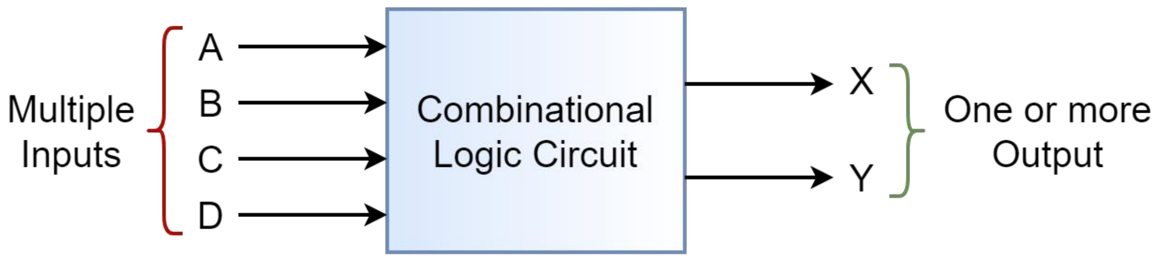
Introduction to Combinational circuits

Combinational Logic Circuits are made from the basic and universal gates. The output is defined by the logic and it is depending only the present input states not the previous states.

Inputs and output(s) : logic 0 (low) or logic 1 (high).



$$(X, Y) = f(A, B, C, D)$$

Analysis and design procedures

The following are the basic steps to design a combinational circuit

1. Define the problem.

2. Determine the number of input and output variables.

3. Fix a letter symbol to the input and the outputs. (e.g. A, B, C, w, x, y, F, etc)

4. Get the relationship between input and output from the truth table.

5. By using K-map obtain the simplified Boolean expression for the outputs.

6. Draw the logic diagram using gates.

Example: Design a combinational logic circuit with three inputs , the output is at logic 1

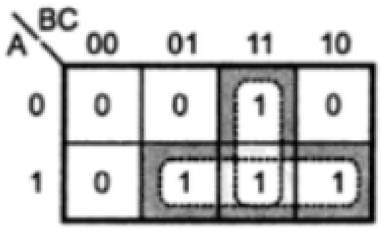
when more than one inputs are at logic 1.

Solution: Assume A, B, C are inputs and Y is output.

Truth table

Inputs			Output
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

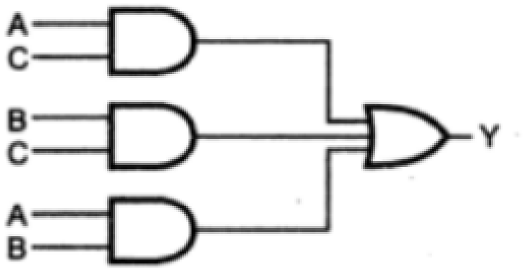
K map Simplification



Boolean Expression

Y=AC + BC + AB

Logic Diagram



Adder

The Basic operation in digital computer is binary addition. The circuit which perform the addition of binary bits are called as Adder.

The logic circuit which performs the addition of two bit is called Half adder and three bit is called Full adder.

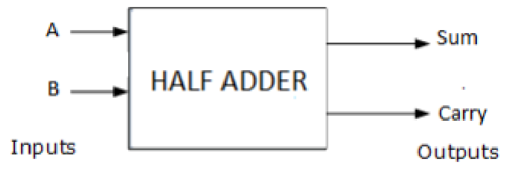
Rules for two-bit addition

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10_2

Half Adder

The two inputs of the half adders are augend and addend, the outputs are sum and carry.

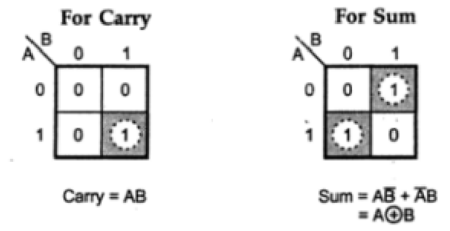
Block diagram of Half adder



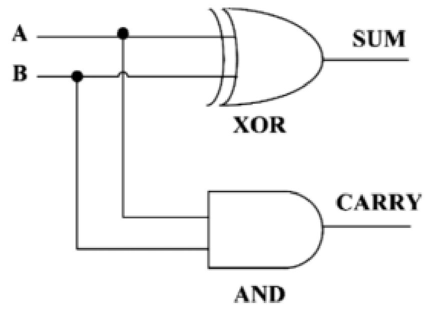
Truth table of Half adder

Inputs		Outputs	
A	**B**	**Carry**	**Sum**
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

K-map simplification



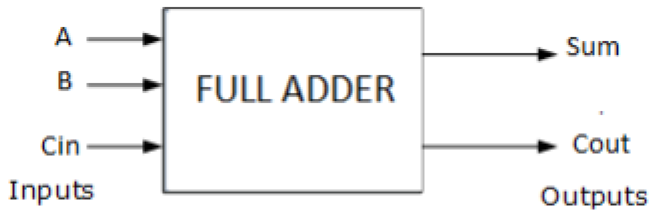
Logic diagram



Full Adder

The three inputs of the full adders are augend, addend and the carry input from the previous addition, the outputs are sum and carry

Block diagram of Full adder

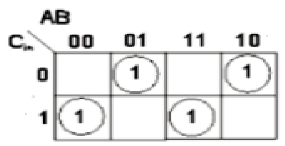


Truth table

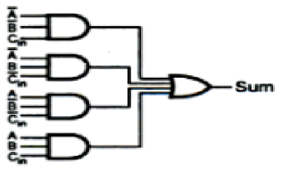
Inputs			Outputs	
A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

