New York City College of Technology
Computer Engineering Technology

Designing Decoders & Programming with The Altera DE-10 Lite

Lab 004

CET 4805 Component and Subsystem Design II
Section OL40
Professor Hakan Pekcan
Spring 2021
Report Written by Maria Pedroza & Galib F. Rahman

Table of Contents

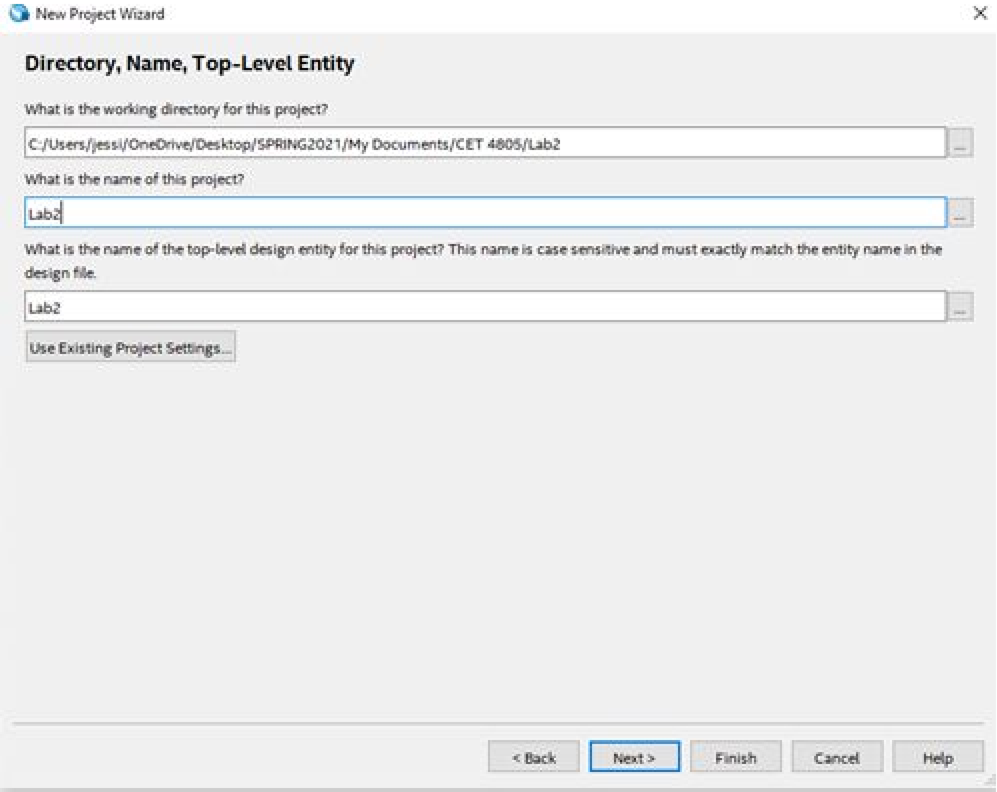
Objective

The objective of this laboratory exercise is to use the Quartus® Prime software to virtually create a 7-segment decoder module that inputs 4 bits [D3, D2, D1, D0] and simulate said module. We will VHDL to express the design concept of a BCD to 7 Segment Decoder and create a block diagram file and waveform vector file to simulate the behavior of the module. Upon observation of the simulated results we will verify the validity of the entity's operation using the table (shown in Part I).

Procedure & Results

Creating A Project

1. To begin, we will open the Quartus Software. Select **File-New Project Wizard**. Enter the appropriate drive letter for the designated storage area on the computer you are using followed by the working directory. Don't forget to create the folder **Lab2** under the subfolder of your path. Assign the project name **Lab2**.



New Project Wizard

Directory, Name, Top-Level Entity

What is the working directory for this project?

C:/Users/jessi/OneDrive/Desktop/SPRING2021/My Documents/CET 4805/Lab2

What is the name of this project?

