



UNIVERSITI  
TEKNOLOGI  
MALAYSIA  
[www.utm.my](http://www.utm.my)

# SCSR1013 DIGITAL LOGIC

## MODULE 9: SHIFT REGISTERS

SCHOOL OF COMPUTING



# MODULE 9

## SHIFT REGISTERS

### Objectives:

1. To introduce the characteristics and function of a shift register
2. To highlight different types of shift register
3. To introduce shift register IC
4. To explain shift register counter

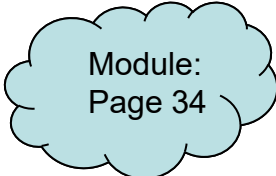
# MODULE CONTENT

---

- Basic Shift Register functions
- Shift Register:
  - Serial In/Serial Out (SISO)
  - Serial In/Parallel Out (SIPO)
  - Parallel In/Serial Out (PISO)
  - Parallel In/Parallel Out (PIPO)
- Shift Register counters
  - Ring Counter
  - Johnson Counter

## Basic Shift Register (SRG) Functions

- **SRG** is a register in which binary data can be **stored**, and this data can be **shifted** to the left or right when a signal is applied.
- It consist of arrangements of **FFs**.

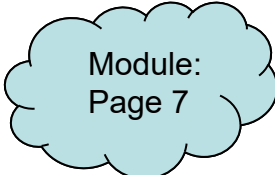


Module:  
Page 34



Memory  
Devices in  
Digital  
Systems

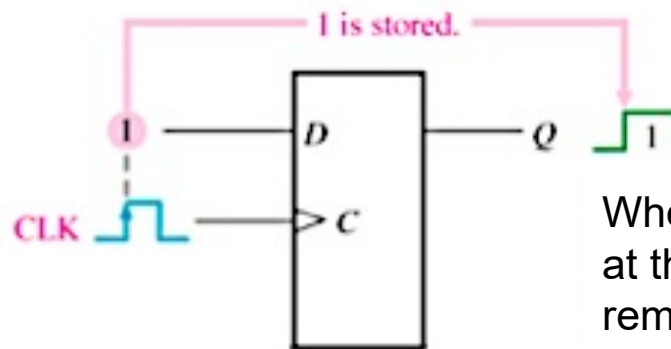
- Digital System : Important in application involving:
  - a) Data storage
  - b) Data transfer / movement



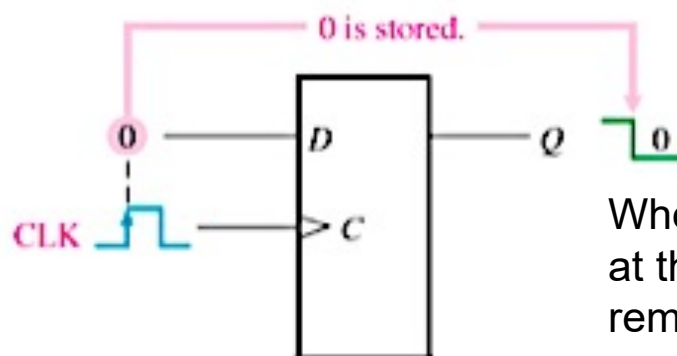
Module:  
Page 7

## a ) The flip-flop as a storage element

- Each FF can store 1 bit data.
- More bits require more FF.



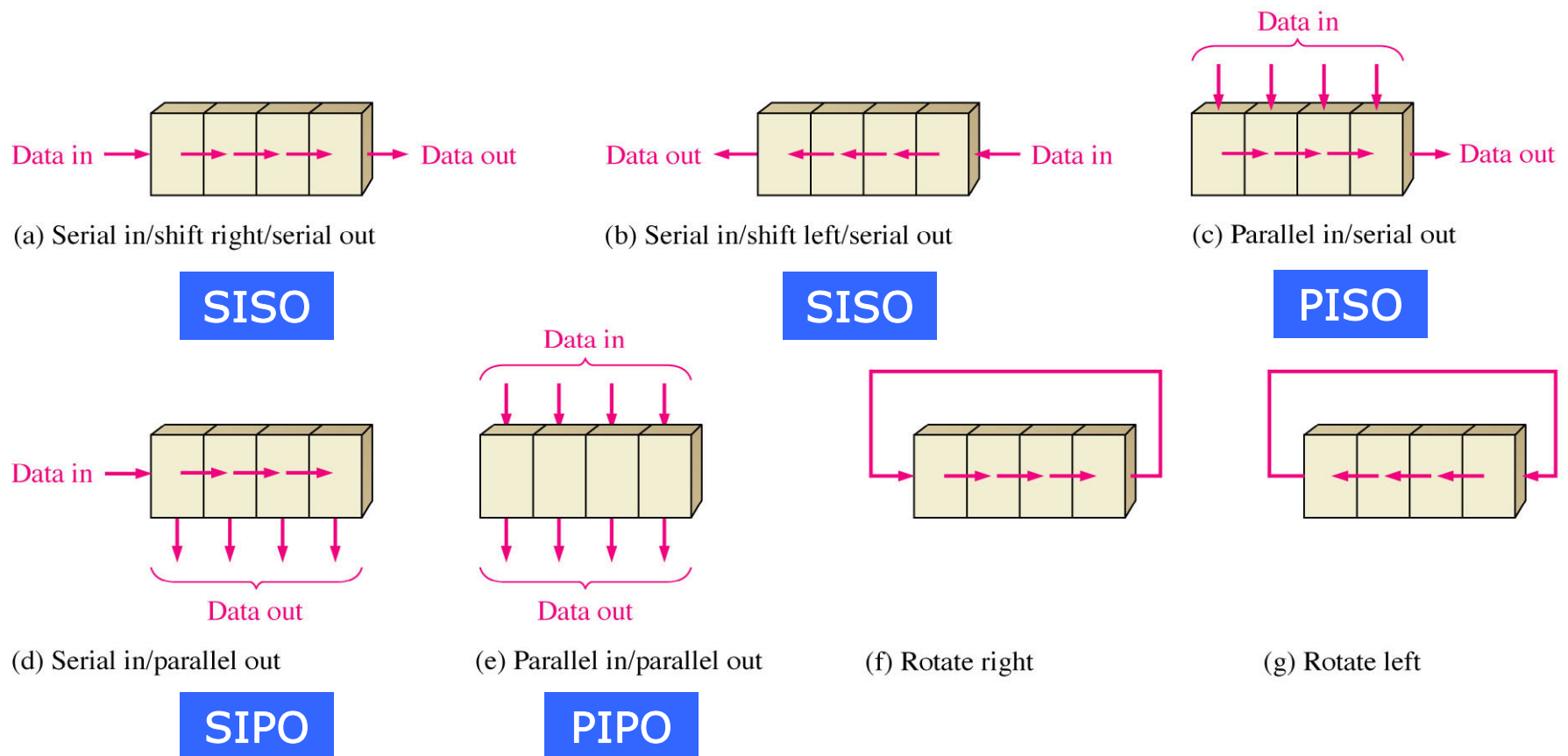
When a 1 is on  $D$ ,  $Q$  becomes a 1 at the triggering edge of  $CLK$  or remains a 1 if already.



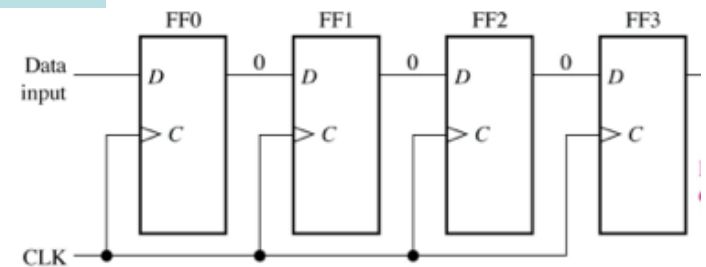
When a 0 is on  $D$ ,  $Q$  becomes a 0 at the triggering edge of  $CLK$  or remains a 0 if already.

## b ) Basic Data Transfer / Movements in SRG.

(Four bits are used for illustration. The bits move in the direction of the arrows.)



- A register can shift data one bit at a time.
- The **logical configuration** of a serial SRG:
  - A chain of FF connected in **cascade**.
- The operation of SRG: **Synchronous**.
  - Using common clock.

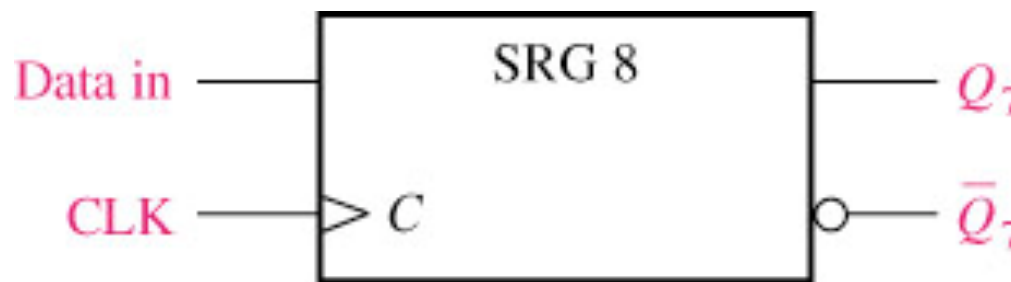


**Problem** : Shifting  $N$  bit SRG will lose all data after  $2N$  clock cycles.

**Solution** : Using rotate.

## (a) SISO

- Accepts data serially, one bit (input) at a time on a single line
- Each clock pulse will enter one bit data (**Data in**) into SRG and at the same time one bit will be shifted to  $Q_7$
- Output produced in serial form too.



**Figure :** Logic symbol for an 8-bit serial in/serial out shift register

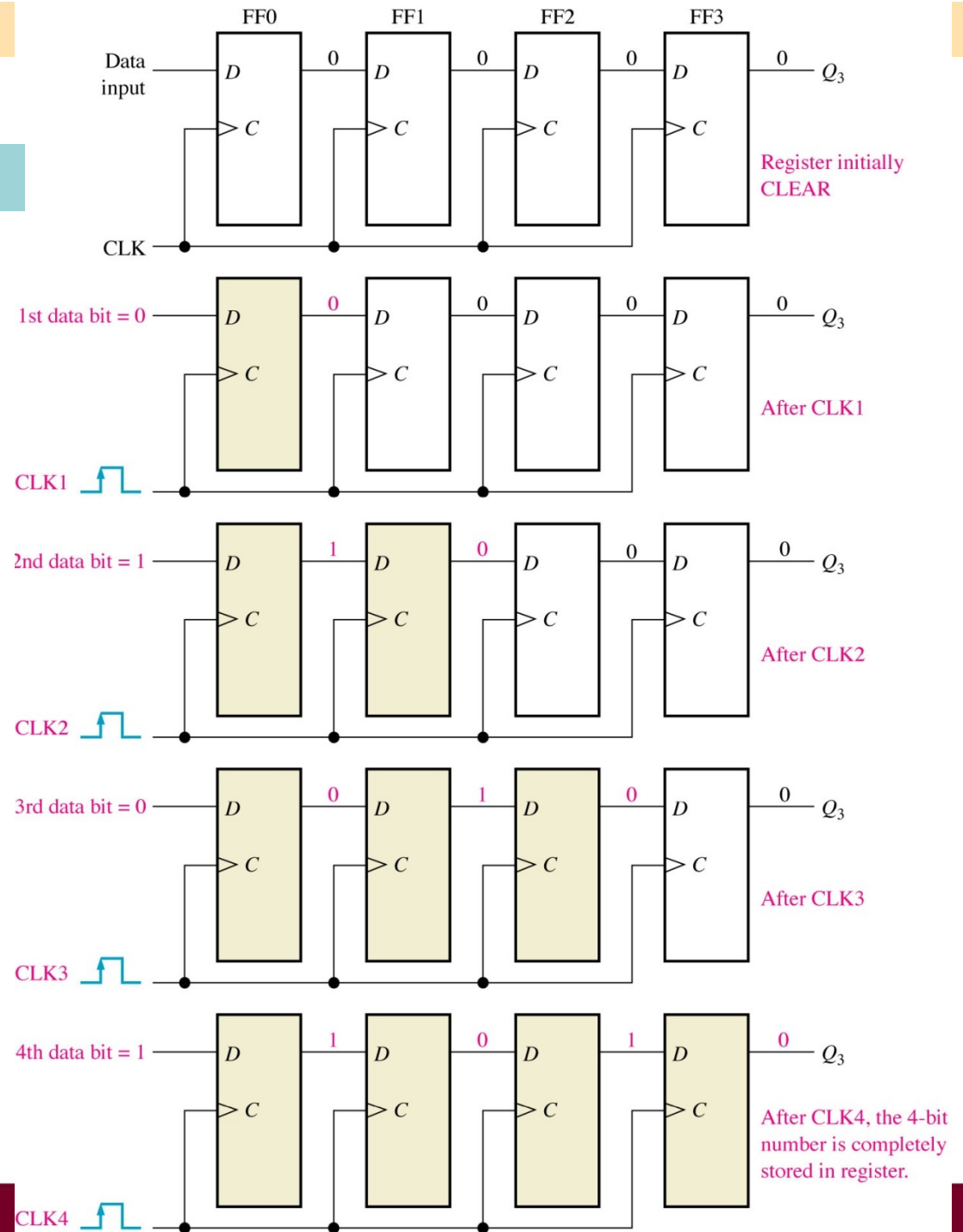


## Example 1:

LSB shifted in first.

