CRN: 22840

CET3136 – Logic Devices Programming

Spring 2025

Experiment 1

Introduction to VHDL Design and Programming

Performed By:

Anthony Paul Sevarino

Submitted to:

Prof. Ashley Evans

Department of Electrical & Computer Engineering Technology (ECET)
School of Engineering, Technology, and Advanced Manufacturing (ETAM)
Valencia College

Date Submitted
01/15/2025

Introduction

The intent of this lab is to install the Quartus software environment, and become familiar with fundamental concepts within it, ranging from project set-up, to software to board interfacing. For this particular lab, a simple logic operation is implemented onto the MAX 10 development board. The first part of the lab is dedicated to the installation of the Quartus software, and establishing the proper driver installations. The second part of the lab focuses on building a project, and viewing a waveform of the subsequent logic. Finally, the third part of the lab will cover programming the project to the development board, and testing it against the theoretical logic behavior.

LIST OF EQUIPMENT/PARTS/COMPONENTS/SOFTWARE

- DE10-Lite MAX-10 Dev Board
- USB 2.0 Type B Cable
- Windows Desktop
- Quartus FPGA Design Software
- Modelsim HDL Simulator

PROCEDURE

Part 1 – Installing Quartus

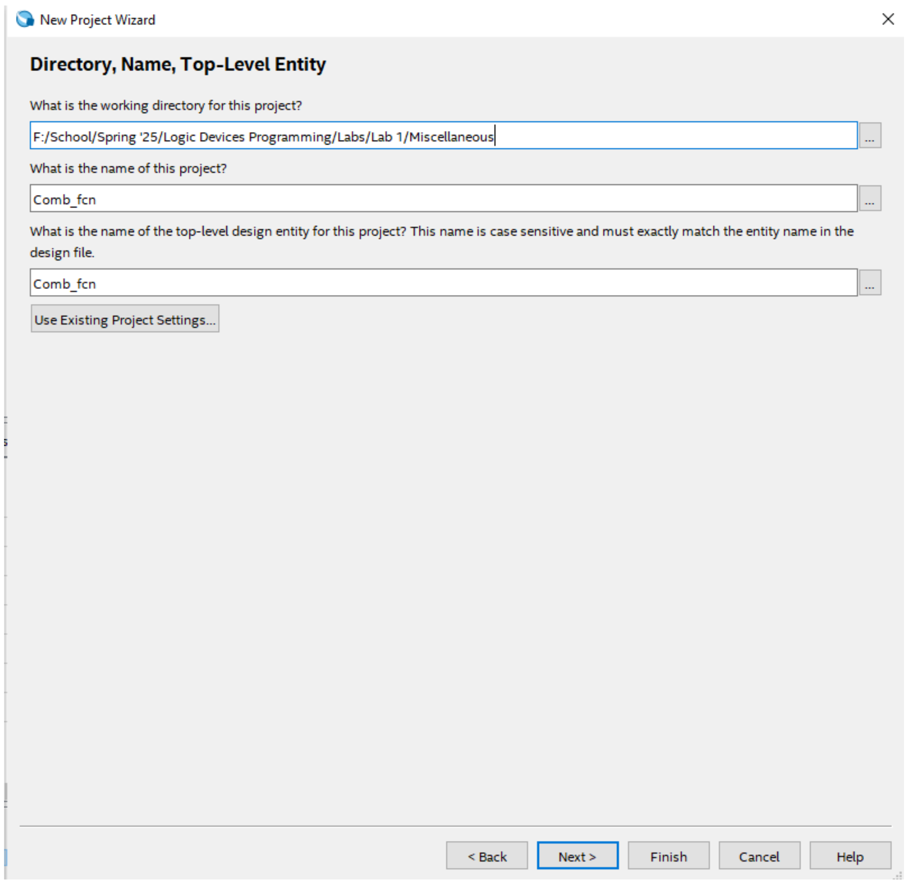
Installing the Quartus software is a simple task, however certain components of the installation must be completed successfully in order to ensure proper completion of this, and subsequent experiments. After creating an account on www.intel.com, the Quartus software can be downloaded, along with Modelsim and MAX 10 FPGA device support. Ensure the correct device support is installed.

In order for the computer to establish communication with the MAX10 board, the Altera USB-Blaster device driver must be properly installed. If it is not, plug the board into the computer, and open the computer's device manager. Under the *Unknown Devices*, select the USB-Blaster and update the drivers based on the folder that was downloaded along with Quartus.

