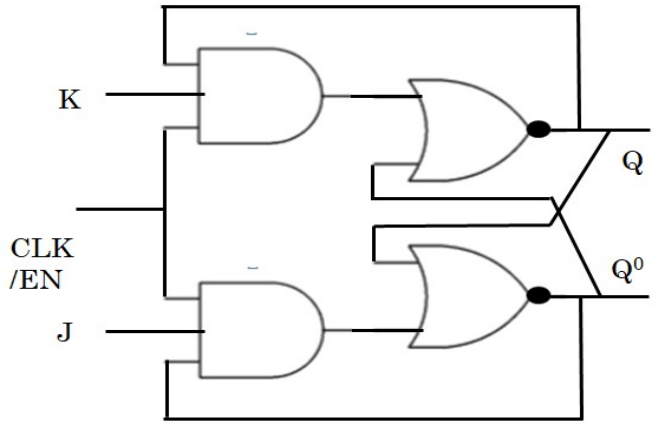
Toggle Condition

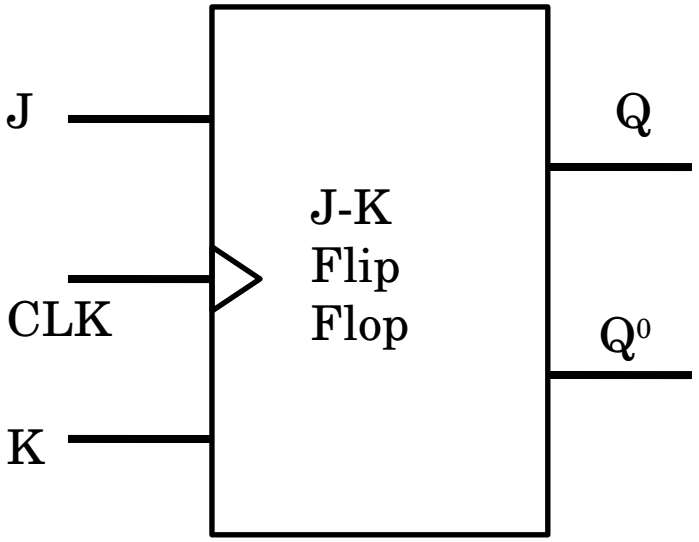
J-K FLIP FLOP (NAND)

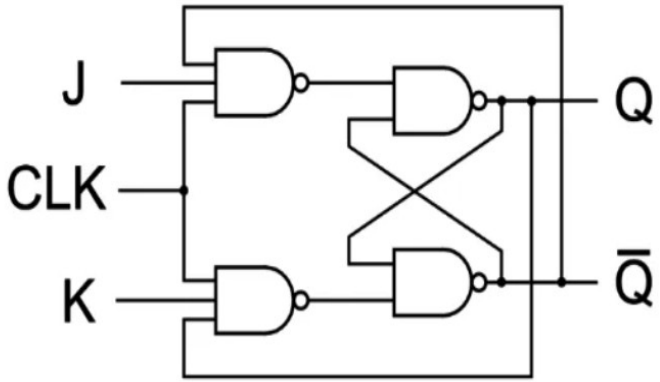
J	K	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	T*	T*



J-K FLIP FLOP

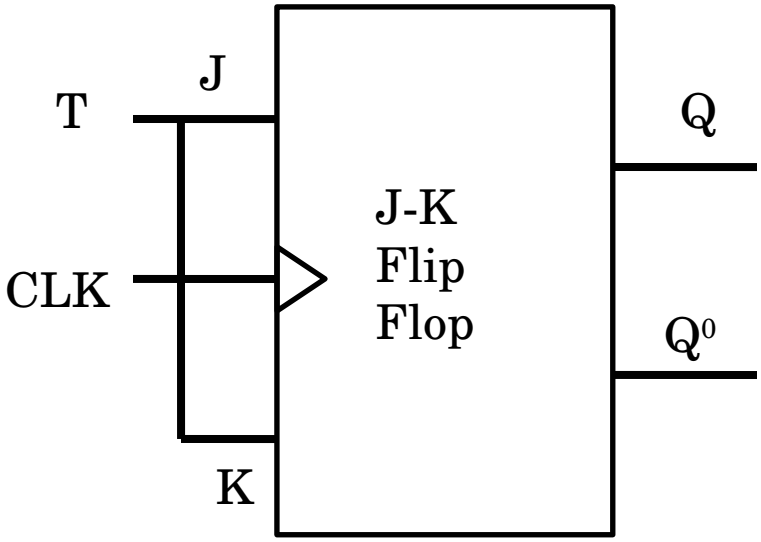






T FLIP FLOP (TOGGLE)

J	K	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	T*	T*



T	Q	Q^0
0	NC	NC
1	Toggle	Toggle

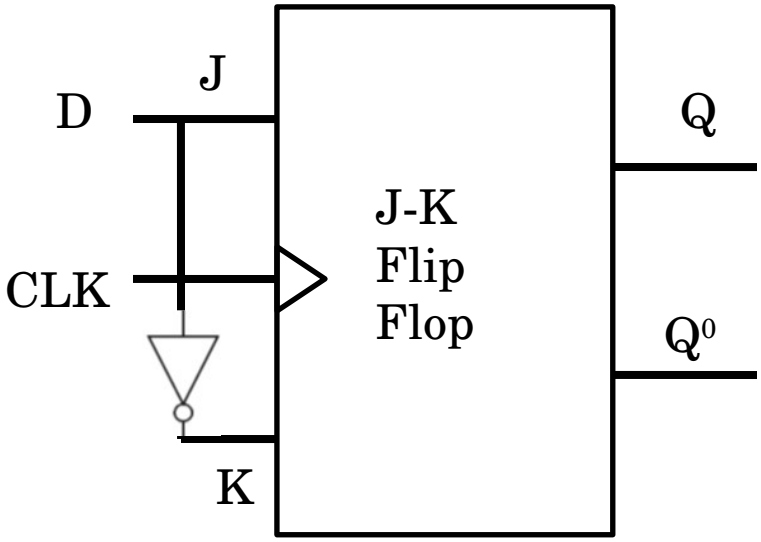
Applications:

- Registers
- Counters
- Event Detectors
- Data Synchronizers
- Frequency Divider

D FLIP FLOP (DATA)

J	K	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	T*	T*

D	Q	Q^0
0	0	1
1	1	0



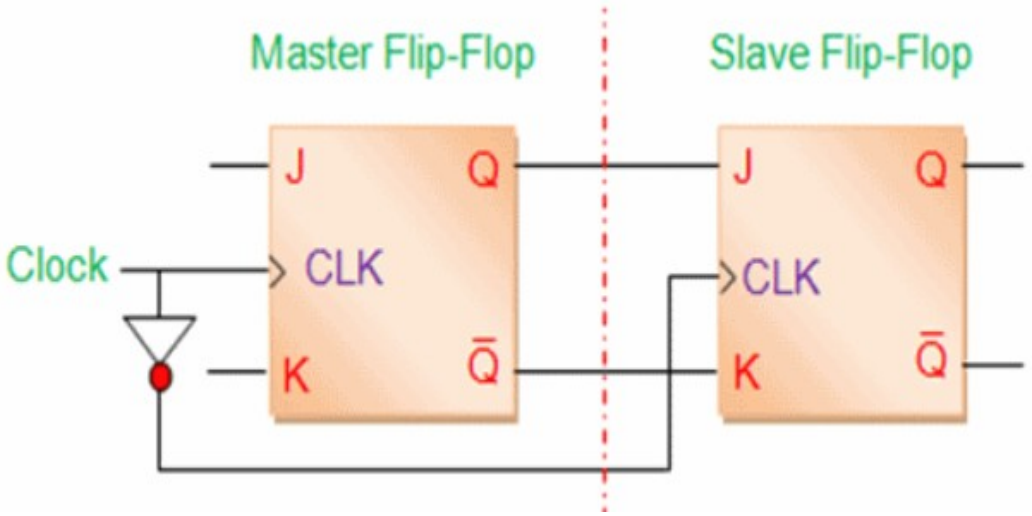
Applications:

- Data storage registers.
- Data transferring as shift registers.
- Frequency division circuits.

MASTER-SLAVE J-K FLIP FLOP

- The **Master-Slave Flip-Flop** is basically two gated SR flip-flops connected together in a series configuration with the slave having an inverted clock pulse.
- The outputs from "Slave" flip-flop are fed back to the inputs of the "Master" with the outputs of the "Master" flip flop being connected to the two inputs of the "Slave" flip flop.
- The master-slave flip-flop eliminates all the timing problems

J	K	Q	Q^0
0	0	NC	NC
0	1	0	1
1	0	1	0
1	1	T*	T*



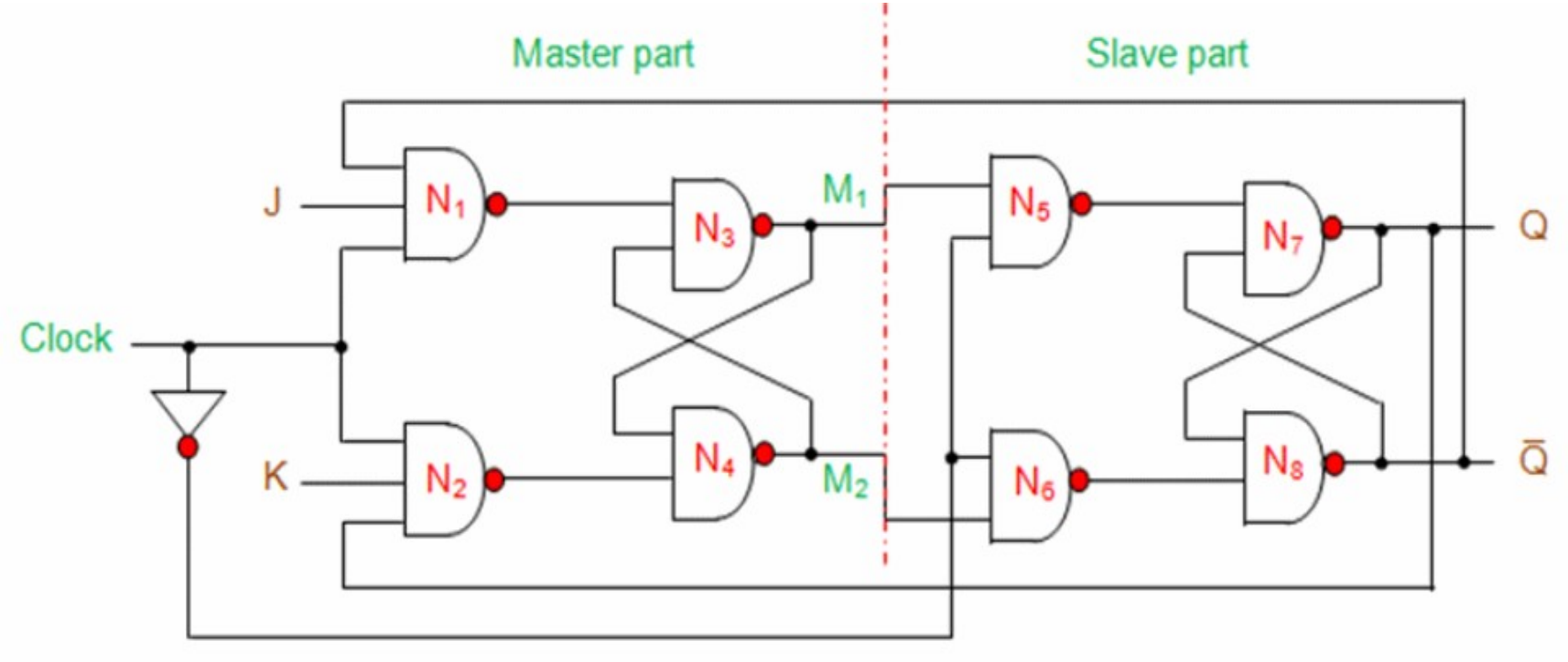
EDGE TRIGGERED & LEVEL TRIGGERED

(a) Response to positive level

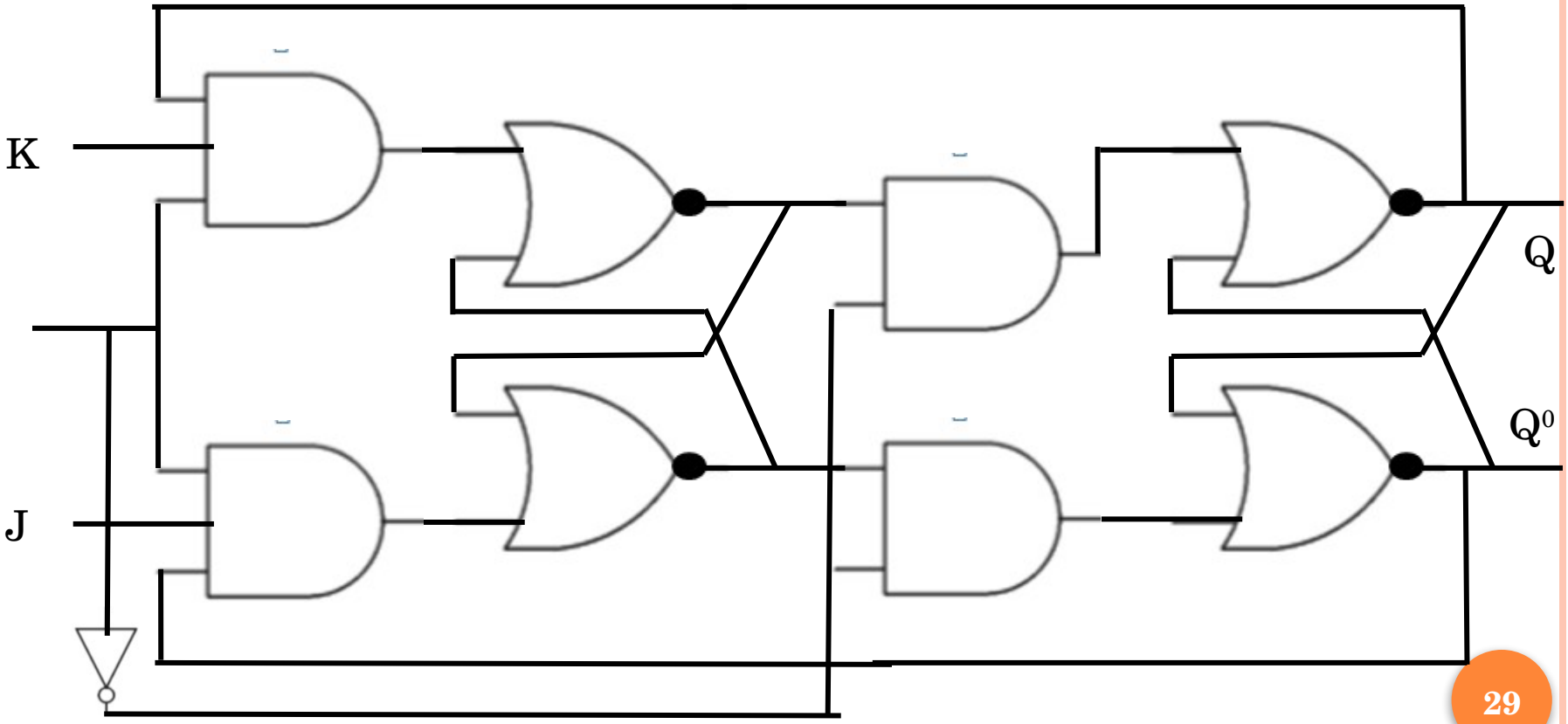
(b) Positive-edge response

(c) Negative-edge response

MASTER-SLAVE J-K FLIP FLOP(NAND)



MASTER-SLAVE J-K FLIP FLOP (NOR)



CSE 219: DIGITAL LOGIC & COMPUTER DESIGN

Tamanna Haque Nipa
Assistant Professor
Dept. of CSE
Stamford University Bangladesh

COUNTER

- Types
  - Asynchronous Counter
  - Synchronous Counter

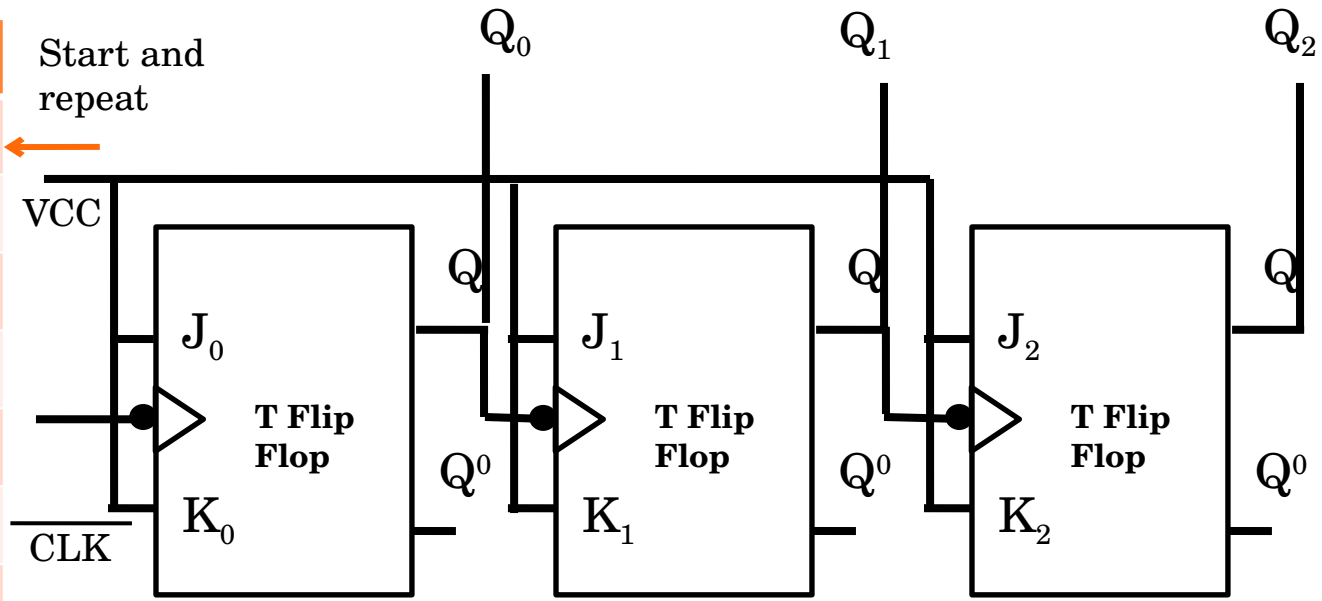
- Applications of Counter
  - Frequency counters
  - Digital clocks
  - Analog to digital convertors.
  - Frequency divider circuits. (The frequency divider circuit is that which divides the input frequency exactly by '2'.)
  - For calculating time or timers

