Unit 7
Registers and Counters

Er. Sachita Nand Mishra
M.E. in Computer and Electronics Engineering

Introduction

- Circuits that include flip-flops are usually classified by the function they perform. Two such circuits are registers and counters.
- Registers

❖ A group of binary cells (FFs) suitable for holding binary data information

❖ In addition to the FFs, a register may have combinational gates to control when and how the new information is transferred into the register

- Counters

❖ A register that goes through a predetermined sequence of states upon the application of input pulses

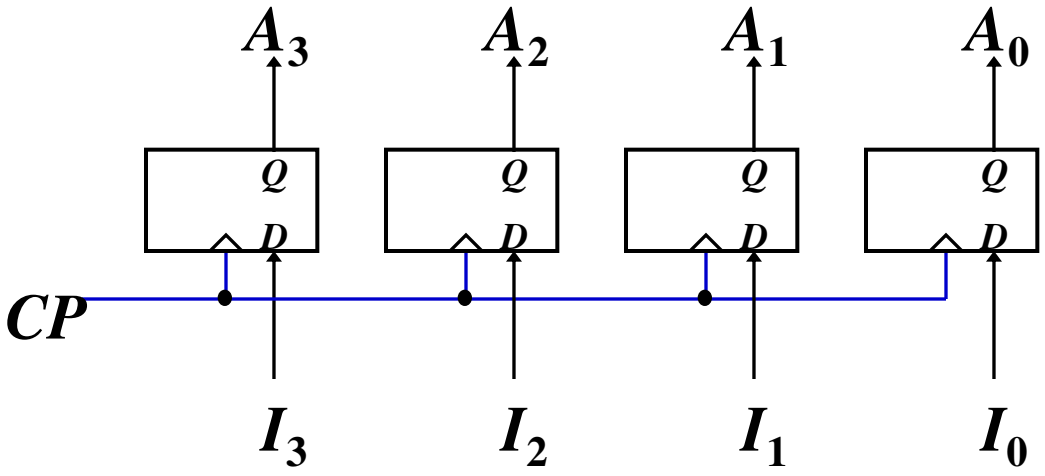
❖ The gates in a counter are connected in such a way as to produce a predefined sequence of binary states in the register.

Introduction to Registers

- An *n*-bit register has a group of *n* flip-flops and some logic gates and is capable of storing *n* bits of information.
- The flip-flops store the information while the gates control when and how new information is transferred into the register.
- Some functions of register:
  - ❖ retrieve data from register
  - ❖ store/load new data into register (serial or parallel)
  - ❖ shift the data within register (left or right)

Introduction to Registers

- No external gates.
- Example: A 4-bit register. A new 4-bit data is loaded every clock cycle.



Registers With Parallel Load

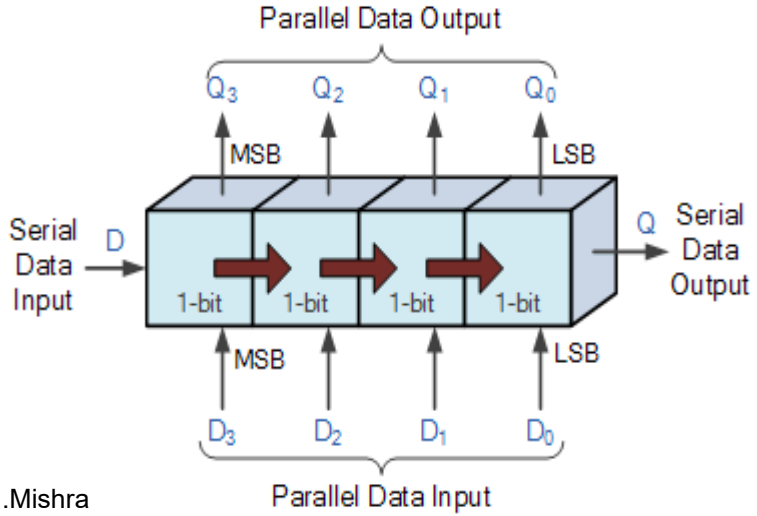
- Instead of loading the register at every clock pulse, we may want to control when to load.
- *Loading* a register: transfer new information into the register. Requires a *load* control input.
- *Parallel loading*: all bits are loaded simultaneously.

Shift Registers

- Another function of a register, besides storage, is to provide for *data movements*.
- Each *stage* (flip-flop) in a shift register represents one bit of storage, and the shifting capability of a register permits the movement of data from stage to stage within the register, or into or out of the register upon application of clock pulses.
- A register capable of shifting its binary information either to the right or to the left is called a shift register.
- The logical configuration of a shift register consists of a chain of flip-flops connected in cascade, with the output of one flip-flop connected to the input of the next flip-flop.
- All flip-flops receive a common clock pulse that causes the shift from one stage to the next.

Shift Registers

- Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:
- Serial-in to Parallel-out (SIPO) - the register is loaded with serial data, one bit at a time, with the stored data being available in parallel form.
- Serial-in to Serial-out (SISO) - the data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.
- Parallel-in to Serial-out (PISO) - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
- Parallel-in to parallel-out (PIPO)
  - the parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.



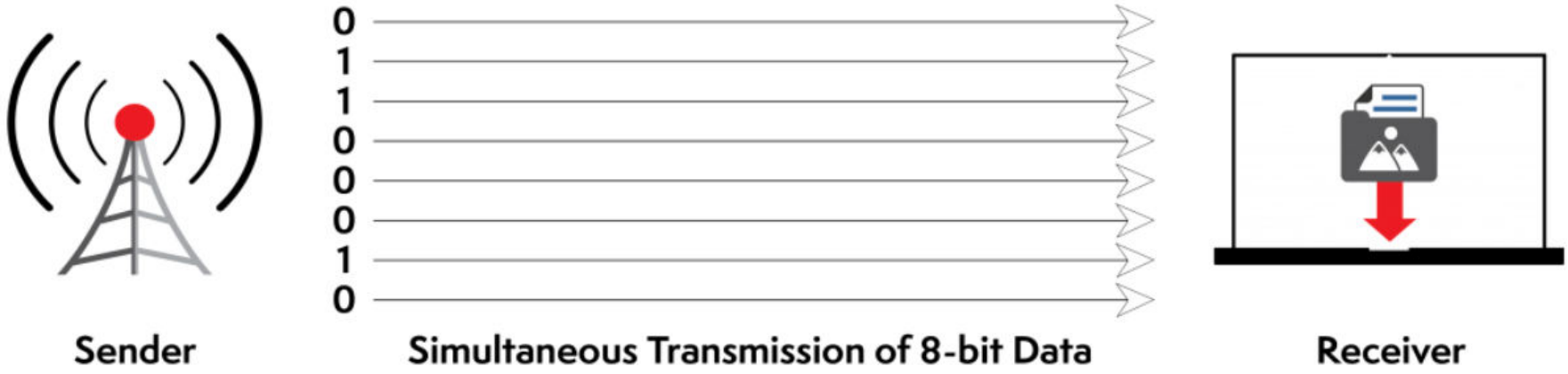
Serial Transmission

- When data is sent or received using serial data transmission, the data bits are organized in a specific order, since they can only be sent one after another.
- The order of the data bits is important as it dictates how the transmission is organized when it is received.
- It is viewed as a reliable data transmission method because a data bit is only sent if the previous data bit has already been received.

Example of Serial Data Transmission

Parallel Transmission

- When data is sent using parallel data transmission, multiple data bits are transmitted over multiple channels at the same time.
- This means that data can be sent much faster than using serial transmission methods.
- Parallel transmission is used for short distance.



Example of Parallel Data Transmission

Difference between Serial and Parallel Transmission

S.NO	SERIAL TRANSMISSION	PARALLEL TRANSMISSION
1	In serial transmission, data(bit) flows in bi-direction.	In Parallel Transmission, data flows in multiple lines.
2	Serial Transmission is cost efficient.	Parallel Transmission is not cost efficient.
3	In serial transmission, one bit transferred at one clock pulse.	In Parallel Transmission, eight bits transferred at one clock pulse.
4	Serial Transmission is slow in comparison of Parallel Transmission.	Parallel Transmission is fast in comparison of Serial Transmission.
5	Generally, Serial Transmission is used for long distance.	Generally, Parallel Transmission is used for short distance.
6	The circuit used in Serial Transmission is simple.	The circuit used in Parallel Transmission is relatively complex.

Serial-in to Parallel-out (SIPO) Shift Register

- Accepts data serially.
- Outputs of all stages are available simultaneously.

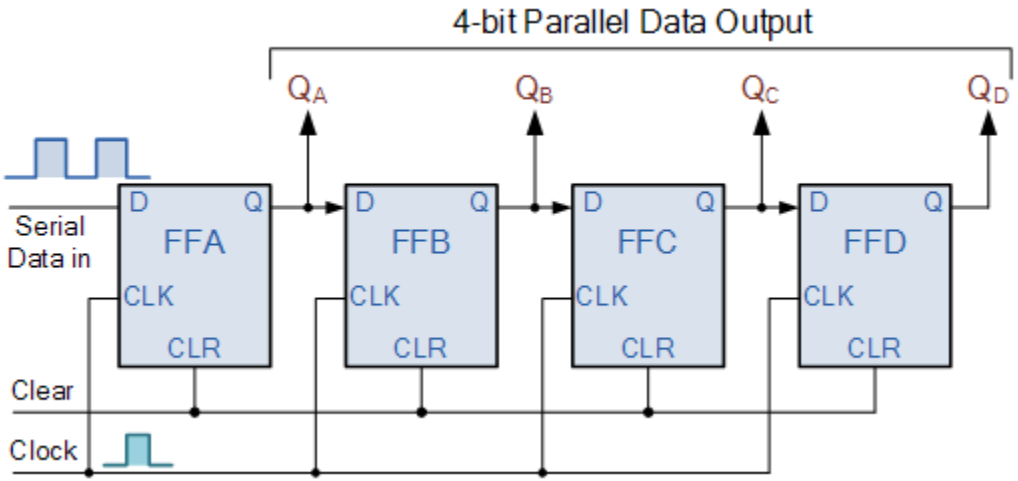
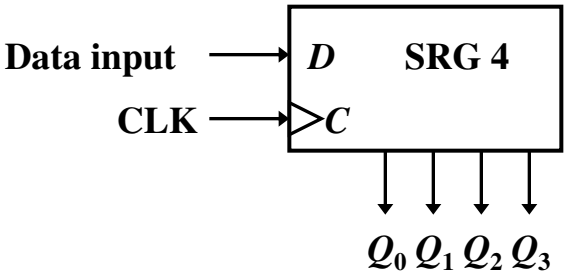


Fig. 4 bit serial in parallel out shift register



Logic symbol

Serial-in to Parallel-out (SIPO) Shift Register

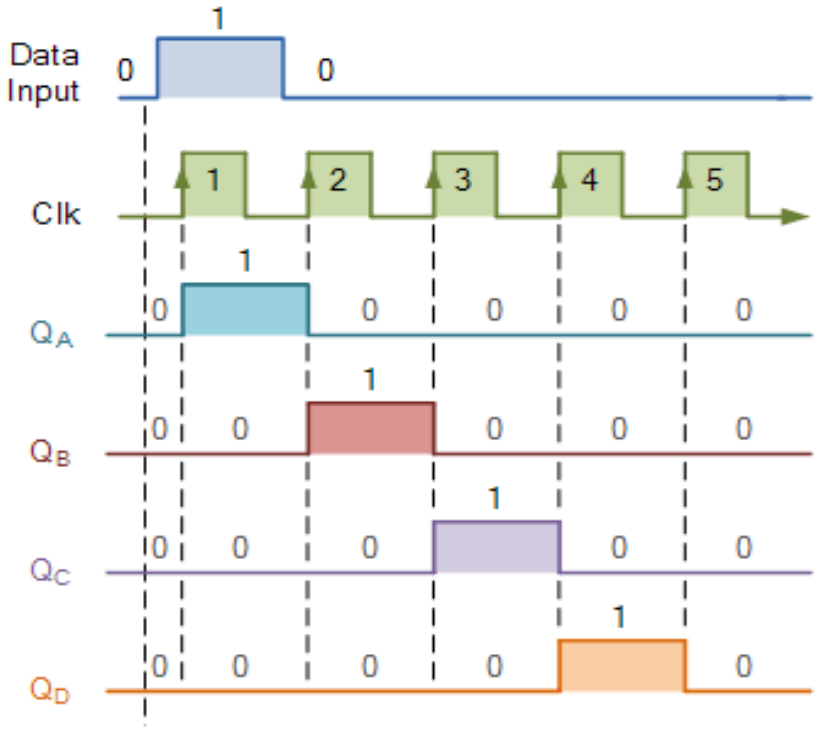
- Lets assume that all the flip-flops (FFA to FFD) have just been RESET (CLEAR input) and that all the outputs Q_A to Q_D are at logic level "0" ie, no parallel data output.
- If a logic "1" is connected to the DATA input pin of FFA then on the first clock pulse the output of FFA and therefore the resulting Q_A will be set HIGH to logic "1" with all the other outputs still remaining LOW at logic "0".
- Assume now that the DATA input pin of FFA has returned LOW again to logic "0" giving us one data pulse or 0-1-0.
- The second clock pulse will change the output of FFA to logic "0" and the output of FFB and Q_B HIGH to logic "1" as its input D has the logic "1" level on it from Q_A.
- When the third clock pulse arrives this logic "1" value moves to the output of FFC (Q_C) and so on until the arrival of the fifth clock pulse which sets all the outputs Q_A to Q_D back again to logic level "0" because the input to FFA has remained constant at logic level "0".

Serial-in to Parallel-out (SIPO) Shift Register

- The effect of each clock pulse is to shift the data contents of each stage one place to the right, and this is shown in the following table until the complete data value of 0-0-0-1 is

Clock Pulse No	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0

Shifting data 10000 serially



Timing diagram

Serial In/Serial Out Shift Registers

- Accepts data serially – one bit at a time – and also produces output serially.

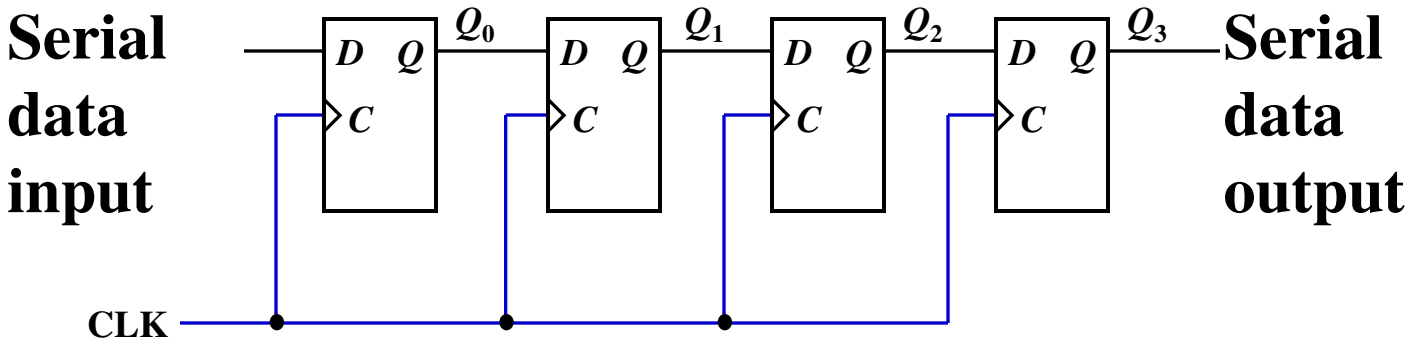
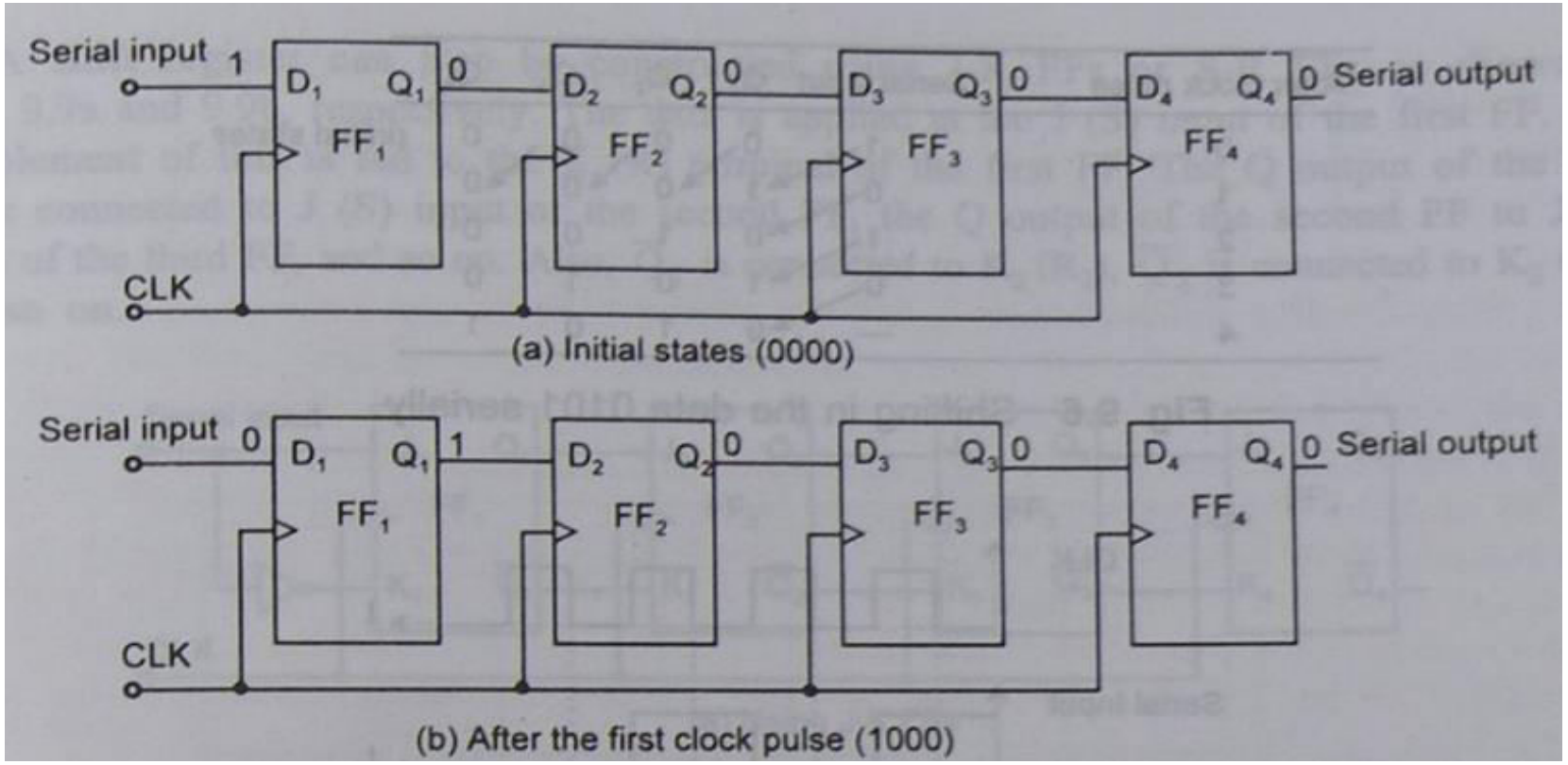


