 The combinational circuit is time-independent. The output it generates does not depend on any of its previous inputs. On the other hand, sequential circuits are the ones that depend on clock cycles. They depend entirely on the past as well as the present inputs for generating output.

What is a Combinational Circuit?

* The output of a Combinational Circuit depends entirely on the present input.
* It exhibits a faster speed.
* It is comparatively easier to design.
* No feedback is present between the input and output.
* The combinational circuit depends on time.
* Logic gates form the building blocks of such circuits.
* One can make use of it for both boolean and arithmetic operations.
* They don’t hold the capacity of storing any state.
* These circuits do not have a clock- thus, they don’t require triggering.
* They do not possess any memory element.
* Users can feasibly use as well as handle them.
* Example –Decoder, Encoder, [Multiplexer, demultiplexer](https://unstop.com/blog/difference-between-multiplexer-and-demultiplexer), half-adder, full-adder etc.

What is a Sequential Circuit?

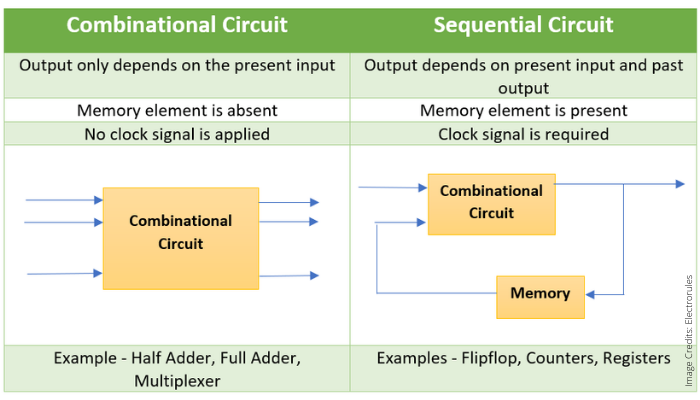
* The output of a Sequential Circuit depends on both- past as well as present inputs.
* It works at a comparatively slower speed.
* The design of these circuits is comparatively much tougher than the Combinational Circuit.
* A feedback path exists between the output and the input.
* The circuit is time-dependent.
* Flip-flops constitute the building blocks of such a circuit.
* People mainly use them for storing data and information.
* They possess the capability of storing any data state or retaining an earlier state at any given point.
* Because a Sequential circuit depends on a clock, it usually requires triggering.
* They always possess a memory element.
* A user may not be able to handle and use these circuits easily.

For Example:

Flip-flops like SR flip-flop, JK flip-flop, T flip-flop, registers, counters, etc.

Difference Between Combinational and Sequential Circuit

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Combinational Circuit** | **Sequential Circuit** |
| Meaning and Definition | It is a type of circuit that generates an output by relying on the input it receives at that instant, and it stays independent of time. | It is a type of circuit in which the output does not only rely on the current input. It also relies on the previous ones. |
| Feedback | A Combinational Circuit requires no feedback for generating the next output. It is because its output has no dependency on the time instance. | The output of a Sequential Circuit, on the other hand, relies on both- the previous feedback and the current input. So, the output generated from the previous inputs gets transferred in the form of feedback. The circuit uses it (along with inputs) for generating the next output. |
| Performance | We require the input of only the current state for a Combinational Circuit. Thus, it performs much faster and better in comparison with the Sequential Circuit. | In the case of a Sequential Circuit, the performance is very slow and also comparatively lower. Its dependency on the previous inputs makes the process much more complex. |
| Complexity | It is very less complex in comparison. It is because it basically lacks implementation of feedback. | This type of circuit is always more complex in its nature and functionality. It is because it implements the feedback, depends on previous inputs and also on clocks. |
| Elementary Blocks | Logic gates form the building/ elementary blocks of a Combinational Circuit. | Flip-flops form the building/ elementary blocks of a Sequential Circuit. |
| Operation | One can use these types of circuits for both- Boolean as well as Arithmetic operations. | You can mainly make use of these types of circuits for storing data. |



**What is an Adder?**

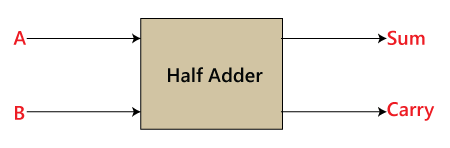
An adder is a [digital logic circuit](https://www.elprocus.com/different-types-of-digital-logic-circuits/) in electronics that is extensively used for the addition of numbers. In many computers and other types of processors, adders are even used to calculate addresses and related activities and calculate table indices in the ALU and even utilized in other parts of the processors. These can be built for many numerical representations like excess-3 or binary coded decimal. Adders are basically classified into two types: Half Adder and Full Adder.