ASYNCHRONOUS COUNTER (RIPPLE COUNTER)

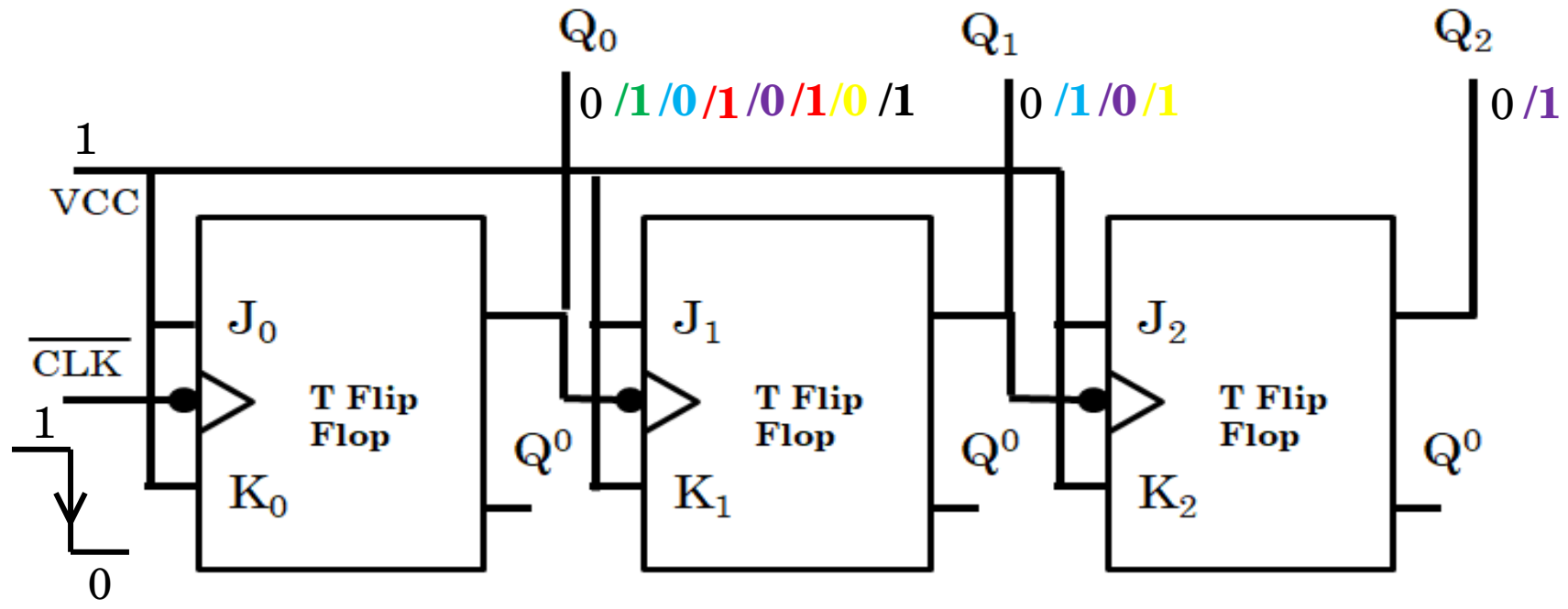
- An asynchronous counter is one in which the flip-flops (FF) within the counter do not change states at exactly the same time because they do not have a common clock pulse.
- Propagation delay is higher than that of synchronous counters. [Propagation delay is the length of time taken for a signal to reach its destination.]
- These are slow in operation.
- Less number of logic gates required.
- Low cost
- Complex to design
- Standard logic packages are not available

3 BIT ASYNCHRONOUS UP COUNTER

Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



3 BIT ASYNCHRONOUS UP COUNTER

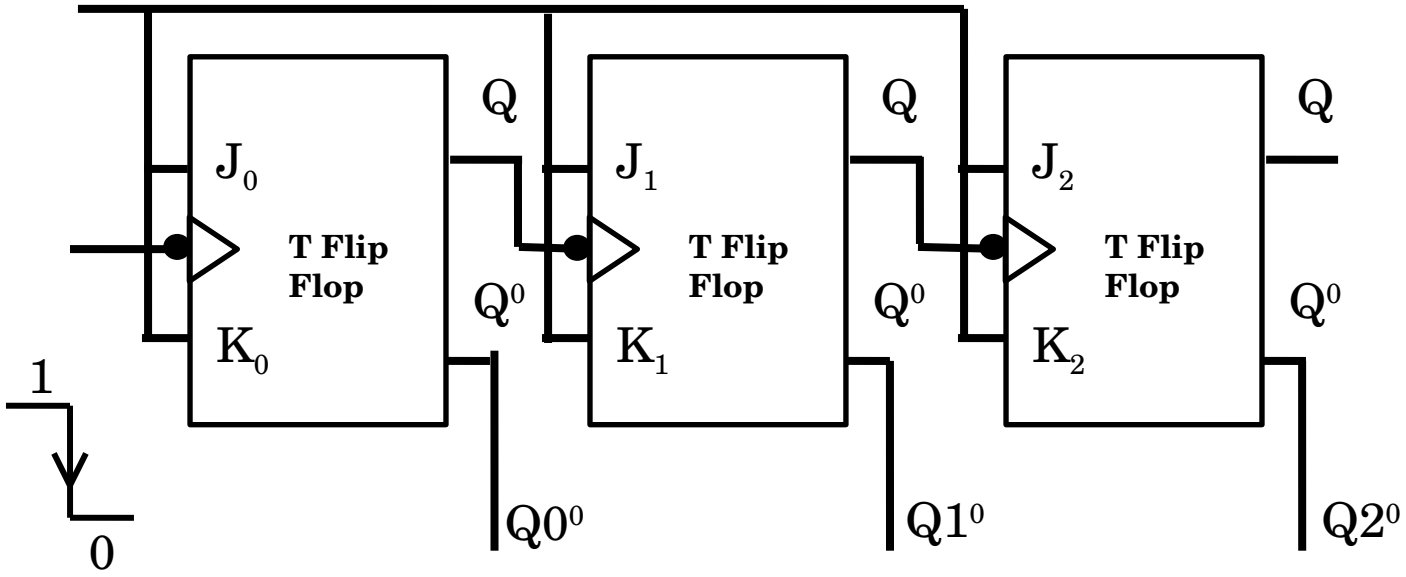


1 ⟶ 0

Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1

Q2	Q1	Q0
1	0	0
1	0	1
1	1	0
1	1	1

3 BIT ASYNCHRONOUS DOWN COUNTER



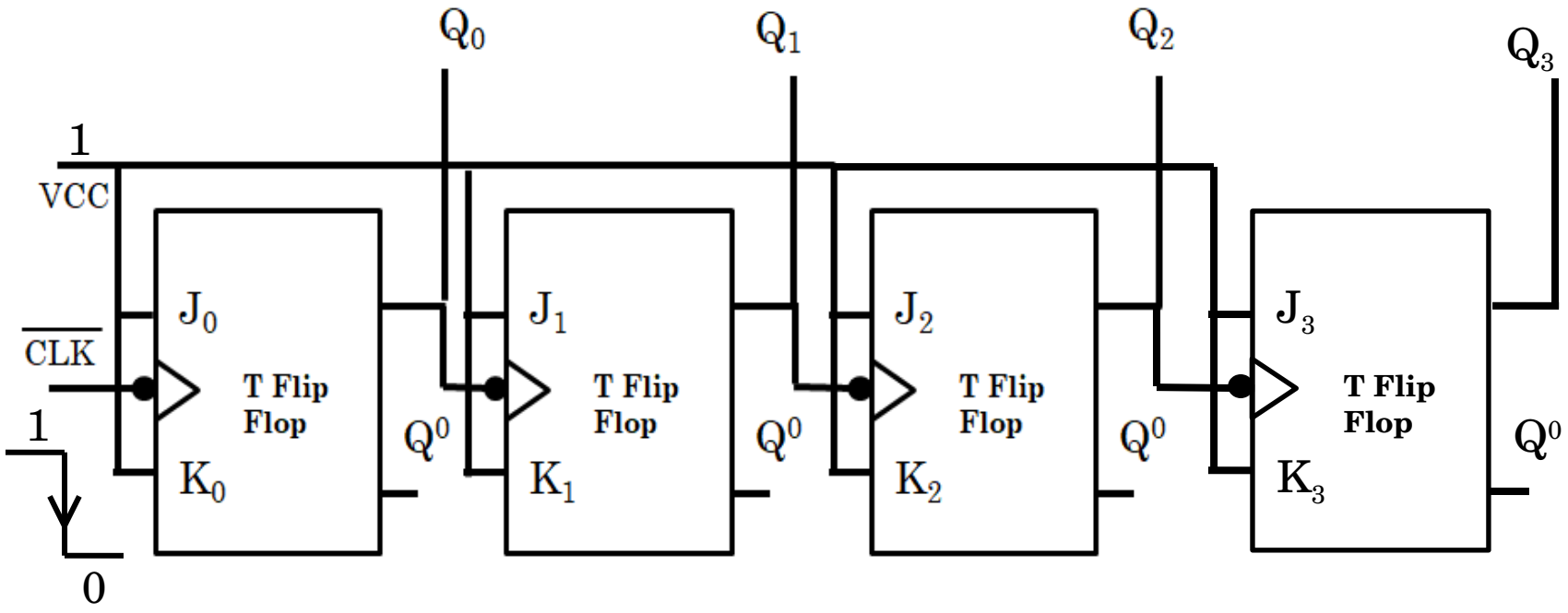
Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1

Q2⁰	Q1⁰	Q0⁰
1	1	1
1	1	0
1	0	1
1	0	0

Q2	Q1	Q0
1	0	0
1	0	1
1	1	0
1	1	1

Q2⁰	Q1⁰	Q0⁰
0	1	1
0	1	0
0	0	1
0	0	0

4 BIT ASYNCHRONOUS UP COUNTER



1 ⟶ 0

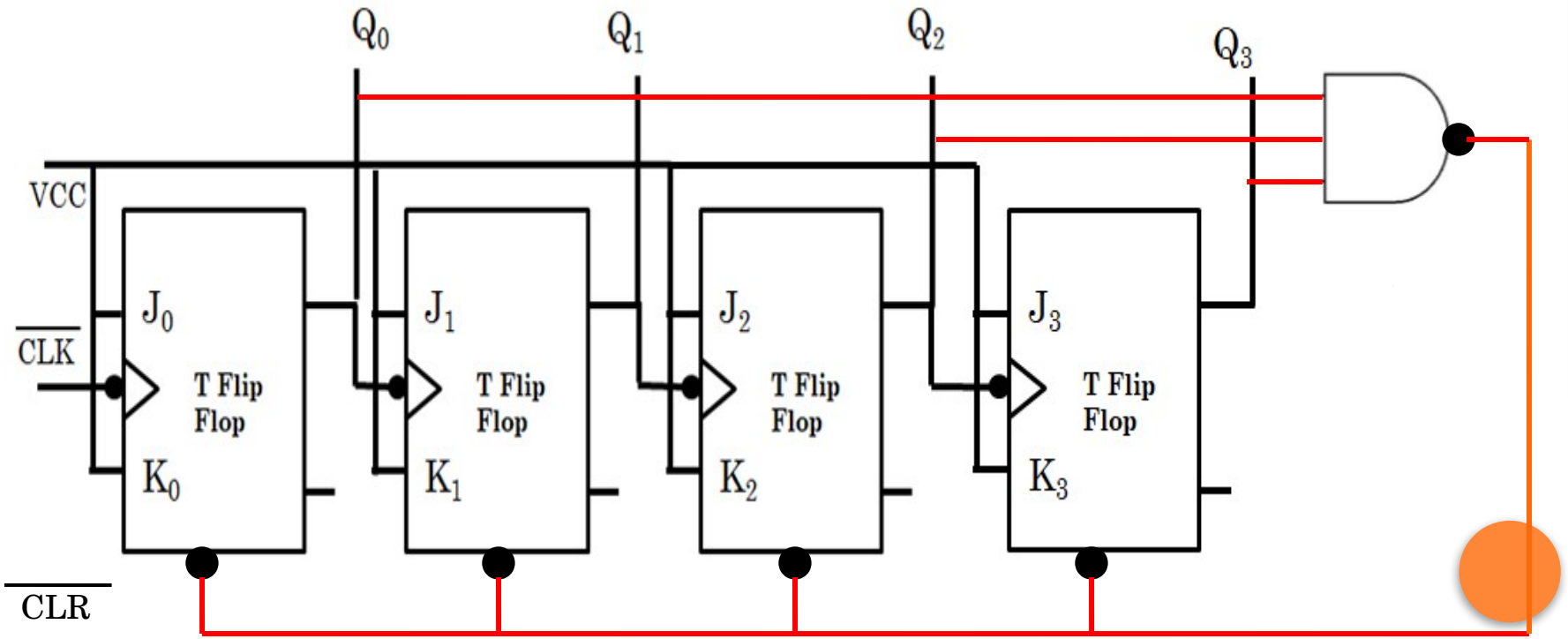
It counts 0000 to 1111.

ASYNCHRONOUS MOD 13 COUNTER

MOD 13 mean it will count 0000 to 1100 (0 to 12)

$13 = (1101)_2$

Q3	Q2	Q1	Q0
1	1	0	1



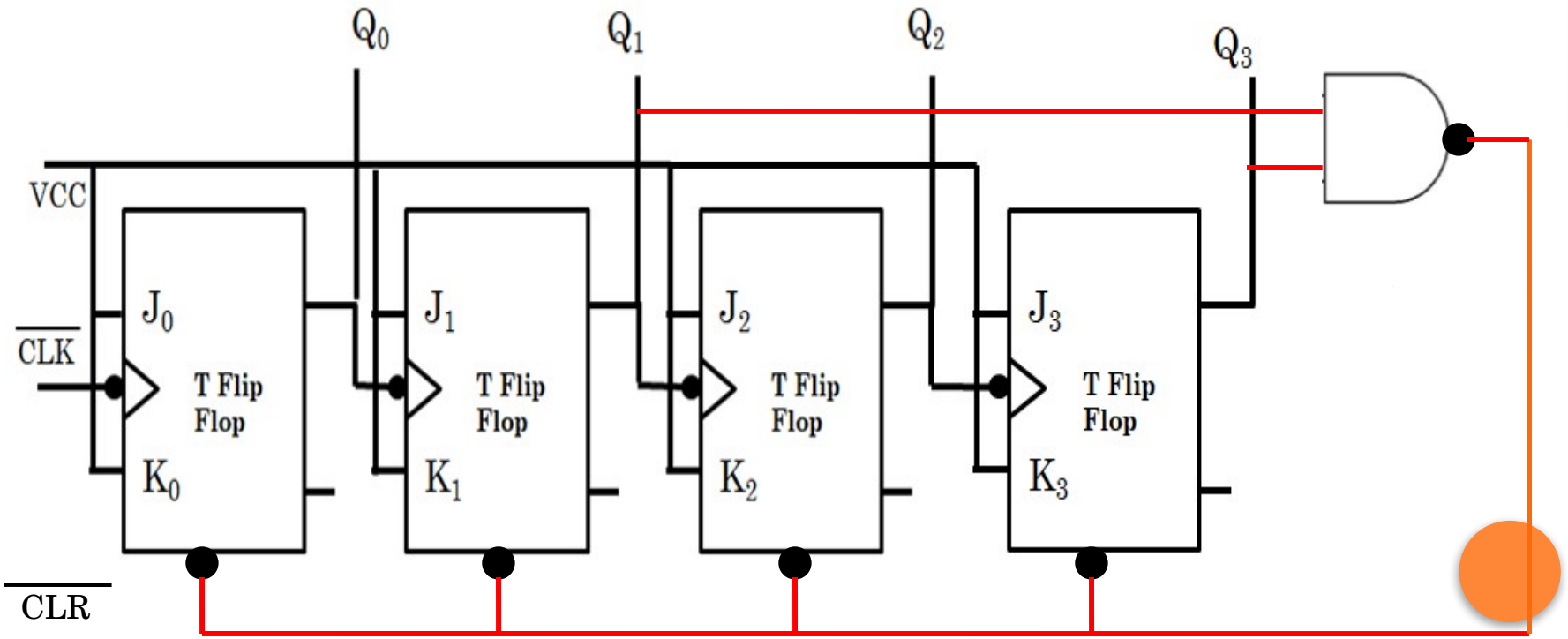
ASYNCHRONOUS MOD 10 COUNTER

MOD 10 mean it will count 0000 to 1001 (0 to 9)

