SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF Computer Science and Engineering

COURSE MATERIAL

Subject Name: Fundamentals of Digital Systems UNIT IV Subject Code: SCSA1201

Flip Flops - Analysis of clocked sequential circuit - C - Flip flop excitation tables - Design Procedure - Design of counters - Registers - Shift registers - Synchronous Counters - Timing sequences- Algorithmic State Machines - ASM chart - timing considerations - control implementation

1. FLIP-FLOP

In electronics, a flip-flop or latch is a circuit that has two stable states and can be used to store state information. Flip-flops and latches are used as data storage elements. A flip-flop stores a single *bit* (binary digit) of data; one of its two states represents a "one" and the other represents a "zero". Such data storage can be used for storage of *state*, and such a circuit is described as sequential logic. When used in a finite-state machine, the output and next state depend not only on its current input, but also on its current state (and hence, previous inputs). It can also be used for counting of pulses, and for synchronizing variably-timed input signals to some reference timing signal.

Flip-flops can be either simple (transparent or opaque) or clocked (synchronous or edge-triggered). Although the term flip-flop has historically referred generically to both simple and clocked circuits, in modern usage it is common to reserve the term *flip-flop* exclusively for discussing clocked circuits; the simple ones are commonly called *latches*.

Using this terminology, a latch is level-sensitive, whereas a flip-flop is edge-sensitive. That is, when a latch is enabled it becomes transparent, while a flip flop's output only changes on a single type (positive going or negative going) of clock edge.

Flip-flop types

Flip-flops can be divided into common types

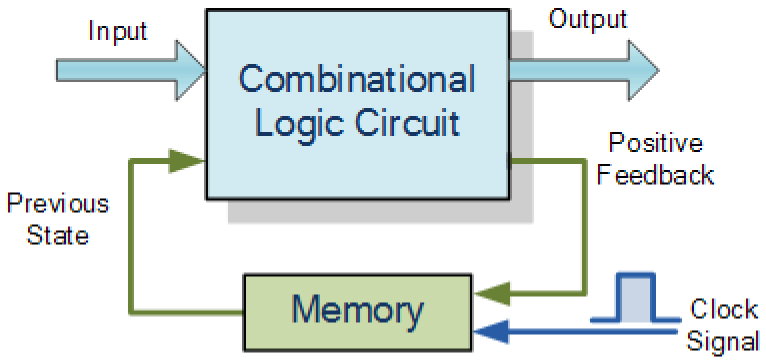
1. **SR** ("set-reset")
2. **D** ("data" or "delay"[12])

3. **T** ("toggle")
4. **JK** types are the common ones.

2. Analysis of clocked sequential circuit

Sequential Logic Circuits

Unlike Combinational Logic circuits that change state depending upon the actual signals being applied to their inputs at that time, Sequential Logic circuits have some form of inherent "Memory" built in to them as they are able to take into account their previous input state as well as those actually present, a sort of "before" and "after" effect is involved with sequential logic circuits.



In other words, the output state of a "sequential logic circuit" is a function of the following three states, the "present input", the "past input" and/or the "past output". *Sequential Logic circuits* remember these conditions and stay fixed in their current state until the next clock signal changes one of the states, giving sequential logic circuits "Memory".

Sequential logic circuits are generally termed as *two state* or Bistable devices which can have their output or outputs set in one of two basic states, a logic level "1" or a logic level "0" and will remain "latched" (hence the name latch) indefinitely in this current state or condition until some other input trigger pulse or signal is applied which will cause the bistable to change its state once again.

The word "Sequential" means that things happen in a "sequence", one after another and in **Sequential Logic** circuits, the actual clock signal determines when things will happen next. Simple sequential logic circuits can be constructed from standard Bistable circuits such as: Flip-flops, Latches and Counters and which themselves can be made by simply connecting together universal NAND Gates and/or NOR Gates in a particular combinational way to produce the required sequential circuit

3. Flip flop excitation tables - Flip flop excitation tables - Design Procedure of SR, Jk, D T flipflops

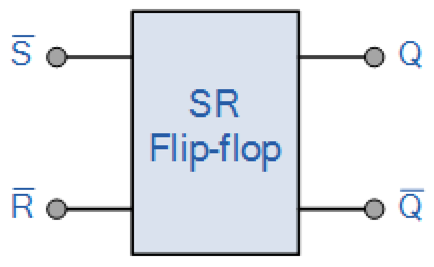
SR Flip-Flop

The **SR flip-flop**, also known as a *SR Latch*, can be considered as one of the most basic sequential logic circuit possible. This simple flip-flop is basically a one-bit memory bistable device that has two inputs, one which will “SET” the device (meaning the output = “1”), and is labelled S and another which will “RESET” the device (meaning the output = “0”), labelled R.

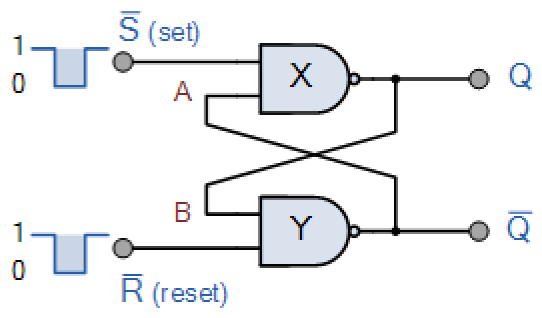
Then the SR description stands for “Set-Reset”. The reset input resets the flip-flop back to its original state with an output Q that will be either at a logic level “1” or logic “0” depending upon this set/reset condition.

A basic NAND gate SR flip-flop circuit provides feedback from both of its outputs back to its opposing inputs and is commonly used in memory circuits to store a single data bit. Then the SR flip-flop actually has three inputs, Set, Reset and its current output Q relating to it’s current state or history. The term “Flip-flop” relates to the actual operation of the device, as it can be “flipped” into one logic Set state or “flopped” back into the opposing logic Reset state.

The Basic SR Flip-flop



Symbol



Circuit

The Set State

Consider the circuit shown above. If the input R is at logic level “0” (R = 0) and input S is at logic level “1” (S = 1), the NAND gate *Y* has at least one of its inputs at logic “0” therefore, its output Q must be at a logic level “1” (NAND Gate principles). Output Q is also fed back to input “A” and so both inputs to NAND gate *X* are at logic level “1”, and therefore its output Q must be at logic level “0”.

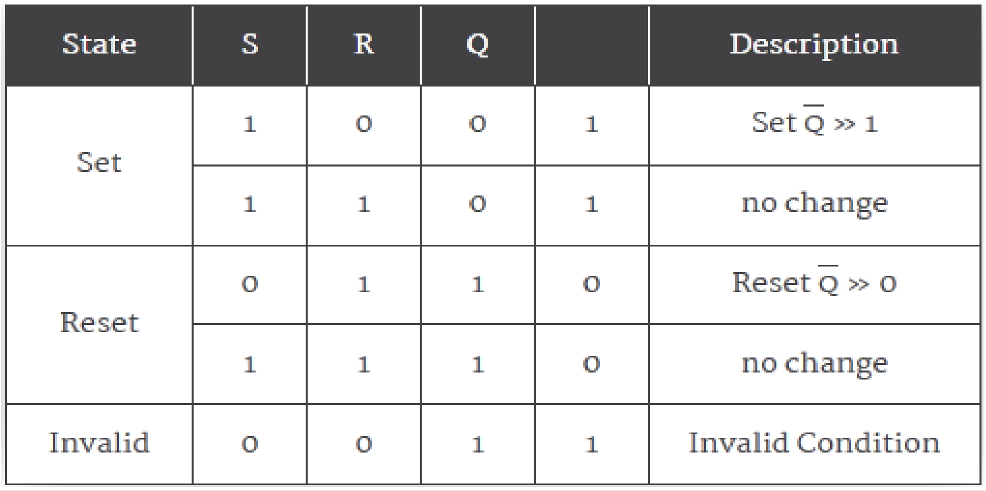
Again NAND gate principals. If the reset input R changes state, and goes HIGH to logic “1” with S remaining HIGH also at logic level “1”, NAND gate *Y* inputs are now R = “1” and B = “0”. Since one of its inputs is still at logic level “0” the output at Q still remains HIGH at logic level “1” and there is no change of state. Therefore, the flip-flop circuit is said to be “Latched” or “Set” with Q = “1” and Q = “0”.

Reset State

In this second stable state, Q is at logic level “0”, (not Q = “0”) its inverse output at Q is at logic level “1”, (Q = “1”), and is given by R = “1” and S = “0”. As gate *X* has one of its inputs at logic “0” its output Q must equal logic level “1” (again NAND gate principles). Output Q is fed back to input “B”, so both inputs to NAND gate *Y* are at logic “1”, therefore, Q = “0”.

If the set input, S now changes state to logic “1” with input R remaining at logic “1”, output Q still remains LOW at logic level “0” and there is no change of state. Therefore, the flip-flop circuits “Reset” state has also been latched and we can define this “set/reset” action in the following truth table.

Truth Table for this Set-Reset Function

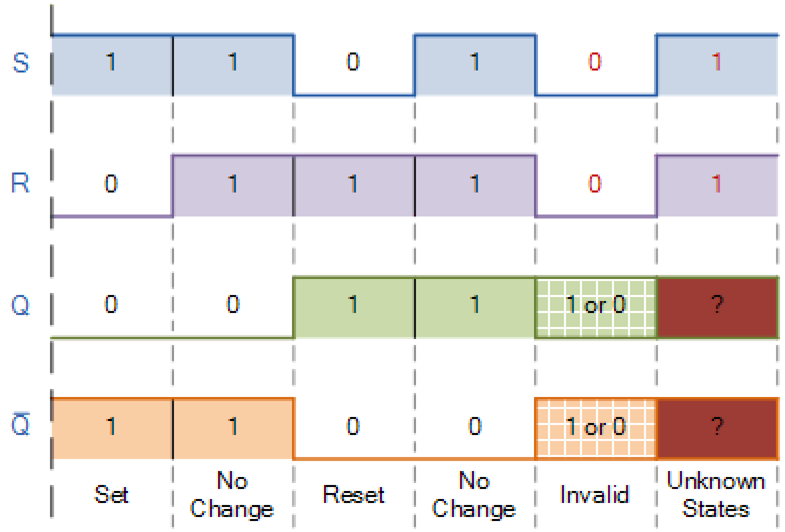


State	S	R	Q		Description
Set	1	0	0	1	Set $\overline{Q}$ » 1
	1	1	0	1	no change
Reset	0	1	1	0	Reset $\overline{Q}$ » 0
	1	1	1	0	no change
Invalid	0	0	1	1	Invalid Condition

It can be seen that when both inputs S = “1” and R = “1” the outputs Q and Q can be at either logic level “1” or “0”, depending upon the state of the inputs S or R BEFORE this input condition existed. Therefore the condition of S = R = “1” does not change the state of the outputs Q and Q.

