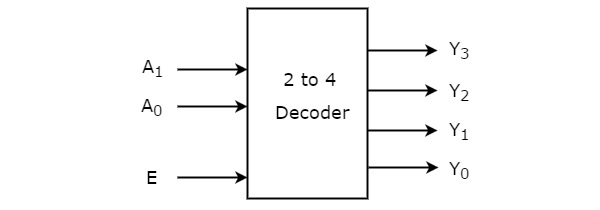
**AIM:-** Implementing 3-8 line Decoder.

**Theory:-** Decoder is a combinational circuit that has ‘n’ input lines and maximum of 2n output lines. One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code. The outputs of the decoder are nothing but the **min terms** of ‘n’ input variables lines, when it is enabled.

## 2 to 4 Decoder

Let 2 to 4 Decoder has two inputs A1 & A0 and four outputs Y3, Y2, Y1 & Y0. The **block diagram** of 2 to 4 decoder is shown in the following figure.



One of these four outputs will be ‘1’ for each combination of inputs when enable, E is ‘1’. The **Truth table** of 2 to 4 decoder is shown below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Enable | Inputs | | Outputs | | | |
| E | A1 | A0 | Y3 | Y2 | Y1 | Y0 |
| 0 | x | X | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

From Truth table, we can write the **Boolean functions** for each output as

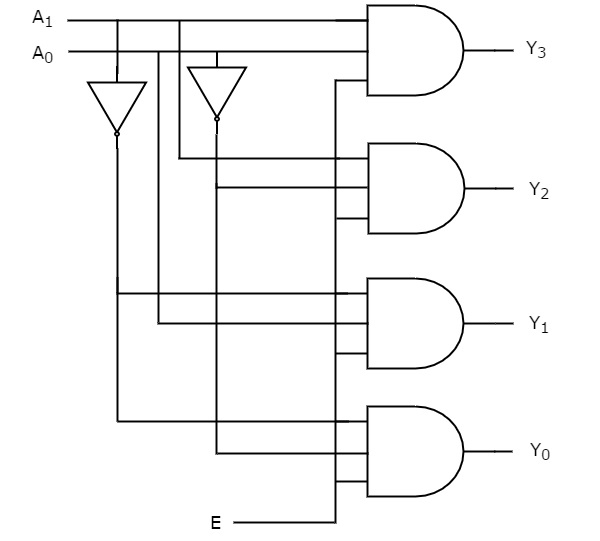
Y0=EA’1.A’0

Y1=EA’1.A0

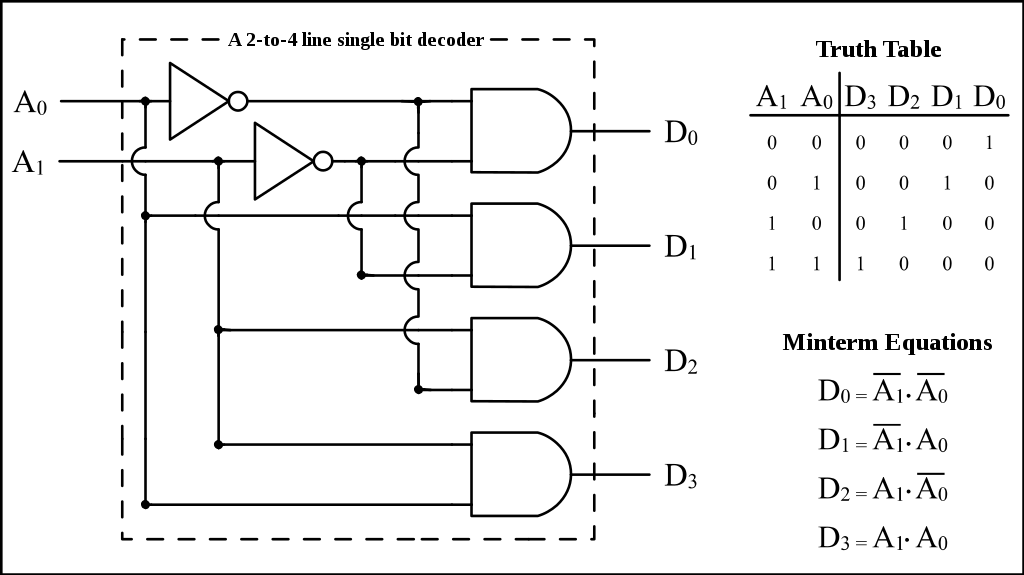
Y2=EA1.A’0

Y3=EA1.A0

Each output is having one product term. So, there are four product terms in total. We can implement these four product terms by using four AND gates having three inputs each & two inverters. The **circuit diagram** of 2 to 4 decoder is shown in the following figure.



