**LAB NOTE**

**Subject: Digital Design Principles**

**Topic: Octal Decoder**

**Student: Minh Quan Tran**

**Jun 6th, 2024**

Table of Contents

[1. Objectives 4](#_Toc168323125)

[2. Theory and Design 6](#_Toc168323126)

[2.1 Theory 6](#_Toc168323127)

[2.2 Requirement 6](#_Toc168323128)

[2.3 Solution 6](#_Toc168323129)

[3. VHDL and Verilog 6](#_Toc168323130)

[3.1 VHDL code 6](#_Toc168323131)

[3.2 Verilog code 7](#_Toc168323132)

[4. Pin Planner 10](#_Toc168323133)

[4.1 Input and Output 10](#_Toc168323134)

[5. Result 12](#_Toc168323135)

**TABLE OF FIGURES**

[Figure 2‑1: Octal Decoder's Truth Table 6](#_Toc168323136)

[Figure 4‑1: SCEMA5F31C6N board 10](#_Toc168323137)

[Figure 4‑2: SW0 and SW1 Pin No 11](#_Toc168323138)

[Figure 4‑3: LEDR0's Pin No 11](#_Toc168323139)

[Figure 4‑4: HEX's Pin No 11](#_Toc168323140)

[Figure 5‑1: When input = ‘000’. 12](#_Toc168323141)

[Figure 5‑2: When input = ‘111’ 12](#_Toc168323142)

[Figure 5‑3: When input ‘101’ 13](#_Toc168323143)

[Figure 5‑4: Input ‘101’ with Complement mode 13](#_Toc168323144)

[Figure 5‑5: Input ‘101’ with Multiline mode 14](#_Toc168323145)

[Figure 5‑6: Result for adding them up 14](#_Toc168323146)

# Objectives

* Design an Octal Decoder using Quartus software.
* Write a VHDL and Verilog code from the design.
* Push the code and run to the SCEMA5F31C6N board.

# Theory and Design

## Theory

Decoder circuit is a combinational logic circuit used to decode binary input. As an example consider the octal decoder depicted in fig.(1). It has three inputs; x2, x1, x0 and eight outputs y0 to y7. . Only one of the output lines is asserted at any time. The valuation of the binary number of the input lines: (x2, x1, x0) determines which output line is activated. Other names of the octal decoder are 3-to-8 decoder and 1-out-of-8 decoder. The truth table in fig. (2) demonstrates the relation between the outputs and the inputs of the decoder.

In any decoder, the index of the asserted output is always equal to the valuation of the binary number of the input lines. Applying 100 at the input lines (x2, x1, x0) will assert output y4 and leave all other outputs are de-asserted.

Number of input lines N of the decoder is calculated based on the number of outputs M:

N= log2 M

Other decoder circuits are the 2-to-4 decoder, decimal decoder, hexadecimal decode.

## Requirement

* Use VHDL and Verilog to implement an Octal Decoder and display to LED [0:7].
* Introduction to the VHDL Case statement.
* **Above and beyond:** Adding 3 more modes: Complement Mode, Multiline Mode, Complement Multiline Mode.

## Solution

To design this system, first need to identify input and output:

* Input will be from SW0 to SW2.
* Output will be the 7 LED.

Then identify what the output will be from the 3 Switches,

A table with numbers and a text

Description automatically generated with medium confidence

Figure 2‑1: Octal Decoder's Truth Table

# VHDL and Verilog

## VHDL code

**library** ieee;

**use** ieee.std\_logic\_1164.**all**;

-- Declaring input and output

**entity** MTran\_Lab5\_VHDL\_OctalDecoder **is**

**port**

(

EN : **in** std\_logic;

SW : **in** std\_logic\_vector (2 **downto** 0);

mode : **in** std\_logic\_vector (1 **downto** 0);

LED : **out** std\_logic\_vector (7 **downto** 0);

HEX : **out** std\_logic\_vector (6 **downto** 0)

);

**end** MTran\_Lab5\_VHDL\_OctalDecoder;

-- Describing the relationship between output and input

**architecture** behavioral **of** MTran\_Lab5\_VHDL\_OctalDecoder **is**

-- Variable Declaring

**signal** LED\_output : std\_logic\_vector (7 **downto** 0) := "00000000";

-- Function's Declaration

-- Function for Octal Decoder

**function** Decoder3to8\_NormalMode(Switch: **in** std\_logic\_vector (2 **downto** 0)) **return** std\_logic\_vector **is**