However, the input state of S = “0” and R = “0” is an undesirable or invalid condition and must be avoided. The condition of S = R = “0” causes both outputs Q and Q to be HIGH together at logic level “1” when we would normally want Q to be the inverse of Q. The result is that the flip-flop looses control of Q and Q, and if the two inputs are now switched “HIGH” again after this condition to logic “1”, the flip-flop becomes unstable and switches to an unknown data state based upon the unbalance as shown in the following switching diagram.

S-R Flip-flop Switching Diagram



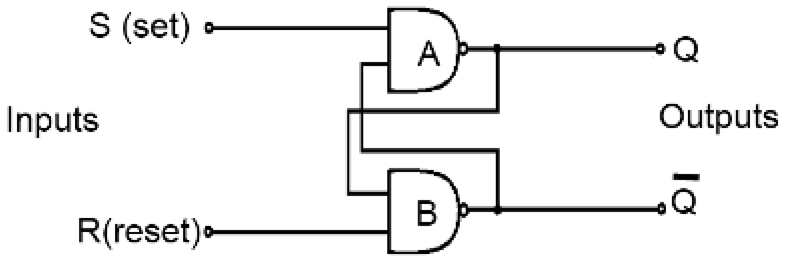
This unbalance can cause one of the outputs to switch faster than the other resulting in the flip-flop switching to one state or the other which may not be the required state and data corruption will exist. This unstable condition is generally known as its **Meta-stable** state.

Then, a simple NAND gate SR flip-flop or NAND gate SR latch can be set by applying a logic “0”, (LOW) condition to its Set input and reset again by then applying a logic “0” to its Reset input. The SR flip-flop is said to be in an “invalid” condition (Meta-stable) if both the set and reset inputs are activated simultaneously.

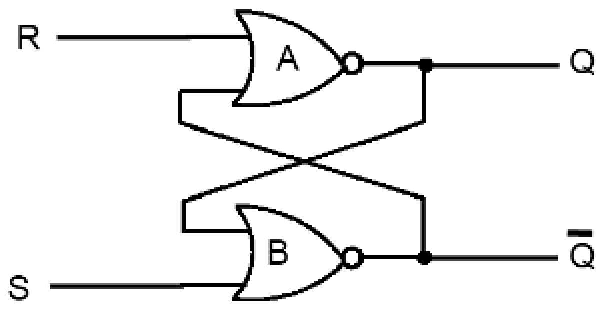
Latch Flip Flop

The R-S (Reset Set) flip flop is the simplest flip flop of all and easiest to understand. It is basically a device which has two outputs one output being the inverse or complement of the other, and two inputs. A pulse on one of the inputs to take on a particular logical state. The outputs will then remain in this state until a similar pulse is applied to the other input. The two inputs are called the Set and Reset input (sometimes called the preset and clear inputs).

Such flip flop can be made simply by cross coupling two inverting gates either NAND or NOR gate could be used Figure 1(a) shows on RS flip flop using NAND gate and Figure 1(b) sh ows the same circuit using NOR gate.



(a) Latch Flip Flop NAND Gate



(b) RS Latch Flip Flop NOR Gate

Figure 1: Latch R-S Flip Flop Using NAND and NOR Gates

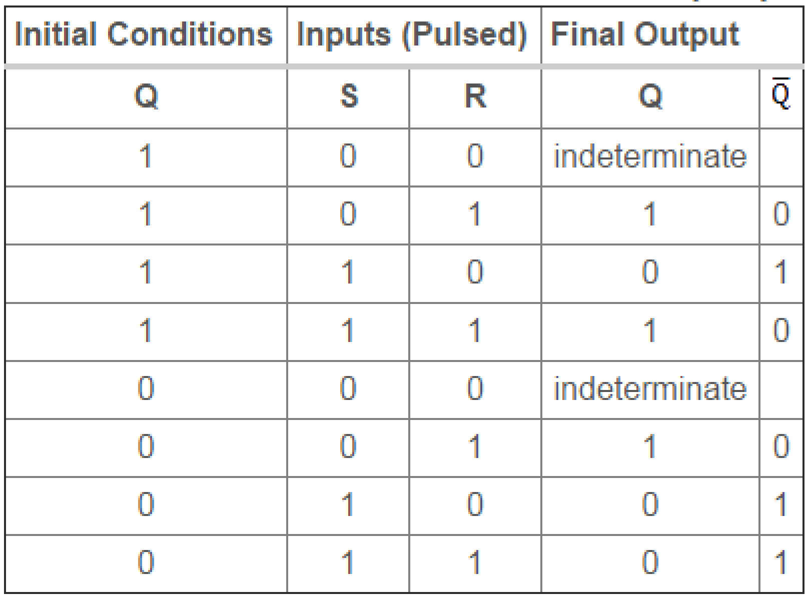
To describe the circuit of Figure 1(a), assume that initially both R and S are at the logic 1 state and that output is at the logic 0 state.

Now, if Q = 0 and R = 1, then these are the states of inputs of gate B, therefore the outputs of gate B is at 1 (making it the inverse of Q i.e. 0). The output of gate B is connected to an input of gate A so if S = 1, both inputs of gate A are at the logic 1 state. This means that the output of gate A must be 0 (as was originally specified). In other words, the 0 state at Q is continuously disabling gate B so that any change in R has no effect. Also the 1 state at $\bar{Q}$ is continuously enabling gate A so that any change S will be transmitted through to Q. The above conditions constitute one of the stable states of the device referred to as the Reset state since Q = 0.

Now suppose that the R-S flip flop in the Reset state, the S input goes to 0. The output of gate A i.e. Q will go to 1 and with Q = 1 and R = 1, the output of gates B ($\bar{Q}$) will go to 0 with $\bar{Q}$ now 0 gate A is disabled keeping Q at 1. Consequently, when S returns to the 1 state it has no effect on the flip flop whereas a change in R will cause a change in the output of gate B. The above conditions constitute the other stable state of the device, called the Set state since Q = 1. Note that the change of the state of S from 1 to 0 has caused the flip flop to change from the Reset state to the Set state.

There is another input condition which has not yet been considered. That is when both the R and S inputs are taken to the logic state 0. When this happens both Q and $\bar{Q}$ will be forced to 1 and will remain so far as long as R and S are kept at 0. However when both inputs return to 1 there is no way of knowing whether the flip flop will latch in the Reset state or the Set state. The condition is said to be indeterminate because of this indeterminate state great care must be taken when using R-S flip flop to ensure that both inputs are not instructed simultaneously.

Table 1: The truth table for the NAND R-S flip flop



Initial Conditions	Inputs (Pulsed)		Final Output	
Q	S	R	Q	$\bar{Q}$
1	0	0	indeterminate	
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0
0	0	0	indeterminate	
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1

Table 2: Simple NAND R-S Flip Flop Truth Table

S	R	Q
0	0	indeterminate
0	1	Set (1)
1	0	Reset(0)
1	1	No Change

Table 3: NOR Gate R-S Flip Flop Truth Table

S	R	Q
0	0	No Change
0	1	Reset (0)
1	0	Set (1)
1	1	Indeterminate

Clocked RS Flip Flop

The RS latch flip flop required the direct input but no clock. It is very use full to add clock to control precisely the time at which the flip flop changes the state of its output.

In the clocked R-S flip flop the appropriate levels applied to their inputs are blocked till the receipt of a pulse from an other source called clock. The flip flop changes state only when clock pulse is applied depending upon the inputs. The basic circuit is shown in Figure 2. This circuit is formed by adding two AND gates at inputs to the R-S flip flop. In addition to control inputs Set (S) and Reset (R), there is a clock input (C) also.

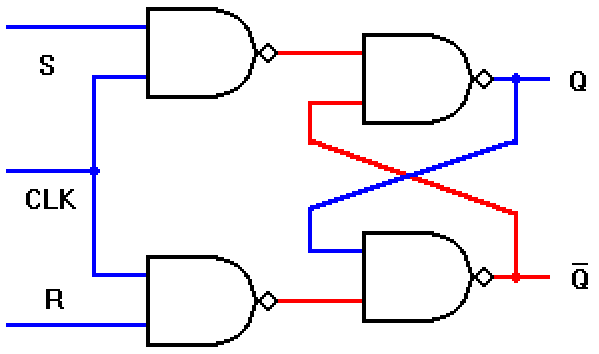


Figure 2: Clocked RS Flip Flop

Table 4: The truth table for the Clocked R-S flip flop

Initial Conditions	Inputs (Pulsed)		Final Output
Q	S	R	Q (t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminate

Table 5: Excitation table for R-S Flip Flop

S	R	Q
0	0	No Change
0	1	Reset (0)
1	0	Set (1)
1	1	Indeterminate

D Flip Flop

A D type (Data or delay flip flop) has a single data input in addition to the clock input as shown in Figure 3.

