

Types of digital circuits

- **Combinational Logic Circuits**

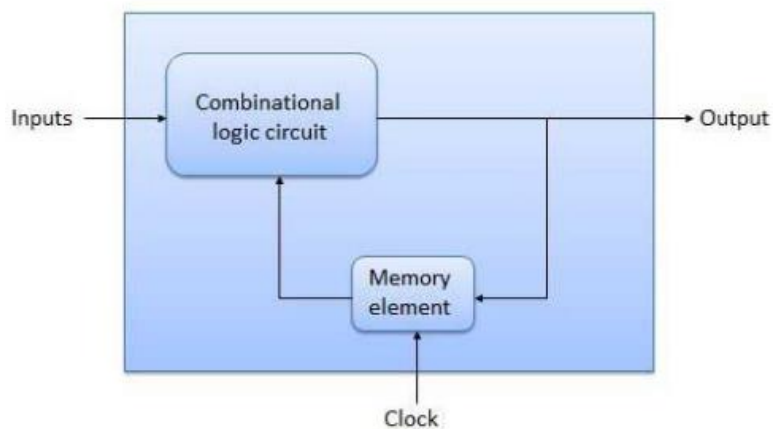
Combinational circuit is a circuit in which we combine the different gates in the Circuit, for example half adder, full adder encoder, decoder, multiplexer and DE multiplexer. Some of the characteristics of combinational circuits are following:

1. The output of combinational circuit at any instant of time, depends only on the Levels present at input terminals.
2. The combinational circuit do not use any memory. The previous state of input Does not have any effect on the present state of the circuit.
3. A combinational circuit can have an n number of inputs and m number of Outputs.



- **Sequential Logic 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, for example, Flip-Flops and counters has memory so output can vary based on input. This type of circuits uses previous input, output, clock and a memory element.

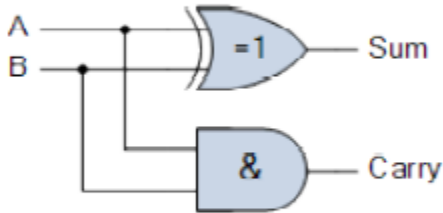


Half Adder and Full Adder Circuit

Half Adder: A half-adder adds two bits and produces a sum and an output carry. Half Adder: is a combinational circuit that performs the addition of two bits, this circuit needs two binary inputs and two binary outputs. With the help of half adder, we can design circuits that are capable of performing simple addition with the help of logic gates.

Here the output '1' of '10' becomes the carry-out. The result is shown in a truth-table below. 'SUM' is the normal output and 'C-out is the carry-out.

Half Adder Truth Table with Carry-Out

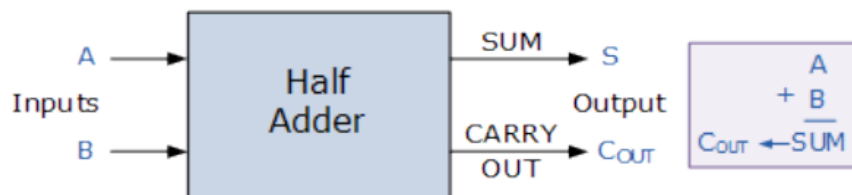
Symbol	Truth Table			
	B	A	C-out	SUM
	0	0	0	0
	0	1	0	1
	1	0	0	1
	1	1	1	0

From the truth table of the half adder we can see that the SUM (S) output is the result of the Exclusive-OR gate and the Carry-out (Cout) is the result of the AND gate. Then the Boolean expression for a half adder is as follows.

For the **SUM** bit $SUM = A \text{ XOR } B = A \oplus B$

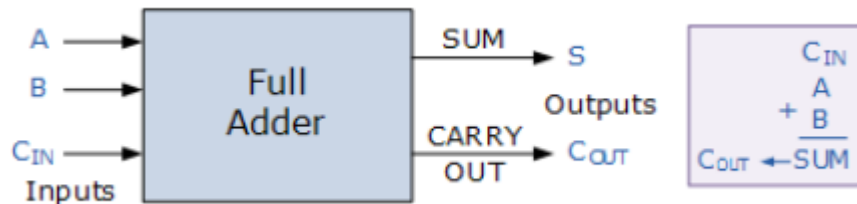
For the **CARRY** bit $C\text{-out} = A \text{ AND } B = A \cdot B$

From the equation it is clear that this 1-bit adder can be easily implemented with the help of EXOR Gate for the output 'SUM' and an AND Gate for the carry. Take a look at the implementation below.



Full Adder:

This type of adder is a little more difficult to implement than a half-adder. The main difference between a half-adder and a full-adder is that the full-adder has three inputs and two outputs. The first two inputs are A and B and the third input is an input carry Designated as CIN. When a full adder logic is designed we will be able to string eight of them together to create a byte-wide adder and cascade the carry bit from one adder to the next.