$10 = (1010)_2$

Q3	Q2	Q1	Q0
1	0	1	0

Asynchronous Decade Counter

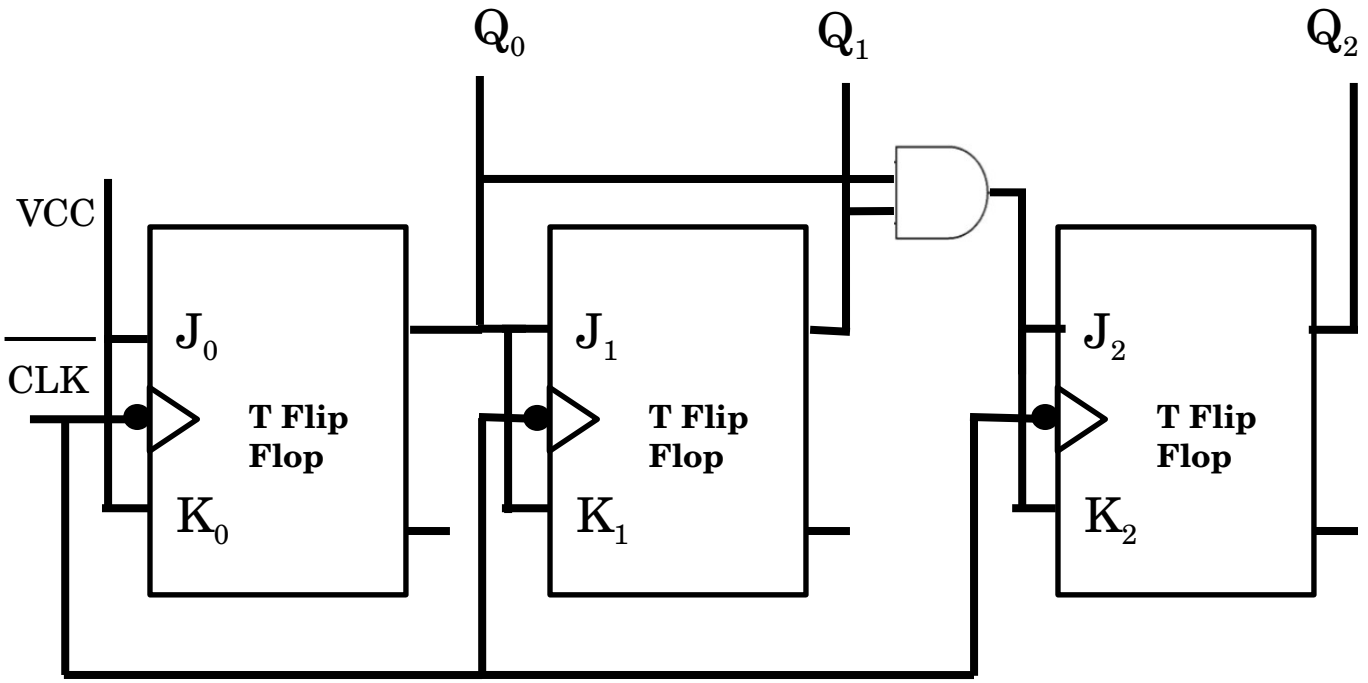


SYNCHRONOUS COUNTER

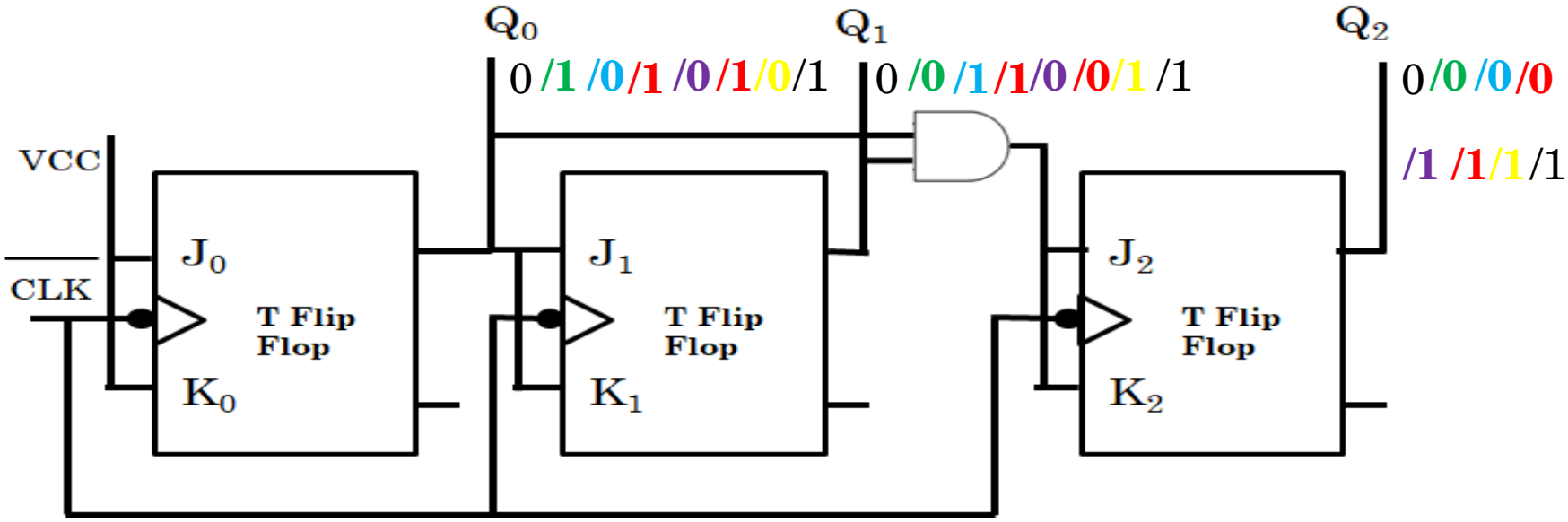
- A synchronous counter is one in which all the flip-flops in the counter are clocked at the same time by a common clock pulse
- The propagation delay is very low
- These are faster than that of ripple counters.
- Large number of logic gates are required to design
- High cost
- Synchronous circuits are easy to design.
- Standard logic packages available for synchronous.

3 BIT SYNCHRONOUS UP COUNTER

Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



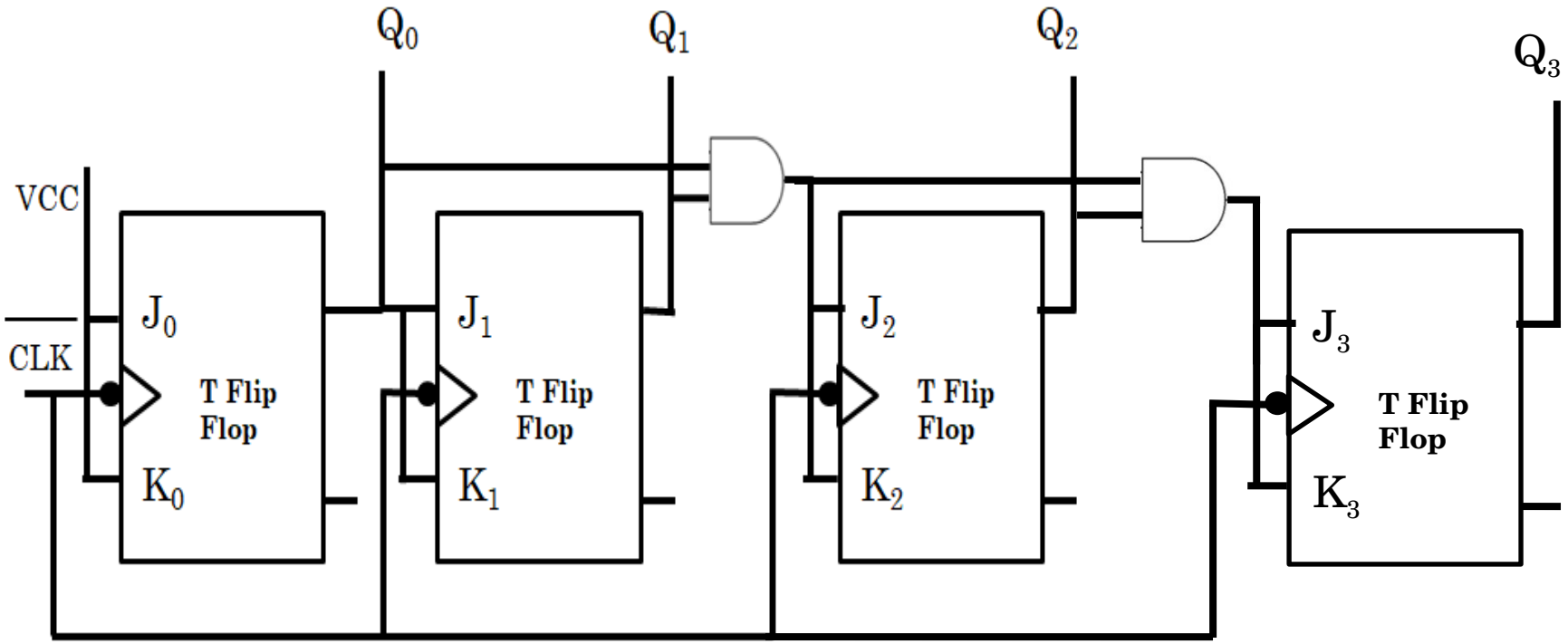
3 BIT SYNCHRONOUS UP COUNTER



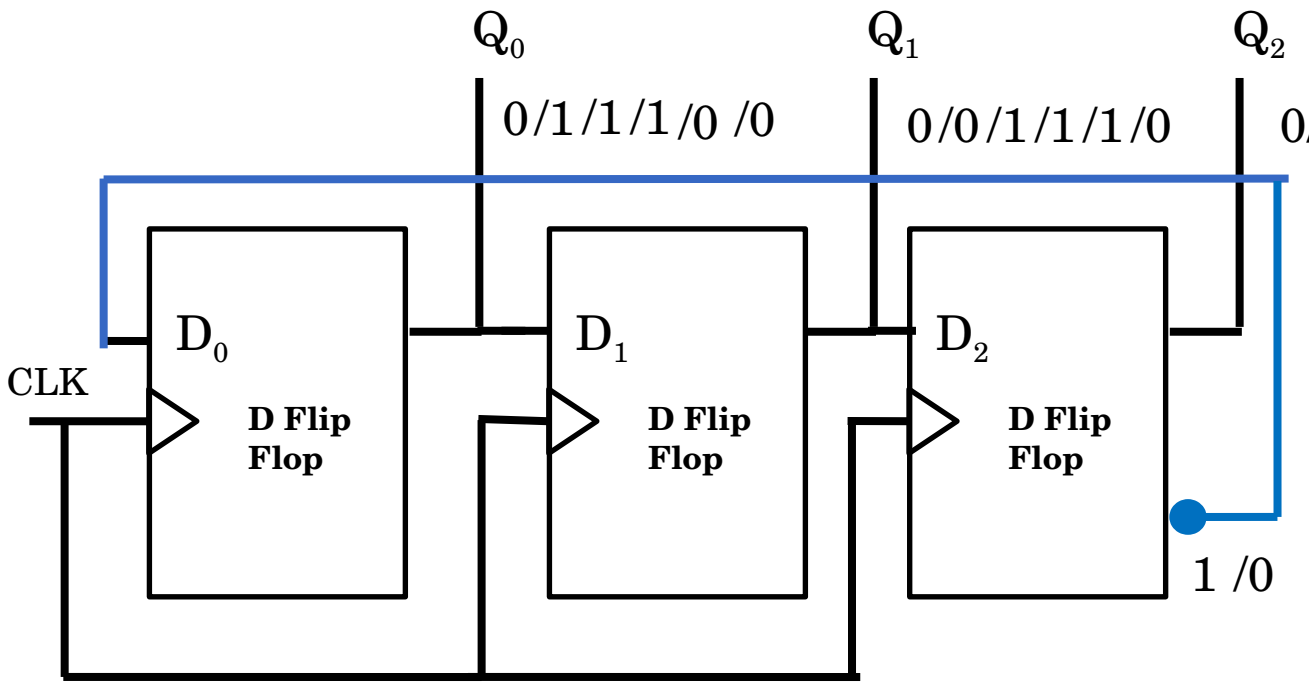
Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1

Q2	Q1	Q0
1	0	0
1	0	1
1	1	0
1	1	1

4 BIT SYNCHRONOUS UP COUNTER



Johnson Counter– 3 bit

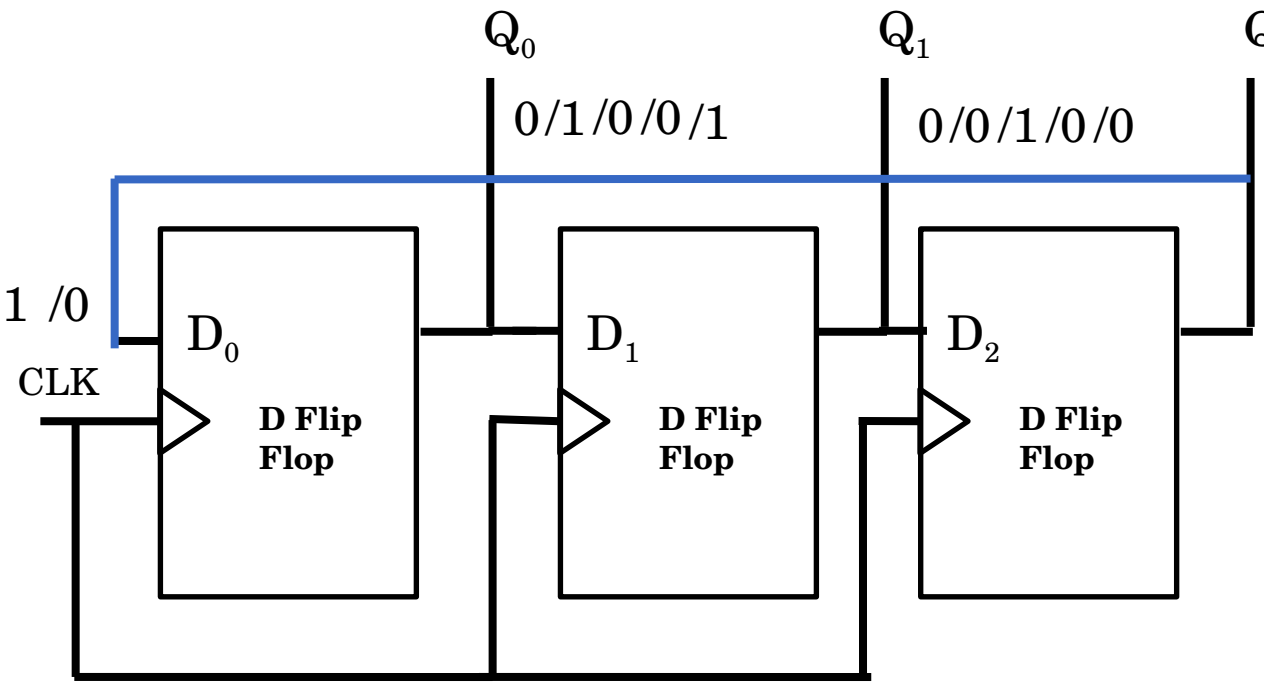


Q2	Q1	Q0
0	0	0
0	0	1
0	1	1
1	1	1
1	1	0
1	0	0
0	0	0

Input n bit so output 2n bit.
(3 bit input so it can count 2x3=6 values)

Applications(read yourself)

Ring Counter– 3 bit



Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
1	0	0
0	0	1

Input n bit so output n bit.
(3 bit input so it can count 3 values)

Applications(read yourself)

EXERCISE

- Synchronous Down Counter
- Synchronous MOD Counter

CSE 219: DIGITAL LOGIC & COMPUTER DESIGN

Tamanna Haque Nipa
Assistant Professor
Dept. of CSE
Stamford University Bangladesh

FLOATING POINT NUMBER REPRESENTATION

- What is floating point?

- 0.00005 = 0.5×10^{-4}
- 0.0125 = 0.125×10^{-1}
- - 0.0023 = -0.23×10^{-2}
- 69.00 = 0.69×10^{2}

Basic Format:

$$\pm Value = \pm Mantissa \times Base^{\pm Exponent}$$

Base will 2 or 10 depending on number system.

BASIC FORMAT

$$\pm Value = \pm Mantissa \times Base^{\pm Exponent}$$

- As we need to represent a float value using 0 / 1, so the base must be 2 (fixed).
- So we need to store only
  - M (Mantissa) and
  - E (Exponent)

EXAMPLE

- Let we are representing a 3 bit float value. That means M and E both are 3 bits.

