

Project 2

**Designing a Simple System, Timing Analysis, and Working with the C
Language**

EE 371 AC

January 25, 2016

| NAME | SIGNATURE | STUDENT ID |
|------------------|------------------|-------------------|
| Denny Ly | | 1231185 |
| Minhhue H. Khuu | | 1329349 |
| Ruchira Kulkarni | | 1234324 |

Table of Contents:

| | | |
|--------|---|----|
| 1. | ABSTRACT | 3 |
| 2. | INTRODUCTION..... | 3 |
| 3. | DISCUSSION OF THE PROJECT | 3 |
| 3.1 | DESIGN SPECIFICATION | 3 |
| 3.1.1 | <i>The Interlock</i> | 4 |
| 3.1.2 | <i>Arrival Mode</i> | 4 |
| 3.1.3 | <i>Departure Mode</i> | 4 |
| 3.2 | DESIGN PROCEDURE..... | 5 |
| 3.2.1 | <i>Tools</i> | 5 |
| 3.2.2 | <i>Naming Scheme</i> | 5 |
| 3.2.3 | <i>State Diagram for the Interlock System</i> | 6 |
| 3.2.4 | <i>Implementation of the Time Delay</i> | 7 |
| 3.2.5 | <i>Implementation of the Pressure Differential and Limit Error Detector</i> | 8 |
| 3.2.6 | <i>Implementation of the Key Press Handler</i> | 8 |
| 3.3 | SYSTEM DESCRIPTION | 8 |
| 3.3.1 | <i>Top Level Module</i> | 8 |
| 3.3.2 | <i>Input</i> | 9 |
| 3.3.3 | <i>Output</i> | 10 |
| 3.3.4 | <i>New State Logic</i> | 11 |
| 3.4 | SOFTWARE IMPLEMENTATION | 13 |
| 3.5 | HARDWARE IMPLEMENTATION | 14 |
| 4. | TESTING | 14 |
| 4.1 | TEST PLAN..... | 14 |
| 4.1.1 | <i>iVerilog and gtkwave</i> | 15 |
| 4.1.2 | <i>Quartus II and Signal Tap</i> | 15 |
| 4.2 | TEST SPECIFICATION | 15 |
| 4.3 | TEST CASE..... | 15 |
| 5. | PRESENTATION, DISCUSSION, AND ANALYSIS OF THE RESULTS..... | 16 |
| 5.1 | IVERILOG AND GTKWAVE ANALYSIS | 16 |
| 5.1.1 | <i>Arrival Sequence</i> | 16 |
| 5.1.2 | <i>Departure Sequence</i> | 17 |
| 5.1.3 | <i>Error Signal</i> | 18 |
| 5.2 | QUARTUS II SIGNAL TAP ANALYSIS..... | 20 |
| 5.2.1 | <i>Initial State</i> | 20 |
| 5.2.2 | <i>Boarding State</i> | 21 |
| 5.2.3 | <i>Preparation State</i> | 21 |
| 5.2.4 | <i>Filling State</i> | 22 |
| 5.2.5 | <i>Outer Port Open State</i> | 23 |
| 5.2.6 | <i>Outer Port Close State</i> | 24 |
| 5.2.7 | <i>Draining State</i> | 24 |
| 5.2.8 | <i>Inner Port Open State</i> | 25 |
| 5.2.9 | <i>Inner Port Close State</i> | 26 |
| 5.2.10 | <i>Differential Pressure Error Trigger to Error State</i> | 27 |
| 5.2.11 | <i>Limit Pressure Error Trigger to Error State</i> | 28 |
| 5.3 | FAILURE MODE AND ANALYSIS..... | 29 |
| 5.3.1 | <i>Arrive Signal</i> | 29 |
| 5.3.2 | <i>Departure Signal</i> | 29 |
| 5.3.3 | <i>Fill Signal</i> | 29 |
| 5.3.4 | <i>Drain Signal</i> | 30 |

| | | |
|-------|-------------------------------------|----|
| 5.3.5 | <i>Open Outer Port Signal</i> | 30 |
| 5.3.6 | <i>Open Inner Port Signal</i> | 30 |
| 5.3.7 | <i>Difference Error</i> | 30 |
| 5.3.8 | <i>Limit Error Signal</i> | 30 |
| 5.3.9 | <i>Clock and Reset Signal</i> | 31 |
| 5.4 | ANALYSIS OF ANY ERRORS | 31 |
| 5.5 | ANALYSIS OF POSSIBLE ERRORS | 31 |
| 6. | SUMMARY AND CONCLUSION | 32 |
| 7. | AUTHORS AND CONTRIBUTIONS | 33 |
| 8. | APPENDIX | 34 |
| 8.1 | INTERLOCK SYSTEM | 34 |
| 8.2 | C PROGRAM (POINTERS)..... | 36 |
| 8.2.1 | <i>Design Requirements</i> | 36 |
| 8.2.2 | <i>Design Specifications</i> | 36 |
| 8.2.3 | <i>Results and Analysis</i> | 37 |

1. Abstract

The goal of this lab was to understand the design process of a system, in our case the system for implementing a bathysphere. We designed the bathysphere system based on the guideline of the provided spec and using the concepts of Mealy and Moore state diagrams learned in class to design a state diagram for the different states taken by the bathysphere. The bathysphere system that we designed was in the point of view of an aquanaut who was controlling the bathysphere. The bathysphere has a lot of different functionality that broadly entails arrival and departure. Inside the arrival and departure modes of the bathysphere, the interlock contains 2 directional ports: the inner and the outer port. The ports provide the following functionality: filling and draining the interlock chamber. It also has the functionality to detect errors in pressure (limit & differential) while executing this functionality. All of this is implemented using various control signals for each of these functions. Our lab resulted in a user-friendly functional system for bathysphere. In addition to the bathysphere, we also implemented a C program that helps us understand the importance and use of pointers.

2. Introduction

In this project we specify and aim to design an interlock between a modern day bathysphere and an experimental ocean bottom habitat. The bathysphere takes various different states based on the functionality the user or controller of the bathysphere wishes to achieve. This functionality is controlled by different control signals. These signals are arrive, depart, fill, prepare, limit error, differential error, drain, open outer and inner ports. We achieve this by modeling this system using Verilog HDL and then testing the model using iVerilog and gtkWave. We use this model and synthesize and transfer the design to the Cyclone V FPGA. On the FPGA, we test and check the operation of the bathysphere using the different DE1_SoC switches, LEDRs, HEX Displays and KEYS. The different states of the bathysphere are achieved by keeping track of the previous state of the bathysphere as well as the current control signal input using different always blocks. The complete functionality of the implementation is discussed below. In addition to this we also wrote a C program to familiarize ourselves with the language and understand how to use pointers by implementing a simple equation.