K-map simplifications

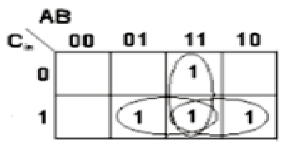
For Sum



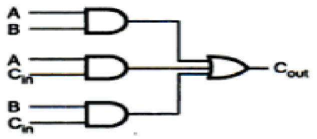
$Sum = \bar{A}\bar{B}Cin + \bar{A}B\overline{Cin} + A\bar{B}\overline{Cin} + ABCin$



For Cout



$Cout = AB + BCin + ACin$



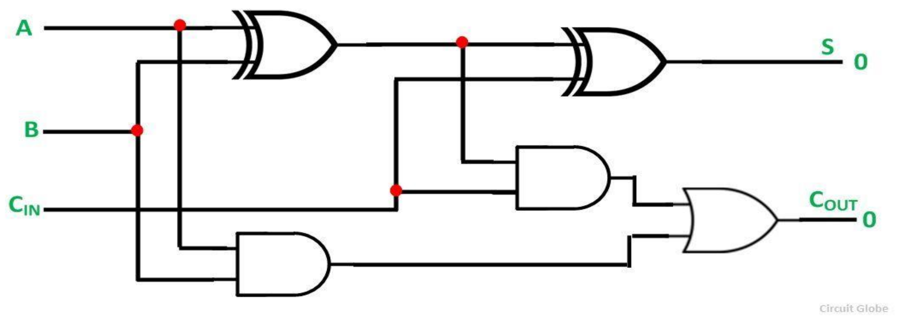
The expression for sum is

$$\begin{aligned}
SUM &= \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in} \\
&= C_{in}(\bar{A}\bar{B} + AB) + \bar{C}_{in}(A\bar{B} + \bar{A}B) \\
&= C_{in}\left[(\bar{A} + B)\cdot(A + \bar{B})\right] + \bar{C}_{in}(A\bar{B} + \bar{A}B) \\
&= C_{in}(\overline{A\bar{B}}\cdot\overline{\bar{A}B}) + \bar{C}_{in}(A\bar{B} + \bar{A}B) \\
&= C_{in}(\overline{A\bar{B} + \bar{A}B}) + \bar{C}_{in}(A\bar{B} + \bar{A}B) \\
&= C_{in} \oplus (A\bar{B} + \bar{A}B) \\
&= C_{in} \oplus (A \oplus B)
\end{aligned}$$

The Expression for carry is

$$
\begin{aligned}
C_{out} &= AB + C_{in}(A\overline{B} + \overline{A}B) \\
&= AB + A\overline{B}C_{in} + \overline{A}BC_{in} \\
&= AB(C_{in}+1) + A\overline{B}C_{in} + \overline{A}BC_{in} && \because C_{in}+1=1 \\
&= AB\,C_{in} + AB + A\overline{B}C_{in} + \overline{A}BC_{in} \\
&= AB + A\,C_{in}(B+\overline{B}) + \overline{A}B\,C_{in} \\
&= AB + AC_{in} + \overline{A}BC_{in} \\
&= AB(C_{in}+1) + A\,C_{in} + \overline{A}BC_{in} && \because C_{in}+1=1 \\
&= ABC_{in} + AB + A\,C_{in} + \overline{A}BC_{in} \\
&= AB + A\,C_{in} + BC_{in}(A+\overline{A}) \\
&= AB + A\,C_{in} + BC_{in}
\end{aligned}
$$

Logic Diagram (Full Adder Using Half adders)



Subtractor

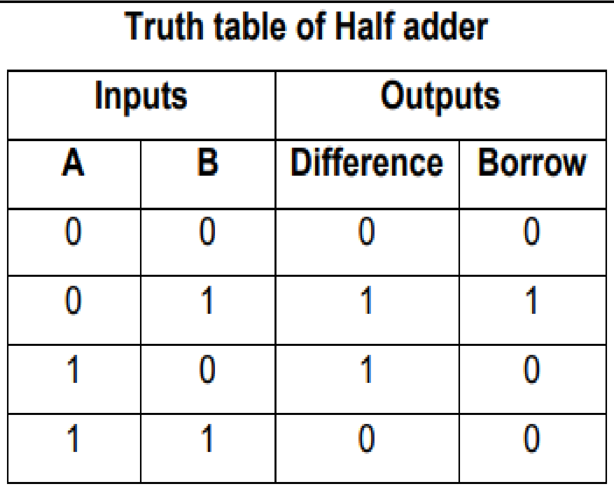
Subtractor is a logic circuit which is used to subtract two binary number (digit) and provides Difference and Borrow as an output. In digital electronics we have two types of subtractor, Half Subtractor and Full Subtractor.

