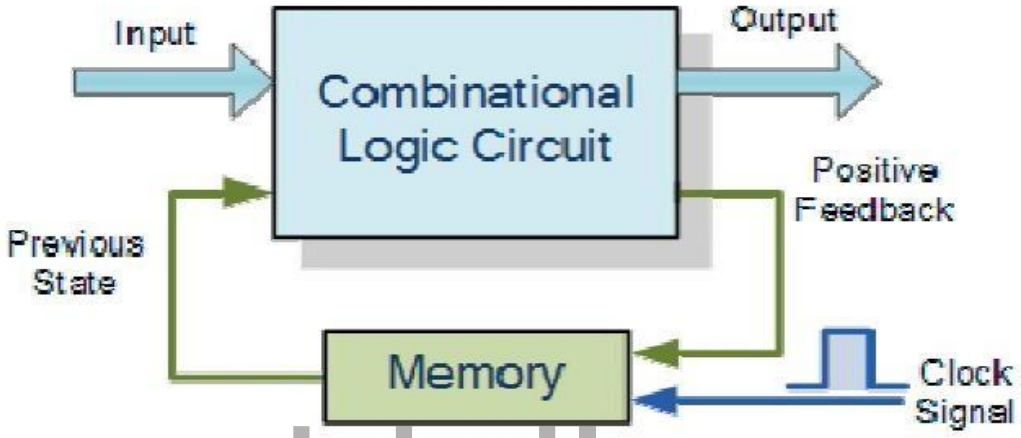
Catalog

3.1 Sequential Logic Representation

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: Flipflops, 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.



Microprocessors

3.1.2 Classification of Sequential Logic

As standard logic gates are the building blocks of combinational circuits, bistable latches and flipflops are the basic building blocks of sequential logic circuits. Sequential logic circuits can be constructed to produce either simple edge-triggered flip-flops or more complex sequential circuits such as storage registers, shift registers, memory devices or counters. Either way sequential logic circuits can be divided into the following three main categories:

1. Event Driven – asynchronous circuits that change state immediately when enabled.

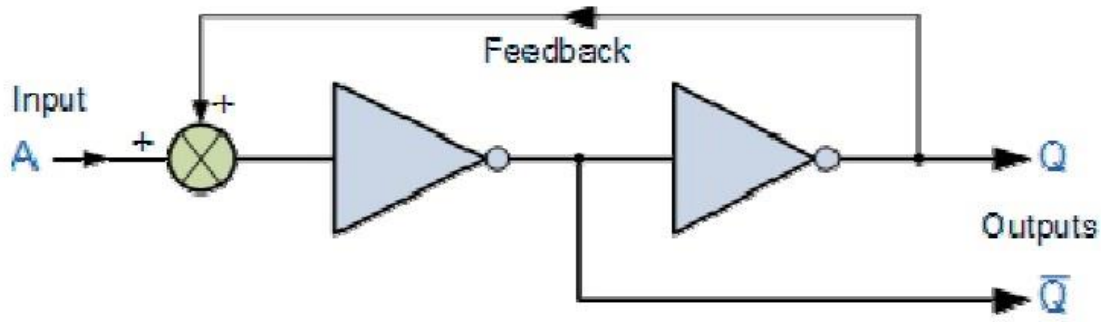
2. Clock Driven – synchronous circuits that are synchronised to a specific clock signal.

3. Pulse Driven – which is a combination of the two that responds to triggering pulses.

As well as the two logic states mentioned above logic level “1” and logic level “0”, a third element is introduced that separates sequential logic circuits from their combinational logic counterparts, namely TIME. Sequential logic circuits return back to their original steady state once reset and sequential circuits with loops or feedback paths are said to be “cyclic” in nature.

We now know that in sequential circuits changes occur only on the application of a clock signal making it synchronous, otherwise the circuit is asynchronous and depends upon an external input. To retain their current state, sequential circuits rely on feedback and this occurs when a fraction of the output is fed back to the input and this is demonstrated as:

Sequential Feedback Loop



The two inverters or NOT gates are connected in series with the output at Q fed back to the input. Unfortunately, this configuration never changes state because the output will always be the same, either a “1” or a “0”, it is permanently set. However, we can see how feedback works by examining the most basic sequential logic components, called the SR flip-flop.