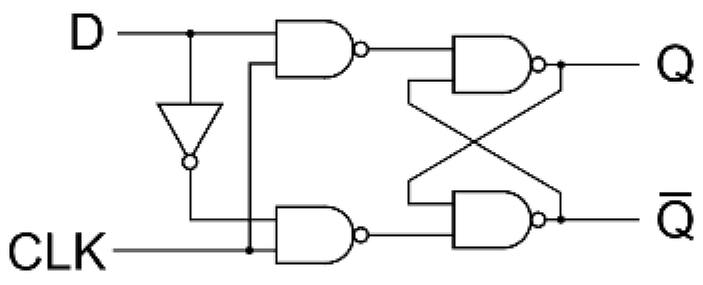


Figure 3: D Flip Flop

Basically, such type of flip flop is a modification of clocked RS flip flop gates from a basic Latch flip flop and NOR gates modify it in to a clock RS flip flop. The D input goes directly to S input and its complement through NOT gate, is applied to the R input.

This kind of flip flop prevents the value of D from reaching the output until a clock pulse occurs. The action of circuit is straight forward as follows.

When the clock is low, both AND gates are disabled, there fore D can change values with out affecting the value of Q. On the other hand, when the clock is high, both AND gates are enabled. In this case, Q is forced equal to D when the clock again goes low, Q retains or stores the last value of D. The truth table for such a flip flop is as given below in table 6.

Table 6: Truth table for D Flip Flop

S	R	Q(t + 1)
0	0	0
0	1	1
1	0	0
1	1	1

Table 7: Excitation table for D Flip Flop

S	Q
0	0
1	1

JK Flip Flop

One of the most useful and versatile flip flop is the JK flip flop the unique features of a JK flip flop are:

1. If the J and K input are both at 1 and the clock pulse is applied, then the output will change state, regardless of its previous condition.
2. If both J and K inputs are at 0 and the clock pulse is applied there will be no change in the output. There is no indeterminate condition, in the operation of JK flip flop i.e. it has no ambiguous state. The circuit diagram for a JK flip flop is shown in Figure 4.

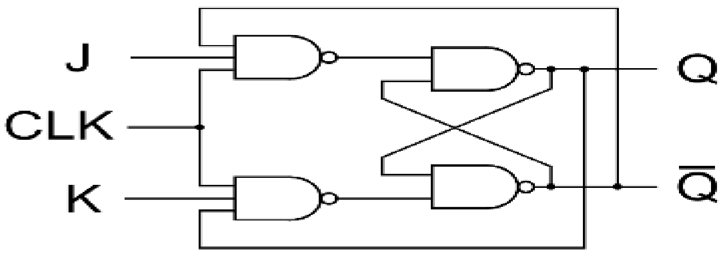


Figure 4: JK Flip Flop

When J = 0 and K = 0

These J and K inputs disable the NAND gates, therefore clock pulse have no effect on the flip flop. In other words, Q returns it last value.

When J = 0 and K = 1,

The upper NAND gate is disabled the lower NAND gate is enabled if Q is 1 therefore, flip flop will be reset (Q = 0 , $\bar{Q}$ =1)if not already in that state.

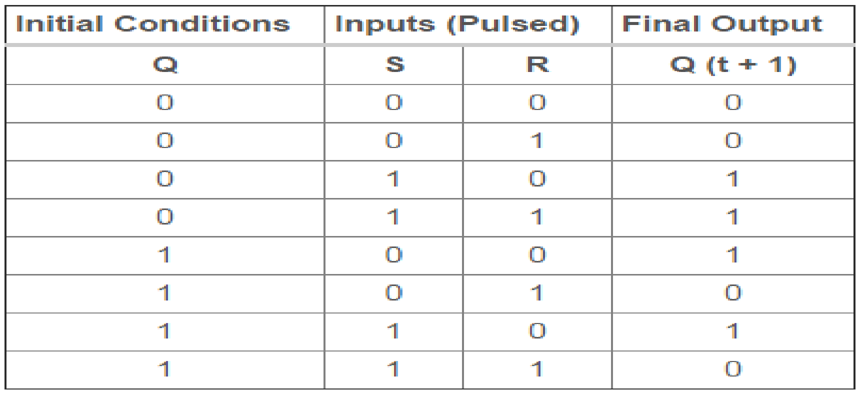
When J = 1 and K = 0

The lower NAND gate is disabled and the upper NAND gate is enabled if $\bar{Q}$ is at 1, As a result we will be able to set the flip flop (Q = 1, $\bar{Q}$ = 0) if not already set

When J = 1 and K = 1

If Q = 0 the lower NAND gate is disabled the upper NAND gate is enabled. This will set the flip flop and hence Q will be 1. On the other hand if Q = 1, the lower NAND gate is enabled and flip flop will be reset and hence Q will be 0. In other words , when J and K are both high, the clock pulses cause the JK flip flop to toggle. Truth table for JK flip flop is shown in table 8.

Table 8: The truth table for the JK flip flop



Initial Conditions	Inputs (Pulsed)		Final Output
Q	S	R	Q (t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Table 6: Excitation table for JK Flip Flop

S	R	Q
0	0	No Change
0	1	0
1	0	0
1	1	Toggle

T Flip Flop

A method of avoiding the indeterminate state found in the working of RS flip flop is to provide only one input (the T input) such, flip flop acts as a toggle switch. Toggle means to change in the previous stage i.e. switch to opposite state. It can be constructed from clocked RS flip flop be incorporating feedback from output to input as shown in Figure 5.

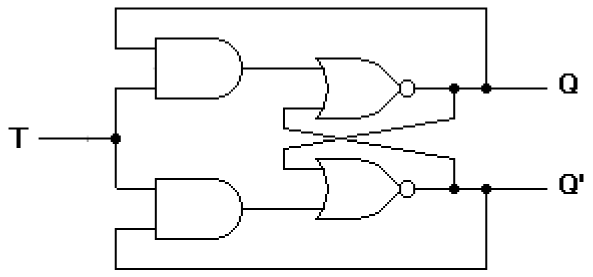


Figure 5: T Flip Flop

Such a flip flop is also called toggle flip flop. In such a flip flop a train of extremely narrow triggers drives the T input each time one of these triggers, the output of the flip flop changes stage. For instance Q equals 0 just before the trigger. Then the upper AND gate is enable and the lower AND gate is disabled. When the trigger arrives, it results in a high S input.

This sets the Q output to 1. When the next trigger appears at the point T, the lower AND gate is enabled and the trigger passes through to the R input this forces the flip flop to reset.

Since each incoming trigger is alternately changed into the set and reset inputs the flip flop toggles. It takes two triggers to produce one cycle of the output waveform. This means the output has half the frequency of the input stated another way, a T flip flop divides the input frequency by two. Thus such a circuit is also called a divide by two circuit.

A disadvantage of the toggle flip flop is that the state of the flip flop after a trigger pulse has been applied is only known if the previous state is known. The truth table for a T flip flop is as given table 7.

Table 7: Truth table for T Flip Flop

Q_n	T	$Q_n + 1$
0	0	0
0	1	1
1	0	1
1	1	0

Table 8: Excitation table for T Flip Flop

T	Q
0	Q_n
1	$\overline{Q}_n$

Generally T flip flop ICs are not available. It can be constructed using JK, RS or D flip flop. Figure 6 shows the relation of T flip flop using JK flip flop.

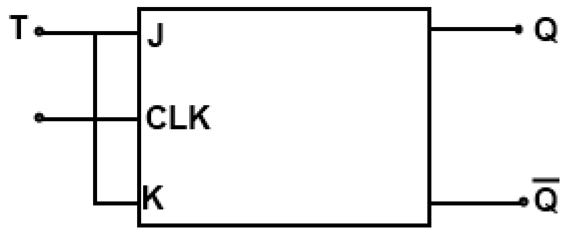


Figure 6: T Flip Flop Using JK Flip Flop

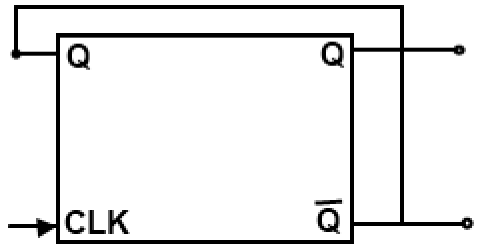


Figure 7: D-type Flip Flop connected as toggle stage

A D-type flip flop may be modified by external connection as a T-type stage as shown in Figure 7. Since the Q logic is used as D-input the opposite of the Q output is transferred into the stage each clock pulse. Thus the stage having Q - 0 transistors $\overline{Q}$ = 1, Providing a toggle action, if the stage had Q = 1 the clock pulse would result in Q = 0 being transferred, again providing the toggle operation. The D-type flip flop connected as in Figure 6 will thus operate as a T-type stage, complementing each clock pulse.

Master Slave Flip Flop

Figure 8 shows the schematic diagram of master sloave J-K flip flop

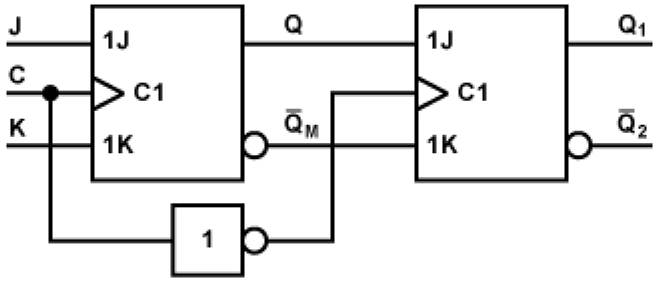


Figure 8: Master Slave JK Flip Flop

A master slave flip flop contains two clocked flip flops. The first is called master and the second slave. When the clock is high the master is active. The output of the master is set or reset according to the state of the input. As the slave is inactive during this period its output remains in the previous state. When clock becomes low the output of the slave flip flop changes because it become active during low clock period. The final output of master slave flip flop is the output of the slave flip flop. So the output of master slave flip flop is available at the end of a clock pulse.

4. Design of counters

Counter is a sequential circuit. A digital circuit which is used for a counting pulses is known counter. Counter is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied. Counters are of two types.

- Asynchronous or ripple counters.
- Synchronous counters

5. Registers

Flip-flop is a 1 bit memory cell which can be used for storing the digital data. To increase the storage capacity in terms of number of bits, we have to use a group of flip-flop. Such a group of flip-flop is known as a Register. The n-bit register will consist of n number of flip-flop and it is capable of storing an n-bit word.