**begin**

**case** Switch **is**

**when** "000" => **return** "00000001";

**when** "001" => **return** "00000010";

**when** "010" => **return** "00000100";

**when** "011" => **return** "00001000";

**when** "100" => **return** "00010000";

**when** "101" => **return** "00100000";

**when** "110" => **return** "01000000";

**when** "111" => **return** "10000000";

**end** **case**;

**end** Decoder3to8\_NormalMode;

**function** Decoder3to8\_MultilineMode(Switch: **in** std\_logic\_vector (2 **downto** 0)) **return** std\_logic\_vector **is**

**begin**

**case** Switch **is**

**when** "000" => **return** "00000001";

**when** "001" => **return** "00000011";

**when** "010" => **return** "00000111";

**when** "011" => **return** "00001111";

**when** "100" => **return** "00011111";

**when** "101" => **return** "00111111";

**when** "110" => **return** "01111111";

**when** "111" => **return** "11111111";

**end** **case**;

**end** Decoder3to8\_MultilineMode;

-- Seven Segment Display

**function** SevenSegmentDisplay(Switch : std\_logic\_vector (2 **downto** 0)) **return** std\_logic\_vector **is**

**begin**

**case** Switch **is**

**when** "000" => **return** "1000000";

**when** "001" => **return** "1111001";

**when** "010" => **return** "0100100";

**when** "011" => **return** "0110000";

**when** "100" => **return** "0011001";

**when** "101" => **return** "0010010";

**when** "110" => **return** "0000010";

**when** "111" => **return** "1111000";

**end** **case**;

**end** SevenSegmentDisplay;

**begin**

**process**(SW, mode)

**begin**

**if** EN = '1' **then**

**case** mode **is**

-- Normal Mode

**when** "00" =>

LED <= Decoder3to8\_NormalMode(SW);

HEX <= SevenSegmentDisplay(SW);

-- Complement Mode

**when** "01" =>

LED\_output <= Decoder3to8\_NormalMode(SW);

HEX <= SevenSegmentDisplay(SW);

LED <= **not** LED\_output;

-- Multiline Mode

**when** "10" =>

LED <= Decoder3to8\_MultilineMode(SW);

HEX <= SevenSegmentDisplay(SW);

-- Complement Multiline Mode

**when** "11" =>

LED\_output <= Decoder3to8\_MultilineMode(SW);

HEX <= SevenSegmentDisplay(SW);

LED <= **not** LED\_output;

**end** **case**;

**else**

LED <= "00000000";

HEX <= "1111111";

**end** **if**;

**end** **process**;

**end** behavioral;

## Verilog code

**module MTran\_Lab5\_Verilog\_OctalDecoder** (

**input** EN,

input [2:0] SW,

input [1:0] mode,

output reg [7:0] LED,

output reg [6:0] HEX

)**;**

// Variable Declaring

**reg** [7:0] LED\_output**;**

// Function's Declaration

// Function for Octal Decoder

function [7:0] Decoder3to8\_NormalMode;

**input** [2:0] Switch**;**

**begin**

**case** **(**Switch**)**

3'd0**:** Decoder3to8\_NormalMode = 8'b00000001**;**

3'd1**:** Decoder3to8\_NormalMode = 8'b00000010**;**

3'd2**:** Decoder3to8\_NormalMode = 8'b00000100**;**

3'd3**:** Decoder3to8\_NormalMode = 8'b00001000**;**

3'd4**:** Decoder3to8\_NormalMode = 8'b00010000**;**

3'd5**:** Decoder3to8\_NormalMode = 8'b00100000**;**

3'd6**:** Decoder3to8\_NormalMode = 8'b01000000**;**

3'd7**:** Decoder3to8\_NormalMode = 8'b10000000**;**

**default:** Decoder3to8\_NormalMode = 8'b00000000**;**

**endcase**

**end**

endfunction

function [7:0] Decoder3to8\_MultilineMode;

**input** [2:0] Switch**;**

**begin**

**case** **(**Switch**)**

3'd0**:** Decoder3to8\_MultilineMode = 8'b00000001**;**

3'd1**:** Decoder3to8\_MultilineMode = 8'b00000011**;**

3'd2**:** Decoder3to8\_MultilineMode = 8'b00000111**;**

3'd3**:** Decoder3to8\_MultilineMode = 8'b00001111**;**

3'd4**:** Decoder3to8\_MultilineMode = 8'b00011111**;**

3'd5**:** Decoder3to8\_MultilineMode = 8'b00111111**;**

3'd6**:** Decoder3to8\_MultilineMode = 8'b01111111**;**

3'd7**:** Decoder3to8\_MultilineMode = 8'b11111111**;**

**default:** Decoder3to8\_MultilineMode = 8'b00000000**;**

**endcase**

**end**

endfunction

// Seven Segment Display

function [6:0] SevenSegmentDisplay;