**Half Adder (HA):**

Half adder is the simplest of all adder circuits. Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit (s) and carry bit (c) both as output. The addition of 2 bits is done using a combination circuit called a Half adder. The input variables are augend and addend bits and output variables are sum & carry bits. A and B are the two input bits.

let us consider two input bits A and B, then sum bit (s) is the X-OR of A and B. it is evident from the function of a half adder that it requires one X-OR gate and one AND gate for its construction.

### **Block diagram**

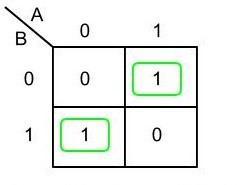
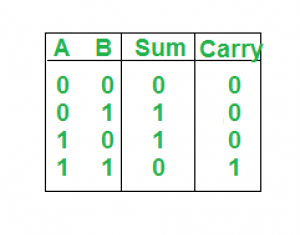


In the above table,

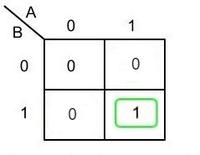
1. 'A' and' B' are the input states, and 'sum' and 'carry' are the output states.
2. The carry output is 0 in case where both the inputs are not 1.
3. The least significant bit of the sum is defined by the 'sum' bit.

The SOP form of the sum and carry are as follows:

Sum = x'y+xy'  
Carry = xy

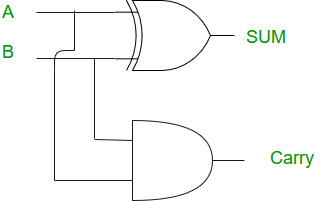


**Sum = A XOR B**



**Carry = A AND B**

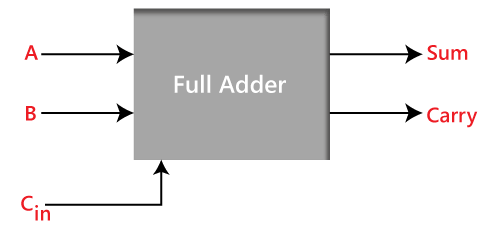
### Implementation:



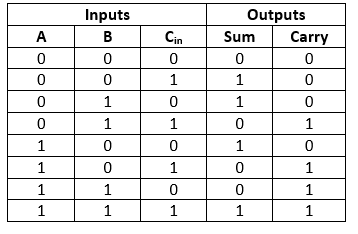
# **Full Adder**

The half adder is used to add only two numbers. To overcome this problem, the full adder was developed. The full adder is used to add three 1-bit binary numbers A, B, and carry C. The full adder has three input states and two output states i.e., sum and carry.

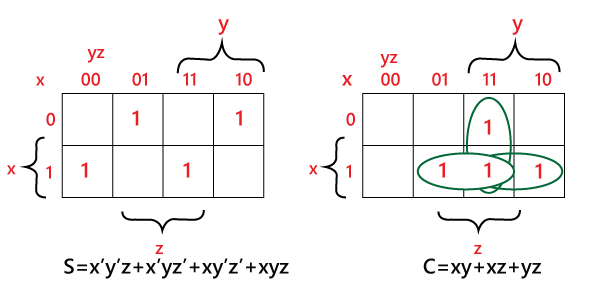
### **Block diagram**



### **Truth Table**



In the above table,

1. 'A' and' B' are the input variables. These variables represent the two significant bits which are going to be added
2. 'Cin' is the third input which represents the carry. From the previous lower significant position, the carry bit is fetched.
3. The 'Sum' and 'Carry' are the output variables that define the output values.
4. The eight rows under the input variable designate all possible combinations of 0 and 1 that can occur in these variable
5. The SOP form can be obtained with the help of K-map as:
6. 
7. Sum = x' y' z+x' yz+xy' z'+xyz  
   Carry = xy+xz+yz