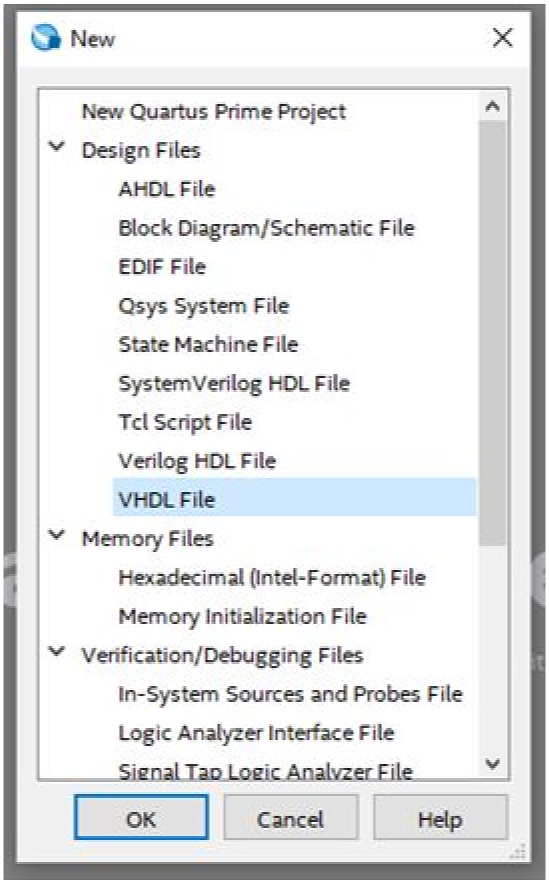
Lab2

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

Lab2

Creating A VHDL File and Symbol for the Code(bdf)

2. Open a new VDHL Device Design file (File>New) by highlighting VHDL File. Type the VHDL codes shown in Text Box.

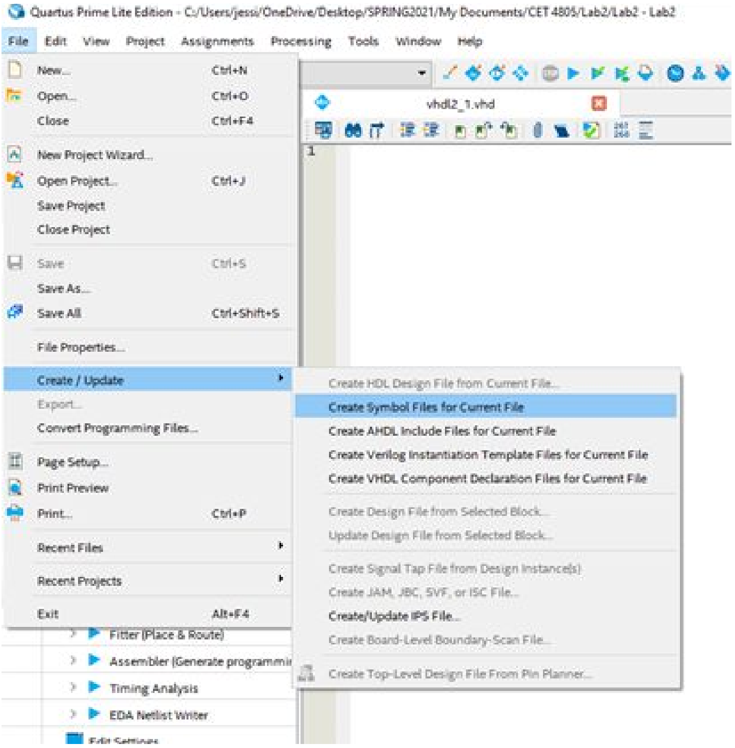

New
New Quartus Prime Project
Design Files
AHDL File
Block Diagram/Schematic File
EDIF File
Qsys System File
State Machine File
SystemVerilog HDL File
Tcl Script File
Verilog HDL File
VHDL File
Memory Files
Hexadecimal (Intel-Format) File
Memory Initialization File
Verification/Debugging Files
In-System Sources and Probes File
Logic Analyzer Interface File
Signal Tap Logic Analyzer File
OK
Cancel
Help

3. File name vhdl2_1.vhd must be chosen according to which approach is being used. Save the VHDL file as vhdl2_1.vhd as part of our project under your subfolder. Place a check mark in the space labeled. Add file to current project and press Save.