(---, ---)

(M , E)

- Here, M and E both are represented using Sign Magnitude format (integer representation)
- We need to find 4 values
  - Highest +ve value
  - Highest –ve value
  - Lowest +ve value and
  - Lowest –ve value

Example cont…
[M and E both are 3 bits (M, E)]

- Highest +ve value (+M, +E)

  (011, 011) = 3 x 2^3 = 3 x 8 = 24

- Lowest +ve value (+M, -E)

  (001, 111) = 1 x 2^{-3} = 1 x 1/8 = 0.125

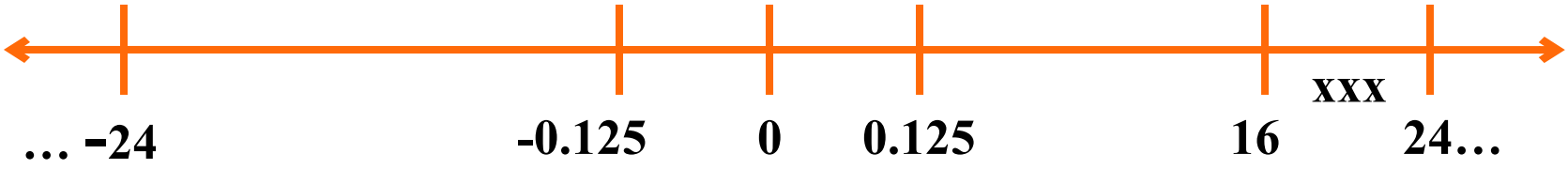
- Lowest -ve value (-M, +E)

  (111, 011) = -3 x 2^3 = -3 x 8 = -24

- Highest –ve value (-M, -E)

  (101, 111) = -1 x 2^{-3} = -1 x 1/8 = -0.125

RANGE – 3 BIT FP



- So range is

  -24 to +24

- All values cant be represented
- The previous value of +24 is

  $(010, 011) = +2 \times 2^3$

  $= 2 \times 8$

  $= 16$

PROBLEMS

- All values cant be represented (So arithmetic operation is not possible)

- Underflow
  - 0.125 to 0 there are many values
  - 0 to -0.125 there are many values
- Overflow
  - +25 to $+\infty$
  - -25 to $-\infty$
- Not a Number (NaN) $ex.\sqrt{-2}$ → Not defined
- Zero representation → too many ways

ZERO REPRESENTATION

- M must be Zero and E can be anything, 0×2^E
- If M=0, then M can be written as 2 ways

  +ve zero , M= 000 and -ve zero, M = 100
- As E can be anything, so using 3 bit for E, we will get 8 possible combinations. That means-

M = 000, E = 000
001
010
.
.
111

and

M = 100, E = 000
001
010
.
.
111

2 x 8 = 16 formats

- If M and E are 4 bits then 2 x 16 =32 ways we can represent zero → Not Possible

NORMALIZATION AND BIASING

- 0.000005
- 0.0005×10^{-2}
- 0.5×10^{-5} √
- 5.0×10^{-6}

→ Must be represented in 1 way

For binary representation, before point will be 0 and after point the first value must be 1.

$0.1bbbbb \times 2^{E}$ [Here, b can be 0 or 1]

IEEE 754 STANDARD

- Types

  1) Single Precision (float) → 32 bit

$$value = (-1)^S \times 1.M \times 2^{E-127}, 0 < E < 255$$

  2) Double Precision (double) → 64 bit

$$value = (-1)^S \times 1.M \times 2^{E-1023}, 0 < E < 2047$$

EXAMPLE (32 BIT)

- **Represent 78.325**

$$value = (-1)^S \times 1.M \times 2^{E-127}, 0 < E < 255$$

- If the value is +ve then S=0 otherwise S=1
- To get1.M, the given value must be divided by immediate smaller power of 2.

1 2 4 8 16 32 64 128 256 512 …..

78.325

EXAMPLE (32 BIT)CONT…

$$value = (-1)^S \times 1.M \times 2^{E-127}, 0 < E < 255$$

64

2^6

$2^{E-127=6}$

Here, E-127=6

» E = 6+127

» E = 133

→Excess 127
(Biasing Exponent)

EXAMPLE (32 BIT)CONT…

$$value = (-1)^S \times 1.M \times 2^{E-127}, 0 < E < 255$$

E = 133 →

1	0	0	0	0	1	0	1
+128	64	32	16	8	4	2	1

133-128=5

5-4=1

We got,

0	10000101	M=23 bit

We have 1.M = 1 is in equation. So it not required to store.
We need to covert M into binary value.

EXAMPLE (32 BIT)CONT…

$$value = (-1)^S \times 1.M \times 2^{E-127}, 0 < E < 255$$

M = → 00111…

(we will calculate 5/6 bit positions)

Multiply	Result	Integer Part
0.223828 x 2	0.447656	0
0.447656 x 2	0.895312	0
0.895312 x 2	1.790624	1
0.790624 x 2	1.581248	1
0.581248 x 2	1.162496	1
…	…	….

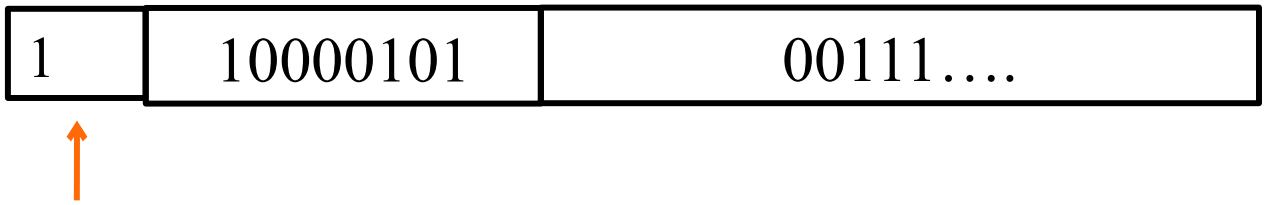
upto 23 bits

EXAMPLE (32 BIT)CONT…

$$value = (-1)^S \times 1.M \times 2^{E-127}, 0 < E < 255$$

We got, 78.325 =

- **Represent -78.325**



SET INTERPRETATIONS (32 BIT)

- If E = 255 and M ≠ 0 then NaN
- If E = 255 and M= 0 then ± ∞ (overflow)
- If E = 0 and M= 0 then zero (0)
- If 0<E < 255 and M ≠ 0 then

$$value = (-1)^S \times 1.M \times 2^{E-127}, 0 < E < 255$$

- If E = 0 and M ≠ 0 then

$$value = (-1)^S \times 0.M \times 2^{E-126} \text{ (Underflow)}$$

EXERCISE

- Represent -1.5

Solve Yourself.
Thank You.

CSE 219: DIGITAL LOGIC & COMPUTER DESIGN

Tamanna Haque Nipa
Assistant Professor
Dept. of CSE
Stamford University Bangladesh

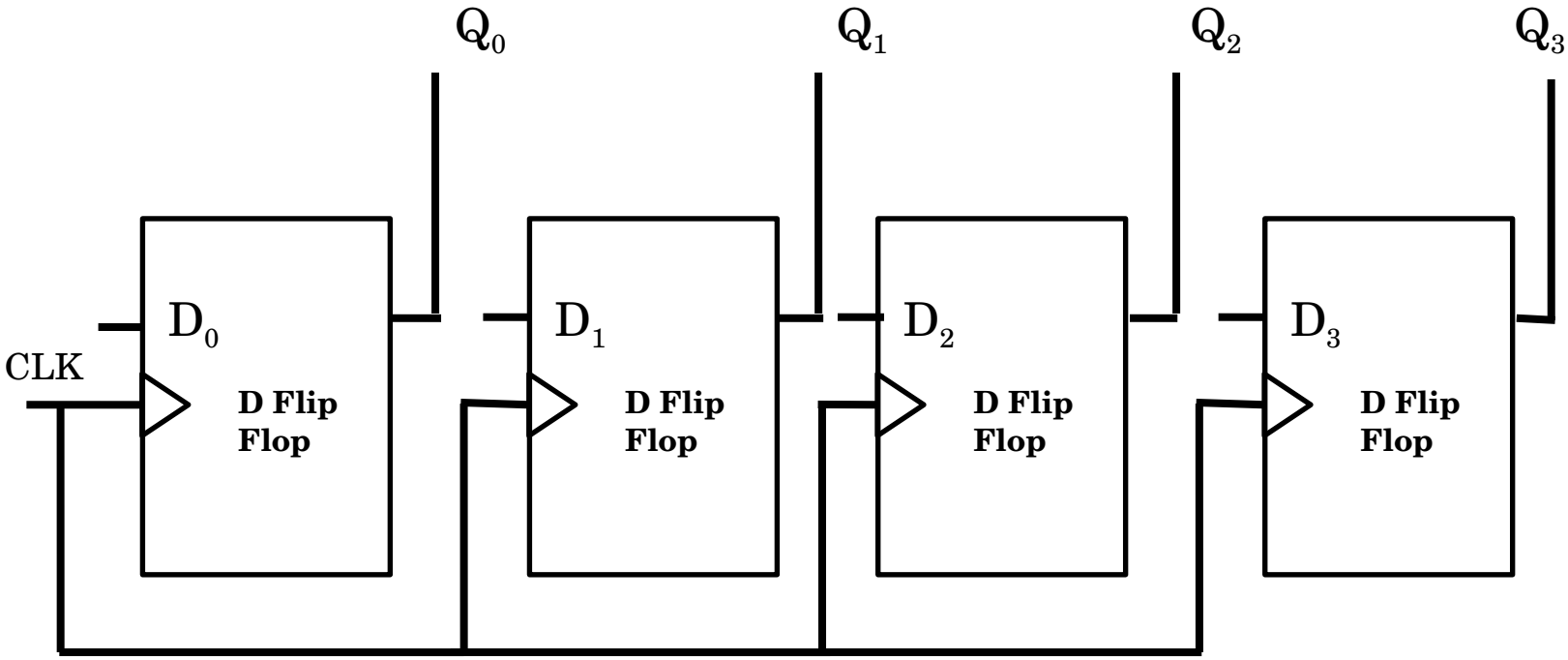
SHIFT REGISTER

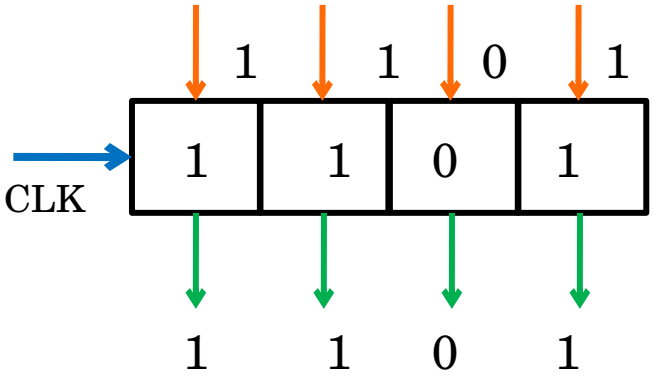
- The Shift Register is another type of sequential logic circuit that can be used for the storage or the transfer of binary data.
- Applications:
  - Used for temporary data storage.
  - Used for data transfer and data manipulation.
  - Used to produce time delay to digital circuits(SISO, PIPO)
  - The serial-in parallel-out shift register is used to convert serial data into parallel data
  - A Parallel in Serial out shift register us used to convert parallel data to serial data.

TYPES OF SHIFT REGISTER

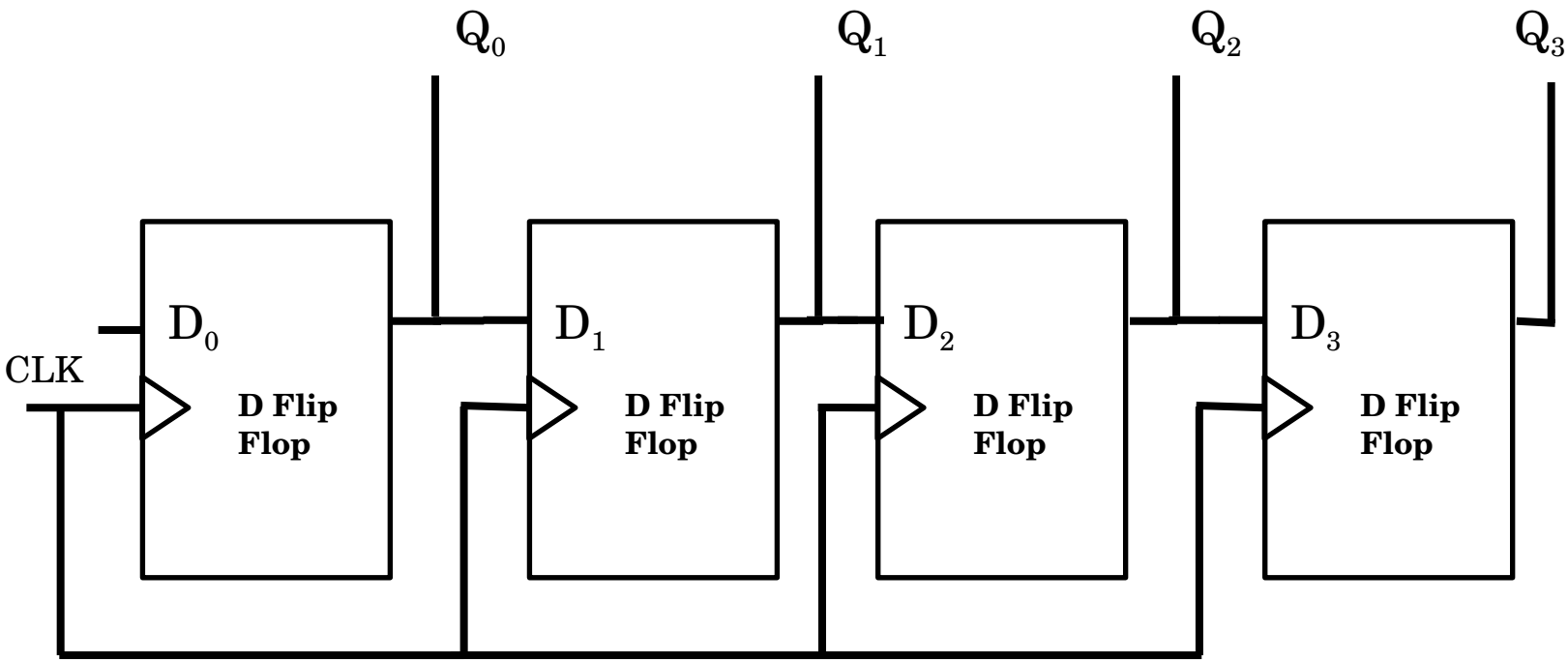
- Basically Shift Register are 4 types
  - Parallel in Parallel out
  - Parallel in Serial out
  - Serial in Serial out
  - Serial in Parallel out
- There are another 2 types of Shift Register
  - Bidirectional Shift Register
  - Universal Shift Register
- There are 2 types of Shift Register Counter
  - Johnson Counter
  - Ring Counter