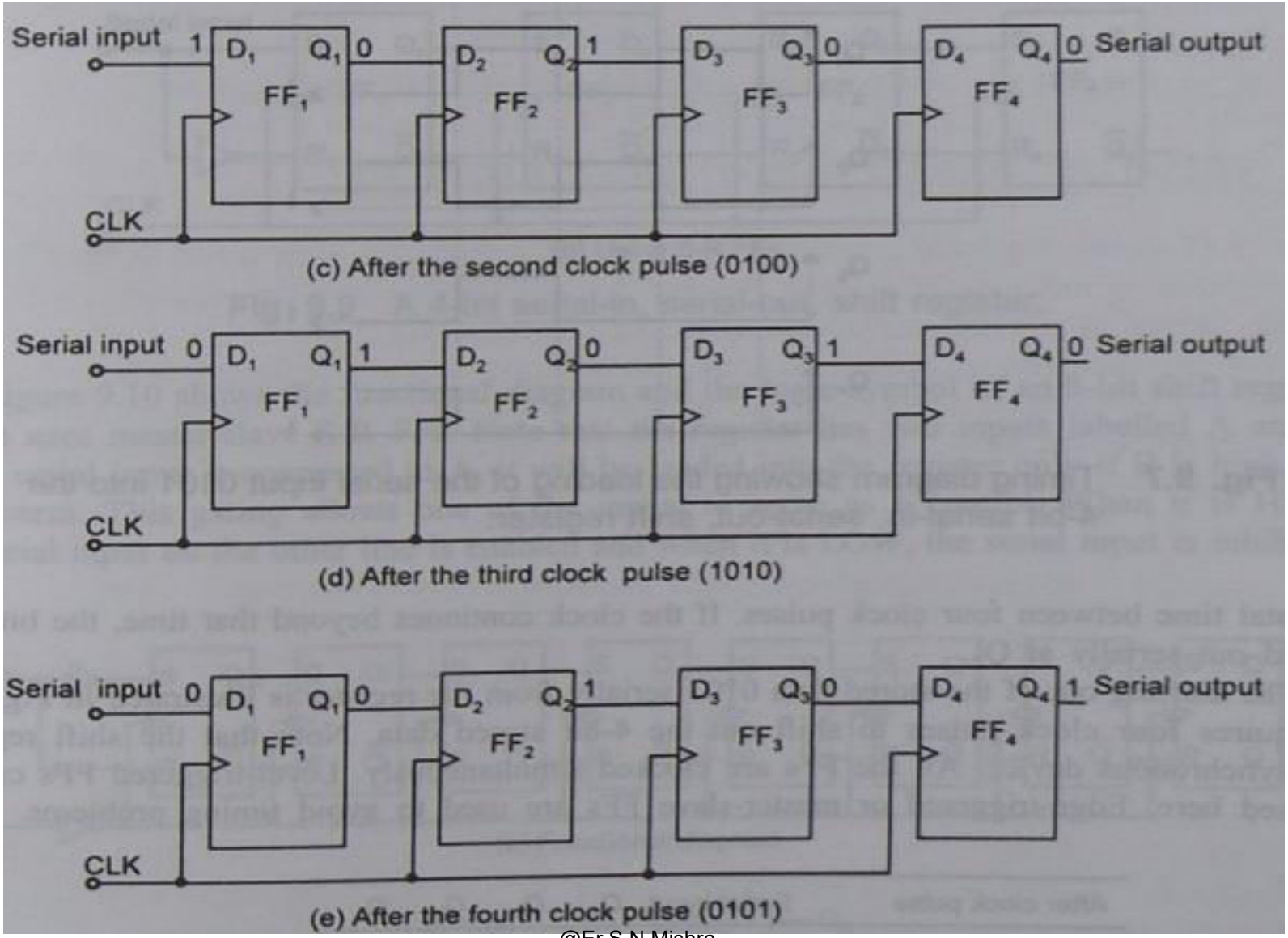
Fig serial in serial out shift register logic diagram

Serial In/Serial Out Shift Registers

- This **shift register** is very similar to the SIPO except the data was read directly in a parallel form from the outputs Q_A to Q_D in SIPO, but the data is allowed to flow straight through the register and data out in the other end.
- Since there is only one output, the DATA leaves the shift register one bit at a time in a serial pattern, hence the name **Serial-in to Serial-Out Shift Register** or **SISO**.



Serial In/Serial Out Shift Registers

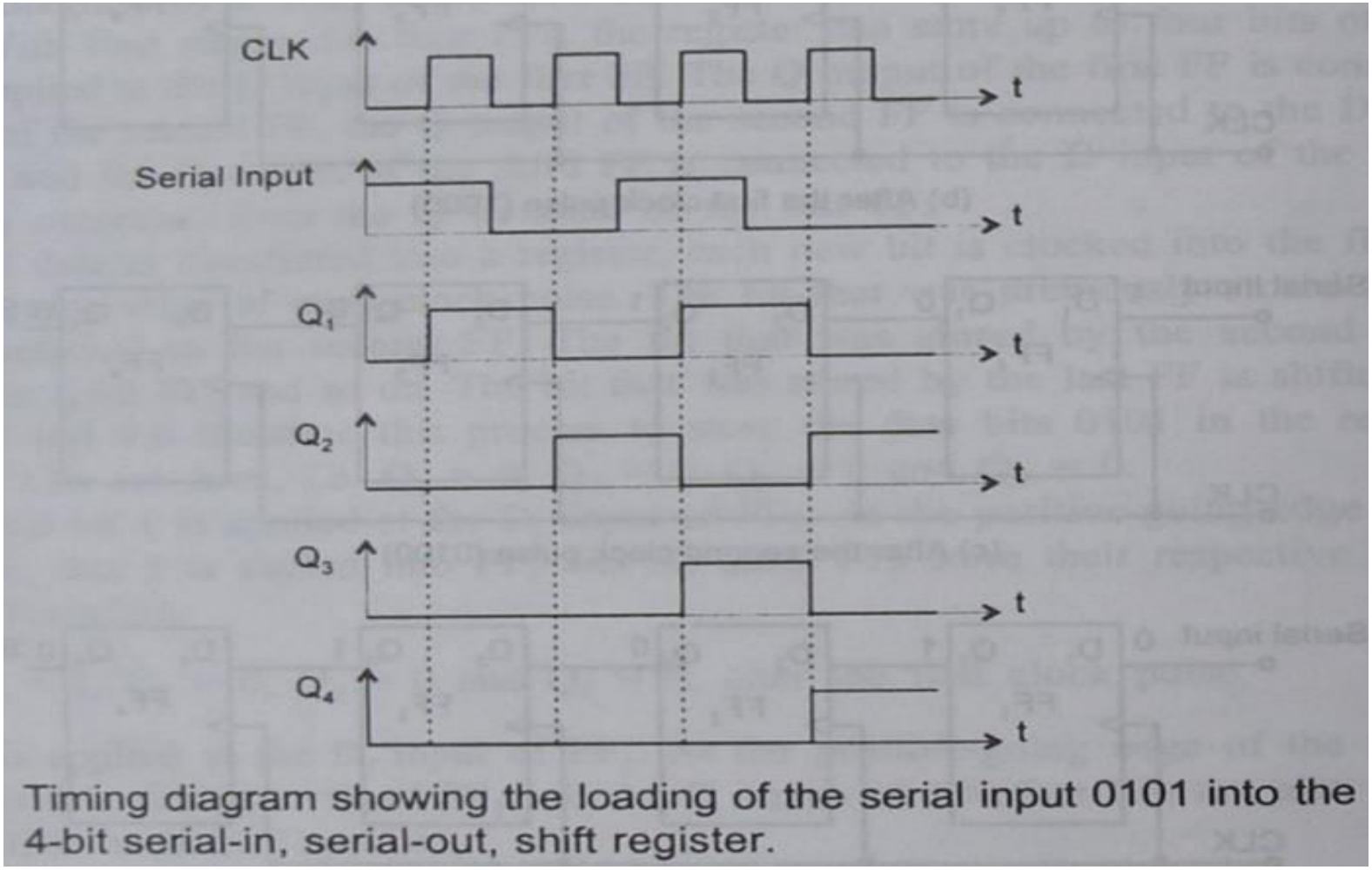


Serial In/Serial Out Shift Registers

After clock pulse	Serial input	Q_1	Q_2	Q_3	Q_4	
0	1	0	0	0	0	(initial states
1	0	1	0	0	0	
2	1	0	1	0	0	
3	0	1	0	1	0	
4	—	0	1	0	1	

Fig. [illegible] Shifting in the data 0101 serially.

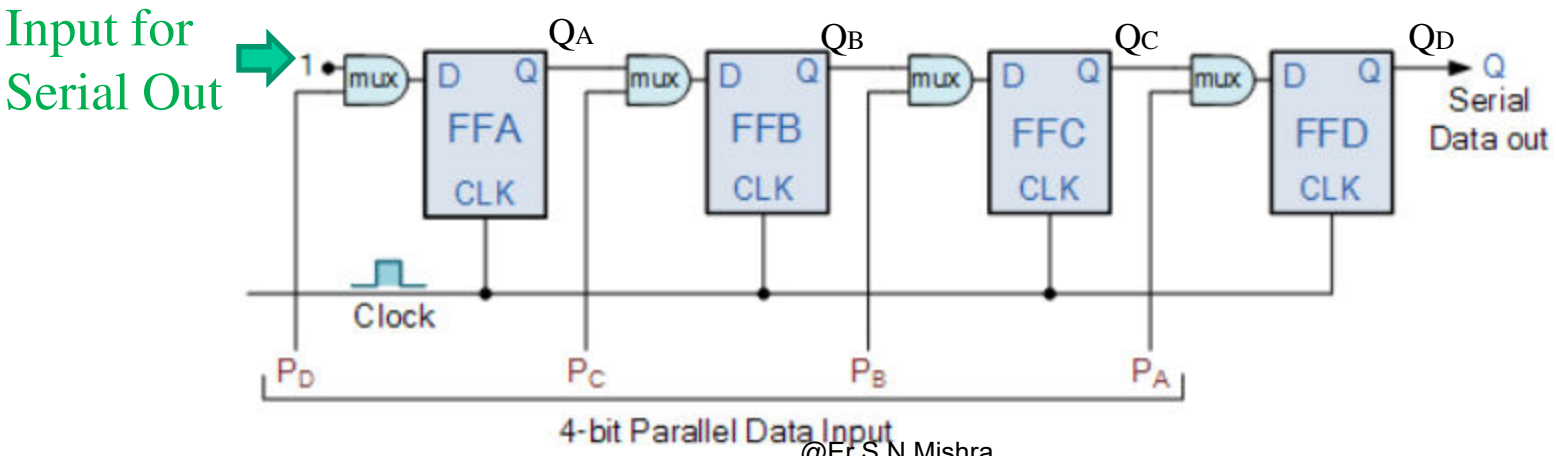
Serial In/Serial Out Shift Registers



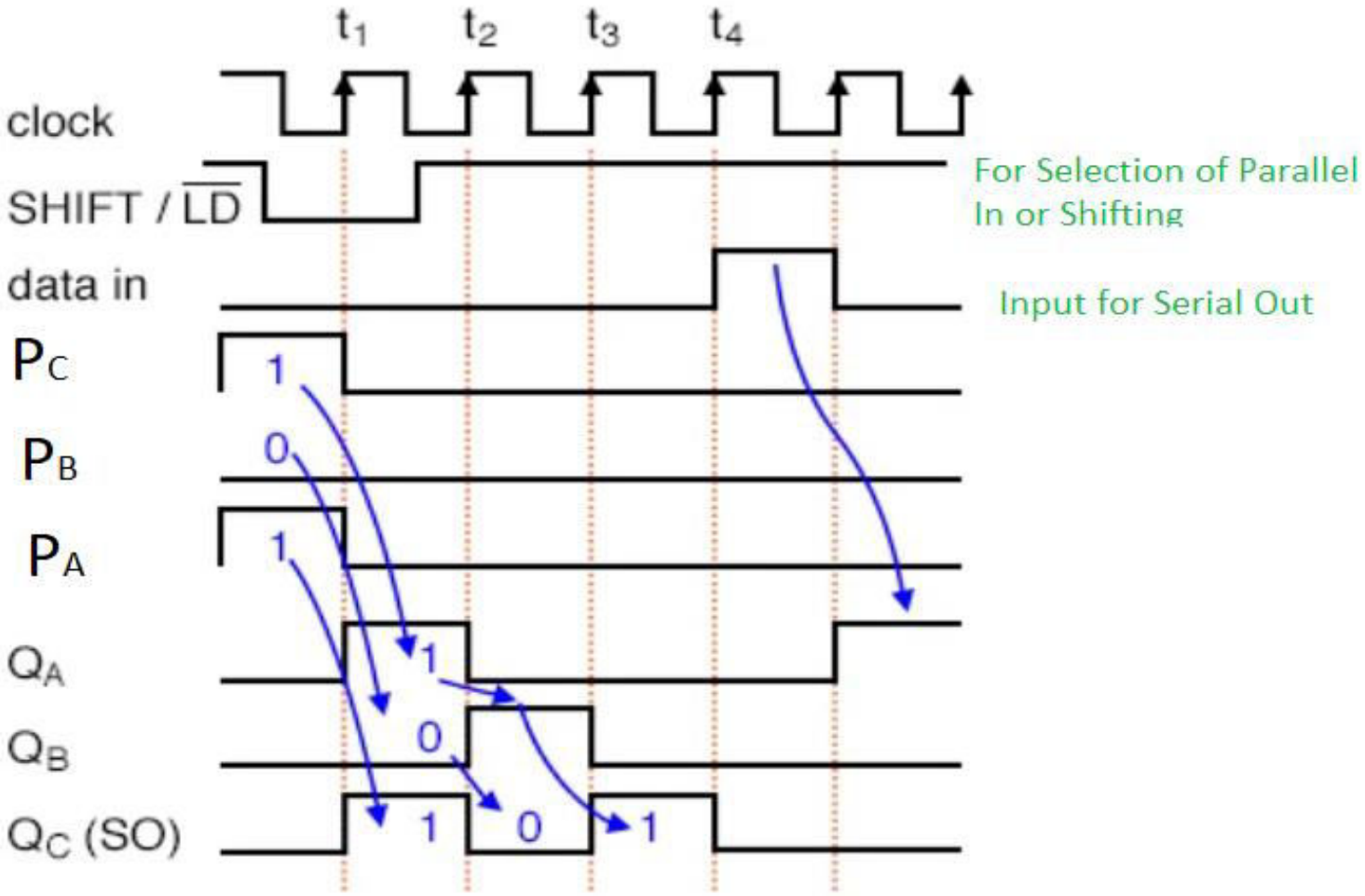
Timing diagram showing the loading of the serial input 0101 into the 4-bit serial-in, serial-out, shift register.

Parallel In/Serial Out Shift Registers

- Bits are entered simultaneously, but output is serial.
- The data is loaded into the register in a parallel in which all the data bits enter their inputs simultaneously, to the parallel input pins P_A to P_D of the register.
- The data is read out sequentially in the normal shift-right mode from the register at Q representing the data present at P_A to P_D.
- It is important to note that in this type of shift register a clock pulse is not required to parallel load the register as it is already present, but four clock pulses are required to unload the data.

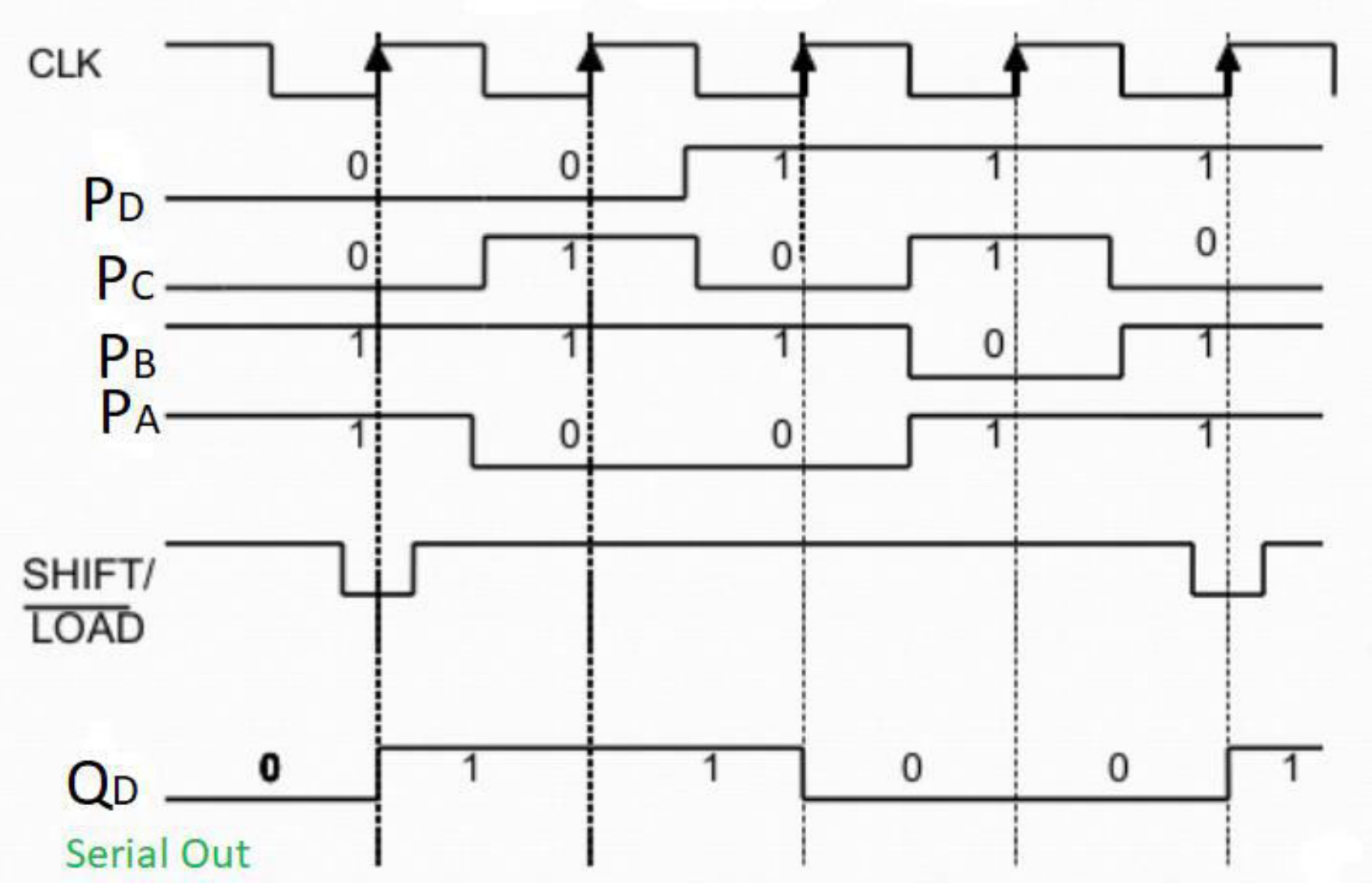


Parallel In/Serial Out Shift Registers



Parallel-in/ serial-out shift register load/ shift waveforms

Parallel In/Serial Out Shift Registers:Timing Diagram



Parallel In/Parallel Out Shift Registers

- Simultaneous input and output of all data bits.
- The data is presented in a parallel format to the parallel input pins P_A to P_D and then transferred together directly to their respective output pins Q_A to Q_D by the same clock pulse.