Rules for one bit Subtraction

0 - 0 = 0
0 - 1 = 1 with borrow 1
1 - 0 = 1
1 - 1 = 0

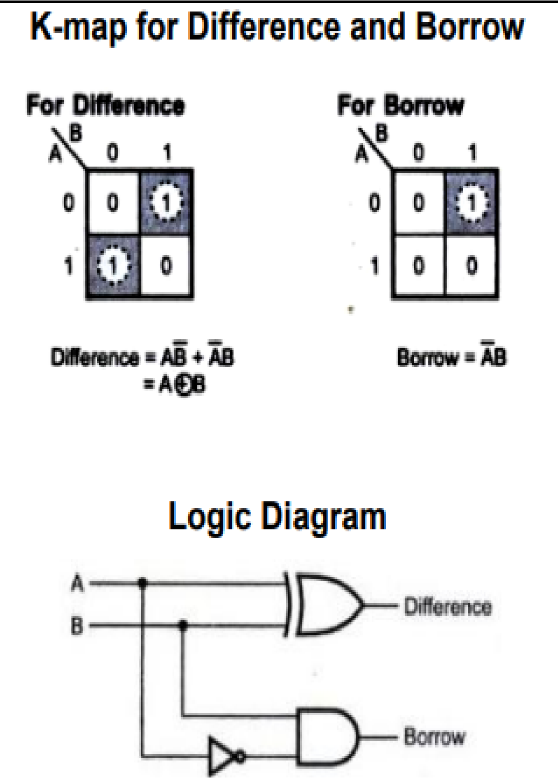
Half Subtractor

Half Subtractor is used for subtracting one single bit binary digit from another single bit binary digit. The truth table of Half Subtractor is shown below.



Truth table of Half adder

Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



Full Subtractor

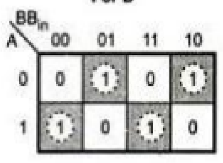
A logic Circuit Which is used for Subtracting Three Single bit Binary digit is known as Full Subtractor. The inputs are A, B, Bin and the outputs are D and Bout.

Truth table

Inputs			Outputs	
A	**B**	**Bin**	**D**	**Bout**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

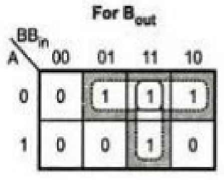
K-map for D and Bout

For D



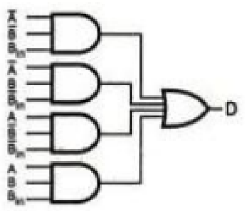
$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$

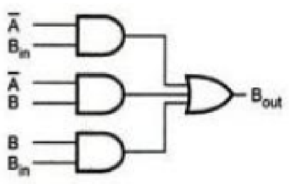
For B_{out}



$B_{out} = \bar{A}B_{in} + \bar{A}B + BB_{in}$

Logic Diagram


$\bar{A}$ $\bar{B}$ B_{in}
$\bar{A}$ B $\bar{B}_{in}$
A $\bar{B}$ $\bar{B}_{in}$
A B B_{in}
D



We can further simplify the function of the Difference (D)

$$\begin{aligned} D &= \bar{A}\,\bar{B}\,B_{in} + \bar{A}\,B\,\bar{B}_{in} + A\,\bar{B}\,\bar{B}_{in} + A\,B\,B_{in} \\ &= B_{in}\,(\bar{A}\,\bar{B} + AB) + \bar{B}_{in}\,(\bar{A}\,B + A\,\bar{B}) \\ &= B_{in}\,(A \odot B) + \bar{B}_{in}\,(A \oplus B) \\ &= B_{in}\,(\overline{A \oplus B}) + \bar{B}_{in}\,(A \oplus B) \\ &= B_{in} \oplus (A \oplus B) \end{aligned}$$

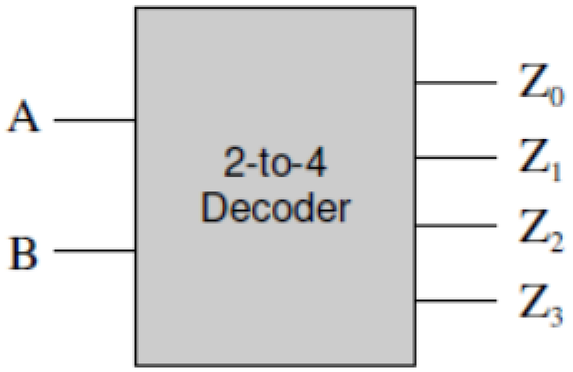
Decoder

Decoder is a combinational circuit. It has N inputs and 2^N outputs.

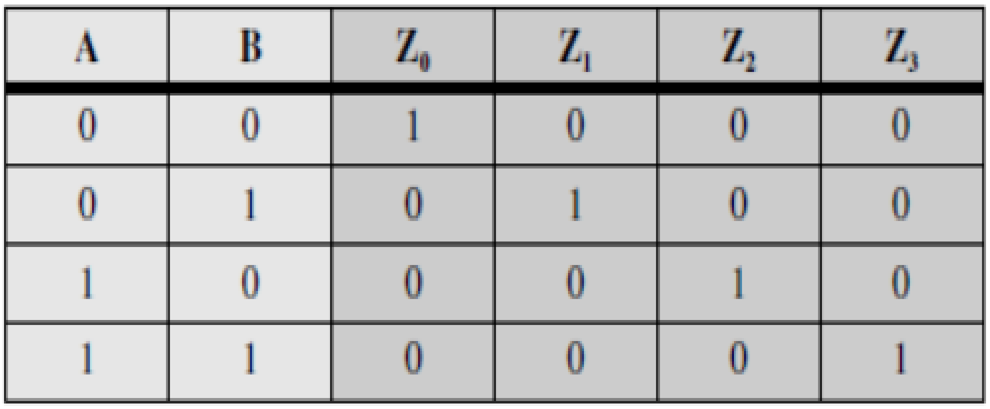
2 to 4 Decoder

It has 2 inputs and 2^2= 4 outputs.