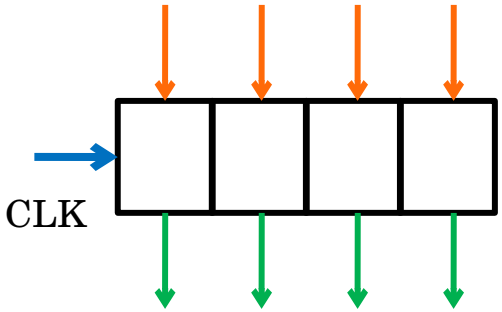
Parallel in Parallel out(PIPO) – 4 bit



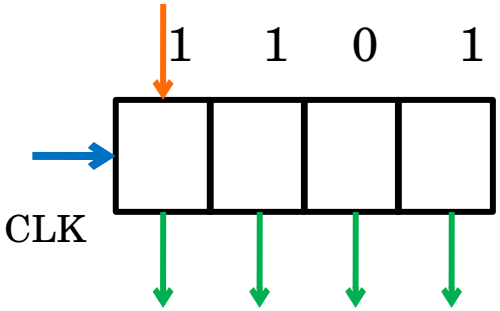


Serial in Parallel out(SIPO) – 4 bit

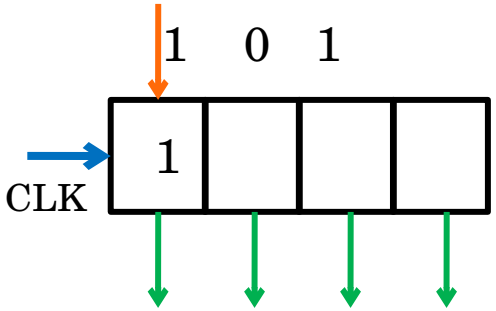




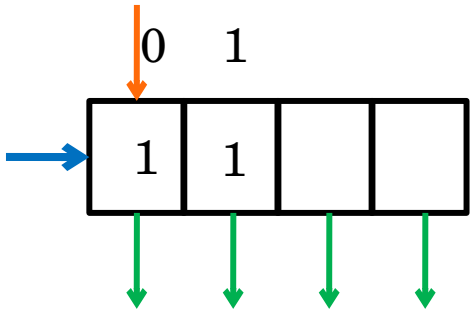
Serial in Parallel out(SIPO) – 4 bit



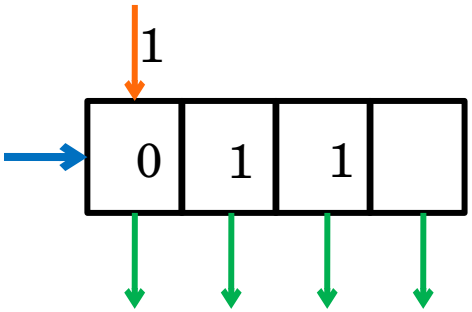
1st Clock Pulse



2nd Clock Pulse



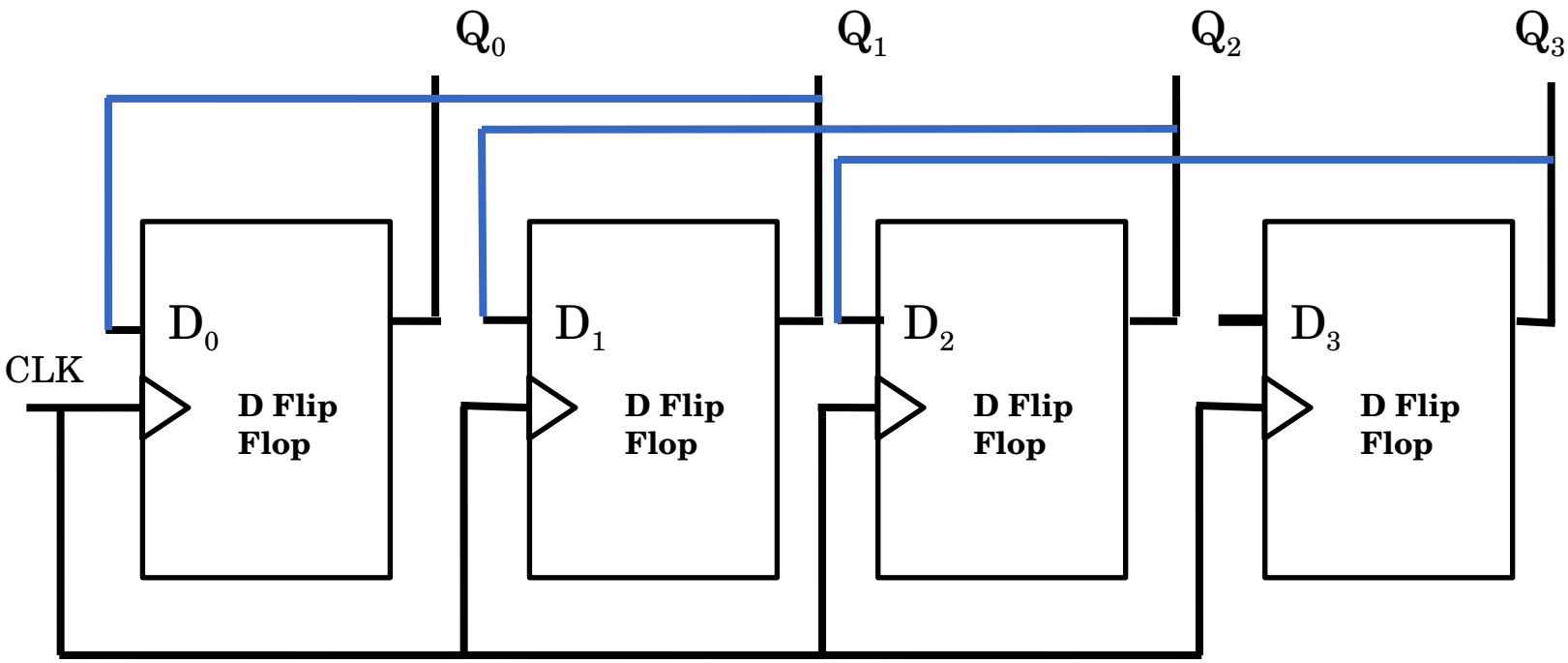
3rd Clock Pulse



4th Clock Pulse

Q_0 Q_1 Q_2 Q_3 → Q_3 Q_2 Q_1 Q_0 Left Shift

SIPO(Right Shift) – 4 bit



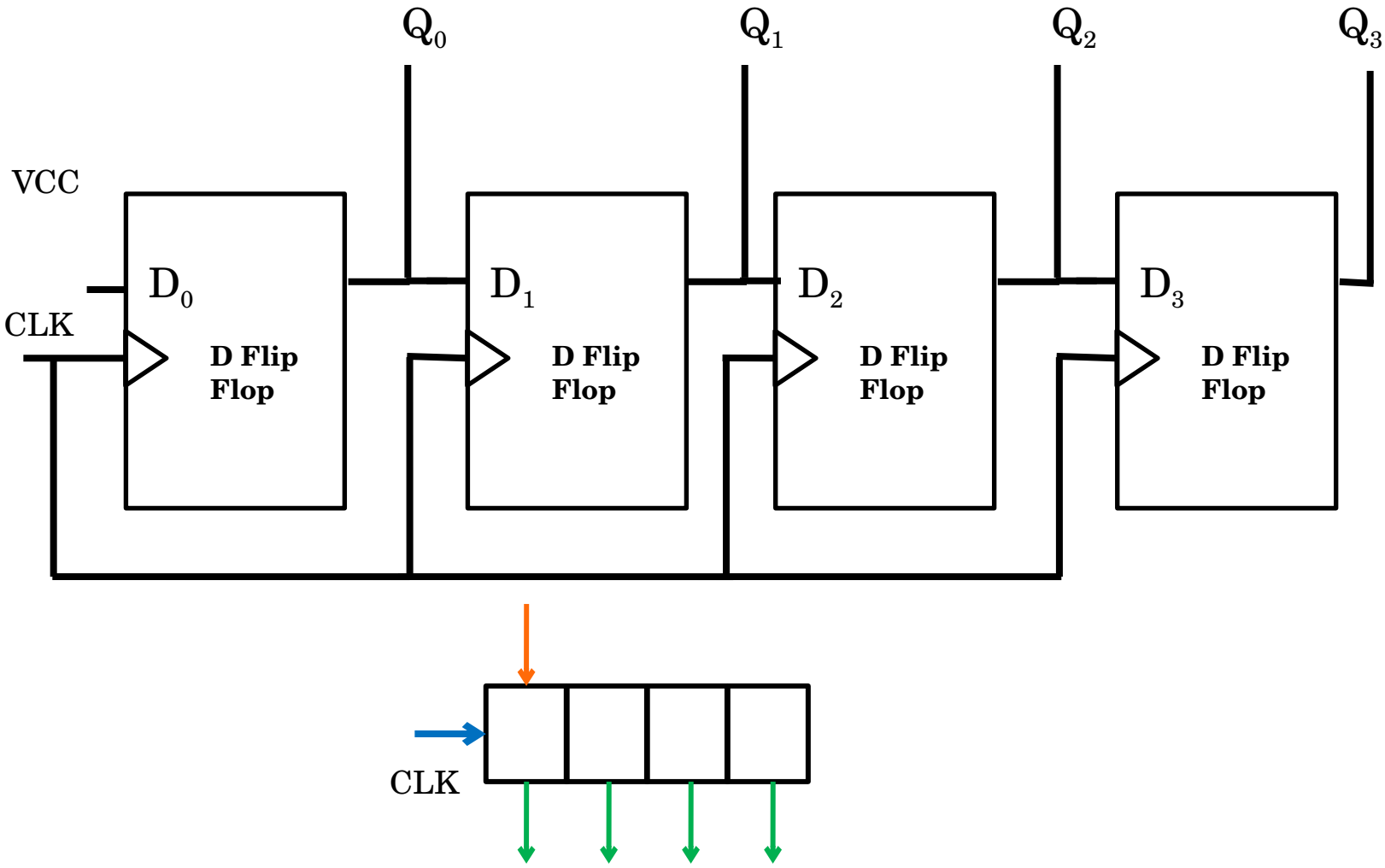
Q_3 Q_2 Q_1 Q_0

Left Shift

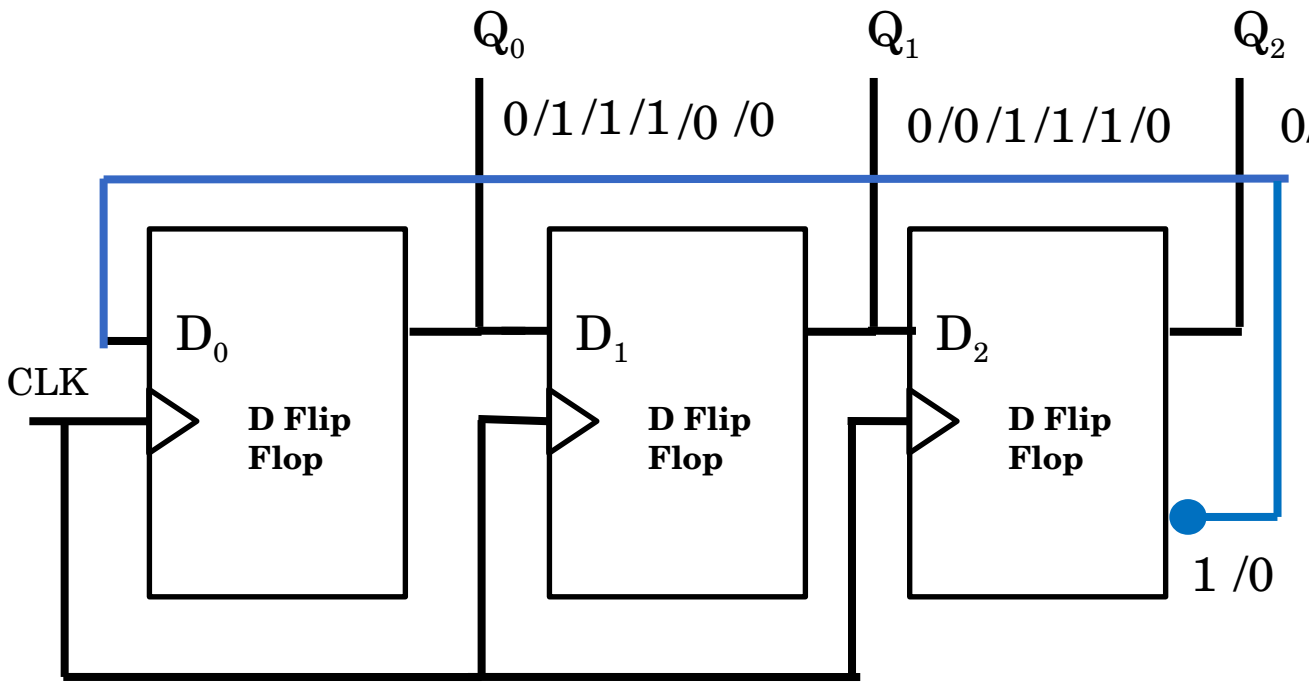
Q_3 Q_2 Q_1 Q_0

Right Shift

Serial in Serial out(SISO) – 4 bit



Johnson Counter– 3 bit

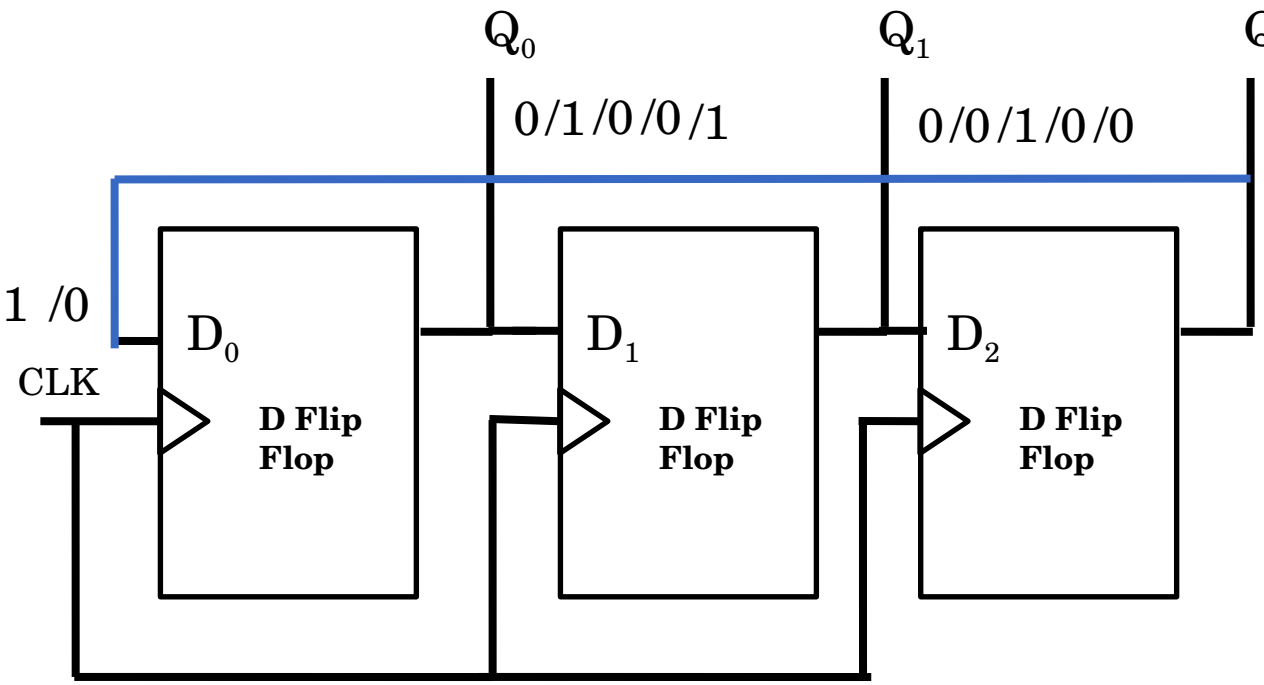


Q2	Q1	Q0
0	0	0
0	0	1
0	1	1
1	1	1
1	1	0
1	0	0
0	0	0

Input n bit so output 2n bit.
(3 bit input so it can count 2x3=6 values)

Applications(read yourself)

Ring Counter– 3 bit

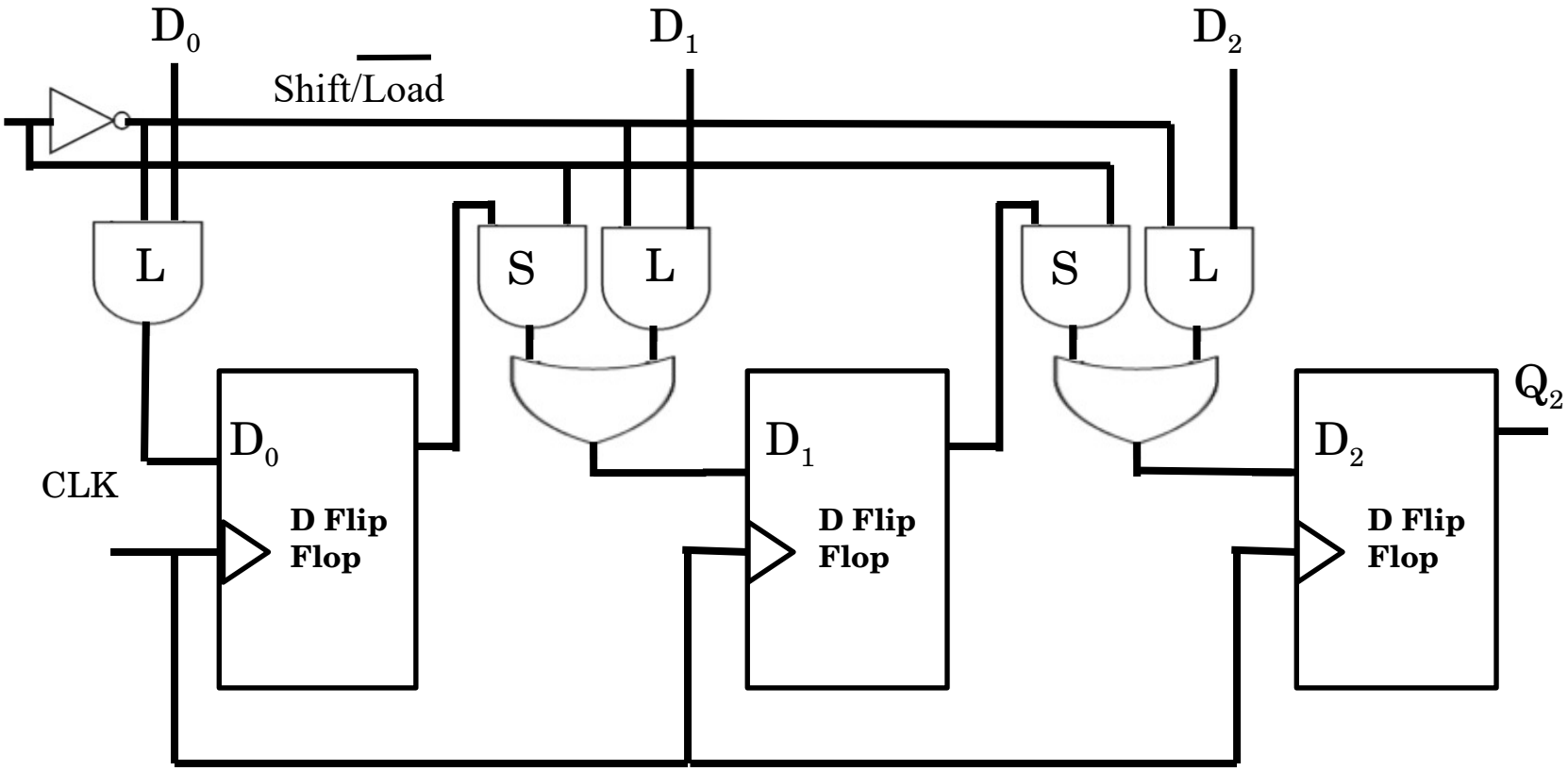


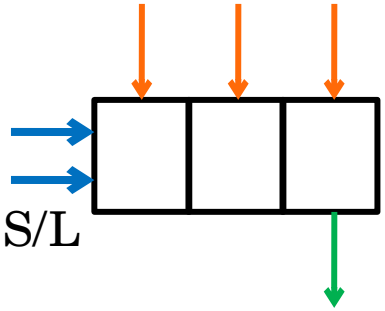
Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
1	0	0
0	0	1