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 one which will “RESET” the device (meaning the output = “0”), labelled **R**. Then the SR description stands for “Set-Reset”.

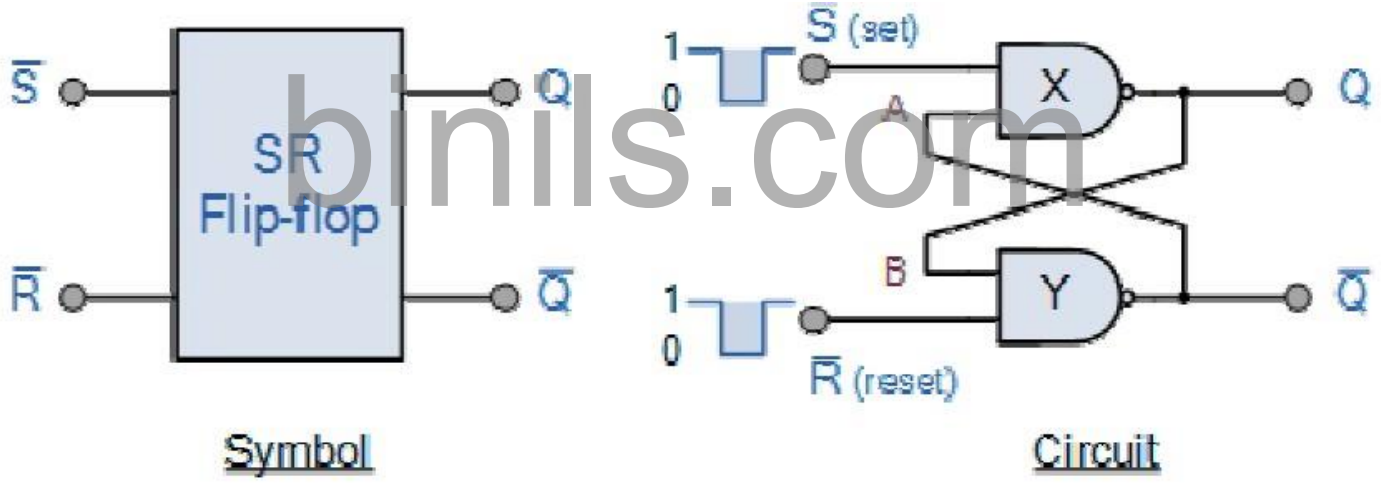
The reset input resets the flip-flop back to its original state with an output Q that will be either at a logic level “1” or logic “0” depending upon this set/reset condition. A basic NAND gate SR flip-flop circuit provides feedback from both of its outputs back to its opposing inputs and is commonly 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 NAND Gate SR Flip-Flop

The simplest way to make any basic single bit set-reset SR flip-flop is to connect together a pair of cross-coupled 2-input NAND gates as shown, to form a Set-Reset Bistable also known as an active LOW SR NAND Gate Latch, so that there is feedback from each output to one of the other NAND gate inputs. This device consists of two inputs, one called the *Set*, S and the other called the *Reset*, R with two corresponding outputs Q and its inverse or complement Q (not-Q) as shown below.

The Basic SR Flip-flop



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.

State	S	R	Q	Q	Description
Set	1	0	0	1	Set Q » 1
	1	1	0	1	no change
Reset	0	1	1	0	Reset 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.

3.2 Triggering

This means making a circuit active. Making a circuit active means allowing the circuit to take input and give output. Like for example supposed we have a flip-flop. When the circuit is not triggered, even if you give some input data, it will not change the data stored inside the flip-flop nor will it change the output Q or Q'. The triggering is given in form of a clock pulse or gating signal.

1. Level Triggering:

In level triggering the circuit will become active when the gating or clock pulse is on a particular level. This level is decided by the designer. We can have a negative level triggering in which the circuit is active when the clock signal is low or a positive level triggering in which the circuit is active when the clock signal is high.

2. Edge Triggering:

In edge triggering the circuit becomes active at negative or positive edge of the clock signal. For example if the circuit is positive edge triggered, it will take input at exactly the time in which the clock signal goes from low to high. Similarly input is taken at exactly the time in which the clock signal goes from high to low in negative edge triggering. But keep in mind after the the input, it can be processed in all the time till the next input is taken.

