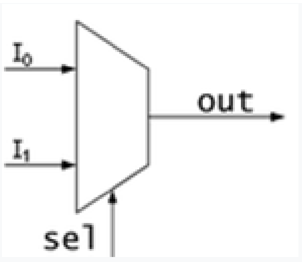
Multiplexer:



Schematic of a 2-to-1 Multiplexer

A **multiplexer**(**mux**) is a device that selects analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2^n inputs has n select lines, which are used to select which input line to send to the output. Multiplexers are mainly used to increase the amount of data that can be sent over the network within a certain amount of time and bandwidth. A multiplexer is also called a **data selector**. Multiplexers can also be used to implement Boolean functions of multiple variables.

Multiplexer is a **combinational logic circuit** which allows only one input at a particular time to generate the output. The signals which control which input will be reflected at the output end is determined by the **SELECT INPUT** lines. It is also called as **Many-to-One** circuit. This is because of its ability to select one signal out of many inputs.

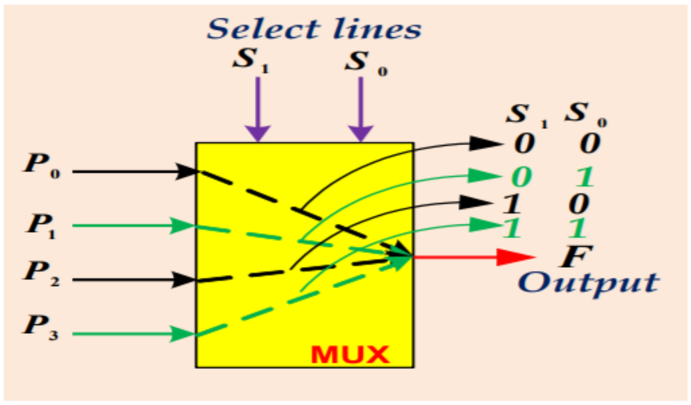
The MUX is the very crucial component of the communication system. This is because, in such systems, we need to select a single channel from various other channels. A multiplexer can be considered as a digitally controlled switch. The controlling code which selects a particular input line can be given as binary input in the form of selection line. The output will be one of the inputs given to MUX, which is decided by selection lines.

Four-to-One Multiplexer:

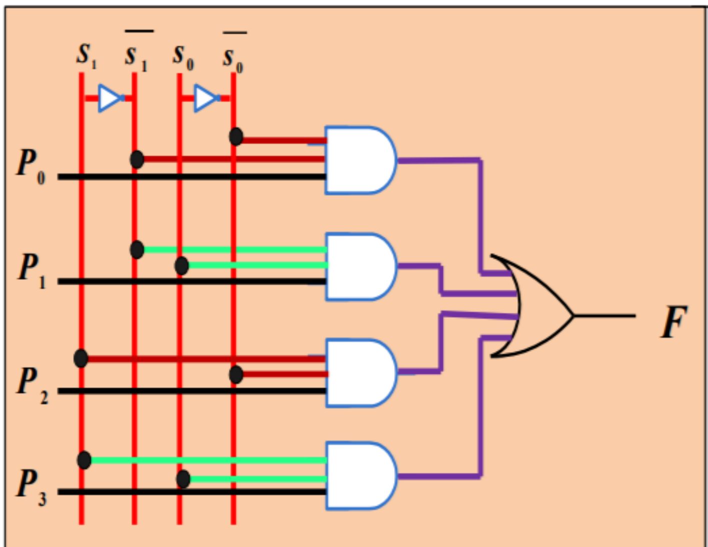
In 4:1 MUX, there will be 4 input lines and 1 output line. And to control which input should be selected out of these 4, we need 2 selection lines.

- ✓ 4-data input MUX
- ✓ s_1, s_0 - Select lines.
- ✓ p_0, p_2, p_3, p_1 - Input lines.
- ✓ F- Single output line.

