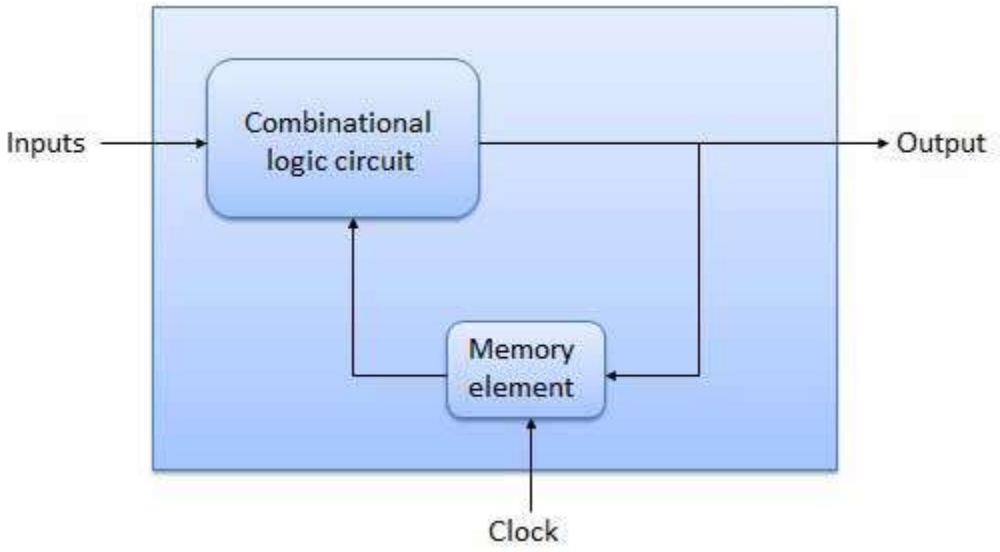
Sequential Circuits

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

Block diagram



Flip Flop

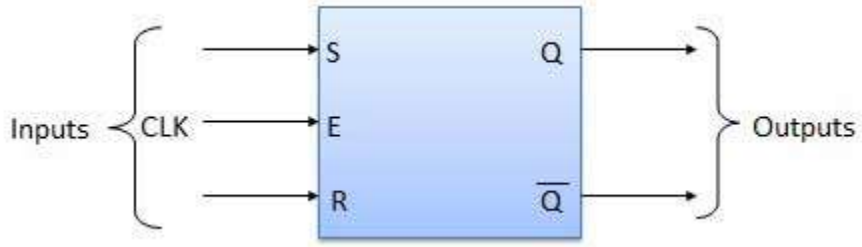
Flip flop is a sequential circuit which generally samples its inputs and changes its outputs only at particular instants of time and not continuously. Flip flop is said to be edge sensitive or edge triggered rather than being level triggered like latches.

S-R Flip Flop

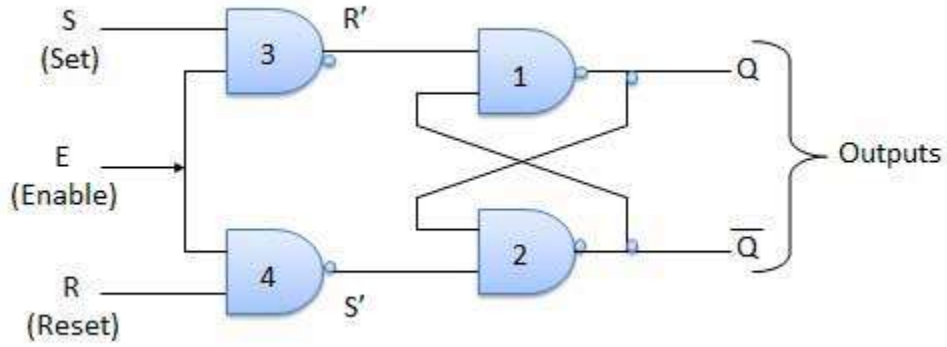
It is basically S-R latch using NAND gates with an additional **enable** input. It is also called as level triggered SR-FF. For this circuit in output will take place if and only if the enable input (E) is made active. In short this circuit will operate as an S-R latch if E= 1 but there is no change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

Prepared by:- Anwar Bari

Block Diagram



Circuit Diagram



Truth Table

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

Prepared by:- Anwar Bari

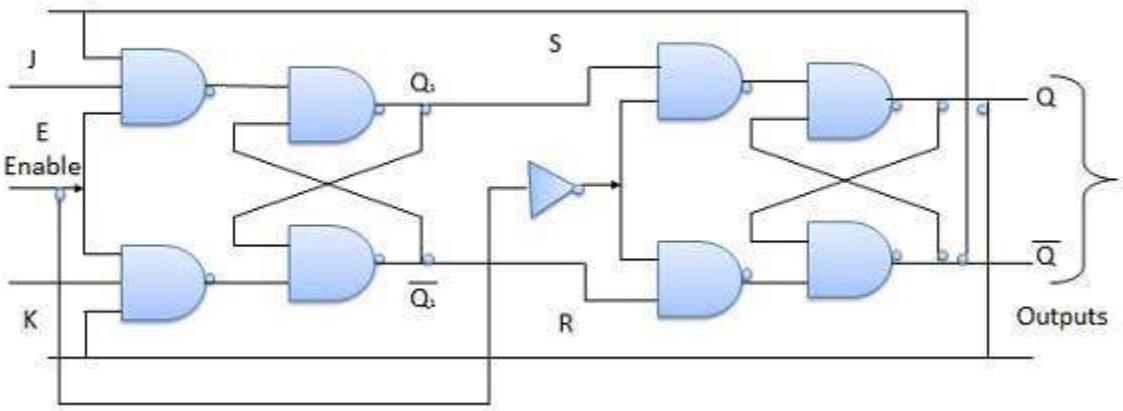
Operation

S.N.	Condition	Operation
1	**S = R = 0 : No change**	• If S = R = 0 then output of NAND gates 3 and 4 are forced to become 1. • Hence R' and S' both will be equal to 1. Since S' and R' are the input of the basic S-R latch using NAND gates, there will be no change in the state of outputs.
2	**S = 0, R = 1, E = 1**	• Since S = 0, output of NAND-3 i.e. R' = 1 and E = 1 the output of NAND-4 i.e. S' = 0. • Hence Q_{n+1} = 0 and Q_{n+1} bar = 1. This is reset condition.
3	**S = 1, R = 0, E = 1**	• Output of NAND-3 i.e. R' = 0 and output of NAND-4 i.e. S' = 1. • Hence output of S-R NAND latch is Q_{n+1} = 1 and Q_{n+1} bar = 0. This is the reset condition.
4	**S = 1, R = 1, E = 1**	• As S = 1, R = 1 and E = 1, the output of NAND gates 3 and 4 both are 0 i.e. S' = R' = 0. • Hence the **Race** condition will occur in the basic NAND latch.