Finally, move into the *Hardware Setup* section of the programmer within Quartus, and add the USB-Blaster device to the software. Then, select the correct device from the MAX10 device family (Max 10 10M50DAF484C7G)

Part 2 – Building a Project

First, the *New Project Wizard* is opened and a directory is selected for the project. For the sake of this course, a directory through a flash drive will be used such that the project can be opened both in the lab PCs, as well as the student's personal home PC. The filename will be *Combfcn*.



Directory, Name, Top-Level Entity

What is the working directory for this project?

F:/School/Spring '25/Logic Devices Programming/Labs/Lab 1/Miscellaneous

What is the name of this project?

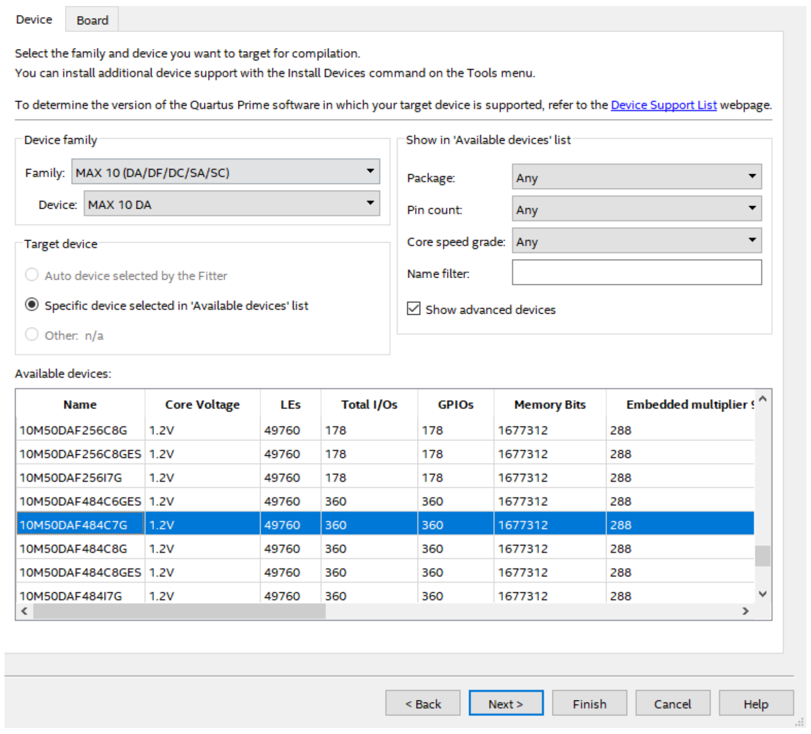
Comb_fcn

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.

Comb_fcn

Figure 1 - Project Directory Creation

The correct device is then selected from the *Device* tab, then the ModelSim-Altera setting is selected from the simulation settings.



Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the Device Support List webpage.

Device family

Family: MAX 10 (DA/DF/DC/SA/SC)

Device: MAX 10 DA

Target device

Auto device selected by the Fitter
[✓ Specific device selected in 'Available devices' list]
Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

Name filter:

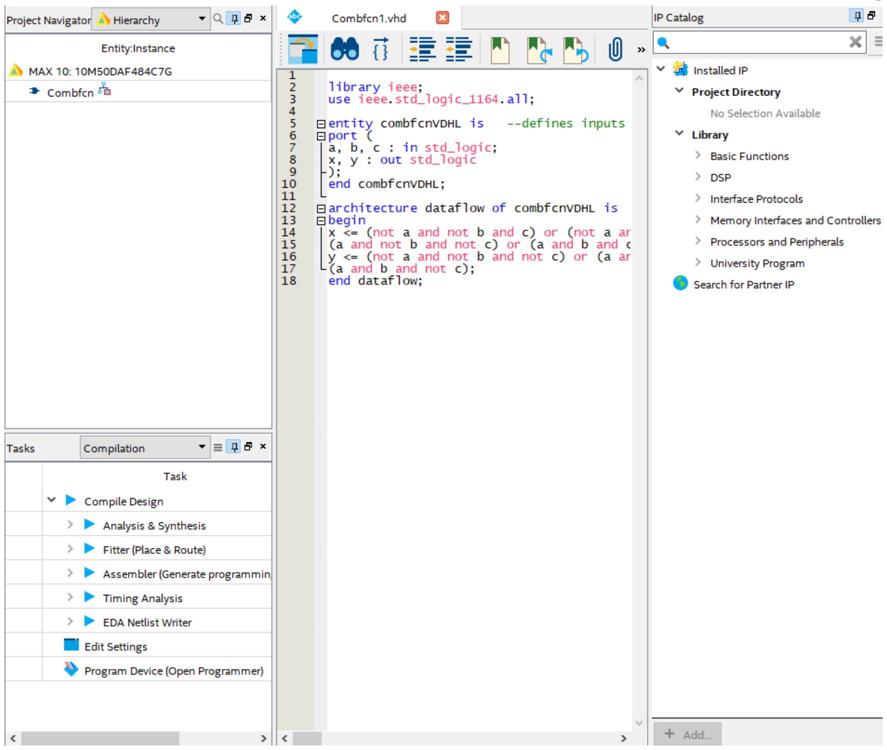
[✓ Show advanced devices]

