Lab 3. Quartus Tutorial and Seven Seg Display – DE2 boards

a. Install the Quartus 20.1 software (see my word document Getting Quartus Installed and Setup 2025). It also contains useful information about Quartus.

b. Work through the Altera Quartus 20_1 Standard Introduction pdf for the light.vhd design. **Use one of our Altera** Terasic DE2-115 (Cyclone IV), the DE10-Standard (Cyclone V) or the DE1- SOC (Cyclone V) FPGA development and education boards **instead of the DE-5 board in the tutorial.** Get the chip number on your board's FPGA chip, e.g., **t**he DE2-115 has a Cyclone IV FPGA EP4CE115F29C7 FPGA chip. Skip the section on using the Assignment Editor to assign the physical pin locations to match those on the your board. Instead use the Assignment/Pin Planner (for help, see my power point MCI Quartus and De2-115 tips 2025).

c. Install the USB Blaster-II driver (see the Altera GettingStartedWithDEBoards ver 20_1.pdf). If your driver will not install on Windows 10, see the fixes in the Getting StartedWithDEBoards Ver20_1 (I had to do the Disable Driver Signature Enforcement on my Dell Laptop.) For the Windows 11 fix, Turn off memory integrity by: Select Start , enter 'Core Isolation' in the taskbar, and select Core Isolation from the list of results to open the Windows security app. On the Core isolation page, turn off the toggle for Memory integrity. You might need to restart your device (see my powerpoint "MCI Quartus and board tips"). Program your board using one of the two methods at the end of the tutorial (depending on which board you have). Test your design using switch 0 and 1. Click the TOOLS menu, then NetList Viewers, then RTL Viewer to see the circuit. If the schematic has blocks, double click them to open them and see the gates inside. **Select File/Export to save the RTL circuit as a pdf file.**

For credit for this assignment, submit:

1) The vhd file of the light circuit.
2) The pdf image of the AND -OR circuit exported from the RTL viewer.

d. Use Quartus (NOT ModelSim)

Use your Altera DE2 Development Board.

To prevent damage to the board, be sure to set unused pins as "input tri-stated". You do this by Assignments > Device > Device and Pin Options > Unused Pins > As input tri-stated.

Seven Segment display (decoder) controlled by 4 switches i_SW(3) to iSW(0). Should output 0123456789AbCdEF on seven segment display HEX0. Refer to your board's user manual, e.g., for the DE2-115, see pages 36 and 37 for HEX0 segments 0 through 6 for the wiring of the 7-seg segment Digit 0, and create a truth

table for the seven seg display wiring on paper. The board's seven-segment displays are active low, so make the truth table active low (use a 0 to turn on a segment). For example, the binary value to display the number 3 is: 0110000. Don't deal with the decimal point because it is not connected on the board. Use STD_LOGIC_VECTOR on both inputs and outputs. Use the **With-Select** method (it is good for truth tables). Try all 16 binary combinations of the 4 switches to check each display pattern for all 16 hex digits.

- For credit for this assignment, submit:
  - Demo all 16 hex values displayed on the seven segment display.
  - Your seven seg display vhd file.
  - The pdf image of the circuit exported with the RTL viewer.

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY light IS
PORT(x1, x2 : IN STD_LOGIC ;
f : OUT STD_LOGIC);
END light ;
ARCHITECTURE LogicFunction OF light IS
BEGIN
f <= (x1 AND NOT x2) OR (NOT x1 AND x2) ;
END LogicFunction ;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sevenSegments is
    Port (
        i_SW  : in  STD_LOGIC_VECTOR(3 downto 0); -- switches i_SW(3 downto 0)
        o_HEX0 : out STD_LOGIC_VECTOR(6 downto 0) -- segments a to g (active low)
    );
end sevenSegments;

architecture Behavioural of sevenSegments is
begin
    -- Active low segments: 0 = on, 1 = off
    with i_SW select
        o_HEX0 <=
        "1000000" when "0000", -- 0
        "1111001" when "0001", -- 1
        "0100100" when "0010", -- 2
        "0110000" when "0011", -- 3
        "0011001" when "0100", -- 4
        "0010010" when "0101", -- 5
        "0000010" when "0110", -- 6
        "1111000" when "0111", -- 7
        "0000000" when "1000", -- 8
        "0010000" when "1001", -- 9
        "0001000" when "1010", -- A
        "0000011" when "1011", -- b
        "1000110" when "1100", -- C
        "0100001" when "1101", -- d
        "0000110" when "1110", -- E
        "0001110" when "1111", -- F
        "1111111" when others; -- default (all segments off)
end Behavioural;
```