3. Discussion of the Project

The overall goal of this project is to design an interlock system between a modern day bathysphere and an experimental ocean bottom habitat. The interlock system will be controlled by a fixed set of control signals as well as sensor signals that will be triggered based on the a differential and limit air pressure signal. There will be time delays in between particular states to allow for the process of draining, filling, and waiting to board. This section will explain the details of the (1) Design Specification, (2) Design Procedure, (3) System Description, (4) Software Implementation, and (5) Hardware Implementation.

Additionally, this project contains a C program that contains pointer arithmetic. The C program will be explained in detail in the Appendix located at Section 8.2 C Program (Pointers).

3.1 Design Specification

The design specification is broken up into three categories: (1) the Interlock, (2) Arrival, and (3) Departure.

3.1.1 The Interlock

The interlock system supports the following:

1. The interlock is comprised of 2 bidirectional ports (one on either side of the interlock)
2. A port can only be opened if the pressure difference across the port is less than 0.1 atm.
3. The interlock chamber can be filled with water and pressurized up to 16000 psi and flushed and depressurized to 13.0 psi.

3.1.2 Arrival Mode

During arrival the system supports the following:

1. The bathysphere must signal the docking port 5 minutes before arrival.
2. If the interlock chamber is empty, the signal from the bathysphere should cause the inner and outer ports of the interlock to be closed and sealed.
3. After the ports are closed and sealed, the chamber shall be filled and pressured to a pressure difference between the chamber interior and the outside of the outer port to less than 0.1 atm.
4. When the pressure difference between the inside and exterior of the chamber reaches less than 0.1 atm, the outer port shall be opened.
5. Filling and pressurizing the chamber takes 7 minutes.
6. When the outer port is fully opened, the bathysphere can enter the interlock.
7. When the bathysphere is fully inside the interlock, the outer port is closed and sealed.
8. After the ports are closed and sealed, the chamber shall be evacuated and managed to a pressure difference between the chamber interior and the outside of the inner port to less than 0.1 atm.
9. When the chamber has been evacuated and the pressure difference between the chamber interior and the exterior of the inner port reaches less than 0.1 atm, the inner port shall be opened.
10. Emptying and depressurizing the chamber takes 8 minutes.
11. When the inner port is opened, the passenger can exit the bathysphere.

3.1.3 Departure Mode

During departure the system supports the following:

1. When the bathysphere is to leave the interlock, the outer port must be closed and sealed.
2. After the outer port is closed and sealed and if the interlock chamber is empty and the pressure difference between the exterior and the inner port and the interlock chamber is less than 0.1 atm, the inner port can be opened and the passengers can enter the interlock and the bathysphere.
3. The bathysphere must signal the docking station 5 minutes prior to departure.
4. The signal from the bathysphere should cause the inner and outer ports of the interlock to be closed and sealed.
5. After the ports are closed and sealed and the chamber flooded, the pressure difference between the chamber interior and the exterior of the outer port shall be managed and controller to less than 0.1 atm.
6. When the pressure difference between the interior of the chamber and the exterior of the outer port reaches less than 0.1 atm, the outer port shall be opened.

7. Filling and pressurizing the chamber takes 7 minutes.
8. When the outer port is fully opened, the bathysphere can exit the interlock.
9. When the bathysphere is fully outside the interlock, the outer port is closed and sealed and the interlock emptied and depressurized to 1 atm.
10. When the pressure difference between the chamber interior and the exterior of the inner port reaches less than 0.1 atm, the inner port can be opened.
11. Pressurizing the chamber takes 8 minutes.

3.2 Design Procedure

3.2.1 Tools

The lab requires the use of iVerilog and gtkwave to produce waveforms of the design for testing and analysis. Quartus II Signal Tap Logic Analyzer is also another tool that can be used to see the output waveforms from the various structures that were built in the lab. The RTL Viewer will be used to examine the differences of the different levels of the modelling language. The gcc compiler will be used to compile C to be ran as an executable on a given computer machine. Finally, to generate the Verilog and C code, Notepad++ was used as the text editor. The tools used in this lab is shown in Table 1.

Table 1. Table of Tools Used for this Lab

| Tool | Purpose |
|-------------------------------------|--|
| iVerilog | Synthesize Verilog files |
| gtkwave | Output waveform of Verilog files |
| Quartus II Signal Tap | Output waveform of hardware result |
| Quartus II RTL Viewer | Synthesizes a structural model of Verilog file |
| Cyclone V FPGA (Model 5CSEMA5F31C6) | Platform to implement project design |
| GCC | C compiler for C files |
| Notepad++ | Text editor |

3.2.2 Naming Scheme

In this project, aliases were assigned to each state to allow for a shorter naming scheme that provides better Verilog coding style. The table that translates a state name to its alias used in Verilog and the State Diagram is shown in Table 2.

Table 2. State Name to Alias use in Verilog and State Diagram

| State Name | Alias (used in Verilog and State Diagram) |
|---------------------------------|--|
| Initial | INIT |
| Preparation | PREP |
| Waiting to fill | WAIT_FILL |
| Filling | FILLING |
| Waiting for outer port to open | WAIT_OPORT_OPEN |
| Waiting for outer port to close | WAIT_OPORT_CLOSE |
| Waiting to drain | WAIT_DRAIN |
| Draining | DRAINING |
| Waiting for inner port to open | WAIT_IPORT_OPEN |
| Waiting for inner port to close | WAIT_IPORT_CLOSE |
| Boarding | WAIT_USER |
| Error | ERROR |

3.2.3 State Diagram for the Interlock System

To design the interlock system, a state diagram was created that explains the change of state from one state to another based on given trigger condition. The state diagram consisted of 12 states: (1) initial, (2) preparation, (3) waiting to fill, (4) filling, (5) wait outer port to open, (6) wait outer port to close, (7) waiting to drain, (8) draining, (9) wait inner port to open, (10) wait inner port close, (11) boarding, and (12) error.

The state diagram is shown in Figure 1.

Table 3. Old and new wait time

| State | Old Time | New Time |
|-------------|-----------|-------------|
| Preparation | 5 Minutes | 5 seconds |
| Filling | 7 Minutes | 7~8 seconds |
| Draining | 8 Minutes | 8~9 seconds |

The time delay was implemented using a counter that incremented each clock cycle. When the counter variable matches the targeted bit pattern, the next state will be triggered and the counter variable is reset back to 0. To find the matching bit pattern, a test was done empirically to find a bit pattern that matched a time delay that was roughly 5 seconds. The matching bit pattern was 16'b1000000000000000. Once, a bit pattern was found for 5 seconds, the bit pattern can be manipulated to find 7.5 seconds which is 16'b1100000000000000 and 8.75 seconds, which is 16'b1110000000000000. The computed time result is shown in Table 4.