6. Shift Register

The Shift Register is another type of sequential logic circuit that can be used for the storage or the transfer of data in the form of binary numbers. This sequential device loads the data present on its inputs and then moves or "shifts" it to its output once every clock cycle, hence the name "shift register".

A shift register basically consists of several single bit "D-Type Data Latches", one for each data bit, either a logic "0" or a "1", connected together in a serial type daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on.

Data bits may be fed in or out of a shift register serially, that is one after the other from either the left or the right direction, or all together at the same time in a parallel configuration.

The number of individual data latches required to make up a single Shift Register device is usually determined by the number of bits to be stored with the most common being 8-bits (one byte) wide constructed from eight individual data latches.

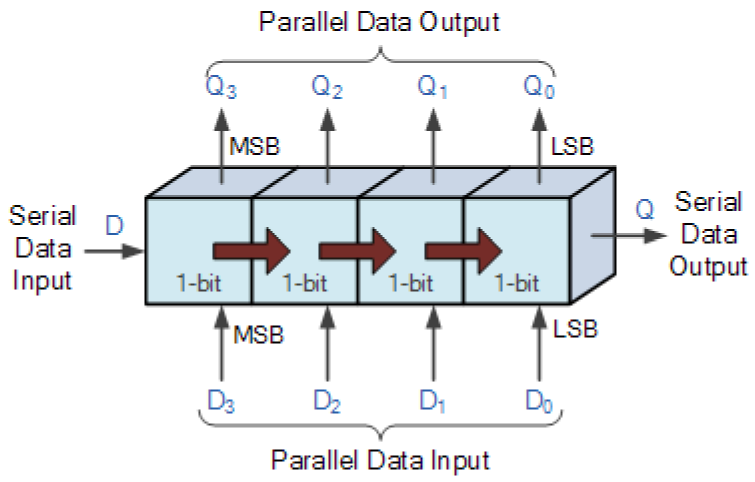
Shift Registers are used for data storage or for the movement of data and are therefore commonly used inside calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format. The individual data latches that make up a single shift register are all driven by a common clock (Clk) signal making them synchronous devices.

Shift register IC's are generally provided with a clear or reset connection so that they can be "SET" or "RESET" as required. 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 at the output in parallel form.

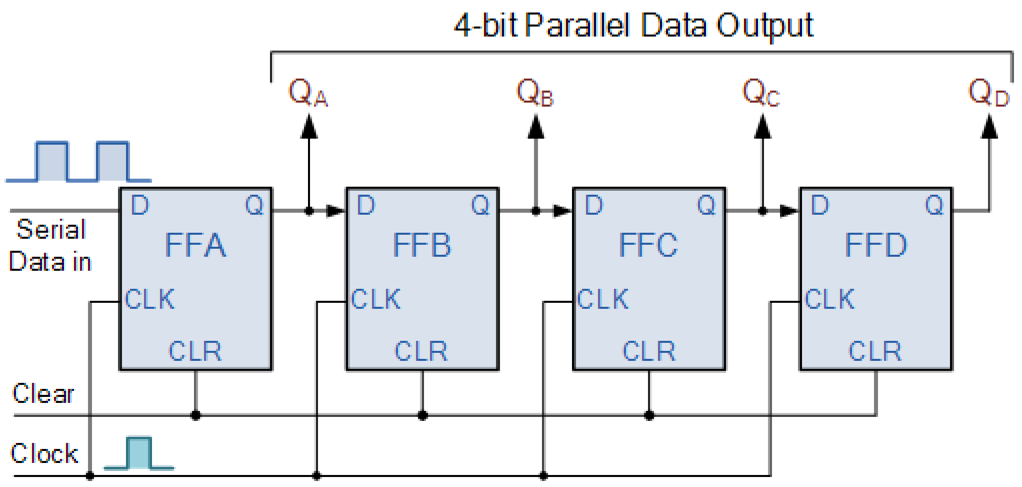
- Serial-in to Serial-out (SISO) - the data is shifted serially “IN” and “OUT” of the register, one bit at a time in either a left or right direction under clock control.
- Parallel-in to Serial-out (PISO) - the parallel data is loaded into the register simultaneously and is shifted out of the register 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.

The effect of data movement from left to right through a shift register can be presented graphically as:



Also, the directional movement of the data through a shift register can be either to the left, (left shifting) to the right, (right shifting) left-in but right-out, (rotation) or both left and right shifting within the same register thereby making it *bidirectional*. In this tutorial it is assumed that all the data shifts to the right, (right shifting).

4-bit Serial-in to Parallel-out Shift Register



The operation is as follows. 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. The logic "1" has now moved or been "shifted" one place along the register to the right as it is now atQ_A.

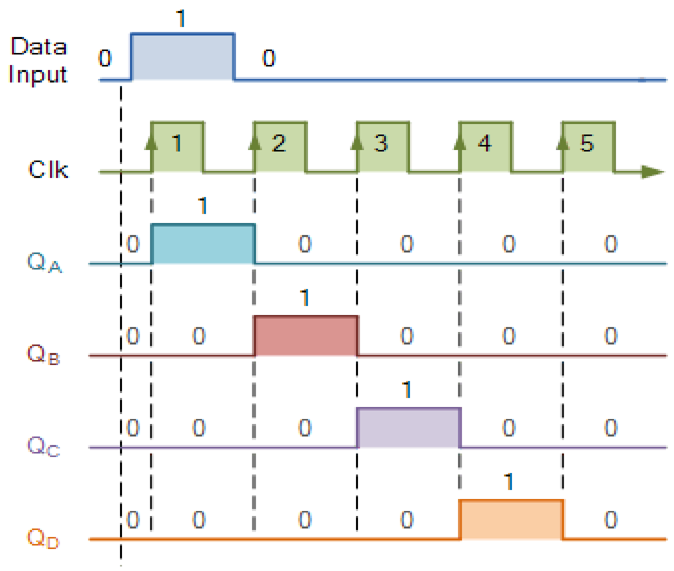
When the third clock pulse arrives this logic "1" value moves to the output of FFC (Q_C) and so on until the arrival of the fifth clock pulse which sets all the outputs Q_A to Q_D back again to logic level "0" because the input to FFA has remained constant at logic level "0".

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 stored in the register. This data value can now be read directly from the outputs of Q_A to Q_D.

Then the data has been converted from a serial data input signal to a parallel data output. The truth table and following waveforms show the propagation of the logic "1" through the register from left to right as follows.

Basic Data Movement Through A Shift Register

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



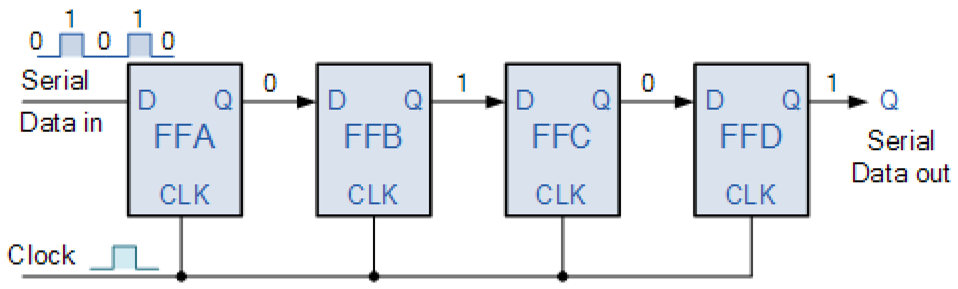
Note that after the fourth clock pulse has ended the 4-bits of data (0-0-0-1) are stored in the register and will remain there provided clocking of the register has stopped. In practice the input data to the register may consist of various combinations of logic "1" and "0". Commonly available SIPO IC's include the standard 8-bit 74LS164 or the 74LS594.

Serial-in to Serial-out (SISO) Shift Register

This shift register is very similar to the SIPO above, except were before the data was read directly in a parallel form from the outputs Q_A to Q_D, this time the data is allowed to flow straight through the register and out of 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.

The SISO shift register is one of the simplest of the four configurations as it has only three connections, the serial input (SI) which determines what enters the left hand flip-flop, the serial output (SO) which is taken from the output of the right hand flip-flop and the sequencing clock signal (Clk). The logic circuit diagram below shows a generalized serial-in serial-out shift register.

4-bit Serial-in to Serial-out Shift Register



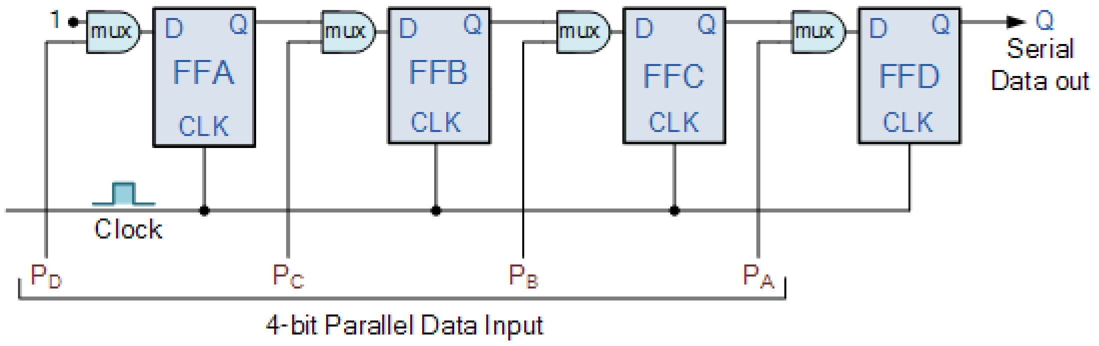
You may think what the point of a SISO shift register is if the output data is exactly the same as the input data. Well this type of Shift Register also acts as a temporary storage device or it can act as a time delay device for the data, with the amount of time delay being controlled by the number of stages in the register, 4, 8, 16 etc or by varying the application of the clock pulses. Commonly available IC's include the 74HC595 8-bit Serial-in to Serial-out Shift Register all with 3-state outputs.