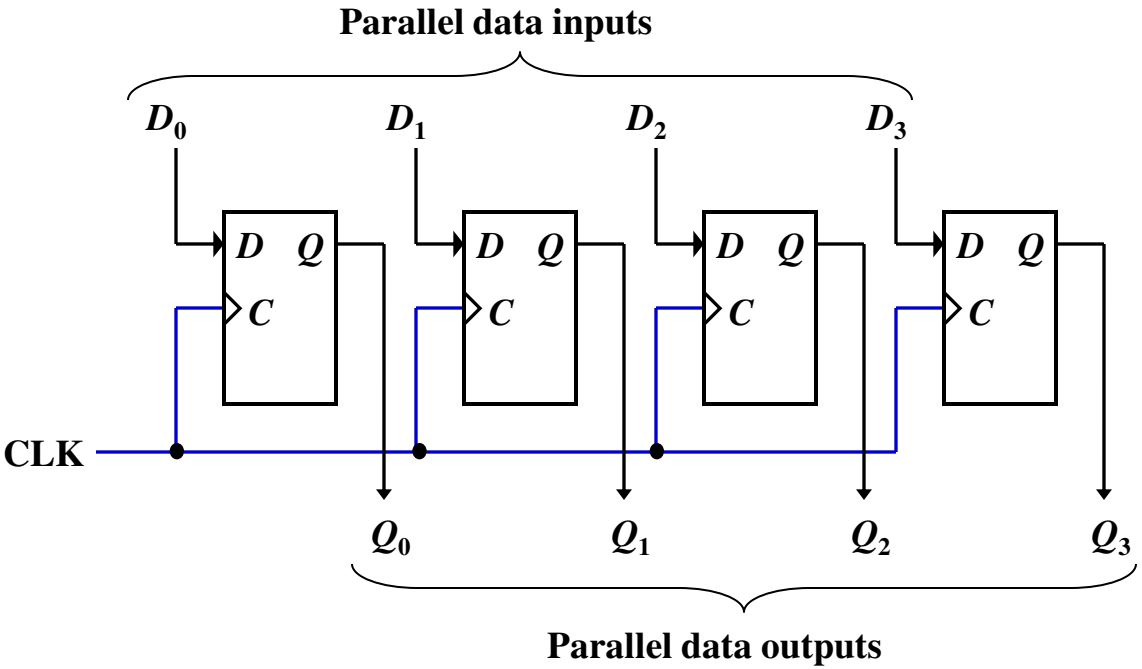


Fig parallel in parallel out shift register logic diagram

Parallel In/Parallel Out Shift Registers

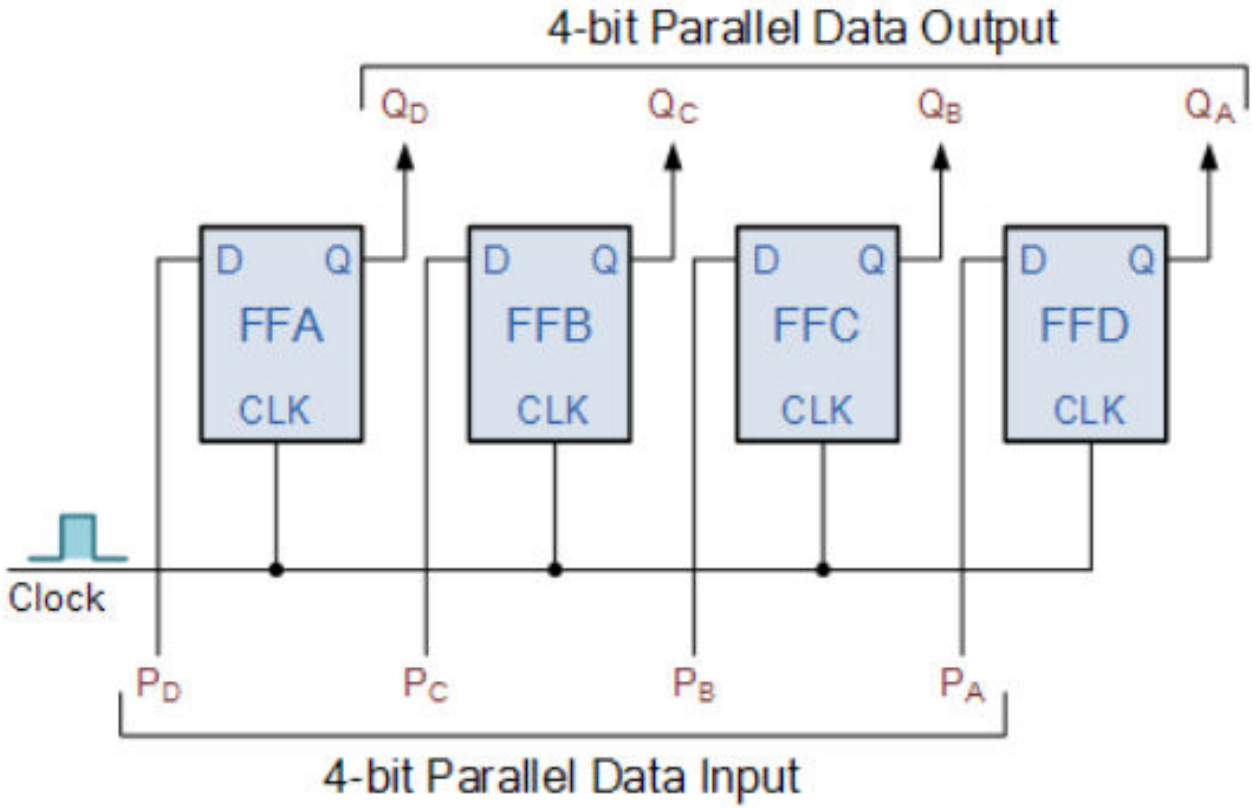
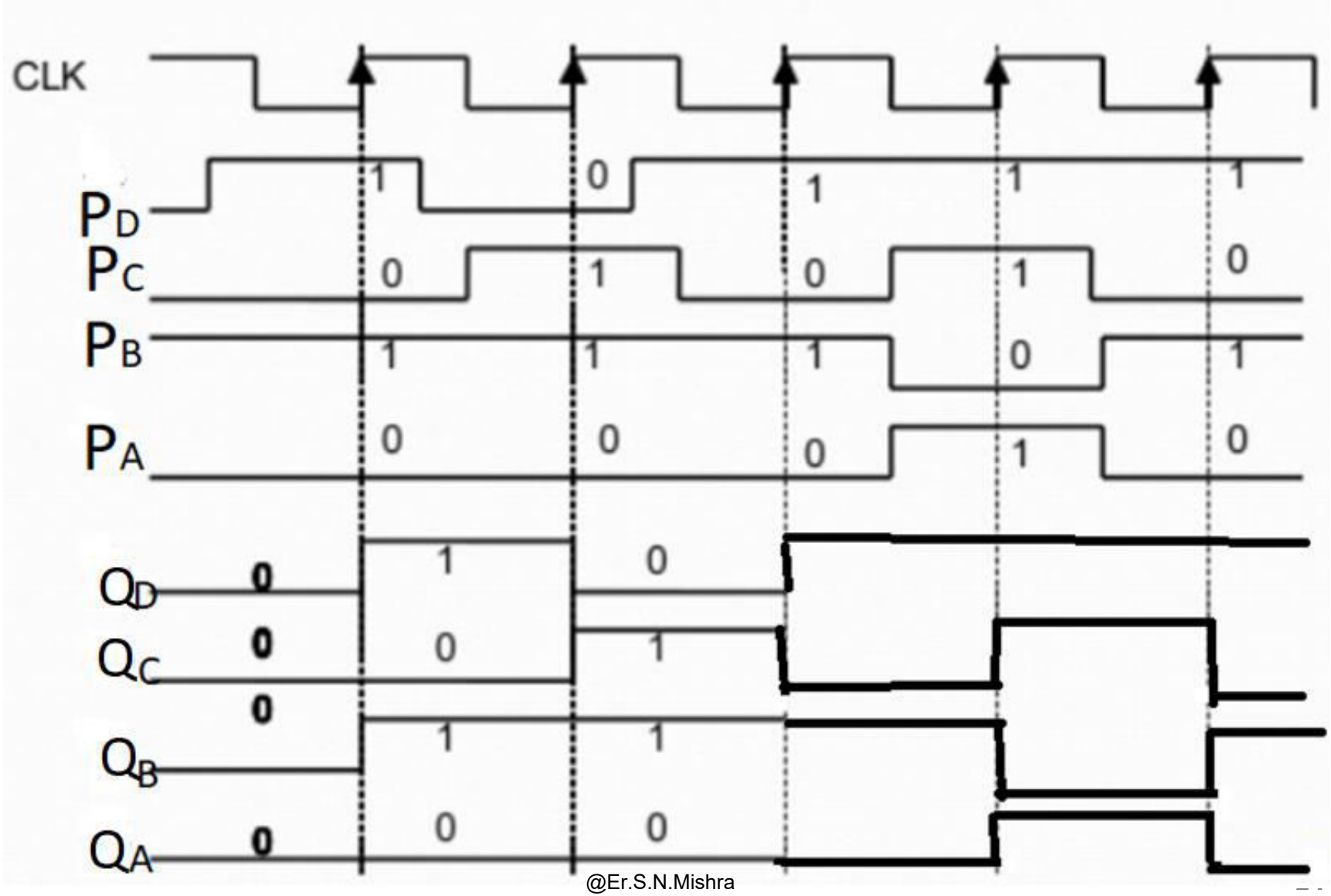


Fig Parallel in Parallel out shift register logic diagram

Parallel In/Parallel Out Shift Registers: Timing Diagram



Test Questions:

- Explain how to convert serial data to parallel and parallel data to serial. What type of register is needed?
  - For serial data to parallel= 4-bit Serial-in to Parallel-out Shift Register
  - For parallel data to serial =4-bit Parallel-in to Serial-out Shift Register

Counters

- A Counter is a register that goes through a predetermined sequence of states upon the application of clock pulses.
- Counters are circuits that cycle through a specified number of states.
- Two types of counters:
  - ❖ synchronous (parallel) counters
  - ❖ asynchronous (ripple) counters
- Ripple counters
  - The clock Signal (CLK) is only used to clock the first FF
  - Flip-flop outputs is used as a source of clock for other flip-flops except First FF.
- Synchronous counters
  - Apply the same clock to all flip-flops.
  - Thus the output will change at the same time.

Asynchronous Vs synchronous Counters

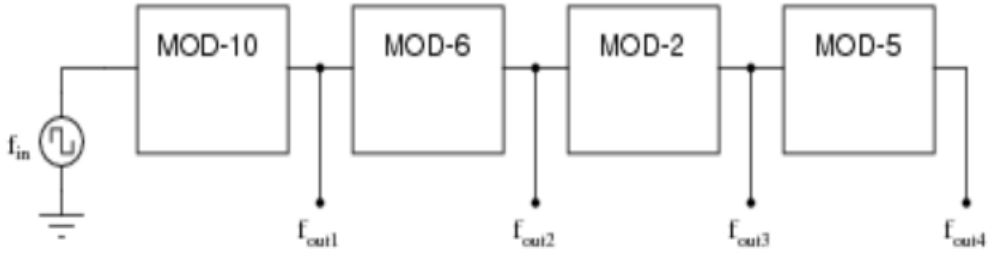
S.N.	Asynchronous Counters	Synchronous Counters
1	In this type of counter flip-flops are connected in such a way that output of first flip-flop drives the clock for the next flip-flop.	In this type there is no connection between output of first flip-flop and clock input of the next flip-flop.
2	All the flip-flops are not clocked simultaneously.	All the flip-flops are clocked simultaneously.
3	Logic circuit is very simple even for more number of states.	Design involves complex logic circuit as number of states Increases.
4	Main drawback of these counters is their low speed as the clock is propagated through number of flip-flops before it reaches last flip-flop.	As clock is simultaneously given to all flip-flops there is no problem of propagation delay. Hence they are preferred when number of flip-flops increases in the given design.

Asynchronous (Ripple) Counters

- Asynchronous counters: the flip-flops do not change states at exactly the same time as they do not have a common clock pulse.
- Also known as ripple counters, as the input clock pulse "ripples" through the counter – cumulative delay is a drawback.
- n flip-flops $\rightarrow$ a MOD (modulus) 2^n counter. (Note: A MOD-x counter cycles through x states.)
- Output of the last flip-flop (MSB) divides the input clock frequency by the MOD number of the counter, hence a counter is also a *frequency divider*.

Counters Concept: Question

When counters are used as frequency dividers, they are often drawn as simple boxes with one input and one output each, like this:



Calculate the four output frequencies (f_{out1} through f_{out4}) given an input frequency of 1.5 kHz:

f_{out1} =

f_{out2} =

f_{out3} =

f_{out4} =

Answers:

f_{out1} = 150 Hz

f_{out2} = 25 Hz

f_{out3} = 12.5 Hz

f_{out4} = 2.5 Hz

Asynchronous (Ripple) Counters

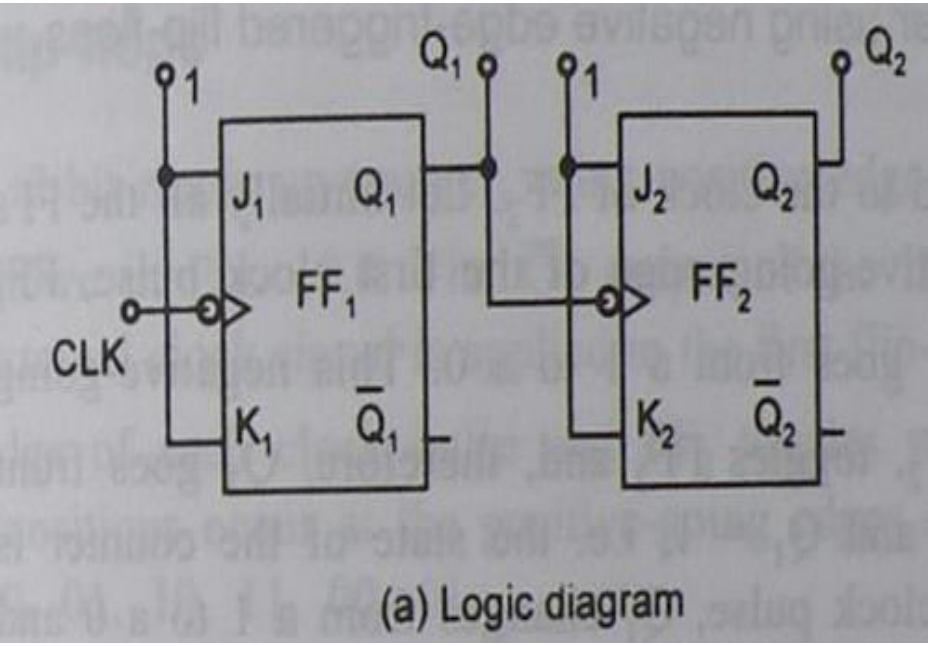
- Idea:
  - to connect the output of one flip-flop to the C input of the next high-order flip-flop

✓ We need "complementing" flip-flops

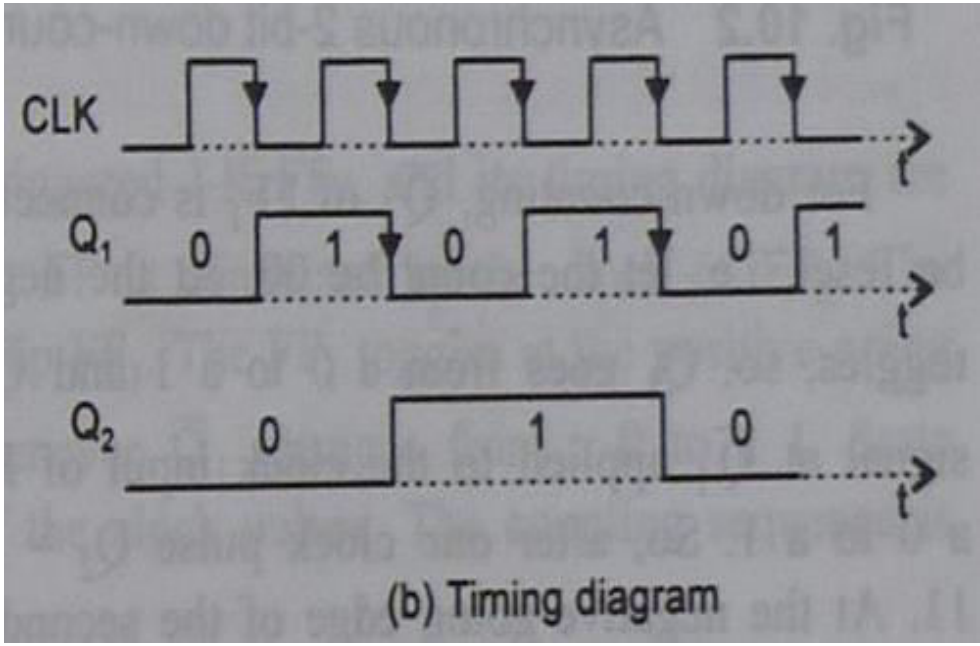
  - We can use T flip-flops to obtain complementing flip-flops or
  - JK flip-flops with its inputs are tied together

2 bit Ripple binary up Counters

- Output of one flip-flop is connected to the clock input of the next more-significant flip-flop.



(a) Logic diagram



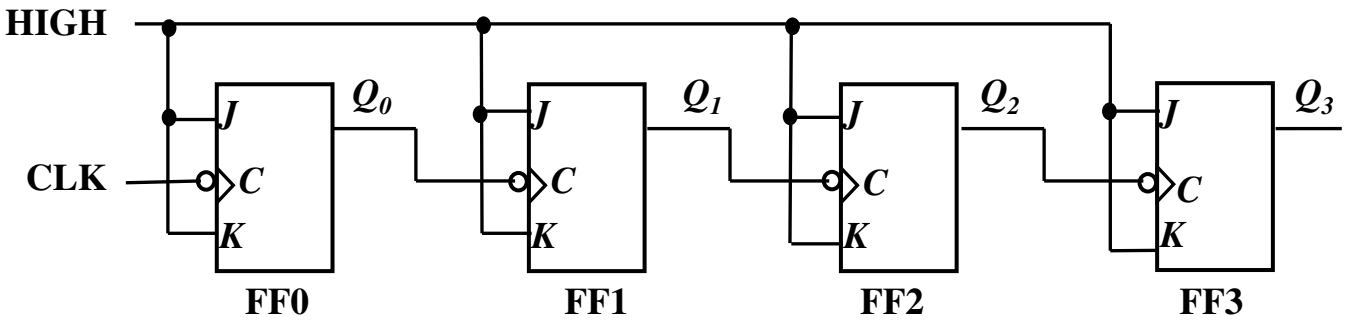
(b) Timing diagram

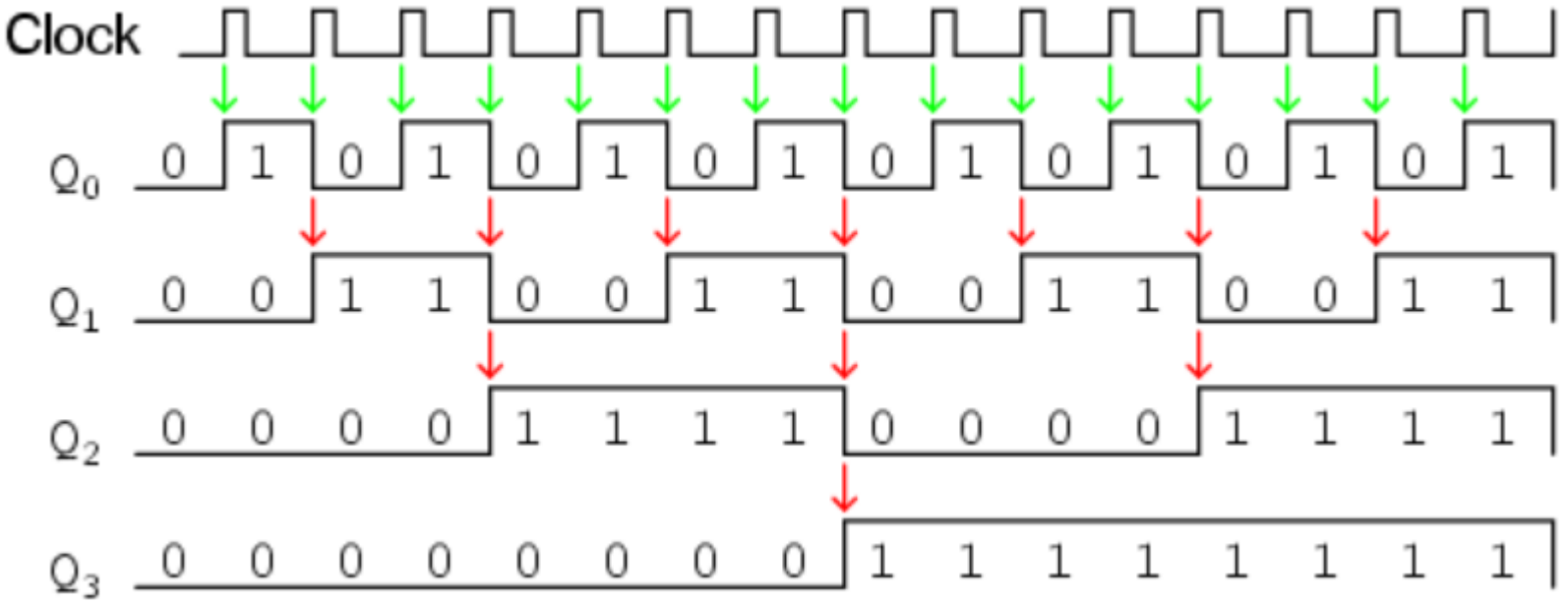
Timing diagram

$00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00$ ...