Difference between Level Triggered and Edge Triggered Level Trigger:

1) The input signal is sampled when the clock signal is either HIGH or LOW.

2) It is sensitive to Glitches.

Example: Latch. Edge Trigger:

1) The input signal is sampled at the RISING EDGE or FALLING EDGE of the clock signal.

2) It is not-sensitive to Glitches. Example: Flipflop.

3.3 Counters

A counter is a register capable of counting the number of clock pulses arriving at its clock input. Count represents the number of clock pulses arrived. A counter that follows a binary number is called binary counter. There are two types of counters 1. Asyncronous Counter 2. Syncronous Counter

Asynchronous or ripple counters

The logic diagram of a 2-bit ripple up counter is shown in figure. The toggle (T) flipflop are being used. But we can use the JK flip-flop also with J and K connected permanently to logic 1. External clock is applied to the clock input of flip-flop A and QA output is applied to the clock input of the next flip-flop i.e. FF-B.

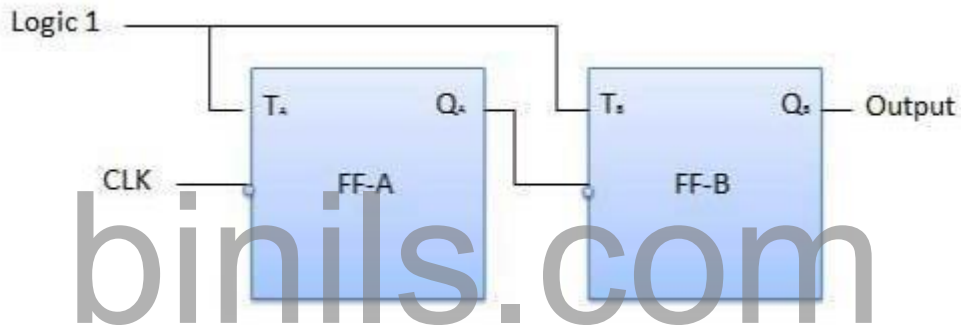


Fig 3.2 Counter Logic Diagram

S.N. Condition Operation 1 Initially let both the FFs be in the reset state QBQA = 00 initially 2 After 1st negative clock edge As soon as the first negative clock edge is applied, FF-A will toggle and QA will be equal to 1. QA is connected to clock input of FF-B. Since QA has changed from 0 to 1, it is treated as the positive clock edge by FF-B. There is no change in QBbecause FF-B is a negative edge triggered FF. QBQA = 01 after the first clock pulse. 3 After 2nd negative clock edge On the arrival of second negative clock edge, FFA toggles again and QA = 0. The change in QA acts as a negative clock edge for FF-B. So it will also toggle, and QBwill be 1. QBQA = 10 after the second clock pulse. 4 After 3rd negative clock edge On the arrival of 3rd negative clock edge, FF-A toggles again and QA become 1 from 0. Since this is a positive going change, FF-B does not respond to it and remains inactive. So QB does not change and continues to be equal to 1. QBQA = 11 after the third clock pulse. 5 After 4th negative clock edge On the arrival of 4th negative

clock edge, FF-A toggles again and QA becomes 1 from 0. This negative change in QAacts as clock pulse for FF-B. Hence it toggles to change QBfrom 1 to 0. QBQA = 00 after the fourth clock pulse.

Clock	Counter output		State number	Deciimal Counter output
	Q_B	Q_A		
Initially	0	0	—	0
1st	0	1	1	1
2nd	1	0	2	2
3rd	1	1	3	3
4th	0	0	4	0

Fig 3.3.2 Truth Table

Synchronous counters

If the "clock" pulses are applied to all the flip-flops in a counter simultaneously, then such a counter is called as synchronous counter.

2-bit Synchronous up counter

The JA and KA inputs of FF-A are tied to logic 1. So FF-A will work as a toggle flip-flop. The JB and KB inputs are connected to QA.

