

Computer Organization and Architecture

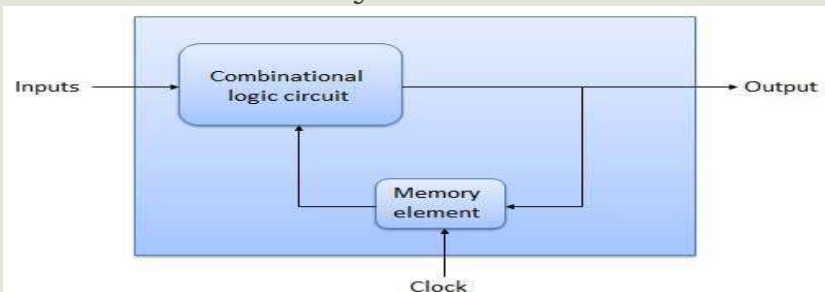
CHAPTER 5

Sequential Circuits

Developed By:
Dr. Vivek Vyas
Department of MCA
DDU

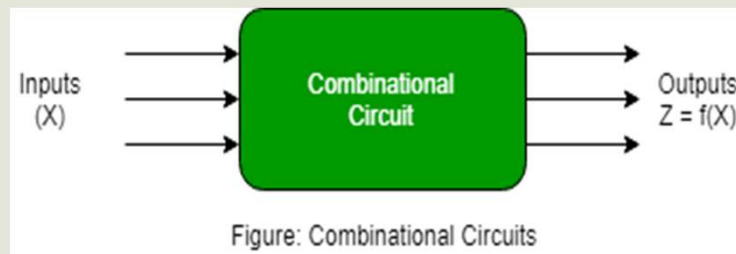
□ Introduction

- The combinational circuit does not use any memory. Hence the previous state of input does not have any effect on the present state of the circuit. But sequential circuit has memory so output can vary based on input. This type of circuits uses previous input, output, clock and a memory element.



□ Introduction

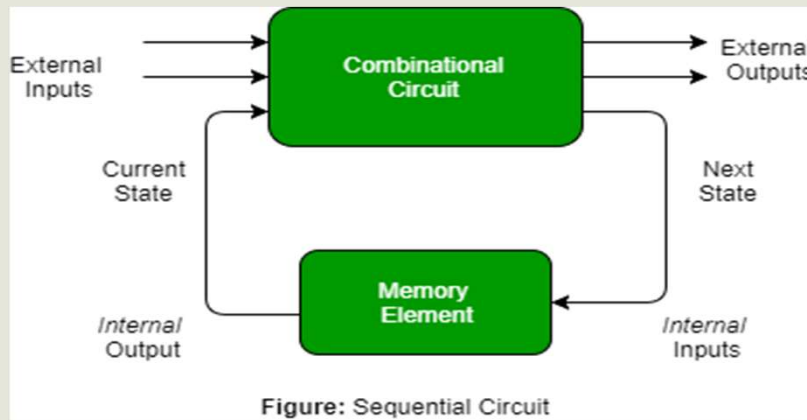
- A **Sequential circuit** combinational logic circuit that consists of inputs variable (X), logic gates (Computational circuit), and output variable (Z).



□ Introduction

- Combinational circuit produces an output based on input variable only, but **Sequential circuit** produces an output based on **current input and previous input variables**.
- That means sequential circuits include memory elements which are capable of storing binary information. That binary information defines the state of the sequential circuit at that time. A latch capable of storing one bit of information.

