United International University

Department of CSE

Course Code: CSE 1326

Course Name: Digital Logic Design Laboratory

Experiment no: 04

Experiment Name: Implementing Decoder

Submitted by:

Name: Talha Jubayer

Student ID: 0112410062

Submitted to:

Fahim Hafiz [Lecturer, Dept. of CSE]

Date of Performance: 27-11-2024

Date of Submission: 12-12-2024

Objective:

- Learn about decoder
- Implementing logic function using decoder

Additionally, the report aims to explore how decoders can be utilized to implement various logic functions, demonstrating their effectiveness in simplifying and optimizing combinational logic designs. Through this exploration, the report will provide a comprehensive understanding of how decoders can be employed for efficient logic implementation in digital systems.

Apparatus/Instruments used in the Lab:

● Trainer Board

● Jumper wires

● Power supply

● NOT Gate (74LS04)

● AND Gate (74LS08

● Dual 2-to-4 Decoder (74139)

● Logisim

Theory:

In digital systems, information is represented using binary codes. An n-bit binary code can represent up to 2^n different values. A decoder is a circuit that takes these binary inputs and converts them into a specific number of output lines, depending on how many values need to be represented. If some of the input combinations are not needed or are ignored, the decoder will have fewer than 2^n output lines. These decoders are called n-to-m-line decoders, where m is less than or equal to 2^n. Their job is to generate the possible output combinations based on the input binary values. Decoders are also used in other tasks, like converting BCD (binary-coded decimal) to a seven-segment display.

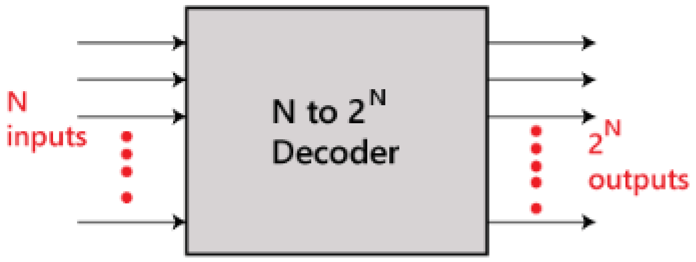


Figure 1 n-to-2^n Line Decoder

The operation of a decoder can be better understood by looking at how its inputs and outputs are related, as shown in Table 1. Notice that the output lines are mutually exclusive, meaning only one output can be "1" at a time. The output that is "1" represents the minterm for the current input.

There are various types of decoders which are as follows:

- **2 to 4 Decoder**
- **3 to 8 Decoder**
- **4 to 16 Decoder**

A. <u>2 to 4 line decoder</u>:

In the 2 to 4 line decoder, there is a total of three inputs, i.e., A_0, and A_1 and E and four outputs, i.e., Y_0, Y_1, Y_2, and Y_3. For each combination of inputs, when the enable 'E' is set to 1, one of these four outputs will be 1. The block diagram and the truth table of the 2 to 4 line decoder are given below.

	E′/G′	A_1	A_0	D_3	D_2	D_1	D_0
m_0	0	0	0	1	0	0	0
m_1	0	0	1	0	1	0	0
m_2	0	1	0	0	0	1	0
m_3	0	1	1	0	0	0	1
	1	X	X	0	0	0	0

Truth Table

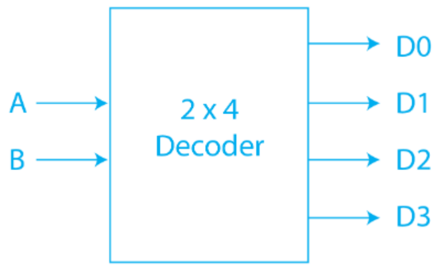


Figure 2 Block Diagram

The logical expression of the term Y0, Y0, Y2, and Y3 is as follows:

D_3=E'.A_1.A_0
D_2=E'.A_1.A_0'

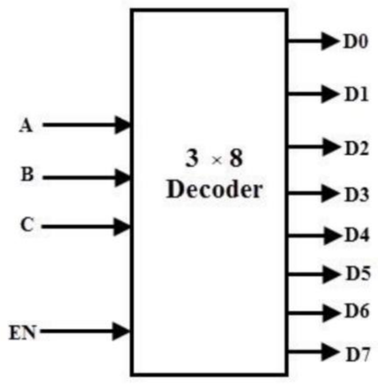
$D_1=E'.A_1'.A_0$
$D_0=E'.A_1'.A_0'$

B. <u>3 to 8 line decoder</u>:

In 3 to 8 line decoder, it includes three inputs and eight outputs. Here the inputs are represented through A, B & C whereas the outputs are represented through D_0, D_1, D_2… D_7.The selection of 8 outputs can be done based on the three inputs. So, the truth table of this 3 line to 8 line decoder is shown below. From the following truth table, we can observe that simply one of 8 outputs from $D_0 – D_7$ can be selected depending on 3 select inputs. This circuit has an enable input E'. Just like 2 to 4 line decoder, when enable E' is set to 0, one of these four outputs will be 1. The block diagram and the truth table of the 3 to 8 line encoder are given below.