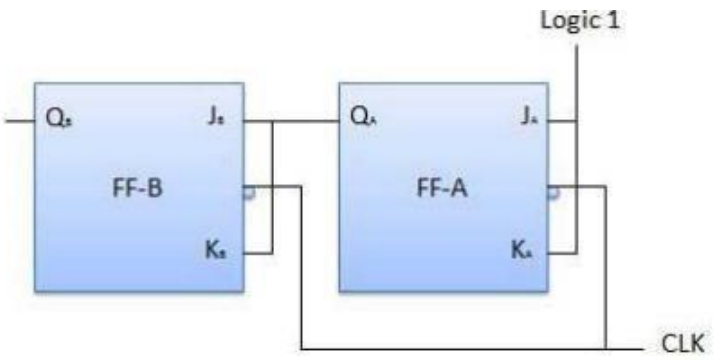


Fig 3.3.3 Synchronous UP counter

1 Initially let both the FFs be in the reset state QBQA = 00 initially. 2 After 1st negative clock edge As soon as the first negative clock edge is applied, FF-A will toggle and QA will change from 0 to 1. But at the instant of application of negative clock edge, QA , JB = KB = 0. Hence FF-B will not change its state. So QB will remain 0. QBQA = 01 after the first clock pulse. 3 After 2nd negative clock edge On the arrival of second negative clock edge, FF-A toggles again and QA changes from 1 to 0. But at this instant QA was 1. So JB = KB= 1 and FFB will toggle. Hence QB changes from 0 to 1. QBQA = 10 after the second clock pulse. 4 After 3rd negative clock edge On application of the third falling clock edge, FFA will toggle from 0 to 1 but there is no change of state for FF-B. QBQA = 11 after the third clock pulse. 5 After 4th negative clock edge On application of the next clock pulse, QA will change from 1 to 0 as QB will also change from 1 to 0. QBQA = 00 after the fourth clock pulse.

3.4 Modulo Counter

Divide by N Counter

A counter, whic is reset (makes the whole output as zeros) at the nth clock pulse is called 'mod n counter' or divide b nthcounter'. Logic gates that are connected externally, make to reset the counter at the nth clock pulse. An extra NAND gate is generally used for making divide by n counters. For this, which flip-flops output are actually is high (1) level in the nthclock pulse. Then connect the outputs of that flip-flops to the input terminals of the NAND gate. The output terminal of the NAND gate. The output terminal of the NAND gate is connected to the reset terminal of the counter. Thus at the nth clock pulse, all inputs connected to the NAND gate are high, makes the output as 0, and to reset the counter. After applying the next clock pulse the counter will start the count from its initial level. The number of flip-flops needed, for this counter depends upon the 'n' value. A three bit counter, counts upto 7 (=23-1). Similarly an 'n' bit counter, counts up to 2n-1 value.

Modulo 5 Counter

A counter which is reset at the fifth clock pulse is called Mod 5 counter or Divide by 5 counter. The circuit diagram of Mod 5 counter. This counter contains three JKMS flipflop.