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier
10M50DAF256C8G	1.2V	49760	178	178	1677312	288
10M50DAF256C8GES	1.2V	49760	178	178	1677312	288
10M50DAF256I7G	1.2V	49760	178	178	1677312	288
10M50DAF484C6GES	1.2V	49760	360	360	1677312	288
10M50DAF484C7G	1.2V	49760	360	360	1677312	288
10M50DAF484C8G	1.2V	49760	360	360	1677312	288
10M50DAF484C8GES	1.2V	49760	360	360	1677312	288
10M50DAF484I7G	1.2V	49760	360	360	1677312	288

Figure 2 - Selecting the Correct Device

Next, a new VHDL file is created, and saved as a filename that which is **DIFFERENT** than the filename assigned to the project. The code to be written into this file is located in **figure 3** below:



```

library ieee;
use ieee.std_logic_1164.all;

entity combfcnVDHL is   --defines inputs
port (
a, b, c : in std_logic;
x, y : out std_logic
);
end combfcnVDHL;

architecture dataflow of combfcnVDHL is
begin
x <= (not a and not b and c) or (not a ar
(a and not b and not c) or (a and b and c
y <= (not a and not b and not c) or (a ar
(a and b and not c);
end dataflow;
```

Figure 3 - VHD File with Written Code

Now, a *symbol file*, otherwise known as a **block diagram** is created from the *file->new* menu (name this file the same thing as the project). In the resulting opened window, double clicking in the empty space will allow the user to add the symbol file to the window.

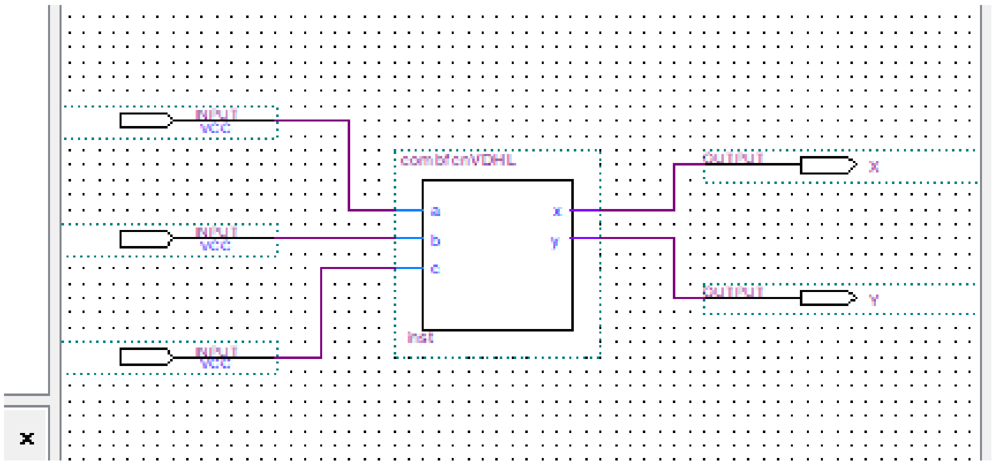


Figure 4 - Successfully Creating Block Diagram

Note that above, there are inputs and outputs connected to the symbol. These can be added to the schematic through the *Pin Tool* button, and renamed/connected accordingly. Pressing the blue play button will compile the work completed thus far.