1010

- Four bits (1010) being entered serially into the register
- Every clock cycle a data is shifted to the right one position
- For n bit SISO it requires n clock cycle to completely entering the shift register
  - i.e. for the first data entered to appear at the output



**Figure :** Four bits (1010) being serially **shifted out** of the register and replaced by all zeros.

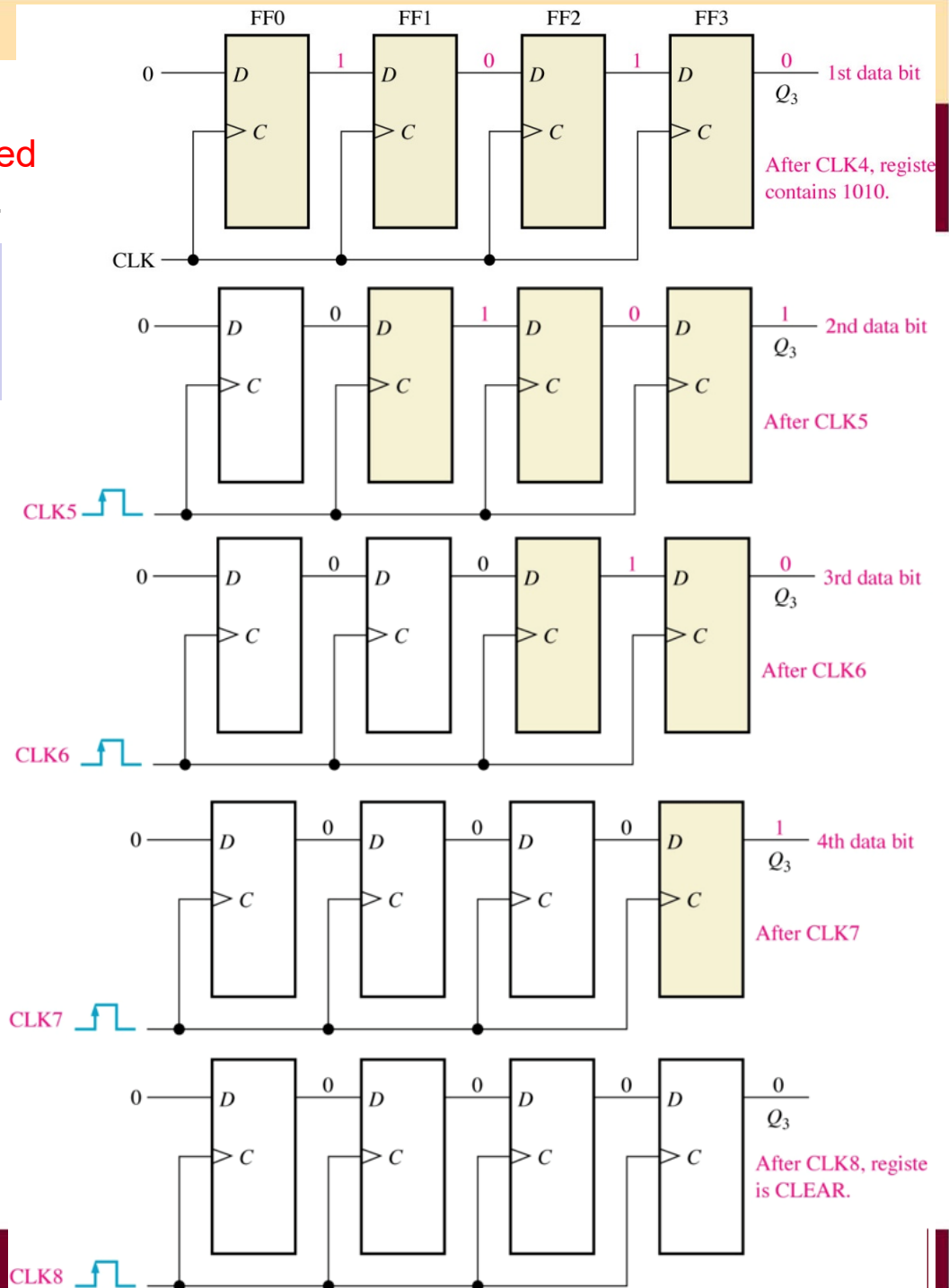
Output:

1 0 1 0

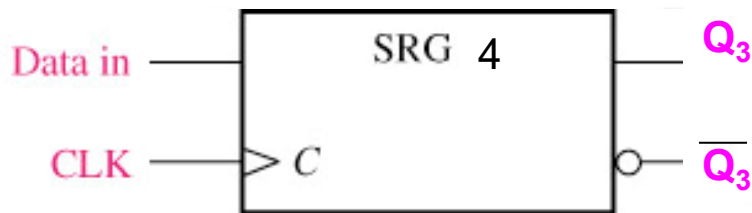
- Once all data is shifted in, every clock cycle will make the data shifted another one position to the right
- On the 5<sup>th</sup> clock cycle, the first 0 is shifted out and lost, its position replaced by the 2<sup>nd</sup> data, at the left most flip-flop new data entering the SISO
- This kind of shifting repeats every clock cycle.
- Assuming every time a new data is 0, at the 8<sup>th</sup> clock cycle all flip-flop in SISO will be replaced by a new data

For  $n$ -bit SISO, it requires:

- $n$  clock cycle to **shifted in** all data
- another  $n$  clock cycle to **shifted out** all data.



## Example 2: SISO



MSB  
1101  
LSB

Data shifted in:

Clock, t	FF0	FF1	FF2	FF3
Initially	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	0	1	1	0
4	1	0	1	1

Bit **MSB** will insert **first**.  
The last FF will hold MSB

Data shifted out:

Clock, t	FF0	FF1	FF2	FF3	Data lost
Initially	1	0	1	1	X
5	0	1	0	1	1
6	0	0	1	0	1
7	0	0	0	1	0
8	0	0	0	0	1

Bit MSB will go out first

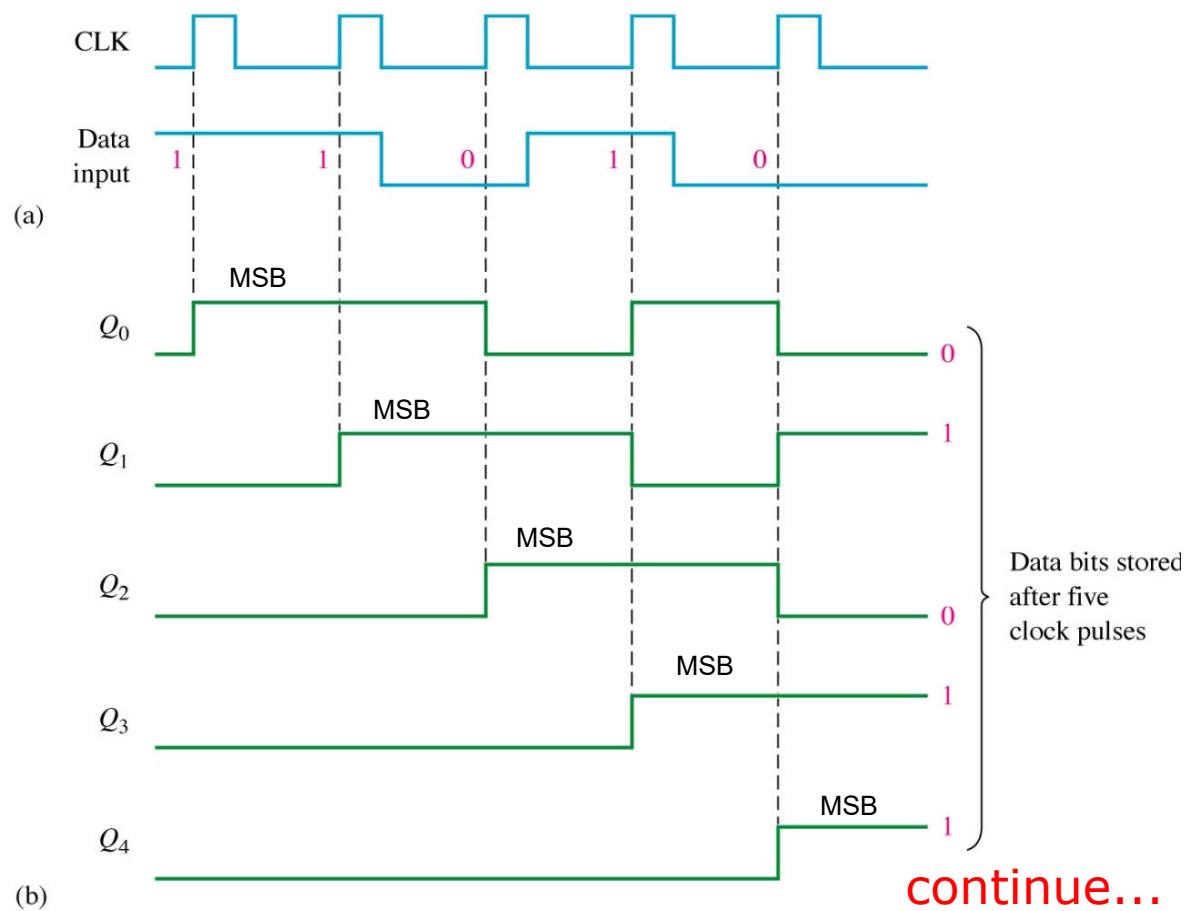
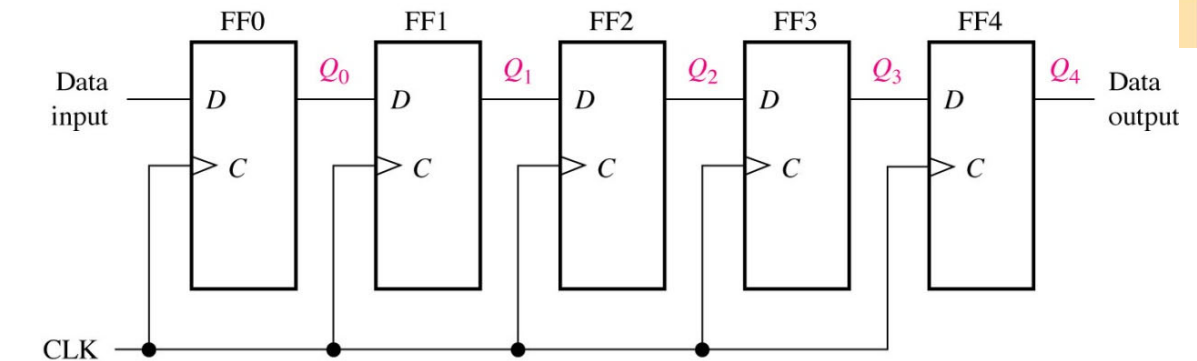
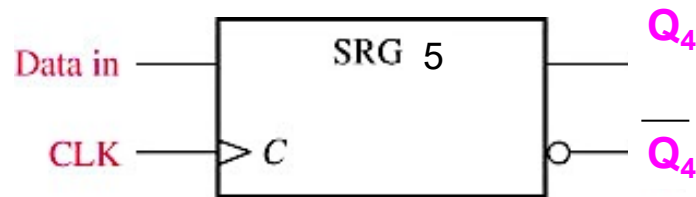
### Example 3:

5-bit SISO (Right SRG)

Data input: **11010**

Initially all FFs are cleared.