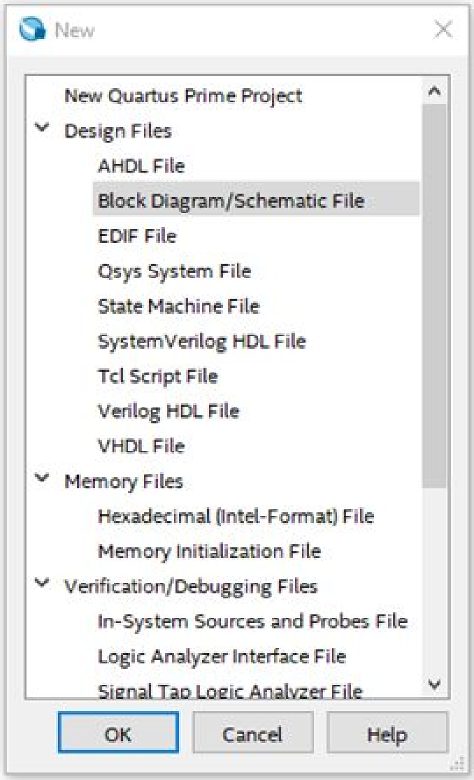
DO NOT COMPILE THE FILE AT THIS POINT

4. Select **File>Create/Update>Create Symbol Files for Current File** to create a symbol file for the VHDL code entered. A display window should soon appear stating that the **Create**

Symbol File was (or not) successful. Click **OK** and close the Compilation Report window.

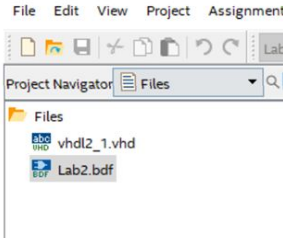

Quartus Prime Lite Edition - C:/Users/jessi/OneDrive/Desktop/SPRING2021/My Documents/CET 4805/Lab2/Lab2 - Lab2
File
Edit
View
Project
Assignments
Processing
Tools
Window
Help
New... Ctrl+N
Open... Ctrl+O
Close Ctrl+F4
New Project Wizard...
Open Project... Ctrl+J
Save Project
Close Project
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
File Properties...
Create / Update
Export...
Convert Programming Files...
Page Setup...
Print Preview
Print... Ctrl+P
Recent Files
Recent Projects
Exit Alt+F4
vhdl2_1.vhd
1
Create HDL Design File from Current File...
Create Symbol Files for Current File
Create AHDL Include Files for Current File
Create Verilog Instantiation Template Files for Current File
Create VHDL Component Declaration Files for Current File
Create Design File from Selected Block...
Update Design File from Selected Block...
Create Signal Tap File from Design Instance(s)
Create JAM, JBC, SVF, or ISC File...
Create/Update IPS File...
Create Board-Level Boundary-Scan File...
Create Top-Level Design File From Pin Planner...
Fitter (Place & Route)
Assembler (Generate programmi
Timing Analysis
EDA Netlist Writer
Edit Settings

5. Open a new Schematic file (**File>New**) by highlighting **Block Diagram/Schematic File**. And click **OK**. And construct the circuit using the symbols you just create. Each symbol should be available in the Project Library in the Symbol diagonal box.