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

The Parallel-in to Serial-out shift register acts in the opposite way to the serial-in to parallel-out one above. The data is loaded into the register in a parallel format 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 then read out sequentially in the normal shift-right mode from the register at Q representing the data present at P_A to P_D.

This data is outputted one bit at a time on each clock cycle in a serial format. It is important to note that with this type of data 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.

4-bit Parallel-in to Serial-out Shift Register

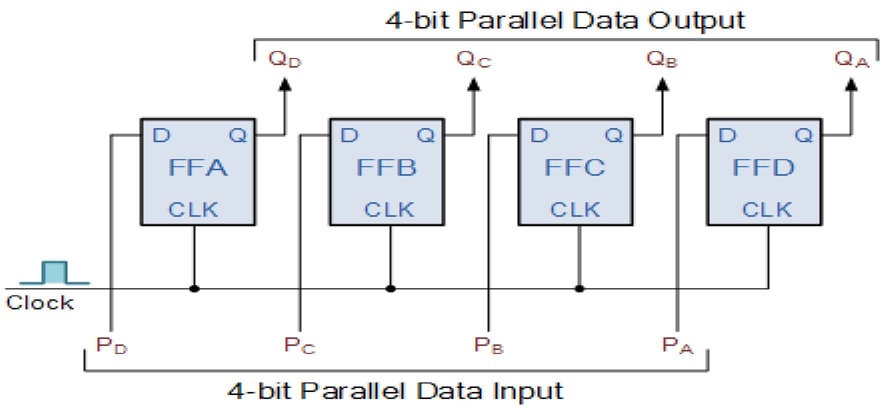


As this type of shift register converts parallel data, such as an 8-bit data word into serial format, it can be used to multiplex many different input lines into a single serial DATA stream which can be sent directly to a computer or transmitted over a communications line. Commonly available IC's include the 74HC166 8-bit Parallel-in/Serial-out Shift Registers.

Parallel-in to Parallel-out (PIPO) Shift Register

The final mode of operation is the Parallel-in to Parallel-out Shift Register. This type of shift register also acts as a temporary storage device or as a time delay device similar to the SISO configuration above. 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_A by the same clock pulse. Then one clock pulse loads and unloads the register. This arrangement for parallel loading and unloading is shown below.

4-bit Parallel-in to Parallel-out Shift Register



The PIPO shift register is the simplest of the four configurations as it has only three connections, the parallel input (PI) which determines what enters the flip-flop, the parallel output (PO) and the sequencing clock signal (Clk).

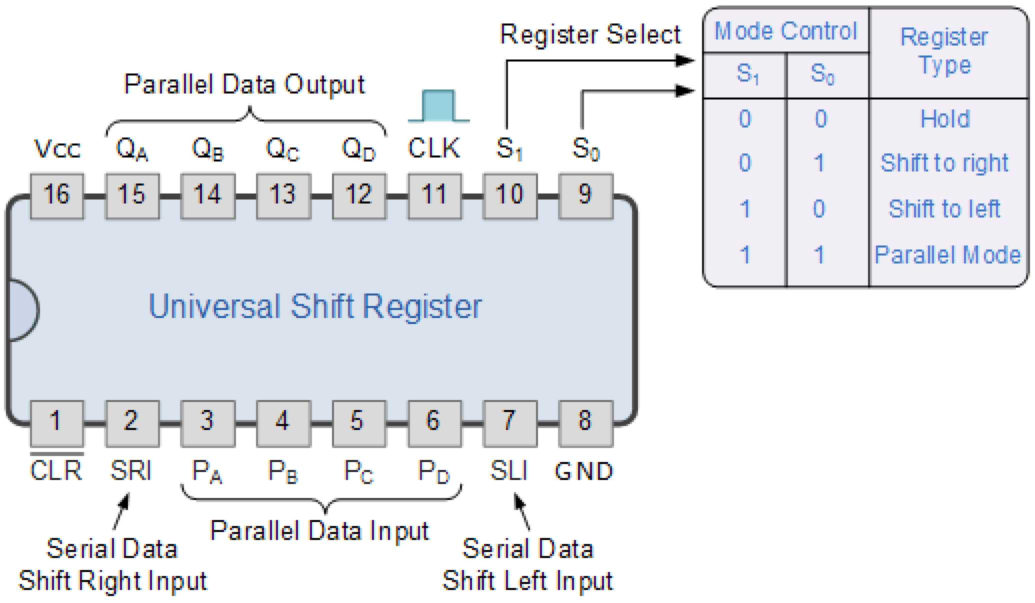
Similar to the Serial-in to Serial-out shift register, this type of register also acts as a temporary storage device or as a time delay device, with the amount of time delay being varied by the frequency of the clock pulses. Also, in this type of register there are no interconnections between the individual flip-flops since no serial shifting of the data is required.

Universal Shift Register

Today, there are many high speed bi-directional “universal” type Shift Registers available such as the TTL 74LS194, 74LS195 or the CMOS 4035 which are available as 4-bit multi-function devices that can be used in either serial-to-serial, left shifting, right shifting, serial-to-parallel, parallel-to-serial, or as a parallel-to-parallel multifunction data register, hence the name “Universal”.

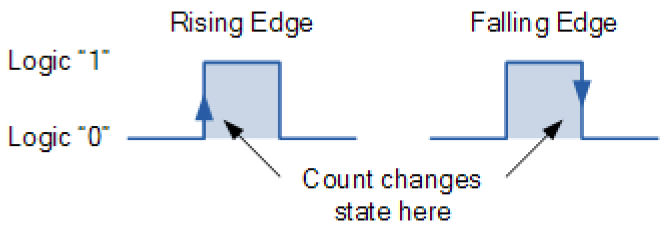
These universal shift registers can perform any combination of parallel and serial input to output operations but require additional inputs to specify desired function and to pre-load and reset the device. A commonly used universal shift register is the TTL 74LS194 as shown below.

4-bit Universal Shift Register 74LS194



Mode Control		Register Type
S_1	S_0	
0	0	Hold
0	1	Shift to right
1	0	Shift to left
1	1	Parallel Mode

Universal shift registers are very useful digital devices. They can be configured to respond to operations that require some form of temporary memory storage or for the delay of information such as the SISO or PIPO configuration modes or transfer data from one point to another in either a serial or parallel format. Universal shift registers are frequently used in arithmetic operations to shift data to the left or right for multiplication or division.

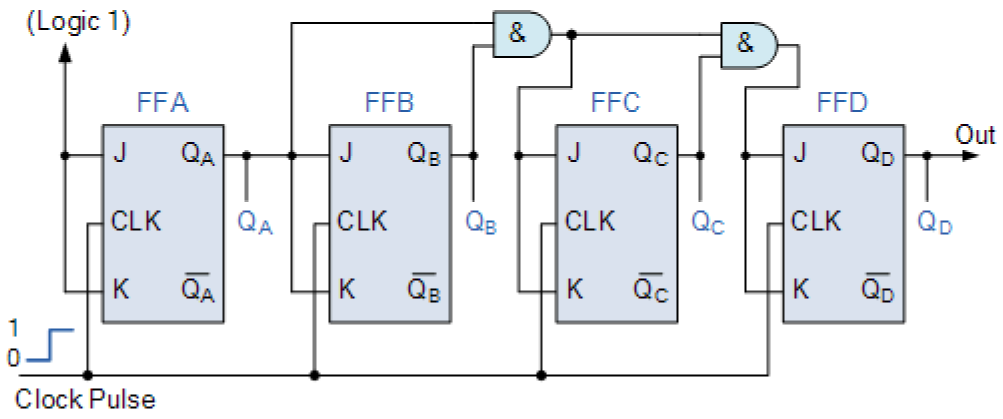


7. **Binary Synchronous Counter**

In Asynchronous binary counter , the output of one counter stage is connected directly to the clock input of the next counter stage and so on along the chain, and as a result the asynchronous counter suffers from what is known as "Propagation Delay" in which the timing signal is delayed a fraction through each flip-flop.However, with the **Synchronous Counter**, the external clock signal is connected to the clock input of EVERY individual flip-flop within the counter so that all of the flip-flops are clocked together simultaneously (in parallel) at the same time giving a fixed time relationship. In other words, changes in the output occur in "synchronisation" with the clock signal.

The result of this synchronisation is that all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay.

