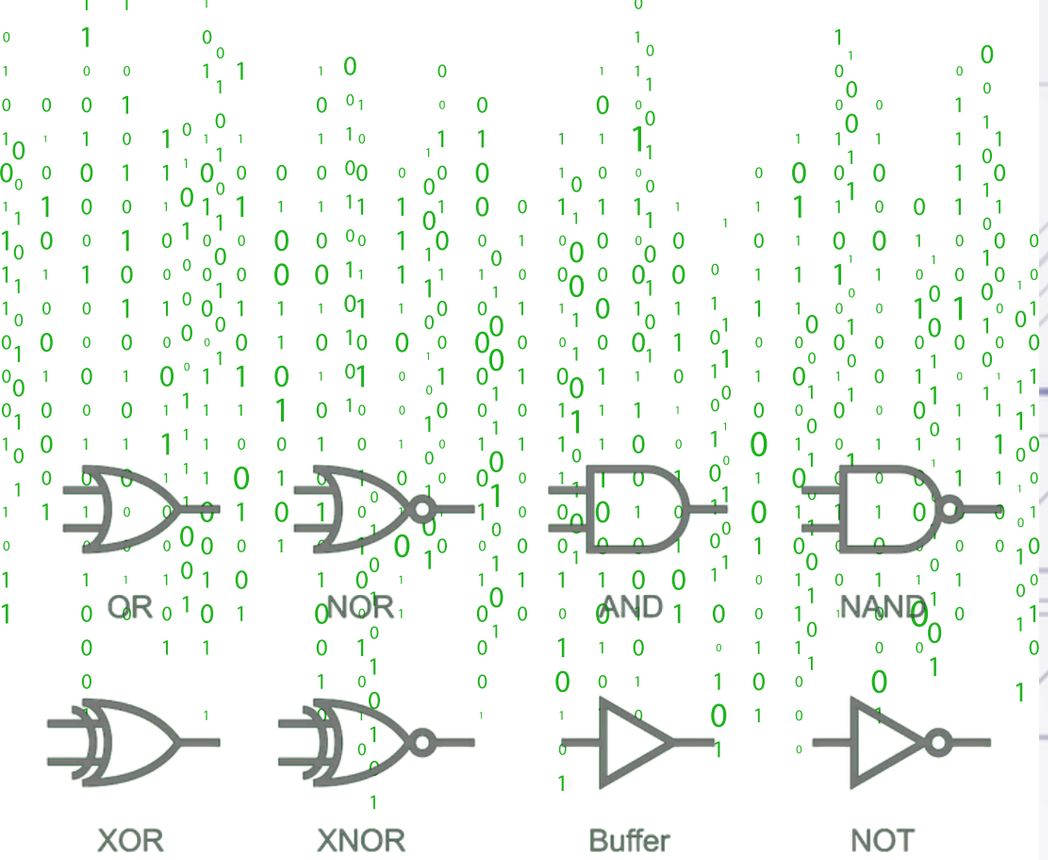


DIGITAL ELECTRONICS

NOTES

Jairaj Mirashi
Design Verification Engineer

Number Systems

Conversion form one numbers system to other

What is the decimal equivalent of hex number 0x3A?
To convert a number from a non-decimal base to a decimal base, following steps are
required:

- Start from the least significant digit, and move towards most significant digit.
- Multiply each digit with "<base> to the power of that <bit position>", i.e. <base><bit position>
- Sum the results over each digit

.

Therefore: 0x3A = [0xA * 160] + [0x3 * 161] = [10*1 + 3*16] = 58

$3A_{(16)} = (?)_{10}$

0011 1010

Write down in Binary formate

So,

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
32	16	8	4	2	1		
0	0	1	1	1	0	1	0

$= 32+16+8+2$

$58_{(10)}$

What is Gray code and what are some of the benefits of using Gray code over Binary code?

A Gray code is a binary number system in which two successive values differ only in one bit. It is also known as reflected binary code

Following table shows the Gray and Binary code for values from 0 to 7

Decimal	Binary	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

In Binary code, a transition between two values could have transition on more than two bits and this could sometimes lead to ambiguity if different bits take different time to transition. For example: transitioning from 3 to 4 in binary (011 to 100) requires all thenbits to toggle. This can lead to some intermediate values if say the three bits have different switching time. Whereas in Gray code, since only one bit changing any time, there is no possibility of any such ambiguity.

One more advantage of using Gray code is: since fewer bits are toggling in Gray code, a design using Gray code would consume less power compared to one using a binary code.

What is a parity bit and how is it computed?

A parity bit is a bit which is added at the end of the string of a binary code, and it indicates whether the number of bits having a value “one” in the string is even or odd. Accordingly, there are two variants of parity code - even parity and odd parity.
To compute parity bit, the total number of bits having a value “one” in a binary code is counted. If number of “ones” are odd and if we use an even parity, then the parity bit is set to 1 so that the total number of ones including parity bit counts to an even number. If number of “ones” are odd and if we use an odd parity, then the parity bit is set to 0 so that total number of ones including parity bit counts to an odd number.
Parity bit is computed by taking XOR of all the bits in the binary string. Parity bit is used as the simplest form of error detecting code.

For a given binary string: 111001, compute the proper odd parity bit.

The given binary string: 111001, has four “1’s”. Using an odd parity, the total number of 1’s in the binary string including the parity bit needs to be odd. Hence, the odd parity bit for this string will be 1.

What are 1's complement and 2's complement? Where are they used?

If all bits in a binary number are inverted by changing each 1 to 0 and each 0 to 1, the resulting binary number is called the 1's complement.
For example: The one's complement of a binary number 110010 is 001101

The 2's complement of a binary number is obtained by adding a 1 to the one's complement of the number.
For example: The two's complement of a binary number 110010 is 001101+1 = 001110

The two's complement of a number is used to represent signed binary numbers. It is also used for subtraction of binary numbers. The one's complement is an intermediate step to get to two's complement.

What is a BCD code and how is it different from binary code? What willbe the BCD and binary code for decimal number 27?

BCD is Binary coded decimal and is a four bit binary code that can be used to represent any decimal digit (from 0 to 9). A binary code is a binary representation of the decimal number, and the number of bits needed for a binary code would depend on the decimal number. For decimal numbers 0 to 9, both BCD and binary code would be same.
A number 27 can be represented in BCD by using four bit code for both 2 and 7.
Hence, BCD for 27 will be 0010 0111, while the binary code for 27 will be 11011

What is weighted code? Give example.

The weighted code will have a fixed weight for each position.
For example, in normal binary system, the decimal equivalent can be obtained by multiplying the position value with position weight and adding them together.

Give an example for Non-weighted code?

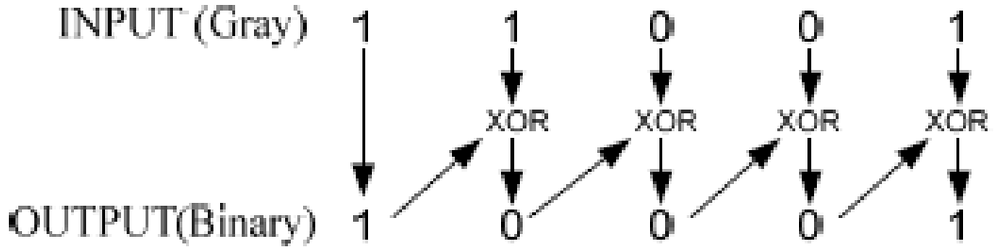
Unlike weighted code, non-weighted codes will not any weights.
For example,
Excess-3 code and Gray code. So the numbers that are represented using non-weighted code can not be directly converted to decimal equivalents.