### **Sum:**

* Perform the XOR operation of input A and B.
* Perform the XOR operation of the outcome with carry. So, the sum is (A XOR B) XOR Cin which is also represented as:  
  (A ⊕ B) ⊕ Cin

### **Carry:**

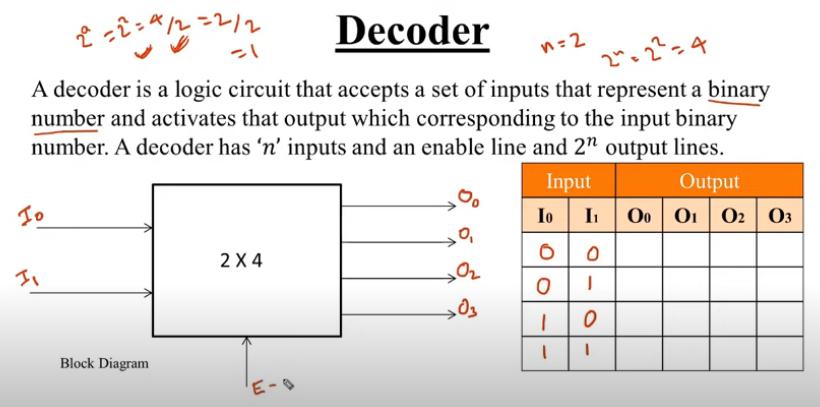
1. Perform the 'AND' operation of input A and B.
2. Perform the 'XOR' operation of input A and B.
3. Perform the 'OR' operations of both the outputs that come from the previous two steps. So the 'Carry' can be represented as:  
   A.B + (A ⊕ B)

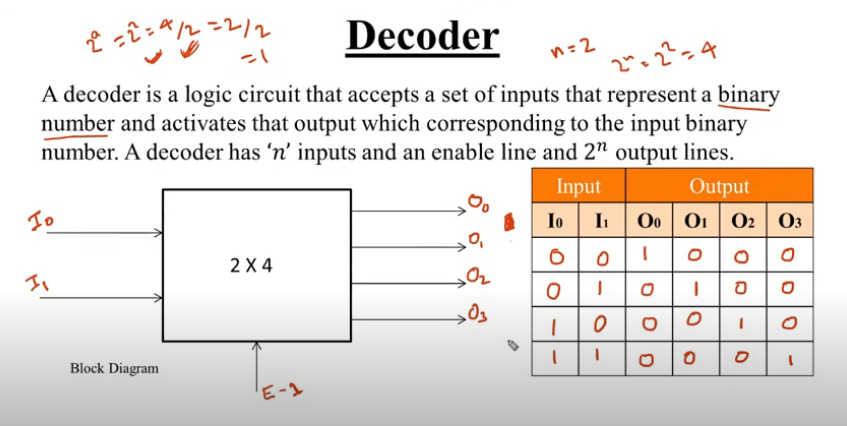
 encoder and decoder. Both are combinational logic circuits. A combinational logic circuit is a circuit that contains different types of logic gates. There are **n** numbers of inputs, and **m** numbers of outputs are possible in a combinational logic circuit.

