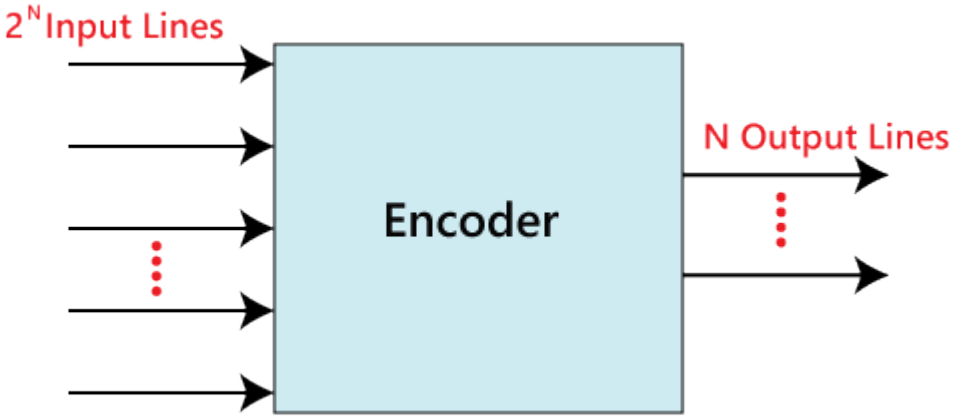
# ENCODERS

The combinational circuits that change the binary information into N output lines are known as **Encoders**. The binary information is passed in the form of $2^N$ input lines. The output lines define the N-bit code for the binary information. In simple words, the **Encoder** performs the reverse operation of the **Decoder**. It encodes the information from 2^n inputs into an n-bit code.

At a time, only one input line is activated for simplicity. The produced N-bit output code is equivalent to the binary information.




There are various types of encoders which are as follows:

## 4 to 2 line Encoder:

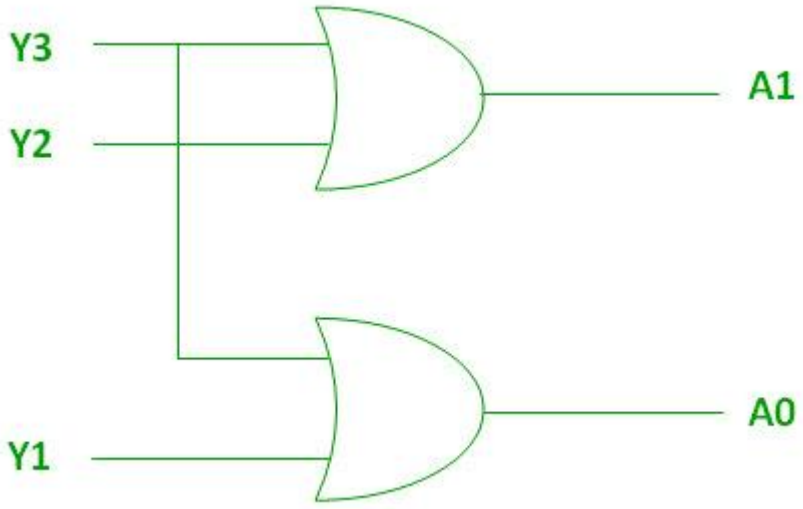
The 4 to 2 Encoder consists of **four inputs Y3, Y2, Y1 & Y0 and two outputs A1 & A0**. At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output. The figure below shows the logic symbol of 4 to 2 encoder :

| INPUTS | | | | OUTPUTS | |
|---|---|---|---|---|---|
| Y3 | Y2 | Y1 | Y0 | A1 | A0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |

**Logical expression for A1 and A0 :**

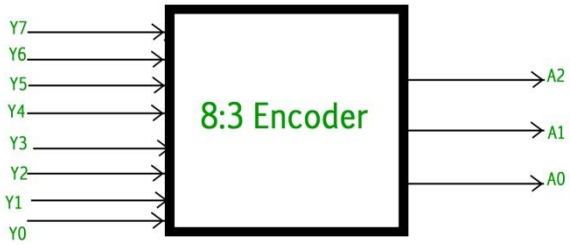
```
A1 = Y3 + Y2

A0 = Y3 + Y1
```

The above two Boolean functions A1 and A0 can be implemented using two input OR gates :




## 8 : 3 Encoder (Octal to Binary) –

The 8 to 3 Encoder or octal to Binary encoder consists of **8 inputs** : Y7 to Y0 and **3 outputs** : A2, A1 & A0. Each input line corresponds to each octal digit and three outputs generate corresponding binary code.
The figure below shows the logic symbol of octal to binary encoder:




| INPUTS | | | | | | | | OUTPUTS | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Y7 | Y6 | Y5 | Y4 | Y3 | Y2 | Y1 | Y0 | A2 | A1 | A0 |
| 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 |