Table 4. Matching bit pattern for count

| State | Count Bit Pattern | Time to Compute |
|-------------|----------------------|-----------------|
| Preparation | 16'b1000000000000000 | 5 seconds |
| Filling | 16'b1100000000000000 | 7.5 seconds |
| Draining | 16'b1110000000000000 | 8.75 seconds |

3.2.5 Implementation of the Pressure Differential and Limit Error Detector

To model the effect of a pressure detection unit for the differential limit of less than 0.1 atm and a pressure limit of being filled up to 16000 psi and flushed and depressurized to 13.0 psi, two modules were implemented to produce error signals, respectively. The detector modules require the user to have a switch on to determine which error signal needs to be flagged, and a keypress to activate or deactivate the error respective error signal. The error signal is also set to low asynchronously the instant the reset key is active.

3.2.6 Implementation of the Key Press Handler

To negate the effect of have a key press register as multiple keypresses within a single clock period, a key press handler was implemented. The key press handler module, detects a key press, and returns only a single keypress per clock cycle. This prevented the user from being able to alternate between error states in a single button press.

3.3 System Description

3.3.1 Top Level Module

The block diagram of the whole system is shown in Figure 2.

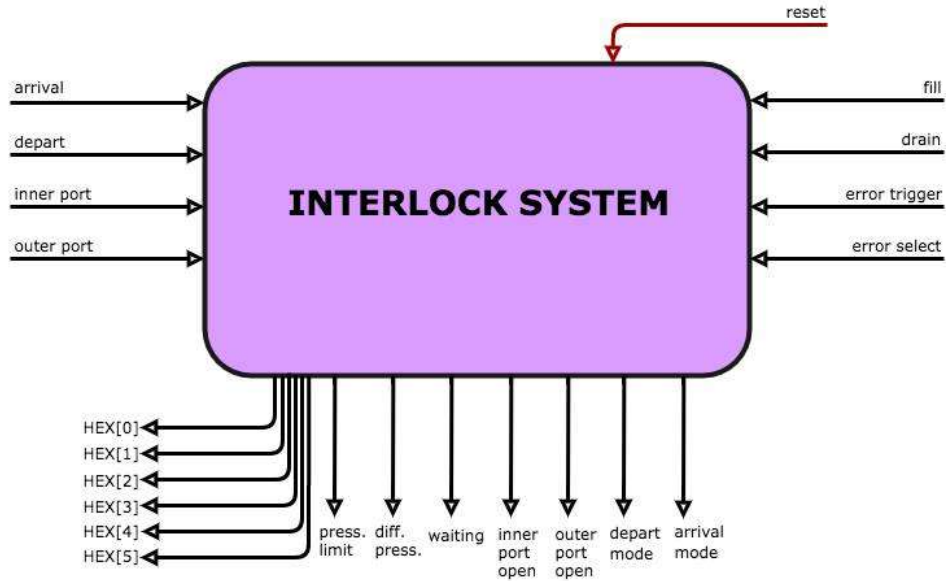


Figure 2. Block Diagram of the Interlock System Project.

3.3.2 Input

The input signals for the system consists of the following signals shown in Table 5.

Table 5. Input Signals

| Input Signal | Value Range | Description |
|---------------|--------------|--|
| Arrival | 1'b0 1'b1 | Signals an arrival mode request on active high. |
| Depart | 1'b0 1'b1 | Signals a depart mode request on active high. |
| Inner Port | 1'b0 1'b1 | Signals the inner port to open on active high and close on active low. |
| Outer Port | 1'b0 1'b1 | Signals the outer port to open on active high and close on active low. |
| Fill | 1'b0 1'b1 | Signals the fill operation on active high. |
| Drain | 1'b0 1'b1 | Signals the drain operation on active high. |
| Error Trigger | 1'b0 1'b1 | Signals the activation of an error signal based on an error select. |
| Error Select | 1'b0 1'b1 | Selects the error signal, active high controls the limit error signal, active low controls the differential error signal |
| Reset | 1'b0 1'b1 | Resets the system back to the initial state, asynchronously, on active high. |

3.3.3 Output

The output signals for the system consists of the following signals shown in Table 6.

Table 6. Output Signals

| Output Signal | Value Range | Description |
|--------------------|---|--|
| HEX[5:0] | Each seven segment display has range of 7b'0000000 7b'1111111 Where each bit represents a segment. | The 6 digit, seven segment hex display used to display the present state the user is current in (see Table 7 for more information) |
| Pressure Limit | 1'b0 1'b1 | The LED that is active when the limit error signal is active. |
| Differential Limit | 1'b0 1'b1 | The LED that is active when the differential error signal is active. |
| Waiting | 1'b0 1'b1 | The LED that blinks for a fixed time interval based on the waiting duration of the preparation, filling, and draining states. At the end of waiting, the LED is low. |
| Inner Port Open | 1'b0 1'b1 | The LED that is active when the inner port door is open. |
| Outer Port Open | 1'b0 1'b1 | The LED that is active when the outer port door is open. |
| Depart Mode | 1'b0 1'b1 | The LED that is active when the system is entering a depart mode sequence. |
| Arrival Mode | 1'b0 1'b1 | The LED that is active when the system is entering an arrive mode sequence. |

The corresponding HEX display output for a given present state is shown in Table 7. Since the HEX display outputs 6 characters, any HEX display output that does not have 6 characters will have the rest of the empty characters as an empty HEX display outputs.

Table 7. Present State's Hex Display Output

| Present State | Hex Display |
|--------------------------|--------------------|
| Initial | Start |
| Boarding | Board |
| Preparation | Prep |
| Waiting to Fill | Hold-f |
| Filling | Fill |
| Wait Outer Port to Open | Oopen |
| Wait Outer Port to Close | Oclose |
| Waiting to Drain | Hold-d |
| Draining | Drain |
| Wait Inner Port to Open | Iopen |
| Wait Inner Port to Close | Iclose |
| Error | Error |

3.3.4 New State Logic

The system has 12 states as shown in the state diagram shown in Figure 1. The full detail of the new state logic for each of the present state is shown in Table 8.