Master Slave JK Flip Flop

Master slave JK FF is a cascade of two S-R FF with feedback from the output of second to input of first. Master is a positive level triggered. But due to the presence of the inverter in the clock line, the slave will respond to the negative level. Hence when the clock = 1 (positive level) the master is active and the slave is inactive. Whereas when clock = 0 (low level) the slave is active and master is inactive.

Circuit Diagram



Truth Table

Inputs			Outputs		Comments
E	J	K	Q_{n+1}	$\overline{Q}_{n+1}$	
1	0	0	Q_n	$\overline{Q}_n$	No change
1	0	1	0	1	Rset
1	1	0	1	0	Set
1	1	1	$\overline{Q}_n$	Q_n	Toggle

Operation

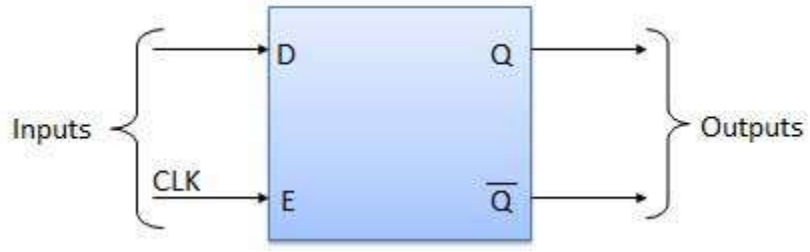
S.N.	Condition	Operation
1	**J = K = 0 (No change)**	• When clock = 0, the slave becomes active and master is inactive. But since the S and R inputs have not changed, the slave outputs will also remain unchanged. Therefore outputs will not change if J = K =0.
2	**J = 0 and K = 1 (Reset)**	• Clock = 1: Master active, slave inactive. Therefore outputs of the master become Q_1 = 0 and Q_1 bar = 1. That means S = 0 and R =1. • Clock = 0: Slave active, master inactive Therefore outputs of the slave become Q = 0 and Q bar = 1.

		• Again clock = 1: Master active, slave inactive. Therefore even with the changed outputs Q = 0 and Q bar = 1 fed back to master, its outputs will Q1 = 0 and Q1 bar = 1. That means S = 0 and R = 1. • Hence with clock = 0 and slave becoming active the outputs of slave will remain Q = 0 and Q bar = 1. Thus we get a stable output from the Master slave.
3	**J = 1 and K = 0 (Set)**	• Clock = 1: Master active, slave inactive. Therefore outputs of the master become Q_1 = 1 and Q_1 bar = 0. That means S = 1 and R =0. • Clock = 0: Slave active, master inactive Therefore outputs of the slave become Q = 1 and Q bar = 0. • Again clock = 1: then it can be shown that the outputs of the slave are stabilized to Q = 1 and Q bar = 0.
4	**J = K = 1 (Toggle)**	• Clock = 1: Master active, slave inactive. Outputs of master will toggle. So S and R also will be inverted. • Clock = 0: Slave active, master inactive. Outputs of slave will toggle. • These changed output are returned back to the master inputs. But since clock = 0, the master is still inactive. So it does not respond to these changed outputs. This avoids the multiple toggling which leads to the race around condition. The master slave flip flop will avoid the race around condition.

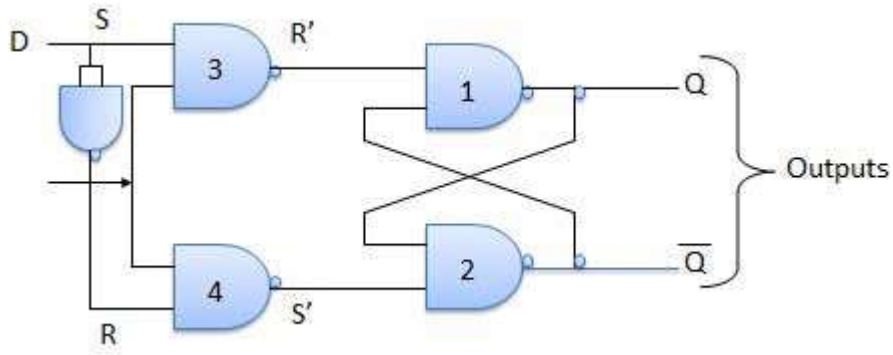
Delay Flip Flop / D Flip Flop

Delay Flip Flop or D Flip Flop is the simple gated S-R latch with a NAND inverter connected between S and R inputs. It has only one input. The input data is appearing at the output after some time. Due to this data delay between i/p and o/p, it is called delay flip flop. S and R will be the complements of each other due to NAND inverter. Hence S = R = 0 or S = R = 1,these input condition will never appear. This problem is avoid by SR = 00 and SR = 1 conditions.

Block Diagram



Circuit Diagram



Truth Table

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

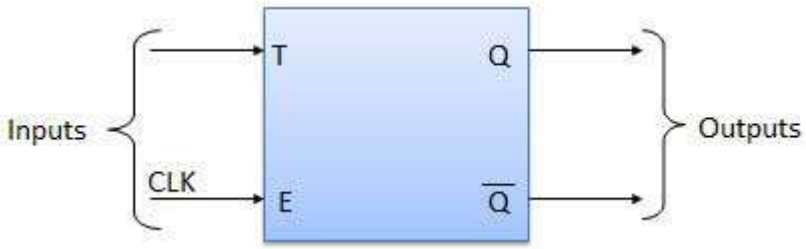
Operation

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

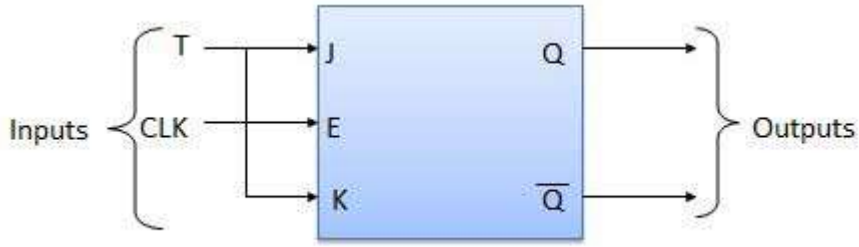
Toggle Flip Flop / T Flip Flop

Toggle flip flop is basically a JK flip flop with J and K terminals permanently connected together. It has only input denoted by **T** is shown in the Symbol Diagram. The symbol for positive edge triggered T flip flop is shown in the Block Diagram.

Symbol Diagram



Block Diagram



Truth Table

Inputs		Outputs		Comments
E	T	Q_{n+1}	$\overline{Q}_{n+1}$	
1	0	Q_n	$\overline{Q}_n$	No change
1	1	$\overline{Q}_n$	Q_n	Toggle

Operation

S.N.	Condition	Operation
1	**T = 0, J = K = 0**	• The output Q and Q bar won't change
2	**T = 1 ,J = K = 1**	• output will toggle corresponding to every leading edge of clock signal.