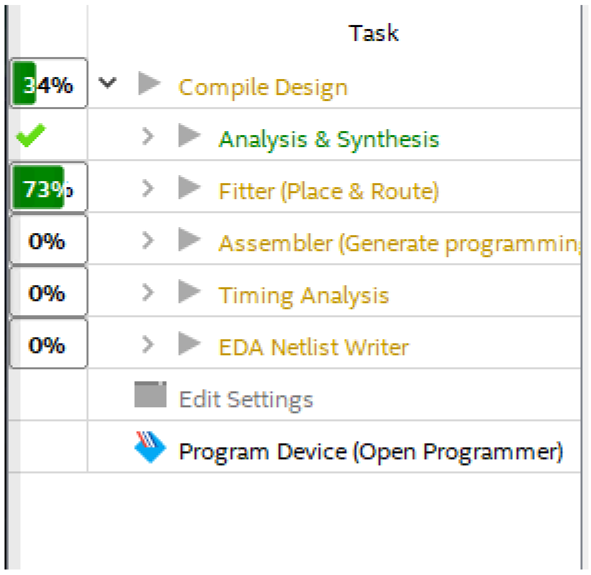
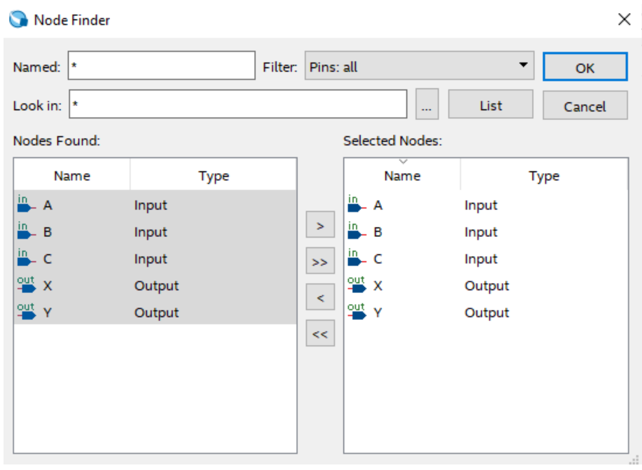
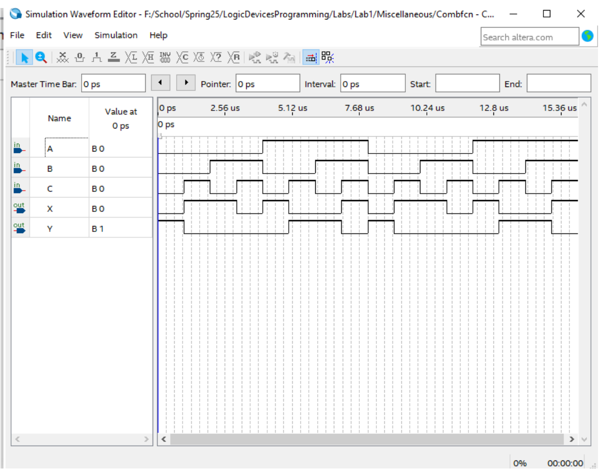


Figure 5 - First Compilation of the Project

Now a waveform should be generated to ensure proper logic flow throughout the initial program. A new *University Program VWF* file is created, and the *Node Finder* tool will be used to automatically add components to the waveform editor. In the *Node Finder*, under *List*, the >> symbol can be used to automatically add all nodes from the schematic board. The final step before simulating is manually setting the period for each component, those being decrementing multiples of 2.