What is the key feature of Excess-3 code?

Self-complementing: The 9's complement of an excess 3 number can be obtained simply by replacing its 1's with 0's and 0's with 1's.

Convert the Gray code number 11001 to binary code?

Conversion from gray to binary: Retain the MSB as it as. XOR the current input bit with the previous output bit to get the new output bit. In this case, given gray code number is 11001



So, the required binary number is 10001.

Basic Gates

Q Which of the following gates is called a universal gate and why?

1. AND
2. NAND
3. OR
4. NOR
5. XOR

A universal gate is a gate which can implement any Boolean function without need to use any other gate type. The NAND and NOR gates are universal gates.

How can you implement following gates using a 2:1 MUX?

a) Single Input NOT

b) Two Input AND

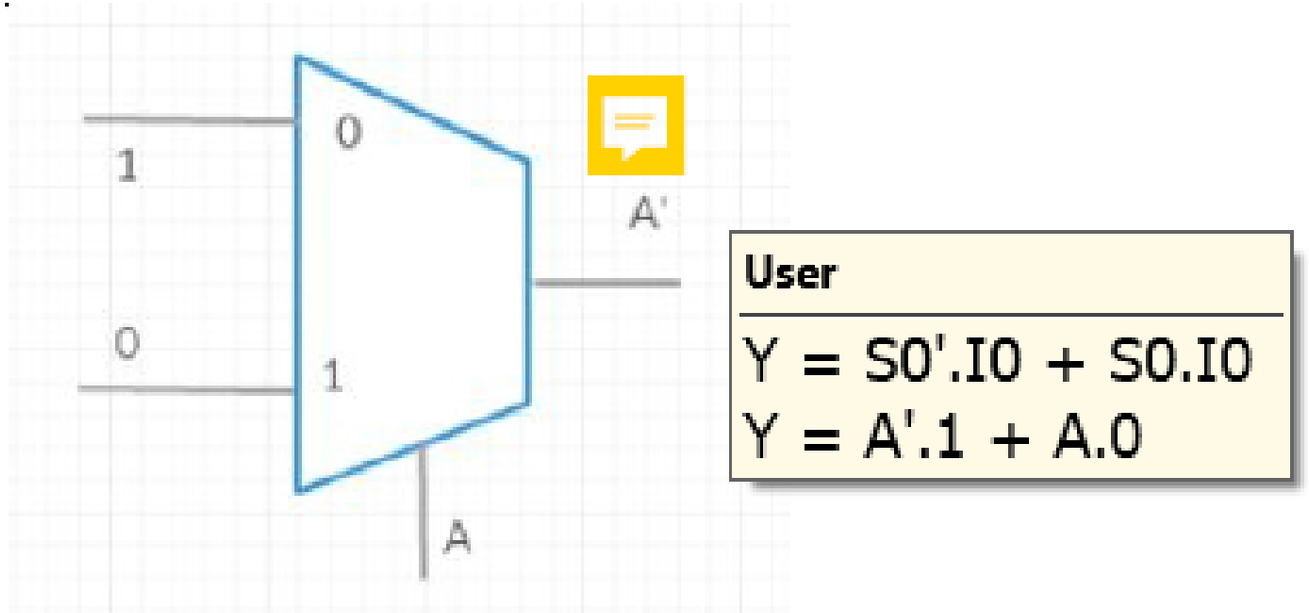
c) Two Input OR

d) Two Input NOR

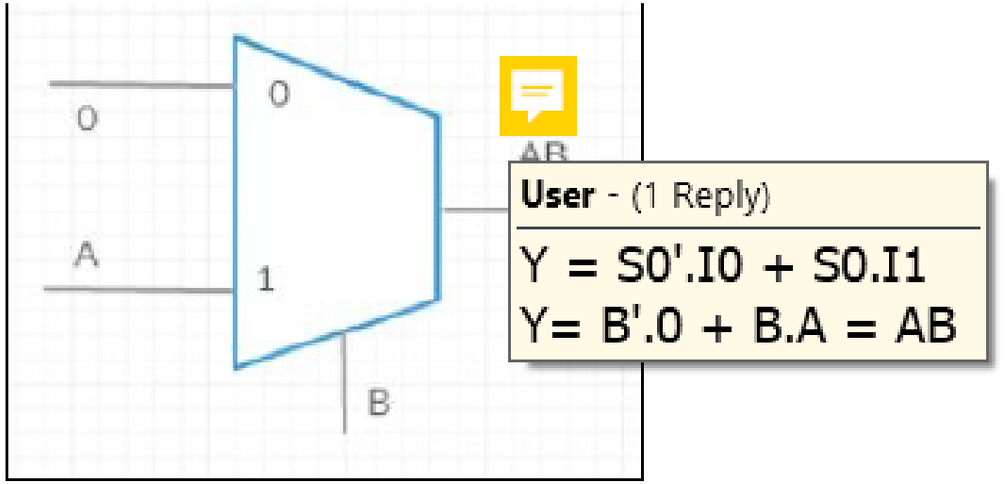
e) Two Input NAND

f) Two Input XOR

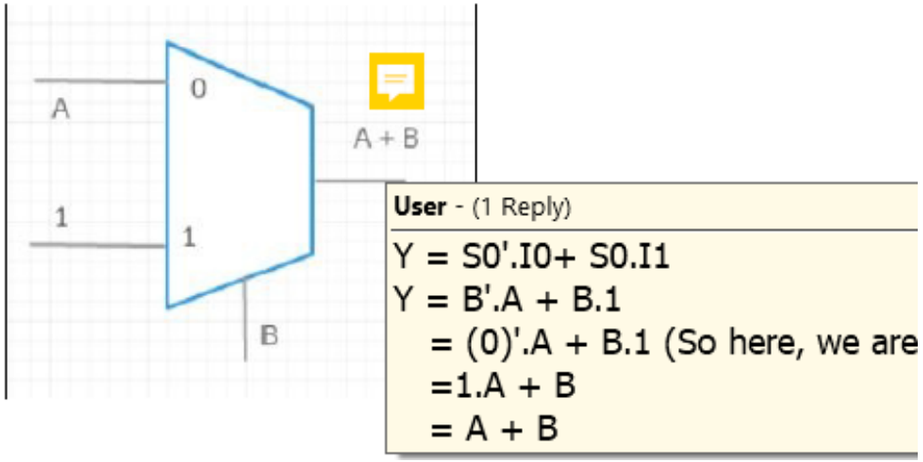
a) NOT Gate: A NOT can be implemented as shown below by connecting the input of NOT gate to select line and the inputs tied to 1 and 0 as shown below.



b) AND Gate: An AND gate can be implemented using MUX as shown below



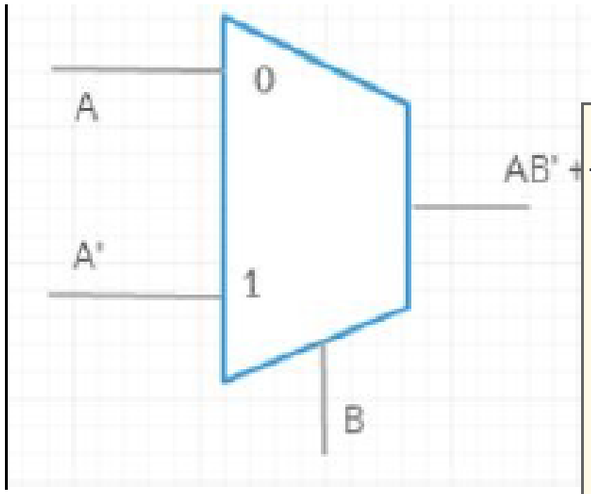
c) OR Gate: OR gate can be implemented using a 2:1 MUX as shown below.



d) NOR Gate: A NOR gate can be implemented using a combination of OR gate and NOT gate from above.

e) NAND Gate: A NAND Gate can be implemented using a combination of the AND gate and NOT gate from above.

f) XOR Gate: A XOR Gate can be implemented using a 2:1 MUX as shown below. The zeroth input is connected to A and the 1 input is connected to A′ (Use another MUX to implement NOT of A). The MUX output will now be AB′ + A′B which is same as XOR gate.