Digital 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.

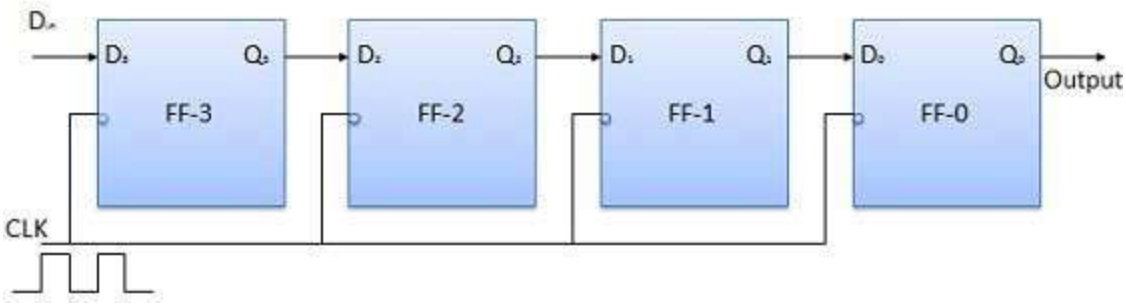
The binary data in a register can be moved within the register from one flip-flop to another. The registers that allow such data transfers are called as **shift registers**. There are four mode of opearation of a shift register.

- Serial Input Serial Output
- Serial Input Parallel Output
- Parallel Input Serial Output
- Parallel Input Parallel Output

Serial Input Serial Output

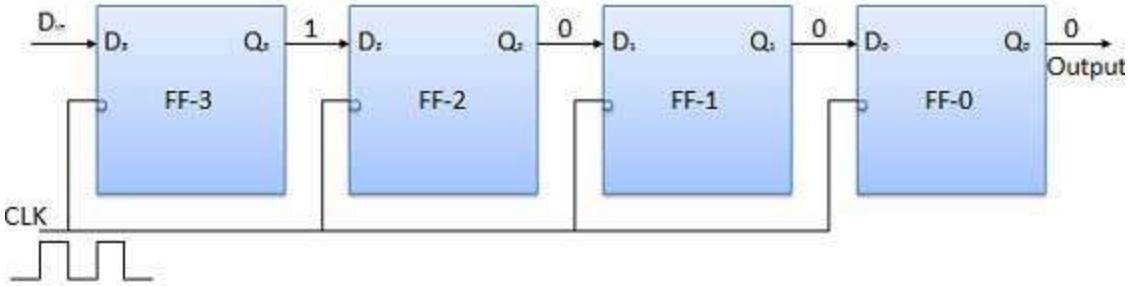
Let all the flip-flop be initially in the reset condition i.e. $Q_3 = Q_2 = Q_1 = Q_0 = 0$. If we entry of a four bit binary number 1 1 1 1 into the register. When this is to be done, this number should be applied to $\mathbf{D_{in}}$ bit by with the LSB bit applied first. The D input of FF-3 i.e. D_3 is connected to serial data input $\mathbf{D_{in}}$. Output of FF-3 i.e. Q_3 is connected to the input of the next flip-flop i.e. D_2 and so on.

Block Diagram

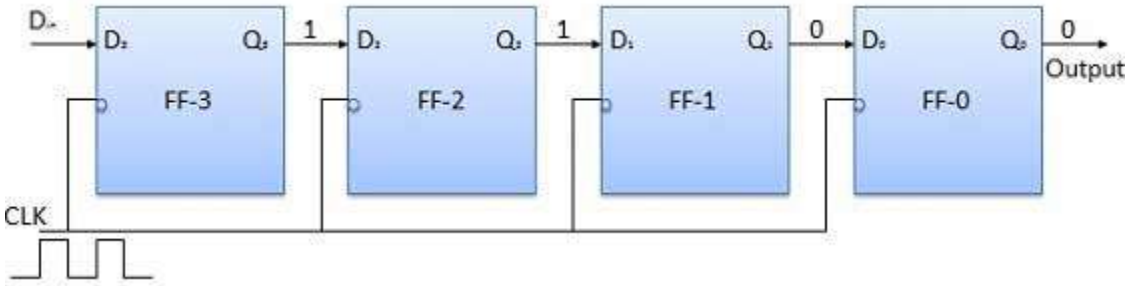


Operation

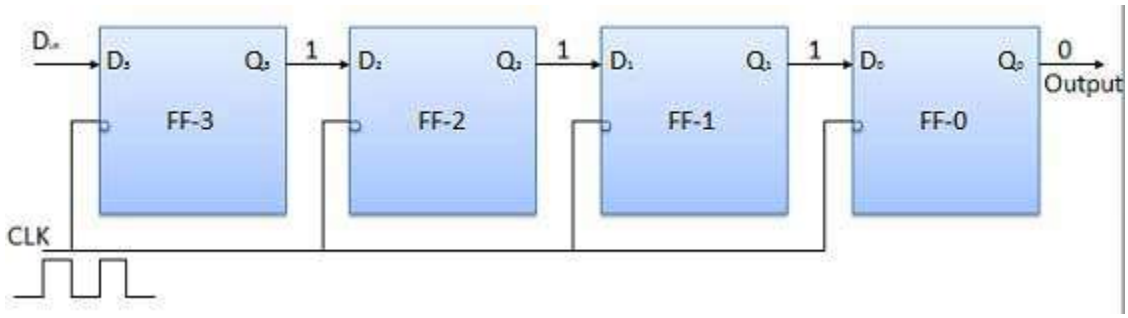
Before application of clock signal let $Q_3\ Q_2\ Q_1\ Q_0 = 0000$ and apply LSB bit of the number to be entered to D_{in}. So $D_{in}=D_3=1$. Apply the clock. On the first falling edge of clock, the FF-3 is set, and stored word in the register is $Q_3\ Q_2\ Q_1\ Q_0 = 1000$.