Table 8. Present State to New State logic

| Present State | New State Logic |
|--------------------------|--|
| Initial | <p>The state that the system is initially on.</p> <p>The state that is set when the reset signal is active.</p> <p>If the arrive signal is active, the next state is set to the boarding state.</p> <p>If the inner port signal is active, the next state is the boarding state.</p> <p>Otherwise, the next state remains the initial state.</p> |
| Boarding | <p>If the depart signal is high and the inner port signal is low, then the next state is the preparation state.</p> <p>Otherwise, the next state remains the boarding state.</p> |
| Preparation | <p>During the preparation state, a counter will start counting to set a time limit.</p> <p>The wait time is roughly 5 seconds.</p> <p>If the counter matched the bit pattern, the next state is set waiting to fill state.</p> <p>Otherwise, the next state remains the preparation state.</p> |
| Waiting to Fill | <p>If the fill signal is active and the depart signal is low and the arrive signal is low, the next state is the filling state.</p> <p>Otherwise, the next state remains the waiting to fill state.</p> |
| Filling | <p>If the limit error signal is active, then the next state is the error state.</p> <p>If the counter matched the bit pattern, the next state is wait outer port to open state.</p> <p>The wait time is roughly 7.5 seconds.</p> <p>Otherwise, the next state remains the filling state.</p> |
| Wait Outer Port to Open | <p>If the differential error signal is active, the next state is the error state.</p> <p>If the outer port signal is active, the next state is wait outer port to close state.</p> <p>Otherwise, the next state remains the wait outer port to open state.</p> |
| Wait Outer Port to Close | <p>If the outer port signal is low, then the next state is wait outer port to waiting to drain state.</p> <p>Otherwise, the next state remains the wait outer port to close state.</p> |
| Waiting to Drain | <p>If the drain signal is active, the next state is the draining state.</p> <p>Otherwise, the next state remains the waiting to drain state.</p> |
| Draining | <p>If the limit error signal is active, then the next state is the error state.</p> <p>If the counter matched the bit pattern, the next state is wait inner port to open state.</p> <p>The wait time is roughly 7.5 seconds.</p> <p>Otherwise, the next state remains the filling state.</p> |
| Wait Inner Port to Open | <p>If differential pressure error signal is active, the next state is the error state.</p> <p>If the inner port signal is high, the next state is wait inner port to close.</p> <p>Otherwise, the next state remains the wait inner port to open state.</p> |
| Wait Inner Port to Close | <p>If the inner port signal is low, the next state is the initial state.</p> <p>Otherwise, the next state remains the wait inner port to close state.</p> |
| Error | <p>Permanently remain in error state, until the reset signal.</p> |

3.4 Software Implementation

The interlock system was implemented using software through the use of Verilog. The code for this lab can be found in the included zip file submitted on canvas. Table 9 shows the filenames of implemented code after extraction.

Table 9. File name of code written for this lab

| File name of Code | Purpose |
|--------------------------------|--|
| /code/Verilog/DE1_SoC.v | Top level module for the interlock project. |
| /code/Verilog/DFlipFlop.v | D Flip Flop module use for new state logic. |
| /code/Verilog/diff_pressure.v | Differential pressure module for detection of differential pressure error signals. |
| /code/Verilog/limit_pressure.v | Limit pressure module for detection of limit pressure error signals. |
| /code/Verilog/inputHandler.v | Input handler module for managing key button presses. |
| /code/Verilog/interlock.v | The interlock system, which contains the state diagram logic. |
| ./code/C/pointer.c | The C program for the pointer arithmetic. |

The C program will be explained in the Appendix located at Section 8.2 C Program (Pointers).

The methods of implementation of the entirety of the interlock system is explain in detail on Section 3.2 Design Procedure. The explanation of the system properties of the interlock system is explained in detail on Section 3.3 System Description.

The Verilog code written in this assignment is purely behavioral and was designed based off of the state diagram that was generated in Figure 1. To implement the state diagram, an always block being triggered at predetermined conditions and case statements in conjunction with if else statements were used to create the new state logic (see Figure 3).

```
always @ (<insert trigger condition>) begin
  case(PS)
    <insert present state parameters>: begin
      if (<insert condition>) .....
      else .....
    end
    ...
    ...
    default: ...
  endcase
end
```

Figure 3. Verilog Code for New State Logic using always block with case statements.

By using multiple instances of always case blocks, as shown in Figure 3, behavioral code can be used to model the change in the new state logic, output logic, and counter for time delays. In particular, the new state variable is updated using D Flip Flop to update the present state value in the next clock cycle.

3.5 Hardware Implementation

The interlock system was implemented in hardware using the DE1_SoC board by compiling and synthesizing the Verilog code written into the board using Quartus II. After synthesizing the code into the DE1_SoC board, the pin configuration was assigned to assign the input and output of the interlock system as described in the top level module in Section 3.3.1 Top Level Module. The signal to FPGA pin pairing can be found in Table 10 at the Appendix in Section 8.1 Interlock System.

The code written maybe purely behavioral, however by using the RTL viewer tool in Quartus, the hardware gate level implementation of the interlock system and is shown in Figure 4.

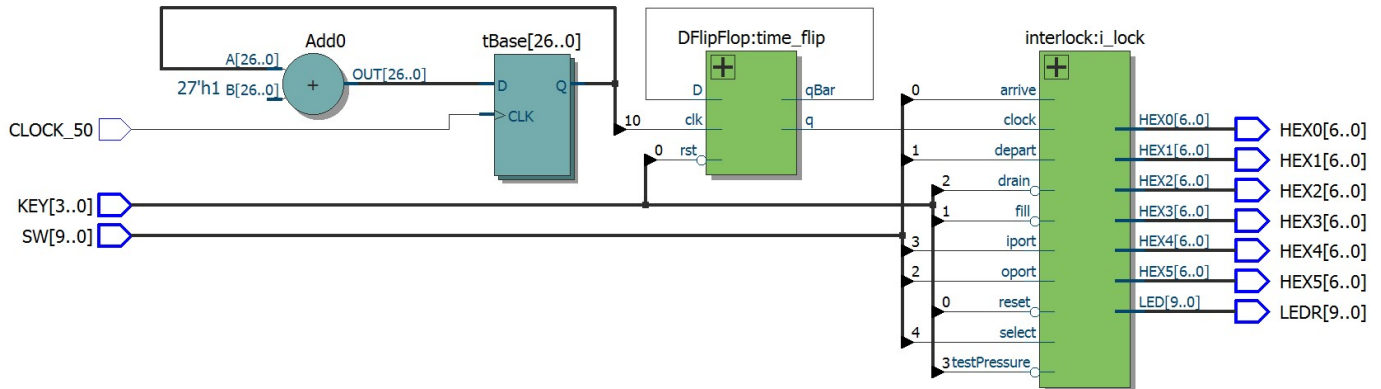


Figure 4. Hardware Gate Level Implementation of the Interlock System using Quartus RTL viewer.

The RTL viewer tool, produced a top level schematic that contained all of the input and output in the system. The contents of the interlock module in the RTL viewer consisted of multiple muxes and logic gates.