New
New Quartus Prime Project
Design Files
AHDL File
Block Diagram/Schematic File
EDIF File
Qsys System File
State Machine File
SystemVerilog HDL File
Tcl Script File
Verilog HDL File
VHDL File
Memory Files
Hexadecimal (Intel-Format) File
Memory Initialization File
Verification/Debugging Files
In-System Sources and Probes File
Logic Analyzer Interface File
Signal Tap Logic Analyzer File
OK
Cancel
Help

(Right click on the bdf window and then >Insert >Symbol > Project >chd2_1)
To insert pins, right click and then >Insert >Symbol >c:/intelfpga…? >primitives > pin > input (or input)

6. Before compiling this bdf file, we need to name this bdf file and save it as a part of our project under your subfolder. Choose File > Save As and enter File name as lab2.bdf. Please a check mark in the space labeled. Add file to current project and press Save.



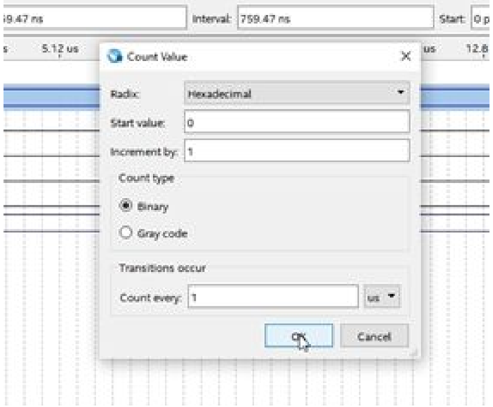
7. Compile the project by selecting Processing > Start Compilation, or press Ctrl-L, or use the Compilation button in the toolbar. The compilation takes several seconds. When it is complete it should give a message that indicates, “Full compilation was successful”. Press OK. If unsuccessful, correct all errors and try to re-compile.

Simulating a Vector Waveform File (University Program vwf)

8. As you have done step in the Part 1, you need to create a Vector Waveform File (vwf) to simulate a design(bdf) file. Add all inputs and output, specify an end time of 20 μs for our waveform display (Edit -> Set End Time) , and then save it as lab2.vwf.
9. When creating the D[3..0] bus, enter D for the bus name, select Hexadecimal for the Radix, and enter 4 for the Bus Width in the Node Properties window. When created, the D waveform will appear with a plus sign implying that it can be ungrouped to show the individual bits, D[3], D[2], D[1], and D[0].

After adding the input vector D and the output vector S, highlight D and go to count Value in the simulation waveform editor toolbar as shown below.

In the Count Value window set the Radix to Hexadecimal. The start value must be 0 and increment the values by 1. This will simulate the switch inputs starting at 0000 and increment D by 1 for each clock count giving 0001 the for the next input. For count enter 1 and select us.