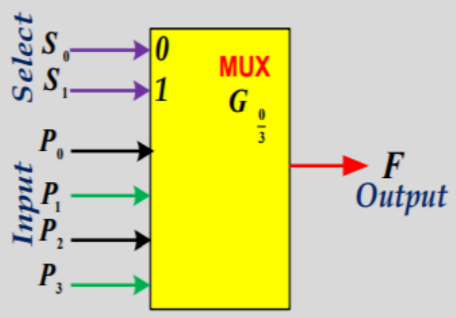
Select lines		***Output***
S_1	S_0	F
0	*0*	P_0
0	*1*	P_1
1	*0*	P_2
1	*1*	P_3



$$F = \overline{S_1}\,\overline{S_0}P_0 + \overline{S_1}S_0P_1 + S_1\overline{S_0}P_2 + S_1S_2P_3$$



Internal logic circuit for 4-1 MUX

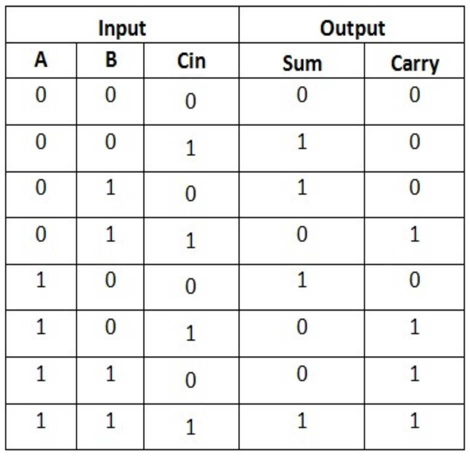


4:1 MUX module "digital element"

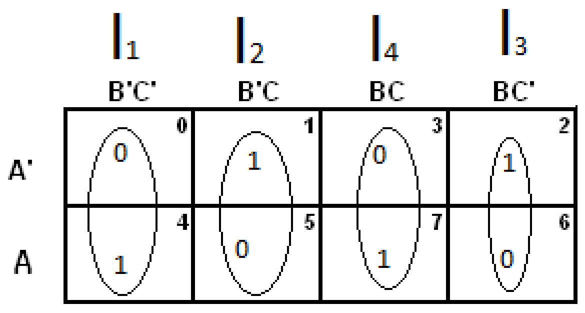
Example: Design a Full adder using 4:1 Mux

For sum output : input B and C acts as select lines, as per the truth table of 4x1 mux output fallows: I_1 when BC=00, I_2 when BC=01, I_3 when BC=10 and I_4 when BC=11.

Truth table :



Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



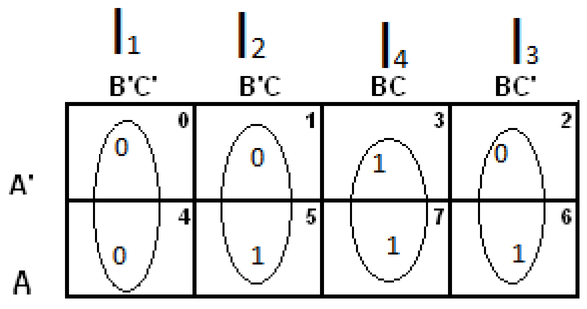
From the K-map expression for 4 inputs of the MUX can be written as

I_1 = A I_2 = A' I_3 = A' I_4 = A

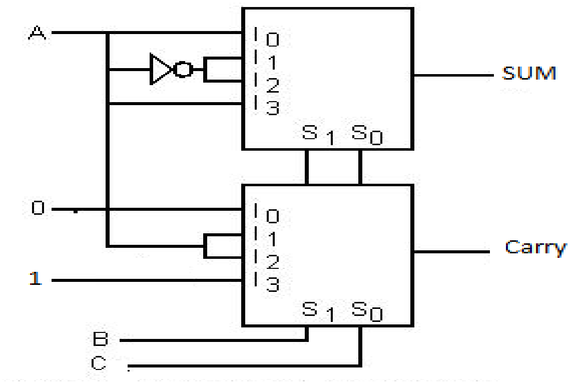
For Carry output:

From the K-map expression for 4 inputs of the MUX can be written as

I_1 = 0 I_2 =A I_4 = 1 I_3= A

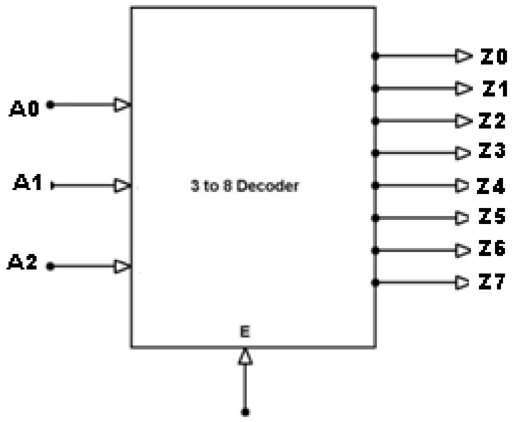


Logic diagram for the full adder using 2 4X1 mux:

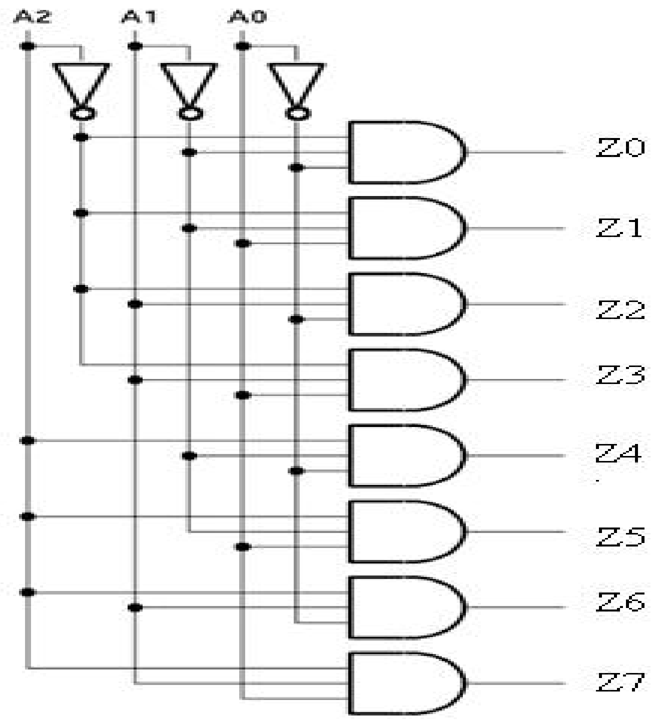


Decoder:

Decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^nunique outputs and has a enable pin.Each outputs represents minterms of n input variable. The circuit is designed with AND and NAND logic gates. It takes 3 binary inputs and activates one of the eight outputs. 3 to 8 line decoder circuit is also called as binary to an octal decoder.



The decoder circuit works only when the Enable input(E) is high, we can also have decoder with low enable input. A0, A1 and A2 are three different inputs and Z0, Z1, Z2, Z3, Z4, Z5, Z6, Z7 are the eight outputs.



Truth table:

A2	A1	A0	E	Z0	Z1	Z2	Z3	Z4	Z5	Z6	Z7
x	x	x	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0	0
1	0	1	1	0	0	0	0	0	1	0	0
1	1	0	1	0	0	0	0	0	0	1	0
1	1	1	1	1	0	0	0	0	0	0	1