Name	Type
A	Input
B	Input
C	Input
X	Output
Y	Output



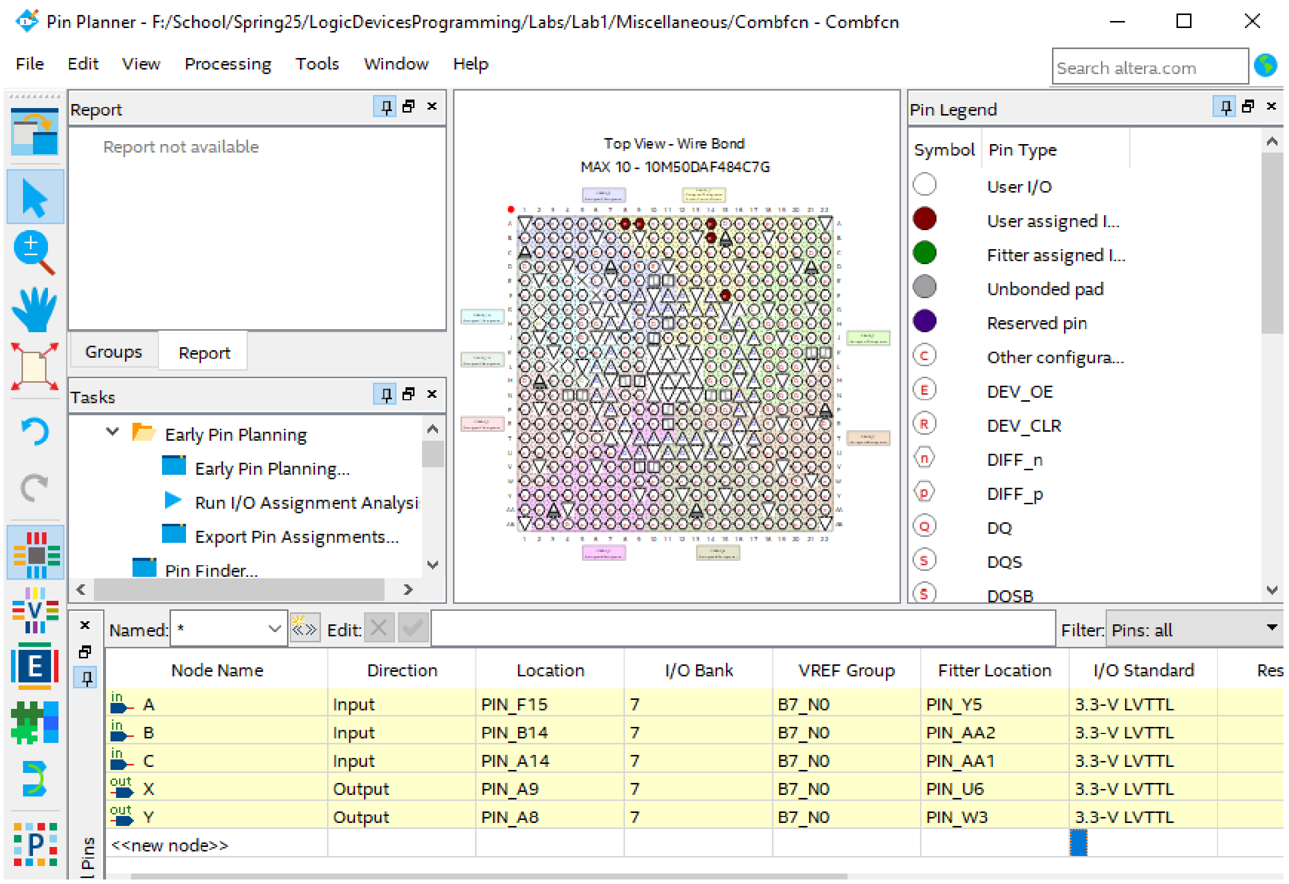
Name	Value at 0 ps
A	B 0
B	B 0
C	B 0
X	B 0
Y	B 1

Figure 6 – Node Finder Window

Figure 7 – Simulated Logic Waveform

Part 3 – Programming to the Board

The final step in this study is to compile the project, and program it's contents to the development board. The first, vital component to this process is pin assignment, which is a simple process within Quartus. Under the *Assignments* tab is the *Pin Planner* window, where the following pin assignments and IO standard should be set:



Pin Planner - F:/School/Spring25/LogicDevicesProgramming/Labs/Lab1/Miscellaneous/Combfcn - Combfcn

File Edit View Processing Tools Window Help

Report: Report not available

Tasks: Early Pin Planning; Early Pin Planning...; Run I/O Assignment Analysi; Export Pin Assignments...; Pin Finder...

Top View - Wire Bond
MAX 10 - 10M50DAF484C7G

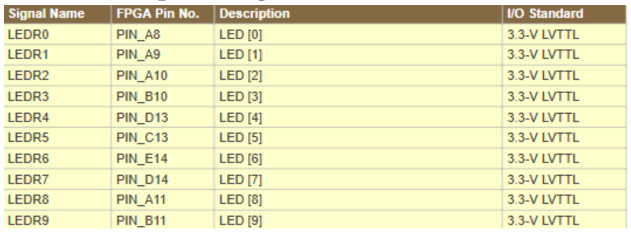
Symbol	Pin Type
○	User I/O
●	User assigned I...
●	Fitter assigned I...
●	Unbonded pad
●	Reserved pin
Ⓒ	Other configura...
Ⓔ	DEV_OE
Ⓡ	DEV_CLR
ⓝ	DIFF_n
ⓟ	DIFF_p
Ⓠ	DQ
Ⓢ	DQS
Ⓢ	DQSB