Input n bit so output n bit.
(3 bit input so it can count 3 values)

Applications(read yourself)

Parallel in Serial out(PISO) – 3 bit



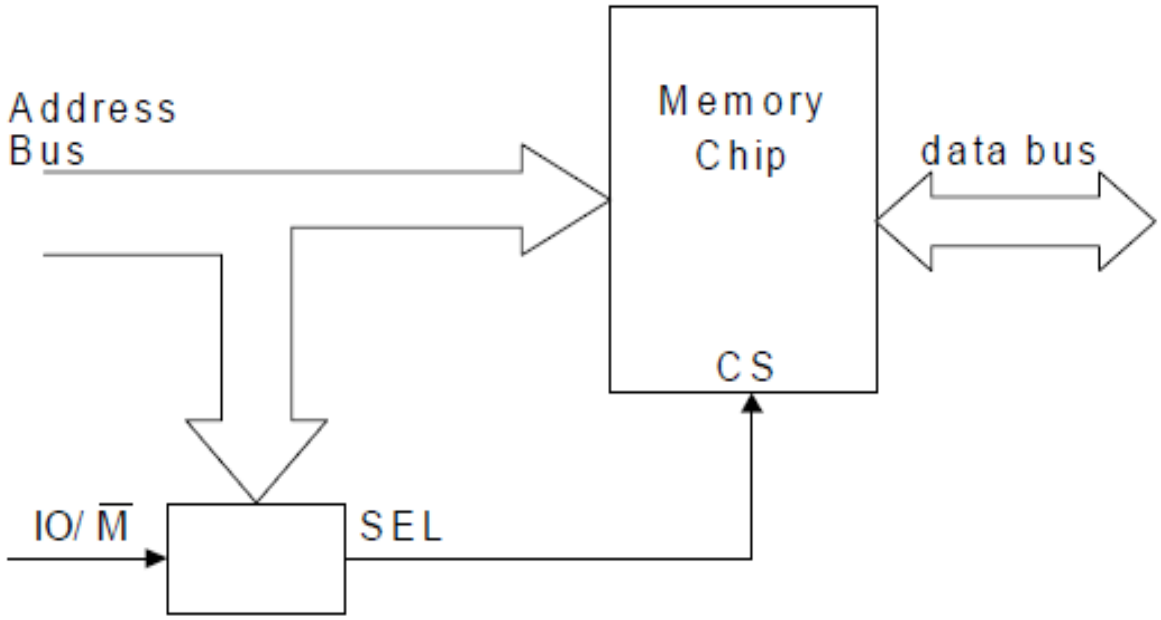


CSE 219: DIGITAL LOGIC & COMPUTER DESIGN

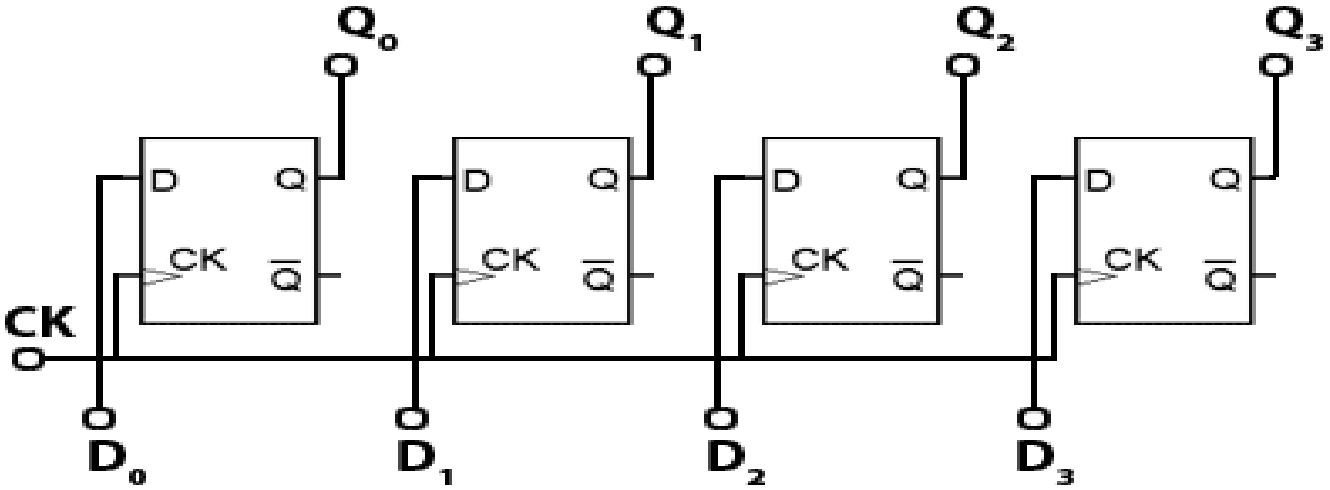
Tamanna Haque Nipa
Assistant Professor
Dept. of CSE
Stamford University Bangladesh

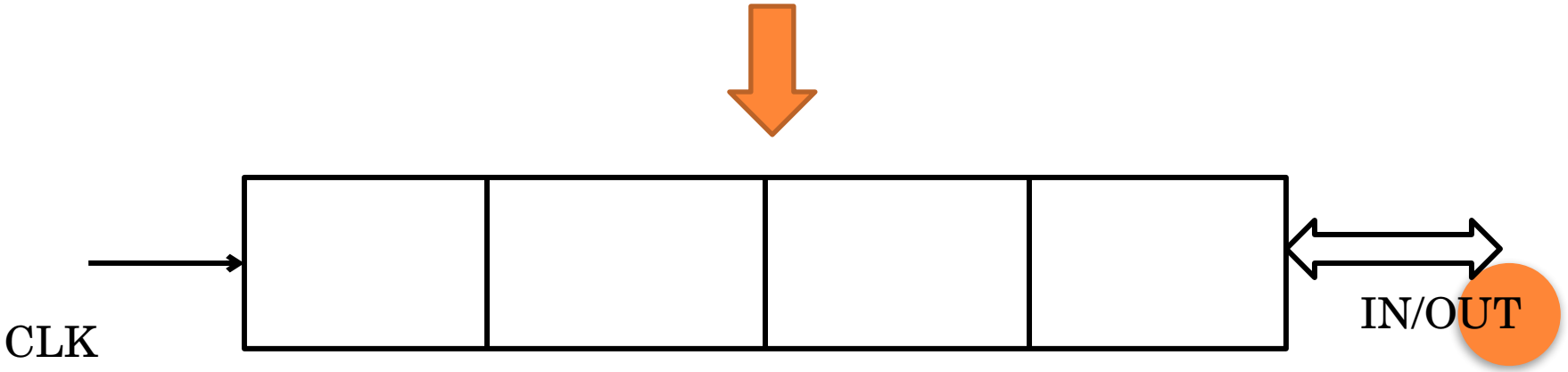
Memory Address Decoding

- *Address decoding* refers to the way a computer system decodes the addresses on the address bus to select memory locations in one or more memory or peripheral devices.

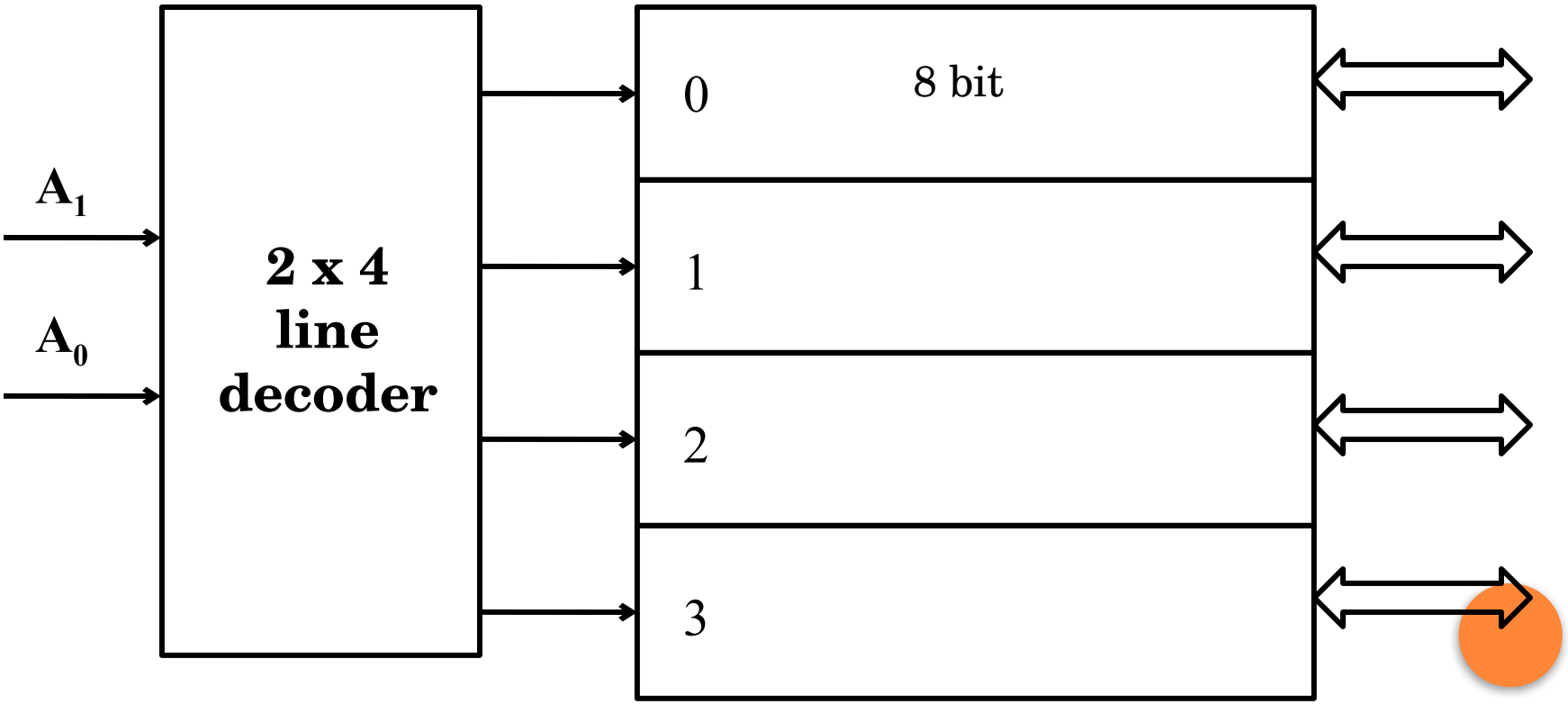


PIPO – 4 BIT

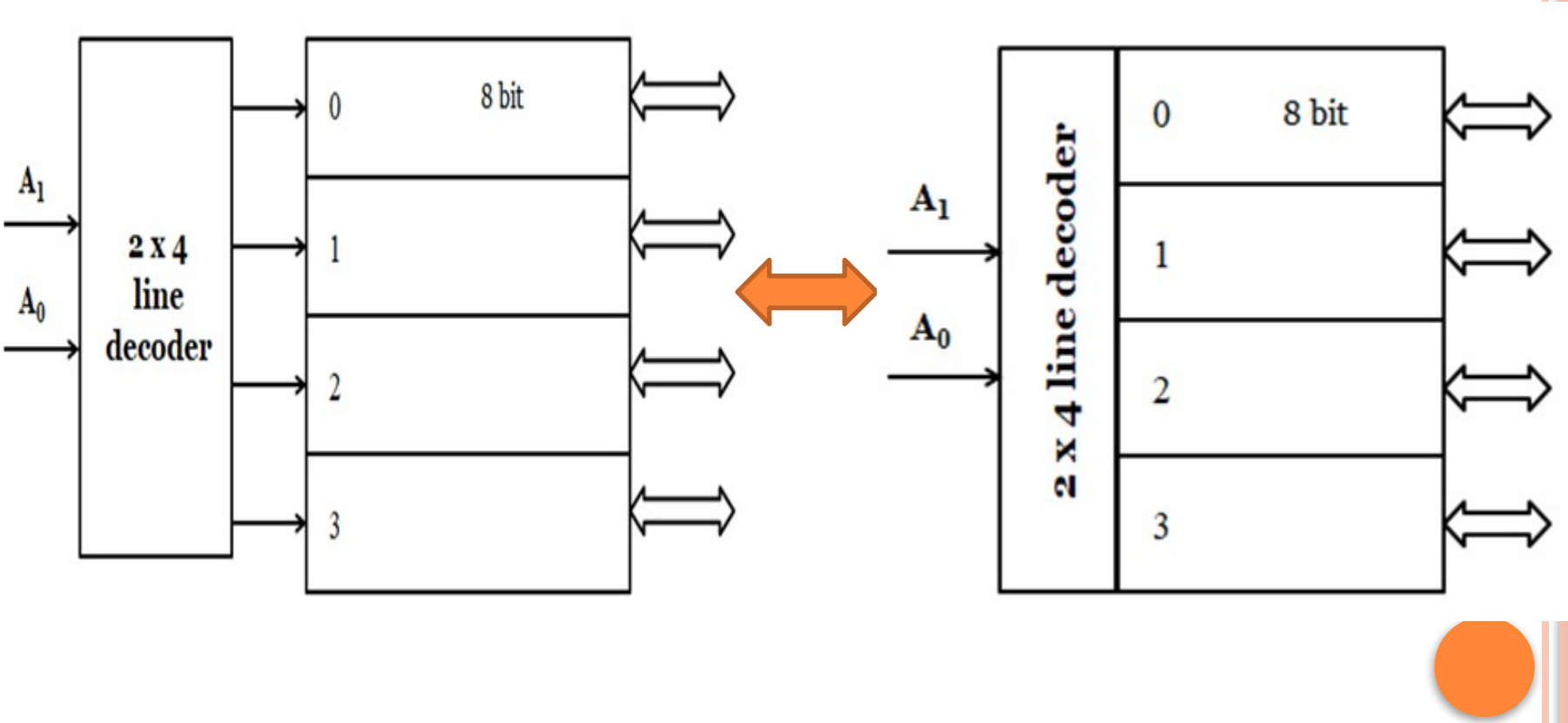




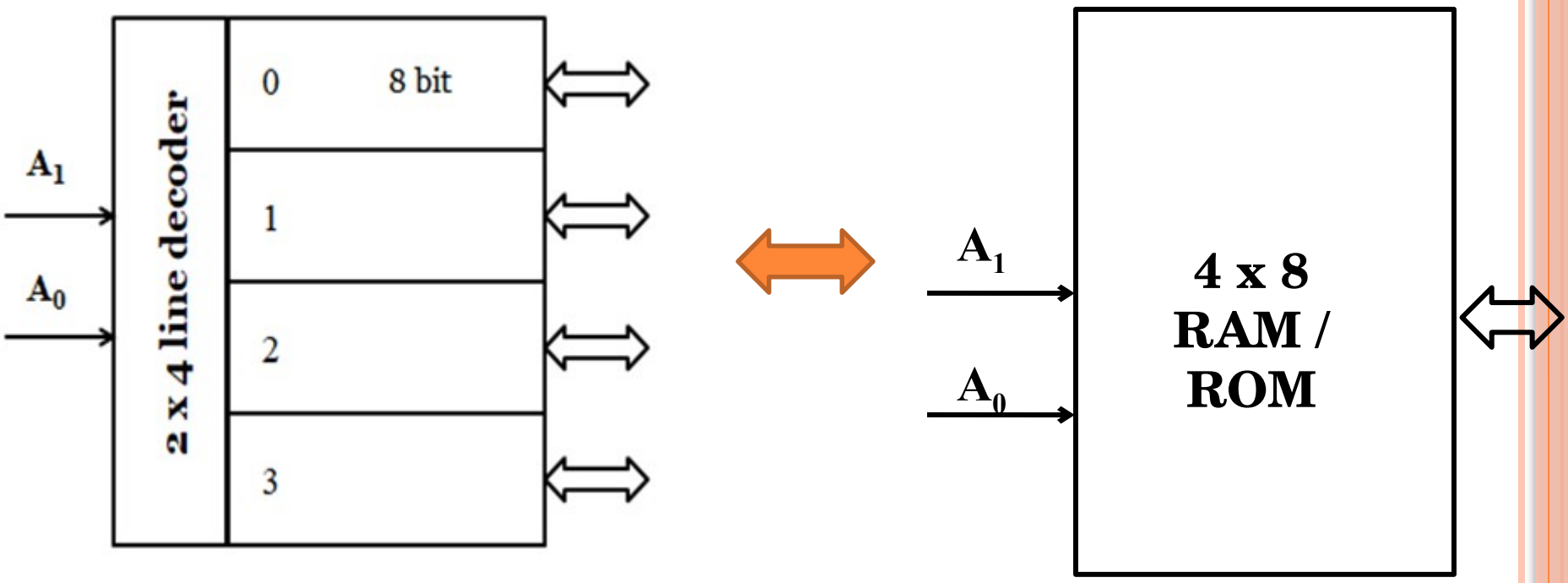
ADDRESS DECODER (4 X 8)