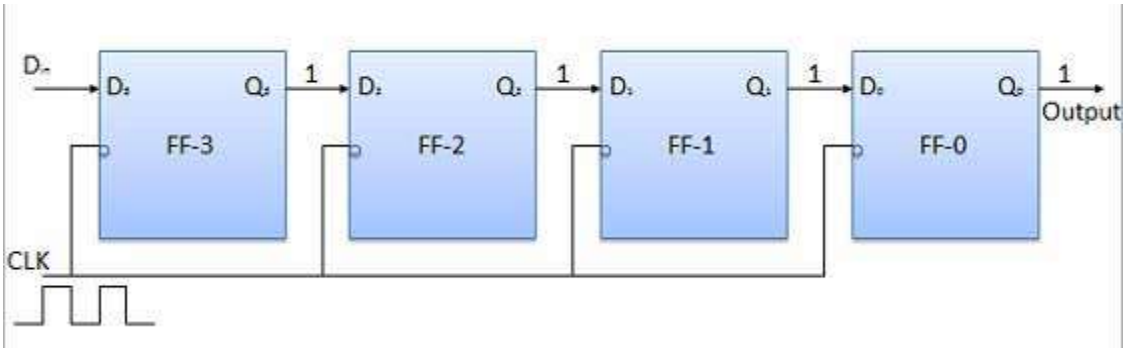
Apply the next bit to D_{in}. So $D_{in}=1$. As soon as the next negative edge of the clock hits, FF-2 will set and the stored word change to $Q_3\ Q_2\ Q_1\ Q_0 = 1100$.



Apply the next bit to be stored i.e. 1 to D_{in}. Apply the clock pulse. As soon as the third negative clock edge hits, FF-1 will be set and output will be modified to $Q_3\ Q_2\ Q_1\ Q_0 = 1110$.



Similarly with $D_{in}=1$ and with the fourth negative clock edge arriving, the stored word in the register is $Q_3\ Q_2\ Q_1\ Q_0 = 1111$.

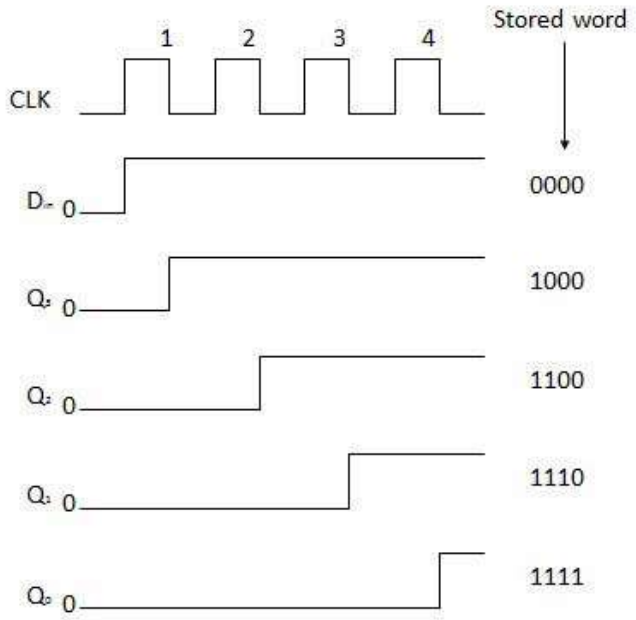


Truth Table

	CLK	$D_- = Q_3$	$Q_3 = D_2$	$Q_2 = D_1$	$Q_1 = D_0$	Q_0
Initially			0	0	0	0
(i)	↓	1	1	0	0	0
(ii)	↓	1	1	1	0	0
(iii)	↓	1	1	1	1	0
(iv)	↓	1	1	1	1	1

→ Direction of data travel

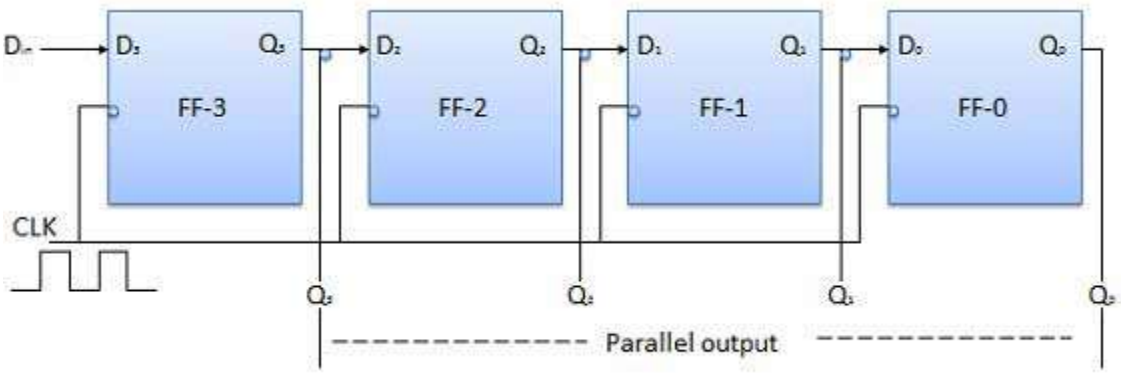
Waveforms



Serial Input Parallel Output

- In such types of operations, the data is entered serially and taken out in parallel fashion.
- Data is loaded bit by bit. The outputs are disabled as long as the data is loading.
- As soon as the data loading gets completed, all the flip-flops contain their required data, the outputs are enabled so that all the loaded data is made available over all the output lines at the same time.
- 4 clock cycles are required to load a four bit word. Hence the speed of operation of SIPO mode is same as that of SISO mode.

Block Diagram



Parallel Input Serial Output (PISO)

- Data bits are entered in parallel fashion.
- The circuit shown below is a four bit parallel input serial output register.
- Output of previous Flip Flop is connected to the input of the next one via a combinational circuit.
- The binary input word B_0,B_1,B_2,B_3 is applied though the same combinational circuit.
- There are two modes in which this circuit can work namely shift mode or load mode.

