The material covered in this class will be as follows:

⇒Decoders
- 3-to-8 line decoder.
- 2-to-4 line decoder using NAND gates.

⇒De-multiplexers.

⇒Combinational logic implementation using decoders.

⇒Encoders and priority encoders

Decoders

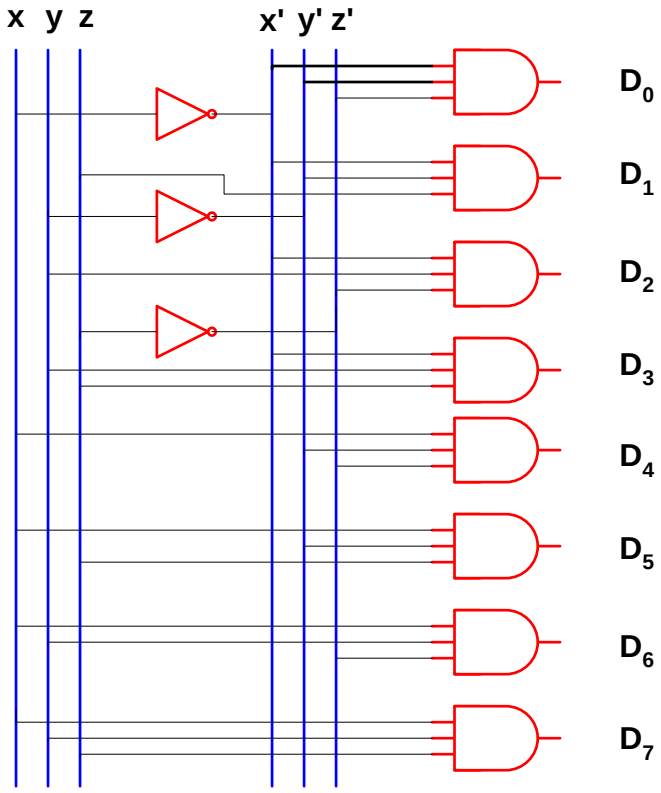
A binary code of n bits is capable of representing up to 2^n distinct elements of coded information. A decoder is a combinational circuit that converts binary information from n input lines to up to 2^n output lines. These decoders are called n-to-m line decoders such that:

$$m \leq 2^n$$

3-to-8 Line Decoder

A 3-to-8 line decoder has three elements and eight outputs. The decoder decodes the input binary code represented by the three bits and generates all eight minterms of the inputs. Only one output is one while the other seven are zeros.

This decoder can be implemented using three inverters and eight AND gates as shown.



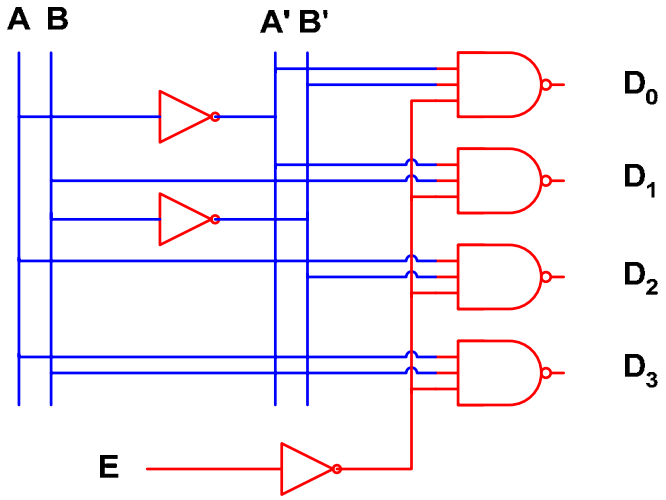
3-to-8 Line Decoder

The following truth table is for the decoder.

Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
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

2-to-4 Line Decoder With Enable Input Using NAND gates

If we use NAND gates to construct the decoder then the outputs are inverted. Decoders are also constructed with one or more enable inputs. An example is shown for the 2-to4 line decoder.



E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Demultiplexers

A decoder with enable input can function as a demultiplexer. A demultiplexer is a combinational circuit that has one input and up to 2^n outputs and it directs the input to an output depending on the values of n selection lines.

A 4 X 16 decoder can be constructed from two 3 X 8 decoders with enable inputs.

Combinational Logic Implementation

Decoders can be used to implement logic functions. Since all the minterms of the function are available at the output then there is no need for simplification. All what is needed is an OR gate for each function to sum the required minterms.

