Counters & Registers

CSE 211

Lecturer Uzma Hasan

Counters : Introduction

A digital circuit which is used for counting pulses is known as a counter.

Counter is a type of **sequential circuit** and it is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied. As we know, flip-flops have a clock input. Depending on the type of clock input used, counters are of two types:

- Asynchronous or ripple counters
- Synchronous counters

Since counters depend on clocks like all sequential circuits, so to understand their working procedure, we will consider every clock cycle. Meaning, there will be changes in the states of some flip flops at every clock pulse applied.

Counting Limit

Mod n or Modulus of n, is a way of referring to the **maximum count** of a counter. Every counter has a limit with regards to the number they can count up or down to. Mod n expresses that limit.

n = Maximum event count of the counter. This is the number of states that the counter has.

N = Number of flip-flops connected in cascade

$2^N - 1$ = Maximum decimal count it can reach. Because binary numbers start counting from 0, so for a counter that can count up to 4 events, its decimal equivalent will be 3 only (0,1,2,3).

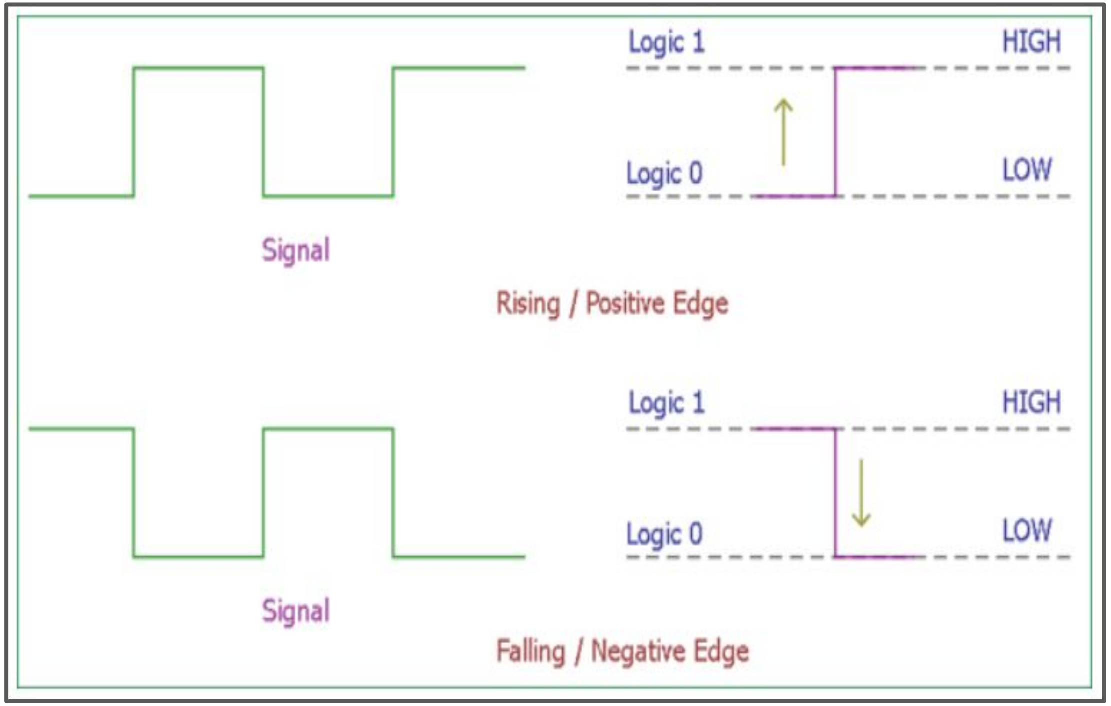
Example: Mod 8 counter

Mod 8 means n = 8. From the equation above $8 = 2^N$

Thus, N = 3.

Which means that this is a counter with three flip-flops, which means three bits, having eight stable states (000 to 111) and capable of counting eight events or up to the decimal number $2^N - 1 = 7$

Trigger Pulse



- There are two type of edge triggered flip-flops available, Positive edge or Negative edge.
- **Positive Edge or Rising Edge flip-flops** count one single step when the clock input changes its state from Logic 0 to Logic 1, i.e. Logic Low to Logic High.
- **Negative Edge or falling Edge flip-flops** count one single step when the clock input changes its state from Logic 1 to Logic 0, i.e. Logic High to Logic Low.

Asynchronous/Ripple Counters

- ➔ In an asynchronous counter, all the clock inputs of the flip-flops have a unique input that is not shared with any other flip-flop in the system.
- ➔ In fact, in an asynchronous counter, only the first flip-flop is given a clock (CLK) input. The output of the first flip-flop is then connected to the clock input of the subsequent flip-flop and so on.
- ➔ Now, think about the output for a second flip-flop. A flip-flop is activated when it receives a clock pulse. So the second flip-flop and all the subsequent flip-flops in an asynchronous counter get active only when their preceding flip-flop gives an output.
- ➔ Thus, the clock passes as a ripple through the cascade of flip-flops. Hence, asynchronous counters are alternatively also known as ***ripple counters.***