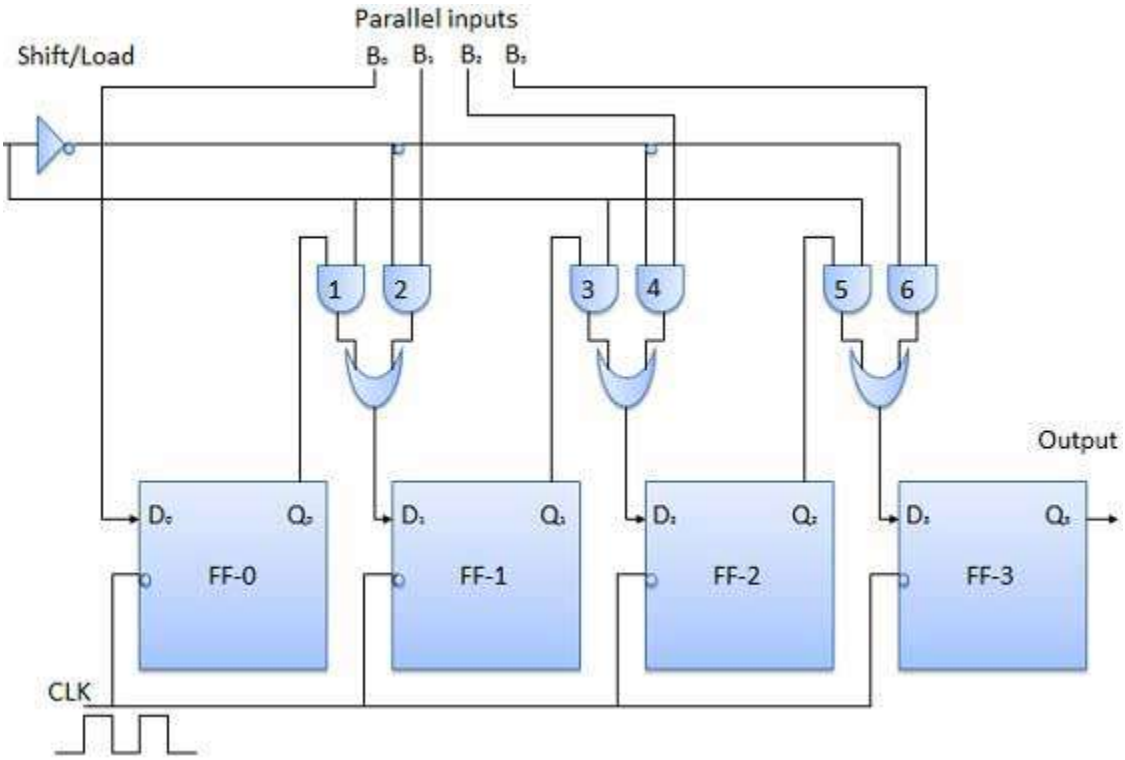
Load mode

When the shift/load bar line is low (0), the AND gate 2,4 and 6 become active. They will pass B_1,B_2,B_3 bits to the corresponding flip-flops. On the low going edge of clock, the binary input B_0,B_1,B_2,B_3 will get loaded into the corresponding flip-flops. Thus parallel loading takes place.

Shift mode

When the shift/load bar line is low (1), the AND gate 2,4 and 6 become inactive. Hence the parallel loading of the data becomes impossible. But the AND gate 1,3 and 5 become active. Therefore the shifting of data from left to right bit by bit on application of clock pulses. Thus the parallel in serial out operation take place.

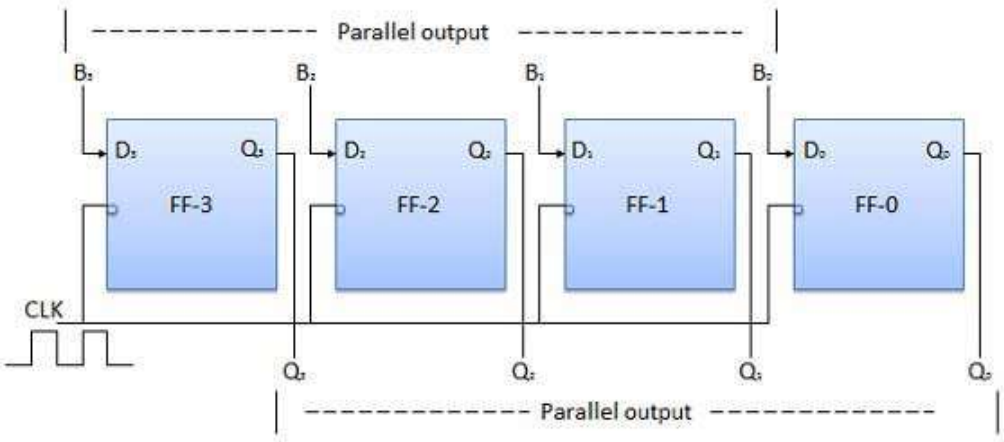
Block Diagram



Parallel Input Parallel Output (PIPO)

In this mode, the 4 bit binary input B_0,B_1,B_2,B_3 is applied to the data inputs D_0,D_1,D_2,D_3 respectively of the four flip-flops. As soon as a negative clock edge is applied, the input binary bits will be loaded into the flip-flops simultaneously. The loaded bits will appear simultaneously to the output side. Only clock pulse is essential to load all the bits.

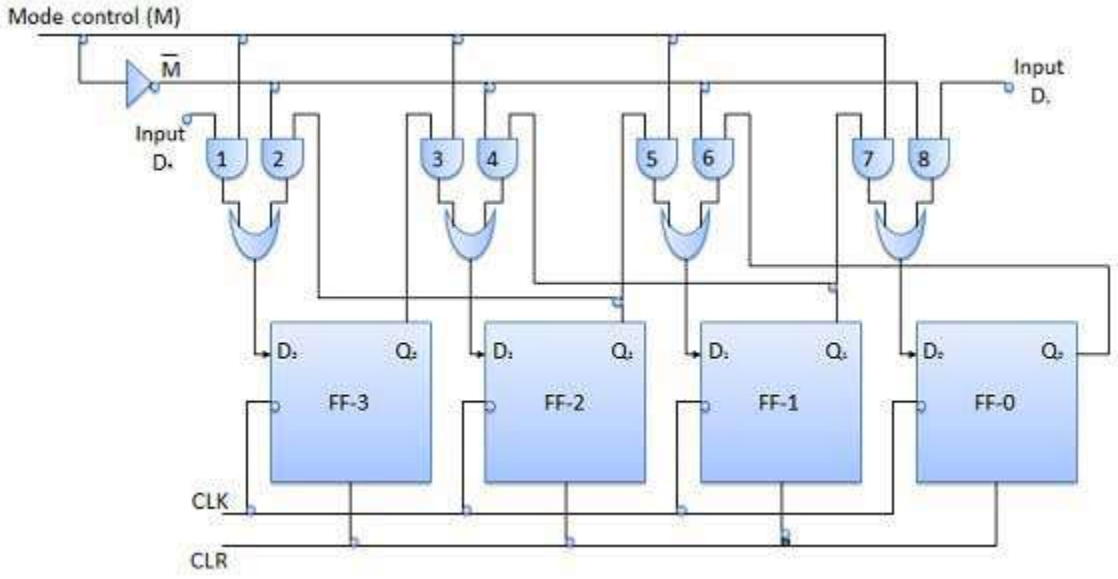
Block Diagram



Bidirectional Shift Register

- If a binary number is shifted left by one position then it is equivalent to multiplying the original number by 2. Similarly if a binary number is shifted right by one position then it is equivalent to dividing the original number by 2.
- Hence if we want to use the shift register to multiply and divide the given binary number, then we should be able to move the data in either left or right direction.
- Such a register is called as a bi-directional register. A four bit bi-directional shift register is shown in fig.
- There are two serial inputs namely the serial right shift data input DR and the serial left shift data input DL along with a mode select input (M).