4. Testing

The testing of the design was broken up into three segments, all of which contributed to the testing process: (1) Test Plan, (2) Test Specifications, and (3) Test Cases. The test cases are developed based on the specifications of the test for the general test plan for each testing tool. The results of the test are shown in Section 3.

4.1 Test Plan

In this lab, the followings tools were used to test the interlock system and the differential and limit error signals.

- iVerilog and gtkwave

- Quartus II and Signal Tap

4.1.1 iVerilog and gtkwave

The process of testing in iVerilog and gtkwave consists of the following steps:

- 1) Design a tester module in Verilog to generate the input signals for the unit under test.
- 2) Design a testbench module in Verilog to generate a .vcd file to view the waveform generated from the unit under test.
- 3) Synthesize associated Verilog files in iVerilog and run the synthesized Verilog program to generate a .vcd file and run gtkwave to view the waveform using the batch commands as shown in Figure 5.

```
1. iverilog -o out <insert associated Verilog .v files>
2. vvp out
3. gtkwave <insert associated gtkwave .vcd file>
```

Figure 5. Command Prompt commands to display waveform from Verilog file.

- 4) Analyze the waveform to see if the waveform has any similarities or irregularities from predicted results from the test specifications.

4.1.2 Quartus II and Signal Tap

The process of testing in signal tap consists of the following steps:

- 1) Design a top level module in Verilog under the Quartus II IDE.
- 2) Assign the output described in Section 3.3.3 Output on the DE1-SoC board to display the output for a given present state.
- 3) Assign the input to the inputs described in Section 3.3.2 Input on the DE1-SoC board.
- 4) Synthesize and open up Signal Tap and set input, output, and clock ports for testing.
- 5) Set the trigger conditions for test condition.
- 6) Set clock and sample depth to CLOCK_50 and 1024 samples, respectively.
- 7) Load the .sof file into the DE1-SoC board.
- 8) Run the Signal Tap and utilize the input signal to activate the trigger condition.
- 9) Analyze the waveform to see if the waveform has any similarities or irregularities from predicted results from the test specifications.

4.2 Test Specification

The test specification matches the system description in Section 3.3 System Description. Therefore, the main test specification consists of the following:

- The new state assigned is correctly assigned when the trigger event occurs
- The new state remains the same when the trigger event does not occur
- The output matches the present state

All of these test specifications are defined in the tables that explain the new state logic, input, output, and hex display output shown in Figure 1, Table 5, Table 6, and Table 7, respectively.

4.3 Test Case

The test cases that will be tested are the following:

- The Arrival Sequence
 - The arrival sequence consists of activating the following signals: ARRIVE, wait 5 seconds, FILL, wait 7 seconds, OPORT, !OPORT, DRAIN, wait 8 seconds, IPORT, !IPORT.
 - Reset returns back to the initial state.
- The Departure Sequence
 - The departure sequence consist of activating the following signals: IPORT, DEPART, wait 5 seconds, FILL, wait 7 seconds, OPORT, !OPORT, DRAIN, wait 8 seconds, IPORT, !IPORT.
 - Reset returns back to the initial state.
- Error State analysis
 - The error occurs when the limit pressure error signal is active high and the state is draining or filling
 - The error occurs when the differential pressure error signal is active high and the state is waiting for inner port open or waiting for outer port open.

To validate the waveform of each sequence, we compare the result from the generated waveform with the expected results defined in the tables that explain the new state logic, input, output, and hex display output shown in Figure 1, Table 5, Table 6, and Table 7, respectively.

5. Presentation, Discussion, and Analysis of the Results

In this first part of the lab, Verilog was used to design and build the interlock system between a bathysphere and an ocean bottom habitat. After completing the design for the system, the modules were compiled using iVerilog and then simulated using the gtkwave tool.

5.1 iVerilog and gtkwave Analysis

5.1.1 Arrival Sequence

Figure 6 shows the simulation of the interlock system when a bathysphere is arriving at the ocean bottom habitat. The simulation has input signals from the clock, reset, arrival, departure, fill, drain, outer port, inner port, select, and test pressure. The outputs are the HEX displays and the LEDs. The HEX display shows the state that the system is currently in. The LEDs displays the current signals being used and what sequence the interlock system is currently completing. There are also a differential pressure error and a limit pressure error signals that will be turned on from the select and test pressure inputs. The counter is used to help determine the wait times during the draining, filling, and preparation periods.

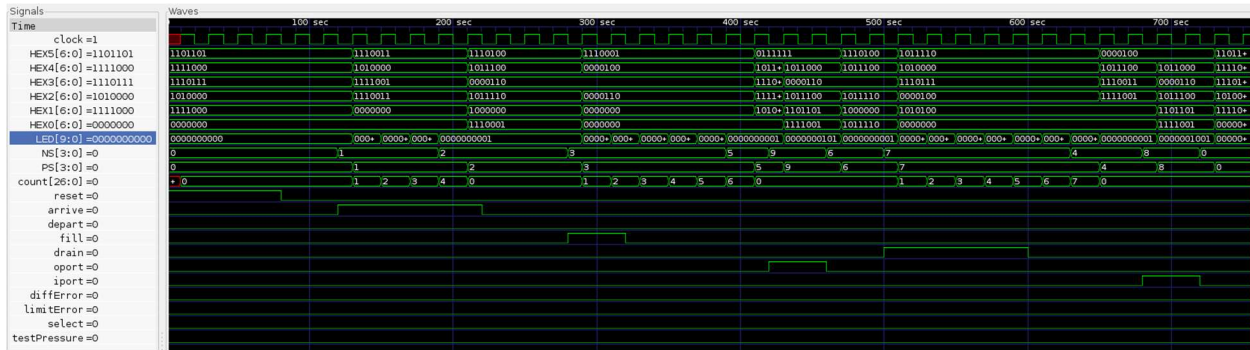


Figure 6. Arrival Sequence Waveform in gtkwave.

The system begins with the reset signal on. When the reset signal is on, the interlock stays in the chamber empty state. The empty state is when the interlock has both inner port and outer port closed. The pressure of the exterior of the inner port and the interlock is within the .1 atm difference.

After reset is turned off, the chamber waits for either an arrival signal or the inner port to open. These two signals determine if the next sequence of operations is for the bathysphere to enter or leave the interlock system. The simulation below shows that the system will receive an arrival signal first. The arrival signal signifies interlock to prepare for an arrival of a bathysphere. The waiting period is around 4 clock cycles to represent the 5 minutes. During waiting periods, the system must wait until the counting is done a pressure limit error occurs in the middle. Errors will be looked at later in the report.