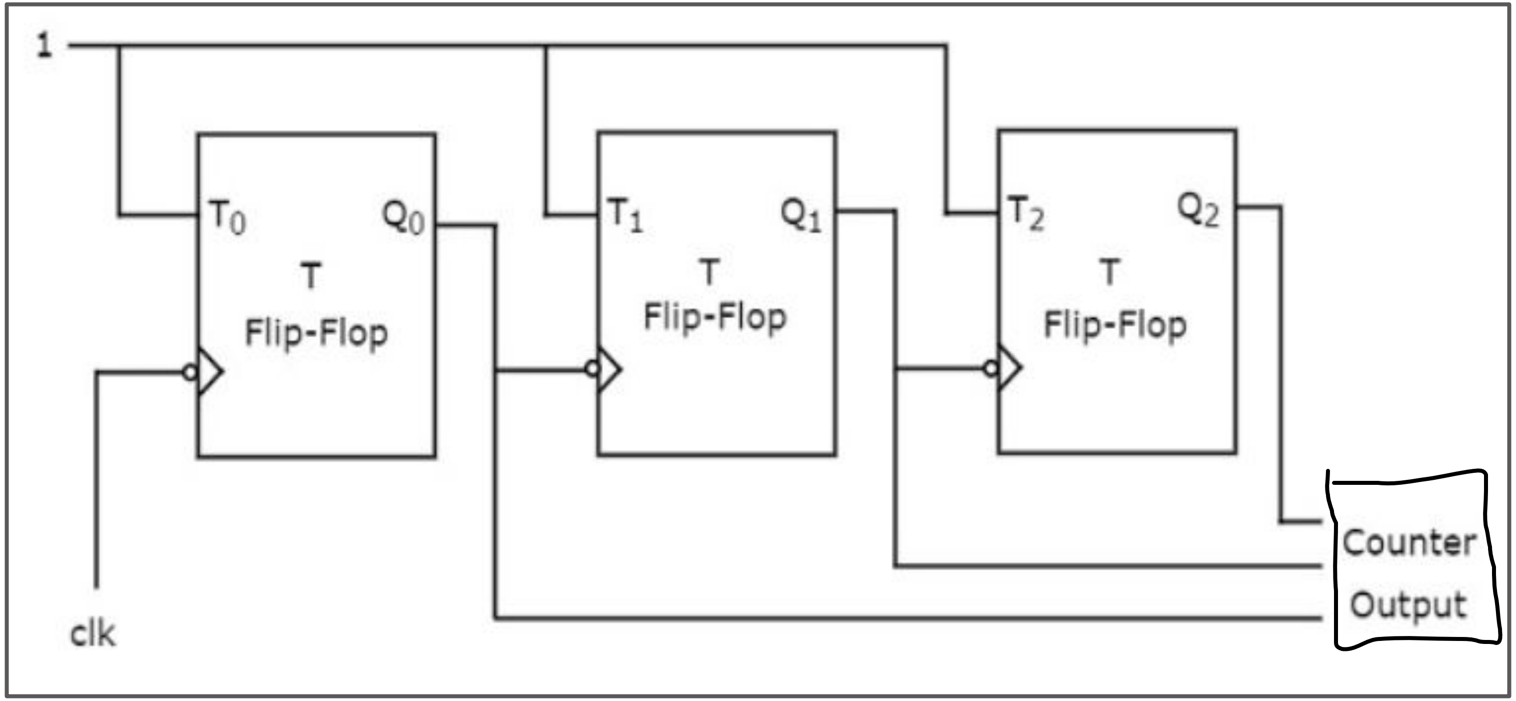
Types of asynchronous counters:

- Asynchronous Binary up counter
- Asynchronous Binary down counter

An ‘N’ bit Asynchronous binary up counter consists of ‘N’ T flip-flops. It counts from 0 to $2^N - 1$. While an ‘N’ bit Asynchronous binary down counter counts from $2^N - 1$ to 0. That is, up counters count upwards or incrementally. Down counters count downwards or in a decremental manner.

Asynchronous/Ripple Up Counter

The following is a 3 bit Asynchronous/Ripple Up counter



3 bit Asynchronous/Ripple Up Counter

An ‘N’ bit Asynchronous binary up counter consists of ‘N’ T flip-flops. It counts from 0 to $2^N - 1$.

A 3-bit Asynchronous binary up counter contains **three T flip-flops** and the **T-input of all the flip-flops are connected to ‘1’**.