Ex:

	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
\overline{A}	0	1	0	1
A	1	0	1	0

$$S = A \oplus B \oplus C_{in}$$

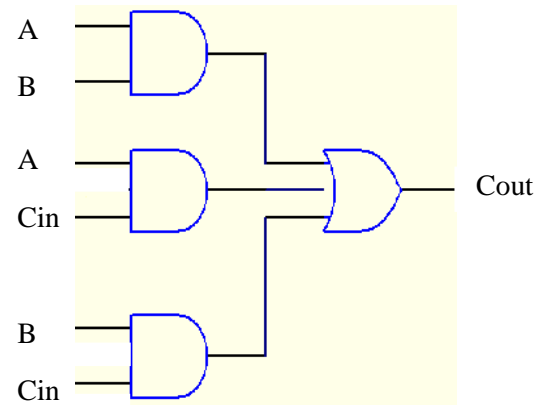
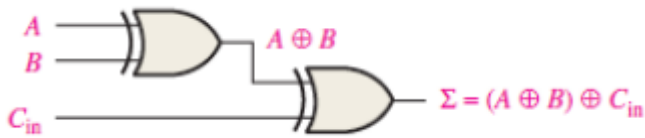
	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
\overline{A}	0	0	1	0
A	0	1	1	1

$$C_{out} = AB + AC + BC$$

A	B	Cin	Cout	S
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

Then the full adder is a logical circuit that performs an addition operation on three binary digits and just like the half adder, it also generates a carry out to the next addition column. Then a Carry-in is a possible carry from a less significant digit, while a Carryout represents a carry to a more significant digit. As the full adder circuit above is basically two half adders connected together, the truth table for the full adder includes an additional column to take into account the Carry-in, CIN input as well as the summed output, S and the Carry-out, COUT bit.

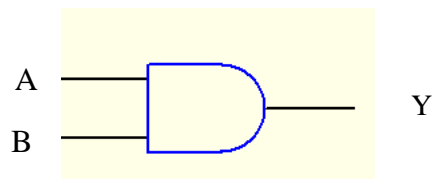
Logic required to form the sum of three bits



Logic circuit design:

Ex: Design a logic circuit to multiply two (1-bit) using minimum number of logic gates.

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

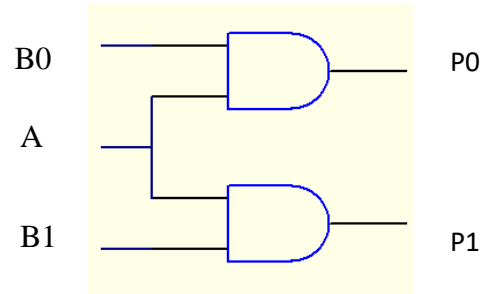


Ex: Design a logic circuit to multiply (2-bit) by (1-bit) number using minimum number of logic gates.

A	B1	B0	P1	P0
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

	$\overline{B}1\overline{B}0$	$\overline{B}1B0$	$B1\overline{B}0$	$B1B0$
\overline{A}	0	0	0	0
A	0	0	1	1

$P1 = AB1$



	$\overline{B}1\overline{B}0$	$\overline{B}1B0$	$B1\overline{B}0$	$B1B0$
\overline{A}	0	0	0	0
A	0	1	1	0

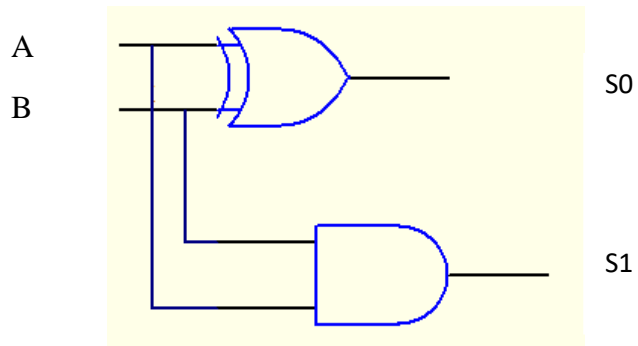
$P0 = AB0$

Ex: Design a logic circuit to add two numbers of (1-bit) each.

$$S0 = A \oplus B$$

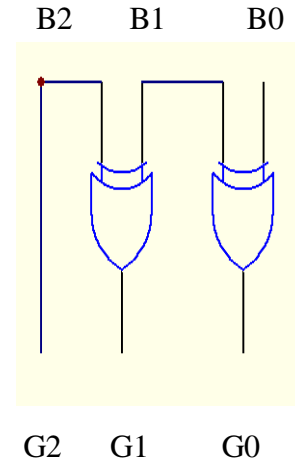
$$S1 = AB$$

A	B	S1	S0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Ex: Design a logic circuit to convert 3-bit from binary to gray.

B2	B1	B0	G2	G1	G0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

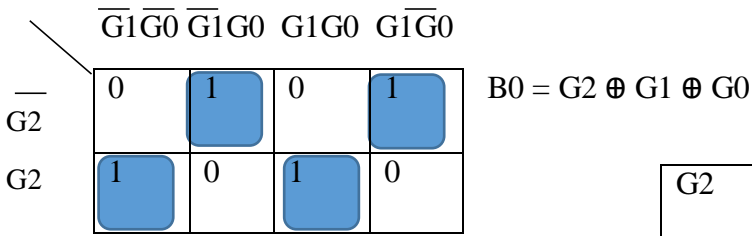


$\overline{B1}B0 \quad B1\overline{B0} \quad B1B0 \quad B1\overline{B0}$
 $\overline{B2}$ 0 1 0 1
 $B2$ 0 1 0 1
 $G0 = \overline{B1}B0 + B1\overline{B0}$
 $= B1 \oplus B0$

$\overline{B1}B0 \quad B1\overline{B0} \quad B1B0 \quad B1\overline{B0}$
 $\overline{B2}$ 0 0 1 1
 $B2$ 1 1 0 0
 $G1 = B2\overline{B1} + \overline{B2}B1$
 $= B1 \oplus B2$

$\overline{B1}B0 \quad B1\overline{B0} \quad B1B0 \quad B1\overline{B0}$
 $\overline{B2}$ 0 0 0 0
 $B2$ 1 1 1 1
 $G2 = B2$

Ex: Design a logic circuit to convert 3-bit from gray to binary.



$G2$	$G1$	$G0$	$B2$	$B1$	$B0$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	0	1