Block Diagram

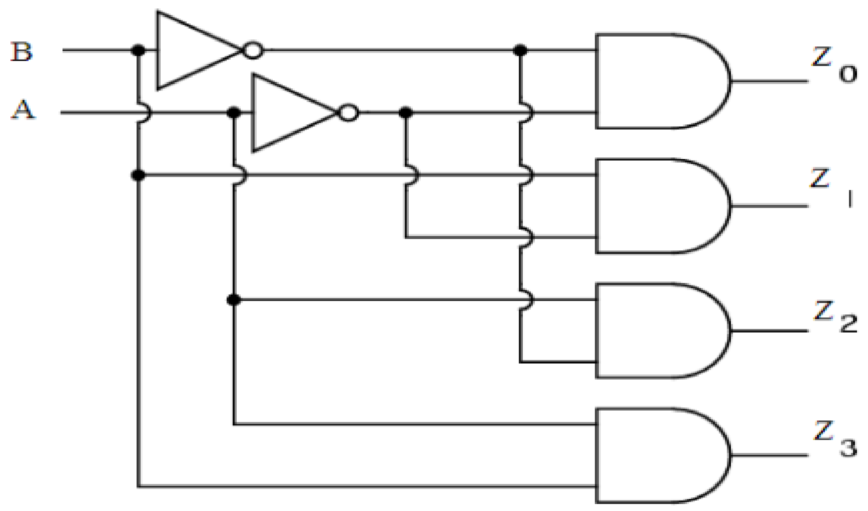


Truth Table

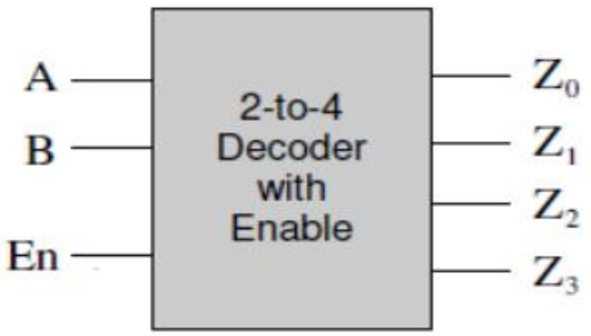


A	B	Z_0	Z_1	Z_2	Z_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Logic Diagram



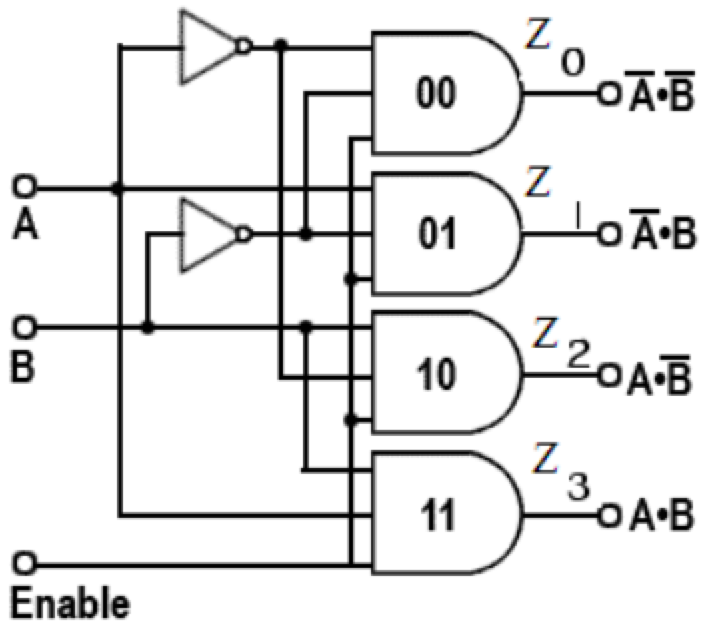
2 to 4 Decoder with Enable input



Truth Table

	En	A	B	Z_0	Z_1	Z_2	Z_3
enabled	1	0	0	1	0	0	0
	1	0	1	0	1	0	0
	1	1	0	0	0	1	0
	1	1	1	0	0	0	1
disabled	0	x	x	0	0	0	0