After the five minutes, the system waits for the fill signal before pressurizing the interlock chamber to the exterior of the outer port's pressure. When the fill signal is received, the interlock will enter the filling stage. The waiting period of 7 minutes is represented using the 6 clock cycles.

Following the pressurization of the interlock, the system waits for the outer port to receive a signal to open the port. The port will then open and let the bathysphere to enter. Once the bathysphere is fully inside the interlock, the outer port will be able to close.

Next the interlock will wait for a drain signal. When the drain signal is high, the interlock begins to depressurize to the exterior inner port's pressure. The duration of the draining is 8 minutes, but represented with the 7 clock cycles in the simulation.

Finally, the chamber will wait for a signal to open the inner port. Opening the inner port will allow the passenger(s) from the bathysphere to enter. After everyone has entered, the inner port will close and be in the empty chamber state again. The chamber will now wait for an arrival or inner port open signal before doing anything else.

5.1.2 Departure Sequence

The interlock chamber's departure sequence will now be observed. In Figure 7, the simulation for the entire departure sequence is shown. The departure sequence is starts in the empty chamber

state and waits for the inner port open signal to signify that a bathysphere is departing the interlock chamber. Once the inner port is opened, the passenger(s) will enter the bathysphere. The boarding stage of the sequence will last until the inner port is closed and a departing sequence from the bathysphere is sent to the docking port.

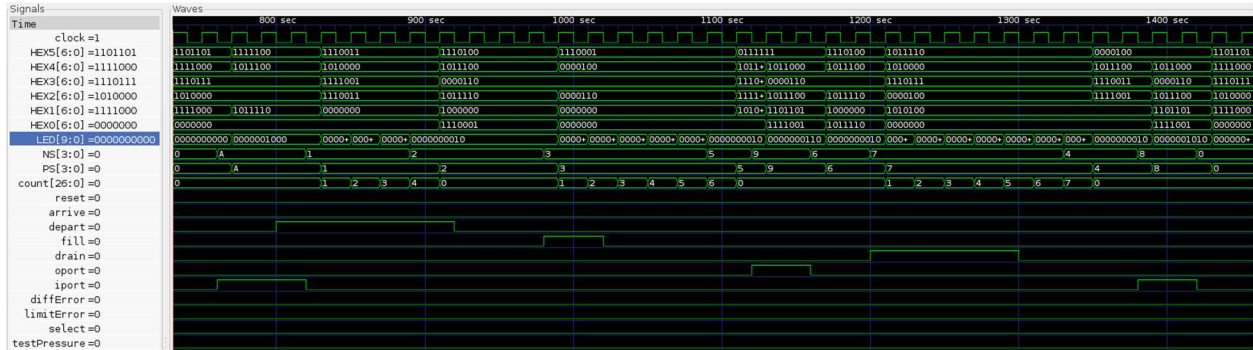


Figure 7. Depart Sequence Waveform in gtkwave.

When the inner port is closed and the departure signal is sent, the interlock will go into the preparation stage. The preparation stage is like the one from the arrival sequence. The system will not be able to do anything during this period of time. The waiting period is also 5 minutes and represented with 4 clock cycles in the simulation.

After the preparation stage, the system will wait for the fill signal to initiate the pressurization of the chamber. The fill signal will start the counter to 6 clock cycles (7 minutes). Following the filling stage is waiting for the outer port to be opened. The outer port signal will then open the port so that the bathysphere can leave. Once the bathysphere is outside of the interlock chamber, the outer port will close.

When the outer port closes, the system will go through a series of actions to get back to the empty chamber state. The drain signal will then be sent to depressurize the chamber to the exterior inner port's pressure levels. The draining period will go on for 7 clock cycles (8 minutes) and wait for the inner port to be opened again.

Once the inner port is opened, the system knows that the depressurization was successful because there were no errors. The inner port is then closed so the chamber can be in the empty chamber state again.

5.1.3 Error Signal

Now, the error signals will be looked at to see the effects on the system. The interlock system has two types of errors that could stall the entire sequence. The first error is if the differential pressure after filling or draining the chamber is not less than the .1 atm. The second error is if the pressure inside the chamber is greater than 16,000 psi or less than 13 psi when the system is in the filling or draining state.

In Figure 8 and Figure 9, the differential pressure error is turned on in the middle of the interlock sequence. To simulate the differential pressure error, the select signal needs to be low

and the test pressure needs to be turned on. If the system is currently trying to open either ports when the differential pressure is beyond the given boundaries, the interlock will go into an error state where only a reset signal can escape the state. The error state forces the interlock users to fix the problem before using the interlock again.



Figure 8. Error Inner Port Waveform in gtkwave.



Figure 9. Error Outer Port Waveform in gtkwave.

In Figure 10 and Figure 11, the limit pressure error is signaled on in the middle of the interlock sequences. More specifically, the error is turned on in the middle of either draining or filling the chamber. Only in these two states, the pressure inside of the interlock is either increasing or decreasing. During the two periods, the pressure inside of the interlock can be greater than the maximum and minimum limits. The limit pressure error is turned on when the select key is on and the test pressure is on. The error will stop the draining or filling phase right away and go to the error state. The error state is the same as when the differential pressure error is on. The system cannot proceed with the sequence until the problem is solved.



Figure 10. Error Drain Waveform in gtkwave.



Figure 11. Error Fill Waveform in gtkwave.

5.2 Quartus II Signal Tap Analysis

The signal tap analysis tool allows for real world analysis of the waveform on the hardware implementation of the interlock chamber system. The clock was set to the DE1_SoC board's 50 MHz clock. The LEDs, switches, and the keys were chosen so the different waveforms can be inspected. The test will analyze each state of the arrival and departure sequence of the system. The error state will also be analyzed as well to show a better example of when the errors might occur in a real life interlock system. The LEDs display the outputs of the system in the current state. Refer to Appendix (containing pin assignments) to see what the LEDs represent.

5.2.1 Initial State

In Figure 12, the reset is transitioning from on to off. While the reset is on, all LEDs are off because the chamber is in an empty state. When the reset is switched off, the system will then be able to change to a different state. Depending on the next signal, the system will either go into the arrival or departure sequence.

