Experiment-7

Title: Implementation of a 3-8 Decoder.

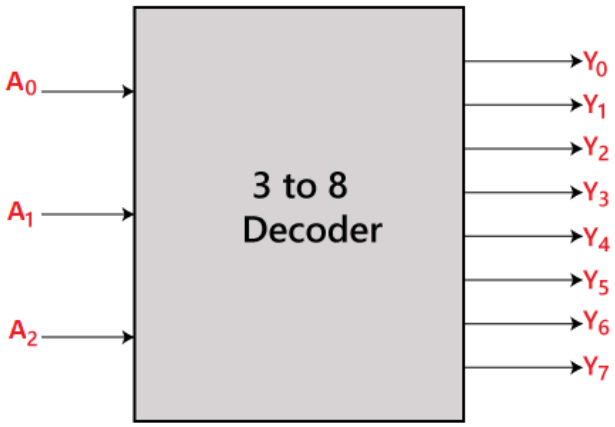
Objective: To implement a 3-8 decoder in behavioural modelling.

Theory:

The combinational circuit that changes the binary information into 2^N output lines is known as **Decoders.** The binary information is passed in the form of N input lines. The output lines define the 2^N-bit code for the binary information. In simple words, the **Decoder** performs the reverse operation of the **Encoder**. At a time, only one input line is activated for simplicity. The produced 2^N-bit output code is equivalent to the binary information.

The 3 to 8 line decoder is also known as **Binary to Octal Decoder**. In a 3 to 8 line decoder, there is a total of eight outputs, i.e., Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 and three outputs, i.e., A_0, A1, and A_2. This circuit has an enable input 'E'. Just like 2 to 4 line decoders, when enable 'E' is set to 1, 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.

Block Diagram:



Truth Table:

Enable	INPUTS			Outputs							
E	A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	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

The logical expression of the term Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 is as follows:

$Y_0=A_0'.A_1'.A_2'$

$Y_1=A_0.A_1'.A_2'$

$Y_2=A_0'.A_1.A_2'$

$Y_3=A_0.A_1.A_2'$

$Y_4=A_0'.A_1'.A_2$

$Y_5=A_0.A_1'.A_2$

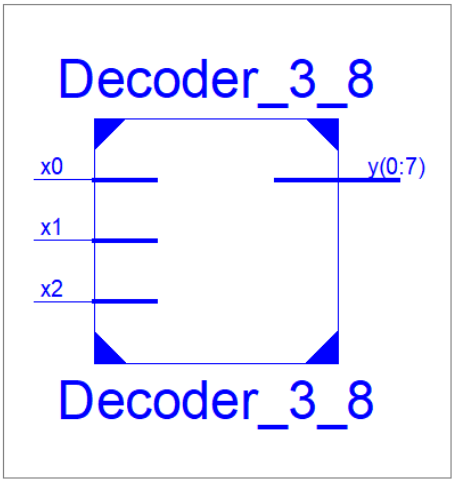
$Y_6=A_0'.A_1.A_2$

$Y_7=A_0.A_1.A_2$

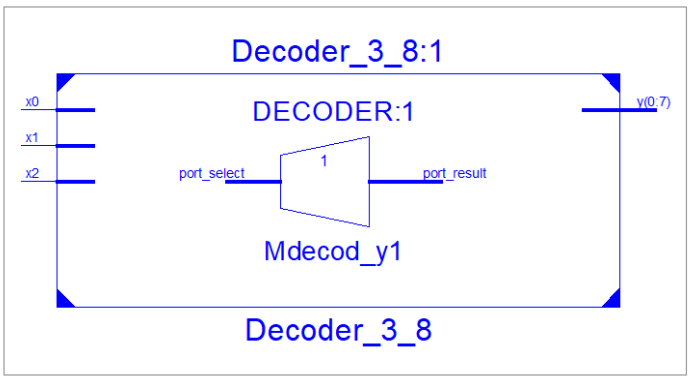
Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Decoder_3_8 is
	Port(x2: in std_logic;
		x1: in std_logic;
		x0: in std_logic;
		y: out std_logic_vector(0 to 7));
end Decoder_3_8;
architecture Decoder of Decoder_3_8 is
begin
	process(x2)
	variable x: std_logic_vector(0 to 2);
		begin
			x:= x2&x1&x0;
			case x is
				when "000" => y <= "00000001";
				when "001" => y <= "00000010";
				when "010" => y <= "00000100";
				when "011" => y <= "00001000";
				when "100" => y <= "00010000";
				when "101" => y <= "00100000";
				when "110" => y <= "01000000";
				when "111" => y <= "10000000";
				when others => y <= "00000000";
			end case;
	end process;
end Decoder;
```

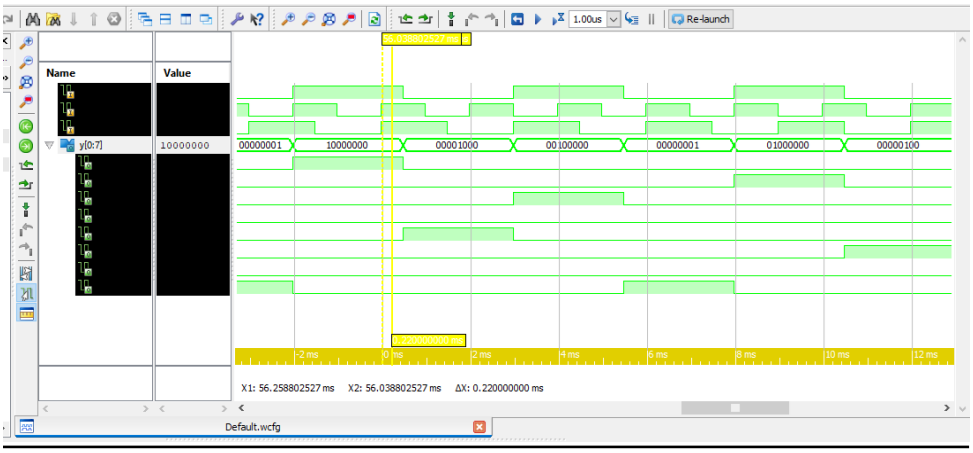
Entity Diagram:



RTL Schematic:



Output:



Result:

We have implemented a 3-8 Decoder in VHDL using Behavioural modelling, and the Entity Diagram, RTL schematic and the output are shown above.

Conclusion:

In this experiment, we learnt about behavioural modelling in VHDL and used it to implement a 3-8 Decoder.