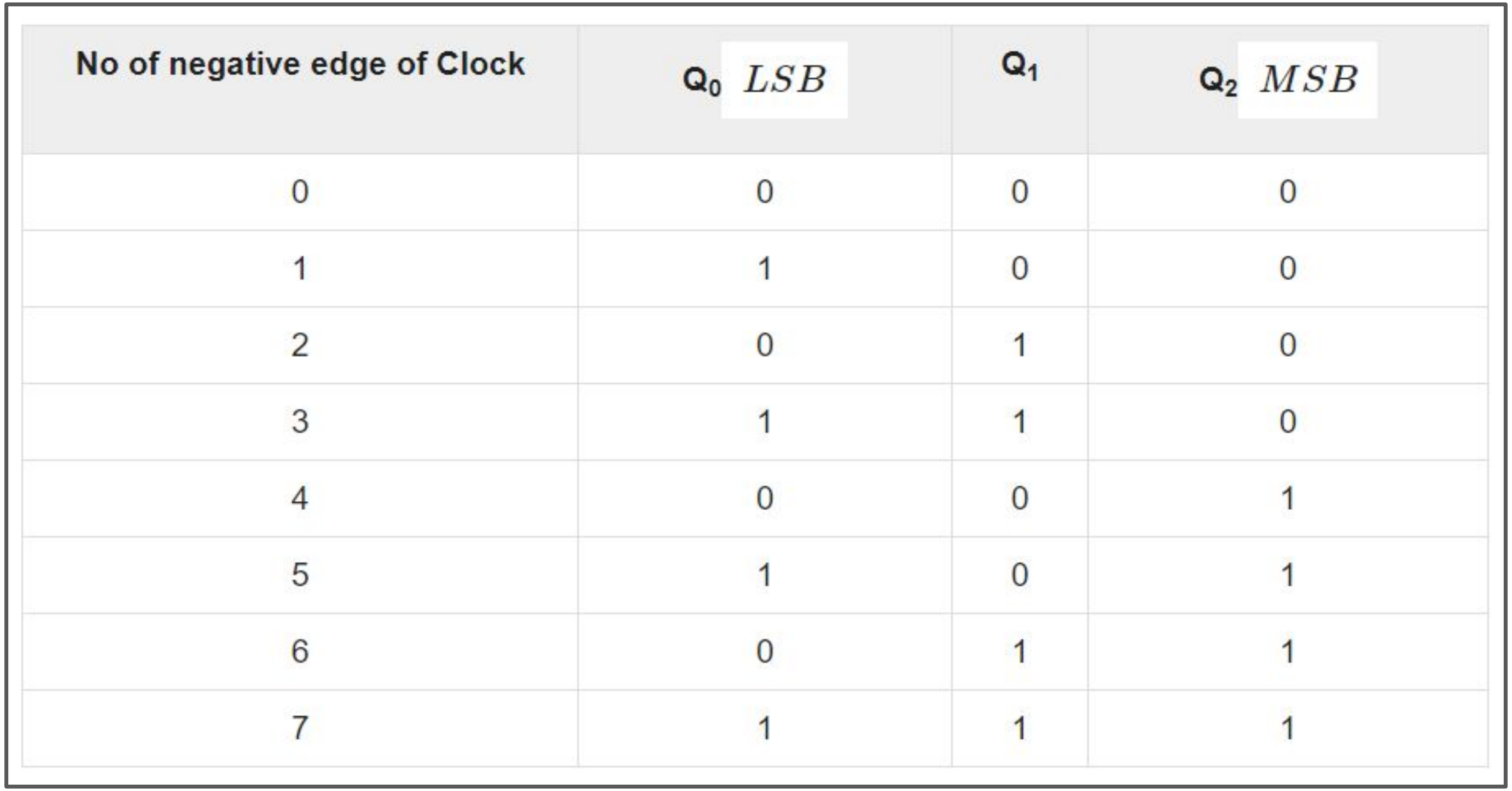
- All these flip-flops are **negative edge triggered** but the outputs change asynchronously (**not simultaneously**). The clock signal is directly applied to the first T flip-flop. So, the output of first T flip-flop **toggles** for every negative edge of clock signal.
- The output of first T flip-flop is applied as clock signal for second T flip-flop. So, the output of second T flip-flop toggles for every negative edge of output of first T flip-flop.
- Similarly, the output of third T flip-flop toggles for every negative edge of output of second T flip-flop, since the output of second T flip-flop acts as the clock signal for third T flip-flop.

3 bit Asynchronous/Ripple Up Counter

- Assume the initial status of T flip-flops from rightmost to leftmost is $Q_2Q_1Q_0$ = 000 . Here, Q_2 & Q_0 are MSB & LSB respectively. We can understand the working of 3-bit asynchronous binary up counter from the table presented in the next slide.
- Here **Q_0 toggled** for every negative edge of clock signal. **Q_1 toggled** for every Q_0 that goes from 1 to 0, otherwise remained in the previous state. Similarly, **Q_2 toggled** for every Q_1 that goes from 1 to 0, otherwise remained in the previous state.
- The **initial status** of the T flip-flops in the absence of clock signal is **$Q_2Q_1Q_0$ = 000** . This is incremented by one for every negative edge of clock signal and reached to maximum value at 7th negative edge of clock signal. This pattern repeats when further negative edges of clock signal are applied.

3 bit Asynchronous Up Counter

Table : 1



No of negative edge of Clock	Q_0 LSB	Q_1	Q_2 MSB
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1
6	0	1	1
7	1	1	1

Synchronous Counters

If **all the flip-flops** have been applied **the same clock signal**, then that counter is called as Synchronous counter. Hence, the **outputs of all flip-flops changes at the same time**.

That is, in a synchronous counter, all the flip-flops are synchronized to the same clock input. It means that for every clock pulse, all the flip-flops will generate an output. Types of synchronous counter are:

- Synchronous Binary up counter
- Synchronous Binary down counter

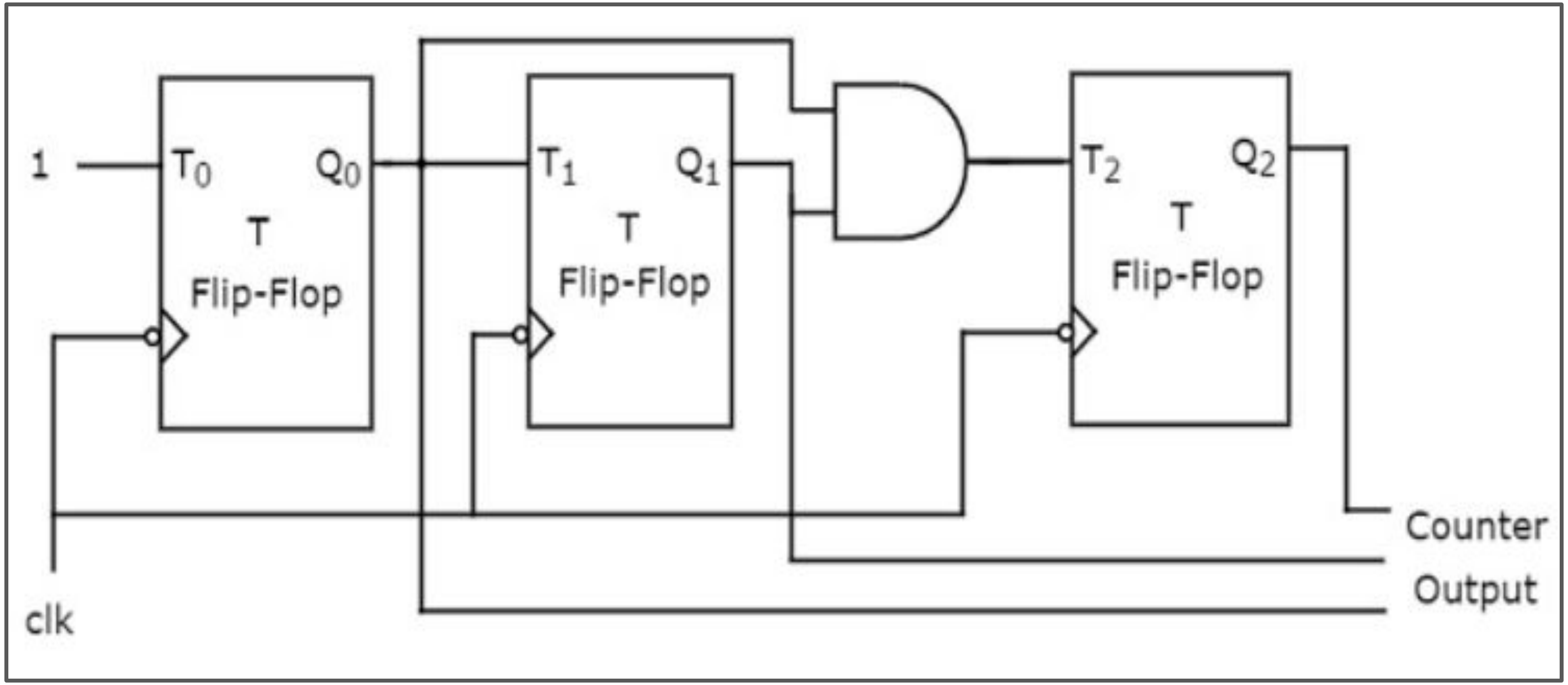
<u>Synchronous Binary Up Counter:</u>

An 'N' bit Synchronous binary up counter consists of 'N' T flip-flops. It counts from 0 to $2^N - 1$. The block diagram of a 3-bit Synchronous binary up counter is shown in the following slide.

- The 3-bit Synchronous binary up counter contains three T flip-flops & one 2-input AND gate.
- All these flip-flops are negative edge triggered and the outputs of the flip-flops change affect synchronously.
- The T inputs of first, second and third flip-flops are 1, Q_o & Q_1Q_o respectively.
- The **output of the first T flip-flop toggles** for every negative edge of clock signal. The **output of second T flip-flop toggles** for every negative edge of clock signal if Q_o is 1. The **output of third T flip-flop toggles** for every negative edge of clock signal if both Q_o & Q_1 are 1.