Example:

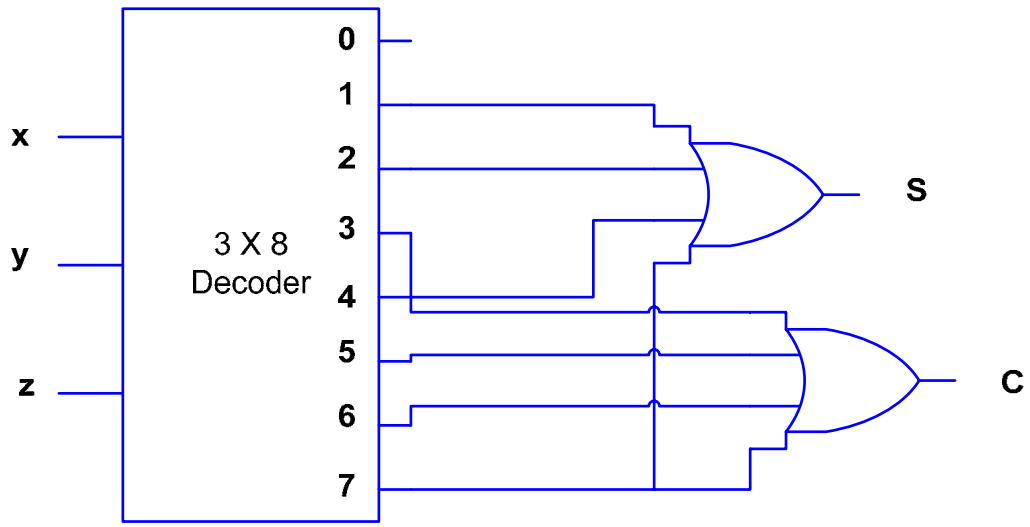
Implement a full adder circuit using an appropriate decoder and OR gates.

The required outputs are:

Sum → $S(x,y,z) = \sum(1,2,4,7)$

Carry → $C(x,y,z) = \sum(3,5,6,7)$

The implementation will be as follows:



Encoders

An encoder is a digital circuit that performs the inverse operation of the decoder. It has 2^n inputs and n outputs that represents the code of the order of the input that is set to one. The truth table of an octal to binary encoder is shown below.

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

The encoder is implements by OR gates. As given in the truth table, the outputs are given by:

$$X = D_4 + D_5 + D_6 + D_7$$

$$Y = D_2 + D_3 + D_6 + D_7$$

$$Z = D_1 + D_3 + D_5 + D_7$$

This encoder has two main drawbacks:

1. When more than one input is 1 at the same time, then the output could indicate a wrong code. E.g. D5 and D6 are one at the same time, then x,y, and z are ones indicating D7 is one.

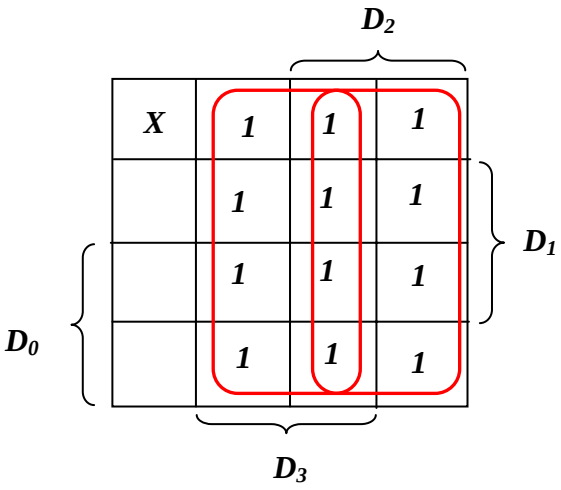
2. If no input is one, which is not valid code, then the outputs are all zeros, which indicates a code for D0.

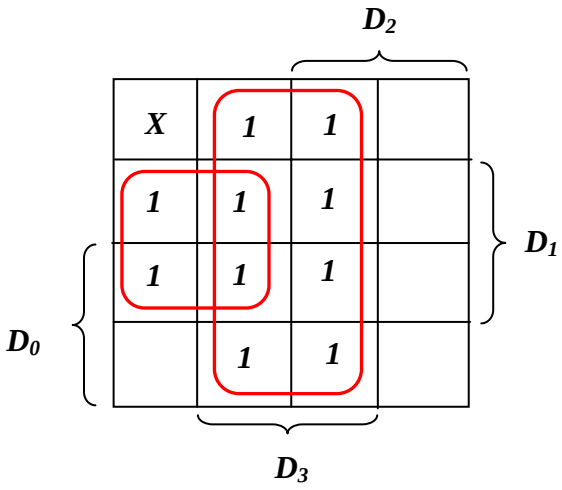
To overcome these problems, we may use a priority encoder.

The truth table of a 4 to 2 priority encoder is given below:

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Using Karnaugh maps to simplify the output functions:





$X = D_2 + D_3$

$Y = D_3 + D_1 D_2'$

$V = D_0 + D_1 + D_2 + D_3$