Asynchronous (Ripple) Counters

- Example: 4-bit ripple binary counter (negative-edge triggered).





4 bit Binary Ripple Counter

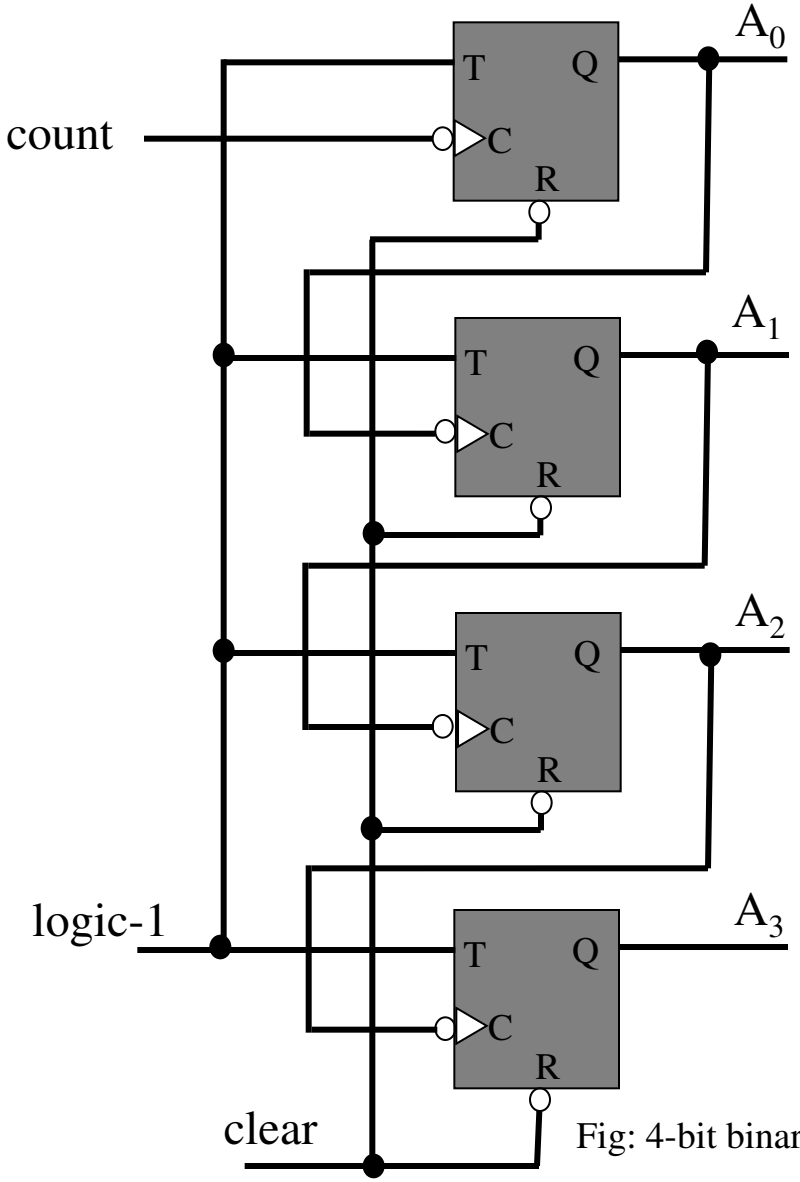


Fig: 4-bit binary ripple counter

0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

Fig: count sequence

4 bit Binary Ripple Counter

- Operation
- It is obvious that the lowest-order bit A_1 must be complemented with each count pulse.
- Every time A_1 goes from 1 to 0, it complements A_2.
- Every time A_2 goes from 1 to 0, it complements A_3.
- Every time A_3 goes from 1 to 0, it complements A_4, and so on.
- The flip-flops change one at a time in rapid succession, and the signal propagates through the counter in a *ripple* fashion.

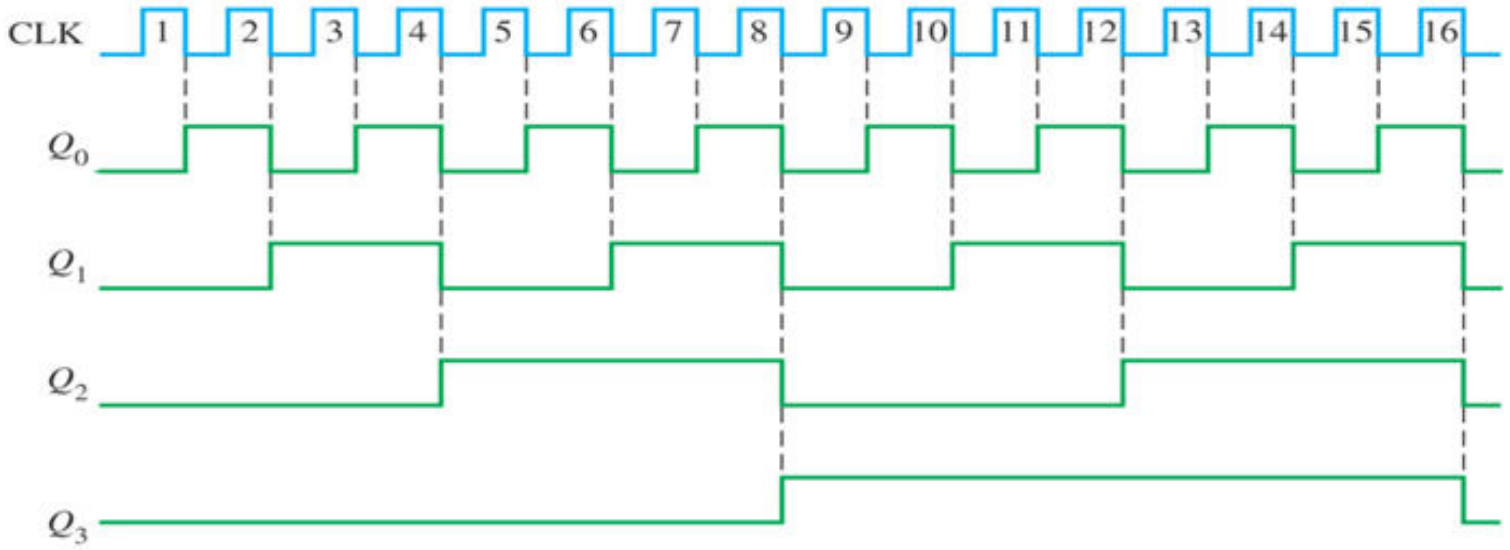
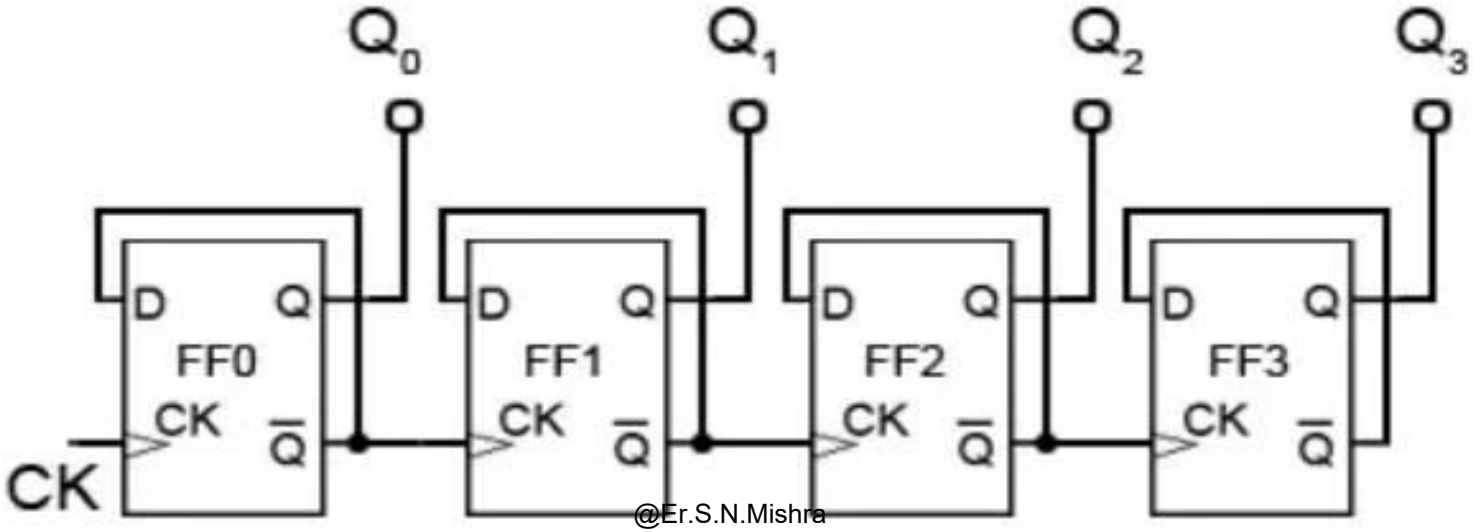


Fig: Timing diagram

Binary Ripple Counter: Questions

- Design a 4-bit binary ripple counter with D Flip Flop.
  - D-flop flop can be set to toggle its state upon the clock edge if its complemented output is feedback to its input. In such setup, D-flip flop can act as a T-flip flop with input T = 1.
  - Since the D-flip flop can be set to act as a T-flip flop, we can use the same design of T-flip flop up counter by replacing T-flip flop with D-flip flop. The input of each individual D-flip flop will be connected its complemented output D = $\overline{Q}$. Whenever the clock edge hits the flip-flop will toggle its state.



BCD Ripple Counter (Decade Counter)

- These counters have 10 states in their sequence.
- A decade counter of states from 0000 to 1001 is called a **BCD decade counter.**
- Since the counter would have 16 states, it is forces to recycle before going through all of its possible state.
- The **BCD decade counter** is recycled to 0000 after the 1001 state.
- Such a counter must have at least four flip-flops to represent each decimal digit, since a decimal digit is represented by a binary code with at least four bits.
- The sequence of states in a decimal counter is dictated by the binary code used to represent a decimal digit.

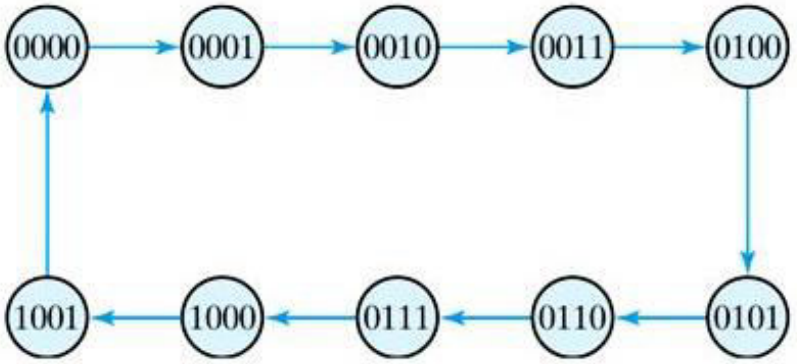


Fig: state diagram

BCD Ripple Counter

Q_8	Q_4	Q_2	Q_1
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
1	0	0	1

Fig: count sequence

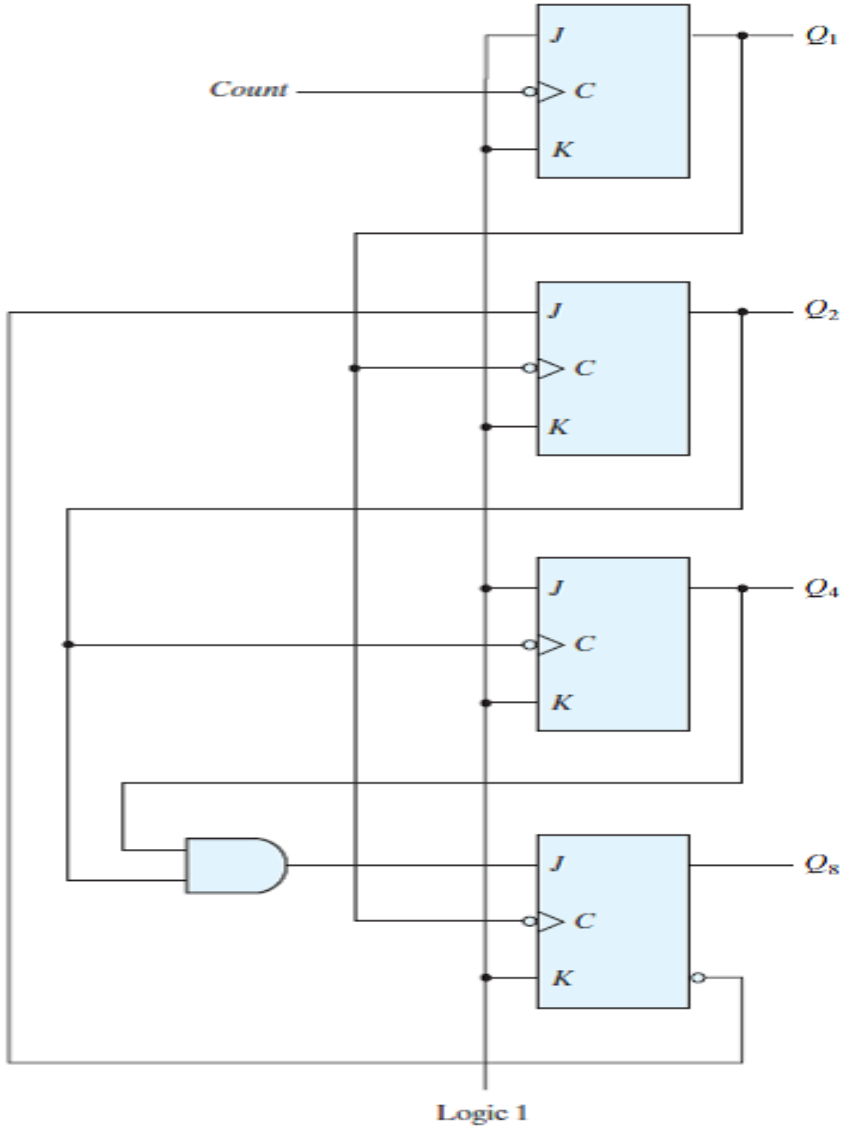


FIGURE
BCD ripple counter

BCD Ripple Counter

- **Operation**:
- When CP input goes from 1 to 0, the flip-flop is set if J = 1, is cleared if K = 1, is complemented if J = K = 1, and is left unchanged if J = K = 0. The following are the conditions for each flip-flop state transition:

1. Q_1 is complemented on the negative edge of every count pulse.
2. Q_2 is complemented if $Q_8 = 0$ and Q_1 goes from 1 to 0. Q_2 is cleared if $Q_8 = 1$ and Q_1 goes from 1 to 0.
3. Q_4 is complemented when Q_2 goes from 1 to 0.
4. Q_8 is complemented when Q_4 $Q_2 = 11$ and Q_1 goes from 1 to 0. Q_8 is cleared if either Q_4 or $Q2$ is 0 and $Q1$ goes from 1 to 0.

BCD Ripple Counter

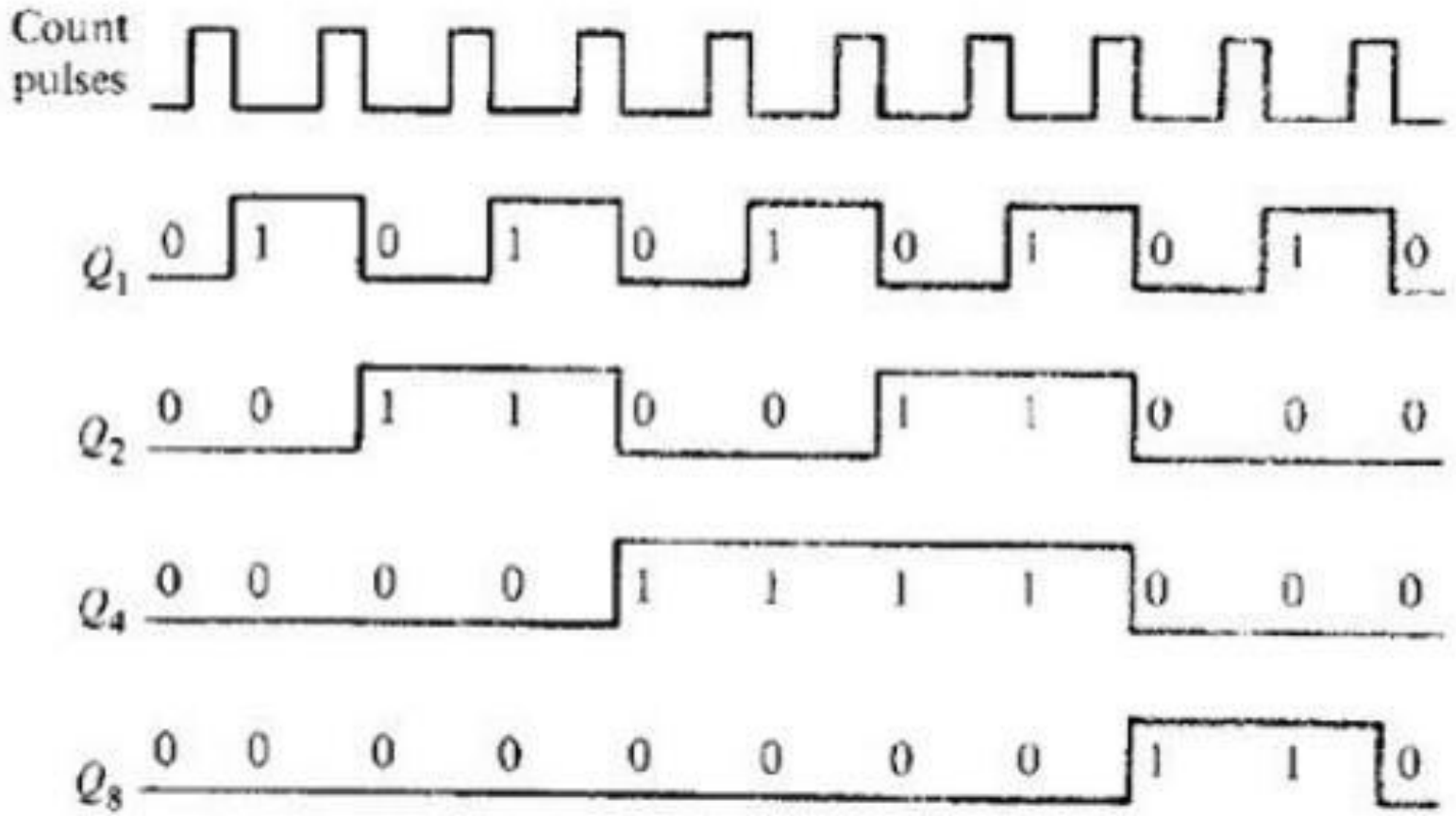


Fig: timing diagram

Synchronous (Parallel) Counters

- Synchronous counters are distinguished from ripple counters in that clock pulses are applied to the *CP* inputs of *all* flip-flops.
- Synchronous counters are counters in which all the flip-flops are triggered simultaneously(in parallel) by the clock input pulses.
- All the FFs change state simultaneously in synchronization with the clock pulse.
- The decision whether a flip-flop is to be complemented or not is determined from the values of the *J* and *K* inputs at the time of the pulse.
  - If $J = K = 0$, the flip-flop remains unchanged. If $J = K = 1$, the flip-flop complements.
- Synchronous counters have the advantage of high speed but the disadvantage of having more circuitry than that of asynchronous counters

Synchronous (Parallel) Counters

- Design procedure is so simple
  - no need for going through sequential logic design process
  - A_0 is always complemented
  - A_1 is complemented when $A_0 = 1$
  - A_2 is complemented when $A_0 = 1$ and $A_1 = 1$
  - so on

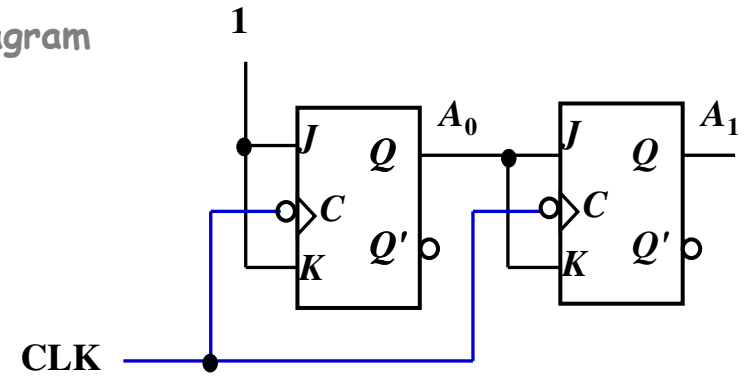
Synchronous (Parallel) Counters

- Example: 2-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J,K inputs).

Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

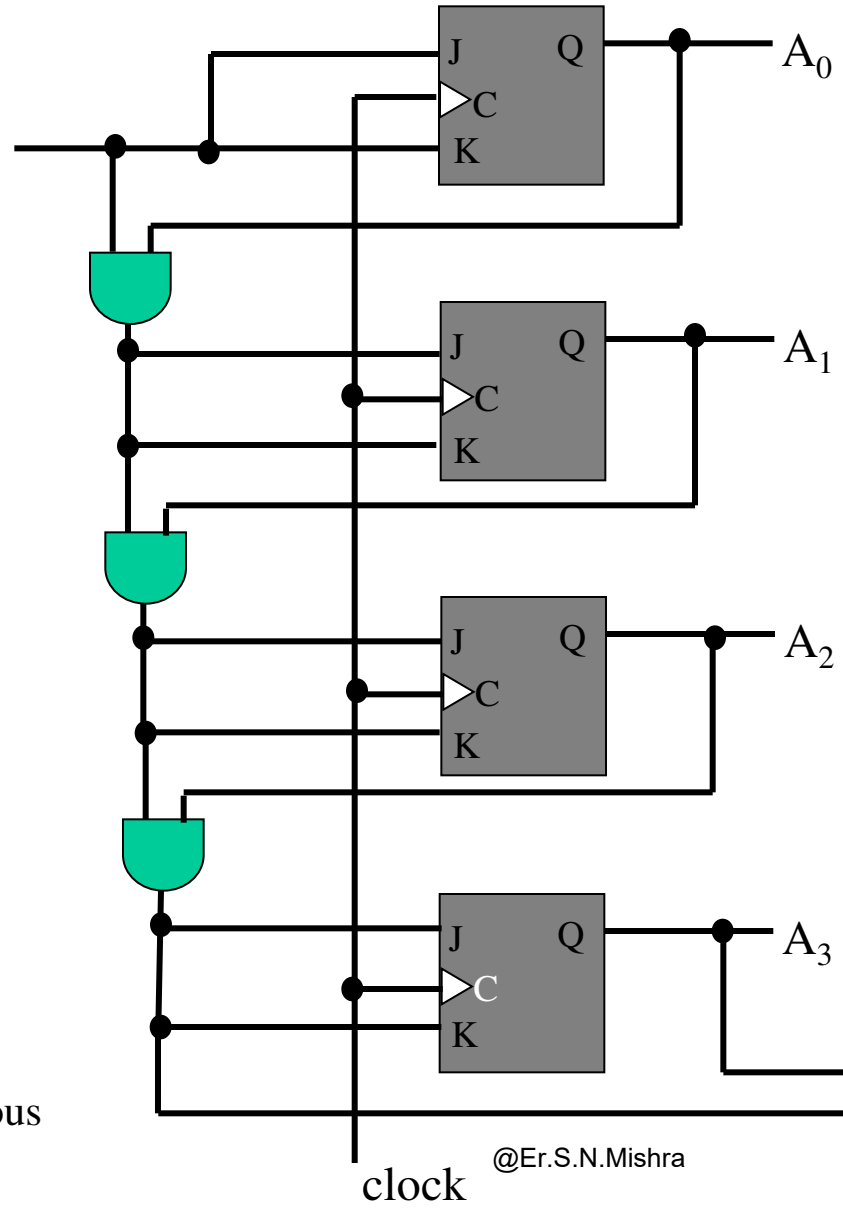
$TA_1 = A_0$
$TA_0 = 1$

State diagram



2-bit Synchronous binary Counters

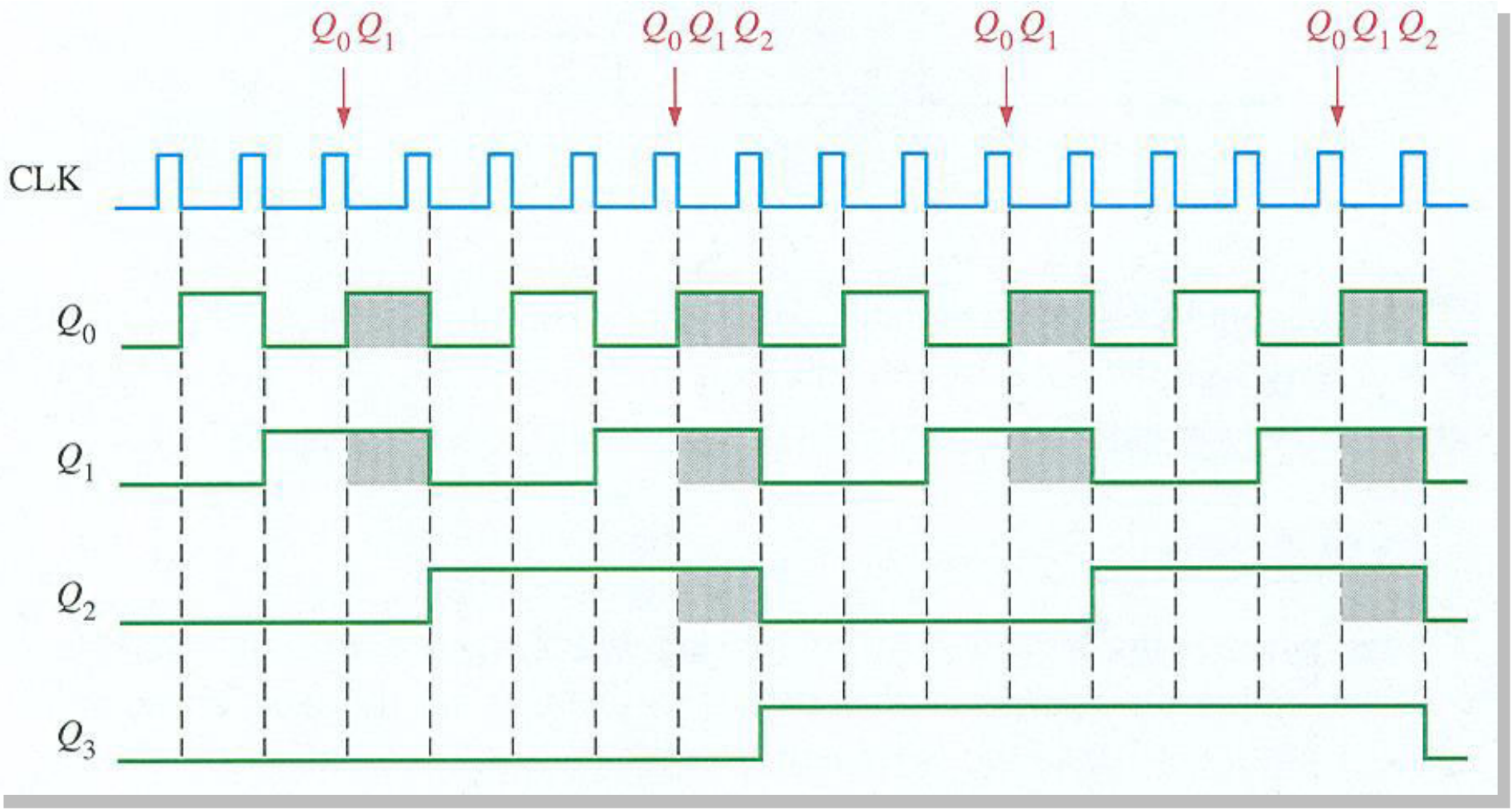
4-bit Binary Synchronous Counter



$$JA_3 = KA_3 = A_2 \cdot A_1 \cdot A_0$$
$$JA_2 = KA_2 = A_1 \cdot A_0$$
$$JA_1 = KA_1 = A_0$$
$$JA_0 = KA_0 = 1$$

Fig: 4-bit Synchronous Binary Counter

4-bit Binary Synchronous Counter

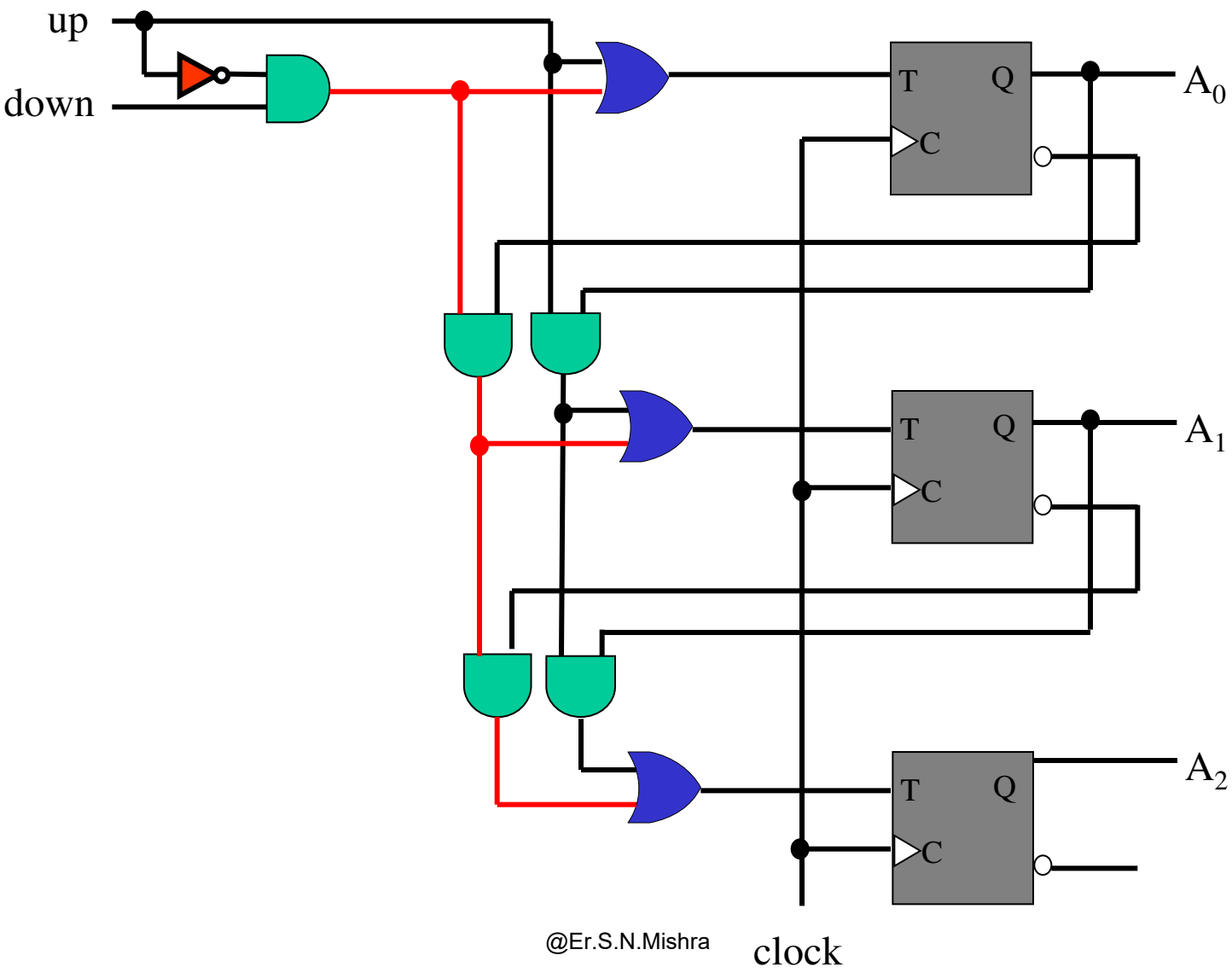


Timing diagram

Binary Up-Down Counter

- When counting downward
  - the least significant bit is always complemented (with each clock pulse)
  - A bit in any other position is complemented if all lower significant bits are equal to 0.
- Up down Counter operation
  - When **up**= 1, the circuit counts up, since the *T* inputs receive their signals from the values of the previous normal outputs of the flip-flops.
  - When **down**= 1 and **up**= 0, the circuit counts down

Binary Up-Down Counter



Synchronous BCD Counter

- A BCD Counter is nothing but a mod 10 counter.
- It requires 4 FFs
- A BCD counter counts in binary-coded decimal from 0000 to 1001 and back to 0000 where states 0000 through 1001 are stable.
- States 1010 through 1111 are invalid.
- After the tenth clock pulse, the counter resets.
- Because of the return to 0 after a count of 9, a BCD counter does not have a regular pattern as in a straight binary count.
- To derive the circuit of a BCD synchronous counter, it is necessary to go through a design procedure discussed earlier.
- The flip-flop input functions from the excitation table can be simplified by means of maps.
  - The unused states for minterms 10 to 15 are taken as don't-care terms.

Synchronous BCD Counter

CLOCK PULSE	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 (recycles)	0	0	0	0

Fig: counter sequence

Synchronous BCD Counter

Table [redacted] Excitation requirements

PS				NS				Required excitation							
Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	0	0	0	0	X	1	0	X	0	X	X	1

Synchronous BCD Counter

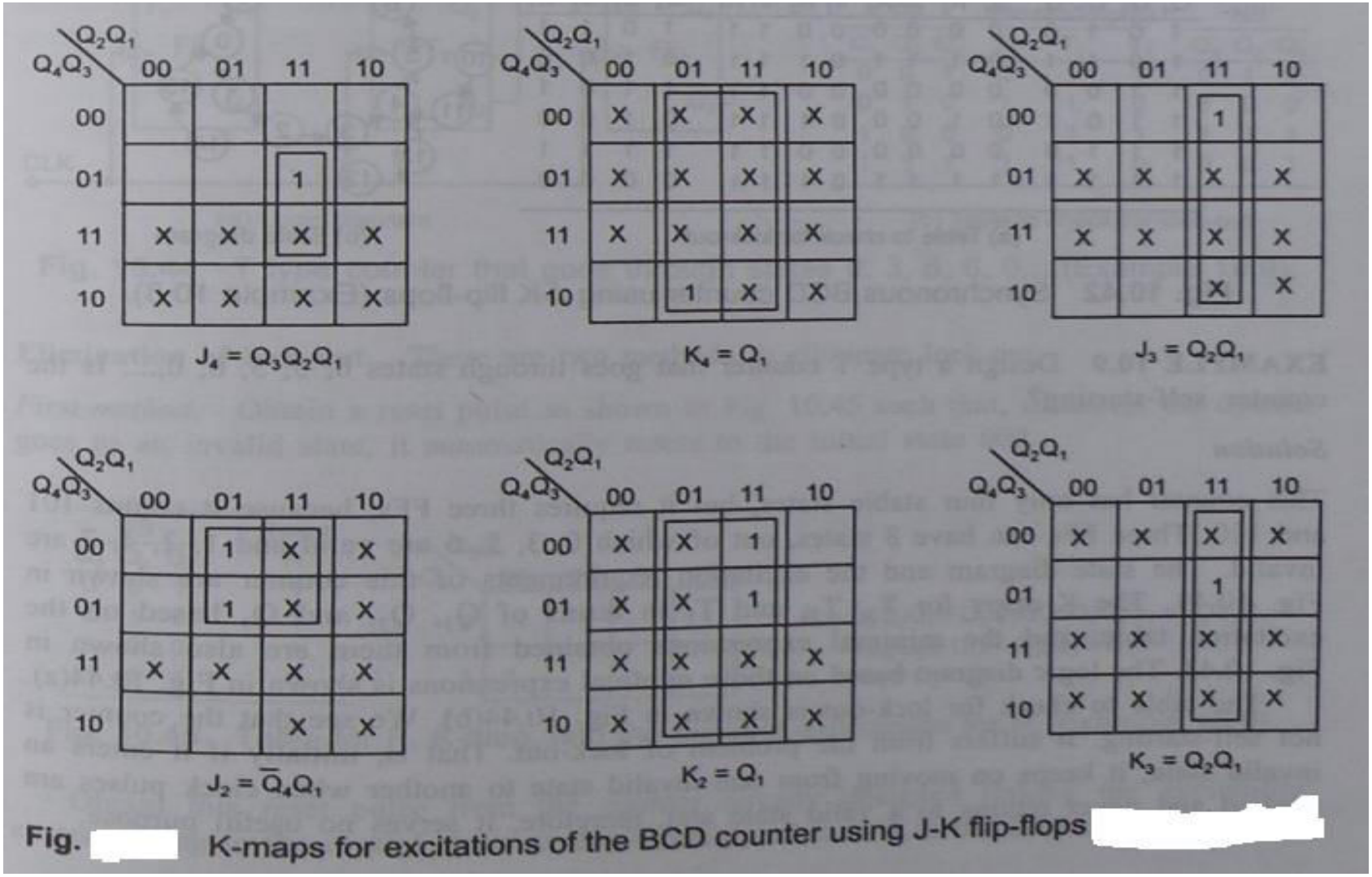


Fig. [redacted] K-maps for excitations of the BCD counter using J-K flip-flops [redacted]

Synchronous BCD Counter

- It does not have regular pattern as in binary counter, so procedure of sequential design should be used

State Table for BCD Counter

Present State				Next State				Output	Flip-Flop Inputs			
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	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	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

Use k-map to find input functions to the T flipflops as:

$T_{Q1}=1$, $T_{Q2}=Q'_8Q_1$ $T_{Q4}= Q_2Q_1$; $T_{Q8}=Q_8Q_1+Q_4Q_2Q_1$

Synchronous BCD Counter

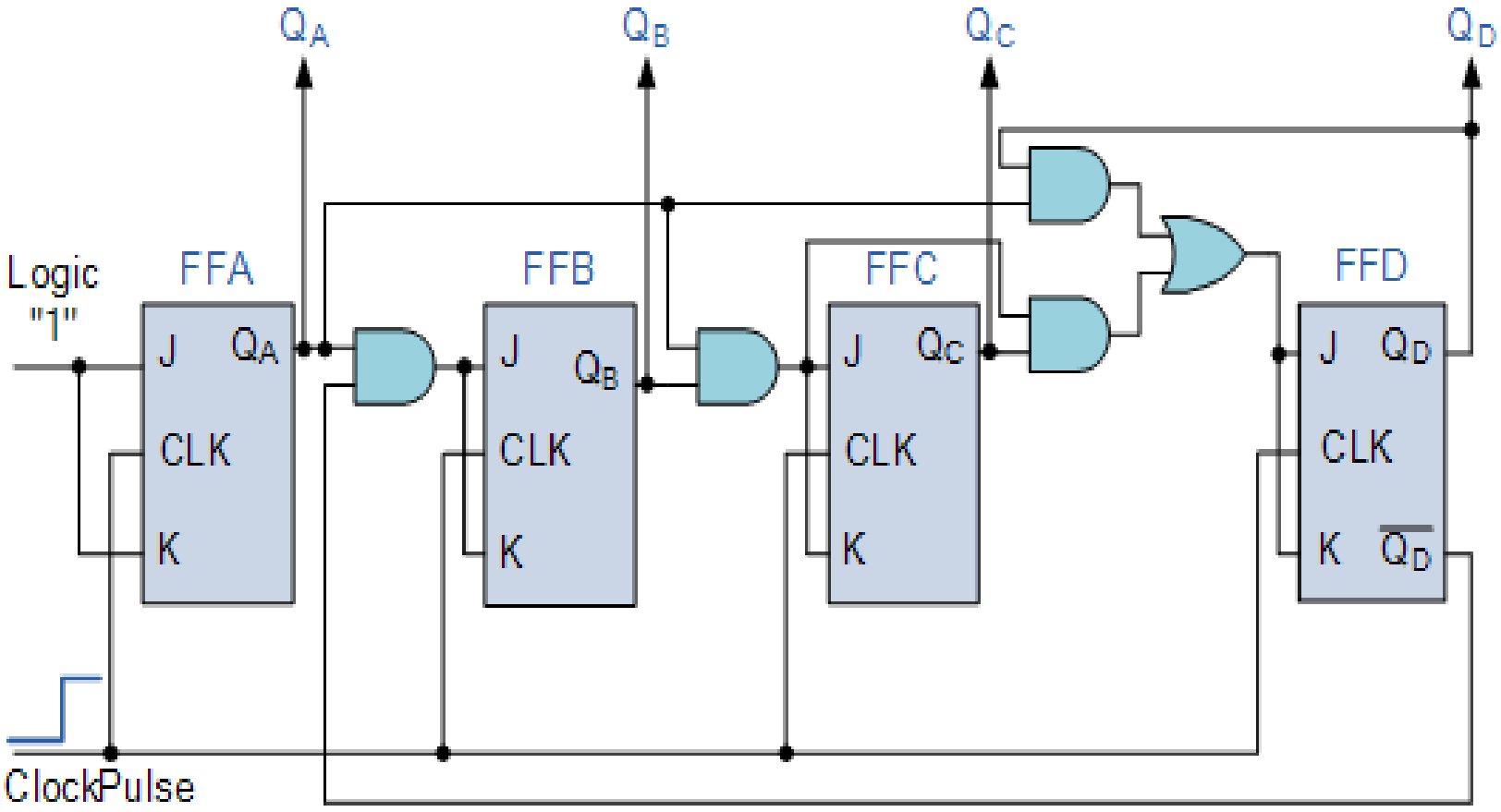
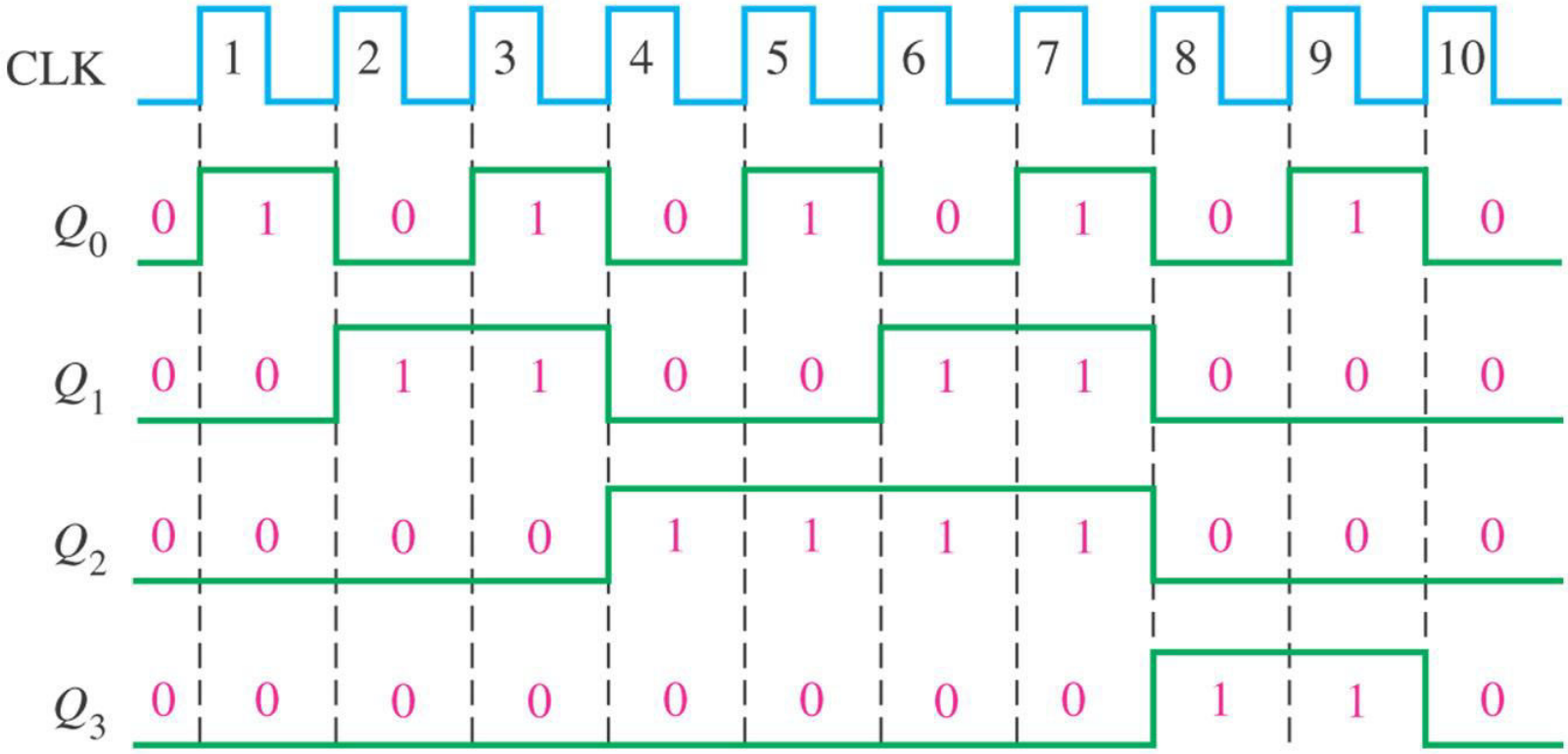


Fig: logic diagram

Synchronous BCD Counter



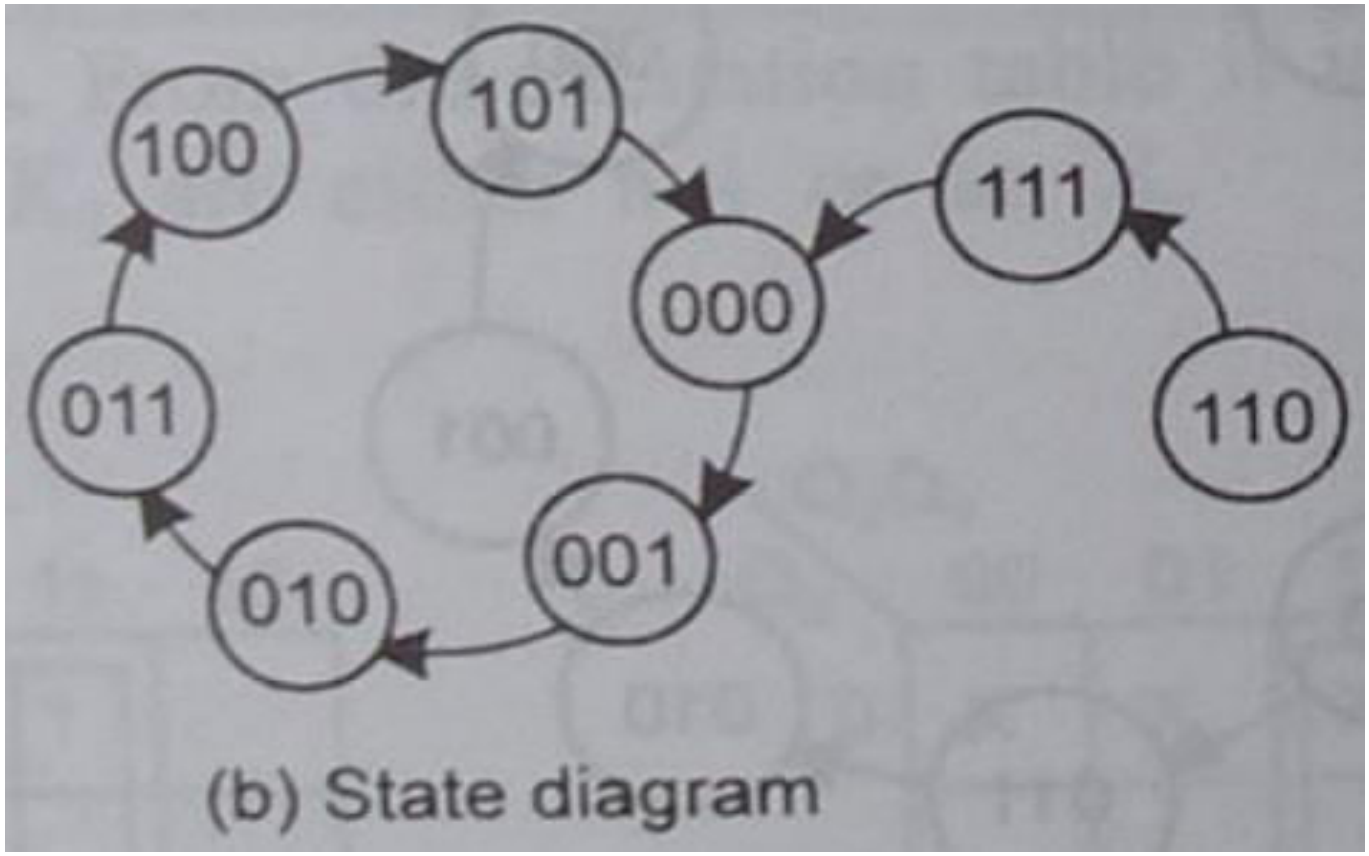
Timing diagram for the BCD decade counter (Q_0 is the LSB).

Mod 6 Counter

PS			NS			Required excitation					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
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	0	0	0	X	1	0	X	X	1

(a) Excitation table

Mod 6 Counter



(b) State diagram

Mod 6 Counter

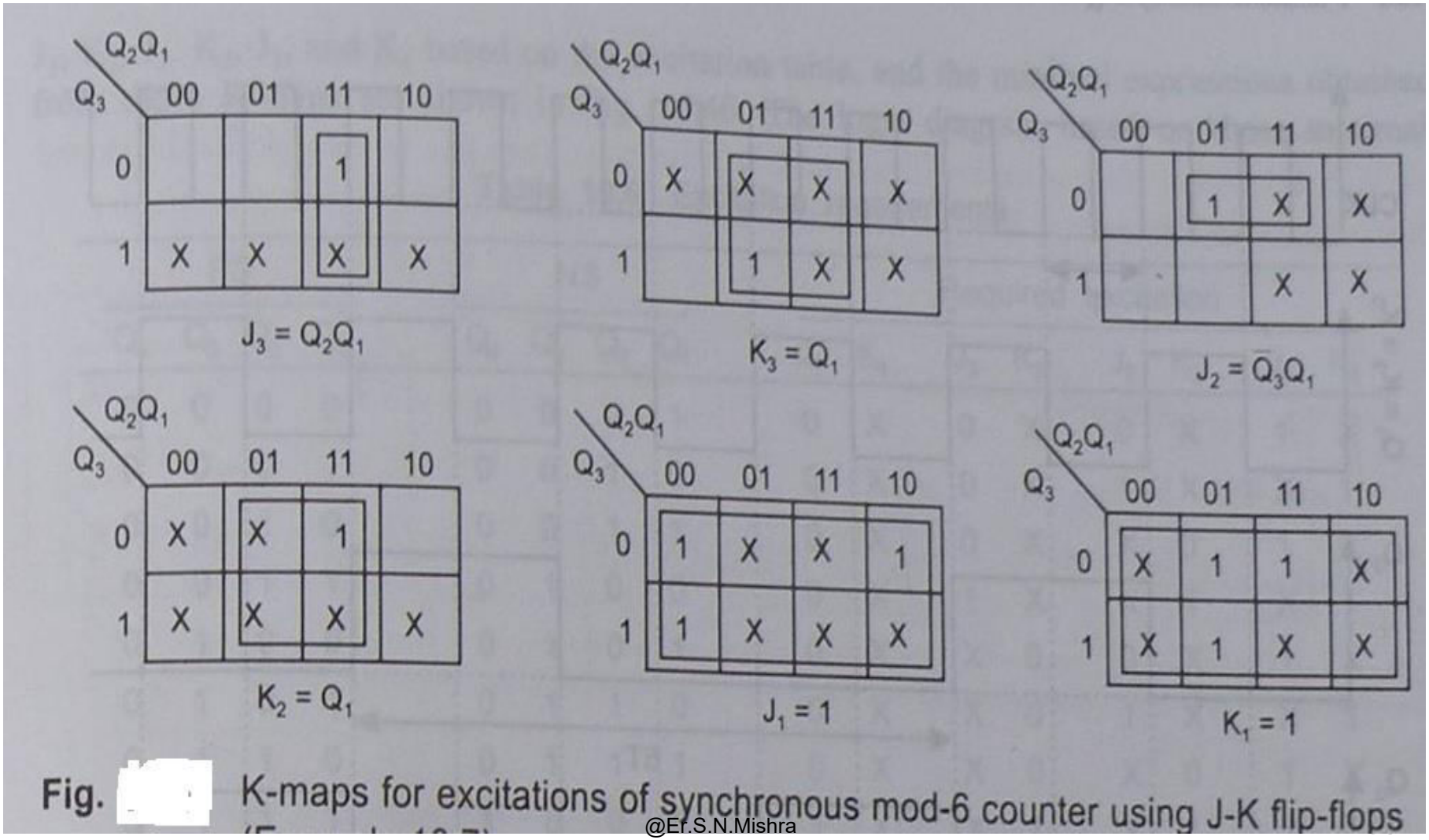
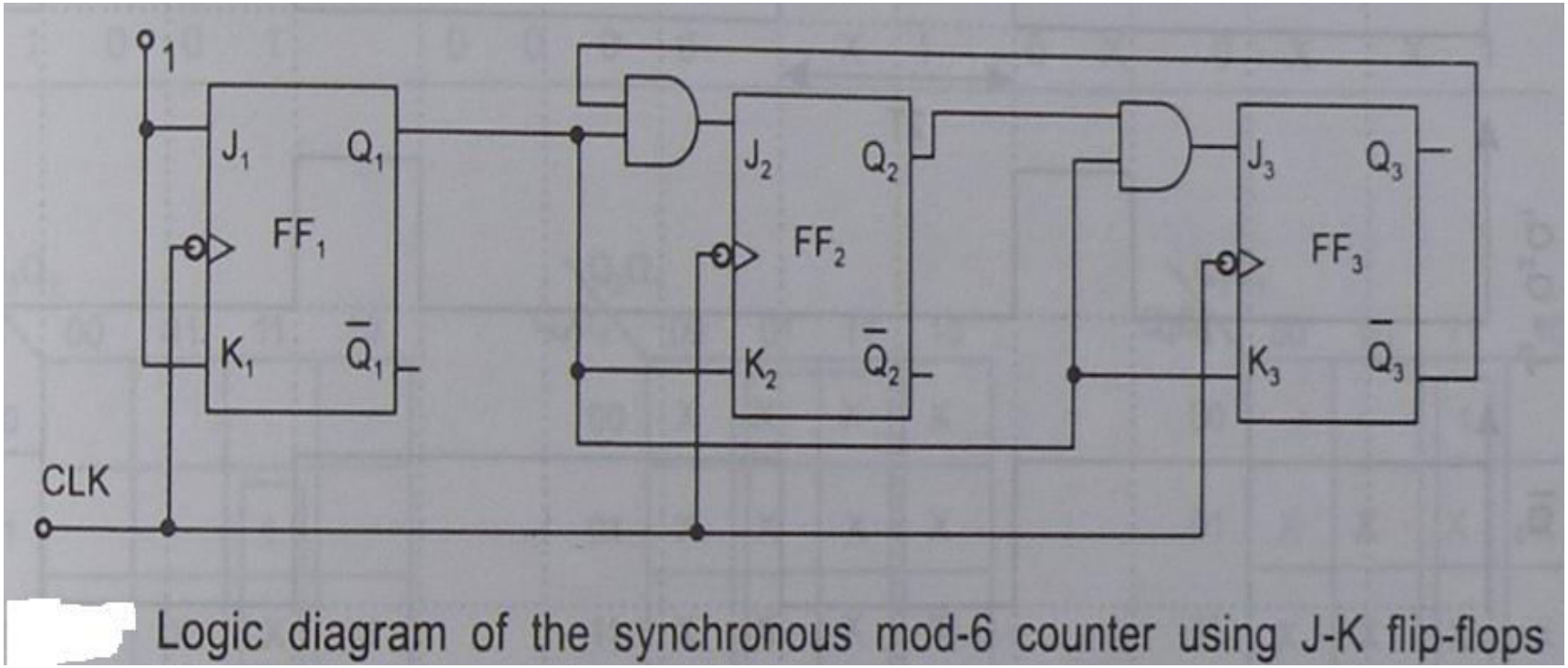


Fig. K-maps for excitations of synchronous mod-6 counter using J-K flip-flops

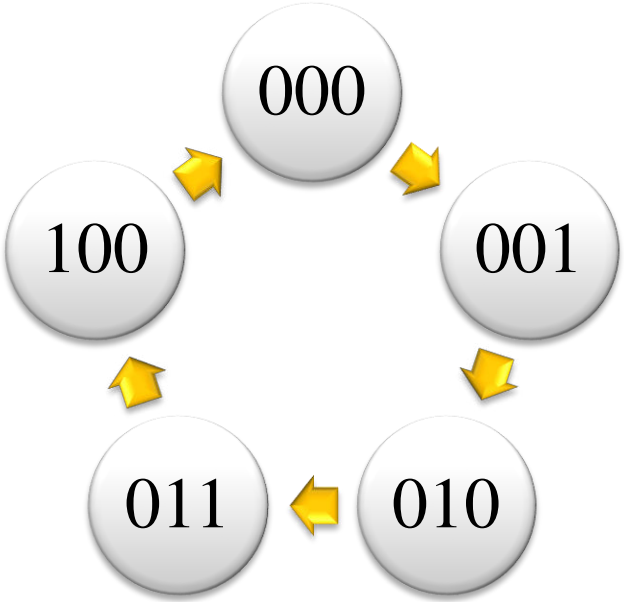
Mod 6 Counter



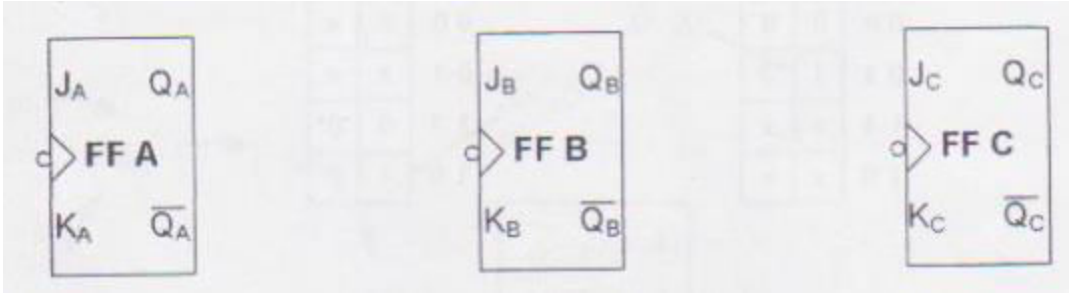
Logic diagram of the synchronous mod-6 counter using J-K flip-flops

Counter Design Example

Step 1: State diagram



- **Amount of FF needed are also determined depending on the bit**



Counter Design Example

Step 2: List all the present state along with next state

Present State			Next State		
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

Counter Design Example

Step 3: Transition Table

Output Transitions			Flip-flop Inputs	
Q_N		Q_{N+1}	J	K
0	→	0	0	X
0	→	1	1	X
1	→	0	X	1
1	→	1	X	0

Present State			Next State			FFC		FFB		FFA	
Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	J_C	K_C	J_B	K_B	J_A	K_A
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

Counter Design Example

Step 4: Karnaugh Map

- Transfer the J and K states from the transition table to K-maps
- There is a K-map for each input of each flip-flop.