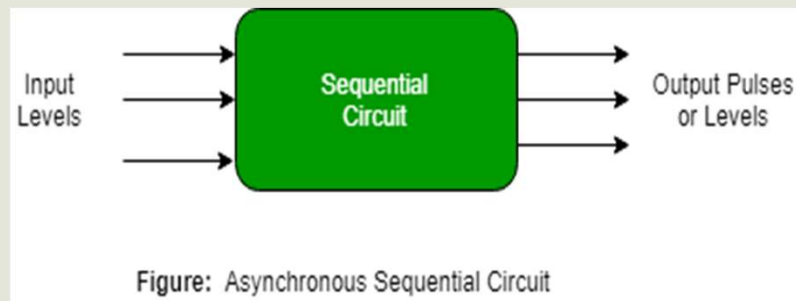
□ Introduction



□ Introduction

- **Types of Sequential Circuits** – There are two types of sequential circuit :
- **Asynchronous sequential circuit** – These circuit **do not use a clock signal** but uses the pulses of the inputs. These circuits are **faster** than synchronous sequential circuits because there is clock pulse and change their state immediately when there is a change in the input signal.
- We use asynchronous sequential circuits when speed of operation is important and **independent** of internal clock pulse. But these circuits are more **difficult** to design and their output is **uncertain**.

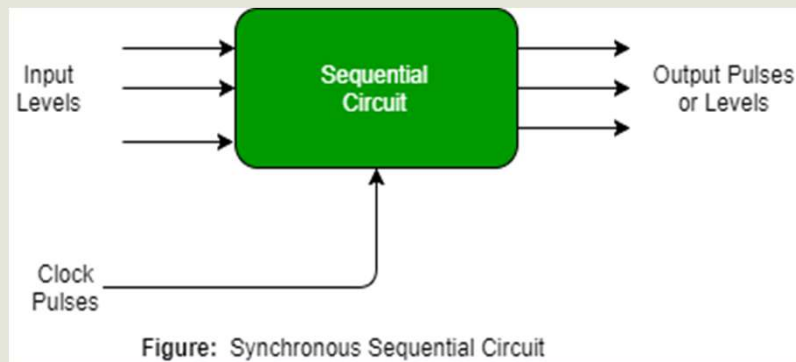
□ Introduction



□ Introduction

- **Synchronous sequential circuit** – These circuit **uses clock signal** and level inputs (or pulsed). The output pulse is the same duration as the clock pulse for the clocked sequential circuits.
- Since they wait for the next clock pulse to arrive to perform the next operation, so these circuits are bit **slower** compared to asynchronous.
- We use sequential circuits to design Counters, Registers, RAM.

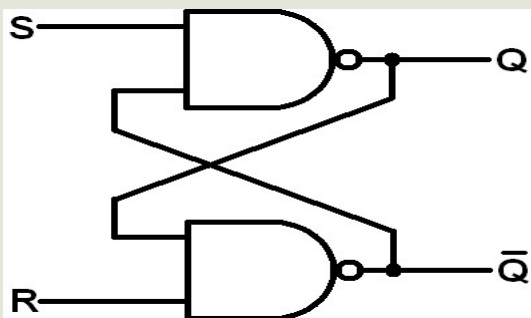
□ Introduction

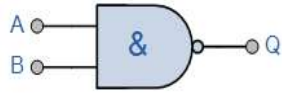


1. Latch

- The basic storage element is called latch. As the name suggest it latches 0 or 1.

- SR NAND Latch



Symbol	Truth Table		
 2-input NAND Gate	B	A	Q
	0	0	1
	0	1	1
	1	0	1
	1	1	0
Boolean Expression $Q = \overline{A \cdot B}$		Read as A AND B gives NOT Q	

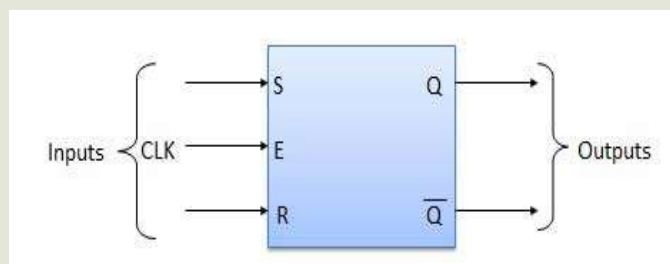
1. Latch

▪ SR NAND Latch Truth Table

S	R	Q	\bar{Q}	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	Undefined

2. S-R Flip Flop

- It is basically S-R latch using NAND gates with an additional **enable** input. It is also called as level triggered SR-FF. For this, circuit in output will take place if and only if the enable input (E) is made active. In short this circuit will operate as an S-R latch if $E = 1$ but there is no change in the output if $E = 0$.