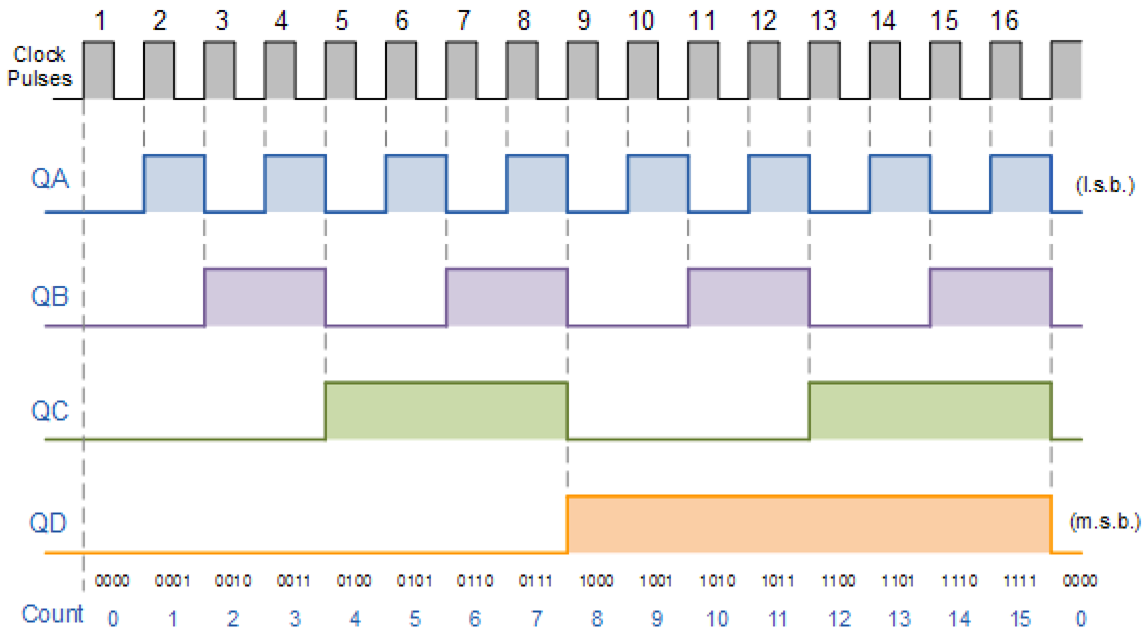
Binary 4-bit Synchronous Up Counter



It can be seen above, that the external clock pulses (pulses to be counted) are fed directly to each of the **J-K flip-flops** in the counter chain and that both the J and K inputs are all tied together in toggle mode, but only in the first flip-flop, flip-flop FFA(LSB) are they connected HIGH, logic "1" allowing the flip-flop to toggle on every clock pulse. Then the synchronous counter follows a predetermined sequence of states in response to the common clock signal, advancing one state for each pulse.The J and K inputs of flip-flop FFB are connected directly to the output Q_A of flip-flopFFA, but the J and K inputs of flip-flops FFC and FFD are driven from separate AND gates which are also supplied with signals from the input and output of the previous stage. These additional AND gates generate the required logic for the JK inputs of the next stage.If we enable each JK flip-flop to toggle based on whether or not all preceding flip-flop outputs (Q) are "HIGH" we can obtain the same counting sequence as with the asynchronous circuit but without the ripple effect, since each flip-flop in this circuit will be clocked at exactly the same time.Then as there is no inherent propagation delay in synchronous counters, because all the counter stages are triggered in parallel at the same time, the maximum operating frequency of this type of frequency counter is much higher than that for a similar asynchronous counter circuit.

8. Timing sequence

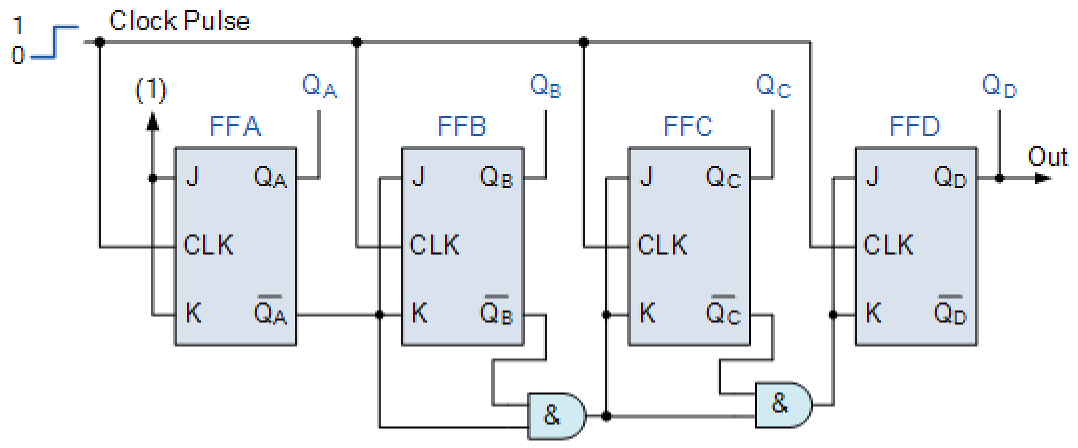
4-bit Synchronous Counter Waveform Timing Diagram.



Because this 4-bit synchronous counter counts sequentially on every clock pulse the resulting outputs count upwards from 0 (0000) to 15 (1111). Therefore, this type of counter is also known as a **4-bit Synchronous Up Counter**.

However, we can easily construct a **4-bit Synchronous Down Counter** by connecting the AND gates to the Q output of the flip-flops as shown to produce a waveform timing diagram the reverse of the above. Here the counter starts with all of its outputs HIGH (1111) and it counts down on the application of each clock pulse to zero, (0000) before repeating again.

Binary 4-bit Synchronous Down Counter

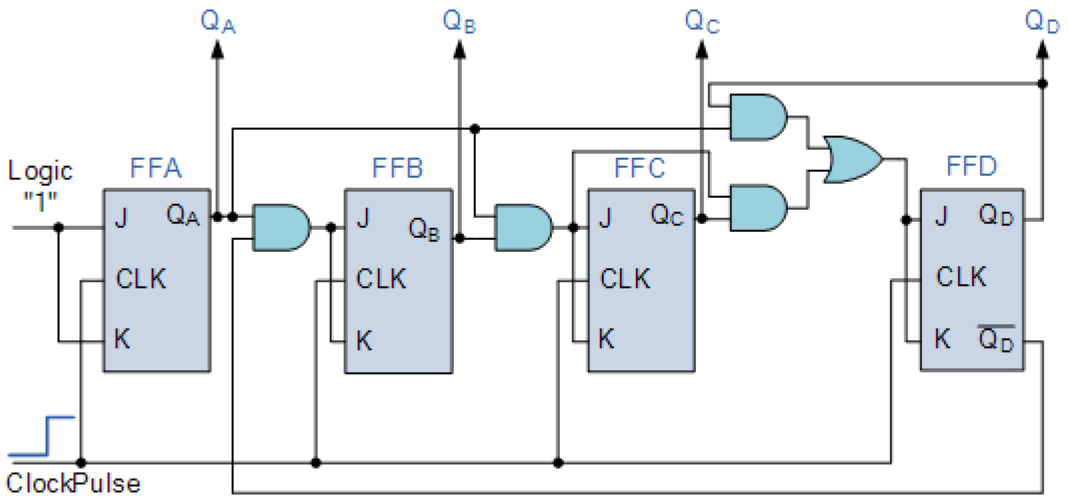


As synchronous counters are formed by connecting flip-flops together and any number of flip-flops can be connected or "cascaded" together to form a "divide-by-n" binary counter, the modulo's or "MOD" number still applies as it does for asynchronous counters so a Decade counter or BCD counter with counts from 0 to 2^n-1 can be built along with truncated sequences. All we need to increase the MOD count of an up or down synchronous counter is an additional flip-flop and AND gate across it.

Decade 4-bit Synchronous Counter

A 4-bit decade synchronous counter can also be built using synchronous binary counters to produce a count sequence from 0 to 9. A standard binary counter can be converted to a decade (decimal 10) counter with the aid of some additional logic to implement the desired state sequence. After reaching the count of "1001", the counter recycles back to "0000". We now have a decade or **Modulo-10** counter.

Decade 4-bit Synchronous Counter



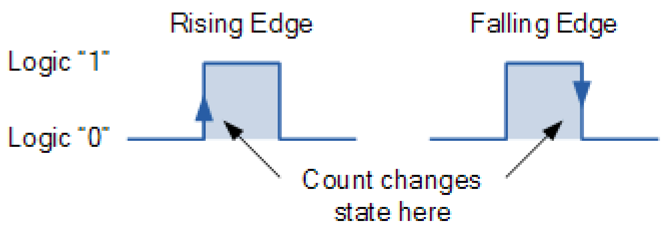
The additional AND gates detect when the counting sequence reaches "1001", (Binary 10) and causes flip-flop FF3 to toggle on the next clock pulse. Flip-flop FF0 toggles on every clock pulse. Thus, the count is reset and starts over again at "0000" producing a synchronous decade counter.

We could quite easily re-arrange the additional AND gates in the above counter circuit to produce other count numbers such as a Mod-12 counter which counts 12 states from"0000″ to "1011" (0 to 11) and then repeats making them suitable for clocks, etc.

Triggering A Synchronous Counter

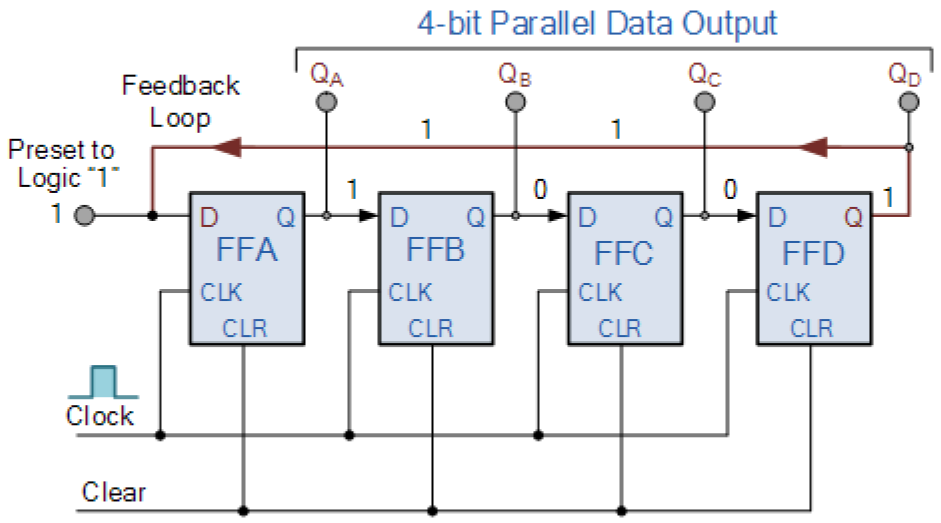
Synchronous Counters use edge-triggered flip-flops that change states on either the "positive-edge" (rising edge) or the "negative-edge" (falling edge) of the clock pulse on the control input resulting in one single count when the clock input changes state.

Generally, synchronous counters count on the rising-edge which is the low to high transition of the clock signal and asynchronous ripple counters count on the falling-edge which is the high to low transition of the clock signal.



It may seem unusual that ripple counters use the falling-edge of the clock cycle to change state, but this makes it easier to link counters together because the most significant bit (MSB) of one counter can drive the clock input of the next.This works because the next bit must change state when the previous bit changes from high to low – the point at which a carry must occur to the next bit. Synchronous counters usually have a carry-out and a carry-in pin for linking counters together without introducing any propagation delays.