Count Value

Radix: Hexadecimal

Start value: 0

Increment by: 1

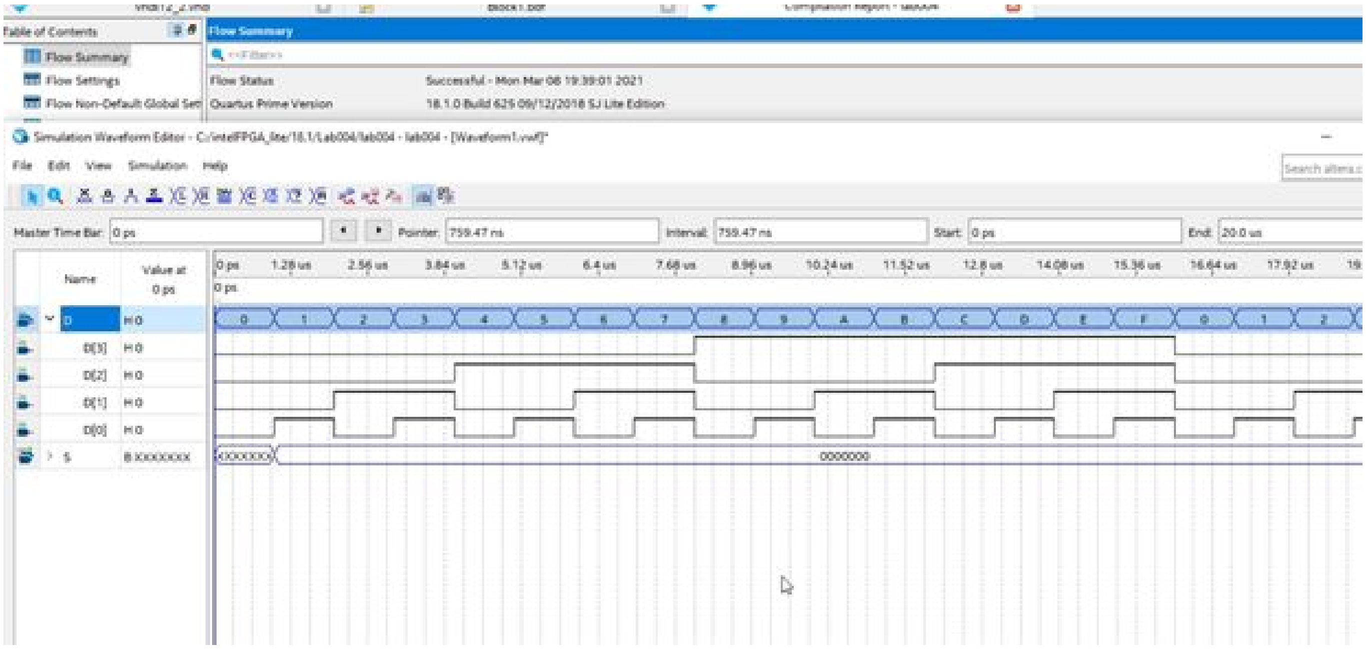
Count type

● Binary

○ Gray code

Transitions occur

Count every: 1 us


Flow Summary

Flow Status: Successful - Mon Mar 08 19:39:01 2021

Quartus Prime Version: 18.1.0 Build 625 09/12/2018 SJ Lite Edition

Simulation Waveform Editor - C:/intelFPGA_lite/18.1/Lab004/lab004 - lab004 - [Waveform1.vwf]*

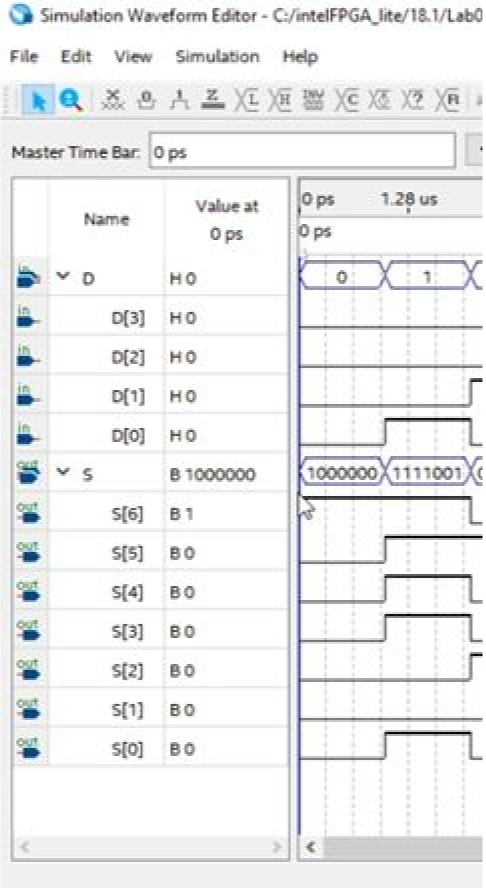
Master Time Bar: 0 ps   Pointer: 759.47 ns   Interval: 759.47 ns   Start: 0 ps   End: 20.0 us

Name	Value at 0 ps
D	H 0
D[3]	H 0
D[2]	H 0
D[1]	H 0
D[0]	H 0
S	B XXXXXXX

Each wave represents your input for your design, when your waveform is low, this will represent a logical 0 in your program. When the waveform is high this represents a logic 1 in your program. Together D[3] - D[0] will represent a number each count as shown below.