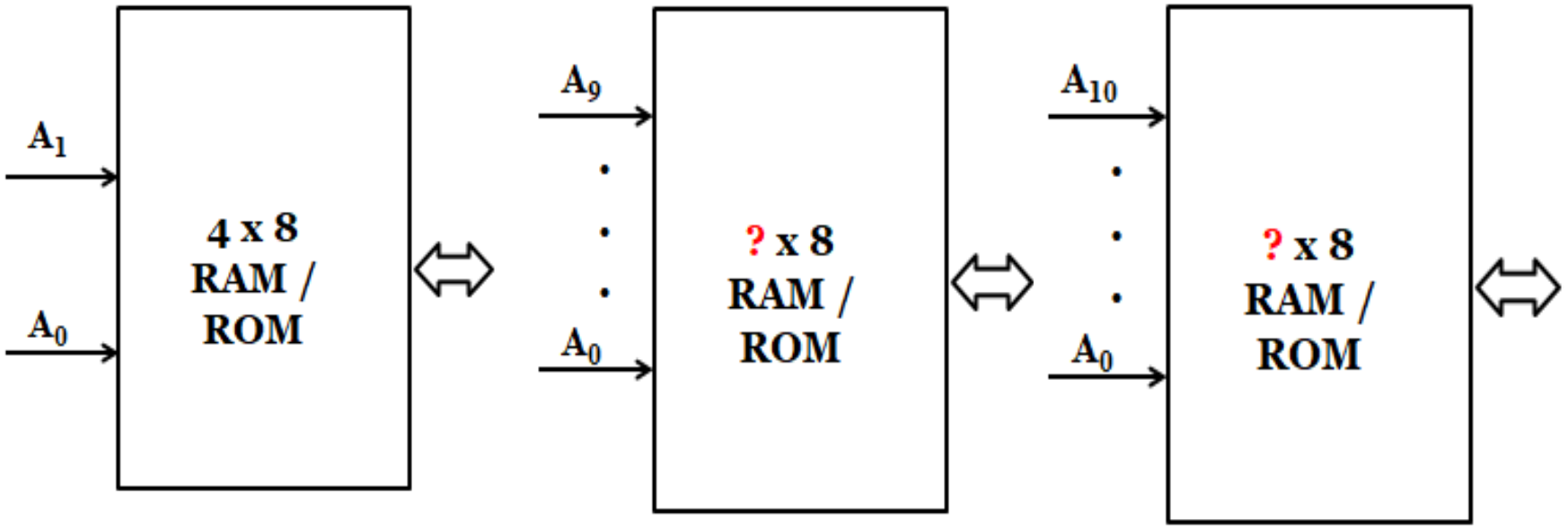
ADDRESS DECODER (4 X 8)



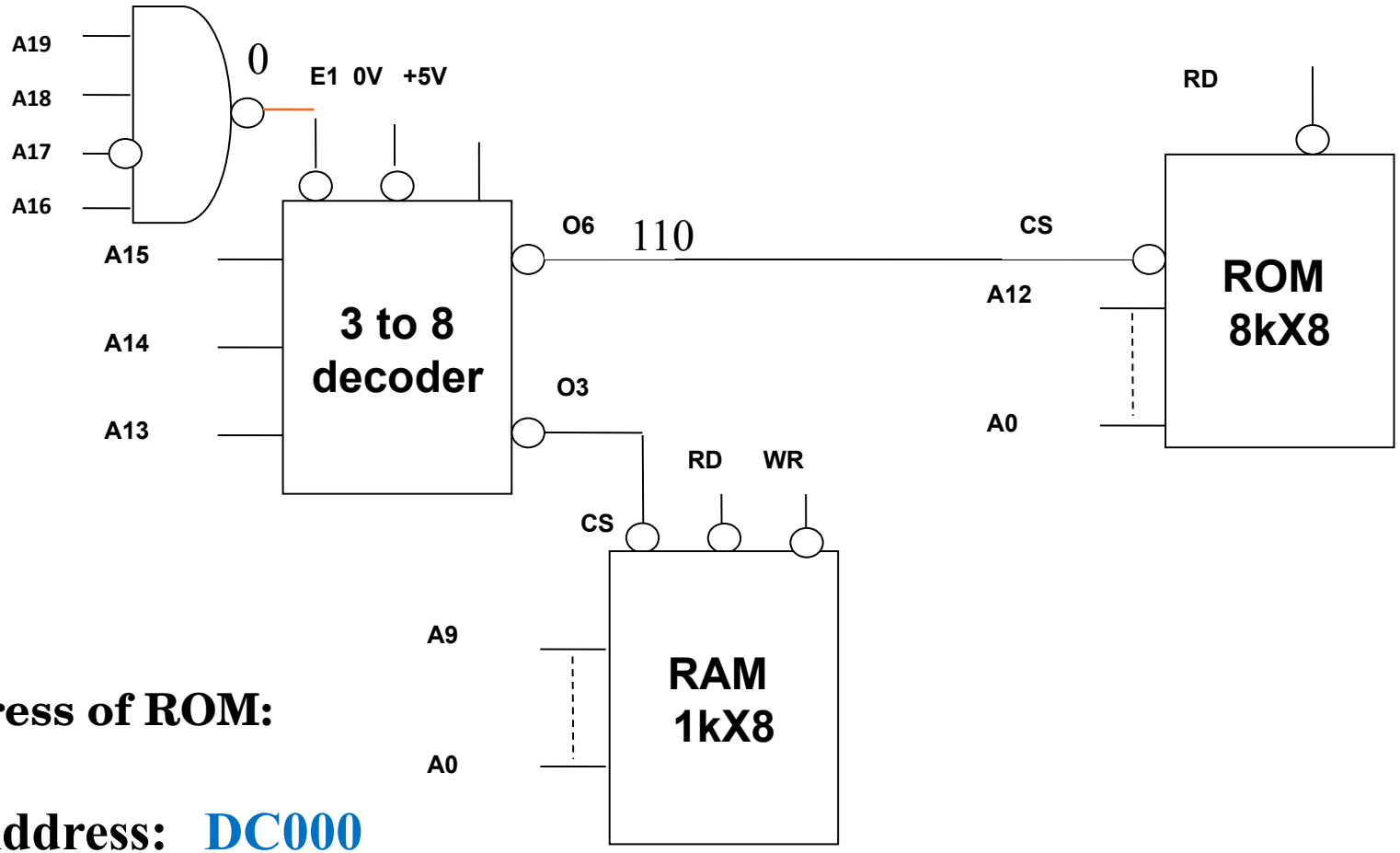
ADDRESS DECODER (4 X 8)



ADDRESS DECODER



- **Address A1 and A0 (2 bit) then addressable memory $2^2 = 4$**
- **Address A0 to A9 (10 bit) then addressable memory $2^{10} =$ 1024 (1 K)**
- **Address A0 to A10 (11 bit) then addressable memory $2^{11} =$ 2048 (2 K)**



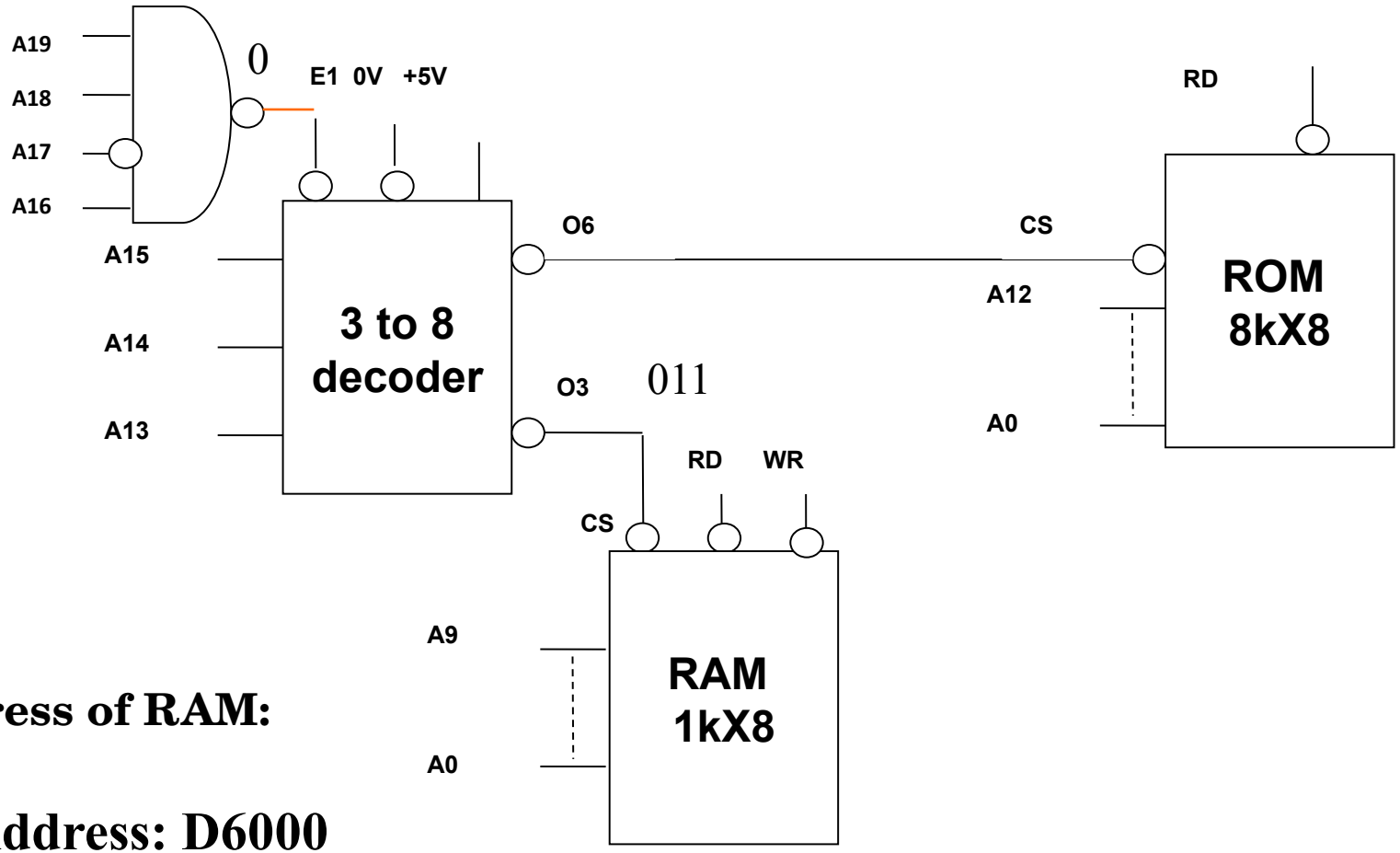
Address of ROM:

1st Address: DC000

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Last Address: DDFFF

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1



Address of RAM:

1st Address: D6000

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Last Address: D63FF

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	0	1	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1

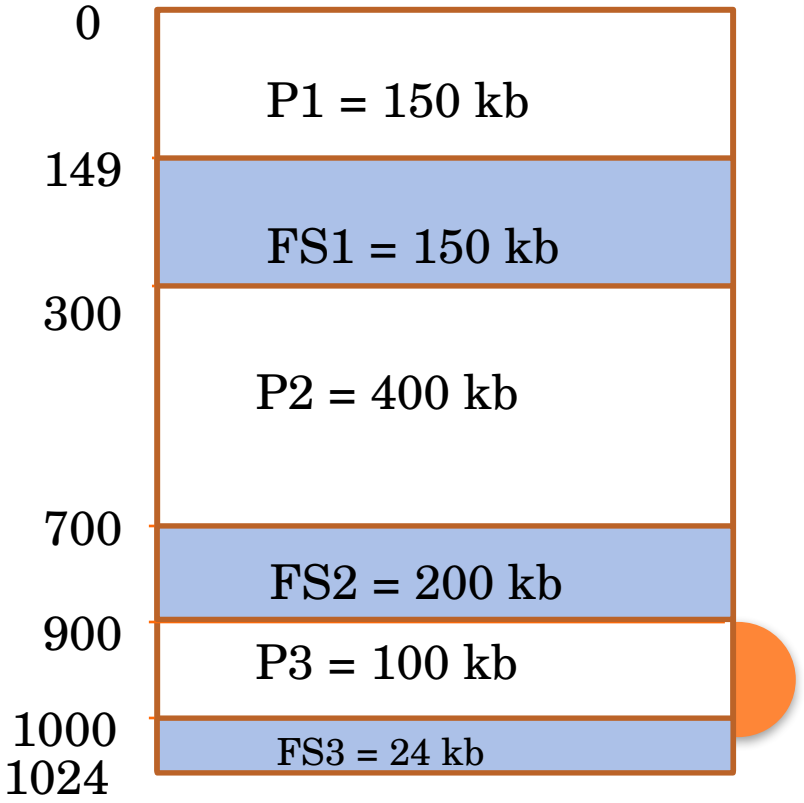
Memory Allocation

- Memory allocation is a process by which computer programs are assigned memory or space.
- Types
  - Non-Preemptive
    - First Fit
    - Best Fit
    - Worst Fit
  - Preemptive
    - First In First Out (FIFO)
    - Least Recently Used(LRU)
    - Optimal(OPT)

NON-PREEMPTIVE-EXAMPLE

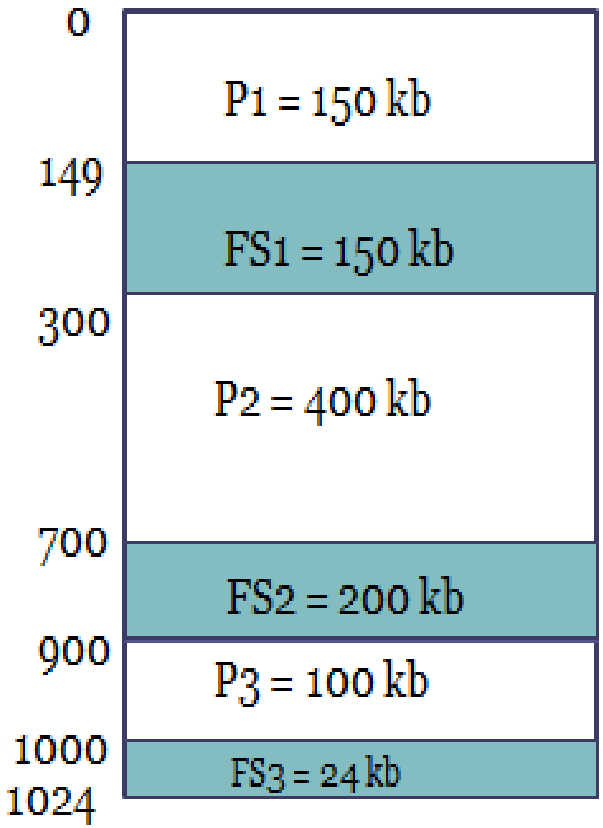
- There are 5 process P1 to P5. P1, P2 and P3 has been already allocated and P4 and P5 need to allocated following the 3 algorithms. Find which one is best?