Step 5: Logic Expression for FF inputs

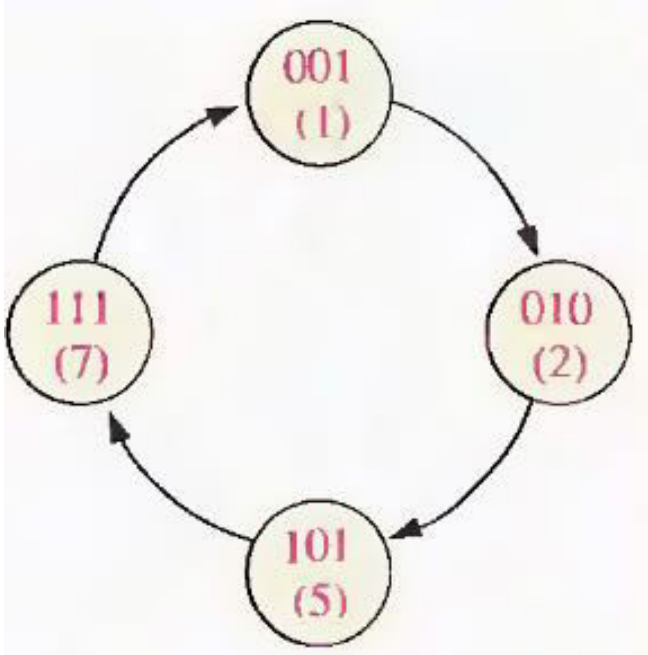
- Group the K-map cells to generate and derive the logic expression for each flip-flop input.

Step 6: Counter Implementation

- Implement the expressions with combinational logic, and combine with the flip-flops to create the counter.

Counter Design Example 2

- Design a counter with the irregular binary count sequence shown in the state diagram below. Use J-K flip-flops.



Counter Design Example 2

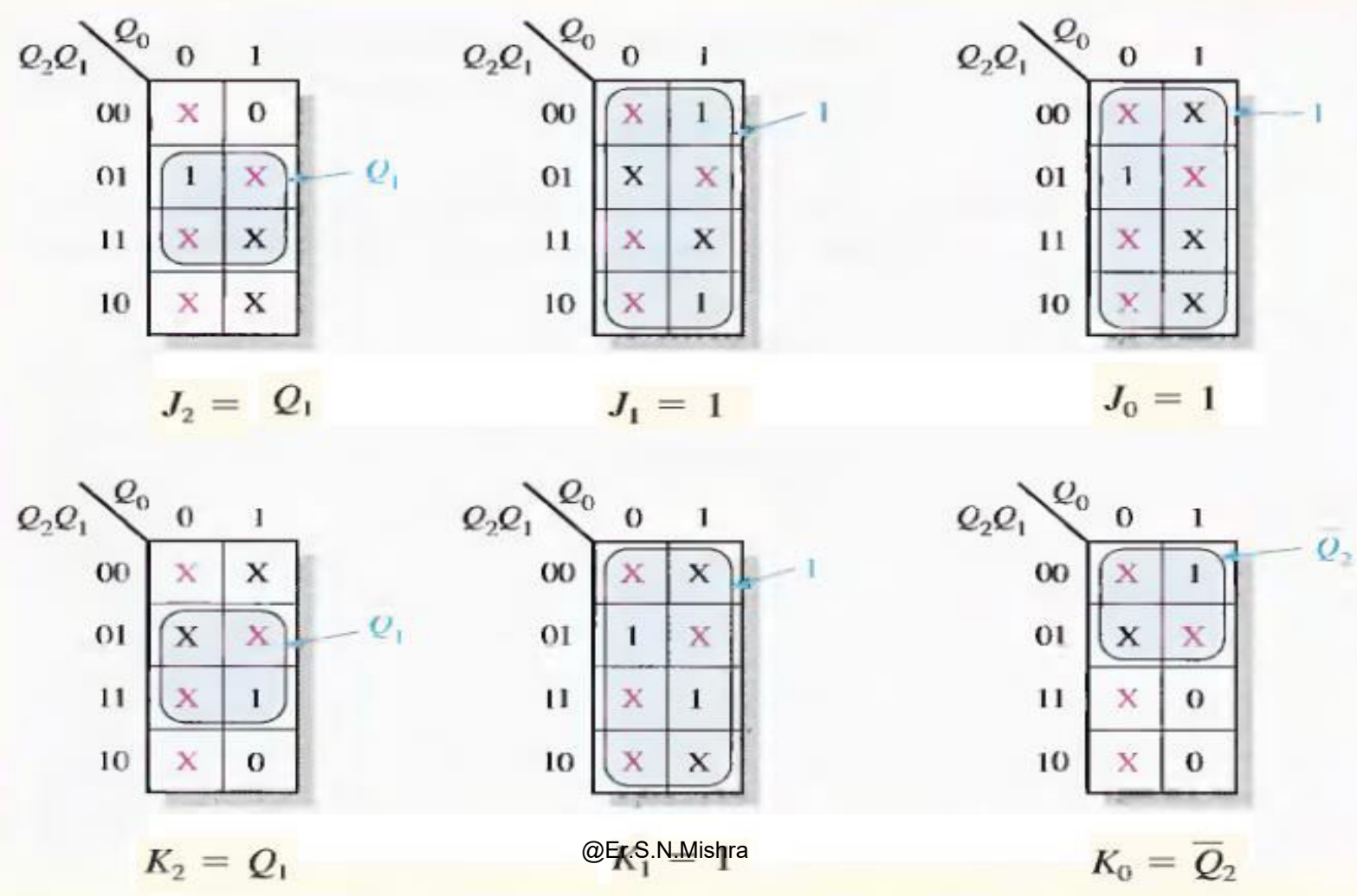
- State Table

Output		Input	
Qn	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Present State			Next State			Inputs					
Q_2	Q_1	Q_0	Q_{2+1}	Q_{1+1}	Q_{0+1}	J_2	K_2	J_1	K_1	J_0	K_0
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	1	1	X	X	1	1	X
1	0	1	1	1	1	X	0	1	X	X	0
1	1	1	0	0	1	X	1	X	1	X	0

Counter Design Example 2

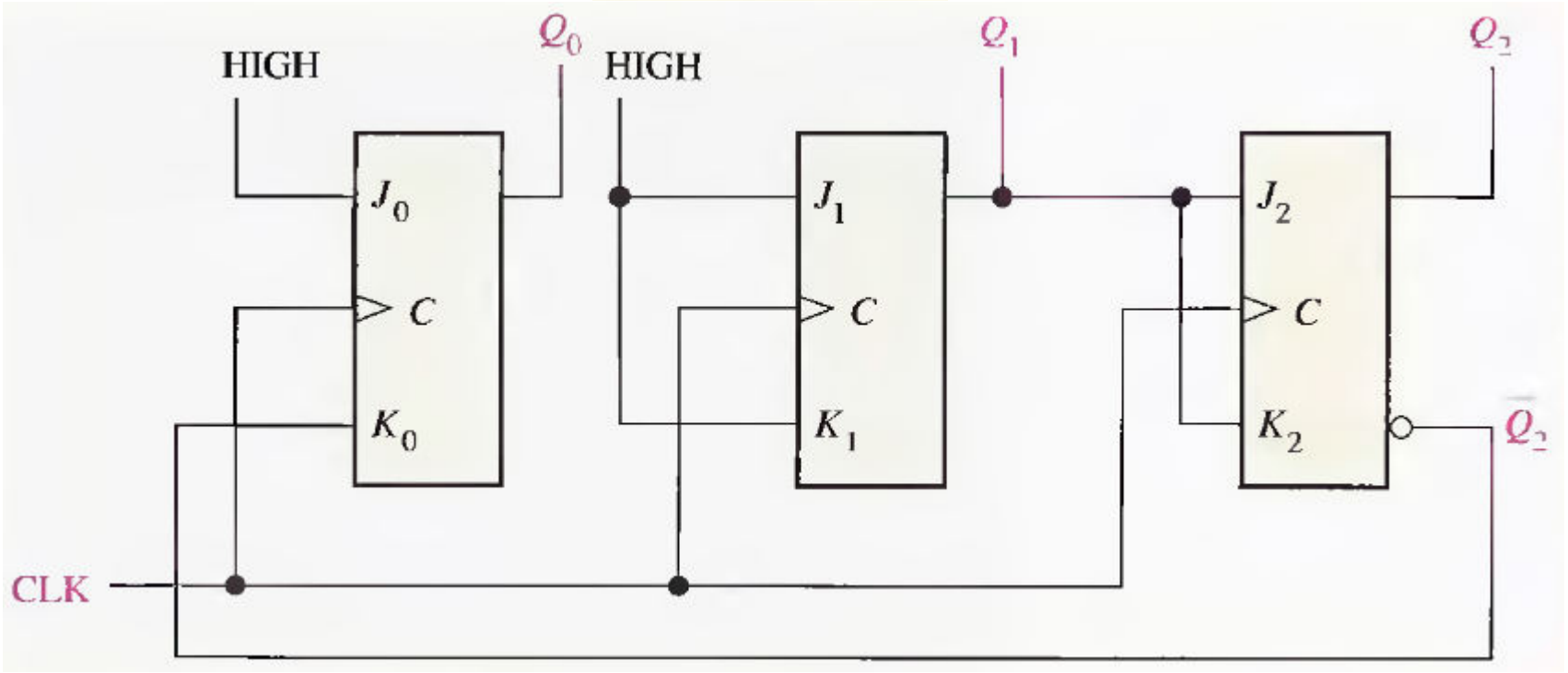
- K-Map Simplification



Counter Design Example 2

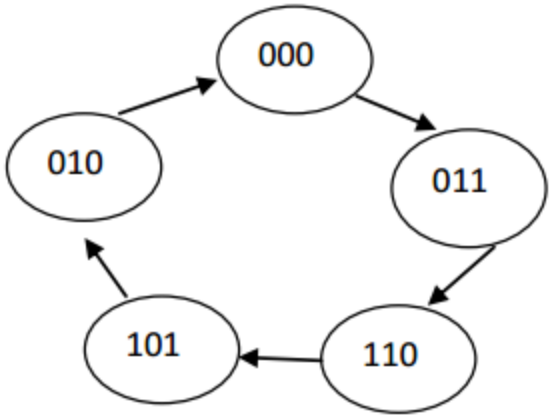
- Circuit Diagram

$$J_0 = 1, K_0 = \overline{Q}_2$$
$$J_1 = K_1 = 1$$
$$J_2 = K_2 = Q_1$$



Counter Design Example

- Designing a counter with the irregular binary sequence 1-> 2 -> 5 -> 7
- Design a counter as shown in state diagram below:

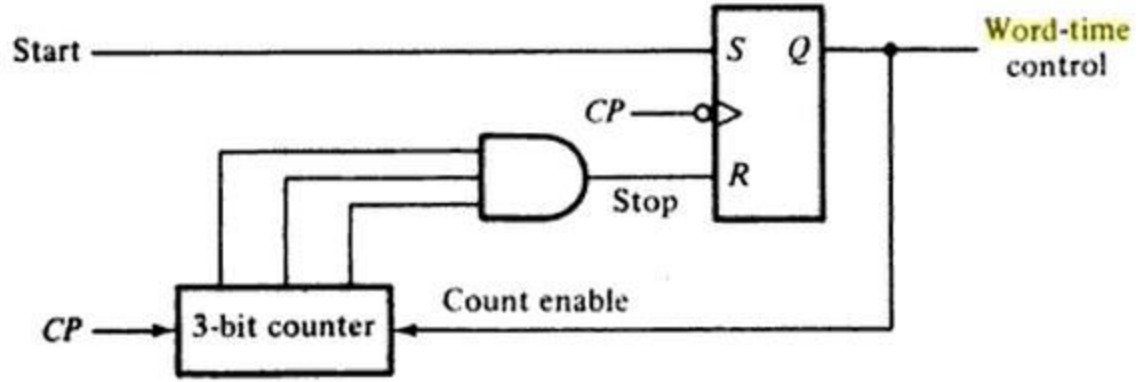


Timing Sequences

- The sequences of operations in a digital system are specified by a control unit.
- The control unit that supervises the operations in a digital system would normally consist of timing signals that determine the time sequence in which the operations are executed.
- The timing sequences in the control unit can be easily generated by means of counters or shift registers

Word-Time Generation

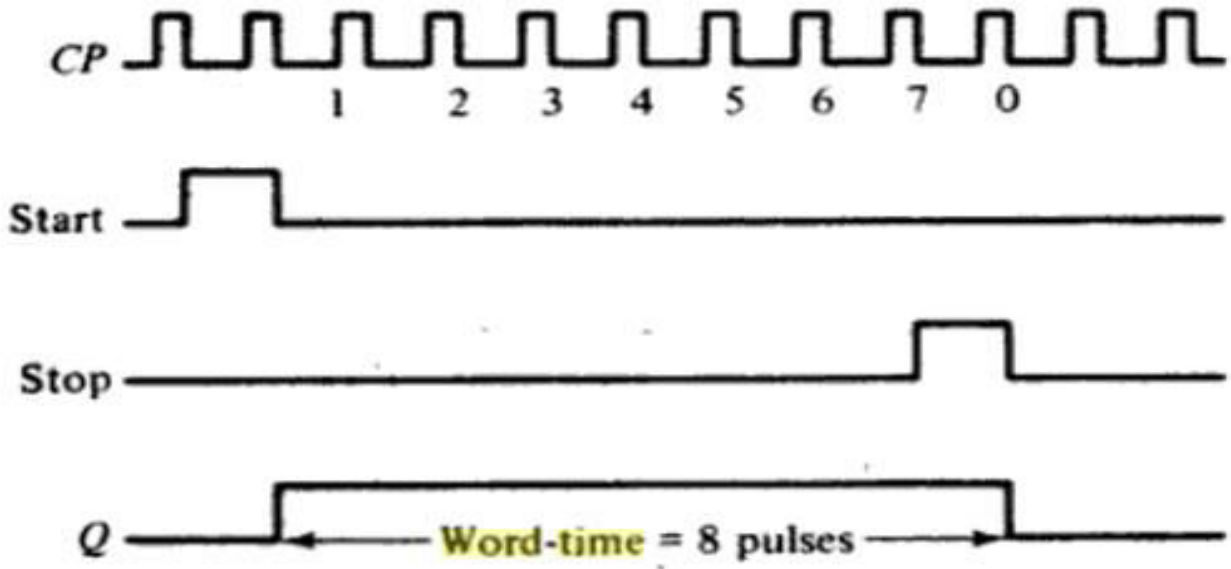
- The control unit in a serial computer must generate a *word-time* signal that stays on for a number of pulses equal to the number of bits in the shift registers.
- The word-time signal can be generated by means of a counter that counts the required number of pulses.
- Initially, the 3-bit counter is cleared to 0. A start signal will set flip-flop Q. The output of this flip-flop supplies the word-time control and also enables the counter. After the count of eight pulses, the flip-flop is reset and Q goes to 0.



(a) Circuit diagram

Word-Time Generation

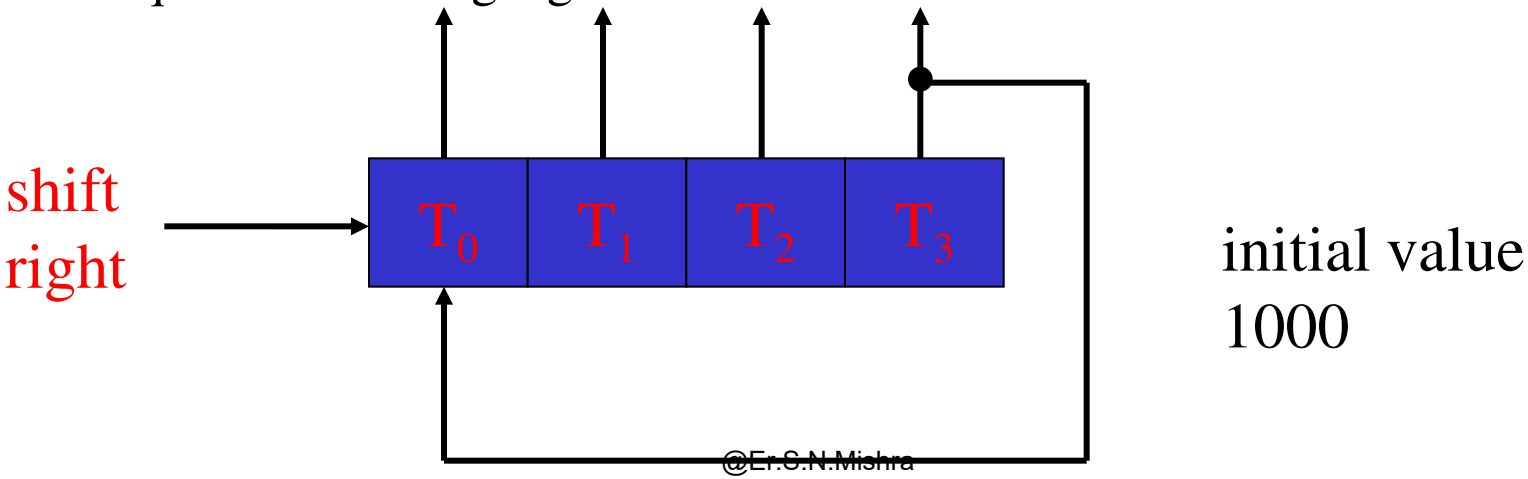
- The start signal is synchronized with the clock and stays on for one clock-pulse period.
- After Q is set to 1, the counter starts counting the clock pulses. When the counter reaches the count of 7 (binary 111), it sends a stop signal to the reset input of the flip-flop.
- The stop signal becomes a 1 after the negative-edge transition of pulse 7
- The next clock pulse switches the counter to the 000 state and also clears *Q* and the word-time signal stays at 0.



(b) Timing diagram

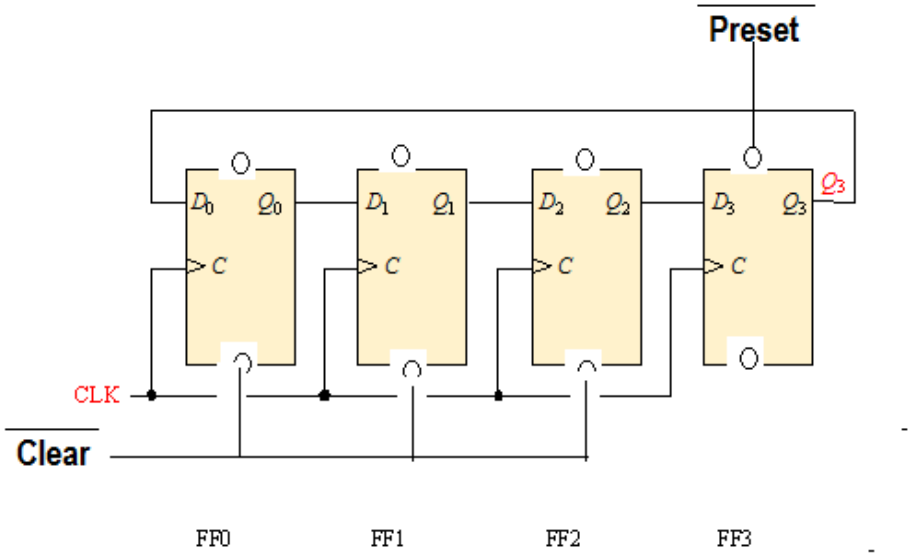
Timing Signals

- The control unit in a digital system that operates in the parallel mode must generate timing signals that stay on for only one clock pulse period.
- Timing signals that control the sequence of operations in a digital system can be generated with a shift register or a counter with a decoder.
- A ring counter is a circular shift register with only one flip-flop being set at any particular time, all others are cleared.
- The single bit is shifted from one flip-flop to the other to produce the sequence of timing signals.