User

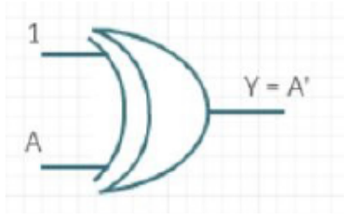
a b y
0 0 0
0 1 1
1 0 1
1 1 0
so B = 0=> Y follows A and B = 1=> Y follows A'

What are typical uses of "XOR" gates in data communication?

An XOR gate is used in computation of error detection codes like parity, CRC and ECC. It is also used in pseudo random number generators.

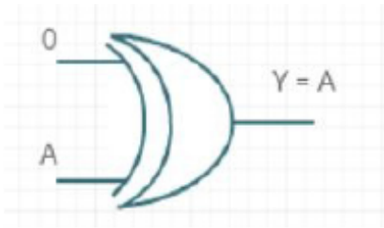
How can you design an inverter using a XOR gate?

An XOR gate is described using the equation Y = AB′ + A′B. If one of the inputs is tied to 1 as shown below, then we get: Y = A.1′ + A′.1 = 0 + A′ = A′, which is a NOT gate or an inverter.



How can you design a pass gate or a buffer using XOR gate?

A pass gate or a buffer passes the input as it is to the output. If A is the input and Y is the output, this can be represented by Y=A. Hence, this can be implemented using XOR gate by connecting one of the inputs to be always zero as below: Y = 0.A′ + 0′.A = 0 + 1.A = A

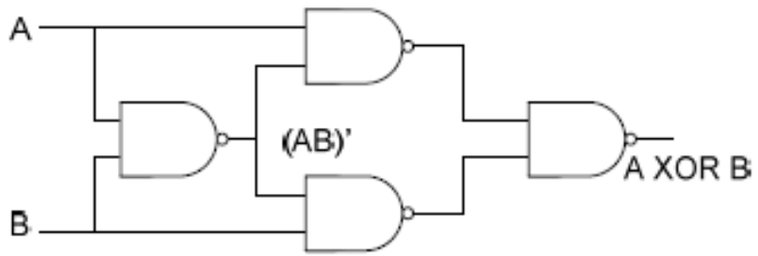


A bulb in a staircase has two switches, one switch being at the ground floor and the other one at the first floor. The bulb can be turned ON and also can be turned OFF by any one of the switches irrespective of the state of the other switch. Which gate does this logic resemble for the bulb turning on?

Let us take S0 and S1 as the two switches. If already a switch is off (0), then changing other switch to 1 should give ON=1. If already a switch is ON (1), then changing other switch to 1 should turn off the bulb (OFF=1).
Accordingly, you can have following table representing how the ON/OFF behaves based on switch.

S0 S1 ON OFF
0 0 0 1
0 1 1 0 (S0 was off, S1=1 causes bulb to turn on)
1 0 1 0 (S1 was off, S0=1 causes bulb to turn on)
1 1 0 1 (either S0/S1 was on and changing other switch to 1 causes bulb to be off)
Hence, the turning on of bulb behaves like an XOR gate.

Give implementation of XOR using minimum number of NAND gates?
A XOR B = A′B + AB′ = A(AB)′ + B(AB)′



Combinational Logic Circuits

What is the difference between Combinational and Sequential circuits?

Combinational circuits

A circuit whose output at any instant depends only on the inputs at the present instant of time is called a combinational circuit. Hence, these circuits do not contain any memory elements.
Some examples of combinational circuits are Half Adder, Full Adder, Multiplexer, Decoder etc.

Sequential circuits

A circuit whose output at any instant depends both on the inputs at the present instant of time as well as output values from the past is called a sequential circuit. These circuits hence have some form of memory elements to remember past values.
Some examples of the sequential circuit are Flip-flops, Registers, Counters, etc.

What is the difference between a Multiplexer and Demultiplexer?

MUX

An n to 1 multiplexer, or MUX, for short, is a device that allows you to pick one of n inputs and direct it to a single output.

DeMUX

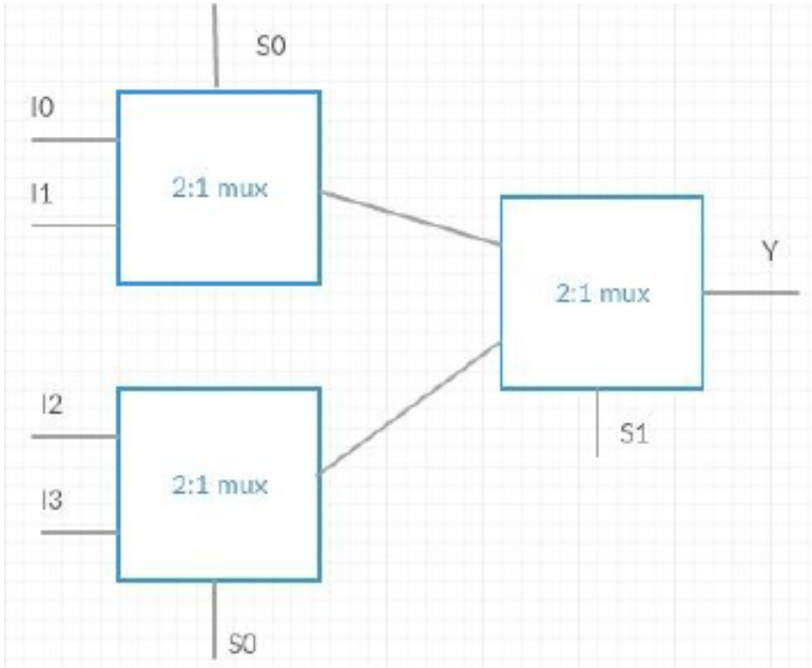
Demultiplexers (or DeMUX for short) are basically multiplexers where the inputs and outputs are reversed. For a 1 to n DeMUX, you have a single input, and n outputs to choose for directing the input.

Design a 4:1 MUX using 2:1 multiplexers?

A 4:1 MUX can be designed using 2:1 MUX as shown below. Following is the truth table for a 4:1 MUX (S1, S0 are the select lines and I0-I3 are the 4 input lines while Y is the output).

S1	S0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

So you can see that S1 can be used to select one half of the inputs and within each half S0 can be used to select one of the two inputs within that half.



What is the difference between an encoder and a decoder?

The encoder converts human-understandable language to machine-understandable language.
Encoder is a combinational circuit that does the other way around. It takes a given number of n inputs and encodes them into a smaller number of outputs
For example: An 8 to 3 encoder could do exactly reverse of above 3 to 8 decoder. There can be 8 inputs and each of them can be encoded into a 3 bit binary output.

The decoder converts machine-understandable language to human-understandable language.
A decoder is a combinational circuit that decodes a given number of inputs into a given number of output signals.
For example: A 3 to 8 decoder will decode a 3 bit input signal to an 8 bit output signal as follows
000 => Out0
001 => Out1
010 => Out2
011 => Out3
100 => Out4
101 => Out5
110 => Out6
111 => Out7

How is an encoder different from a multiplexer?