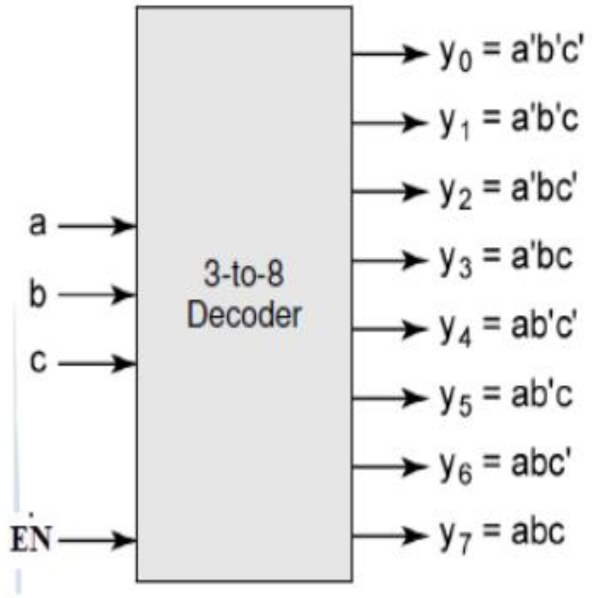
Logic Diagram



3 to 8 Decoder

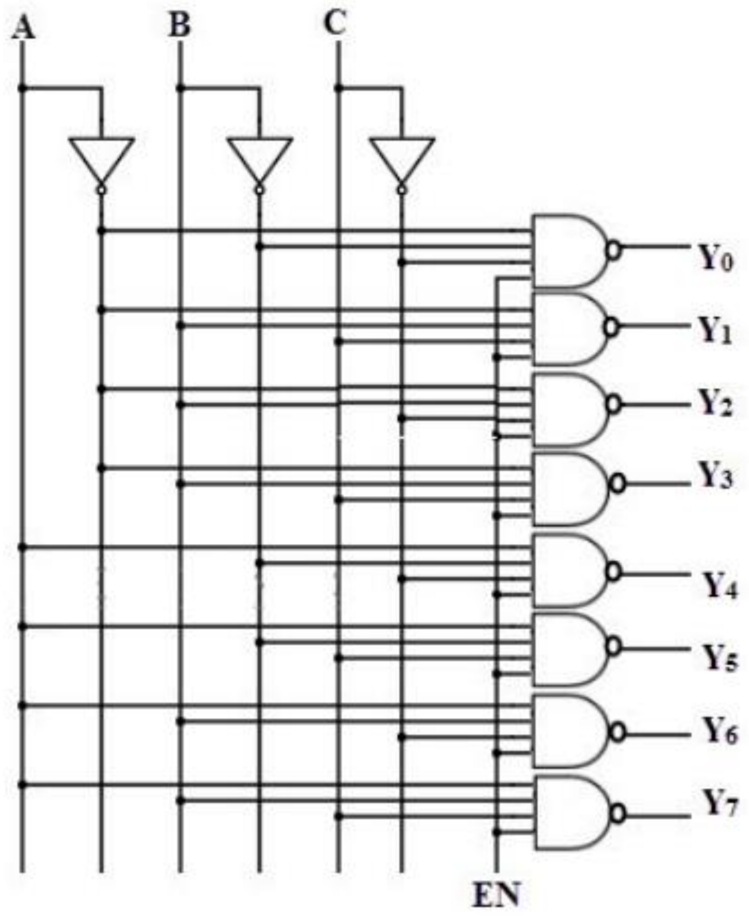
It has 3 inputs and 23 = 8 outputs.

Truth table and block diagram



Inputs				Outputs							
EN	A	B	C	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Logic Diagram



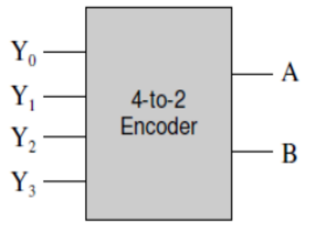
Encoders

An Encoder is a combinational circuit that performs the reverse operation of Decoder. It has maximum of 2^n input lines and ‘n’ output lines. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with ‘n’ bits. It is optional to represent the enable signal in encoders.

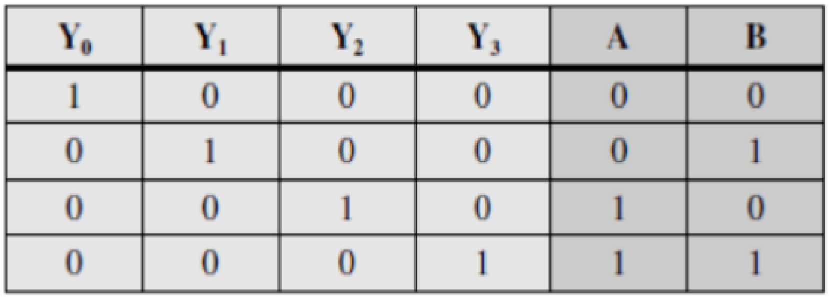
4 to 2 Encoder

Let 4 to 2 Encoder has four inputs Y3, Y2, Y1 & Y0 and two outputs A & B.

Block diagram

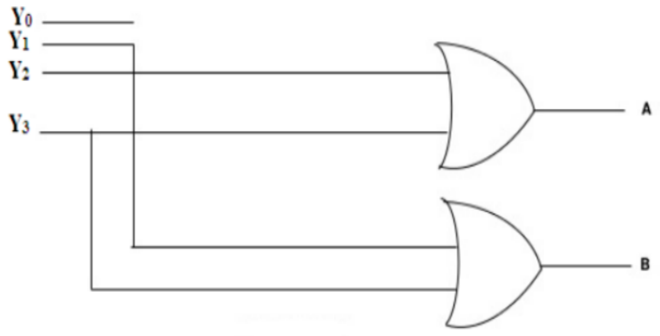


Truth Table



Y_0	Y_1	Y_2	Y_3	A	B
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Logic Diagram



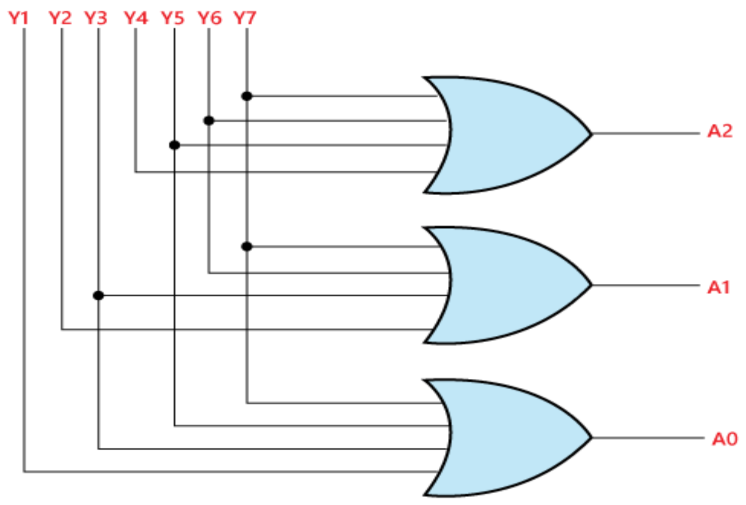
8-to-3-line Encoder:

The 8-to-3-line Encoder is also known as Octal to Binary Encoder. In 8-to-3-line encoder, there is a total of eight inputs, i.e., Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 and three outputs, i.e., A_0, A_1, and A_2. In 8-input lines, one input-line is set to true at a time to get the respective binary code in the output side. Below are the block diagram and the truth table of the 8-to-3-line encoder.

Truth Table

INPUTS								OUTPUTS		
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Circuit diagram



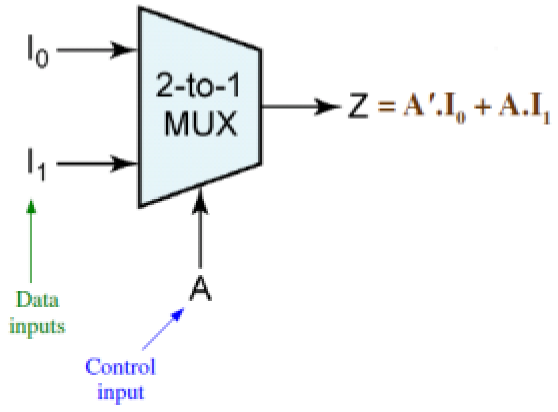
Multiplexer (Mux)

- Multiplexer is a combinational circuit that selects binary information from one of many inputs and directs it into single output.
- The selection of particular input is controlled by a set of selection line
- Multiplexer has 2^n inputs, n select line (control input) and one output line.
- It also called as Data selector

2 to 1 Multiplexer

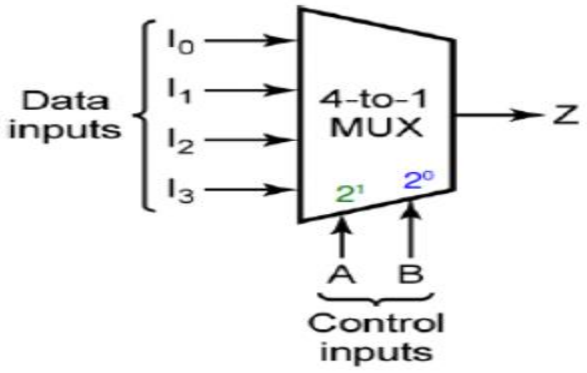
It has 2^1 inputs, 1 select line and one output

Circuit diagram



4 to 1 MUX

4 to 1 MUX has 2^2= 4 inputs, 2 select line and one output



Truth table

A	B	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$Z = A'.B'.I_0 + A'.B.I_1 + A.B'.I_2 + A.B.I_3$$

Introduction to sequential circuits

Sequential circuits are digital circuits that store and use the previous state information to determine their next state. Unlike combinational circuits, which only depend on the current input values to produce outputs, sequential circuits depend on both the current inputs and the previous state stored in memory elements.

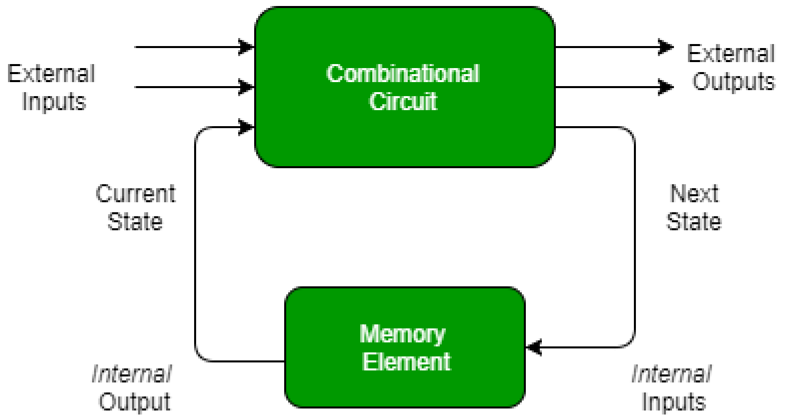


Figure: Sequential Circuit

Types of Sequential Circuits:
There are two types of sequential circuits:

Type 1: Asynchronous sequential circuit: These circuits **do not use a clock signal** but uses the pulses of the inputs. These circuits are **faster** than synchronous sequential circuits because there is clock pulse and change their state immediately when there is a change in the input signal. We use asynchronous sequential circuits when speed of operation is important and **independent** of internal clock pulse.

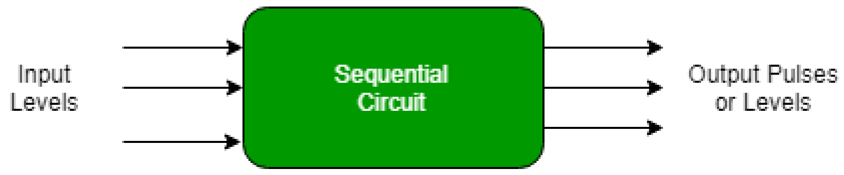


Figure: Asynchronous Sequential Circuit

But these circuits are more **difficult** to design and their output is **uncertain**.