Ring Counter

- The ring counter can also be implemented with either D flip-flops or J-K flip-flops.
- 4-bit ring counters are constructed from a series of D flip-flops J-K flip-flops. Notice the feed



Timing Signals

- To generate 2^n timing signals,
  - we need a shift register with 2^n flip-flops
- or, we can construct the ring counter with a binary counter and a decoder

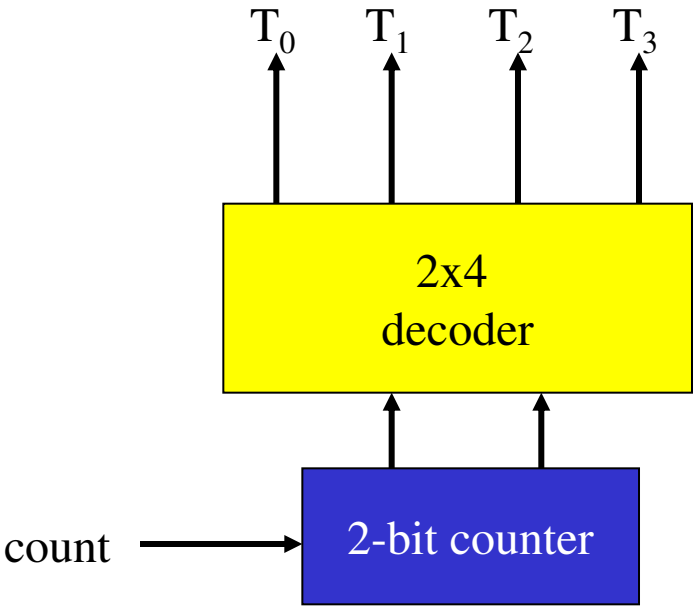


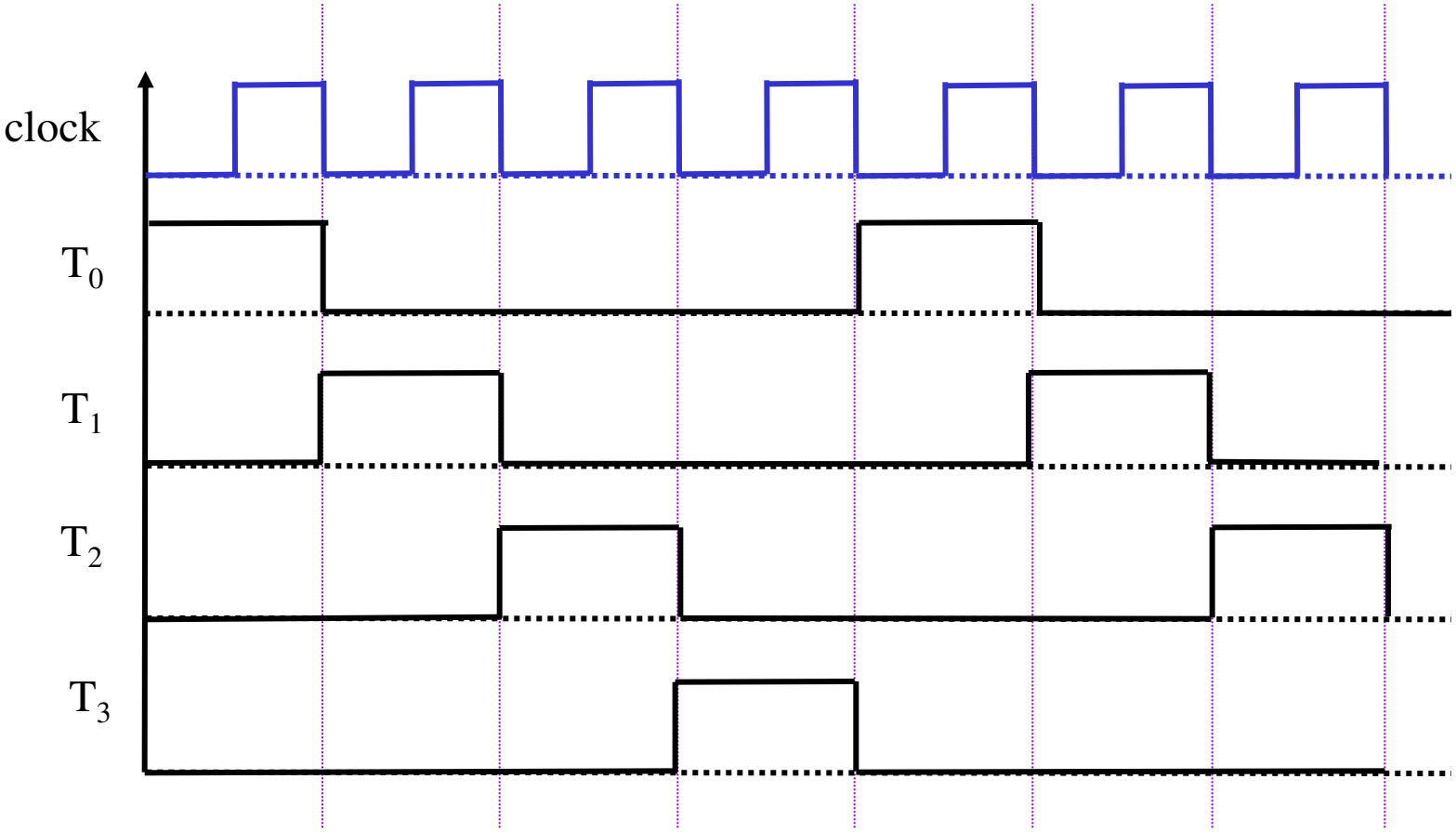
Fig counter and decoder

Cost:
- 2 flip-flops
- 2-to-4 line decoder

Cost in general case:
- n flip-flops
- n-to-2^n line decoder
  - 2^n n-input AND gates
  - n NOT gates

Timing Signals

- Sequence of timing signals



Johnson Counter

- The Johnson counter, also known as the twisted-ring counter, is exactly the same as the ring counter except that the inverted output of the last flip-flop is connected to the input of the first flip-flop.
- The number of states can be doubled if the shift register is connected as a switch-tail ring counter
- A k-bit ring counter can generate k distinguishable states
- Total number of used states= 2*n
- Total number of unused states= $2^n - 2*n$

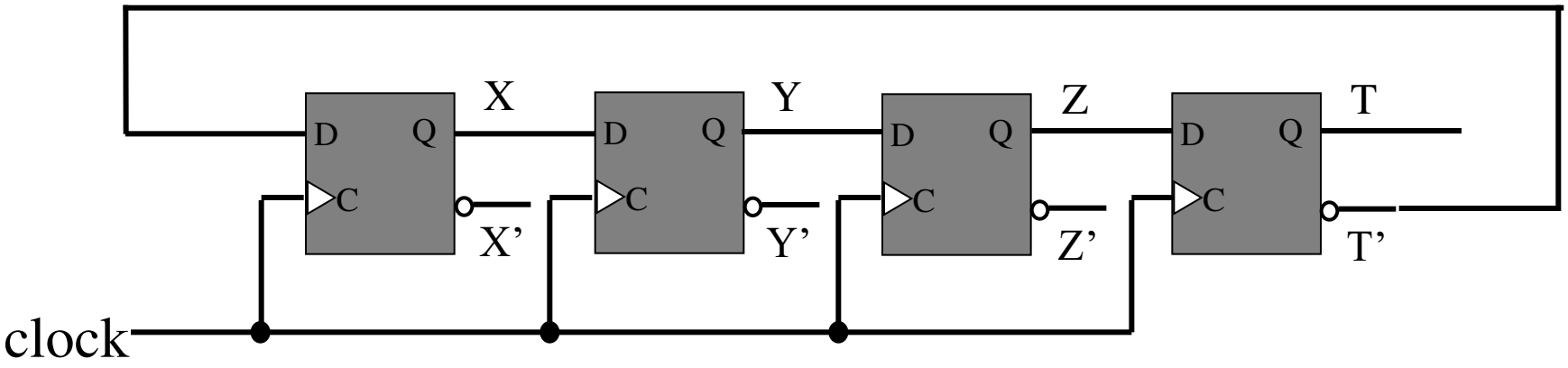


Fig four stage johnson counter

Johnson Counter

- Count sequence and required decoding

sequence number	Flip-flop outputs				Output
	X	Y	Z	T	
1	0	0	0	0	$S_0 = X'T'$
2	1	0	0	0	$S_1 = XY'$
3	1	1	0	0	$S_2 = YZ'$
4	1	1	1	0	$S_3 = ZT'$
5	1	1	1	1	$S_4 = XT$
6	0	1	1	1	$S_5 = X'Y$
7	0	0	1	1	$S_6 = Y'Z$
8	0	0	0	1	$S_7 = Z'T$

Johnson Counter

- Question: Determine the total number of used and unused states in 4-bit Johnson counter.
- Answer: Total number of used states= 2*n
  = 2*4
  = 8
  Total number of unused states= 2^n – 2*n
  = 2^4-2*4
  = 8

Johnson Counter

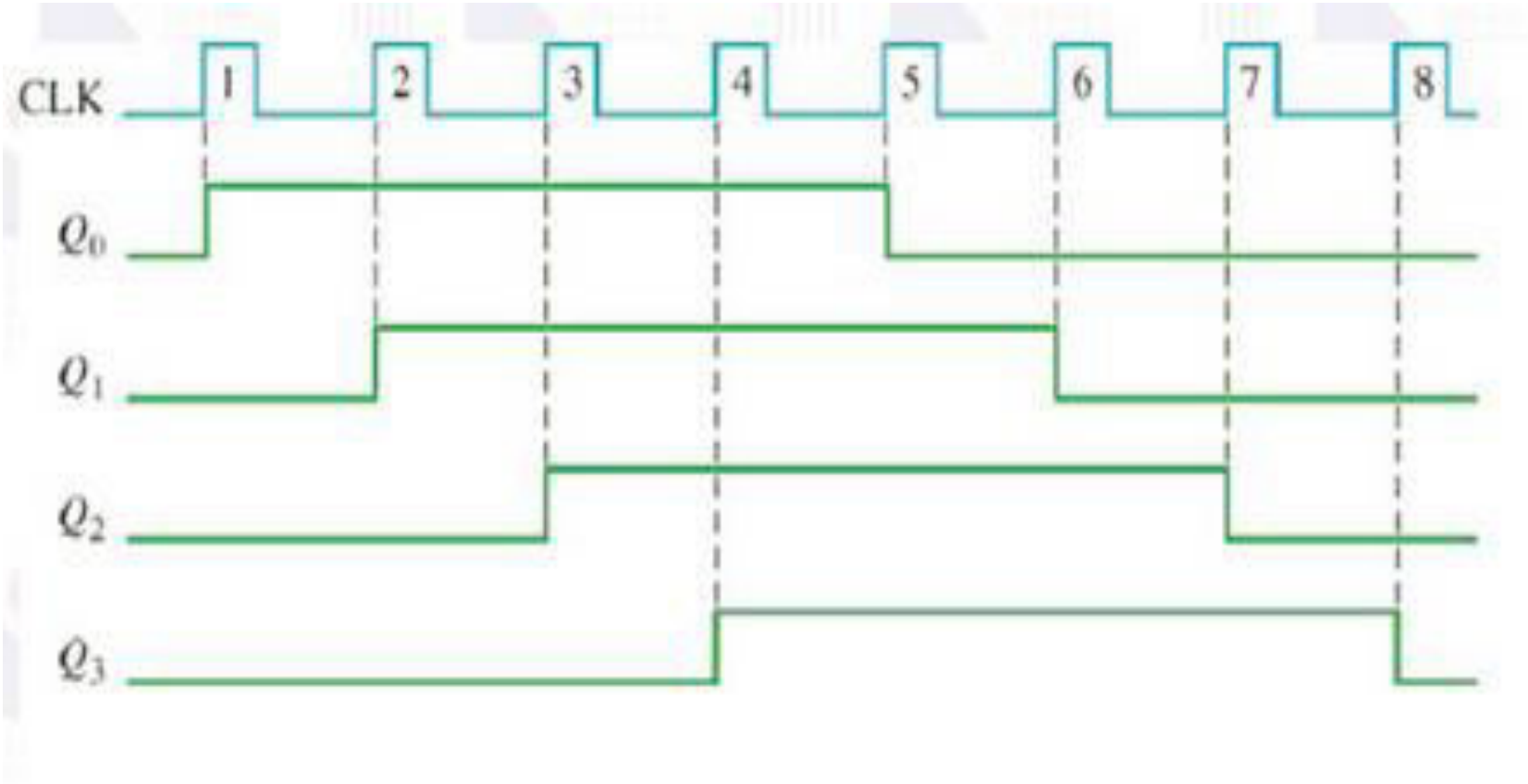
Advantages of Johnson counter:

- The Johnson counter has same number of flip flop but it can count twice the number of states the ring counter can count.
- It can be implemented using D and JK flip flop.
- Johnson ring counter is used to count the data in a continuous loop.
- Johnson counter is a self-decoding circuit.

Disadvantages of Johnson counter:

- Johnson counter doesn't count in a binary sequence.
- In Johnson counter more number of states remain unutilized than the number of states being utilized.
- The number of flip flops needed is one half the number of timing signals.
- It can be constructed for any number of timing sequence.

Johnson Counter



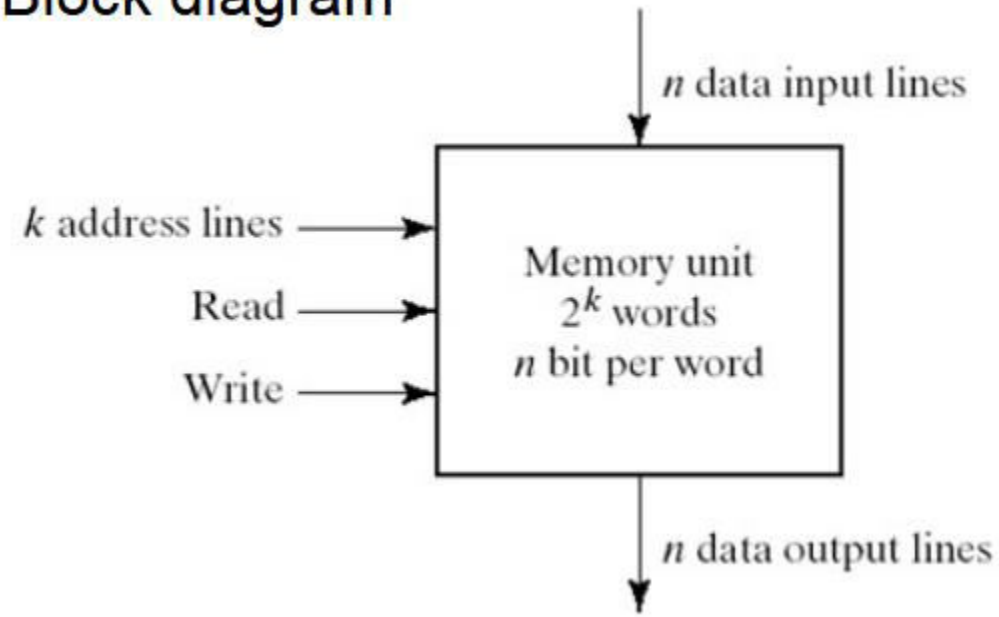
Timing signal

Memory unit

- A memory unit is a collection of storage cells together with associated circuits needed to transfer information in and out of the device.
- Memory cells can be accessed for information transfer to or from any desired random location and hence the name *random access memory,* abbreviated RAM.
- A memory unit stores binary information in groups of bits called ***words***.
- A word in memory is an entity of bits that move in and out of storage as a unit.
- A memory word is a group of 1's and 0's and may represent a number, an instruction, one or more alphanumeric characters, or any other binary-coded information.

Memory unit

- Block diagram



Memory unit

Read and write operation

- Write operation
  - Apply the binary address to the address lines
  - Apply the data bits to the data input lines
  - Activate the *write* input
- Read operation
  - Apply the binary address to the address lines
  - Activate the *read* input

Control Inputs to Memory Chip

Memory Enable	Read/Write	Memory Operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

Integrated Circuit memory (Binary Cell- BC)

The internal construction of a random-access memory of *m* words with *n* bits per word consists of *m* x *n* binary storage cells and associated decoding circuits for selecting individual words. The binary storage cell is the basic building block of a memory unit

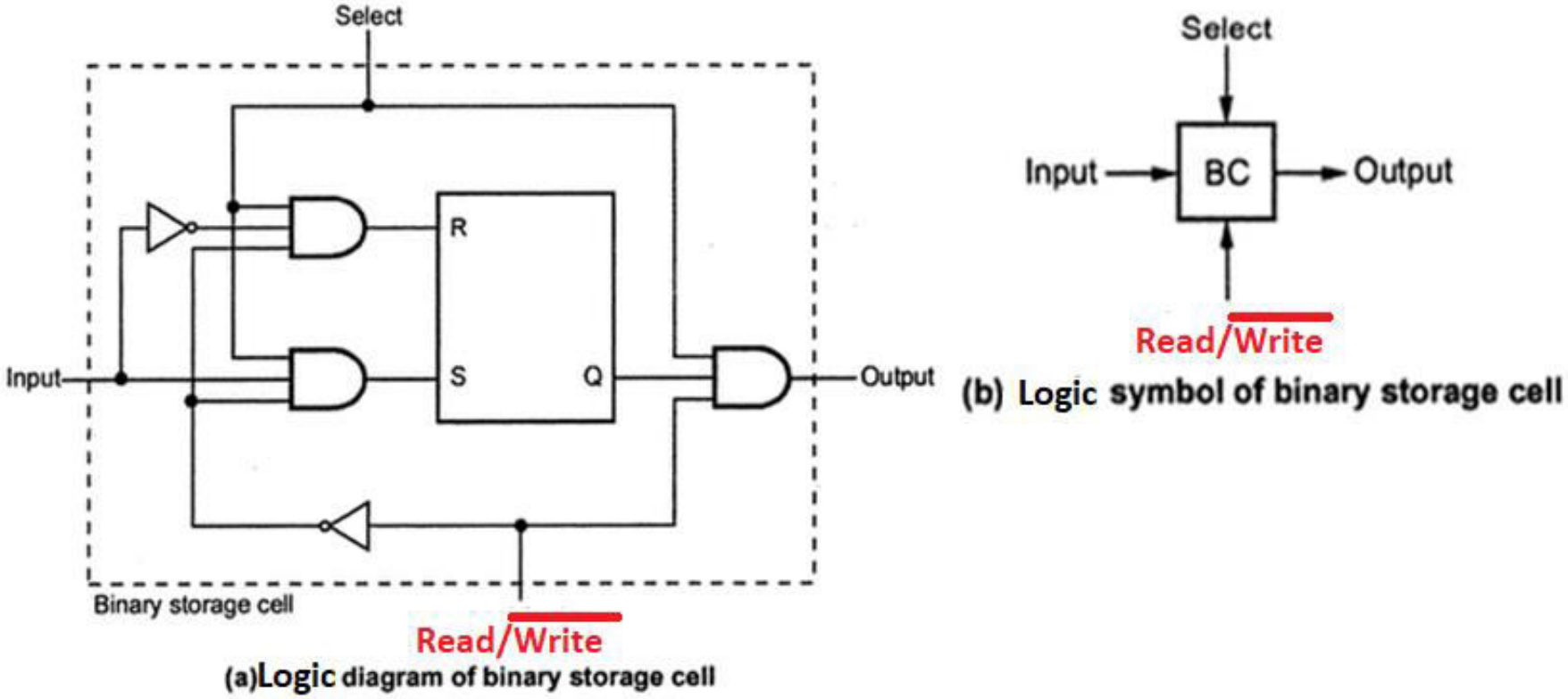


Fig: Memory Cell

Integrated Circuit memory (Binary Cell- BC)

- The equivalent logic of a binary cell that stores one bit of information is shown below.
  - ✓ Read/Write = 0, select = 1, input data to S-R latch
  - ✓ Read/Write = 1, select = 1, output data from S-R latch

Assignment:

- Explain about Shift Registers?
- Design and draw the 3 bit up-down synchronous counter?
- Explain about ripple counter?
- Using D flip flops, design a synchronous counter which counts in the sequence, 000,001,010,011,100,101,110,111,000
- Design a synchronous counter with the following sequence: 0,1,3,7,6,4 and repeats.
- Design a synchronous counter that counts the sequence 000,001,010,011,100,101,110,111,000 Using D flip flop

Assignment:

- Consider the design of 4-bit BCD counter that counts in the following way:

  0000,0010,0011,….,1001 and back to 0000 (i) Draw the state diagram (ii) List the next state table (iii)Draw the logic diagram of the circuit

- Design three bit synchronous counter with T flip flop and draw the diagram.
- Design a binary counter using T flip flops to count in the following sequences: (i) 000, 001, 010, 011, 100, 101, 111, 000 (ii) 000, 100, 111, 010, 011, 000
- Design a modulo 5 synchronous counter using JK Flip Flop and implement it. Construct its timing diagram.

Assignment:

- How many states are there in 3-bit ring counter? What are they?
- Explain the design of a 4 bit binary counter with parallel load in detail?
- Define shift registers?
- Write the differences between synchronous and asynchronous counters?
- Design and draw the 3 bit up-down synchronous counter?

THANKYOU