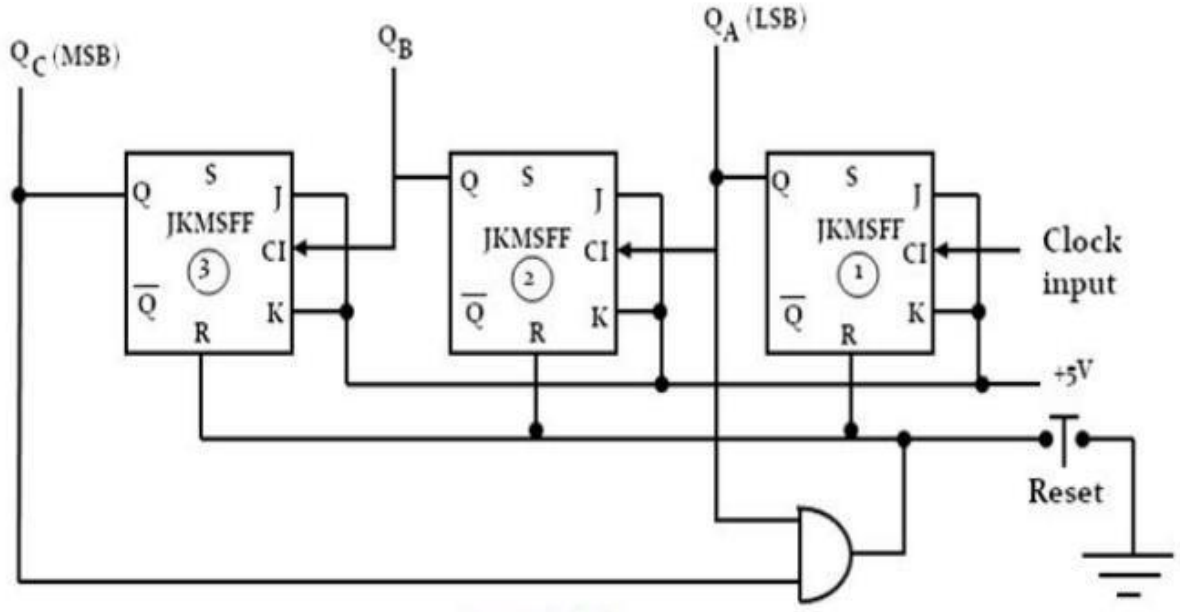


Fig 3.4.1 Modulo 5 Counter

Clock	Q_C	Q_B	Q_A
Reset	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	0	0	0
6	0	0	1

Fig 3.4.2 Truth Table Modulo 5 Counter

- A 3 bit binary counter is normally counting from 000 to 111. The actual output of a 3 bit binary counter at the fifth clock pulse is 101.

- A two input NAND gate is used to make a Mod 5 counter.

- The outputs of the first and third flip flops (QA and QC) are connected to the input of the give NAND gate, and its output is connected to the RESET terminal of the counter,

- Hence the counter is reset at the fifth clock pulse, which produces the output QC,QB,QA as 000. It is called divide by 5thcounter or mod 5 counter.

3.5 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).

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

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 QA to QD are at logic level “0” ie, no parallel data output.

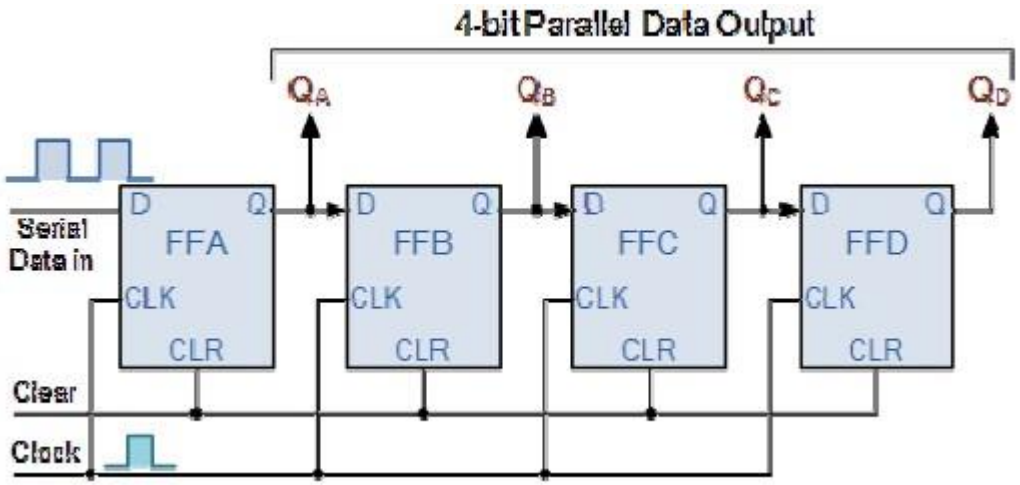


Fig 3.5.1 Serial-in to Parallel-out Shift Register

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 QA 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 QB HIGH to logic “1” as its input D has the logic “1” level on it from QA. The logic “1” has now moved or been “shifted” one place along the register to the right as it is now at QA. When the third clock pulse arrives this logic “1” value moves to the output of FFC (QC) and so on until the arrival of the fifth clock pulse which sets all the outputs QA to QD 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 QA to QD. 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.

Clock Pulse No	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0

Fig 3.5.2 Truth table

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 QA to QD, 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.