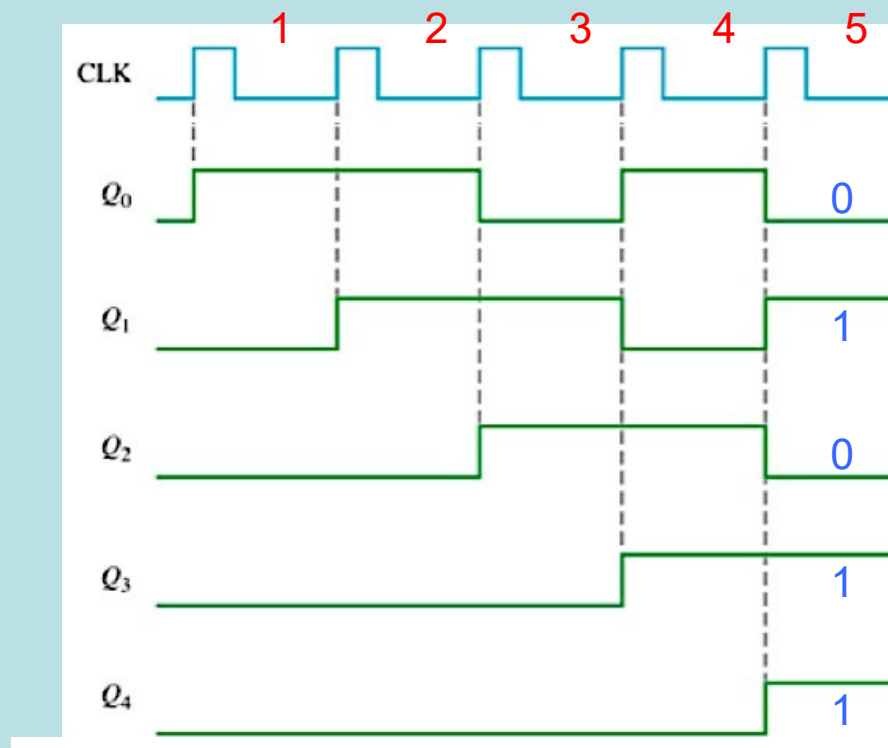
**MSB** are shifted in **first**.



Data input: **11010**

Self-Test:

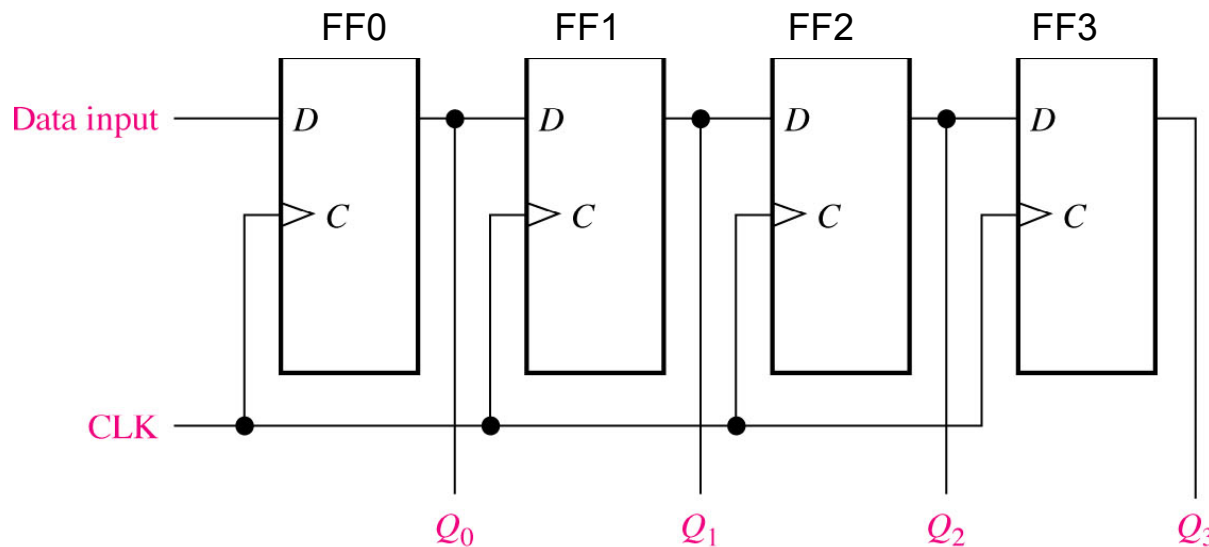
Generate a table for  
the data **shifting out**.  
(**MSB** shifted out **first**)



Clock, t	FF0	FF1	FF2	FF3	FF4
Initially	0	1	0	1	1
6	0	0	1	0	1
7	0	0	0	1	0
8	0	0	0	0	1
9	0	0	0	0	0
10	0	0	0	0	0

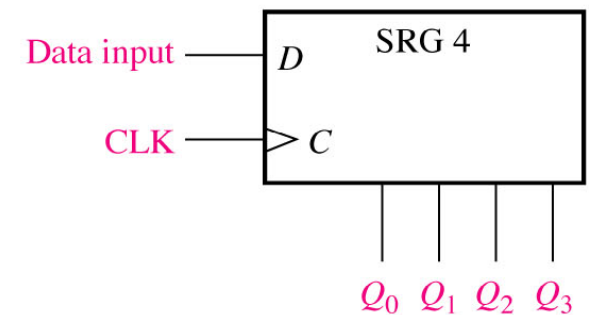
## (b) SIPO

- Data bits are **entered serially** one bit at a time.
- **Output** is in **parallel**
  - All bits are available **simultaneously** after  $n$  clock cycles for  $n$ -bit SIPO.



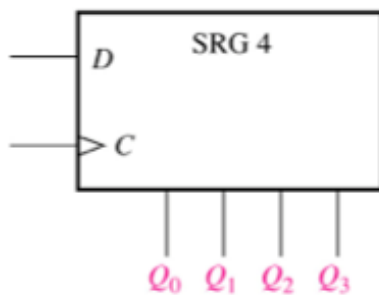
(a)

For SISO, output only at  $Q_3$ .  
For SIPO output  $Q_3Q_2Q_1Q_0$

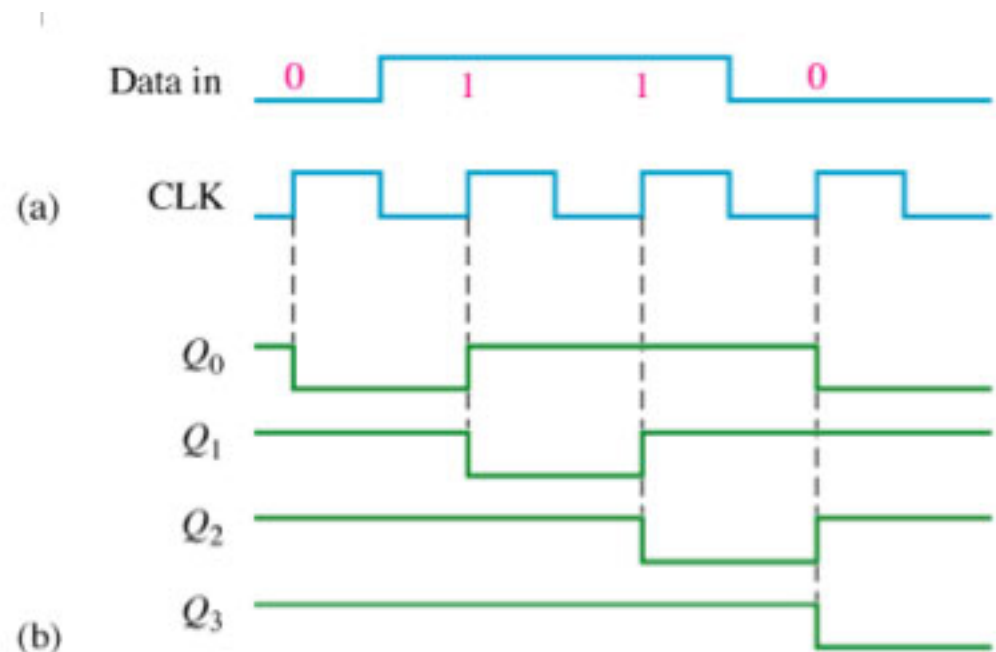


(b)

**Example 4:** Show the states of 4-bit SIPO shift register (SRG 4) for the data input **0110** and clock waveforms. The register initially all is 1s.



Clock, t	FF0	FF1	FF2	FF3
Initially	1	1	1	1
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	0	1	1	0



**Exercise 9.1:** Show the states of 6-bit SIPO shift register (SRG 6) for the data input **011001** and clock waveforms.  
The register initially all is 0s and MSB will enter first.



**Exercise 9.1:** Show the states of 6-bit SIPO shift register (SRG 6) for the data input **011001** and clock waveforms.

The register initially all is 0s and **MSB** will enter **first**.

**Solution :**

Clock, t	FF0	FF1	FF2	FF3	FF4	FF5
Initially	0	0	0	0	0	0
1	0	0	0	0	0	0
2	1	0	0	0	0	0
3	1	1	0	0	0	0
4	0	1	1	0	0	0
5	0	0	1	1	0	0
6	1	0	0	1	1	0