Block Diagram



Operation

S.N.	Condition	Operation
1	**With M = 1 : Shift right operation**	• If M = 1, then the AND gates 1,3,5 and 7 are enable whereas the remaining AND gates 2,4,6 and 8 will be disabled. • The data at D_R is shifted to right bit by bit from FF-3 to FF-0 on the application of clock pulses. Thus with M = 1 we get the serial right shift operation.
2	**With M = 0 : Shift left operation**	• When the mode control M is connected to 0 then the AND gates 2,4,6 and 8 are enabled while 1,3,5 and 7 are disabled. • The data at D_L is shifted left bit by bit from FF-0 to FF-3 on the application of

		clock pulses. Thus with M = 0 we get the serial right shift operation.

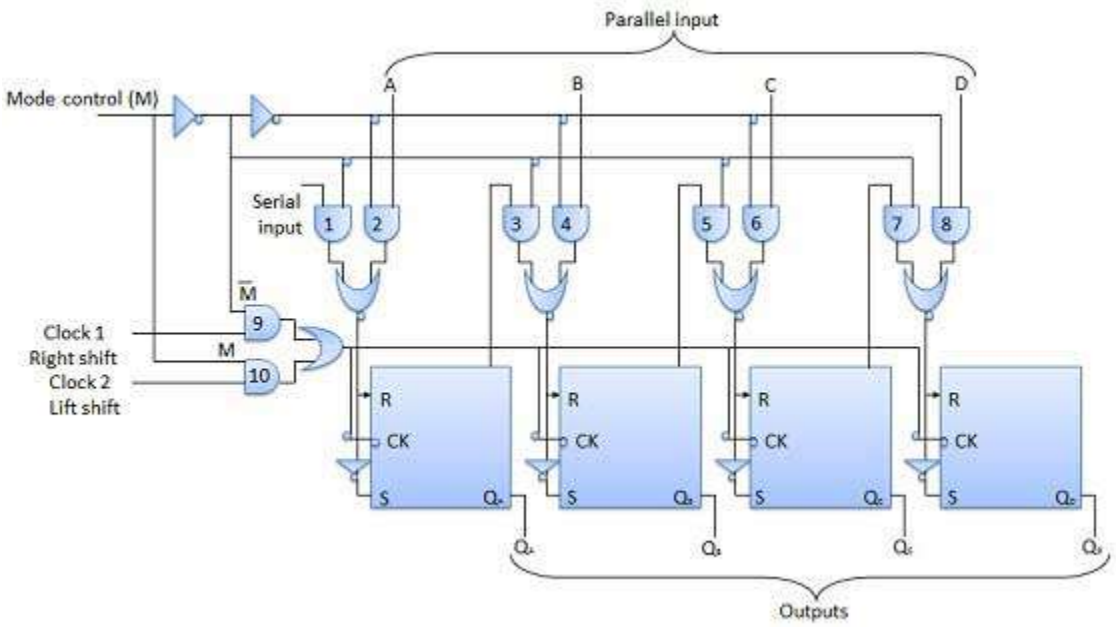
Universal Shift Register

A shift register which can shift the data in only one direction is called a uni-directional shift register. A shift register which can shift the data in both directions is called a bi-directional shift register. Applying the same logic, a shift register which can shift the data in both directions as well as load it parallely, then it is known as a universal shift register. The shift register is capable of performing the following operation

- Parallel loading
- Lift shifting
- Right shifting

The mode control input is connected to logic 1 for parallel loading operation whereas it is connected to 0 for serial shifting. With mode control pin connected to ground, the universal shift register acts as a bi-directional register. For serial left operation, the input is applied to the serial input which goes to AND gate-1 shown in figure. Whereas for the shift right operation, the serial input is applied to D input.

Block Diagram



Digital Counters

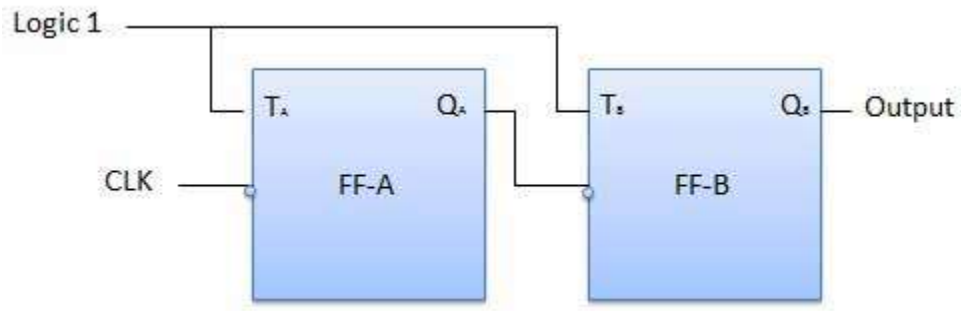
Counter is a sequential circuit. A digital circuit which is used for a counting pulses is known counter. Counter is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied. Counters are of two types.

- Asynchronous or ripple counters
- Synchronous counters.

Asynchronous or ripple counters

The logic diagram of a 2-bit ripple up counter is shown in figure. The toggle(T) flip-flop 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 Q_A output is applied to the clock input of the next flip-flop i.e. FF-B.

Logical Diagram



Operation

S.N.	Condition	Operation
1	**Initially let both the FFs be in the reset state**	Q_BQ_A = 00.....initially
2	**After 1st negative clock edge**	• As soon as the first negative clock edge is applied, FF-A will toggle and Q_A will be equal to 1. • Q_A is connected to clock input of FF-B. Since Q_A has changed from 0 to 1, it is treated as the positive clock edge by FF-B. There is no change in Q_B because FF-B is a negative edge triggered FF.

		Q_BQ_A = 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 Q_A = 0. • The change in Q_A acts as a negative clock edge for FF-B. So it will also toggle, and Q_B will be 1. Q_BQ_A = 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 Q_A become 1 from 0. • Since this is a positive going change,FF-B does not respond to it and remains inactive. So Q_B does not change and continues to be equal to 1. Q_BQ_A = 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 Q_A become 1 from 0. • This negative change in Q_A acts as clock pulse for FF-B. Hence it toggles to change Q_B from 1 to 0. Q_BQ_A = 00.....After the fourth clock pulse

Truth Table

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

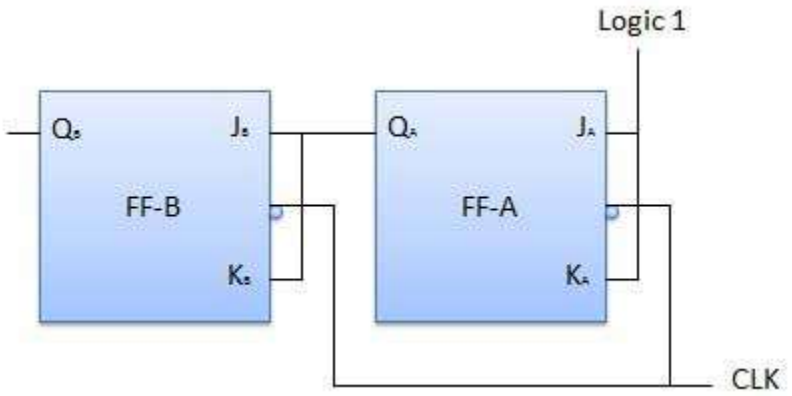
Synchronous counters

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 J_A and K_A inputs of FF-A are tied to logic 1. So FF-A will work as a toggle flip-flop. The J_B and K_B inputs are connected to Q_A.