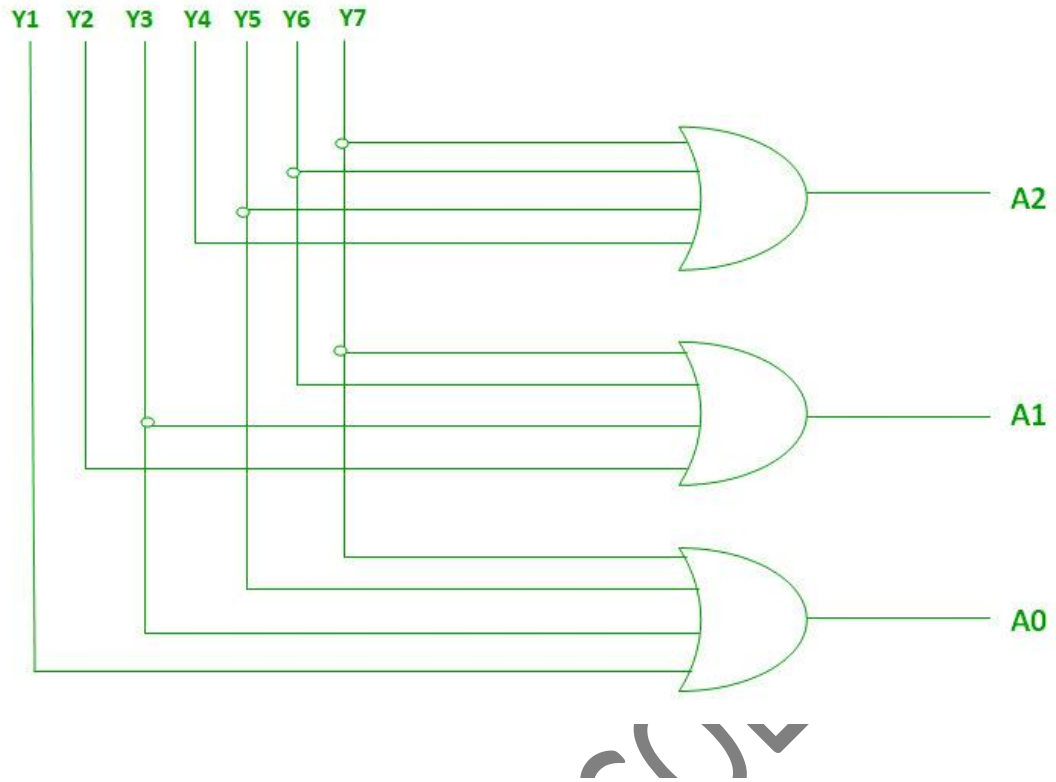
**Logical expression for A2, A1 and A0 :**

A2 = Y7 + Y6 + Y5 + Y4

A1 = Y7 + Y6 + Y3 + Y2

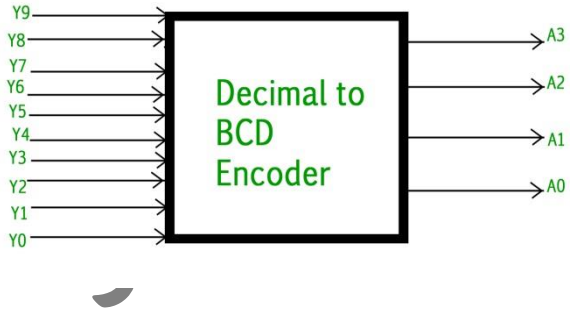
A0 = Y7 + Y5 + Y3 + Y1

The above two Boolean functions A2, A1 and A0 can be implemented using four input OR gates :




## Decimal to BCD Encoder –

The decimal to binary encoder usually consists of **10 input lines** and **4 output lines**. Each input line corresponds to the each decimal digit and 4 outputs correspond to the BCD code. This encoder accepts the decoded decimal data as an input and encodes it to the BCD output which is available on the output lines. The figure below shows the logic symbol of decimal to BCD encoder :



| INPUTS | | | | | | | | | | OUTPUTS | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Y9 | Y8 | Y7 | Y6 | Y5 | Y4 | Y3 | Y2 | Y1 | Y0 | A3 | A2 | A1 | A0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Logical expression for A3, A2, A1 and A0 :**