Table I Truth table for 3 to 8 Decoder

	E′/G′	**A_2**	**A_1**	**A_0**	**Y_0**	**Y_1**	**Y_2**	**Y_3**	**Y_4**	**Y_5**	**Y_6**	**Y_7**
m_0	0	0	0	0	1	0	0	0	0	0	0	0
m_1	0	0	0	1	0	1	0	0	0	0	0	0
m_2	0	0	1	0	0	0	1	0	0	0	0	0
m_3	0	0	1	1	0	0	0	1	0	0	0	0
m_4	0	1	0	0	0	0	0	0	1	0	0	0
m_5	0	1	0	1	0	0	0	0	0	1	0	0
m_6	0	1	1	0	0	0	0	0	0	0	1	0
m_7	0	1	1	1	0	0	0	0	0	0	0	1
	1	X	X	0	0	0	0	0	0	0	0	0



Block Diagram

From the above truth table of 3 lines to 8 line decoder, the logic expression can be defined as

$D_0 = A'B'C'$

$D_1= A'B'C$

$D_2 = A'BC'$

$D_3 = A'BC$

$D_4 = AB'C'$

D_5= AB’C

D_6 = ABC’

D_7 = ABC

There’s terms called Active low and active high. This used to describe how a

function or pin is activated based on the input signal. We can create two kind of decoder.

1. Active High Decoder:
   A function or pin that operates when the input signal is high. To activate an active high pin, connect it to a high voltage, usually 5V. This will give minterm as output. Means the intended output will be 1.
2. Active Low Decoder:
   A function or pin that operates when the input signal is low. To activate an active low pin, connect it to ground. This will give Maxterm as output. Or in another word output will be 0.

An active high decoder generates all the possible minterms for n input variables (2^n minterms). Since any Boolean function can be written as a sum of minterms, a decoder can create these minterms. Then, we can use an external OR gate to combine them into the desired result. This means that any combinational logic circuit with n inputs and m outputs can be built using an n-to-2^n decoder and m OR gates. In an **active high decoder**, the output line corresponding to the binary input is **high** (logic **1**) when the input matches the output line's address, while all other output lines are **low** (logic **0**). This behavior is typical in most decoders.

Similarly we can create Active low Decoder. Only difference is there will be 0 instead of 1 and we need to use AND Gate instead of OR. Truth table for active low shown in Table-3.

Table II Truth Table for Active Low Decoder

	E′/G′	**A_2**	**A_1**	**A_0**	**Y_0**	**Y_1**	**Y_2**	**Y_3**	**Y_4**	**Y_5**	**Y_6**	**Y_7**
M_0	0	0	0	0	0	1	1	1	1	1	1	1
M_1	0	0	0	1	1	0	1	1	1	1	1	1

M_2	0	0	1	0	1	1	0	1	1	1	1	1
M_3	0	0	1	1	1	1	1	0	1	1	1	1
M_4	0	1	0	0	1	1	1	1	0	1	1	1
M_5	0	1	0	1	1	1	1	1	1	0	1	1
M_6	0	1	1	0	1	1	1	1	1	1	0	1
M_7	0	1	1	1	1	1	1	1	1	1	1	0
	1	X	X	0	0	0	0	0	0	0	0	0

Solution of the mentioned problems:

1. **Pin diagram of 74139**

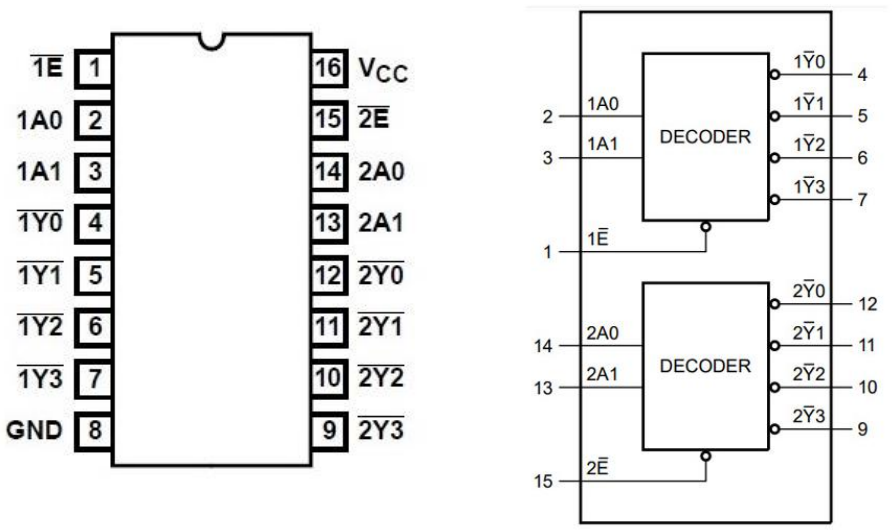


Figure 3 Pin diagram of 74139

2.