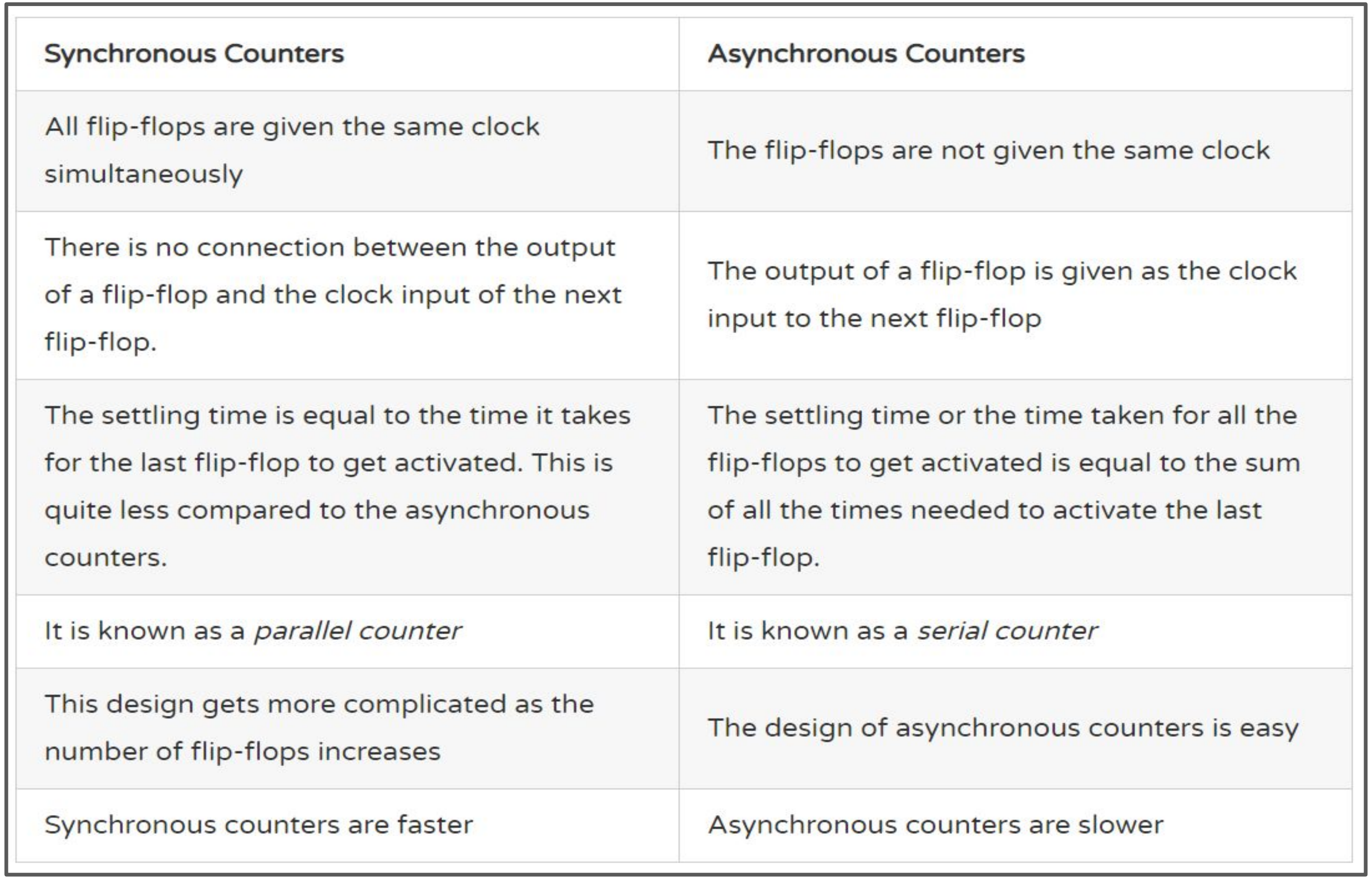
3 bit Synchronous Up Counter

- Initially let all the FFs be in the reset state, i.e. $Q_2Q_1Q_0$=000. The change of states of FFs at every clock pulse is presented in the next slide.



Clock Pulse	Q2	Q1	Q0	Qn2	Qn1	Qn0
1	0	0	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	0	1	1
4	0	1	1	1	0	0
5	1	0	0	1	0	1
6	1	0	1	1	1	0
7	1	1	0	1	1	1
8	1	1	1	0	0	0

Synchronous Vs Asynchronous Counters



Synchronous Counters	Asynchronous Counters
All flip-flops are given the same clock simultaneously	The flip-flops are not given the same clock
There is no connection between the output of a flip-flop and the clock input of the next flip-flop.	The output of a flip-flop is given as the clock input to the next flip-flop
The settling time is equal to the time it takes for the last flip-flop to get activated. This is quite less compared to the asynchronous counters.	The settling time or the time taken for all the flip-flops to get activated is equal to the sum of all the times needed to activate the last flip-flop.
It is known as a *parallel counter*	It is known as a *serial counter*
This design gets more complicated as the number of flip-flops increases	The design of asynchronous counters is easy
Synchronous counters are faster	Asynchronous counters are slower

Registers

- Flip flops can be used to store a single bit of binary data (1 or 0). However, in order to store multiple bits of data, we need multiple flip flops. N flip flops are to be connected in an order to store n bits of data.
- A **Register** is a device which is used to store such information. It is a group of flip flops connected in series used **to store multiple bits of data.**
- Types of registers include memory address register, memory buffer register, input output address register, input output buffer register, and shift register. Here, we will be focusing on shift registers.
- **Shift registers** are digital memory circuitry found in devices such as calculators, computers, and data processing systems. **With the shift register, data or bits are entered into the system in a serial or parallel manner**. They enter from one direction, and as more data is added, shift positions until they get to the output end. The two ends are referred to as the left and right ends. Movement of data can be from left to right, from right to left, or in both directions to make a bidirectional register.
- The bits stored in such registers can be made to move within the registers and in/out of the registers by applying clock pulses. An n-bit shift register can be formed by connecting n flip-flops.
- The registers which will **shift the bits to left are called "Shift left registers"** and the **registers which will shift the bits to right are called "Shift right registers"**.