An encoder is similar to a multiplexer with the difference that a multiplexer only has a single output to which n inputs are multiplexed while an encoder normally has 2n inputs (or less) and n outputs.

What is a priority encoder and how is it different from a simple encoder?

A simple encoder is a circuit that converts a 2n bit one-hot vector to an n-bit output.
For example: a 4 to 2 simple encoder encodes as per following table. The simple encoder expects the inputs to be one hot and if more than one input is high, then the outputs becomes X.

4 to 2 Simple Encoder

I_3	I_2	I_1	I_0	O_1	O_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	1	0	0	1	0	1
1	0	0	0	1	1	1

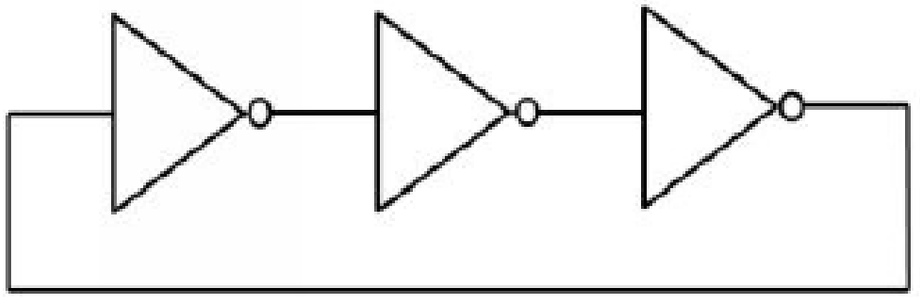
A priority encoder on the other hand encodes inputs with more than one bit being high using a priority.
For example: a 4 to 2 priority encoder will encode a 4 bit input and if more than one bit is high, the MSB takes priority.
Hence, it can be represented as table shown below.

4 to 2 Priority Encoder

I_3	I_2	I_1	I_0	O_1	O_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

Q What is a ring oscillator? What would be the frequency of a ring oscillator implemented using three NOT gates if each gate has a delay of 2 ps?

A ring oscillator is a device composed of an odd number of NOT gates whose output oscillates between two voltage levels, representing true and false. The NOT gates or inverters are attached in a chain and the output of the last inverter is fed back into the first.



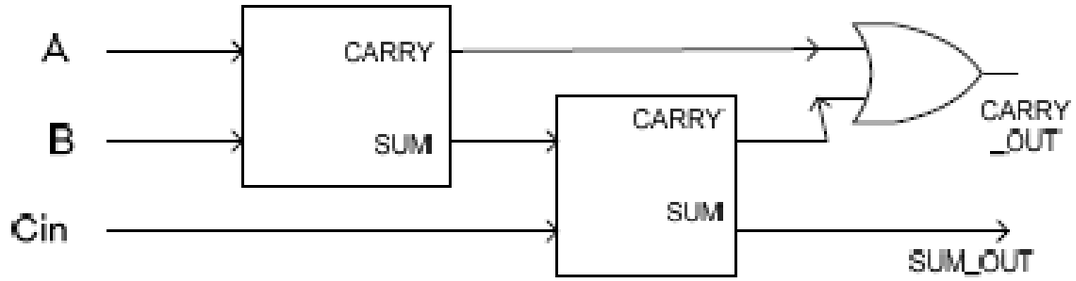
If three NOT gates are connected in a chain, then it would take three times the inverter delay for a value from input to transition to the output. Therefore, for two transitions it takes 6 times inverter delay.
Hence the clock frequency will be 1/ [6*(inverter delay)].

For our present case, clock frequency will be = 1/(6*2) THz = 1000/12 GHz = 83.33 GHz.

Design a full adder using half-adders and minimum number of external gates?

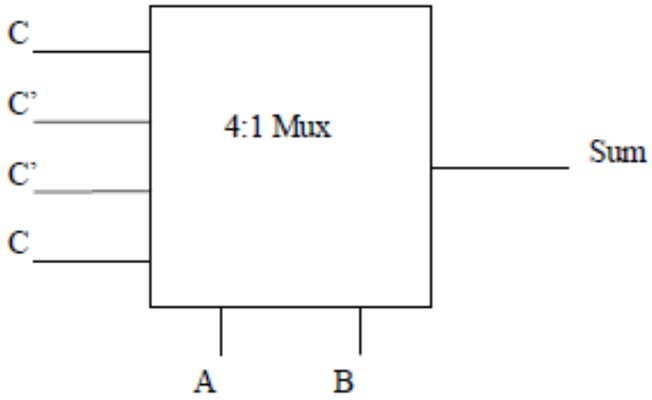
Sum = A XOR B XOR C and Carry = AB + BC + AC

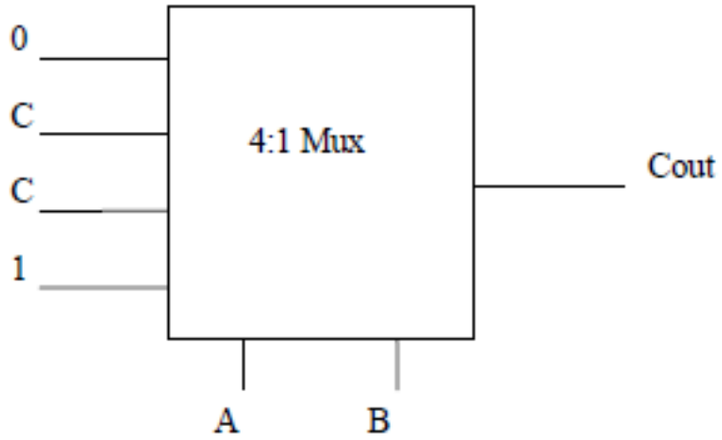
Full adder from 2 HA and one OR gate:



Implement a full adder using two 4:1 Muxes?

Sum = A XOR B XOR C and Carry = AB + BC + AC





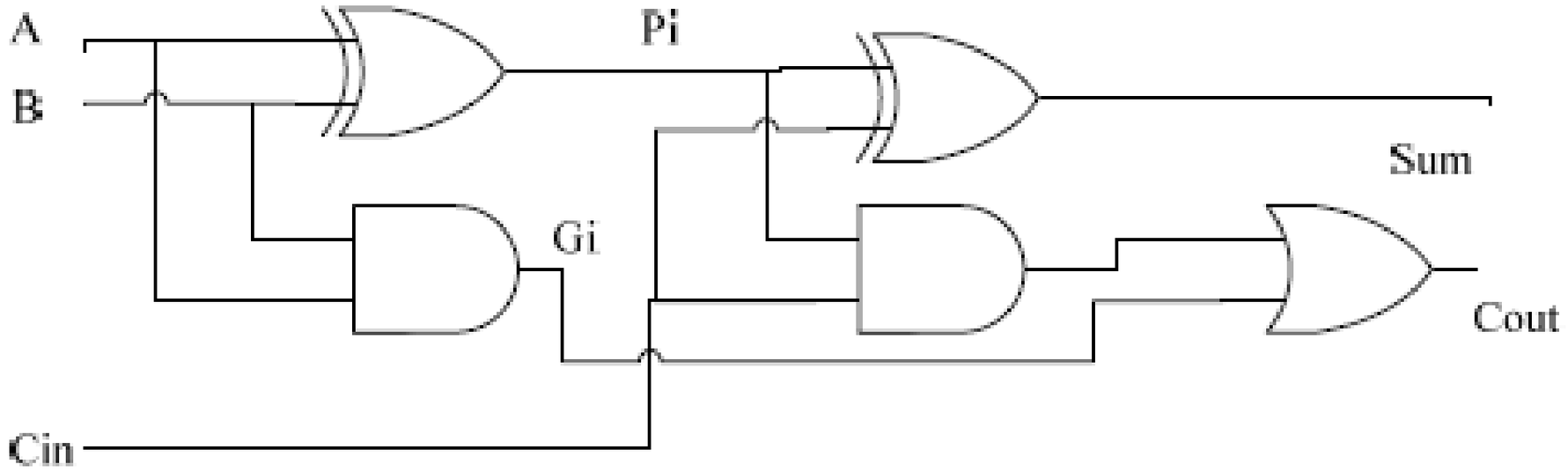
Q **A full adder can be implemented using basic gates in many ways. Show the efficient implementation among them, which needs minimal hardware?**