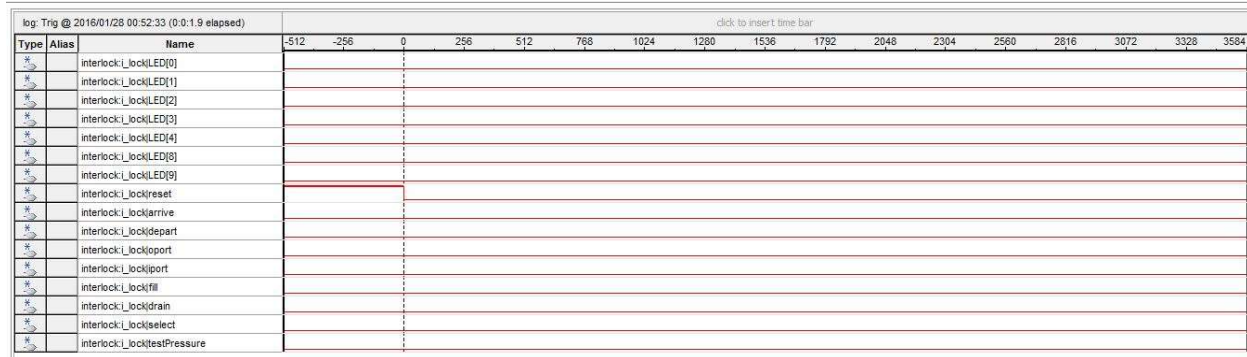


Figure 12. Reset to Initial State in Signal Tap

5.2.2 Boarding State

In Figure 13, the inner port is switched open when the system was in the empty state. This allows the passenger(s) to enter the interlock and go into the bathysphere. Once the passengers are in they will need to close the inner port and send a depart signal to the docking port to initialize the preparation.

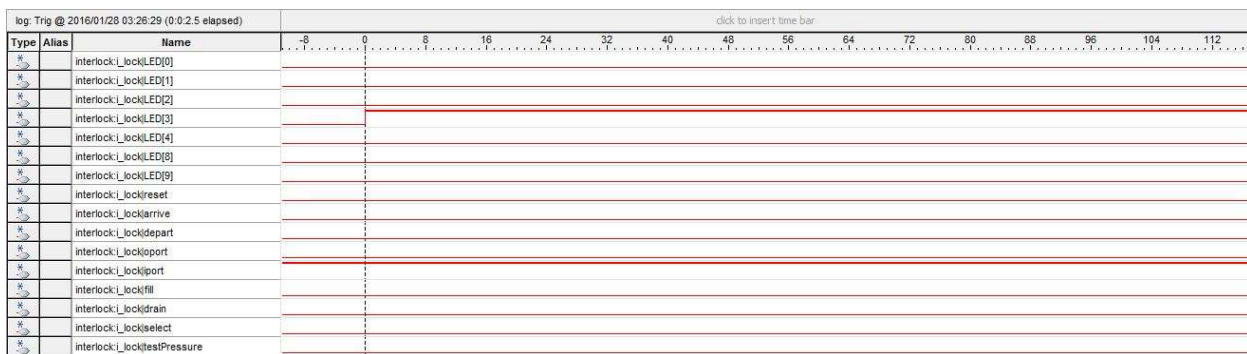


Figure 13. Initial to Boarding State in Signal Tap

5.2.3 Preparation State

In Figure 14 and Figure 15, the system changed to the preparation stage. The sequence of the system depends on whether the arrival or depart signal is currently on. The LED[0] represents that the interlock is going to do an arrival sequence while the LED[1] represents the interlock is going through the depart signal. The LED corresponding to the sequence will be on until the system reaches the empty state again. The system will now wait for 5 minutes to prepare for the bathysphere to either arrive or depart. When the waiting duration is complete, the system will wait for a fill signal.

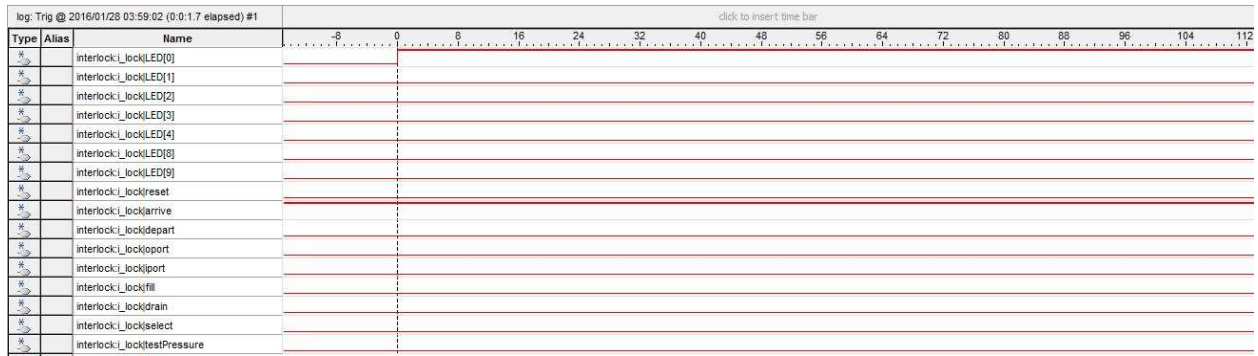


Figure 14. Initial State to Preparation State under Arrival in Signal Tap



Figure 15. Boarding State to Preparation State under Departure in Signal Tap

5.2.4 Filling State

In Figure 16 and Figure 17, the system is pressurizing the interlock up with water until the difference between pressure of the exterior outer port and the interlock is less than .1 atm. The chamber requires a filling period of 7 minutes before pressurization is done. After the interlock is filled, the interlock waits for the outer port to be opened.

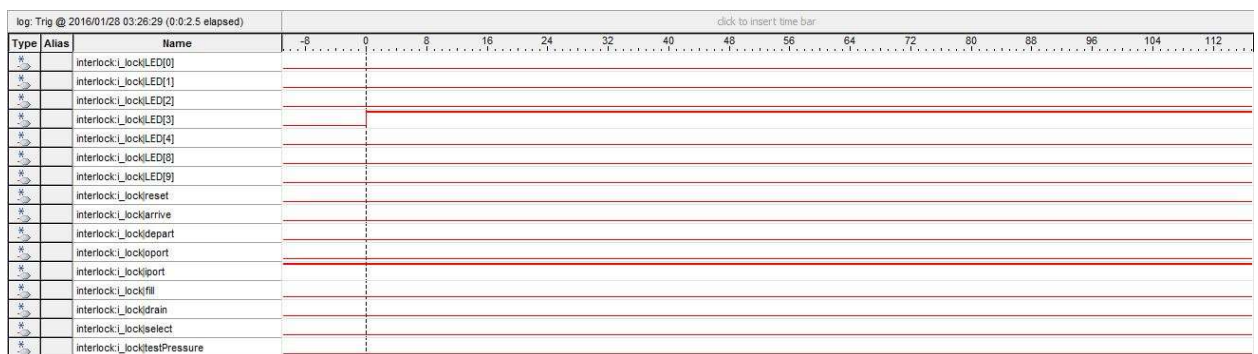


Figure 16. After Waiting to Fill State to Filling State under Arrival in Signal Tap.