$Q_0$



$Q_1$



$Q_2$



$Q_3$



$Q_4$

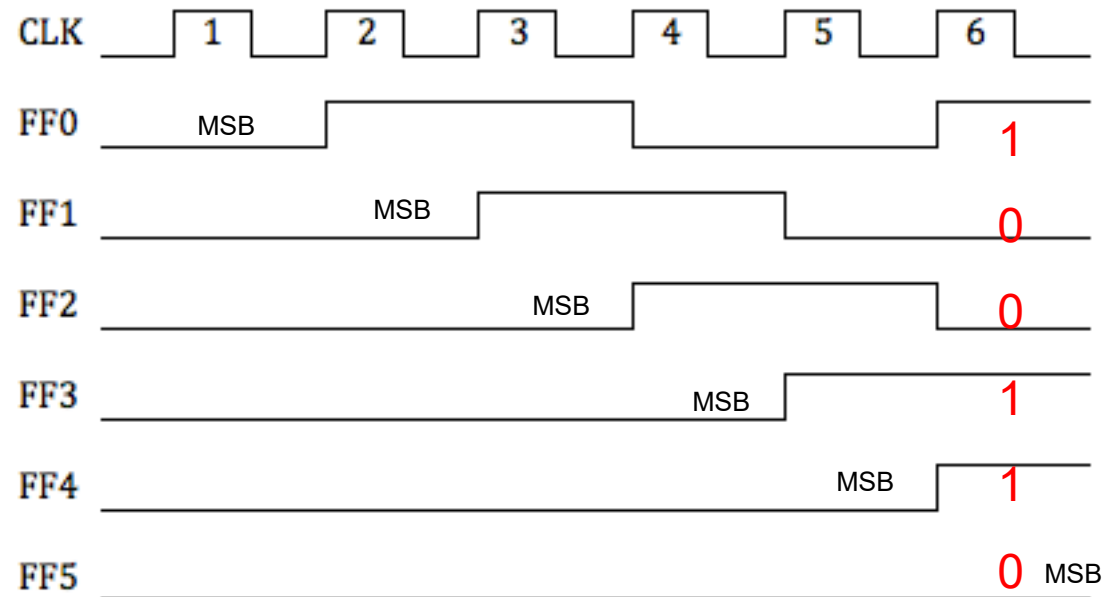


$Q_5$

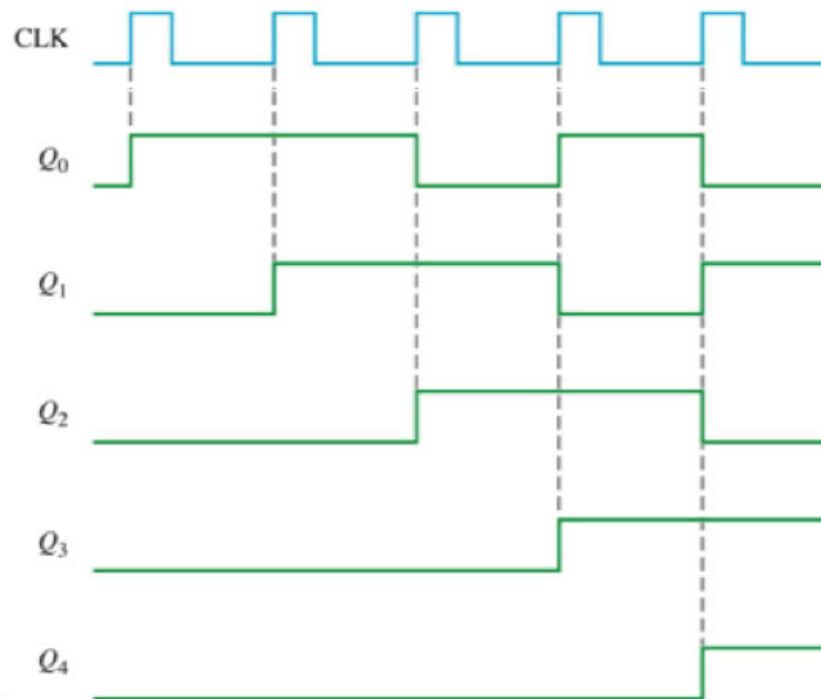
... data input **011001**

Clock, t	FF0	FF1	FF2	FF3	FF4	FF5
Initially	0	0	0	0	0	0
1	0	0	0	0	0	0
2	1	0	0	0	0	0
3	1	1	0	0	0	0
4	0	1	1	0	0	0
5	0	0	1	1	0	0
6	1	0	0	1	1	0

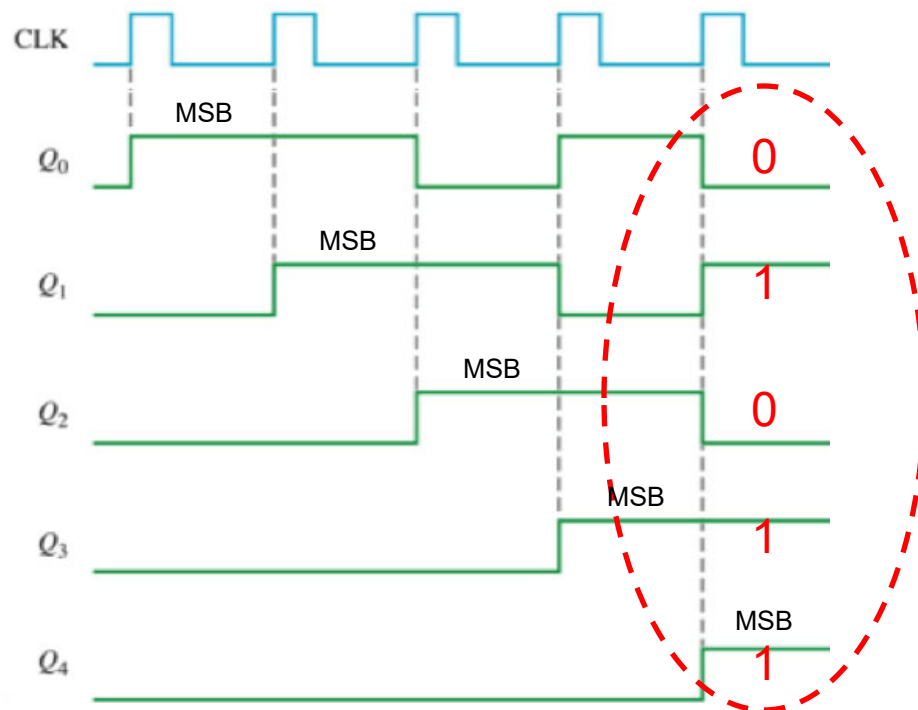
Clock waveform



- Exercise 9.2:**
- a) Determine how many bit entered to the SIPO register?
  - b) Circle on the timing diagram the valid output.
  - c) Determine the data entered. (Assume MSB shifted in first)



- Exercise 9.2:**
- a) Determine how many bit entered to the SIPO register?
  - b) Circle on the timing diagram the valid output.
  - c) Determine the data entered. (Assume **MSB** shifted in **first**)



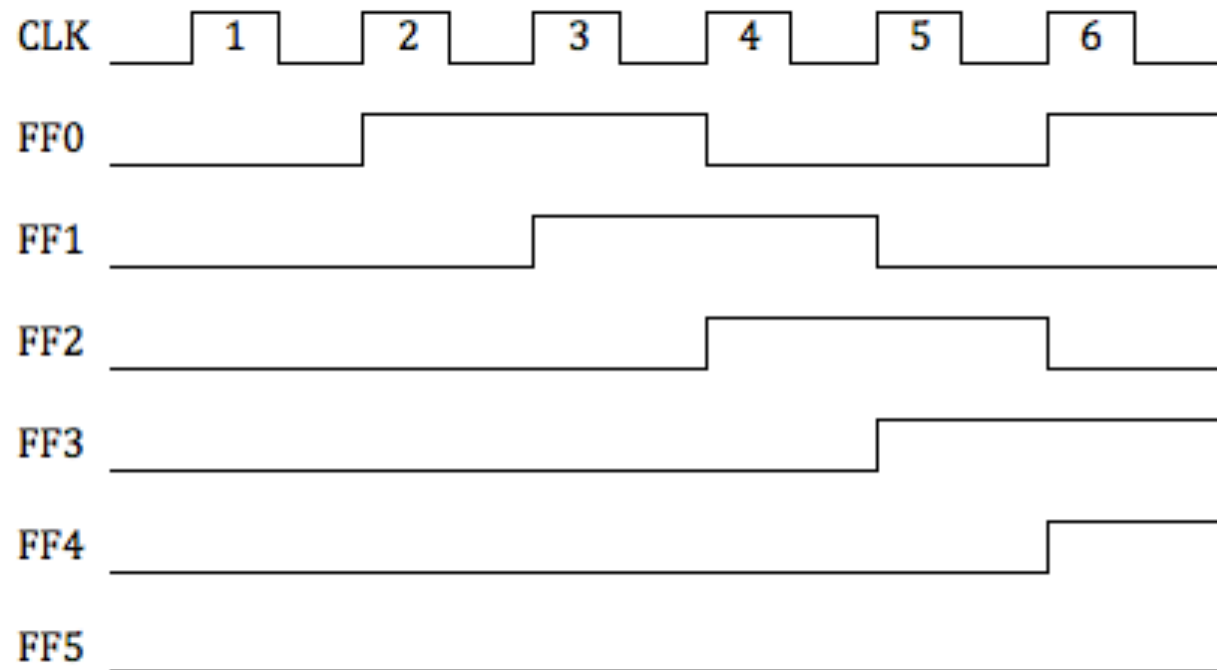
**Solution:**

a) 5 bits

b) The valid output

c) 11010

- Exercise 9.3:** a) Determine how many bit entered to the SIFO register?  
b) Circle on the timing diagram the valid output.  
c) Determine the data entered. (Assume **LSB** shifted in first)



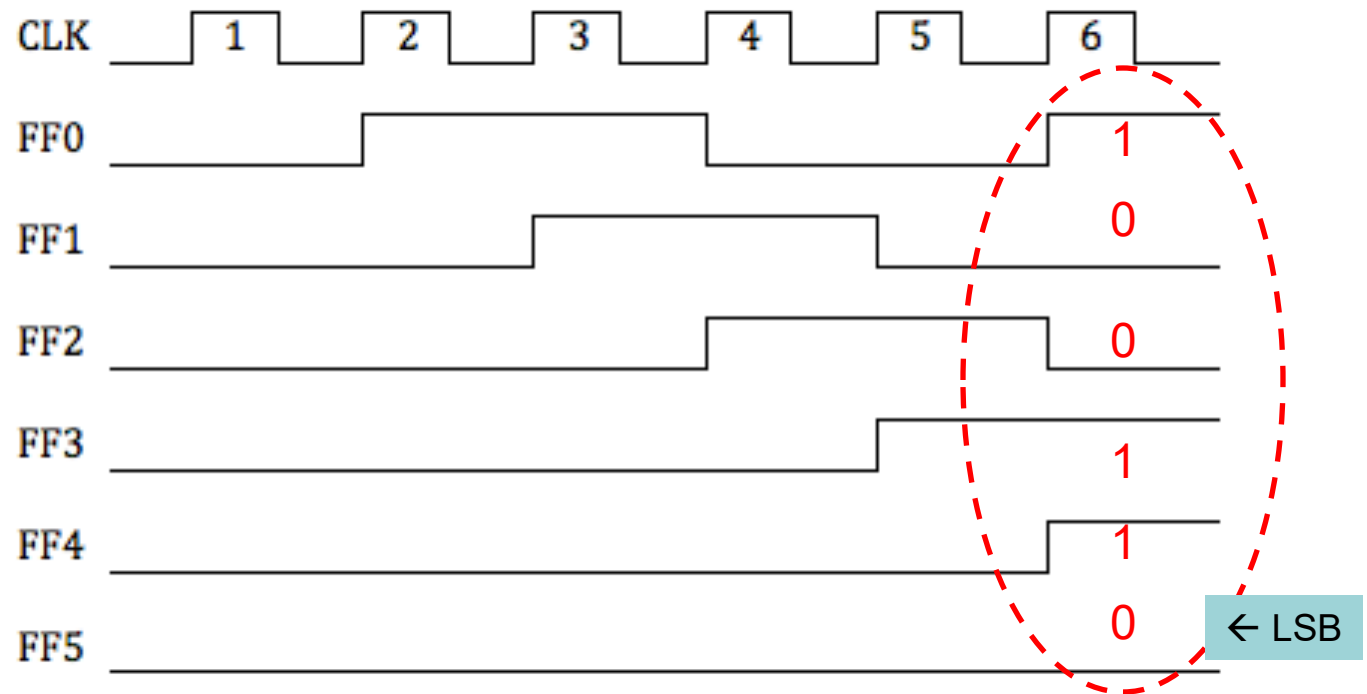
- Exercise 9.3:**
- a) Determine how many bit entered to the SIPO register?
  - b) Circle on the timing diagram the valid output.
  - c) Determine the data entered. (Assume **LSB** shifted in first)