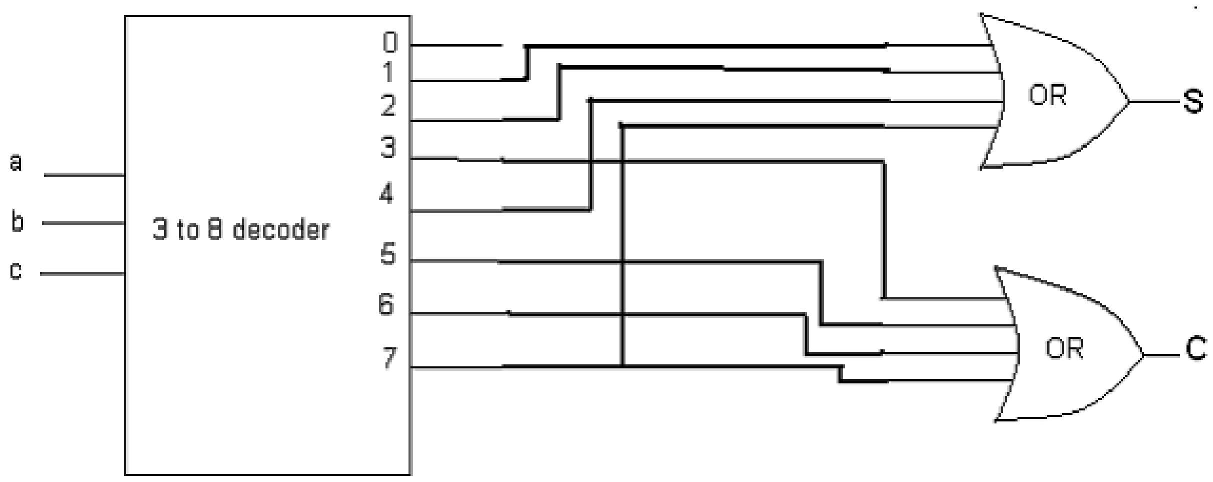
When the Enable pin (E) is low all the output pins are low.

<u>IMPLEMENTING FULL ADDER USING DECODER</u>

Equation for sum S = ab’c’ + a’b’c + a’bc’ + abc = Σ(1,2,4,7)

Equation forcarry C_{out}= ab + ac + bc
 = ab(c + c’) + ac (b + b’) + bc (a + a’) = abc + abc’ + abc + ab’c + abc + a’bc
= abc +a’bc +ab’c+abc’= Σ (3, 5, 6, 7)

So we can implement it from decoder using OR gates as follow:



SR Flip-Flop

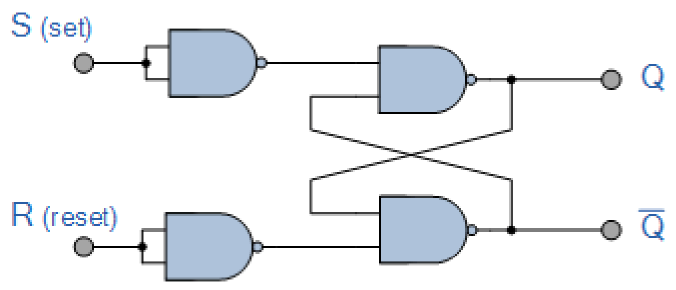
This simple flip-flop is basically a one-bit memory bistable device that has two inputs, one which will “SET” the device (meaning the output = “1”), and is labelled **S** and one which will “RESET” the device (meaning the output = “0”), labelled **R**.

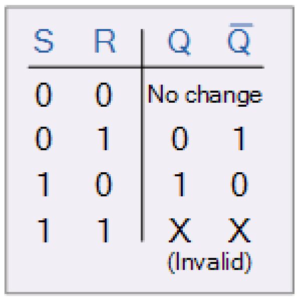
A basic NAND gate SR flip-flop circuit provides feedback from both of its outputs back to its opposing inputs and is commonly used in memory circuits to store a single data bit. Then the SR flip-flop actually has three inputs, Set, Reset and its current output Q relating to it’s current state or history. 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 Latch

The simplest way to make any basic single bit set-reset SR latch is to connect together a pair of cross-coupled 2-input NAND gates as shown, 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.