**input** [2:0] Switch**;**

**begin**

**case** **(**Switch**)**

3'd0**:** SevenSegmentDisplay = 7'b1000000**;**

3'd1**:** SevenSegmentDisplay = 7'b1111001**;**

3'd2**:** SevenSegmentDisplay = 7'b0100100**;**

3'd3**:** SevenSegmentDisplay = 7'b0110000**;**

3'd4**:** SevenSegmentDisplay = 7'b0011001**;**

3'd5**:** SevenSegmentDisplay = 7'b0010010**;**

3'd6**:** SevenSegmentDisplay = 7'b0000010**;**

3'd7**:** SevenSegmentDisplay = 7'b1111000**;**

**default:** SevenSegmentDisplay = 7'b1111111**;**

**endcase**

**end**

endfunction

// Main code

**always** @(SW,EN,mode)

**begin**

**if** (EN)

**begin**

**case** **(**mode**)**

// Normal Mode

2'b00**:**

begin

LED = Decoder3to8\_NormalMode**(**SW**);**

HEX = SevenSegmentDisplay**(**SW**);**

end

// Complement Mode

2'b01**:**

begin

LED\_output = Decoder3to8\_NormalMode**(**SW**);**

HEX = SevenSegmentDisplay**(**SW**);**

LED = ~LED\_output**;**

end

// Multiline Mode

2'b10**:**

begin

LED = Decoder3to8\_MultilineMode**(**SW**);**

HEX = SevenSegmentDisplay**(**SW**);**

end

// Complement Multiline Mode

2'b11**:**

begin

LED\_output = Decoder3to8\_MultilineMode**(**SW**);**

HEX = SevenSegmentDisplay**(**SW**);**

LED = ~LED\_output**;**

end

**endcase**

**end**

**else**

**begin**

LED = 8'b00000000;

HEX = 7'b1111111;

**end**

**end**

endmodule

# Pin Planner

## Input and Output

A close-up of a circuit board

Description automatically generated

Figure 4‑1: SCEMA5F31C6N board

Assigning:

SW[0]: SW0

SW[1]: SW1

SW[2]: SW2

Mode[0]: SW7

Mode[1]: SW8

EN: SW9

LED[0]: LEDR0

LED[1]: LEDR1

LED[2]: LEDR2

LED[3]: LEDR3

LED[4]: LEDR4

LED[5]: LEDR5

LED[6]: LEDR6

LED[7]: LEDR7

HEX0[0]: HEX0[0]

HEX0[1]: HEX0[1]

HEX0[2]: HEX0[2]

HEX0[3]: HEX0[3]

HEX0[4]: HEX0[4]

HEX0[5]: HEX0[5]

HEX0[6]: HEX0[6]

From the DE1\_SoC\_User\_Manual,

A screenshot of a slide switch

Description automatically generated

Figure 4‑2: SW0 and SW1 Pin No

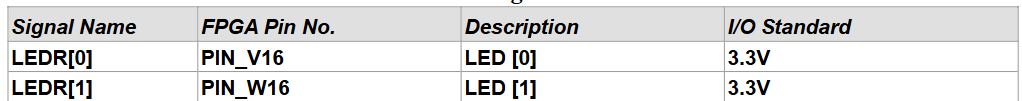


Figure 4‑3: LEDR0's Pin No

 A group of black text

Description automatically generated

Figure 4‑4: HEX's Pin No

# Result

A computer chip on a desk

Description automatically generated

Figure 5‑1: When input = ‘000’.

SW2 = 0, SW1 = 0, SW0 = 0 which should be 0, **LED0** asserted and the 7 segment LED displaying **0**.

A close up of a computer

Description automatically generated

Figure 5‑2: When input = ‘111’

SW2 = 1, SW1 = 1, SW0 = 1 which should be 7, **LED7** asserted and the 7 segment LED displaying **7**.

A close up of a computer chip

Description automatically generated

Figure 5‑3: When input ‘101’

SW2 = 1, SW1 = 0, SW0 = 1 which should be 5, **LED5** asserted and the 7 segment LED displaying **5**.

**Different mode:**

* **Complement mode:**

A close up of a computer

Description automatically generated

Figure 5‑4: Input ‘101’ with Complement mode

In this mode, rather than LED5 assert, all other LED except LED5 asserted. To activate this mode SW [8:7] = ‘01’.

* **Multiline Mode:**

A close up of a circuit board

Description automatically generated

Figure 5‑5: Input ‘101’ with Multiline mode

In this mode, when input is ‘101’ which is **5**, LED0 to LED5 lit up. To activate this mode SW [8:7] = ‘10’.

* **Complement Multiline Mode:**

A close up of a circuit board

Description automatically generated

Figure 5‑6: Result for adding them up

In this mode, it take the output of Multiline Mode and complemented it. To activate this mode SW [8:7] = ‘11’

**REFERENCES**