**Solution:**

a) 6 bits

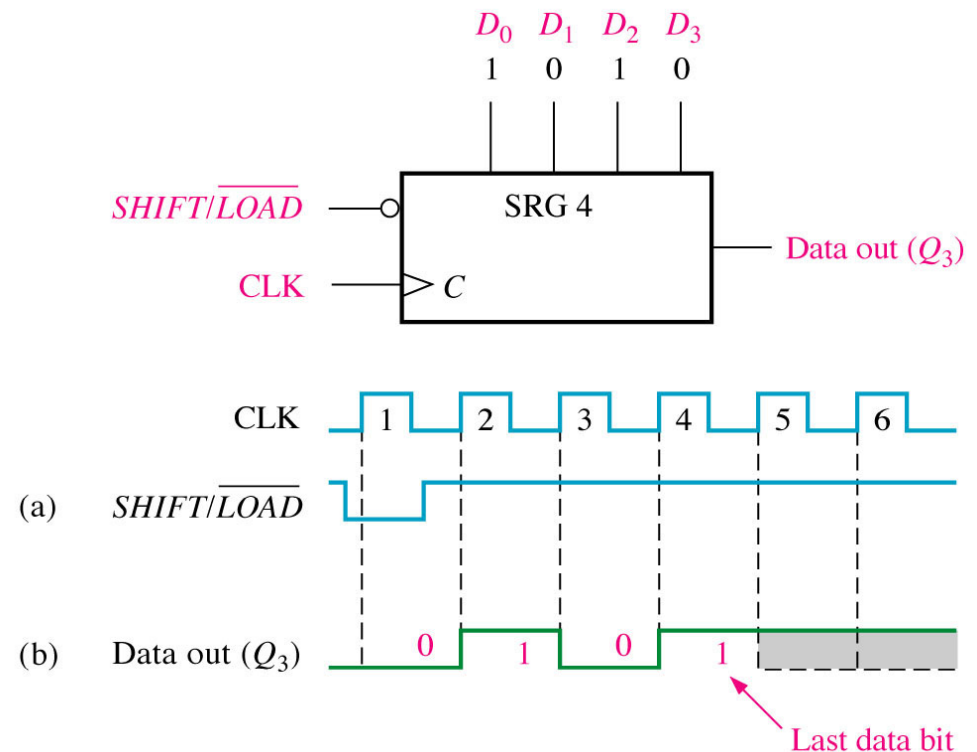
b) The valid output:

c) 100110



## (c) PISO

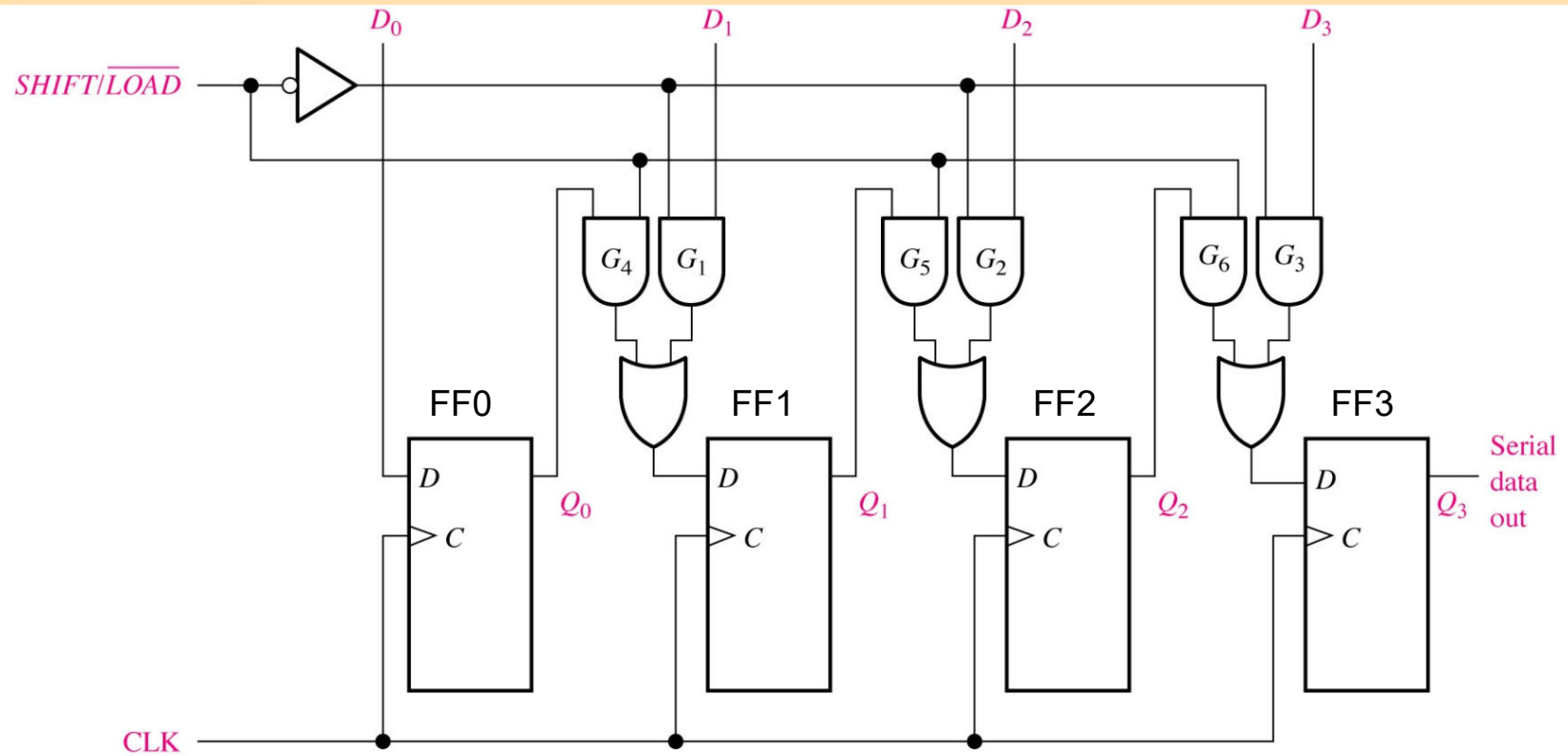
- Inputs entered **simultaneously** ( **in parallel** ) into respective stages on parallel lines
- Outputs are one bit at a time ( **serial** )
- Modes: *SHIFT /  $\overline{LOAD}$*



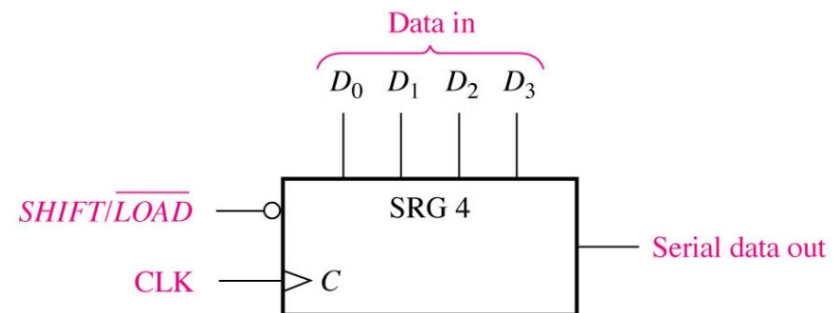
Mode	Value	Function
LOAD	0	Loads the input values
SHIFT	1	Shifts them out at clock pulse

### NOTES:

- Input pins:  $D_3D_2D_1D_0$
- MSB appear at Data out ( $Q_{n-1}$ )
- After  $n+1$  clock cycle, all  $n$  bits data will lost.



(a) Logic diagram

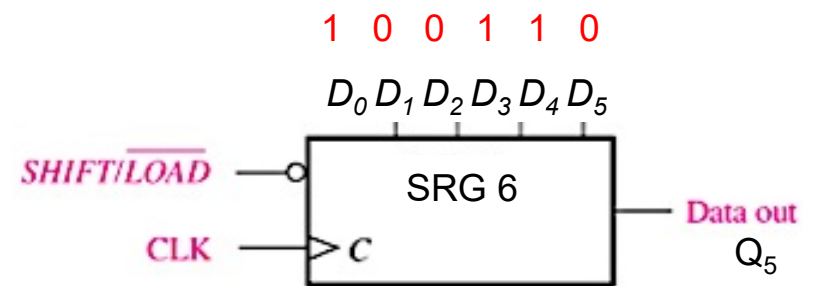


(b) Logic symbol

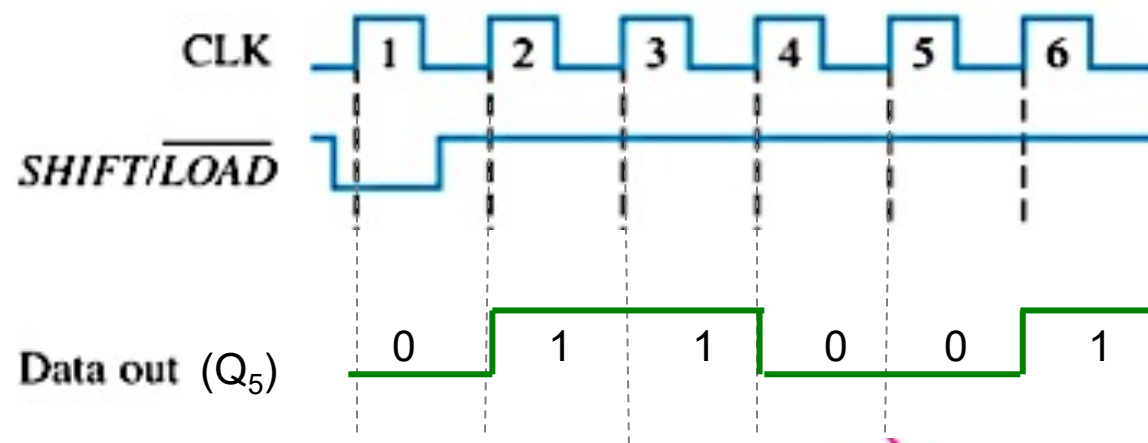


**Exercise 9.4:** Complete the timing diagram of 6-bit PISO shift register (SRG 6) for the data input **011001**. The register initially all is 0s and assume **MSB** shifted out **first**.

a) Draw the shift register logic symbol with the inputs.

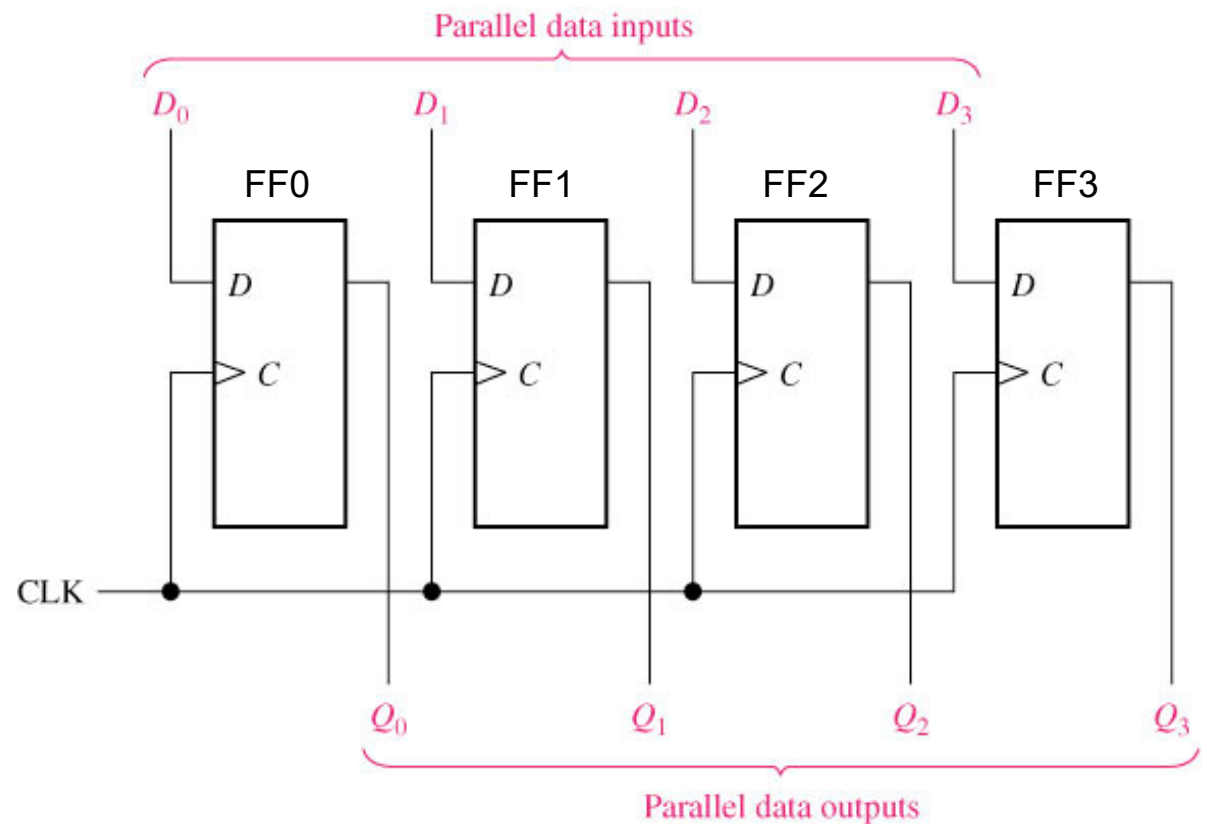


b) Complete the timing diagram.