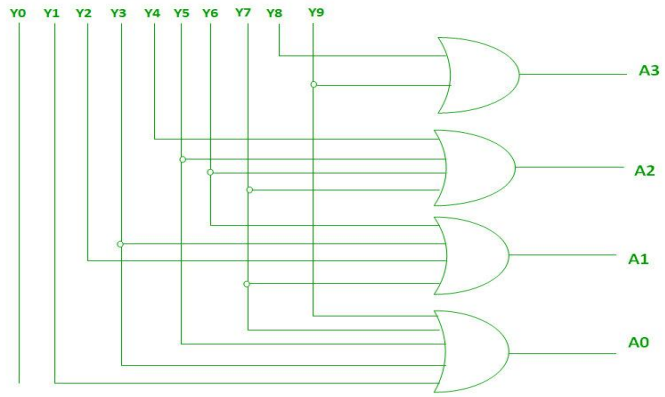
A3 = Y9 + Y8

A2 = Y7 + Y6 + Y5 +Y4

A1 = Y7 + Y6 + Y3 +Y2

A0 = Y9 + Y7 +Y5 +Y3 + Y1

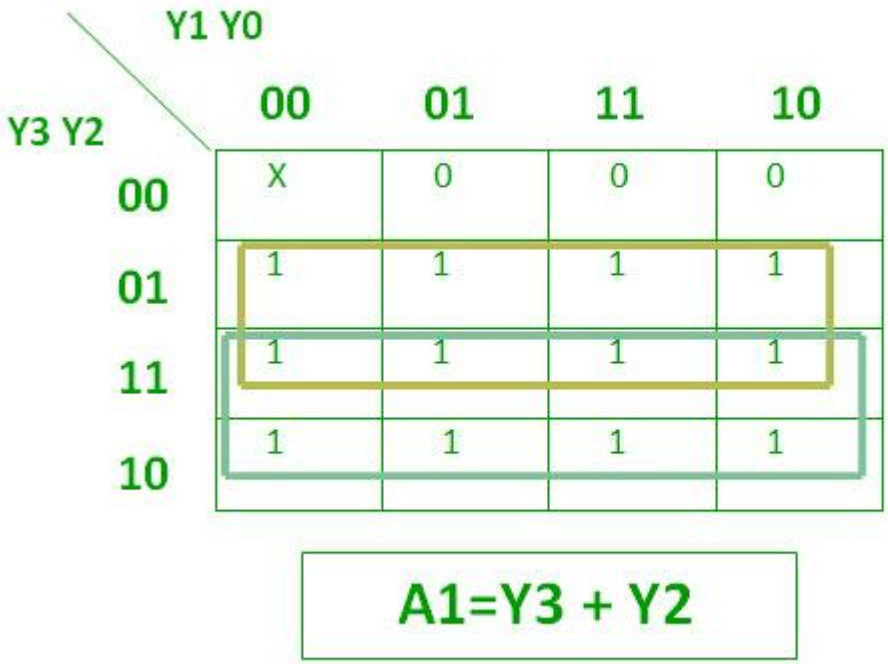
The above two Boolean functions can be implemented using OR gates :

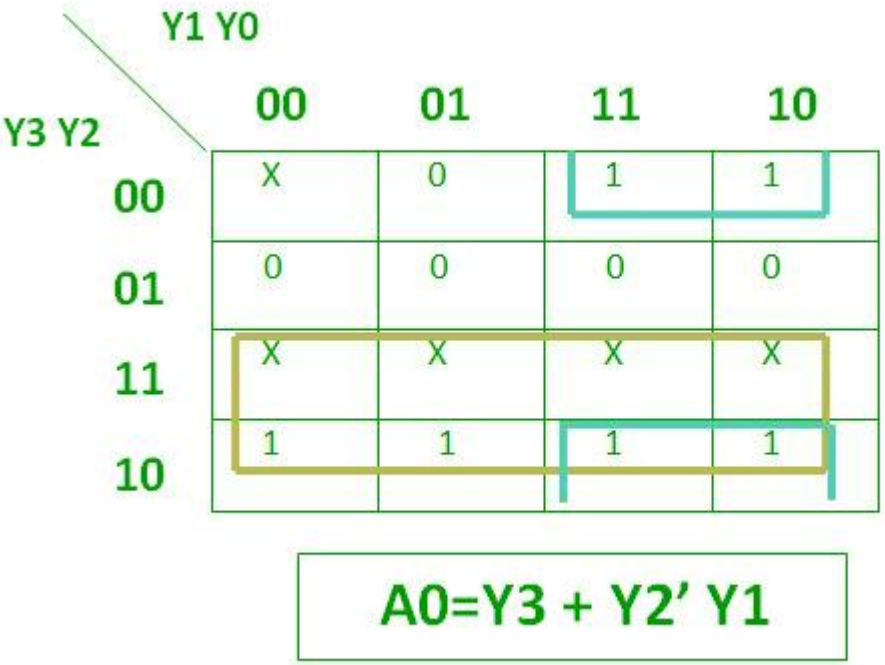


## Priority Encoder –

A 4 to 2 priority encoder has **4 inputs** : Y3, Y2, Y1 & Y0 and **2 outputs** : A1 & A0. Here, the input, Y3 has the **highest priority**, whereas the input, Y0 has the **lowest priority**. In this case, even if more than one input is '1' at the same time, the output will be the (binary) code corresponding to the input, which is having **higher priority**.
The truth table for priority encoder is as follows :

| INPUTS | | | | OUTPUTS | | |
|---|---|---|---|---|---|---|
| Y3 | Y2 | Y1 | Y0 | A1 | A0 | 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 |






The above two Boolean functions can be implemented as :