Named: * Edit: Filter: Pins: all

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Res
A	Input	PIN_F15	7	B7_N0	PIN_Y5	3.3-V LVTTL	
B	Input	PIN_B14	7	B7_N0	PIN_AA2	3.3-V LVTTL	
C	Input	PIN_A14	7	B7_N0	PIN_AA1	3.3-V LVTTL	
X	Output	PIN_A9	7	B7_N0	PIN_U6	3.3-V LVTTL	
Y	Output	PIN_A8	7	B7_N0	PIN_W3	3.3-V LVTTL	
<<new node>>							

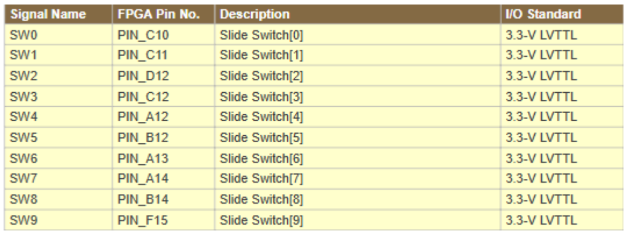
Figure 8 - Pin Assignments

The correct pin assignments can be found within the DE10 Lite User Manual:



Signal Name	FPGA Pin No.	Description	I/O Standard
LEDR0	PIN_A8	LED [0]	3.3-V LVTTL
LEDR1	PIN_A9	LED [1]	3.3-V LVTTL
LEDR2	PIN_A10	LED [2]	3.3-V LVTTL
LEDR3	PIN_B10	LED [3]	3.3-V LVTTL
LEDR4	PIN_D13	LED [4]	3.3-V LVTTL
LEDR5	PIN_C13	LED [5]	3.3-V LVTTL
LEDR6	PIN_E14	LED [6]	3.3-V LVTTL
LEDR7	PIN_D14	LED [7]	3.3-V LVTTL
LEDR8	PIN_A11	LED [8]	3.3-V LVTTL
LEDR9	PIN_B11	LED [9]	3.3-V LVTTL

Figure 9 – LED Pinout Table



Signal Name	FPGA Pin No.	Description	I/O Standard
SW0	PIN_C10	Slide Switch[0]	3.3-V LVTTL
SW1	PIN_C11	Slide Switch[1]	3.3-V LVTTL
SW2	PIN_D12	Slide Switch[2]	3.3-V LVTTL
SW3	PIN_C12	Slide Switch[3]	3.3-V LVTTL
SW4	PIN_A12	Slide Switch[4]	3.3-V LVTTL
SW5	PIN_B12	Slide Switch[5]	3.3-V LVTTL
SW6	PIN_A13	Slide Switch[6]	3.3-V LVTTL
SW7	PIN_A14	Slide Switch[7]	3.3-V LVTTL
SW8	PIN_B14	Slide Switch[8]	3.3-V LVTTL
SW9	PIN_F15	Slide Switch[9]	3.3-V LVTTL