The suitable equations are:

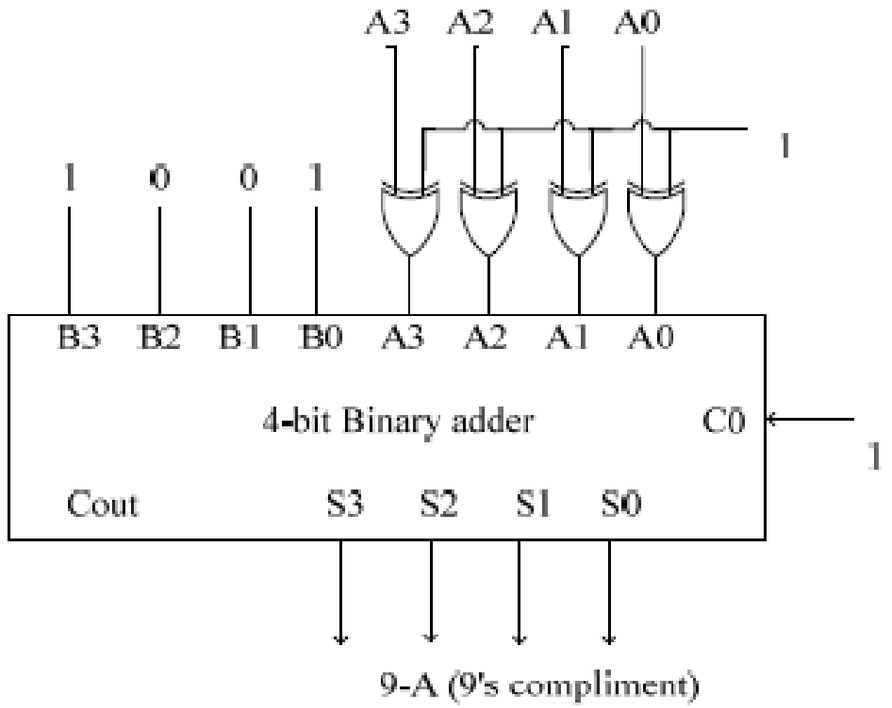
Sum = (A XOR B) XOR C

Cout = AB + (A XOR B) C

The implementation is as follows:



Design a circuit that generates 9's compliment of a BCD digit using binary adder?



9's compliment of a BCD number d is given by 9-d. That is just find the 2's compliment of d and add that to 9. Cout is needed. Just the S3-S0 of the binary adder, shown in the above figure, gives the required result.

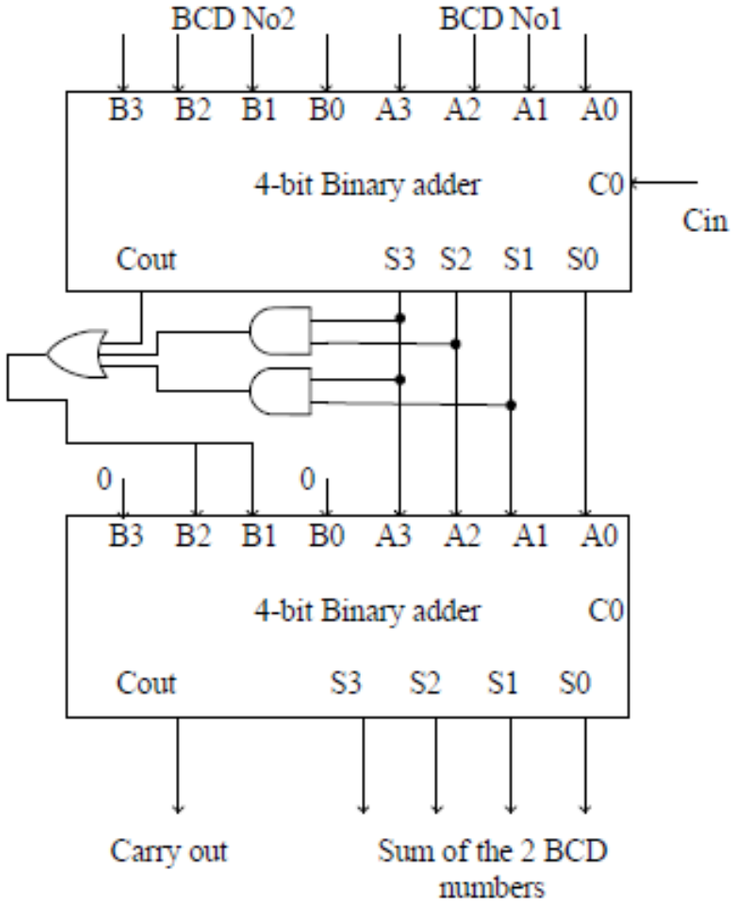
Q Give the circuit that adds two BCD numbers and gives out a BCD number?

If the result is above 9, it is needed to add 6 to obtain the result in BCD number system. So we need two 4-bit binary adders: One is just to add the two BCD numbers. The second one is for adding 6 or 0 to the result.

Extra combination logic is needed to identify the overflow. The condition for detecting the overflow can be derived as,

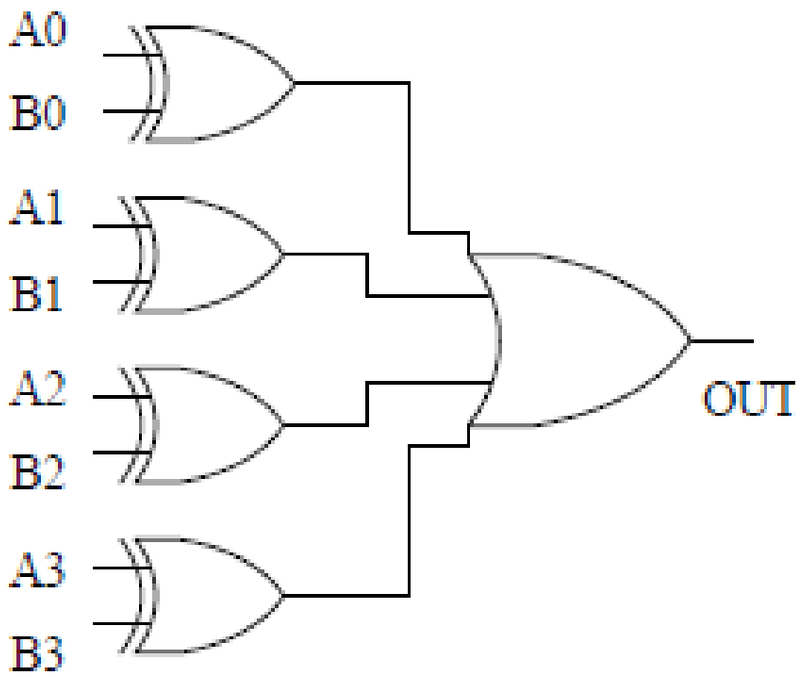
K = Cout + S3 S2 + S3 S1

K = 1 indicates the overflow and addition of 6 is needed. The complete design is shown below:



Q Give a combinational circuit which checks out whether two 4-bit input signals are same or not?