Type2: Synchronous sequential circuit: These circuits **use clock signal** and level inputs (or pulsed) (with restrictions on pulse width and circuit propagation). The output pulse is the same duration as the clock pulse for the clocked sequential circuits. Since they wait for the next clock pulse to arrive to perform the next operation, so these circuits are bit **slower** compared to asynchronous. Level output changes state at the start of an input pulse and remains in that until the next input or clock pulse.

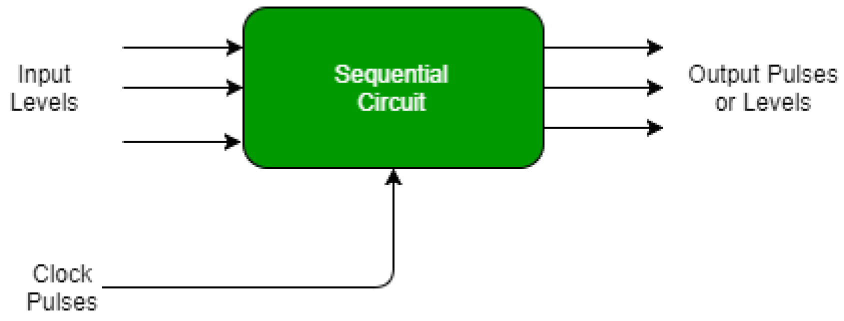


Figure: Synchronous Sequential Circuit

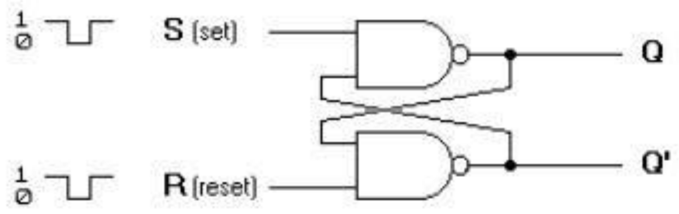
Latches

Latches are digital circuits that store a single bit of information and hold its value until it is updated by new input signals. They are used in digital systems as temporary storage elements to store binary information. Latches can be implemented using various digital logic gates, such as AND, OR, NOT, NAND, and NOR gates.

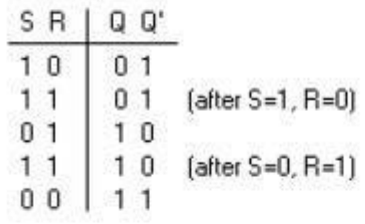
SR Latch

S-R latches i.e., Set-Reset latches are the simplest form of latches and are implemented using two inputs: S (Set) and R (Reset). The S input sets the output to 1, while the R input resets the output to 0. When both S and R inputs are at 1, the latch is said to be in an "undefined" state. They are also known as preset and clear states. The SR latch forms the basic building blocks of all other types of flip-flops.

Truth Table of SR Latch



(a) Logic diagram



S R	Q Q'	
1 0	0 1	
1 1	0 1	(after S=1, R=0)
0 1	1 0	
1 1	1 0	(after S=0, R=1)
0 0	1 1	

(b) Truth table

Flip-Flop
The flip-flop is a circuit that maintains a state until directed by input to change the state. A basic flip-flop can be constructed using four-NAND or four-NOR gates. Flip-flop is popularly known as the basic digital memory circuit. It has its two states as logic 1(High) and logic 0(low) states.

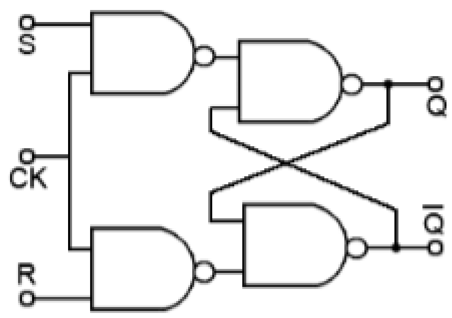
Types of Flip-Flops
Given Below are the Types of Flip-flops

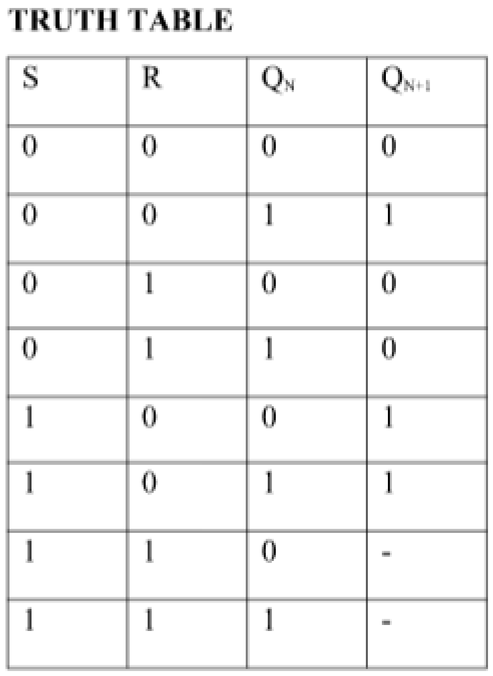
- SR Flip Flop
- JK Flip Flop
- D Flip Flop
- T Flip Flop

S-R Flip Flop

It is a Flip Flop with two inputs, one is S and the other is R. S here stands for Set and R here stands for Reset. Set basically indicates set the flip flop which means output 1 and reset indicates resetting the flip flop which means output 0. Here, a clock pulse is supplied to operate this flip-flop, hence it is a clocked flip-flop.