10. Run the simulation for your program by clicking on Run Functional Simulation in the toolbar or by going to Simulation > Run Functional simulation. The Simulation Waveform editor will give you the output for S when D = any given input. This is based on what you set it equal to in the code. In our example vhdl2_1:

    D = ‘0000’ then S = ‘1000000’
    D = ‘0001’ then S = ‘1111001’



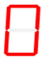
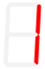
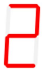
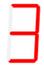
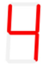
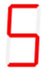
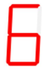
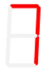
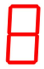
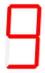
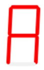
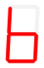
Master Time Bar: 0 ps

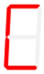
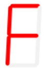
Name	Value at 0 ps
D	H 0
D[3]	H 0
D[2]	H 0
D[1]	H 0
D[0]	H 0
S	B 1000000
S[6]	B 1
S[5]	B 0
S[4]	B 0
S[3]	B 0
S[2]	B 0
S[1]	B 0
S[0]	B 0

11. The Simulation Waveforms appear in the Simulation Report. You may have to expand the size of the Simulation Waveforms to suit your need and choose View > Fit in Window to see the entire waveform.

Part I: Design BCD to Seven Segment Decoder

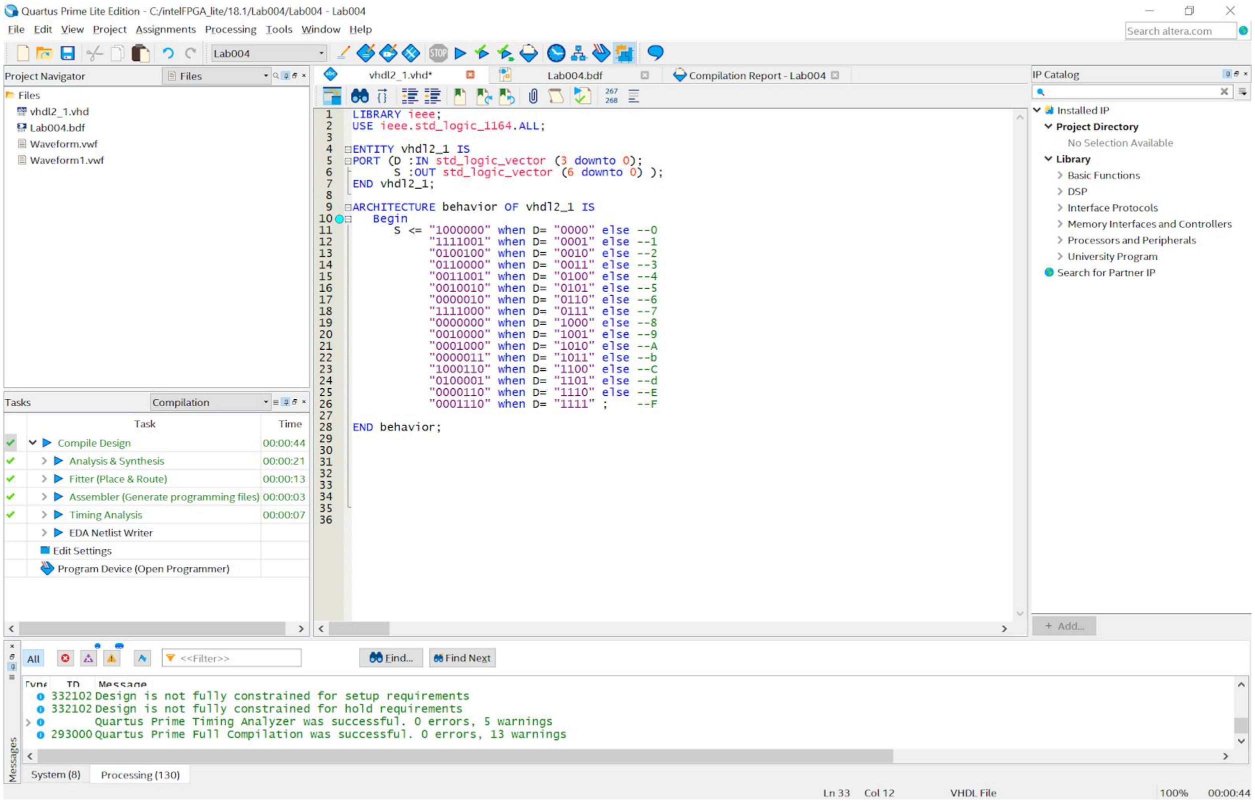
The 7 LEDs within the segment display are configured such that the anodes are tied together & connected to V_{CC}. Thus, the individual segments are activated once a logic value of 0 is applied. The table below demonstrates the 4-bit inputs (D[3..0])and corresponding outputs of the module (S[6..0]).