b) 3-to-8 line decoder with basic gate

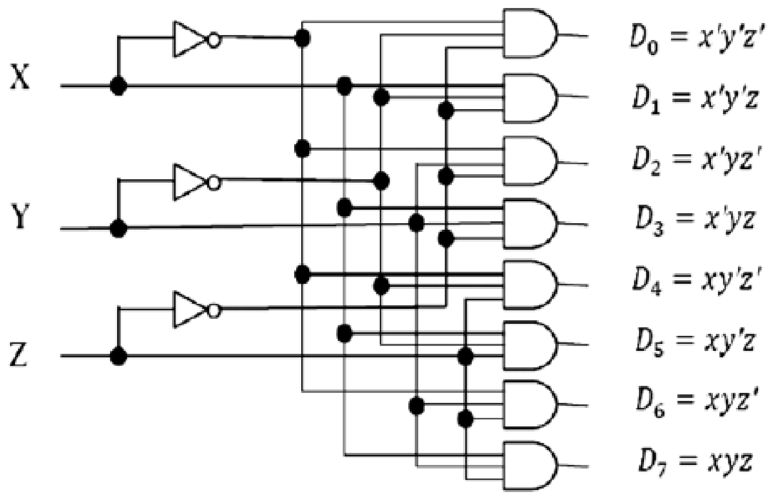


Figure 4 3-to-8 line decoder with basic gate

c) 2-to-4 line decoders with enable

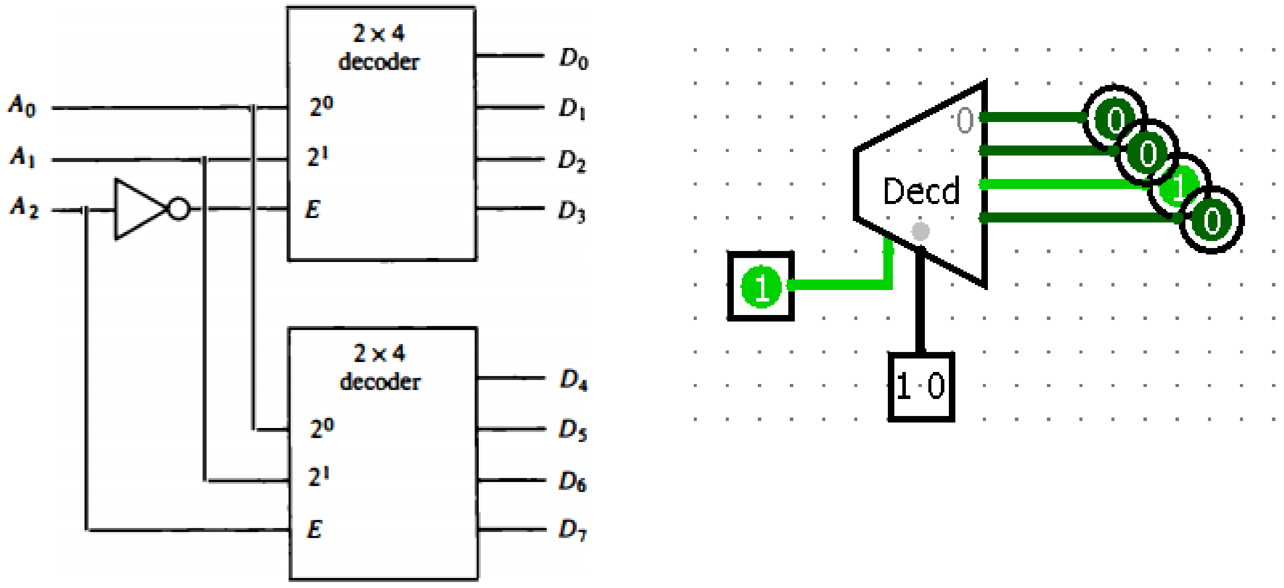


Figure 5 2-to-4 line decoders

3. Logic diagram using 3-to-8 line decoder

F = M1 . M3 . M6 . M7

To implement this in Logisim we need to convert it to SOP form. Which will be-

$F = m_0 + m_2 + m_4 + m_5$

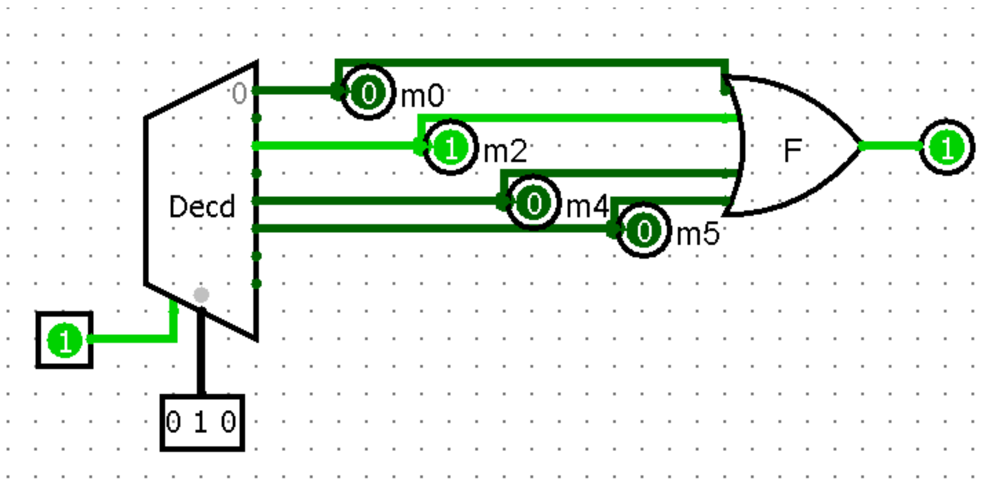
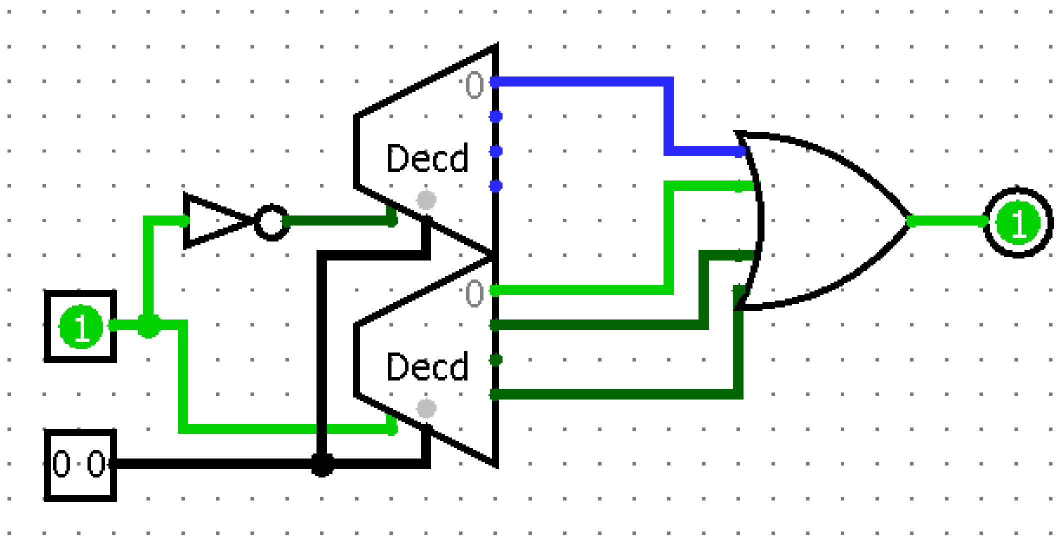


Figure 6 Logic diagram using 3-to-8 line decoder

4.

Answer the question: How do you use the 3-to-8-line decoder (The one you designed with 2-to-4-line decoders) to implement the following function?

F=m0+m2+m4+m5



1. At first, I'll make a 3-to-8-line decoder using two 2-to-4-line decoders.

2. Secondly, I will take three inputs 'A, B & C'. Where 'A' will be acting as the enable key for

both 2-to-4-line decoders. 'B & C' will be the inputs for both decoders.

3. Thirdly, I will pass the enable key ‘A’ to the first decoder by negating it with a NOT gate.

And will pass the enable key ‘A’ to the second decoder normally. Both the decoders will get

the Input ‘B & C’ normally.

4.Then finally, I’ll add the 0, 4, 5 and 7 number output with an OR gate to obtain the function,

F = m0 + m4 + m5 + m7.

Discussion:

Through this experiment, I learned a lot of new things about decoders. I found out how to build larger decoders by combining smaller ones and understood how the enable pin works. I also discovered a new concept called "don't care values." Additionally, I got a basic understanding of the decoder IC and how to create a decoder on a trainer board without using an IC. I now have a clearer idea of how and why decoders are used. Lastly, I learned how to design a logic circuit and write output equations based on a decoder's truth table.