4-bit Ring Counter

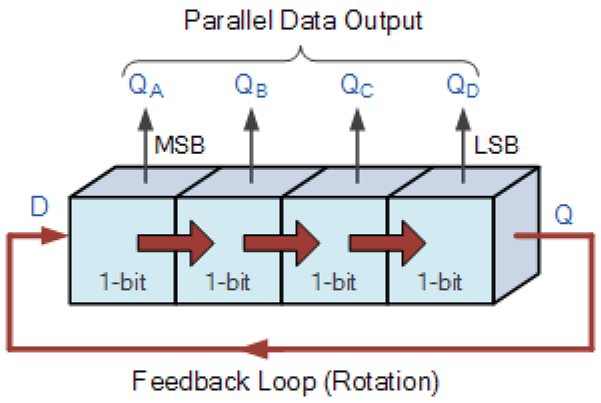


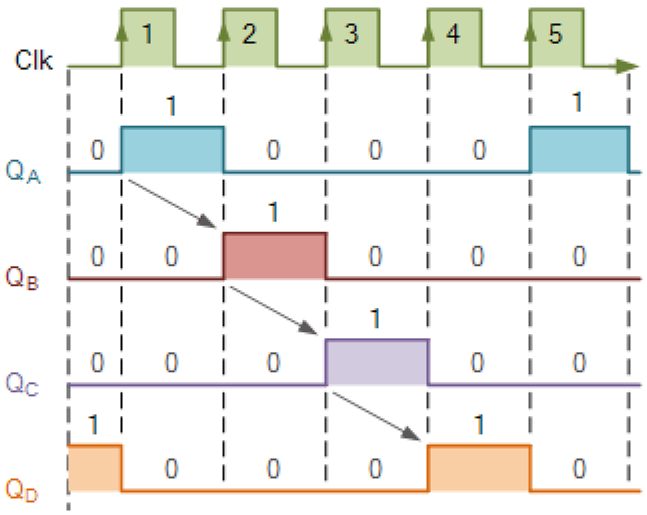
The synchronous Ring Counter example above is preset so that exactly one data bit in the register is set to logic "1" with all the other bits reset to "0". To achieve this, a "CLEAR" signal is firstly applied to all the flip-flops together in order to "RESET" their outputs to a logic "0" level and then a "PRESET" pulse is applied to the input of the first flip-flop (FFA) before the clock pulses are applied. This then places a single logic "1" value into the circuit of the ring counter.

So on each successive clock pulse, the counter circulates the same data bit between the four flip-flops over and over again around the "ring" every fourth clock cycle. But in order to cycle the data correctly around the counter we must first "load" the counter with a suitable data pattern as all logic "0's" or all logic "1's" outputted at each clock cycle would make the ring counter invalid.

This type of data movement is called "rotation", and like the previous shift register, the effect of the movement of the data bit from left to right through a ring counter can be presented graphically as follows along with its timing diagram:

Rotational Movement of a Ring Counter





Since the ring counter example shown above has four distinct states, it is also known as a "modulo-4" or "mod-4" counter with each flip-flop output having a frequency value equal to one-fourth or a quarter (1/4) that of the main clock frequency.

The “MODULO” or “MODULUS” of a counter is the number of states the counter counts or sequences through before repeating itself and a ring counter can be made to output any modulo number. A “mod-n” ring counter will require “n” number of flip-flops connected together to circulate a single data bit providing “n” different output states.

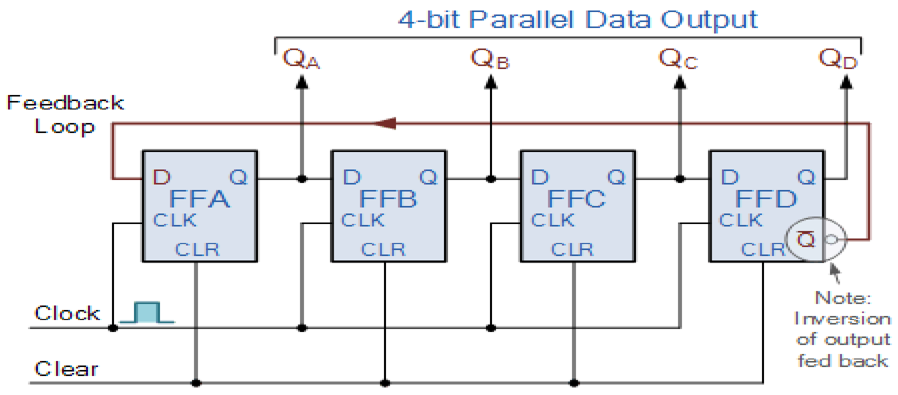
For example, a mod-8 ring counter requires eight flip-flops and a mod-16 ring counter would require sixteen flip-flops. However, as in our example above, only four of the possible sixteen states are used, making ring counters very inefficient in terms of their output state usage.

Johnson Ring Counter

The Johnson Ring Counter or “Twisted Ring Counters”, is another shift register with feedback exactly the same as the standard Ring Counter above, except that this time the inverted output Q of the last flip-flop is now connected back to the input D of the first flip-flop as shown below.

The main advantage of this type of ring counter is that it only needs half the number of flip-flops compared to the standard ring counter then its modulo number is halved. So a “n-stage” Johnson counter will circulate a single data bit giving sequence of 2ndifferent states and can therefore be considered as a “mod-2n counter”.

4-bit Johnson Ring Counter



This inversion of Q before it is fed back to input D causes the counter to "count" in a different way. Instead of counting through a fixed set of patterns like the normal ring counter such as for a 4-bit counter, "0001"(1), "0010"(2), "0100"(4), "1000"(8) and repeat, the Johnson counter counts up and then down as the initial logic "1" passes through it to the right replacing the preceding logic "0".

A 4-bit Johnson ring counter passes blocks of four logic "0" and then four logic "1" thereby producing an 8-bit pattern. As the inverted output Q is connected to the input D this 8-bit pattern continually repeats. For example, "1000", "1100", "1110", "1111", "0111", "0011", "0001", "0000" and this is demonstrated in the following table below.

Truth Table for a 4-bit Johnson Ring Counter

Clock Pulse No	FFA	FFB	FFC	FFD
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

As well as counting or rotating data around a continuous loop, ring counters can also be used to detect or recognize various patterns or number values within a set of data. By connecting simple logic gates such as the AND or the OR gates to the outputs of the flip-flops the circuit can be made to detect a set number or value.

Standard 2, 3 or 4-stage Johnson Ring Counters can also be used to divide the frequency of the clock signal by varying their feedback connections and divide-by-3 or divide-by-5 outputs are also available.

For example, a 3-stage Johnson Ring Counter could be used as a 3-phase, 120 degree phase shift square wave generator by connecting to the data outputs at A, B and NOT-B.

The standard 5-stage Johnson counter such as the commonly available CD4017 is generally used as a synchronous decade counter/divider circuit.

Other combinations such as the smaller 2-stage circuit which is also called a “Quadrature” (sine/cosine) Oscillator or Generator can be used to produce four individual outputs that are each 90 degrees “out-of-phase” with respect to each other to produce a 4-phase timing signal as shown below.

9. Algorithmic State Machines

Algorithm State Machines(ASM) ASM stands for 'Algorithm State Machine 'or simply state machine is the another name given to sequential network is used to control a digital system which carries out a step by a step –by step procedure .It should be noted that ASM charts represent physical hardware and offers several advantages.

1. Operation of a digital system can be easily understand by inspection of the SM chart .

2. ASM charts represent physical hardware.

3. The ASM chart are equivalent to a state graph, and it leads directly to a hardware realization . 4. ASM charts can be described the operation of both combinational and sequential circuits .

5. ASM charts are easier to understand and can be converted several equivalent form.

6. The ASM chart may be equivalently expressed as a state and output table .

10. ASM chart

Principal Component Of An ASM Chart

•State Box.

The state of the system is represented by a state box .It is a rectangular box .At the top left hand corner the name of state is shown ,which at the top right hand corner the state assignment is given .Within the state box ,the output signals are listed .

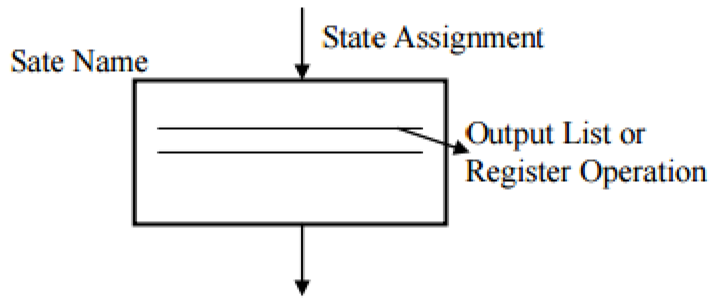


Fig. State box

• Decision box .It a diamond –shaped box with true false branches .Boolean condition is placed in the box and the decision is made from the value of one or more input signals .The decision box must follow and be associated with a state

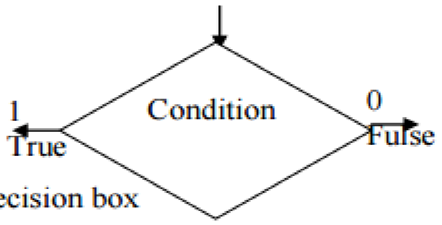
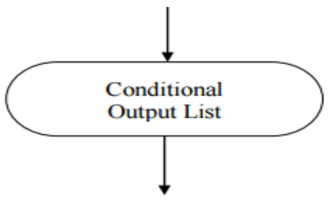


Fig. Decision box

Conditional output box .A condition output box is shown in Fig. is a rectangular box with curved ends .It contain conditional output list .The conditional output depends on both the state of the system and the inputs .Therefore the conditional output signals are sometimes known as Mealy output .A condition output must follow a decision box



Conditional output box

Equivalent ASM charts ASM charts are not unique, it may have more than one equivalent form Fig. shown three equivalent ASM charts for combinational network Z=A(B+C).