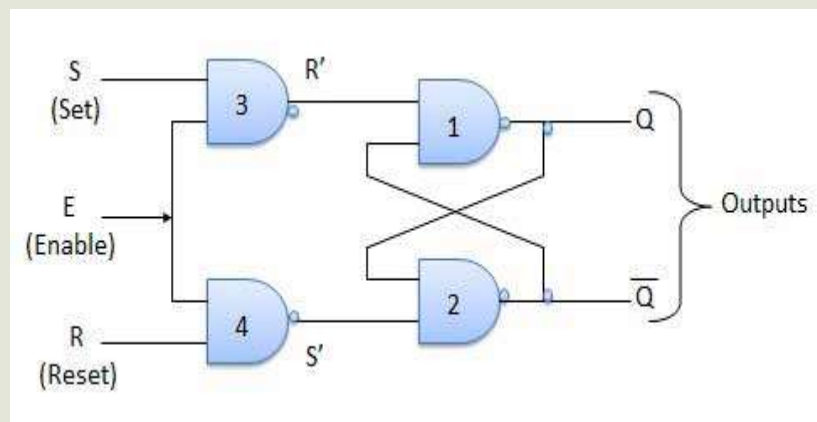
2. S-R Flip Flop

▪ Truth Table

Inputs			Outputs		Comments
E	S	R	Q_{n+1}	\overline{Q}_{n+1}	
1	0	0	Q_n	\overline{Q}_n	No change
1	0	1	0	1	Rset
1	1	0	1	0	Set
1	1	1	x	x	Indeterminate

2. S-R Flip Flop

▪ Circuit Diagram



2. S-R Flip Flop

▪ Operation

▪ 1. **S = R = 0 : No change**

- If $S = R = 0$ then output of NAND gates 3 and 4 are forced to become 1.
- Hence R' and S' both will be equal to 1. Since S' and R' are the input of the basic S-R latch using NAND gates, there will be no change in the state of outputs.

▪ 2. **S = 0, R = 1, E = 1**

- Since $S = 0$, output of NAND-3 i.e. $R' = 1$ and $E = 1$ the output of NAND-4 i.e. $S' = 0$.
- Hence $Q_{n+1} = 0$ and $Q_{n+1} \text{ bar} = 1$. This is reset condition.

2. S-R Flip Flop

▪ 3. **S = 1, R = 0, E = 1**

- Output of NAND-3 i.e. $R' = 0$ and output of NAND-4 i.e. $S' = 1$.
- Hence output of S-R NAND latch is $Q_{n+1} = 1$ and $Q_{n+1} \text{ bar} = 0$. This is the reset condition.

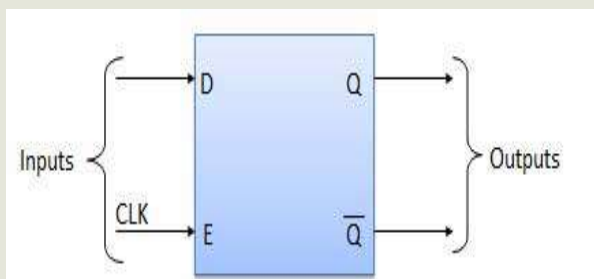
▪ 4. **S = 1, R = 1, E = 1**

- As $S = 1$, $R = 1$ and $E = 1$, the output of NAND gates 3 and 4 both are 0 i.e. $S' = R' = 0$.
- Hence the **Race** condition will occur in the basic NAND latch.