Logical Diagram



Operation

S.N.	Condition	Operation
1	**Initially let both the FFs be in the reset state**	Q_BQ_A = 00.....initially
2	**After 1st negative clock edge**	• As soon as the first negative clock edge is applied, FF-A will toggle and Q_A will change from 0 to 1. • But at the instant of application of negative clock edge, Q_A ,J_B = K_B =0 Hence FF-B will not change its state. So Q_B will remain 0. Q_BQ_A = 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 Q_A change from 1 to 0. • But at this instant Q_A was 1. So J_B = K_B=1 and FF-B will toggle.

		Hence Q_B changes from 0 to 1. Q_BQ_A = 10.....After the second clock pulse
4	**After 3rd negative clock edge**	• On application of the third falling clock edge, FF-A will toggle from 0 to 1 but there is no change of state for FF-B. Q_BQ_A = 11.....After the third clock pulse
5	**After 4th negative clock edge**	• On application of the next clock pulse, Q_A will change from 1 to 0 as Q_B will also change from 1 to 0. Q_BQ_A = 00.....After the fourth clock pulse

Classification of counters

Depending on the way in which the counting progresses, the synchronous or asynchronous counters are classified as follows.

- Up counters
- Down counters
- Up/Down counters

UP/DOWN Counter

In the up/down counter, when up counter and down counter combined together to obtain an UP/DOWN counter. A mode control (M) input is also provided to select either up or down mode. A combinational circuit is required to be designed and used between each pair of flip-flop in order to achieve the up/down operation.

- Type of up/down counters
- UP/DOWN ripple counters
- UP/DOWN synchronous counters

UP/DOWN Ripple Counters

In the UP/DOWN ripple counter all the FFs operate in the toggle mode. So either T flip-flops or JK flip-flops are to be used. The LSB flip-flop receives clock directly. But the clock to every other FF is obtained from (Q = Q bar) output of the previous FF.

- **UP counting mode (M=0)** - The Q output of the preceding FF is connected to the clock of the next stage if up counting is to be achieved. For this mode, the mode select input M is at logic 0 (M=0).

- **DOWN counting mode (M=1)** - If M =1, then the Q bar output of the preceding FF is connected to the next FF. This will operate the counter in the counting mode.

Example

3-bit binary up/down ripple counter.

- 3-bit : hence three FFs are required.

- UP/DOWN : So a mode control input is essential.

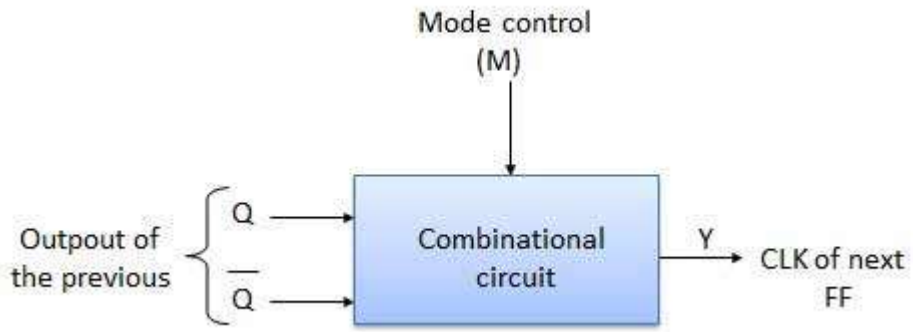
- For a ripple up counter, the Q output of preceding FF is connected to the clock input of the next one.

- For a ripple up counter, the Q output of preceding FF is connected to the clock input of the next one.

- For a ripple down counter, the Q bar output of preceding FF is connected to the clock input of the next one.

- Let the selection of Q and Q bar output of the preceding FF be controlled by the mode control input M such that, If M = 0, UP counting. So connect Q to CLK. If M = 1, DOWN counting. So connect Q bar to CLK

Block Diagram



Truth Table

Inputs			Outputs	
M	Q	$\overline{Q}$	Y	
0	0	0	0	Y = Q for up counter
0	0	1	0	
0	1	0	1	
0	1	1	1	
1	0	0	0	$Y = \overline{Q}$ for up counter
1	0	1	1	
1	1	0	0	
1	1	1	1	

Operation

S.N.	Condition	Operation
1	**Case 1: With M = 0 (Up counting mode)**	• If M = 0 and M bar = 1, then the AND gates 1 and 3 in fig. will be enabled whereas the AND gates 2 and 4 will be disabled. • Hence Q_A gets connected to the clock input of FF-B and Q_B gets connected to the clock input of FF-C. • These connections are same as those for the normal up counter. Thus with M = 0 the circuit work as an up counter.
2	**Case 2: With M = 1 (Down counting mode)**	• If M = 1, then AND gates 2 and 4 in fig. are enabled whereas the AND gates 1 and 3 are disabled. • Hence Q_A bar gets connected to the clock input of FF-B and Q_B bar gets connected to the clock input of FF-C. • These connections will produce a down counter. Thus with M = 1 the circuit works as a down counter.

Modulus Counter (MOD-N Counter)

The 2-bit ripple counter is called as MOD-4 counter and 3-bit ripple counter is called as MOD-8 counter. So in general, an n-bit ripple counter is called as modulo-N counter. Where,MOD number = 2^n

Type of modulus

- 2-bit up or down (MOD-4)

- 3-bit up or down (MOD-8)
- 4-bit up or down (MOD-16)

Application of the counters

- Frequency counters
- Digital clock
- Time measurement
- A to D converter
- Frequency divider circuits
- Digital triangular wave generator

Complement Arithmetic

Complements are used in the digital computers in order to simplify the subtraction operation and for the logical manipulations. For each radix-r system (radix r represent base of number system) there are two types of complements

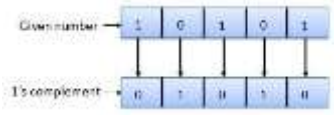
S.N.	Complement	Description
1	Radix Complement	The radix complement is referred to as the r's complement
1	Diminished Radix Complement	The diminished radix complement is referred to as the (r-1)'s complement

Binary system complements

As the binary system has base r = 2. So the two types of complements for the binary system are 2's complement and 1's complement.

1's complement

The 1's complement of a number is found by changing all 1's to 0's and all 0's to 1's. This is called as taking complement or 1's complement. Example of 1's Complement is as follows.