Figure 10 – Switch Pinout Table

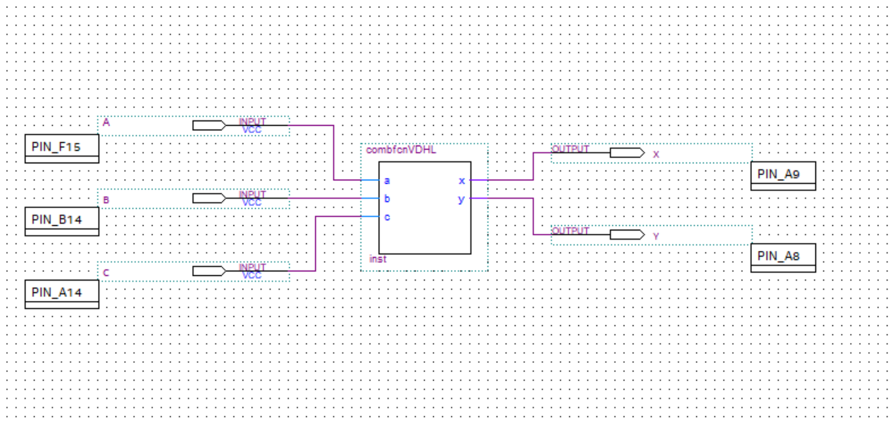
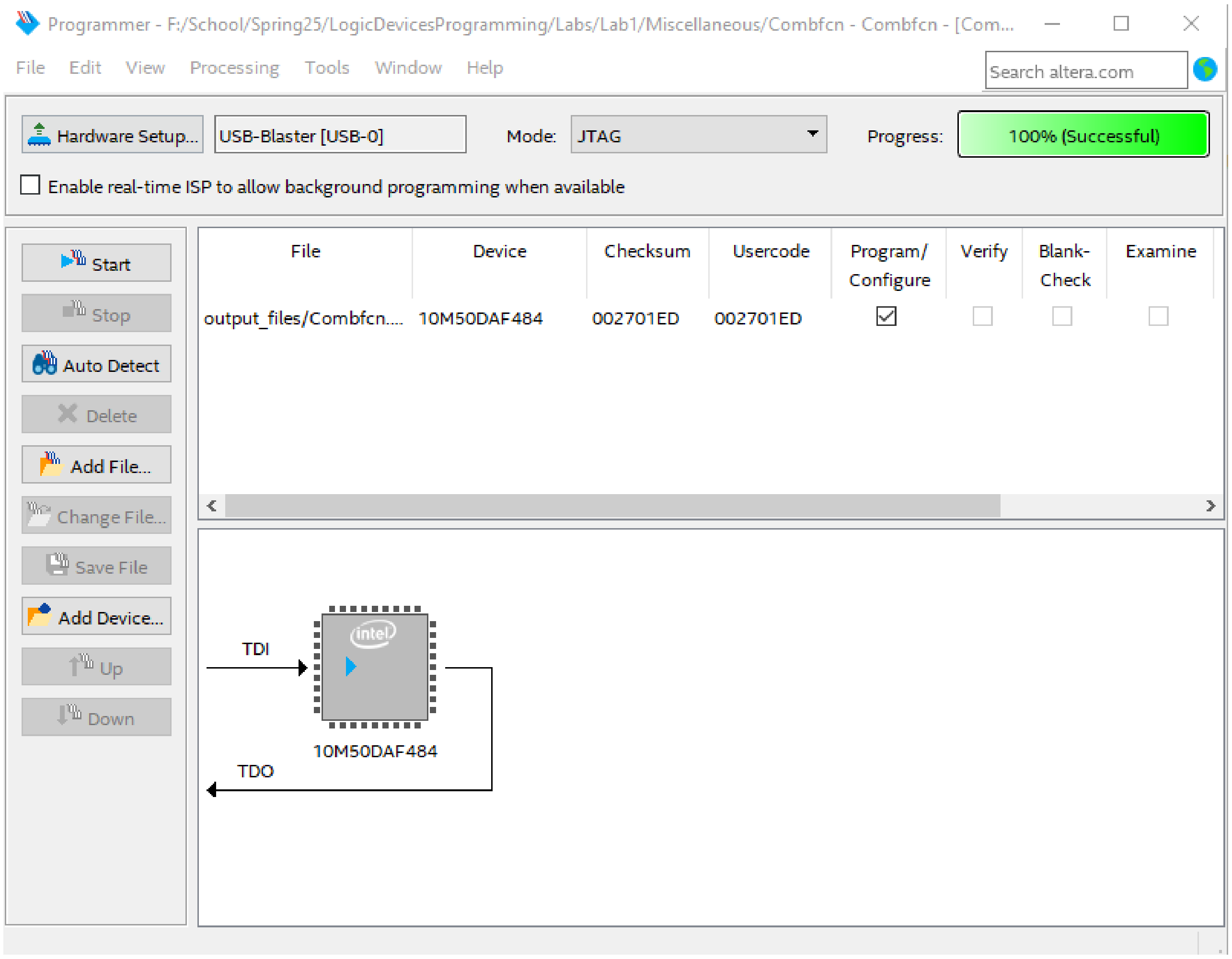


Figure 11 - Block Diagram with Pin Assignments

After assigning the correct pins, the project can be compiled once more, and the board can finally be programmed. To conduct this procedure, look to the compilation window, and select *Program Device*. Then, in the subsequent *Hardware Setup* window, ensure the correct device is listed, and select *Start*.



Programmer - F:/School/Spring25/LogicDevicesProgramming/Labs/Lab1/Miscellaneous/Combfcn - Combfcn - [Com...