## (d) PIPO

- Input and output done in parallel.
- Enter all inputs - Bits appear on the parallel outputs





**UTM**  
UNIVERSITI TEKNOLOGI MALAYSIA

[www.utm.my](http://www.utm.my)

---

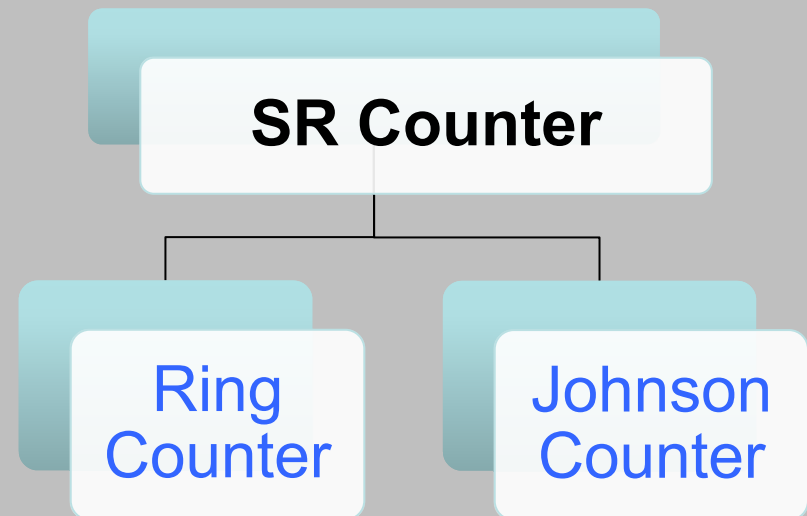
# Shift Register Counters

---

## Shift Register Counter

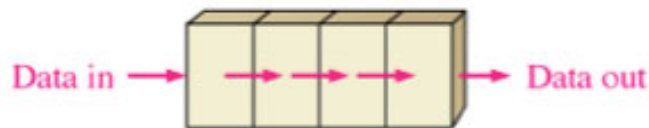
- A shift register with serial output connected back to the serial input to produce special sequences.
- In normal counter, to provide individual digit outputs instead of a binary or BCD output.
  - adding a **decoder**.
- But much simpler to use a different counter structure with simple decoder
  - **Shift Registers (SR)**

- 2 most common types:



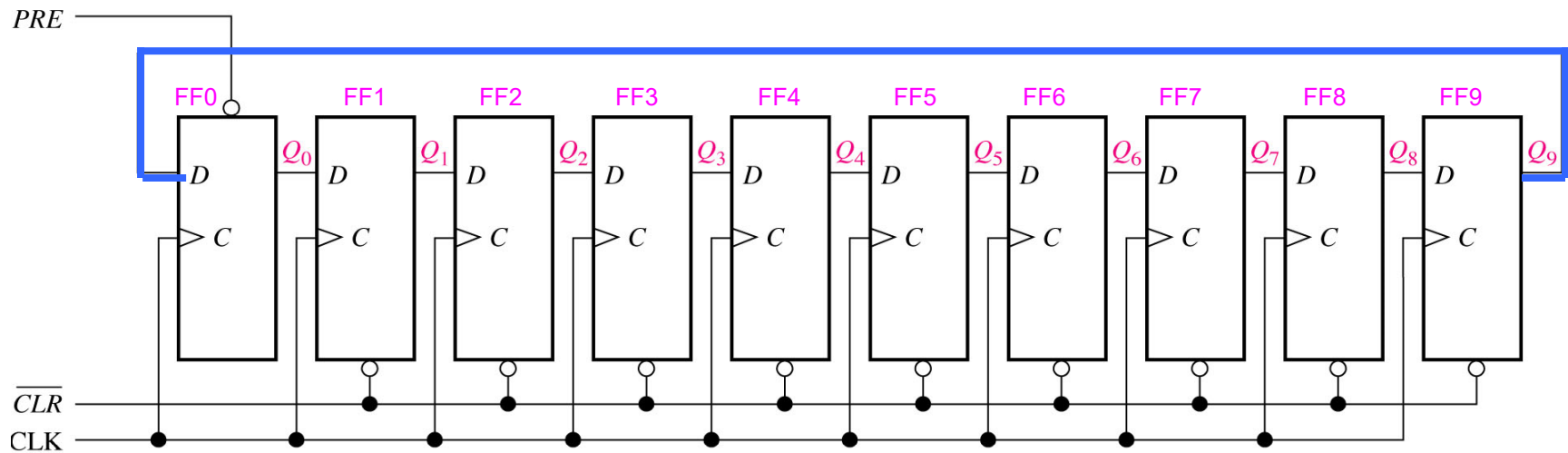
## Shift Register Counter: (a) Ring Counter

- A type of **SISO shift register** but final output is **fed back** to the first input.



(a) Serial in/shift right/serial out

- The ring counter uses 1 FF for each state in its sequence
- For a **10-bit ring counter**, there is a unique output for each decimal digit



$n\text{-bit} = \text{MOD } n$   
(state number)

### Example:

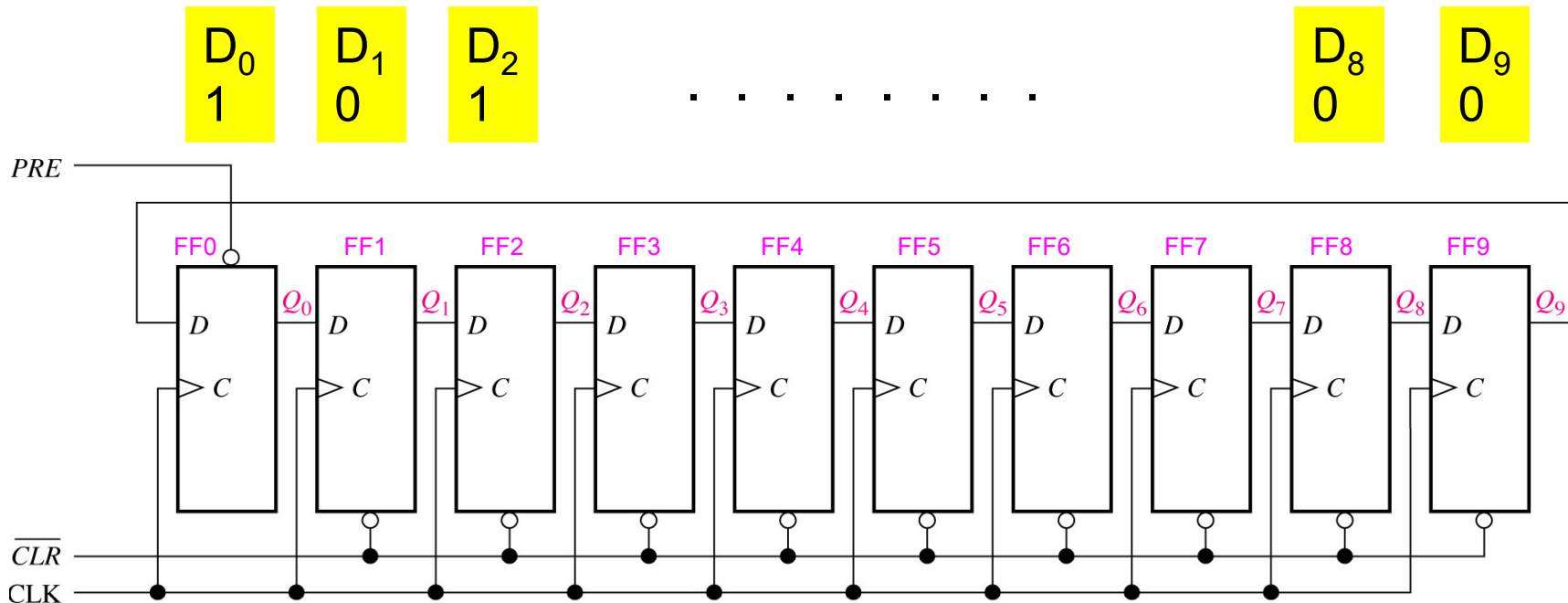
- For a 10-bit ring counter, there are 10 FFs which make a **MOD 10 counter**.
- It will **recycle** after 10 clock cycles.

**Example 7:** If a 10-bit ring counter has the initial state 0000000101, determine the waveform for each of the Q outputs.

**Solution:**

$D_9$	$D_8$	$D_7$	$D_6$	$D_5$	$D_4$	$D_3$	$D_2$	$D_1$	$D_0$
0	0	0	0	0	0	0	1	0	1

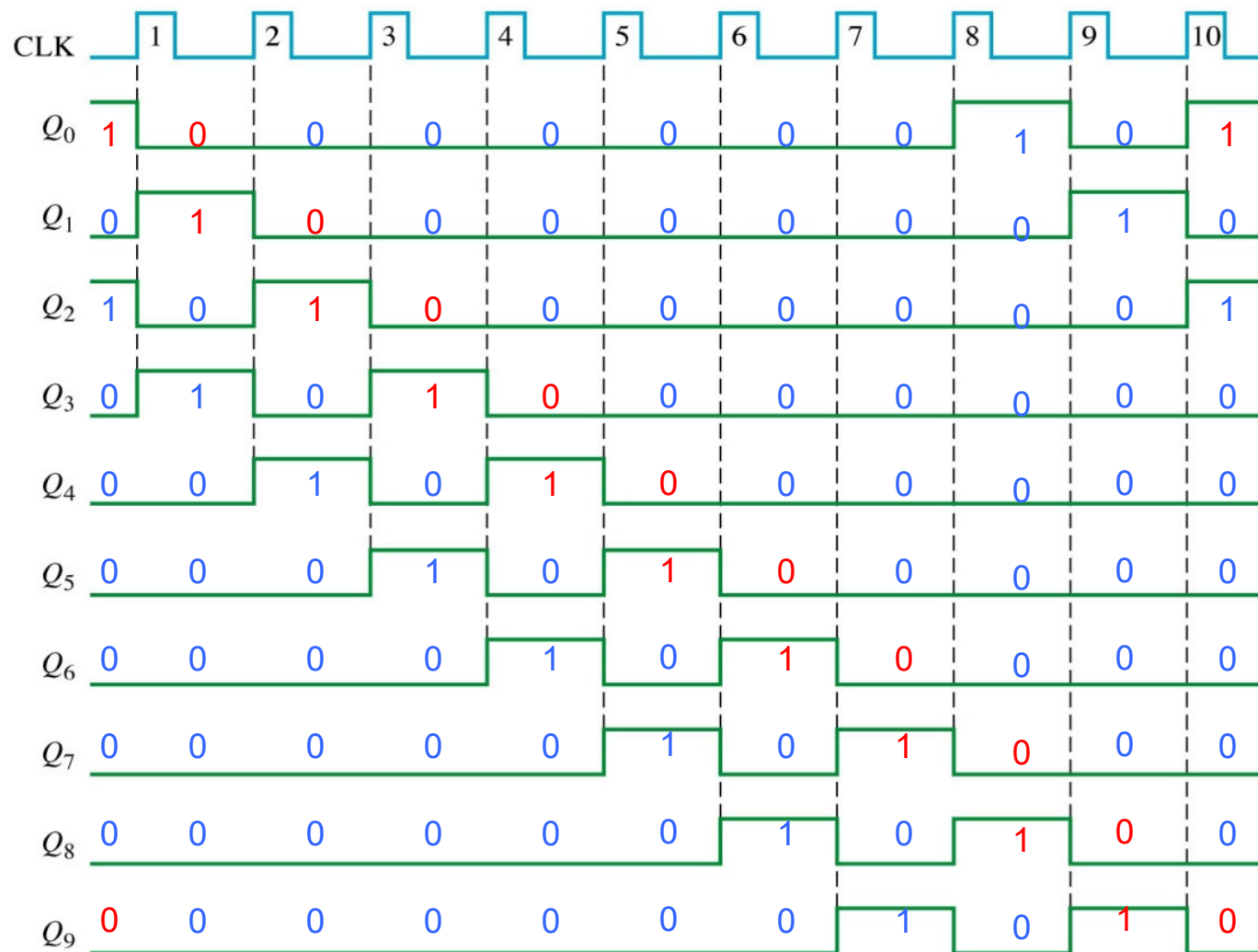
MSB LSB



D <sub>9</sub>	D <sub>8</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	0	0	0	0	0	0	1	0	1