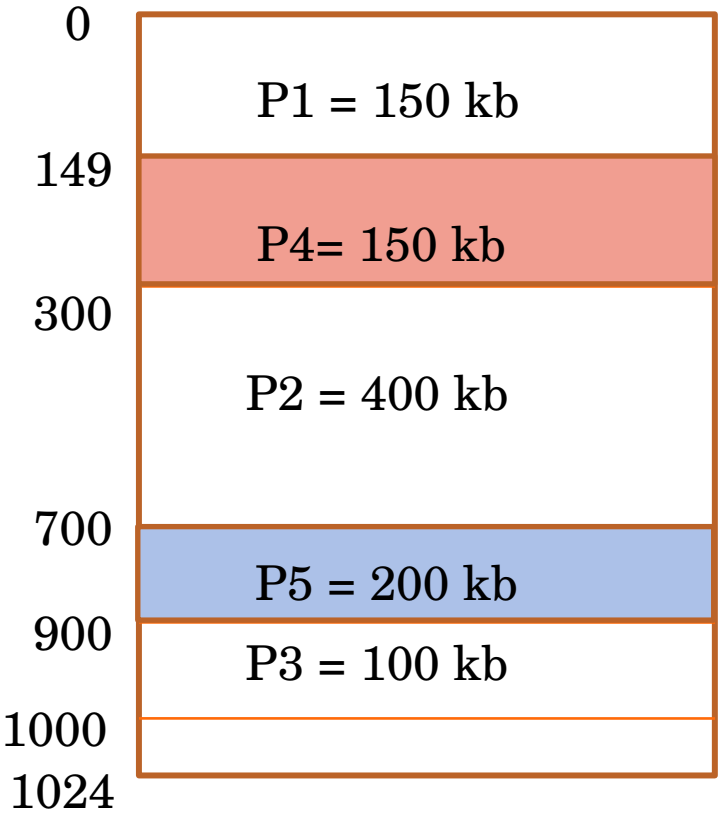
Process	Size (kb)	Address
P1	150	0
P2	400	300
P3	100	900
P4	150	
P5	200	



FIRST FIT-EXAMPLE

- Need to fit P4=150 kb and P5= 200kb



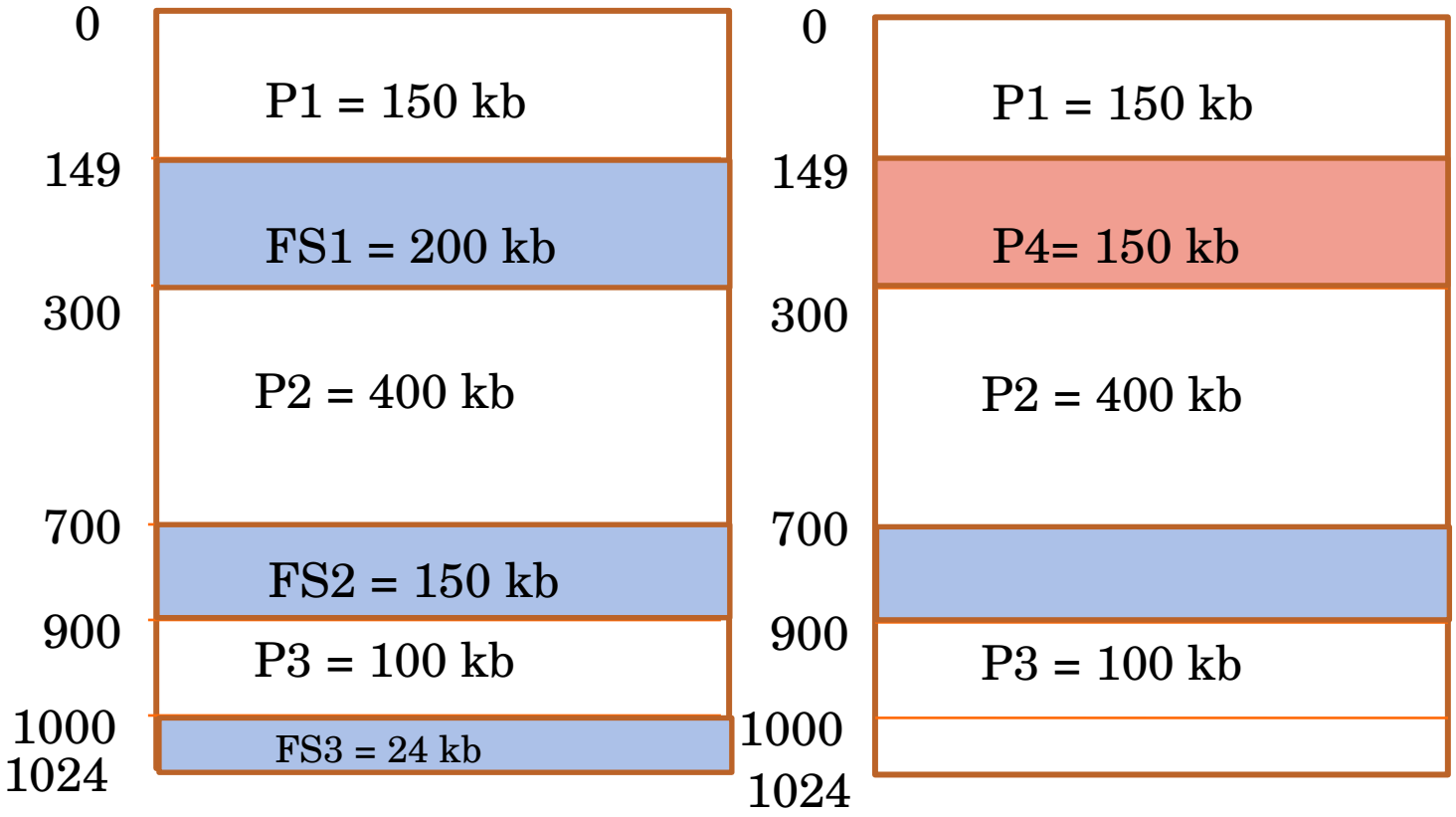


Choose the first adjustable option

- Select FS1(150 kb) / FS2(200kb) / FS3(24kb) for P4= 150kb???
- It will select FS1 then P5 will adjust with FS2

FIRST FIT-EXAMPLE

- Need to fit P4=150 kb and P5= 200kb

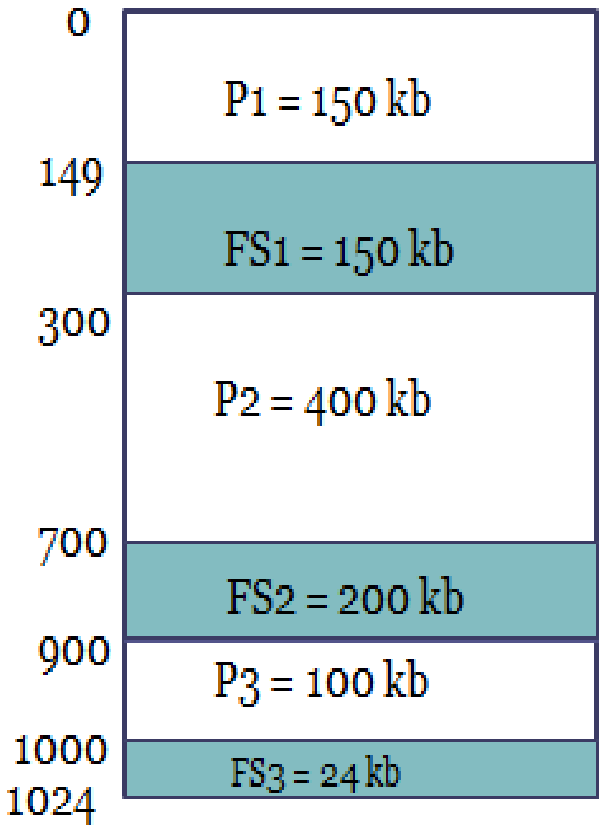


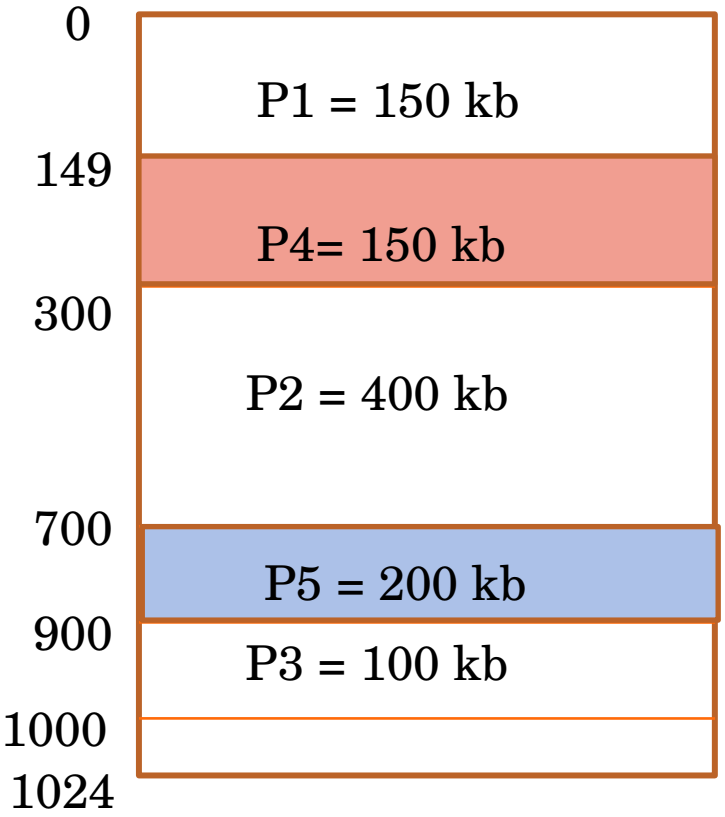
Choose the first adjustable option

- Select FS1(150 kb) / FS2(200kb) / FS3(24kb) for P4= 150kb???
- It will select FS1 then P5 will adjust with FS2

BEST FIT-EXAMPLE

- Need to fit P4=150 kb and P5= 200kb



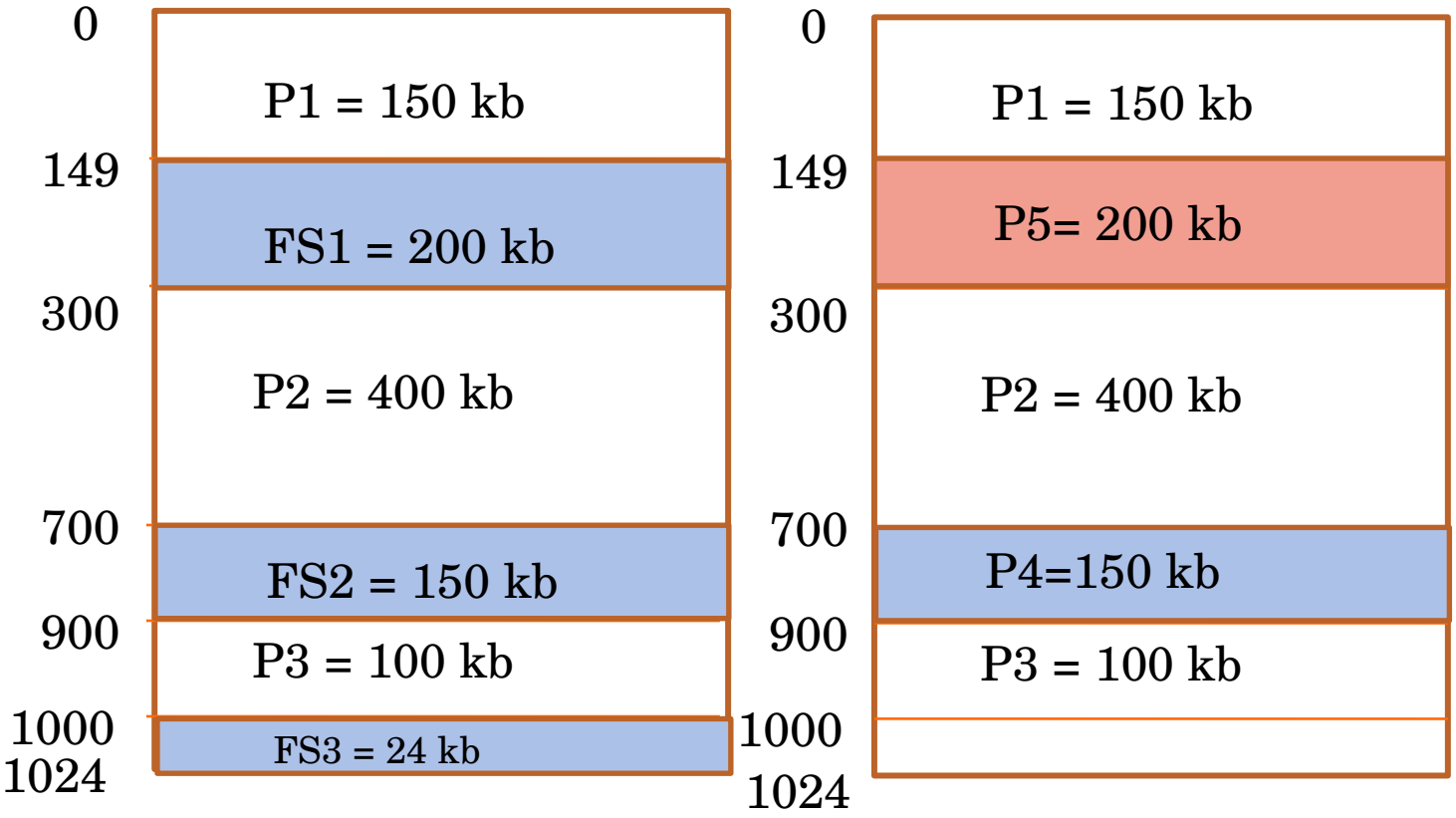


Choose the best adjustable option

- Select FS1(150 kb) / FS2(200kb) / FS3(24kb) for P4= 150kb???
- It will select FS1 then P5 will adjust with FS2

BEST FIT-EXAMPLE

- Need to fit P4=150 kb and P5= 200kb

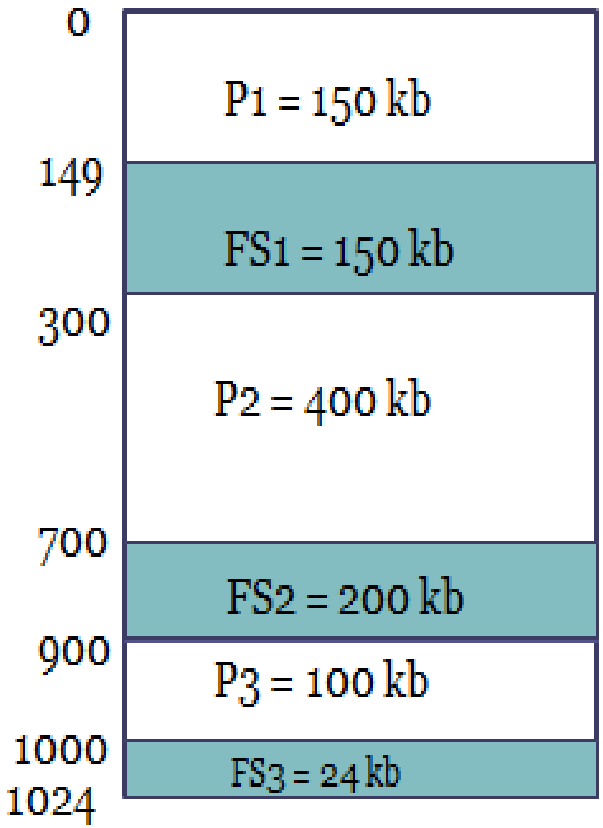


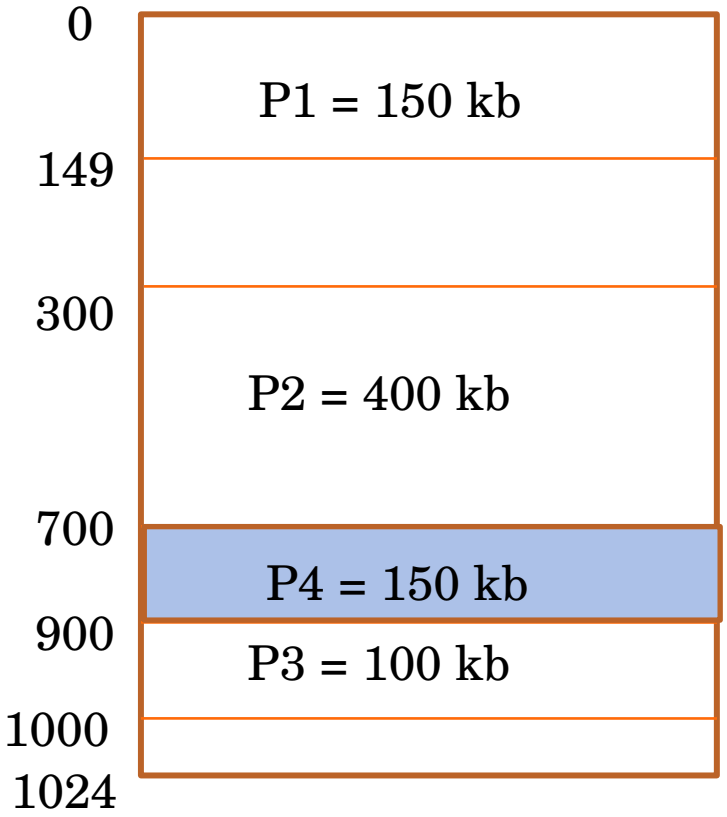
Choose the best adjustable option

- Select FS1(150 kb) / FS2(200kb) / FS3(24kb) for P4= 150kb???
- It will select FS2 for P4 and then P5 will adjust with FS2

WORST FIT-EXAMPLE

- Need to fit P4=150 kb and P5= 200kb





Choose the highest free space adjustable option

- Select FS1(150 kb) / FS2(200kb) / FS3(24kb) for P4= 150kb???
- P4 will select FS2 then P5 need to wait.

PREEMPTIVE ALLOCATION-EXAMPLE

Consider the following page address traces generated by CPU with 3 available frames. Find out the action generated for FIFO, LRU and OPT replacement policies. Which algorithm is best?

1, 3, 1, 4, 7, 4, 5, 3, 2, 1, 2, 1, 3, 4, 6, 7

FIFO → hit =5

1	3	*1	4	7	* 4	5	3
1	1	1	1	7	7	7	7
	3	3	3	3	3	5	5
			4	4	4	4	3

2	1	* 2	* 1	*3	4	6	7
2	2	2	2	2	2	6	6
5	1	1	1	1	1	1	7
3	3	3	3	3	4	4	4

LRU → hit =5

1	3	* 1	4	7	* 4	5	3
1	1	1	1	1	1	5	5
	3	3	3	7	7	7	3
			4	4	4	4	4

2	1	* 2	* 1	* 3	4	6	7
3	3	3	3	3	3	3	7
5	1	1	1	1	1	6	6
2	2	2	2	2	4	4	4

OPT → hit =6

1	3	* 1	4	7	* 4	5	* 3
1	1	1	1	7	7	5	5
	3	3	3	3	3	3	3
			4	4	4	4	4

2	1	* 2	* 1	* 3	4	6	7
2	2	2	2	2	4	6	7
3	3	3	3	3	3	3	3
4	1	1	1	1	1	1	1