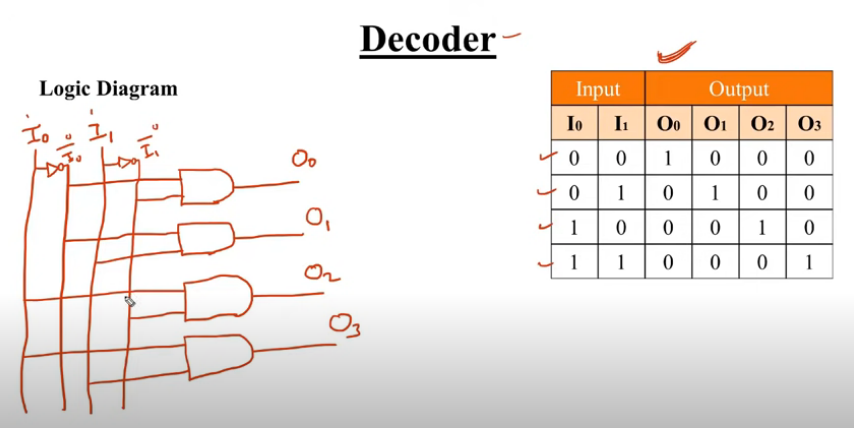
## Encoder

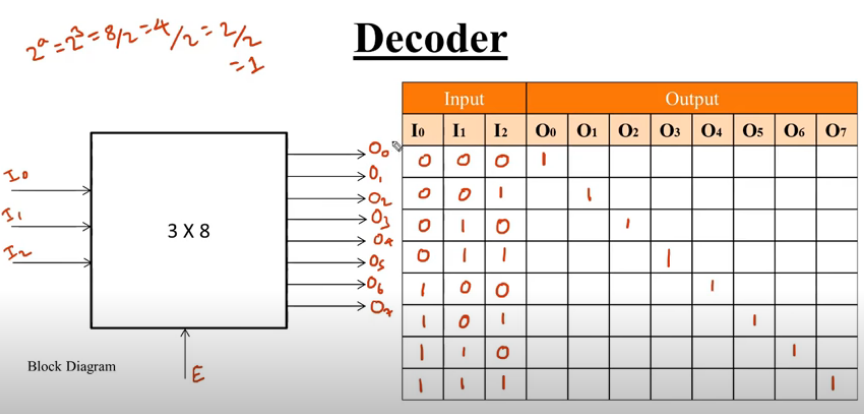
An encoder can also be described as a combinational circuit that has a maximum of 2^n (or less) input lines and n output lines.

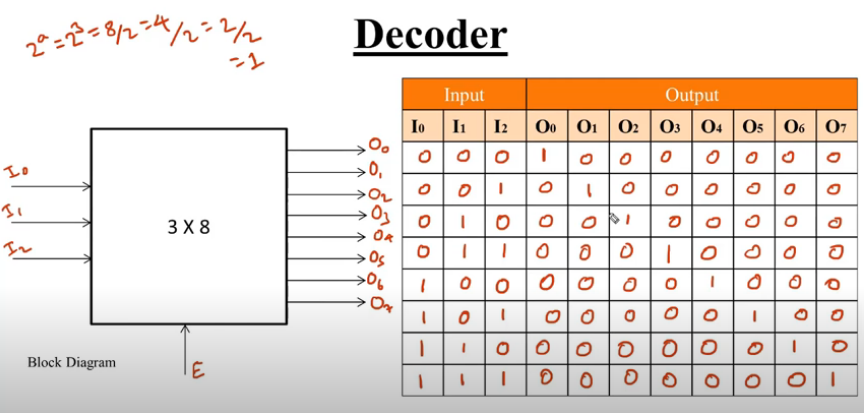
Encoders are used to convert a decimal number into a binary number. The objective is to perform a binary operation such as addition, subtraction, multiplication, etc. The basic logic element that is used in encoders is **OR gate.** The operation of the encoder is fairly simple. Encoders are commonly used in Videos, E-mail, etc.





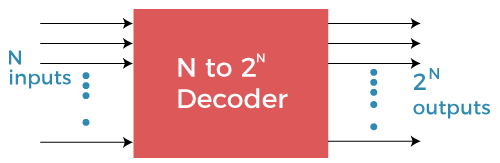






**Decoders –** A decoder does the opposite job of an encoder. It is a combinational circuit that converts n lines of input into 2n lines of output.

Let’s take an example of 3-to-8 line decoder.



In the **2 to 4 line decoder**, there are a total of three inputs, i.e., **A0, A1,** and **E** and four outputs, i.e., **Y0, Y1, Y2,** and **Y3**.