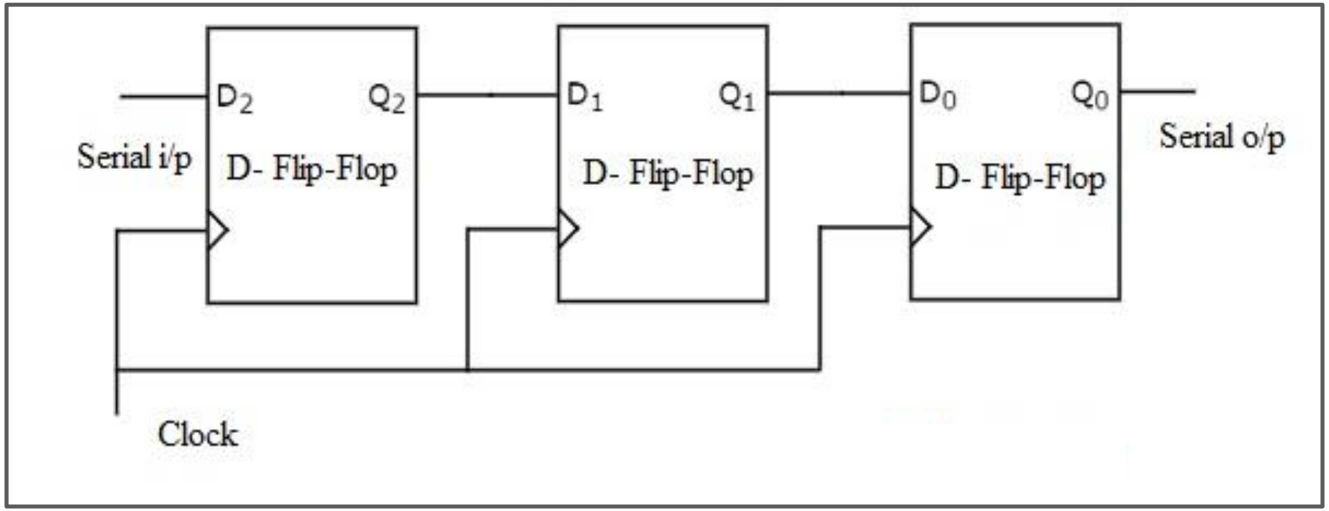
Shift Registers

Shift registers are basically of 4 types. These are:

- **Serial In Serial Out shift register**
- Serial In Parallel Out shift register
- Parallel In Serial Out shift register
- **Parallel In Parallel Out shift register**

- 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.
- Serial-in to Parallel-out (SIPO) - the register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.
- 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 In Serial Out Shift register



- 3 bit SISO Shift Right register
- The above circuit is an example of shift right register, taking the serial data input from the left side of the flip flop. The main use of a SISO is to act as a delay element.

Serial In Serial Out Shift register (memorize)

- The shift register, which **allows serial input** (one bit after the other through a single data line) and **produces a serial output** is known as Serial-In Serial-Out shift register.
- Since **there is only one output**, the **data leaves the shift register one bit at a time in a serial pattern**, thus the name Serial-In Serial-Out Shift Register.
- The block diagram consists of three D flip-flops, which are connected in a serial manner. That means, output of one D flip-flop is connected as the input of next D flip-flop. **All these flip-flops are synchronous with each other** since, the same clock signal is applied to each one.
- In this shift register, **we can send the bits serially from the input of left most D flip-flop.** Hence, this input is also called as serial input. **For every positive edge triggering of clock signal, the data shifts from one stage to the next. So, we can receive the bits serially from the output of right most D flip-flop**. Hence, this output is also called as serial output. That is, at last the data bits stored within the register is obtained at the output pin of the rightmost flip-flop in a serial-fashion (one by one).

Serial In Serial Out Shift register

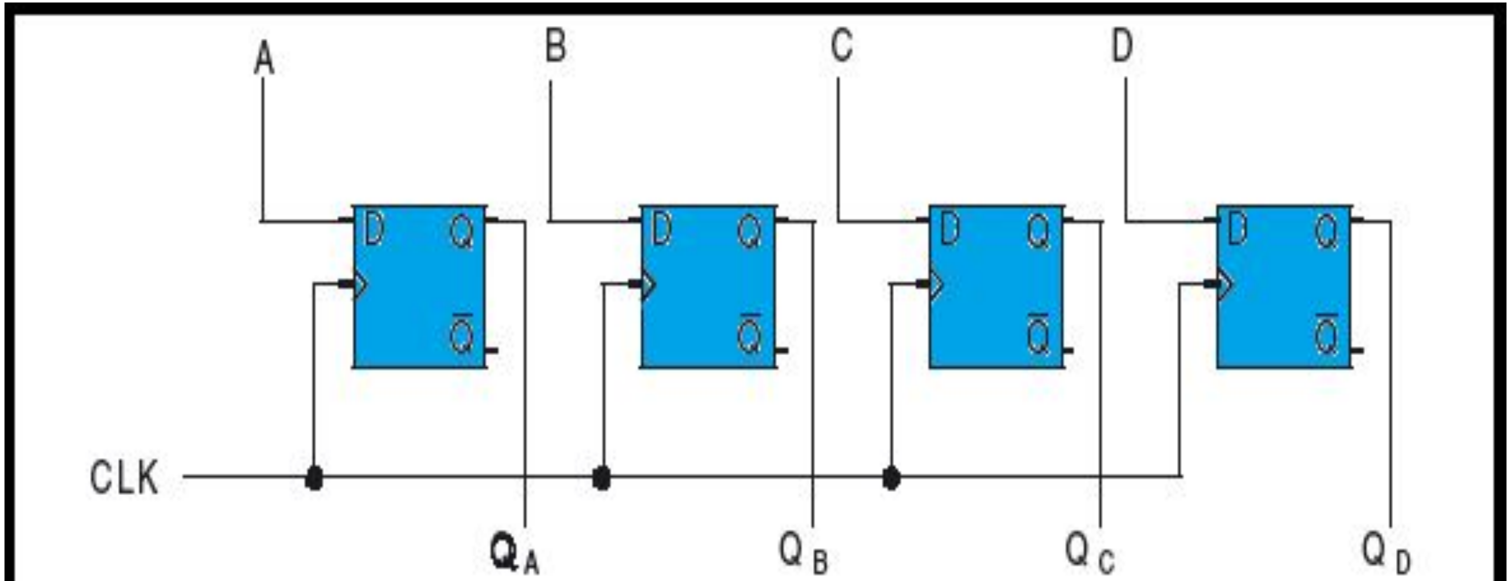
Example: **Store** binary **101** in the register