3. D Flip Flop

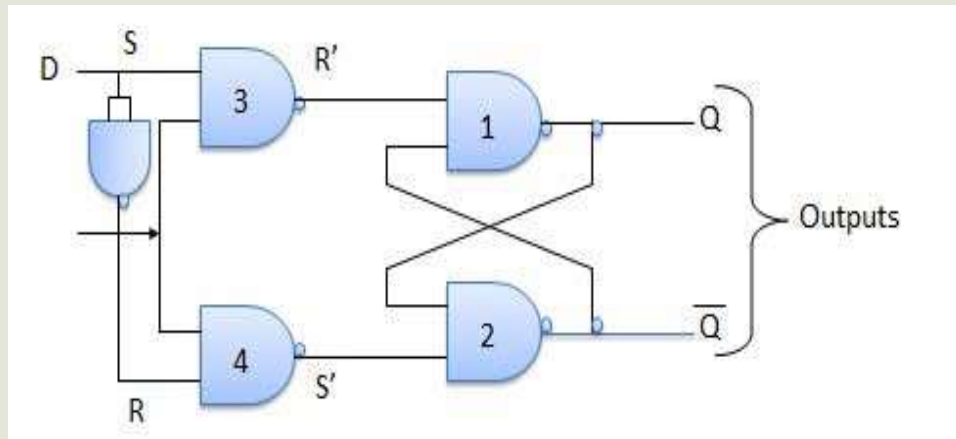
- Data Flip Flop or D Flip Flop is the simple gated S-R latch with a NAND inverter connected between S and R inputs.
- Hence $S = R = 0$ or $S = R = 1$, these input condition will never appear. This problem is avoid by $SR = 00$ and $SR = 11$ conditions.

3. D Flip Flop



Inputs		Outputs		Comments
E	D	Q_{n+1}	\overline{Q}_{n+1}	
1	0	0	1	Rset
1	1	1	0	Set

3. D Flip Flop



3. D Flip Flop

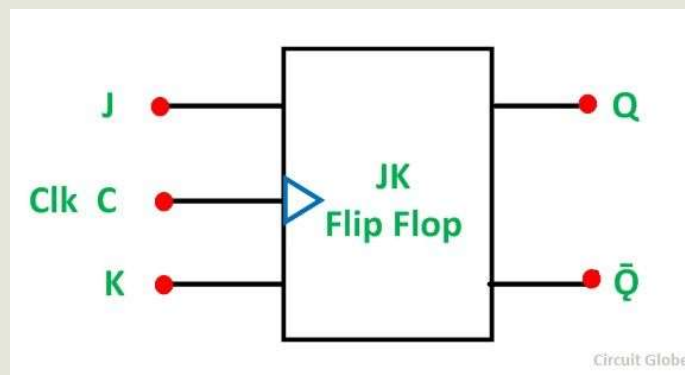
▪ Operation

- **1. $E = 0$:** Latch is disabled. Hence no change in output.
- **2 $E = 1$ and $D = 0$:** If $E = 1$ and $D = 0$ then $S = 0$ and $R = 1$. Hence irrespective of the present state, the next state is $Q_{n+1} = 0$ and $Q_{n+1} \text{ bar} = 1$. This is the reset condition.
- **3 $E = 1$ and $D = 1$:** If $E = 1$ and $D = 1$, then $S = 1$ and $R = 0$. This will set the latch and $Q_{n+1} = 1$ and $Q_{n+1} \text{ bar} = 0$ irrespective of the present state.