In **3 to 8 line decoder**, there are a total of eight outputs, i.e., **Y0, Y1, Y2, Y3, Y4, Y5, Y6,** and **Y7** and three inputs, i.e., **A0, A1,** and **A2**. It is also called as Binary to octal decoder.

The commonly used or most preferred decoders are **n to m decoders,** where m<= 2^n. It is also referred to as **n \* m decoder** that has n inputs and m outputs.

**Truth Table –**

| X | Y | Z | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Implementation –**  
D0 is high when X = 0, Y = 0 and Z = 0. Hence,

D0 = X’ Y’ Z’

Similarly,

D1 = X’ Y’ Z

D2 = X’ Y Z’

D3 = X’ Y Z

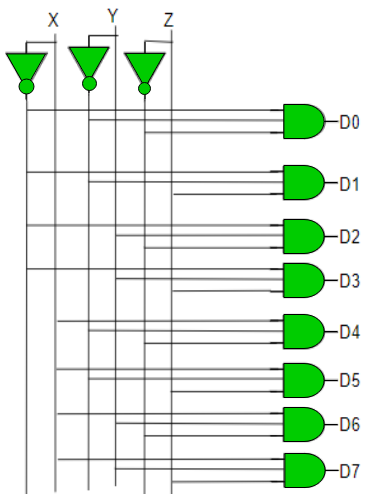
D4 = X Y’ Z’

D5 = X Y’ Z

D6 = X Y Z’

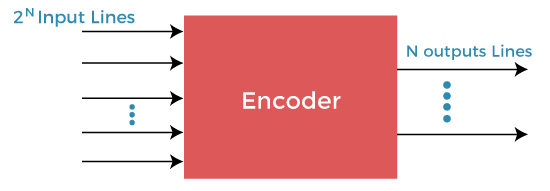
D7 = X Y Z

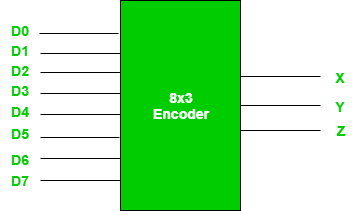
Hence,



**Encoders –**  
An encoder is a combinational circuit that converts binary information in the form of a 2N input lines into N output lines, which represent N bit code for the input. For simple encoders, it is assumed that only one input line is active at a time.

As an example, let’s consider **Octal to Binary** encoder. As shown in the following figure, an octal-to-binary encoder takes 8 input lines and generates 3 output lines.





In **4 to 2 line encoder**, there are a total of four inputs, i.e., **Y0, Y1, Y2,** and **Y3**, and two outputs, i.e., **A0** and **A1**.

In **8 to 3 line encoder**, there are a total of eight inputs, i.e., **Y0, Y1, Y2, Y3, Y4, Y5, Y6,** and **Y7** and three outputs, i.e., **A0, A1,** and **A2**. It is also known as Octal to Binary Encoder.

**Truth Table –**

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | X | Y | Z |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

As seen from the truth table, the output is 000 when D0 is active; 001 when D1 is active; 010 when D2 is active and so on.

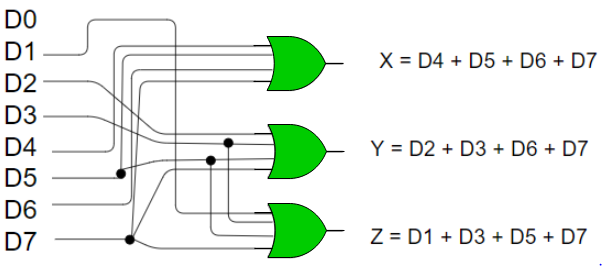
**Implementation –**  
From the truth table, the output line Z is active when the input octal digit is 1, 3, 5 or 7. Similarly, Y is 1 when input octal digit is 2, 3, 6 or 7 and X is 1 for input octal digits 4, 5, 6 or 7. Hence, the Boolean functions would be:

X = D4 + D5 + D6 + D7

Y = D2 +D3 + D6 + D7

Z = D1 + D3 + D5 + D7

Hence, the encoder can be realised with OR gates as follows:



**Truth Table –**

| D3 | D2 | D1 | D0 | X | Y | 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 |