3.6 State Tables and State Diagrams

In this model the effect of all previous inputs on the outputs is represented by a state of the circuit. Thus, the output of the circuit at any time depends upon its current state and the input. These also determine the next state of the circuit. The relationship that exists among the inputs, outputs, present states and next states can be specified by either the state table or the state diagram. State Table The state table representation of a sequential circuit consists of three sections labelled present state, next state and output. The present state designates the state of flip-flops before the occurrence of a clock pulse. The next state shows the states of flip-flops after the clock pulse, and the output section lists the value of the output variables during the present state. State Diagram In addition to graphical symbols, tables or equations, flip-flops can also be represented graphically by a state diagram. In this diagram, a state is represented by a circle, and the transition between states is indicated by directed lines (or arcs) connecting the circles. An example of a state diagram is shown in Figure 3.6.1 below

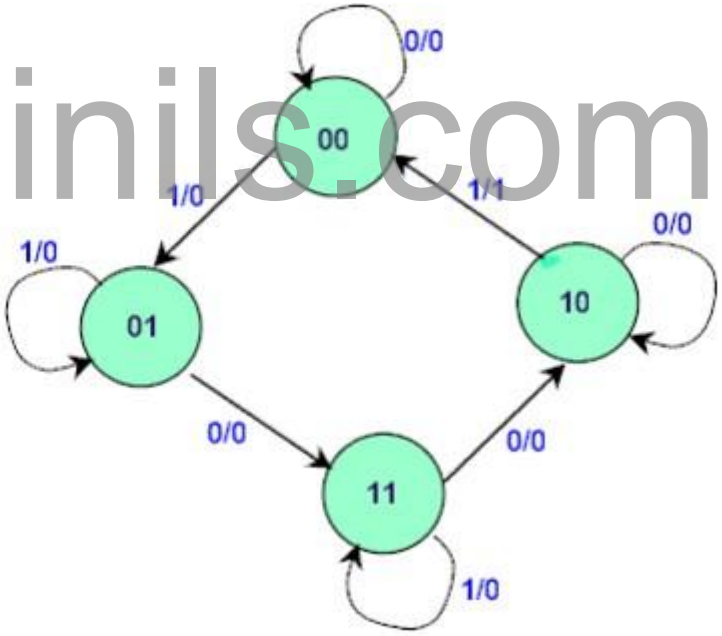


Fig 3.6.1 state diagram

The binary number inside each circle identifies the state the circle represents. The directed lines are labelled with two binary numbers separated by a slash (/). The input value that causes the state transition is labelled first. The number after the slash symbol / gives the value of the output. For example, the directed line from state 00 to 01 is labelled 1/0, meaning that, if the sequential circuit is in a present state and the input is 1, then the next state is 01

and the output is 0. If it is in a present state 00 and the input is 0, it will remain in that state. A directed line connecting a circle with itself indicates that no change of state occurs. The state diagram provides exactly the same information as the state table and is obtained directly from the state table. Consider a sequential circuit shown in Figure 4. It has one input x, one output Z and twostate variables Q1Q2 (thus having four possible present states 00, 01, 10, 11).

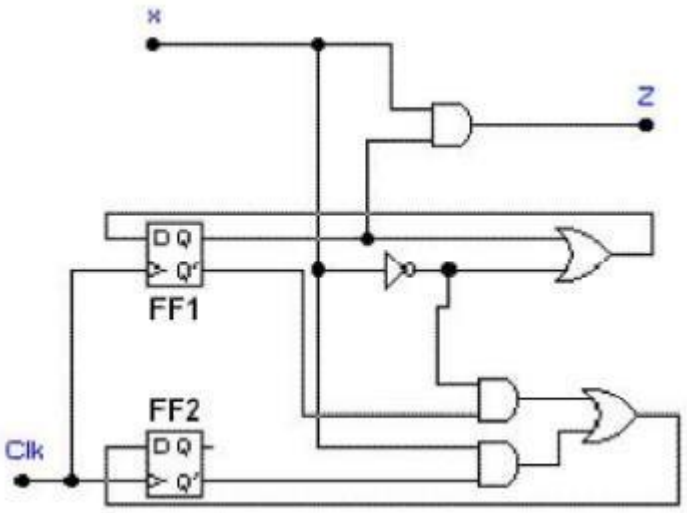


Fig 3.6.2 Sequential Circuit

The behaviour of the circuit is determined by the following Boolean expressions: Z = x*Q1 D1 = x' + Q1 D2 = x*Q2' + x'*Q1' These equations can be used to form the state table. Suppose the present state (i.e. Q1Q2) = 00 and input x = 0. Under these conditions, we get Z = 0, D1 = 1, and D2 = 1.

Thus the next state of the circuit D1D2 = 11, and this will be the present state after the clock pulse has been applied. The output of the circuit corresponding to the present state Q1Q2 = 00 and x = 1 is Z = 0. This data is entered into the state table as shown in Table 3.6.1.