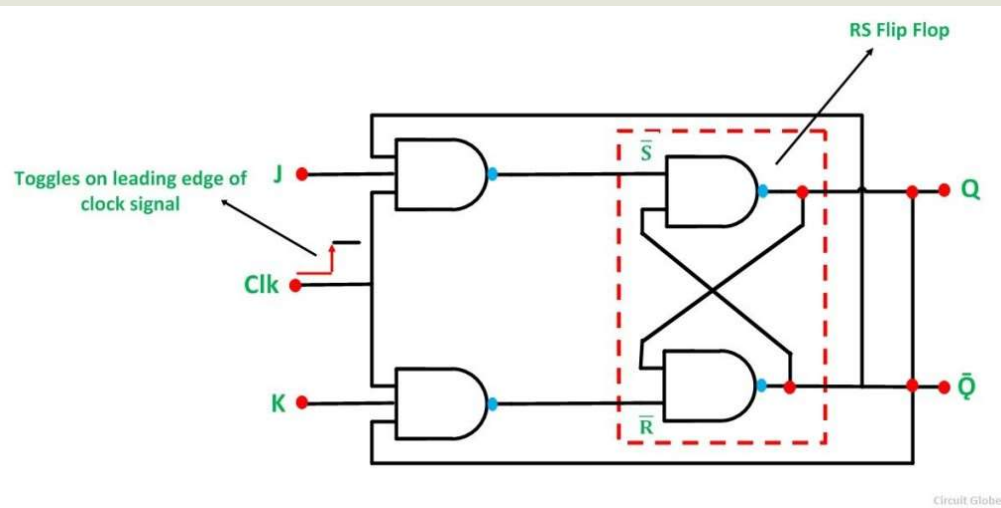
4. J K Flip Flop

- The **JK Flip Flop** is the most widely used flip flop. It is considered to be a universal flip-flop circuit.
- The sequential operation of the JK Flip Flop is same as for the SR flip-flop with the same **SET** and **RESET** input.
- The difference is that the JK Flip Flop does not have the invalid input states of the SR Latch (when S and R are both 1).
- The JK Flip Flop name has been kept on the inventor name of the circuit known as **Jack Kilby**. The basic **symbol** of the JK Flip Flop is shown below.

4. J K Flip Flop



4. J K Flip Flop



4. J K Flip Flop

Truth Table

J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	\bar{Q}_0 (toggles)

5. Master Slave J K Flip Flop

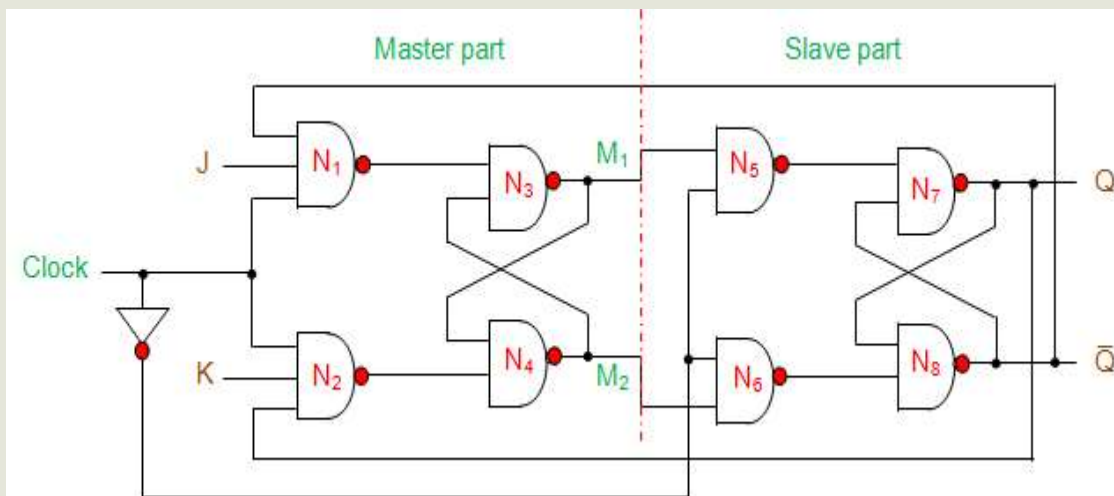
▪ Race Around Condition In JK Flip-flop

- For J-K flip-flop, if $J=K=1$, and if $\text{clk}=1$ for a long period of time, then Q output will toggle as long as CLK is high, which makes the output of the flip-flop unstable or uncertain. This problem is called race around condition in J-K flip-flop.
- This problem (Race Around Condition) can be avoided by ensuring that the clock input is at logic “1” only for a very short time. This introduced the concept of **Master Slave JK** flip flop.