Clock Pulse	Serial Input	Q2 (MSB)	Q1	Q0 (LSB)
---	---	0	0	0
1st cp	1	1	0	0
2nd cp	0	0	1	0
3rd cp	1	1	0	1

Parallel In Parallel Out Shift register

- The shift register, which **allows parallel input** (data is given separately to each flip flop and in a simultaneous manner) and also **produces a parallel output** is known as Parallel-In parallel-Out shift register.
- The logic circuit given in the next slide shows a parallel-in-parallel-out (PIPO) shift register. The circuit consists of four D flip-flops which are connected. The same clock signal is connected to all the 4 flip flops.
- In this type of register, there are no interconnections between the individual flip-flops since no serial shifting of the data is required. **Data is given as input separately for each flip flop** and in the same way, **output is also collected individually from each flip flop**.
- A Parallel in Parallel out (PIPO) shift register is used as a temporary storage device and like SISO Shift register it acts as a delay element.

Parallel In Parallel Out Shift register



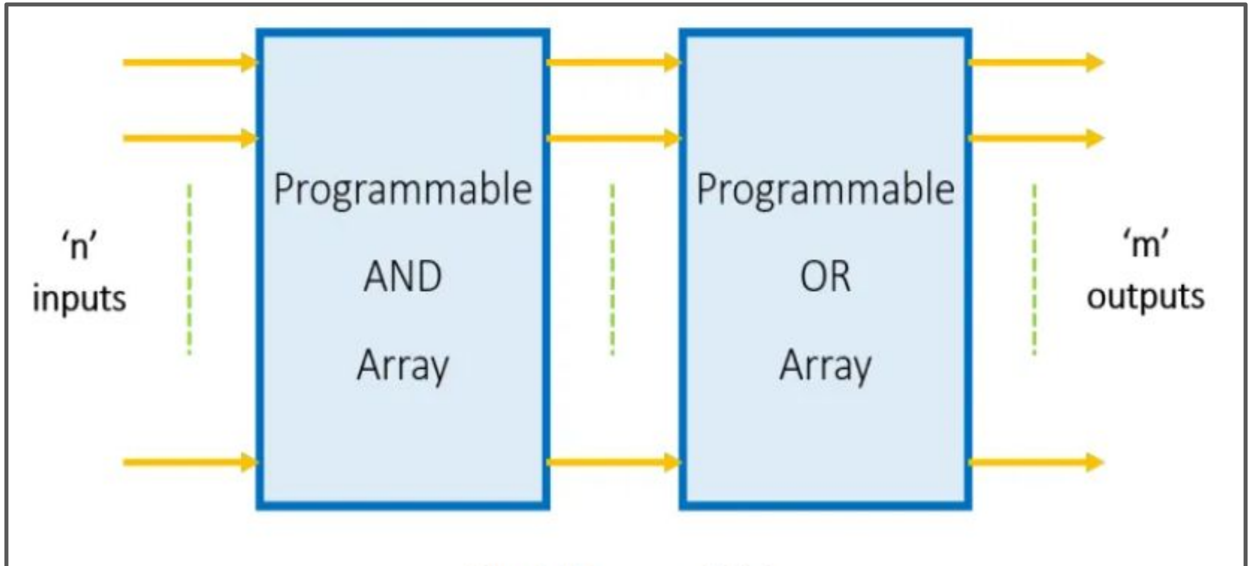
A 4-bit parallel-in–parallel-out shift register.

Programmable Vs Fixed Logic

- Programmable Logic Devices (PLDs) are devices that work on a **programmable logic** – the logic (the way to do something) comes from a program code stored in the device. This program code comprises **instructions** for the device **which can be changed, edited or replaced** on the requirement. PLDs are made using an integrated circuit with a code given to the system.
- On the contrary, a fixed logic system has circuits whose **configurations** are **preset** i.e. permanent. Their instructions **perform only a fixed set of operations** repeatedly. **Once manufactured and programmed, the logic cannot be changed.** This system is a fantastic asset for repeated tasks. But one tiny mistake in the manufacturing process like uploading the wrong code in the device, and the entire system is discarded, and a new design is developed. The **fixed logic system** thus offers **limited usability** while a **programmable logic** seems **more feasible and beneficial**.

Programmable Logic Array

A **programmable logic array** (**PLA**) is a kind of programmable logic device used to implement combinational/sequential logic circuits. It has both **Programmable AND array** & **Programmable OR array**. It is the most flexible PLD. The block diagram of a PLA is shown in the following figure.