NAND Gate SR latch

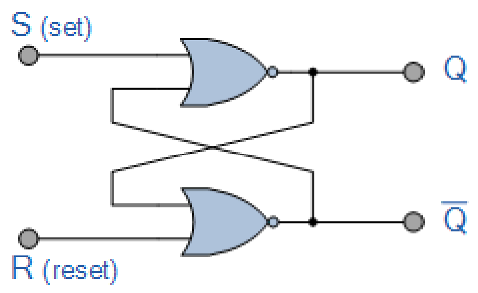


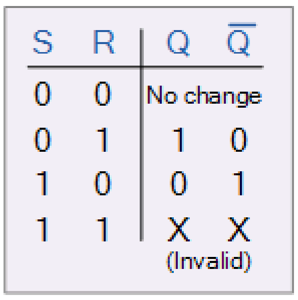


S	R	Q	$\overline{Q}$
0	0	No change	
0	1	0	1
1	0	1	0
1	1	X	X
		(Invalid)	

As well as using NAND gates, it is also possible to construct simple one-bit **SR** latch using two cross-coupled NOR gates connected in the same configuration. The circuit will work in a similar way to the NAND gate circuit above, except that the inputs are active HIGH and the invalid condition exists when both its inputs are at logic level "1", and this is shown below.

The NOR Gate SR Flip-flop

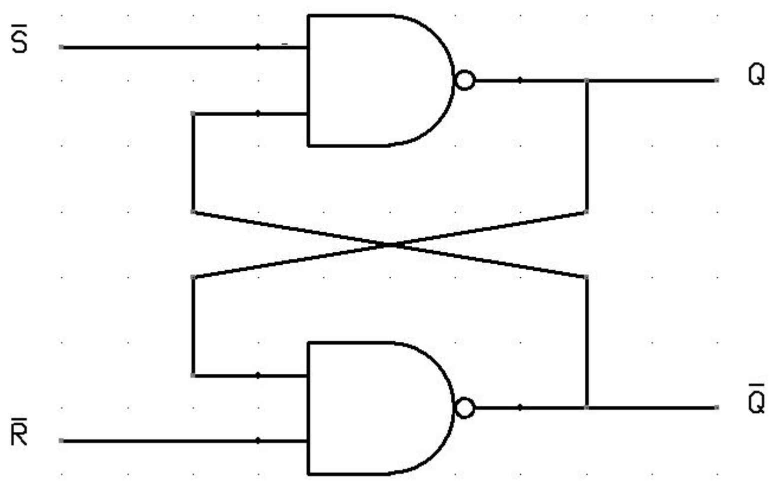




S	R	Q	$\overline{Q}$
0	0	No change	
0	1	1	0
1	0	0	1
1	1	X	X
		(Invalid)	

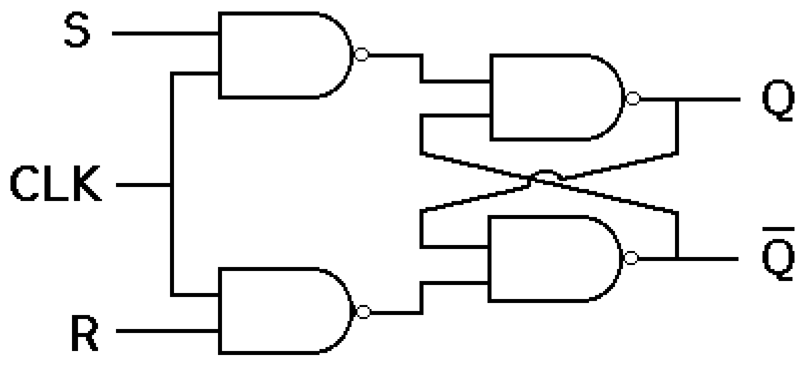
Flip flop v/s Latch

The basic difference between a latch and a flip-flop is a gating or clocking mechanism. For example, let us talk about SR latch and SR flip-flops. In this circuit when you Set S as active the output Q would be high and Q' will be low. This is irrespective of anything else. (This is an active low circuit so active here means low, but for an active high circuit active would mean high)



SR Latch

A flip flop on the other hand is synchronous and is also known as gated or clocked SR latch.



SR Flip-Flop

Advertisement

In this circuit, the output is changed (i.e. the stored data is changed) only when you give an active clock signal. Otherwise, even if the S or R is active the data will not change. Let's look at the types of flip flops to understand better.

SR Flip Flop

There are majorly 4 types of flip flops, with the most common one being SR flip flop. As shown above, it is the simplest and the easiest to understand. The two outputs as shown above are the inverse of each other. The outputs of a

SR flip flop are highlighted in the table below.