5. Master Slave J K Flip Flop

- The Master-Slave Flip-Flop is basically a combination of two JK flip-flops connected together in a series configuration.
- Out of these, one acts as the “**master**” and the other as a “**slave**”. The output from the master flip flop is connected to the two inputs of the slave flip flop whose output is fed back to inputs of the master flip flop.
- In addition to these two flip-flops, the circuit also includes an **inverter**. The inverter is connected to clock pulse in such a way that the inverted clock pulse is given to the slave flip-flop.
- In other words if $\text{CP}=0$ for a master flip-flop, then $\text{CP}=1$ for a slave flip-flop and if $\text{CP}=1$ for master flip flop then it becomes 0 for slave flip flop.

5. Master Slave J K Flip Flop



5. Master Slave J K Flip Flop

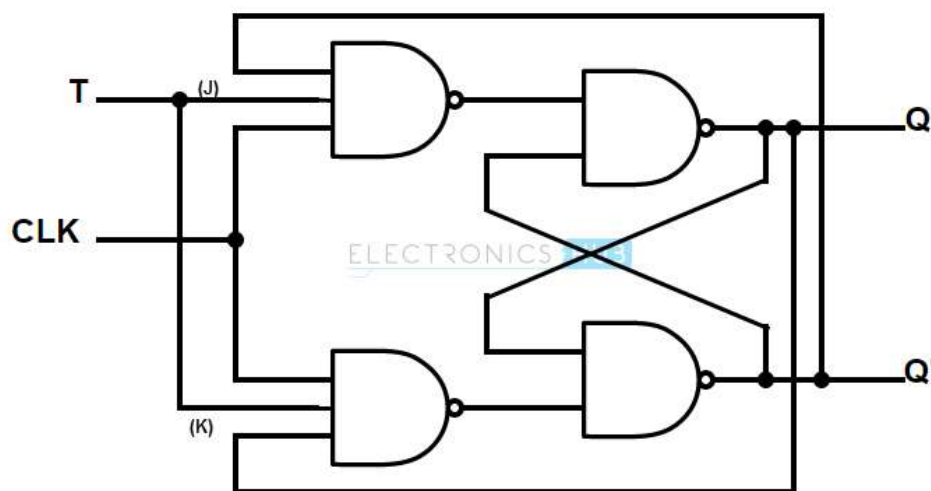
Truth Table

Trigger	Inputs		Output						Inference
			Present State		Intermediate		Next State		
CLK	J	K	Q	\bar{Q}	M ₁	M ₂	Q	\bar{Q}	
↑	0	0	0	1	0	1	Latched		No Change
↓			0	1	Latched		0	1	
↑			1	0	1	0	Latched		
↓			1	0	Latched		1	0	
↑	0	1	0	1	0	1	Latched		Reset
↓			0	1	Latched		0	1	
↑			1	0	0	1	Latched		
↓			1	0	Latched		0	1	
↑	1	0	0	1	1	0	Latched		Set
↓			0	1	Latched		1	0	
↑			1	0	1	0	Latched		
↓			1	0	Latched		1	0	
↑	1	1	0	1	1	0	Latched		Toggles
↓			0	1	Latched		1	0	
↑			1	0	0	1	Latched		
↓			1	0	Latched		0	1	

6. T Flip Flop

- The name T flip-flop is termed from the nature of toggling operation. The major applications of T flip-flop are counters and control circuits.
- **T flip flop is modified form of JK flip-flop** making it to operate in toggling region.
- Whenever the **clock signal is LOW, the input is never going to affect the output state**. The clock has to be high for the inputs to get active.

6. T Flip Flop



6. T Flip Flop

▪ Truth Table

	Previous		Next	
T	Q_{Prev}	Q'_{Prev}	Q_{Next}	Q'_{Next}
0	0	1	0	1
0	1	0	1	0
1	0	1	1	0
1	1	0	0	1