Symbol	D3	D2	D1	D0	Shape	S6 (g)	S5 (f)	S4 (e)	S3 (d)	S2 (c)	S1 (b)	S0 (a)
0	0	0	0	0		1	0	0	0	0	0	0
1	0	0	0	1		1	1	1	1	0	0	1
2	0	0	1	0		0	1	0	0	1	0	0
3	0	0	1	1		0	1	1	0	0	0	0
4	0	1	0	0		0	0	1	1	0	0	1
5	0	1	0	1		0	0	1	0	0	1	0
6	0	1	1	0		0	0	0	0	0	1	0
7	0	1	1	1		1	1	1	1	0	0	0
8	1	0	0	0		0	0	0	0	0	0	0
9	1	0	0	1		0	0	1	0	0	0	0
A	1	0	1	0		0	0	0	**1**	0	0	0
B	1	0	1	1		0	0	0	0	0	**1**	**1**

C	1	1	0	0		**1**	0	0	0	**1**	**1**	0
D	1	1	0	1		0	**1**	0	0	0	0	**1**
E	1	1	1	0		0	0	0	0	**1**	**1**	0
F	1	1	1	1		0	0	0	**1**	**1**	**1**	0

VHDL Code

For this portion of the exercise we utilized when/else with the conditional concurrent signal assignment statement to describe the behavior of the designed entity *vhdl12_1*. When one or more of the signals change value (on the right hand side), the statement is executed and evaluates the conditions from top to bottom, if true the corresponding expression is executed and the value is assigned to the given signal.