MSB

LSB





Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	Q <sub>4</sub>	Q <sub>5</sub>	Q <sub>6</sub>	Q <sub>7</sub>	Q <sub>8</sub>	Q <sub>9</sub>
0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	0	0	1	0
2										
3										
4										
5										
6										
7										
8										
9										

Q<sub>9</sub>

Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	Q <sub>4</sub>	Q <sub>5</sub>	Q <sub>6</sub>	Q <sub>7</sub>	Q <sub>8</sub>	Q <sub>9</sub>
0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	0	1
3										
4										
5										
6										
7										
8										
9										

Q<sub>9</sub>

Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	Q <sub>4</sub>	Q <sub>5</sub>	Q <sub>6</sub>	Q <sub>7</sub>	Q <sub>8</sub>	Q <sub>9</sub>
0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	0	1
3	1	0	1	0	0	0	0	0	0	0
4										
5										
6										
7										
8										
9										

Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	Q <sub>4</sub>	Q <sub>5</sub>	Q <sub>6</sub>	Q <sub>7</sub>	Q <sub>8</sub>	Q <sub>9</sub>
0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	0	1
3	1	0	1	0	0	0	0	0	0	0
4	0	1	0	1	0	0	0	0	0	0
5										
6										
7										
8										
9										

Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	Q <sub>4</sub>	Q <sub>5</sub>	Q <sub>6</sub>	Q <sub>7</sub>	Q <sub>8</sub>	Q <sub>9</sub>
0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	0	1
3	1	0	1	0	0	0	0	0	0	0
4	0	1	0	1	0	0	0	0	0	0
5	0	0	1	0	1	0	0	0	0	0
6										
7										
8										
9										

Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	Q <sub>4</sub>	Q <sub>5</sub>	Q <sub>6</sub>	Q <sub>7</sub>	Q <sub>8</sub>	Q <sub>9</sub>
0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	0	1
3	1	0	1	0	0	0	0	0	0	0
4	0	1	0	1	0	0	0	0	0	0
5	0	0	1	0	1	0	0	0	0	0
6	0	0	0	1	0	1	0	0	0	0
7										
8										
9										

Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	Q <sub>4</sub>	Q <sub>5</sub>	Q <sub>6</sub>	Q <sub>7</sub>	Q <sub>8</sub>	Q <sub>9</sub>
0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	0	1
3	1	0	1	0	0	0	0	0	0	0
4	0	1	0	1	0	0	0	0	0	0
5	0	0	1	0	1	0	0	0	0	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	0	1	0	0	0
8										
9										

Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	Q <sub>4</sub>	Q <sub>5</sub>	Q <sub>6</sub>	Q <sub>7</sub>	Q <sub>8</sub>	Q <sub>9</sub>
0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	0	1
3	1	0	1	0	0	0	0	0	0	0
4	0	1	0	1	0	0	0	0	0	0
5	0	0	1	0	1	0	0	0	0	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	0	1	0	0	0
8	0	0	0	0	0	1	0	1	0	0
9										



Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	Q <sub>4</sub>	Q <sub>5</sub>	Q <sub>6</sub>	Q <sub>7</sub>	Q <sub>8</sub>	Q <sub>9</sub>
0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	0	1
3	1	0	1	0	0	0	0	0	0	0
4	0	1	0	1	0	0	0	0	0	0
5	0	0	1	0	1	0	0	0	0	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	0	1	0	0	0
8	0	0	0	0	0	1	0	1	0	0
9	0	0	0	0	0	0	1	0	1	0



Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	Q <sub>4</sub>	Q <sub>5</sub>	Q <sub>6</sub>	Q <sub>7</sub>	Q <sub>8</sub>	Q <sub>9</sub>
0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	0	1
3	1	0	1	0	0	0	0	0	0	0
4	0	1	0	1	0	0	0	0	0	0
5	0	0	1	0	1	0	0	0	0	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	0	1	0	0	0
8	0	0	0	0	0	1	0	1	0	0
9	0	0	0	0	0	0	1	0	1	0

## Shift Register Counter: (b) Johnson Counter

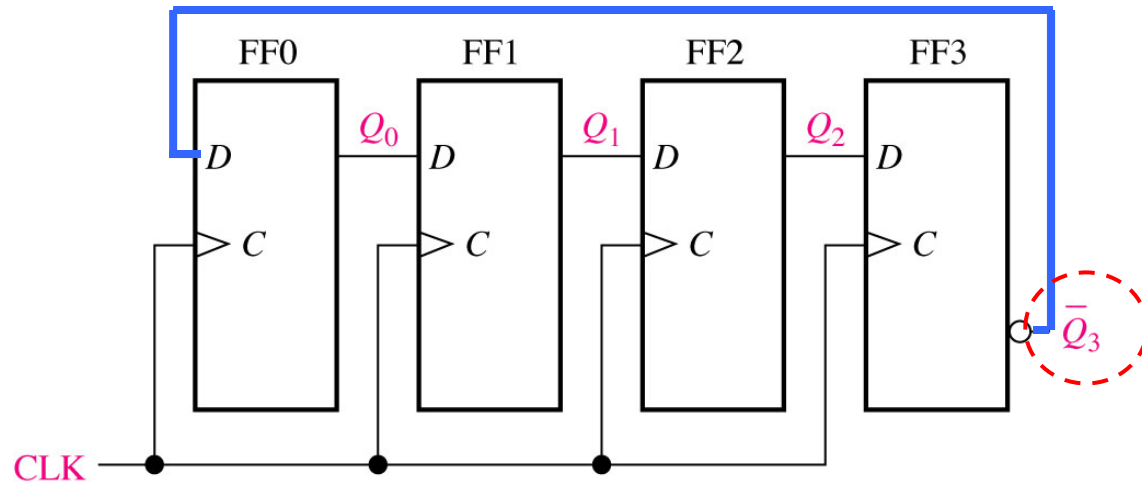
- In the Johnson counter the last output is complemented and fed back in as an input to the first FF
- Examples shown with **D** FF, but can be implemented with other types of FF as well.
- Number of unique states are **2 times** the number of bits (FF)
  - 4 bits  $\rightarrow 4*2 = 8$  states
  - 5 bits  $\rightarrow 5*2 = 10$  states
- Johnson counter will produce a modulus of  $2n$ ;  
( $n$  = number of stages)
- \*\*modulus 10 a.k.a. mod 10

$$n\text{-bit} = \text{MOD } 2n \\ (\text{state number})$$

$n\text{-bit} = \text{MOD } 2n$   
(state number)

**Example (a):**

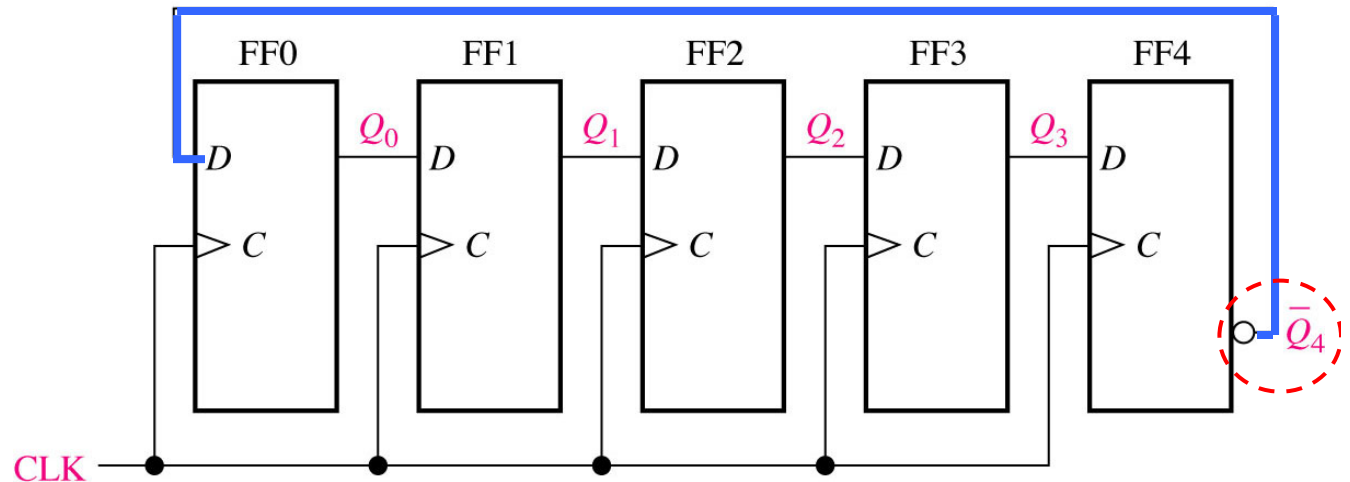
- For a **4-bit** ring counter, there are 4 FFs which make a **MOD 8 counter**.
- It will recycle after 8 clock cycles.



(a) Four-bit Johnson counter

**Example (b):**

- For a **5-bit** ring counter, there are 5 FFs which make a **MOD 10 counter**.
- It will recycle after 10 clock cycles.



(b) Five-bit Johnson counter

Example:

Clock Pulse	$Q_0$	$Q_1$	$Q_2$	$Q_3$
0	0	0	0	0
1				
2				
3				
4				
5				
6				
7				
8				
9				

$\overline{Q_3}$

Example:

Clock Pulse	$Q_0$	$Q_1$	$Q_2$	$Q_3$
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5				
6				
7				
8				
9				

Example:

Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8				
9				

Complete  
one cycle

Repete  
cycle

Example:

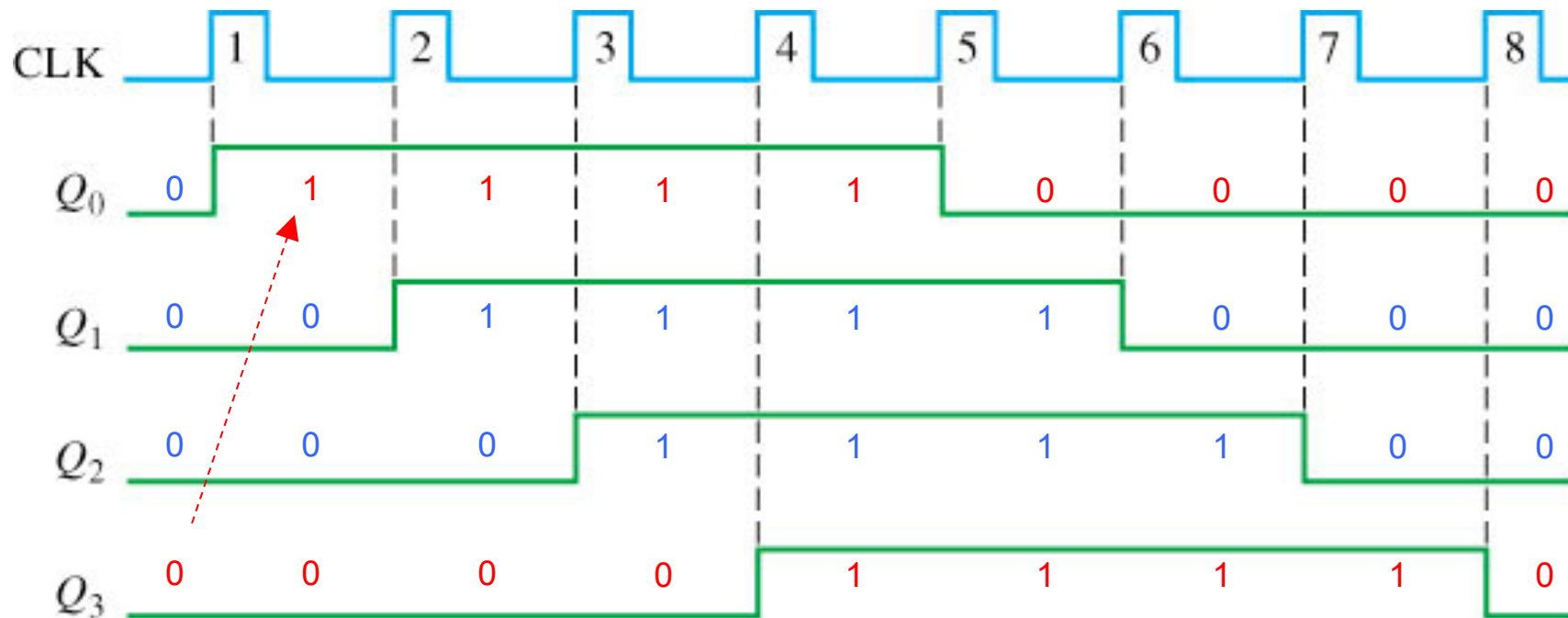
Clock Pulse	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0
9	.	.	.	.