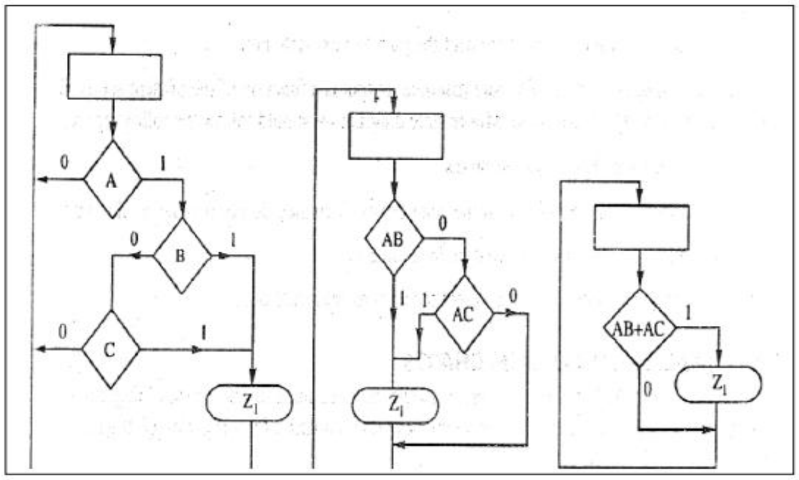


Fig. Equivalent ASM charts for Z=A(B+C)

11. **control implementation**

Conversion Of State Diagram To An ASM Chart

ASM chart can be derived derived an ASM from state diagram of machine ,but certain rules must be followed when constructing an ASM block. First for every valid combination of input, there must be exactly one exit path defined .Second ,no internal feedback within an SM block is allowed.

Mealy Machine.

In case of Mealy machine, output is a function of both present state and input . For construction of ASM chart from Mealy state diagram, we should follow the following steps.

1. Represent each state by state boxes.

2. Put input in decision box after each state box.

3. The Mealy output appears in conditional output boxes since they depend on both the state and input.

4. Mealy circuit output written only when it is equal to '1' i.e. true.

5. Depending on value of input connect the path to next state box.

Example1

Convert the state diagram diagram of Fig. below to ASM chart.

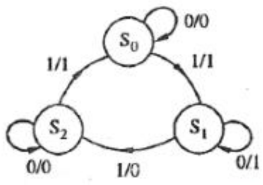


Fig.State diagram

Solution

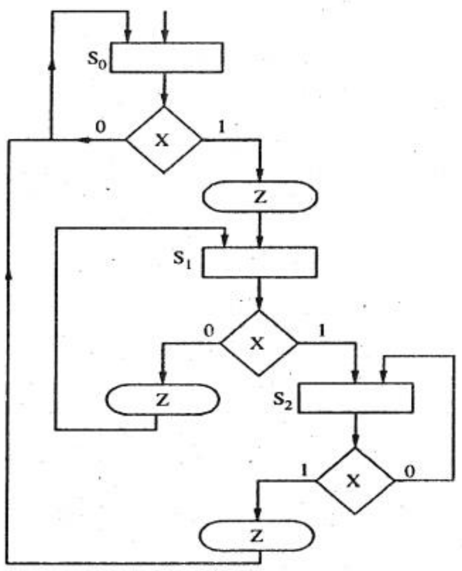


Fig.ASM chart

Example2

Dram an ASM chart to describe a mealy state machine that detects a sequence of 101 and that asserts a logical 1 at the output during the last state of the sequence .

Solution

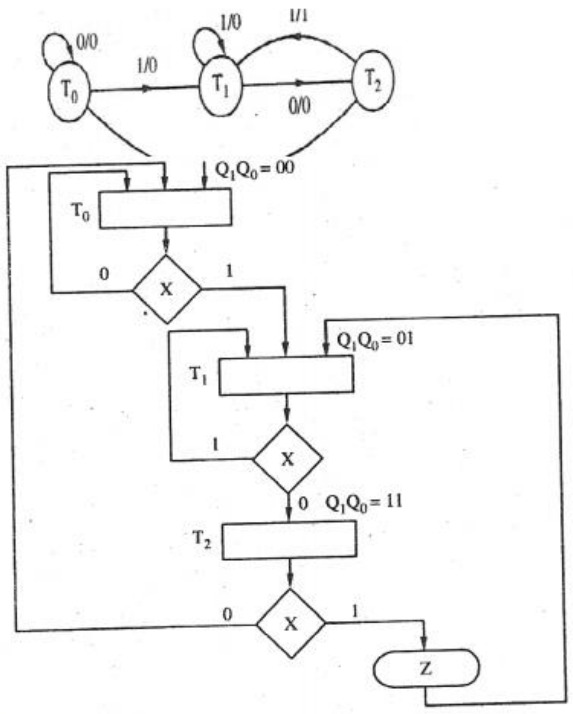


Fig.ASM chart

Moore Machine. In case of Moore machine, output is a function of the present state only . For construction of ASM chart from Moore state diagram, we should follow the following steps

1. Represent each states by state boxes.

2. The Moore output are placed in the state boxes since they do not depend on the input .

3. After each state box put the input in decision box.

4. Depending on value of input connect the path to next state box.

Example3 Convert the state diagram of Fig. below to ASM chart.

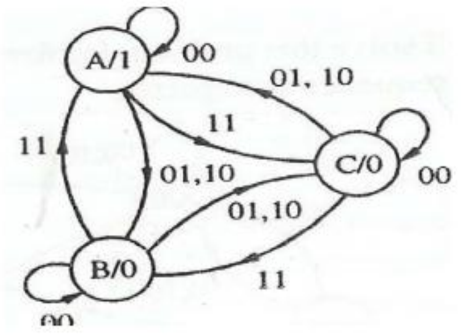


Fig.State Diagram

Solution

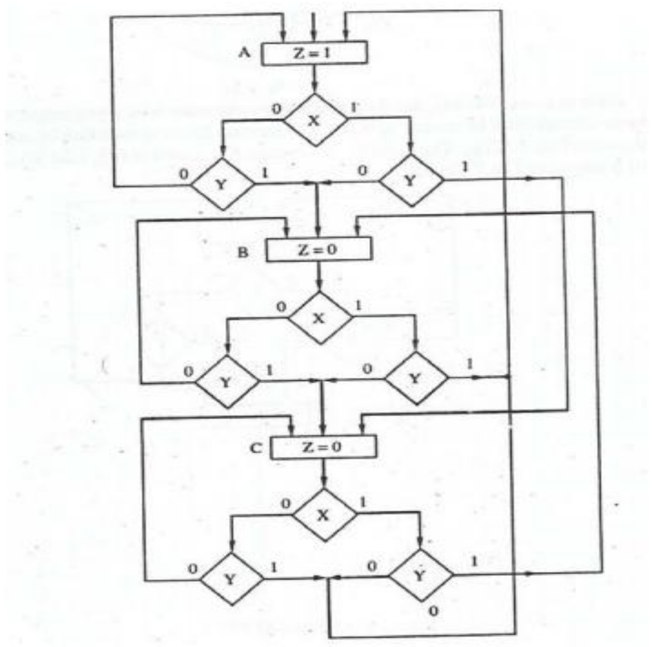
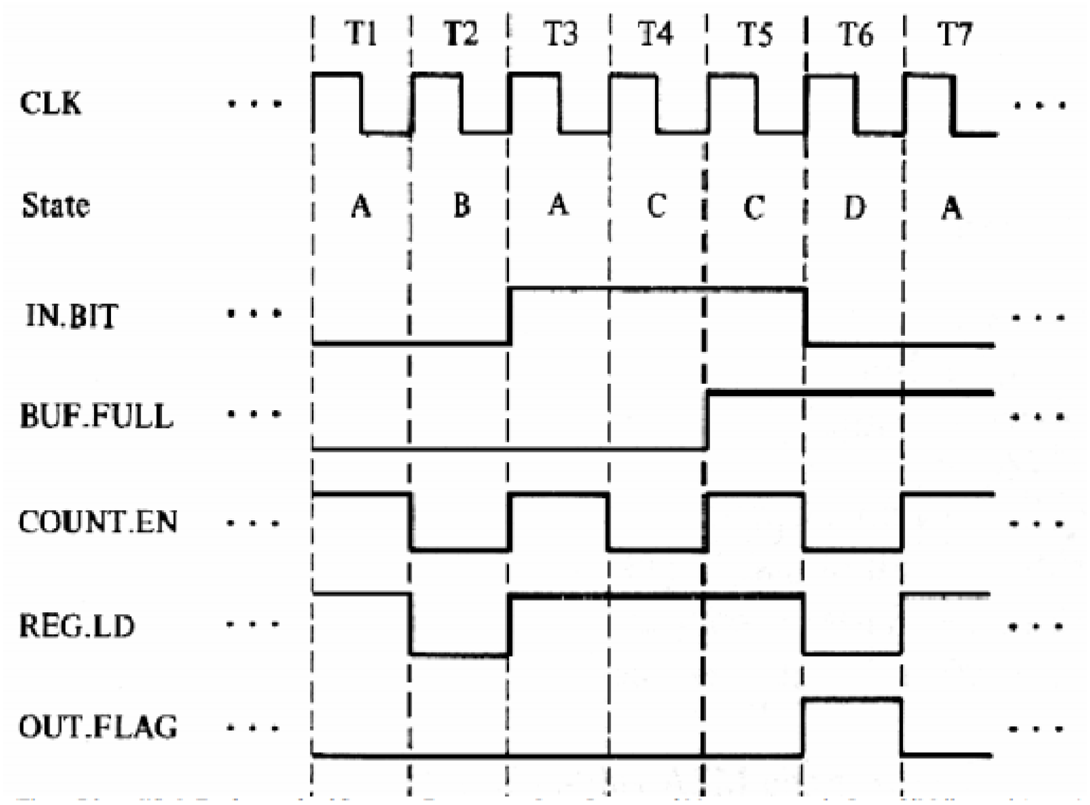


Fig.ASM chart

12. Constructing an ASM Chart from a Timing Diagram



References

1. "Digital Logic and Computer design" by M. Morris Mano
2. Textbook of Digital Fundamentals by Thomas L. Floyd (9th Edition)
3. Logic and Computer Design Fundamentals (4th Edition) 4th Edition by M. Morris R. Mano , Charles R. Kime.