For finding out whether two signals are equal or not, the best logical gate is XOR.
The design is shown below. OUT =1 implies that the two binary numbers are not equal.



How will you count the number of 1's that are present in a given 3-bit input using full adder?

The binary number that is formed from the Carry out as MSB and Sum as LSB,gives the number of 1s of input. The same thing is illustrated in the following table. The sixth column shows Cout-Sum together where as the last column shows the actually number of 1s in the input. Note that both are exactly same.

A	B	C	Cout	Sum	Cout-Sum	No. of 1s in input
0	0	0	0	0	00(0)	0
0	0	1	0	1	01(1)	1
0	1	0	0	1	01(1)	1
0	1	1	1	0	10(2)	2
1	0	0	0	1	01(1)	1
1	0	1	1	0	10(2)	2
1	1	0	1	0	10(2)	2
1	1	1	1	1	11(3)	3

Sequential Circuits

What is the difference between Synchronous and Asynchronous circuits?

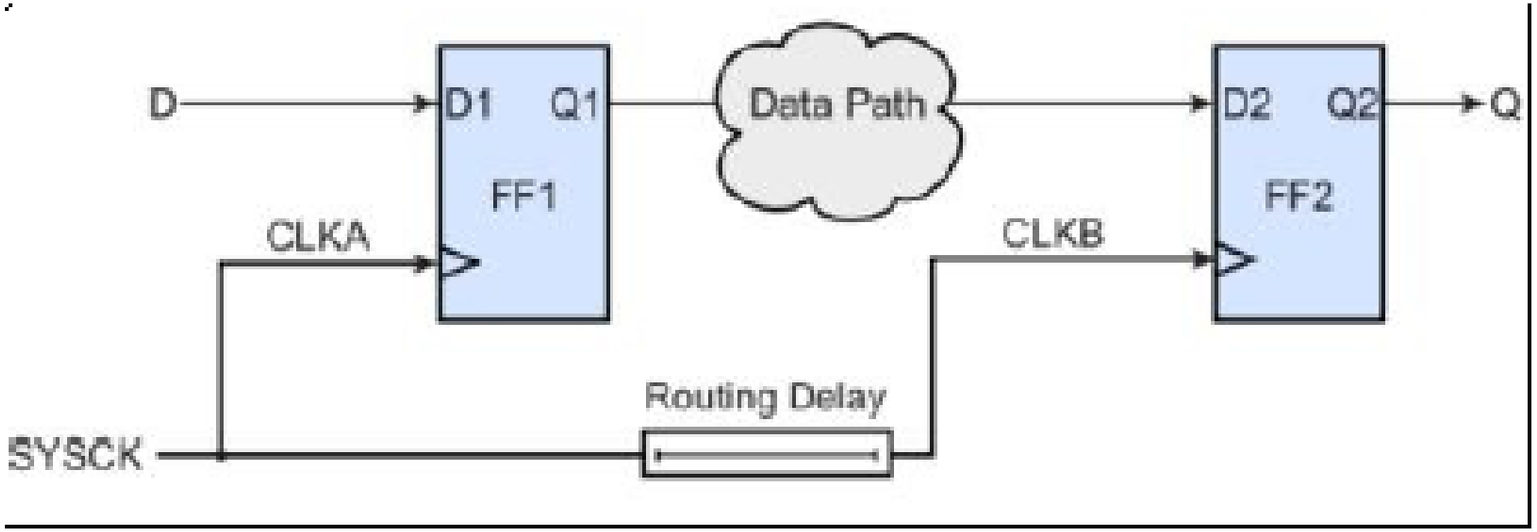
Synchronous circuits	Asynchronous circuits
Synchronous sequential circuits change their states and output values at discrete instants of time, which are specified by the rising (transition from 0 to 1) or falling edge (transition from 1 to 0) of a clock signal. A simple example is a flip-flop which stores a binary value and can change on an edge of clock based on input values.	In Asynchronous sequential circuits, the transition from one state to another is initiated bythe change in the primary inputs without any external synchronization like a clock edge. It can be considered as combinational circuits with feedback loop. Because of the feedback among logic gates, asynchronous sequential circuits may, at times, become unstable due to transient conditions and are not used commonly. A simple example: RS Latch.

Explain the concept of "Setup" and "Hold" times?

- **Setup time** is the minimum amount of time during which data signal should be stable before the clock makes a valid transition.
- **Hold time** is the minimum amount of time during which data signal should be stable after the clock makes a valid transition.

What is meant by clock skew?

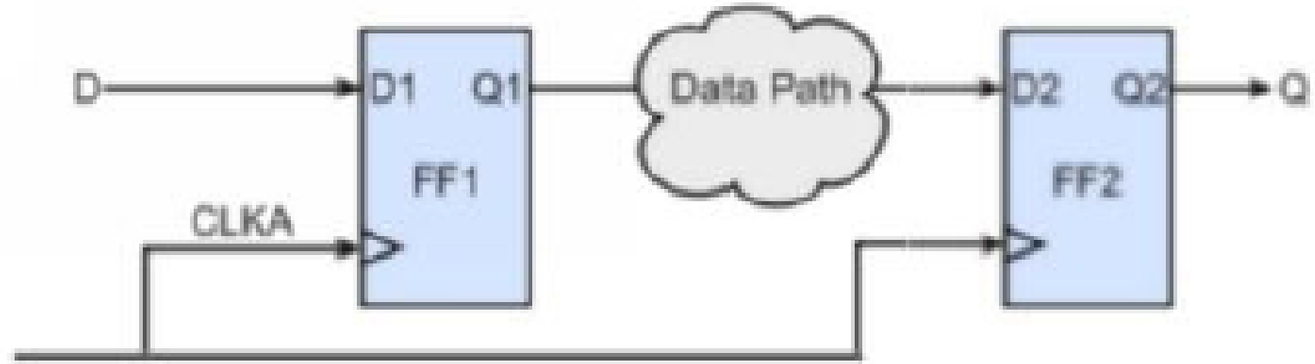
The difference in arrival times of the clock signal at any two flops which are interacting with one another is referred to as clock skew



For example in the above diagram, the D input from first flip-flop propagates through a combinational datapath circuit to second flip-flop. A clock from a common source (SYSCK) is routed to both flip-flops, but because of wire or routing delay, there could be a small difference when the edges are seen on the two flip-flops. The difference of this time is known as clock skew.

The clock skew is only important between two flip-flops where one flip-flop output is being sampled by the second flip-flop.

Q **For a given sequential circuit as shown below, assume that both the flip flops have a clock to output delay = 10ns, setup time=5ns and hold time=2ns. Also assume that the combinatorial data path has a delay of 10ns. Calculate the maximum frequency of CLKA that is possible for design to operate correctly?**



For this sequential circuit to operate correctly, output of the first flip-flop should propagate through the combinatorial data path and should be stable for a minimum duration equal to the setup time of the second flip-flop before the next clock edge. If TCLKA is the clock period, TCQ is the clock to output delay, TPD is the propagation delay for the data path and TSET is the set up time of flip flop, then we have this condition as

TCLKA ≥ T CQ + TPD + TSET

Hence, the clock period in this example needs to be >= 10+10+5 = 25ns and the max frequency will be 1/25ns = 40MHz.

Q What is the difference between a flip-flop and a latch?