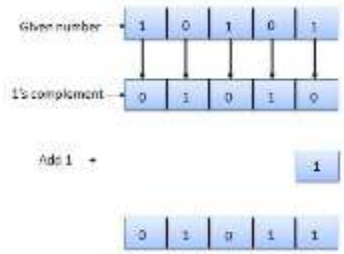


2's complement

The 2's complement of binary number is obtained by adding 1 to the Least Significant Bit (LSB) of 1's complement of the number.

2's complement = 1's complement + 1

Example of 2's Complement is as follows.



Finding the r's and (r-1)'s complement

Here we are going to learn how to convert a number to its r's and (r-1)'s complement.
Method:
Let 'N' is a number and r is its base where r>1 and in N, 'n' is the number of digits before its decimal point then we can write
r's complement of number = r^n-N

EX.

N = (23)10
here r = 10
n = 2 and N = 23

hence we can write the 10's complement of this number as 10^2 - 23 = 77.
hence we can say that 10's comp of 23 is 77.
Although this method is good enough to solve any problem regarding to this concept, but we will follow different method for finding r's and r-1's complement.

Easy Method:
Let we have to find again the 10's comp of 23 then this method tells us to divide 3 from 10 and 2 from 9 (i.e 10-9). which gives us a result of 77.

```
   9  10
-  2   3

   7   7
```

i.e the generalized form of writing a r's comp of a number 'abc' which is in r base, we can write.

```
    (r-1)  (r-1)  r
-     a      b    c
```

this difference gives us the r's comp of that number.
i.e we can find r's complement of a number by subtracting its right most digit by r and all digits by r-1.

Finding (r-1)'s complement:

We can do this easily by subtracting all the digits of that number from (r-1) where r is the base of that number.

EXAMPLES:

Find the 10's and 9's complement of (348)10.

ans:

```
   9   9   10
 - 3   4    8

   6   5    2    here 652 is 10's comp of 348
```

9's comp

```
   9   9   9
 - 3   4   8

   6   5   1    here 651 is 9's comp of 348
```

from this method you can find the r's and (r-1)'s complement of any number with base r.

Floating-Point Representation

The Institute of Electrical and Electronics Engineers (IEEE) standardizes floating-point representation in IEEE 754. Floating-point representation is similar to scientific notation in that there is a number multiplied by a base number raised to some power. For example, 118.625 is represented in scientific notation as 1.18625 x 102. The main benefit of this representation is that it provides varying degrees of precision based on the scale of the numbers that you are using. For example, it is beneficial to talk in terms of angstroms (10-10 m) when you are working with the distance between atoms. However, if you are dealing with the distance between cities, this level of precision is no longer practical or necessary.

IEEE 754 defines binary representations for 32-bit single-precision and 64-bit double-precision (64-bit) numbers as well as extended single-precision and extended double-precision numbers. Examine the specification for single-precision, floating-point numbers, also called floats.

A float consists of three parts: the sign bit, the exponent, and the mantissa. The division of the three parts is as follows:

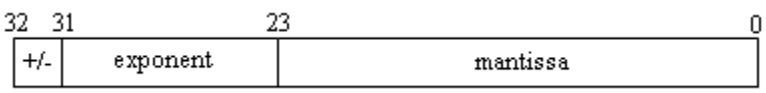


Figure 1. A float consists of three parts: the sign bit, the exponent, and the mantissa.

The sign bit is 0 if the number is positive and 1 if the number is negative.

The exponent is an 8-bit number that ranges in value from -126 to 127. The exponent is actually not the typical two's complement representation because this makes comparisons more difficult. Instead, the value is biased by adding 127 to the desired exponent and representation, which makes it possible to represent negative numbers.

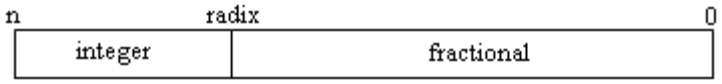
The mantissa is the normalized binary representation of the number to be multiplied by 2 raised to the power defined by the exponent.

Now look at how to encode 118.625 as a float. The number 118.625 is a positive number, so the sign bit is 0. To find the exponent and mantissa, first write the number in binary, which is 1110110.101 (get more details on finding this number in the "Fixed-Point Representation" section). Next, normalize the number to 1.110110101 x 26, which is the binary equivalent of scientific notation. The exponent is 6 and the mantissa is 1.110110101. The exponent must be biased, which is 6 + 127 = 133. The binary representation of 133 is 10000101.

Thus, the floating-point encoded value of 118.65 is 0100 0010 1111 0110 1010 0000 0000 0000. Binary values are often referred to in their hexadecimal equivalent. In this case, the hexadecimal value is 42F6A000

2. Fixed-Point Representation

In fixed-point representation, a specific radix point - called a decimal point in English and written "." - is chosen so there is a fixed number of bits to the right and a fixed number of bits to the left of the radix point. The bits to the left of the radix point are called the integer bits. The bits to the right of the radix point are called the fractional bits.



In fixed-point representation, a specific radix point - called a decimal point in English and written "." - is chosen so there is a fixed number of bits to the right and a fixed number of bits to the left of the radix point. The bits to the left of the radix point are called the integer bits. The bits to the right of the radix point are called the fractional bits.

In this example, assume a 16-bit fractional number with 8 magnitude bits and 8 radix bits, which is typically represented as 8.8 representation. Like most signed integers, fixed-point numbers are represented in two's complement binary. Using a positive number keeps this example simple.

To encode 118.625, first find the value of the integer bits. The binary representation of 118 is 01110110, so this is the upper 8 bits of the 16-bit number. The fractional part of the number is represented as 0.625 x 2n where n is the number of fractional bits. Because 0.625 x 256 = 160, you can use the binary representation of 160, which is 10100000, to determine the fractional bits. Thus, the binary representation for 118.625 is 0111 0110

1010 0000. The value is typically referred to using the hexadecimal equivalent, which is 76A0.

The major advantage of using fixed-point representation for real numbers is that fixed-point adheres to the same basic arithmetic principles as integers. Therefore, fixed-point numbers can take advantage of the general optimizations made to the Arithmetic Logic Unit (ALU) of most microprocessors, and do not require any additional libraries or any additional hardware logic. On processors without a floating-point unit (FPU), such as the Analog Devices Blackfin Processor, fixed-point representation can result in much more efficient embedded code when performing mathematically heavy operations.

In general, the disadvantage of using fixed-point numbers is that fixed-point numbers can represent only a limited range of values, so fixed-point numbers are susceptible to common numeric computational inaccuracies. For example, the range of possible values in the 8.8 notation that can be represented is +127.99609375 to -128.0. If you add 100 + 100, you exceed the valid range of the data type, which is called overflow. In most cases, the values that overflow are saturated, or truncated, so that the result is the largest representable number.