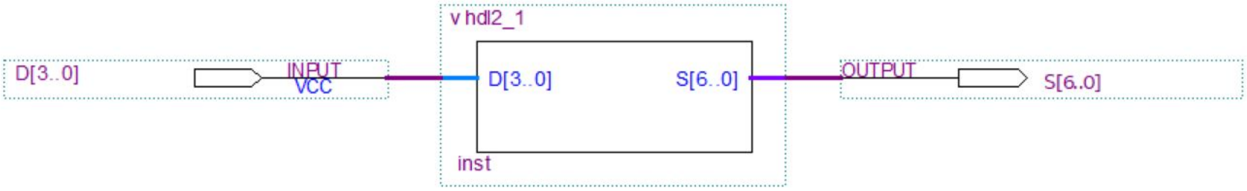


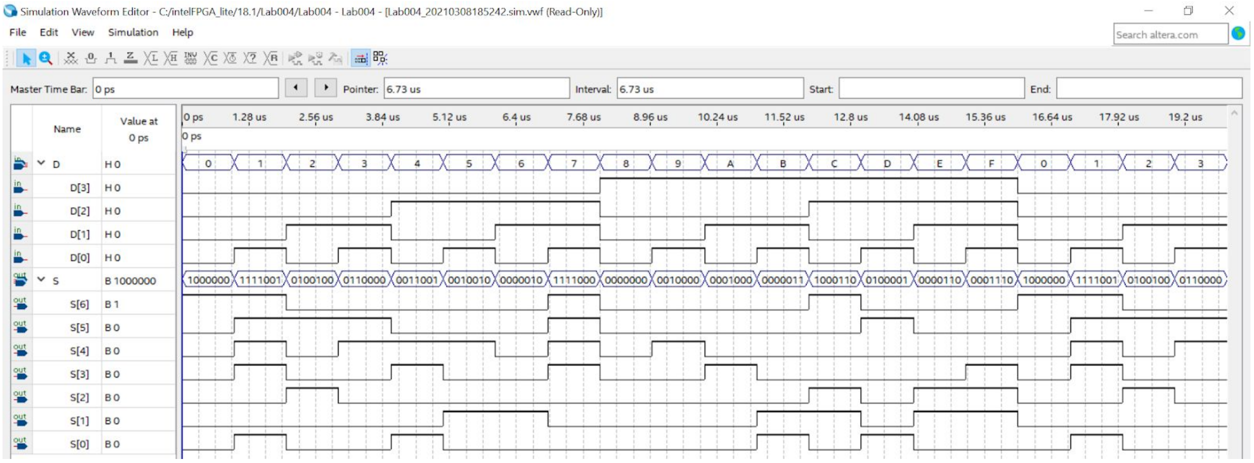
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY vhdl2_1 IS
PORT (D :IN std_logic_vector (3 downto 0);
        S :OUT std_logic_vector (6 downto 0) );
END vhdl2_1;

ARCHITECTURE behavior OF vhdl2_1 IS
   Begin
        S <= "1000000" when D= "0000" else --0
             "1111001" when D= "0001" else --1
             "0100100" when D= "0010" else --2
             "0110000" when D= "0011" else --3
             "0011001" when D= "0100" else --4
             "0010010" when D= "0101" else --5
             "0000010" when D= "0110" else --6
             "1111000" when D= "0111" else --7
             "0000000" when D= "1000" else --8
             "0010000" when D= "1001" else --9
             "0001000" when D= "1010" else --A
             "0000011" when D= "1011" else --b
             "1000110" when D= "1100" else --C
             "0100001" when D= "1101" else --d
             "0000110" when D= "1110" else --E
             "0001110" when D= "1111" ;    --F

END behavior;
```

Task	Time
✓ Compile Design	00:00:44
✓ Analysis & Synthesis	00:00:21
✓ Fitter (Place & Route)	00:00:13
✓ Assembler (Generate programming files)	00:00:03
✓ Timing Analysis	00:00:07
EDA Netlist Writer	
Edit Settings	
Program Device (Open Programmer)	

```
Type  ID     Message
      332102 Design is not fully constrained for setup requirements
      332102 Design is not fully constrained for hold requirements
             Quartus Prime Timing Analyzer was successful. 0 errors, 5 warnings
      293000 Quartus Prime Full Compilation was successful. 0 errors, 13 warnings
```

VHDL Code




Simulation Waveform Editor - C:/intelFPGA_lite/18.1/Lab004/Lab004 - Lab004 - [Lab004_20210308185242.sim.vwf (Read-Only)]

Master Time Bar: 0 ps

Pointer: 6.73 us

Interval: 6.73 us

Start:

End:

Name	Value at 0 ps
D	H 0
D[3]	H 0
D[2]	H 0
D[1]	H 0
D[0]	H 0
S	B 1000000
S[6]	B 1
S[5]	B 0
S[4]	B 0
S[3]	B 0
S[2]	B 0
S[1]	B 0
S[0]	B 0

Conclusion

Upon completion of this laboratory exercise, we have successfully utilized the Quartus® Prime software to create & simulate a 4-bit input 7 segment (common anode) decoder module. We have used VHDL to define the behavior of our circuit and designed a block diagram to represent the decoder as an entity by assigning the appropriate bus values for both input and output. Upon successful compilation of both the VHDL and block diagram we created a Waveform Vector File to observe the results.

Intel® Quartus® Prime Program Files