Block Diagram of PLA

Programmable Logic Array

- The PLA has a set of programmable AND gate planes, which is linked to a set of programmable OR gate planes, which can then be conditionally complemented to produce an output.
- Here, the **inputs of AND gates** are **programmable**. That means each AND gate has both normal and complemented inputs of variables. So, based on the requirement, we can program any of those inputs. So, we can generate **only the required product terms** by using these AND gates.
- Here, the **inputs of OR gates are also programmable**. So, we can program any number of required product terms, since all the outputs of AND gates are applied as inputs to each OR gate. Therefore, the **outputs of PLA** will be in the form of **sum of product terms**.

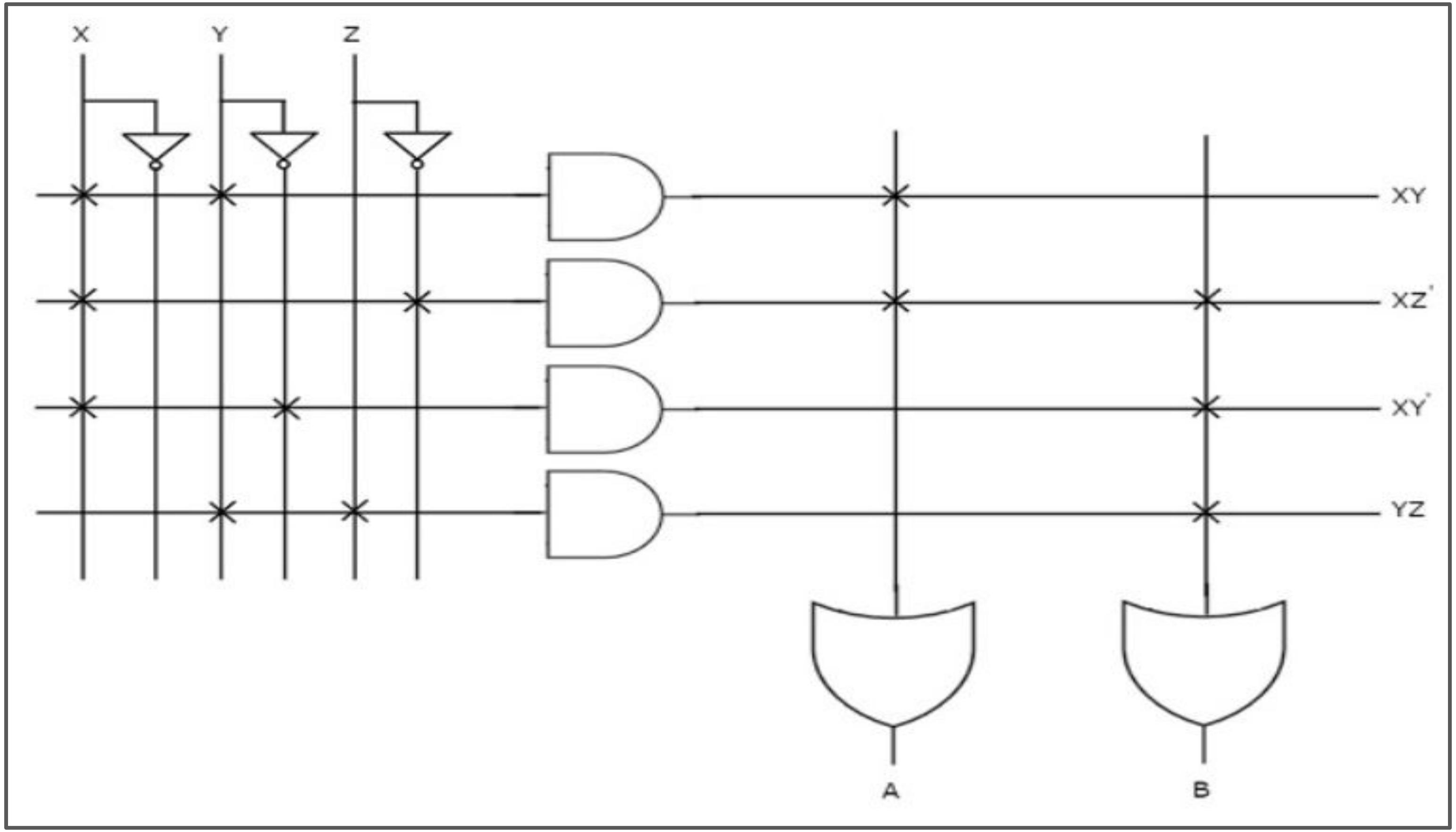
Programmable Logic Array

Example: Let us implement the following Boolean functions using PLA.

$$A = XY + XZ' \qquad B = XY' + YZ + XZ'$$

1. The given two functions **are in sum of products form**. The number of product terms present in the given Boolean functions A & B are two and three respectively. One product term, $Z'X$ is common in each function.
2. So, we require **four programmable AND gates & two programmable OR gates** for producing those two functions. The corresponding PLA is shown in the following slide's figure.
3. The programmable AND gates have the access of both normal and complemented inputs of variables. In the figure, the inputs X, X', Y, Y', Z & Z', are available at the inputs of each AND gate. So, we have to program only the required literals in order to generate one product term by each AND gate.
4. All these product terms are available at the inputs of each programmable OR gate. But, only program the required product terms in order to produce the respective Boolean functions by each OR gate. The **symbol 'X'** is used for **programmable connections**.

Programmable Logic Array


X
Y
Z
XY
XZ'
XY'
YZ
A
B