- Latches and flip-flops are the basic elements for storing information. One latch or flip-flop can store one bit of information.
- The main difference between latches and flip-flops is that for latches, their outputs areconstantly affected by their inputs as long as the enable signal is asserted. In other words, when the enable signal is high, the contents of latches changes immediately when inputs changes.
- Flip-flops, on the other hand, will change the contents only at the rising or falling edge of the enable signal which is usually a clock signal. After the rising or falling edge of the clock, the flip-flop content remains constant even if the input changes.

Q What is a race condition? Where does it occur and how can it be avoided?

The race condition, is when we get the output of neither 1 nor 0. Metastability case.
If we don't maintain stable data during stable and hold time it goes to a metastable state(neither 0 nor 1).
To avoid it we have to maintain step and hold time.

What is difference between a synchronous counter and an asynchronous counter?

A counter is a sequential circuit that counts in a cyclic sequence which can be either counting up or counting down. Counters are implemented using a number of flip flops and combinational logic that feeds output of one flip-flop to another. There are two types of counters - synchronous and asynchronous.

- In synchronous counters, the clock inputs of all flip-flops are connected to a common clock signal and hence all flip-flops changes synchronously.

Some examples of synchronous counters are Ring counter and Johnson counter while

- In asynchronous counters, the clock input is connected only to the first flip-flop and the output for first flip-flop is connected to the clock input of second flip-flop and similarly every other flip-flop is clocked by the output of previous flip-flop.

some examples for asynchronous counters are Up-Down counters.

What is the difference between synchronous and asynchronous reset?

- A Reset is synchronous when it is sampled on a clock edge. When reset is synchronous, it s treated just like any other input signal which is also sampled on clock edge.
- A reset is asynchronous when reset can happen even without clock. The reset gets the highest priority and can happen any time.

How many flip-flops are needed to implement a 32 bit register?

Each flip-flop can save one bit of information. Hence to implement a 32 bit register, we would need 32 flip-flops.

What is the difference between a Mealy and a Moore finite state machine?

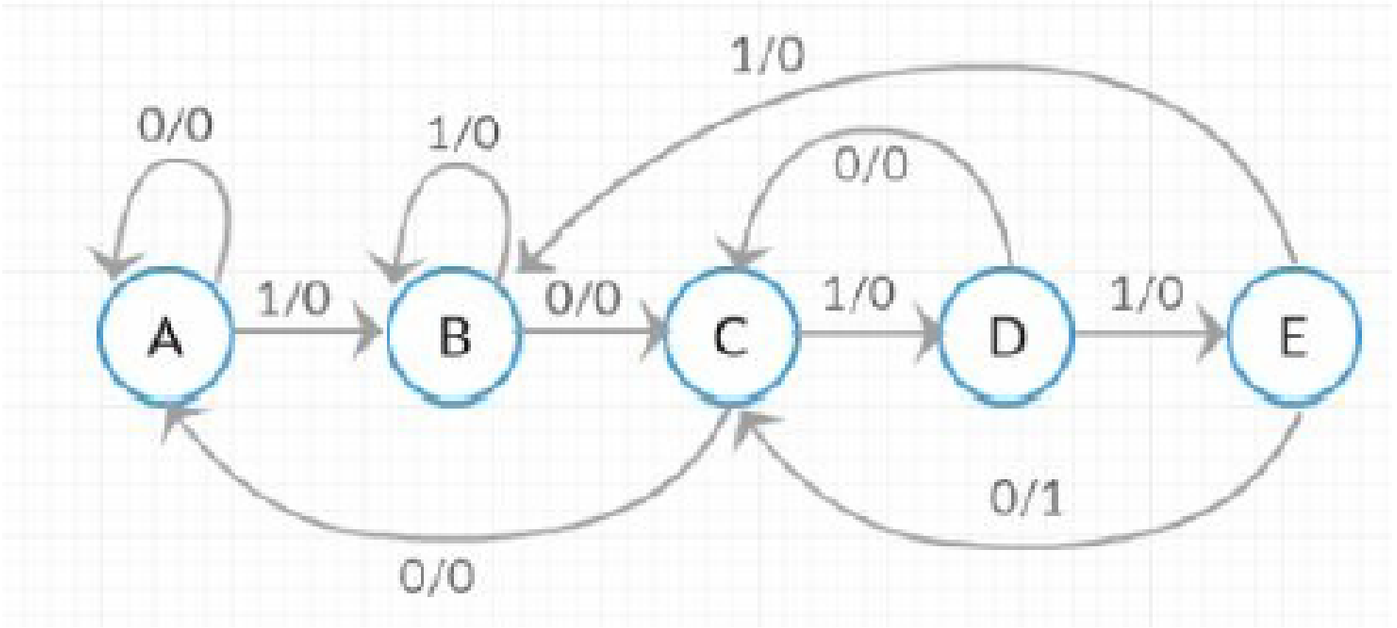
- A Mealy Machine is a finite state machine whose output depends on the present state as well as the present input.
- A Moore Machine is a finite state machine whose output depends only on the present state.

What is the difference between a Combinational Circuit and a Sequential Circuit?

- Output depends only on Input
- Doesn't process any memory
- There will be no feedback from output to input
- Easier to design and faster in operation
- Eg: MUX, DEMUX, Encoder, Decoder, Adder

- Output depends on Input and past state
- Process memory element
- Involves some kind of feedback
- Lots of complexity invloves in the design and slower in operation
- Eg: Flip-flops, Lathes, Registers, Counters

Q **Design a sequence detector state machine that detects a pattern "10110" from an input serial stream.**

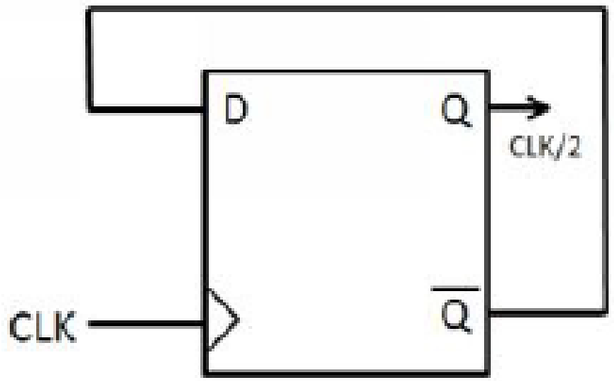


Sequence detector state machine that detects a pattern questions:

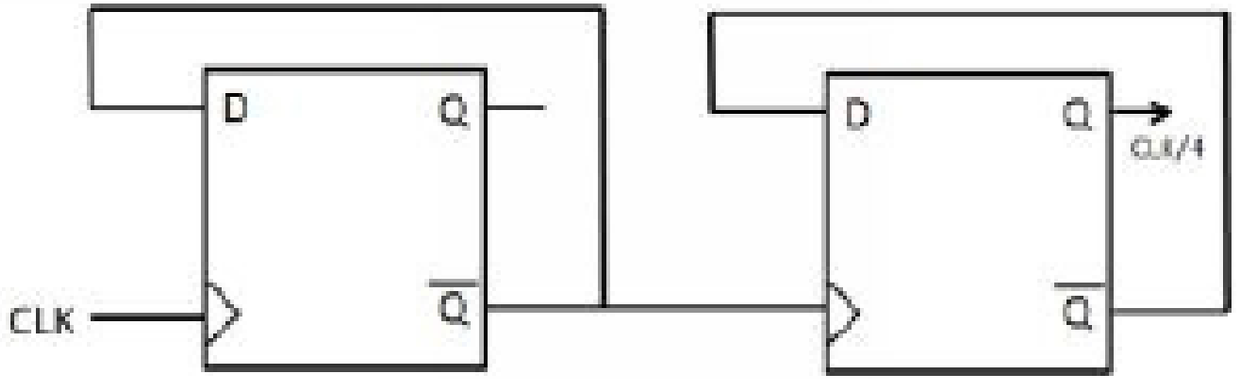
- **101**
- **1010**
- **1001**

Implement following logics using minimum number of D Flip-Flops:

a) Clock Divide by 2



b) Clock Divide by 4



Mention two basic applications of flip-flops?