Present State Q1Q2	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0 0	1 1	0 1	0	0
0 1	1 1	0 0	0	0
1 0	1 0	1 1	0	1
1 1	1 0	1 0	0	1

Table 3.6.1 State table

3.7 Mealy Machine

A Mealy Machine is an FSM whose output depends on the present state as well as the present input. It can be described by a 6 tuple (Q, ∑, O, δ, X, q0) where − • Q is a finite set of states. • ∑ is a finite set of symbols called the input alphabet. • O is a finite set of symbols called the output alphabet. • δ is the input transition function where δ: Q × ∑ → Q • X is the output transition function where X: Q × ∑ → O • q0 is the initial state from where any input is processed (q0 ∈ Q). The state table of a Mealy Machine is shown below ,

Present state	Next state			
	input = 0		input = 1	
	State	Output	State	Output
→ a	b	x_1	c	x_1
b	b	x_2	d	x_3
c	d	x_3	c	x_1
d	d	x_3	d	x_2

Table 3.7.1 State Table

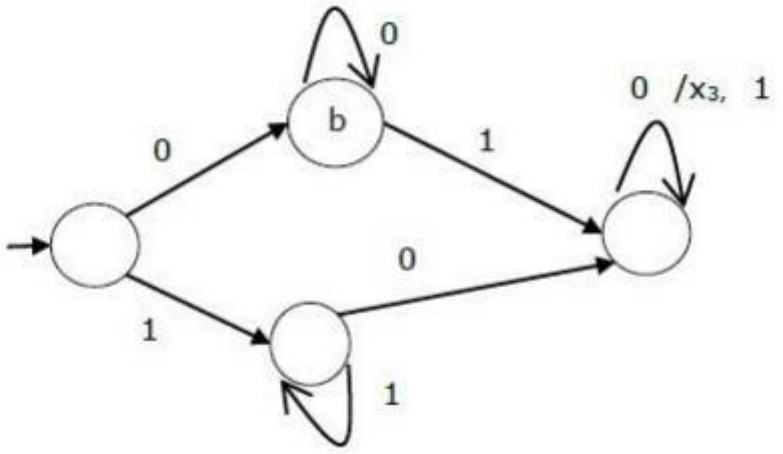


Fig 3.7.1 State Diagram

Moore Machine

Moore machine is an FSM whose outputs depend on only the present state. A Moore machine can be described by a 6 tuple (Q, ∑, O, δ, X, q0) where − • Q is a finite set of states. • ∑ is a finite set of symbols called the input alphabet. • O is a finite set of symbols called the output alphabet. • δ is the input transition function where δ: Q × ∑ → Q • X is the output transition function where X: Q → O • q0 is the initial state from where any input is processed (q0 ∈ Q). The state table of a Moore Machine is shown below ,

Present state	Next State		Output
	Input = 0	Input = 1	
→ a	b	c	x_2
b	b	d	x_1
c	c	d	x_2

Table 3.7.2 State Table Moore

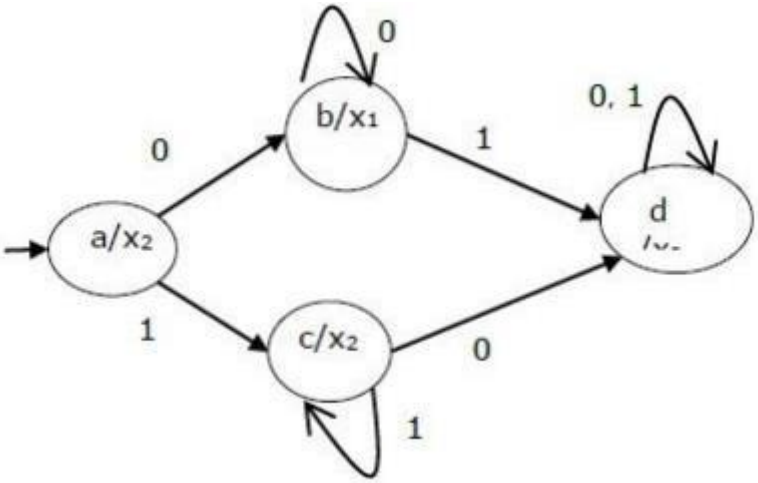


Fig 3.7.2 state diagram Moore Machine