Hardware Setup... USB-Blaster [USB-0] Mode: JTAG Progress: 100% (Successful)

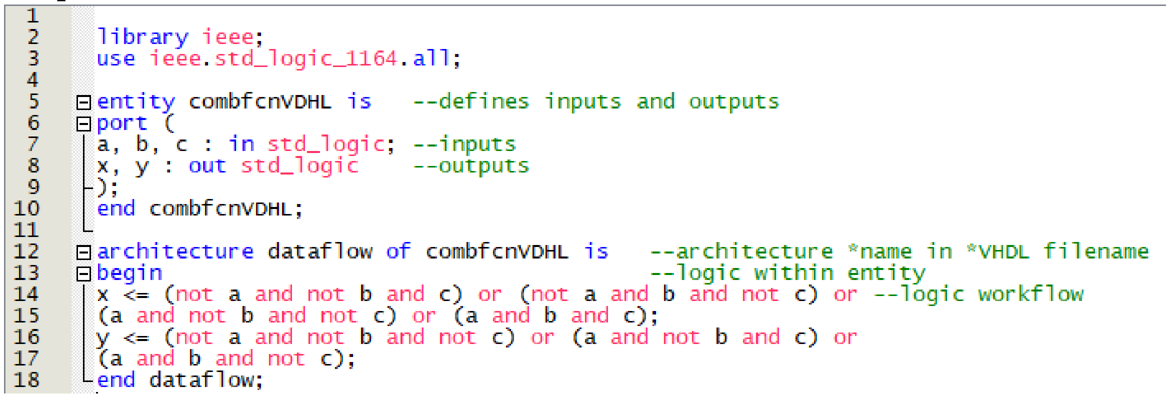
☐ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
output_files/Combfcn....	10M50DAF484	002701ED	002701ED	☑	☐	☐	☐

Figure 12 - Project Successfully Programmed to Board

DISCUSSION

Code Explanation



```

library ieee;
use ieee.std_logic_1164.all;

entity combfcnVDHL is   --defines inputs and outputs
port (
a, b, c : in std_logic; --inputs
x, y : out std_logic    --outputs
);
end combfcnVDHL;

architecture dataflow of combfcnVDHL is   --architecture *name in *VHDL filename
begin                                     --logic within entity
x <= (not a and not b and c) or (not a and b and not c) or --logic workflow
(a and not b and not c) or (a and b and c);
y <= (not a and not b and not c) or (a and not b and c) or
(a and b and not c);
end dataflow;
```

Figure 13 - VHD Code

Text Inclusion of Code

```
library ieee;
use ieee.std_logic_1164.all;

entity combfcnVDHL is   --defines inputs and outputs
port (
a, b, c : in std_logic; --inputs
x, y : out std_logic      --outputs
);
end combfcnVDHL;

architecture dataflow of combfcnVDHL is --architecture *name in *VHDL filename
begin                                                                          --
logic within entity
x <= (not a and not b and c) or (not a and b and not c) or --logic workflow
(a and not b and not c) or (a and b and c);
y <= (not a and not b and not c) or (a and not b and c) or
(a and b and not c);
end dataflow;
```

The program starts with the inclusion of the necessary library (IEEE Standard), followed by a *use* instruction which tells the compiler what definitions to use from IEEE (here, is .all or all definitions).

Next, the entity definition is instantiated, where the inputs and outputs can be defined of the VHDL. This is evident within the *port* line, where the input ports are defined, and the outputs follow.

Finally, the architecture named dataflow is created, where the logic workflow is created through a series of boolean expressions.

Validation of Data

The validation of any data collected for this study, which is minimal here as this experiment serves as a foundation for subsequent labs, are visible both in the provided waveform in the **procedures** section of this report, as well as the attached video showcasing the resulting board.

CONCLUSION

The experiment was conducted successfully and efficiently demonstrated the setup, foundational understanding, and navigation of both the Quartus software, as well as it's interface with a development board. If the lab were to be conducted again, more intricate, detailed code with different purposes could be added to the software to test against different parameters. This would allow for more components of the software to be tested against of such parameters. However, it should be noted that this study served as an introduction to the content discussed, and the inclusion of such ideas could overload or skew the original intention of the study.

REFERENCES

[1] Professor Ashley Evans, *Logic Devices Programming Lab Manual*, 1st ed. Orlando, FL: Valencia College, 2025.