By neglecting Enable bit we can design 2-4 Decoder as shown below-



Therefore, the outputs of 2 to 4 decoder are nothing but the **min terms** of two input variables A1 & A0, when enable, E is equal to one. If enable, E is zero, then all the outputs of decoder will be equal to zero.

Similarly, 3 to 8 decoder produces eight min terms of three input variables A2, A1 & A0 and 4 to 16 decoder produces sixteen min terms of four input variables A3, A2, A1 & A0.

### **3 to 8 Decoder**

Similarly 3 to 8 Decoder has three inputs A2, A1 & A0 and eight outputs, Y7 to Y0.

We can find the number of lower order decoders required for implementing higher order decoder using the following formula.

The **Truth table** of 3 to 8 decoder is shown below.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Enable | Inputs | | | Outputs | | | | | | | |
| E | A2 | A1 | A0 | Y7 | Y6 | Y5 | Y4 | Y3 | Y2 | Y1 | Y0 |
| 0 | X | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 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 |

From Truth table, we can write the **Boolean functions** for each output as

Y0=E A’2.A’1.A’0

Y1=E A’2A’1.A0

Y2=E A’2A1.A’0

Y3=E A’2A1.A0

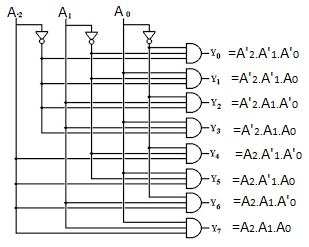
Y4=E A2A’1.A’0

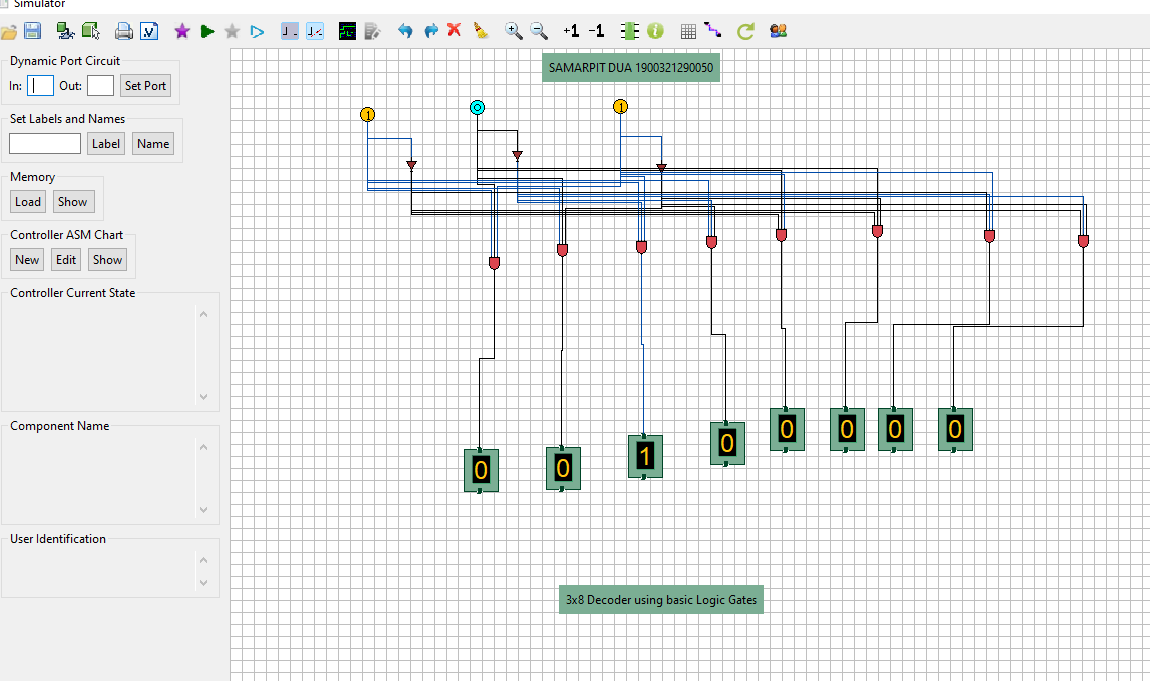
Y5=E A2A’1.A0

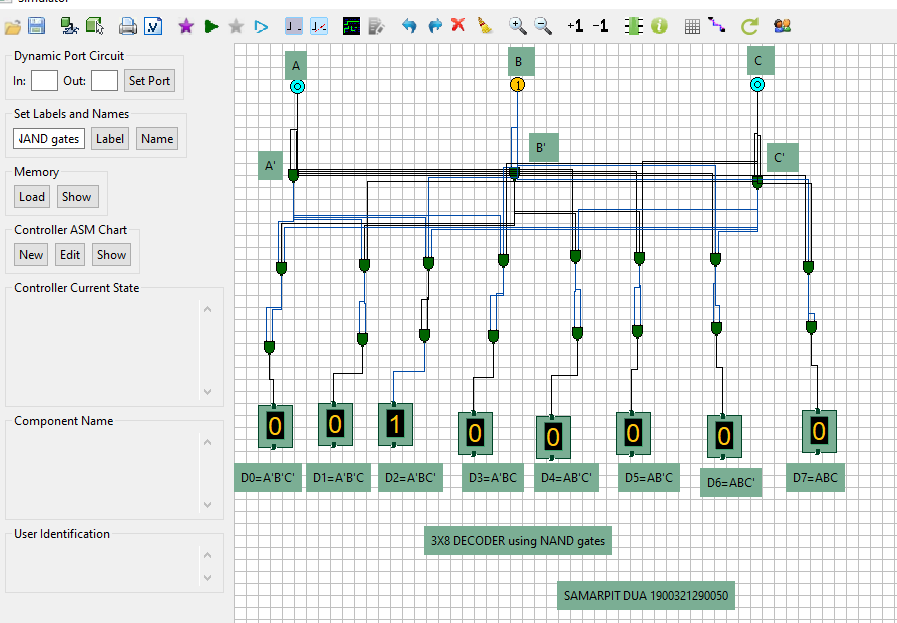
Y6=E A2A1.A’0

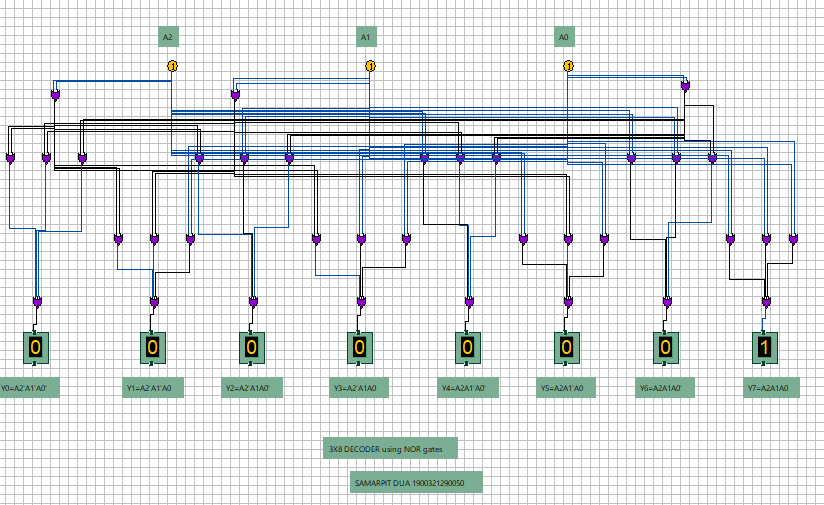
Y7=E A2A1.A0

By ignoring Enable bit we can design 3-8 line Decoder as shown in Diagram below-



**Simulation:- 3-8 Decoders are simulated in simulator, screenshots are given below as- **

****

****