Circuit Diagram and Truth Table of S-R Flip Flop

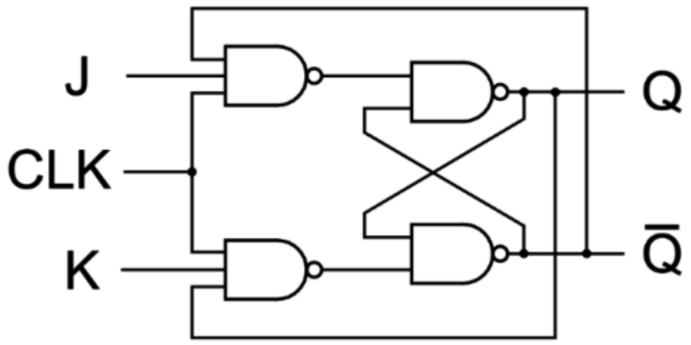



TRUTH TABLE

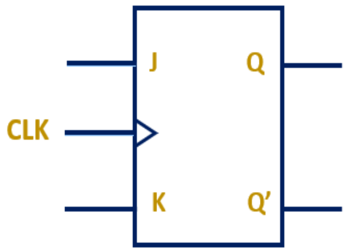
S	R	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

JK flip flop

The JK flip flop is one of the most used flip flops in digital circuits. The JK flip flop is a universal flip flop having two inputs 'J' and 'K'. The JK flip flop work in the same way as the SR flip flop work. The JK flip flop has 'J' and 'K' flip flop instead of 'S' and 'R'. The only difference between JK flip flop and SR flip flop is that when both inputs of SR flip flop is set to 1, the circuit produces the invalid states as outputs, but in case of JK flip flop, there are no invalid states even if both 'J' and 'K' flip flops are set to 1.



Symbol

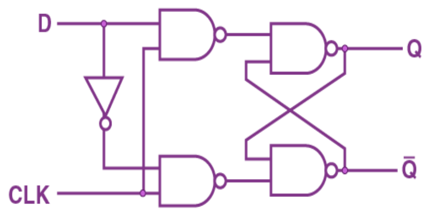


Truth Table

CLK	J	K	Q_{n+1}
↑	0	0	Q_n
↑	0	1	0
↑	1	0	1
↑	1	1	$Q_n’$

D Flip Flop

D flip flop is known as “delay flip flop” or “data flip flop” which is used to store single bit of data. The D flip flop has two inputs, data and clock input which controls the flip flop. when clock input is high, the data is transferred to the output of the flip flop and when the clock input is low, the output of the flip flop is held in its previous state.



Truth Table

Q	D	$Q_{(t+1)}$
0	0	0
0	1	1
1	0	0
1	1	1

Working of D Flip Flop

D flip flop consist of a single input D and two outputs (Q and Q'). The basic working of D Flip Flop is as follows:

- When the clock signal is low, the flip flop holds its current state and ignores the D input.
- When the clock signal is high, the flip flop samples and stores D input.
- The value that was previously fed into the D input is reflected at the flip flop's Q output.
    - If D = 0 then Q will be 0.
    - If D = 1 then Q will be 1.
- The Q' output of the flip flop is complemented by the Q output.
    - If Q = 0 then Q' will be 1.
    - If Q = 1 then Q' will be 0

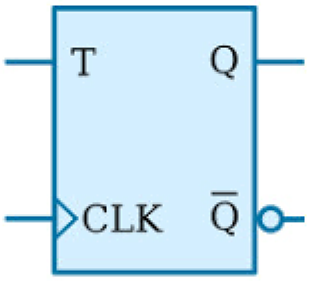
T Flip Flop

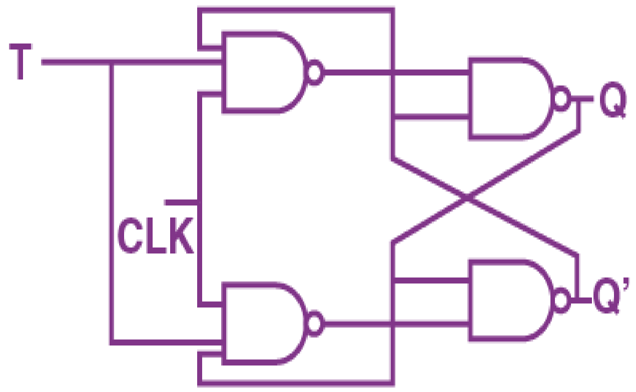
In T flip flop, "T" defines the term "Toggle". In SR Flip Flop, we provide only a single input called "Toggle" or "Trigger" input to avoid an intermediate state occurrence.

Now, this flip-flop work as a Toggle switch. The next output state is changed with the complement of the present state output. This process is known as "Toggling"'.

We can construct the "T Flip Flop" by making changes in the "JK Flip Flop". The "T Flip Flop" has only one input, which is constructed by connecting the input of JK flip flop. This single input is called T. In simple words, we can construct the "T Flip Flop" by converting a "JK Flip Flop". Sometimes the "T Flip Flop" is referred to as single input "JK Flip Flop".

Block diagram of the "T-Flip Flop" is given where T defines the "Toggle input", and CLK defines the clock signal input.





Truth Table

T	Q_N	Q_{N+1}
0	0	0
0	1	1
1	0	1
1	1	0

.