S	R	Q	Q'
0	0	No change	
0	1	0	1
1	0	1	0
1	1	Invalid state	

J-K flip-flop

The basic S-R NAND flip-flop circuit has many advantages and uses in sequential logic circuits but it suffers from following problem.

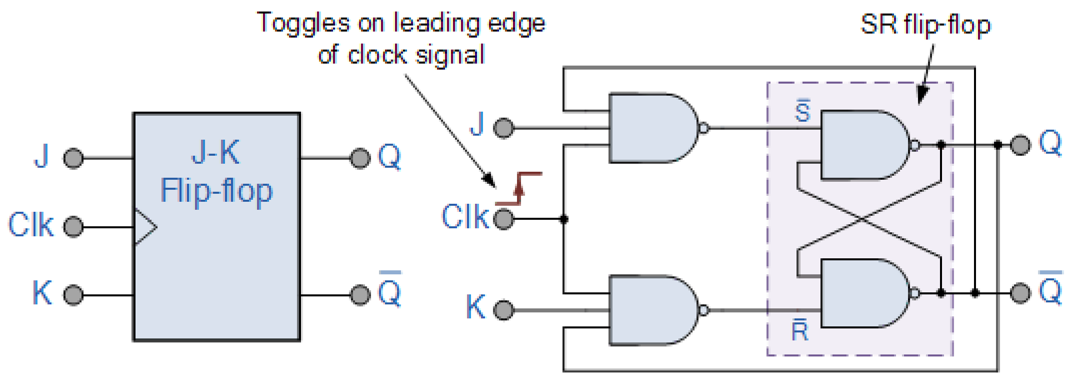
- The Set = 1 and Reset = 1 condition must always be avoided

This simple **JK flip Flop** is the most widely used of all the flip-flop designs and is considered to be a universal flip-flop circuit. The two inputs labelled “J” and “K” are not shortened abbreviated letters of other words, such as “S” for Set and “R” for Reset, but are themselves autonomous letters chosen by its inventor Jack Kirby to distinguish the flip-flop design from other types.

The sequential operation of the JK flip flop is exactly the same as for the previous SR flip-flop with the same “Set” and “Reset” inputs. The difference this time is that the “JK flip flop” has no invalid or forbidden input states of the SR Latch even when S and R are both at logic “1”.

The **JK flip flop** is basically a gated SR flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level “1”. Due to this additional clocked input, a JK flip-flop has four possible input combinations, “logic 1”, “logic 0”, “no change” and “toggle”. The symbol for a JK flip flop is similar to that of an *SR Bistable Latch* except for the addition of a clock input.

The Basic JK Flip-flop



Symbol

Circuit

Both the S and the R inputs of the previous SR bistable have now been replaced by two inputs called the J and K inputs, respectively after its inventor Jack Kilby. Then this equates to: J = S and K = R.
The two 2-input AND gates of the gated SR bistable have now been replaced by two 3-input NAND gates with the third input of each gate

connected to the outputs at Q and Q . This cross coupling of the SR flip-flop allows the previously invalid condition of S = "1" and R = "1" state to be used to produce a "toggle action" as the two inputs are now interlocked.

The Truth Table for the JK Function

	Input			Output		Description
	Clock	J	K	Q	Q	
same as for the SR Latch	0	X	X	Same as previous state		Memory no change
	1	0	0	Same as previous state		Memory no change
	1	0	1	0	1	Reset Q » 0
	1	1	0	1	0	Set Q » 1

toggle action	1	1	1	0	1	Toggle
	1	1	1	1	0	

Then the JK flip-flop is basically an SR flip flop with feedback which enables only one of its two input terminals, either SET or RESET to be active at any one time thereby eliminating the invalid condition seen previously in the SR flip flop circuit.

Also when both the J and the K inputs are at logic level “1” at the same time, and the clock input is pulsed “HIGH”, the circuit will “toggle” from its SET state to a RESET state, or visa-versa.