Complete  
one cycle

Repeat  
cycle

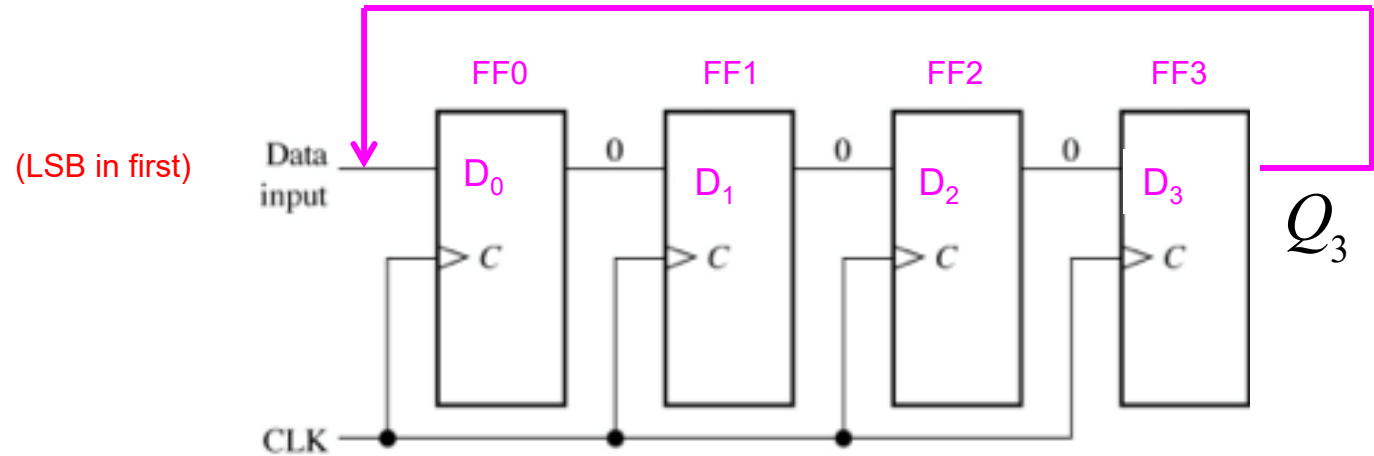


Clock Pulse	$Q_0$	$Q_1$	$Q_2$	$Q_3$
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

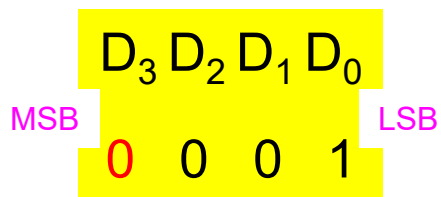


# Summary: Ring Counter

$n$ -bit = MOD  $n$   
(state number)



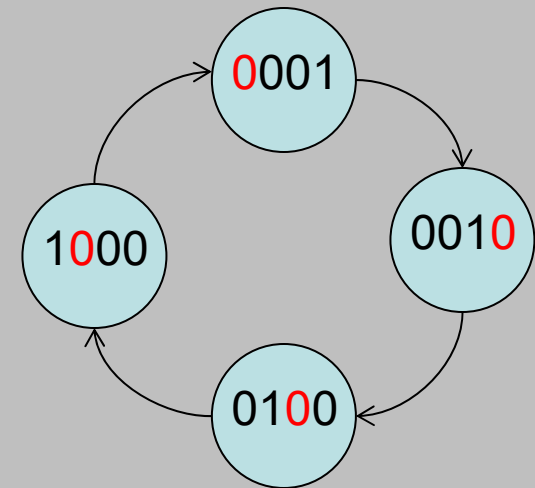
Data representation:



Truth Table:

Clock	FF0	FF1	FF2	FF3
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

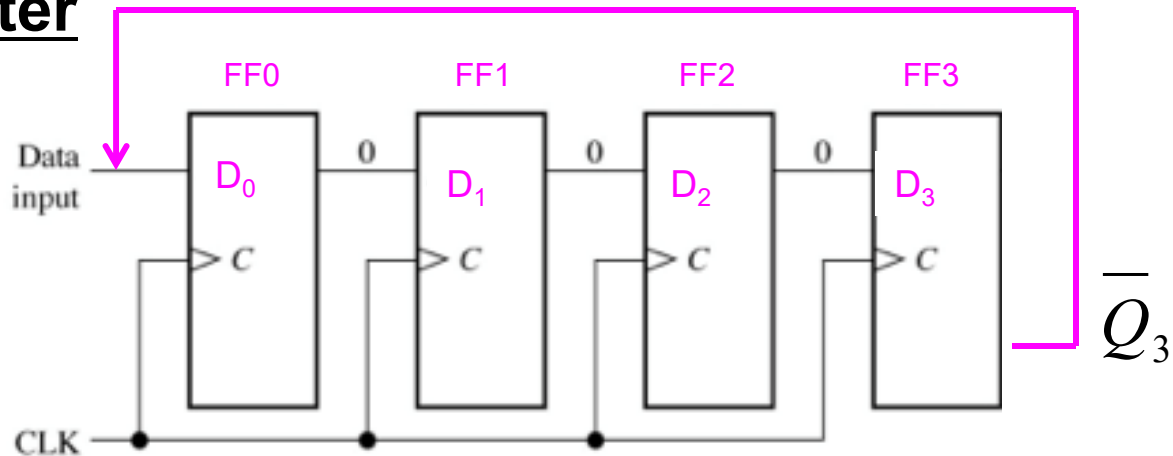
State Diagram:



# Summary: Johnson Counter

(LSB in first)

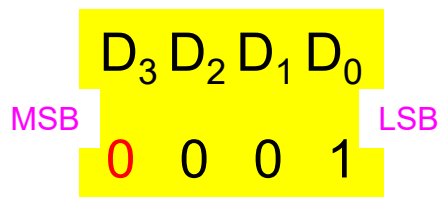
$n\text{-bit} = \text{MOD } 2n$   
(state number)



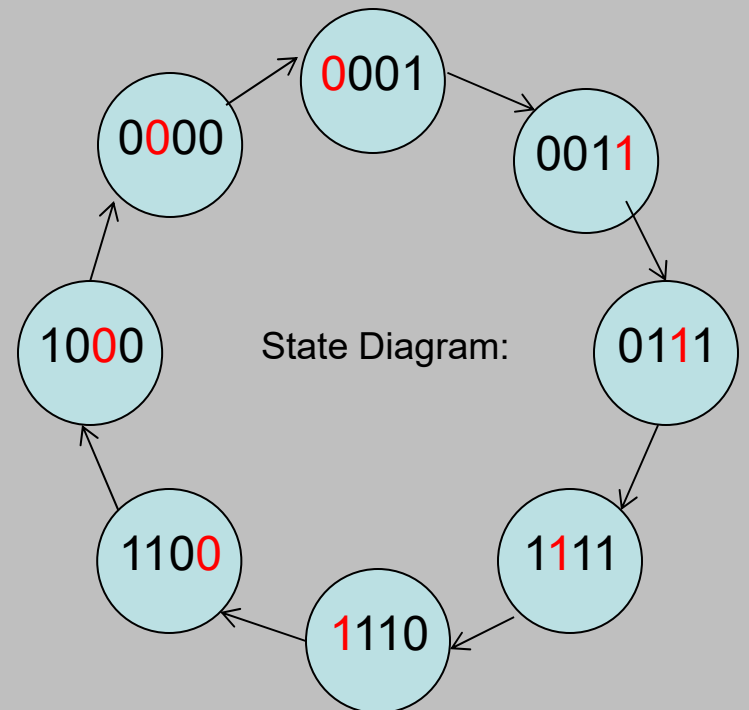
Truth Table:

Clock	FF0	FF1	FF2	FF3
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

Data representation:



State Diagram:



## Summary: Johnson Counter

Clock	FF0	FF1	FF2	FF3
	A	B	C	D
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

Similar

(Compare to other state)

### Other characteristic of Johnson counter:

- Always exist 2 unique bits compared to other state.

## Summary: Johnson Counter

Clock	FF0	FF1	FF2	FF3
	A	B	C	D
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

(Compare to other state)

### Other characteristic of Johnson counter:

- Always exist 2 unique bits compared to other state.

Similar

## Summary: Johnson Counter

Clock	FF0	FF1	FF2	FF3
	A	B	C	D
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

(No other states similar; therefore, bit **A** and **B** (i.e. **1** and **0**) are unique)

(Compare to other state)

### Other characteristic of Johnson counter:

- Always exist 2 unique bits compared to other state.

## Summary: Johnson Counter

Clock	FF0	FF1	FF2	FF3
	A	B	C	D
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

### Other characteristic of Johnson counter:

- Always exist 2 unique bits compared to other state.

### Self-Test:

What are the unique bits for the rest of the clocks?

## Summary: Johnson Counter

Clock	FF0	FF1	FF2	FF3
	A	B	C	D
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

**AND** gate is required for output

$$\overline{A}B$$

$$B\overline{C}$$

$$C\overline{D}$$

$$AD$$

$$\overline{A}B$$

$$B\overline{C}$$

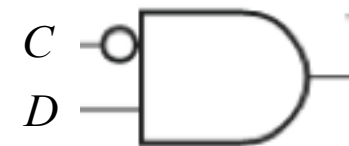
$$C\overline{D}$$

$$\overline{A}D$$

### Other characteristic of Johnson counter:

- Always exist 2 unique bits compared to other state.
- It will only requires 2 input **AND** decoder to decode a state to an active **HIGH** device.

### Example:





# Comparison of a counter

- For a given  $n$  flip-flop, the binary counter can produce the most state and ring is the worst
- In term of decoding , Ring counter is the best because there is no need for a decoder to decode each state, binary counter is the worst because require a more complex decoder
- Johnson always in the middle when comparing those feature

	MOD for $n$ flip-flop	Decoder Input
Ring	$n$	no need for a decoder
Johnson	$2n$	2 input decoder
Binary	$2^n$	usually $> 2$ input decoder

(No. of state)

### Self-Test:

1. Given a 3-bit Johnson counter with initial value 000, draw the appropriate sequence table for 7 clock pulses.

### Solution:

Clock Pulse	$Q_0$	$Q_1$	$Q_2$
Initial	0	0	0
1	1	0	0
2	1	1	0
3	1	1	1
4	0	1	1
5	0	0	1
6	0	0	0

2. A modulus-10 Ring counter requires a minimum of A.

A) 10 FF

C) 5 FF

B) 20 FF

D) 12 FF

3. A modulus-10 Johnson counter requires B.

A) 10 FF

C) 20 FF

B) 5 FF

D) 12 FF

4. The group of 8 bits  $1011\ 0110_2$  is serially shifted (right most bit first, MSB) into an 8-bit parallel output shift register with an initial state of  $1110\ 1100_2$ . After 2 clock pulses, the register contains C. Assume the first FF is the LSB.

A)  $1011\ 0001_2$

C)  $1011\ 0010_2$

B)  $0100\ 1101_2$

D)  $1101\ 1110_2$

5. To serially shift a byte of data into a shift register, there must be B.
- A) 1 clock pulse
  - B) 8 clock pulses
  - C) 8 load pulses
  - D) one clock pulse for each 1 in the data
6. To parallel load a byte of data into a shift register with a synchronous load, there must be A.
- A) 1 clock pulse
  - B) 8 clock pulses
  - C) 8 load pulses
  - D) one clock pulse for each 0 in the data



**UTM**  
UNIVERSITI TEKNOLOGI MALAYSIA

[www.utm.my](http://www.utm.my)

# END OF DIGITAL LOGIC

## MODULES



**FINAL**

## **SCSR1013 Digital Logic**

*(Module 6, 7, 8, 9)*

**9 February 2023 (Thursday) || 2:30 pm – 5:30 pm**

**Dewan Sultan Iskandar (DSI)**