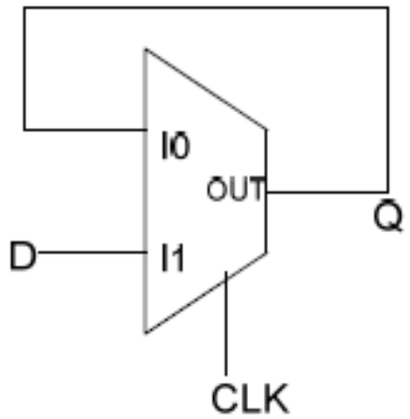
The major applications of flip-flops are:

- Data storage
- Data transfer
- Counting and
- Frequency division

What is transparent latch?

D-Latch is called transparent Latch. As it transfers the data as it is to the output on enable.

Design a D-latch using 2:1 Mux.



What is race-around condition? Explain it in case of J-K Latch and solution to avoidthat?

The race around condition means: the output oscillating between 0s & 1s.
This problem will occur in Latches especially if the clock is high for long time. In case of J-K Latch, J=K=1 gives Q(t+1) = Q(t)' Consider the case when clock is high for long time and J=K=1. Then the output oscillates between 0 & 1 continuously as long as the clock is high. To avoid this, use Master-Slave configuration which latches the input only at clock edges. So in that case, irrespective of the duration of clock high, output will just compliment of previous output. There won't be any oscillation as such.

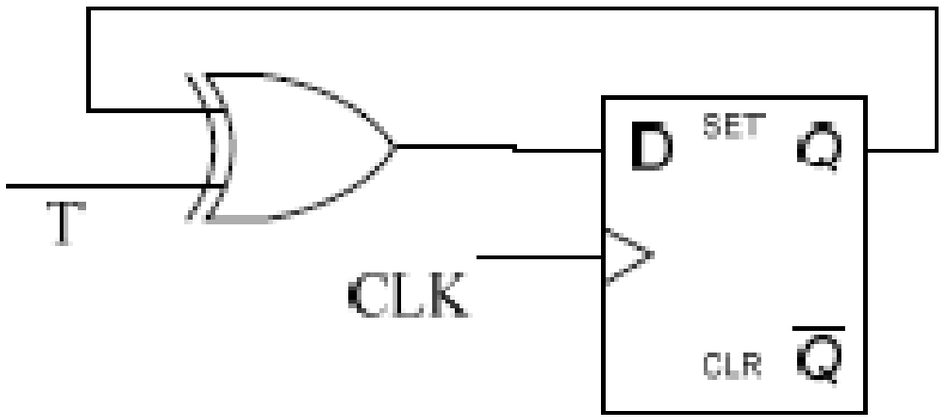
Implement T-flip flop from D-flip flop?

b) A T flip flop to a D flip flop.

D	Q_n	Q_{n+1}	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

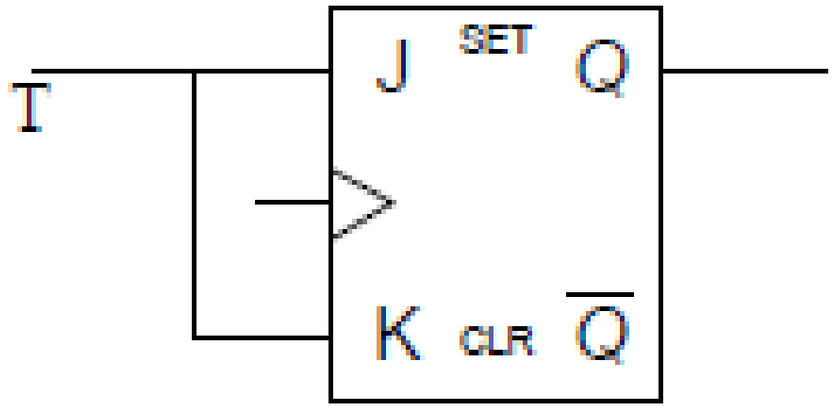
$T = f(D, Q_n) = D\overline{Q_n} + \overline{D}Q_n$

$T = f(D, Q_n) = D \oplus Q_n$

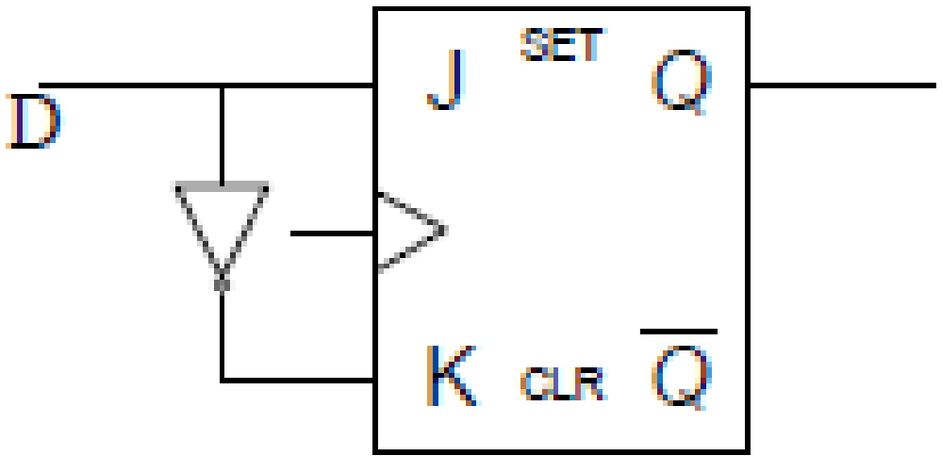


Show how to convert J-K flip flop into (a) T-flip flop (b) D-flip flop?

(a) TFF:



(b) DFF:



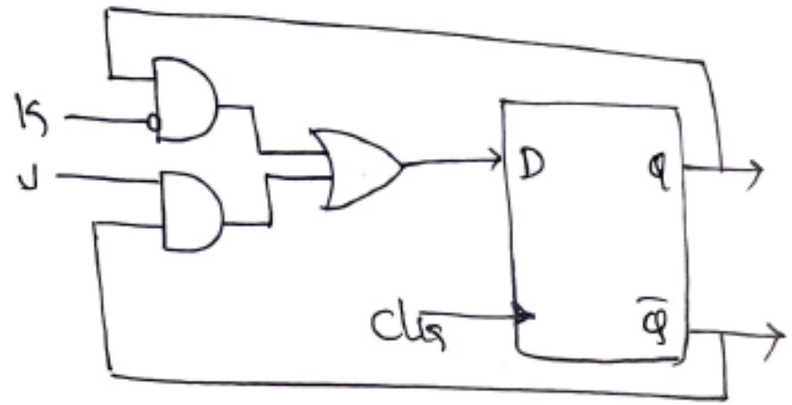
Convert by adding external gate D-flip flop into J-K flip flop?

J	K	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

$$D = f(J, K, Q_n) = \sum m(1, 4, 5, 6)$$

D	KQ_n 00	01	11	10
J = 0	0	1	0	0
J = 1	1	1	0	1

$$D = J\bar{Q}_n + \bar{K}Q_n$$



Enjoy The Content?

If this post useful for your business,
please like, share, comment and save.

Thank you