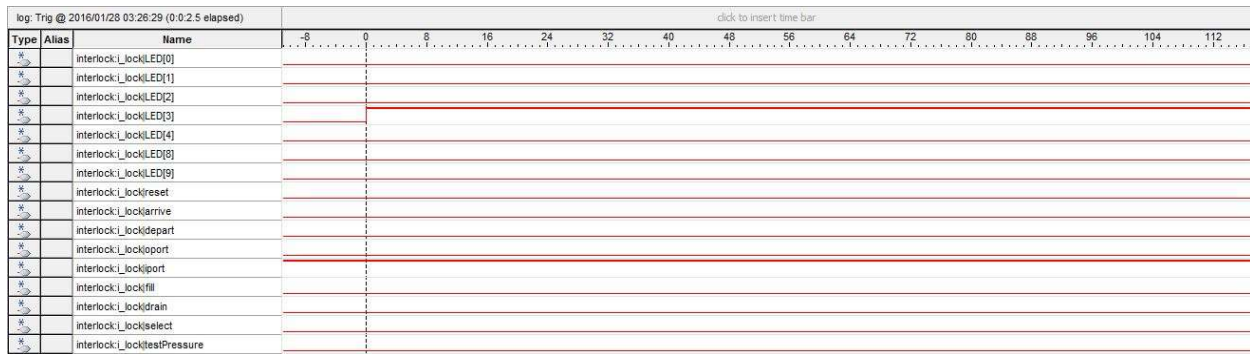


Figure 17. After Waiting to Fill state to Filling State under Departure in Signal Tap.

5.2.5 Outer Port Open State

In Figure 18 and Figure 19, the outer port is opened. LED[2] is now on because that output represents if the outer port is opened or not. If the bathysphere is arriving, the bathysphere would enter into the interlock. If the bathysphere is leaving, the contraption will leave the interlock. Once the bathysphere has gone through the gate. The outer port will close.

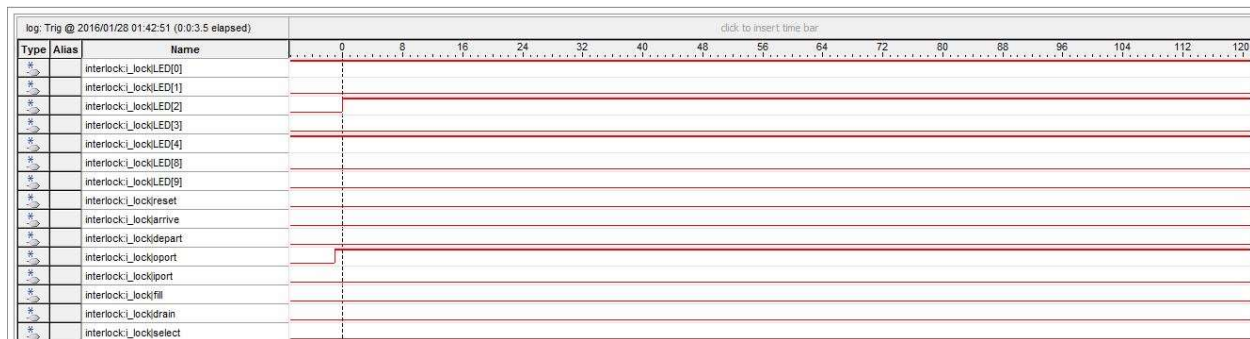


Figure 18. From Filling State to Outer Port Open State under Arrival in Signal Tap.

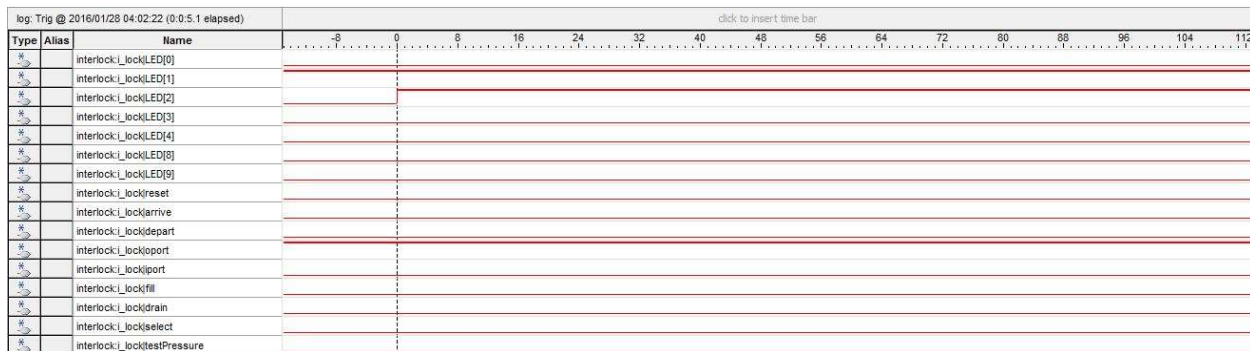


Figure 19. From Filling State to Outer Port Open State under Departure in Signal Tap.

5.2.6 Outer Port Close State

In Figure 20 and Figure 21, the outer port is closed and the LED[2] also turns off. After the outer port is closed, the interlock system will wait for the drain signal.

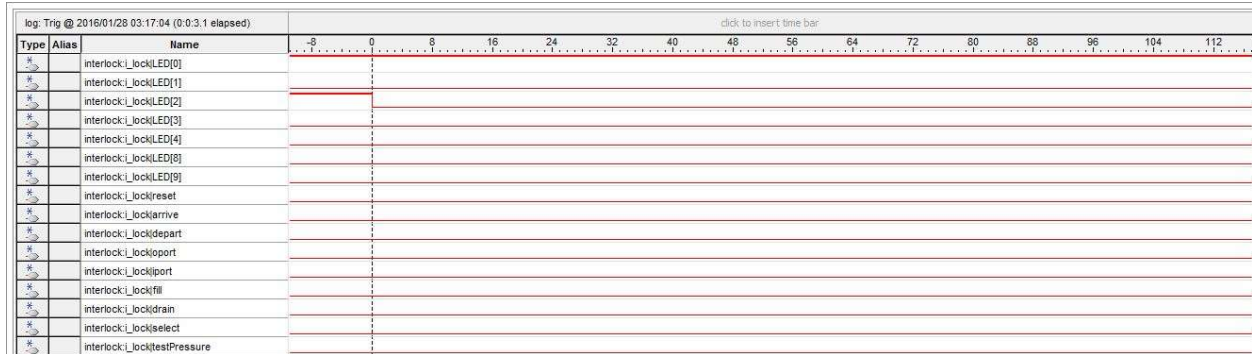


Figure 20. From Outer Port Open State to Outer Port Close State under Arrival in Signal Tap.

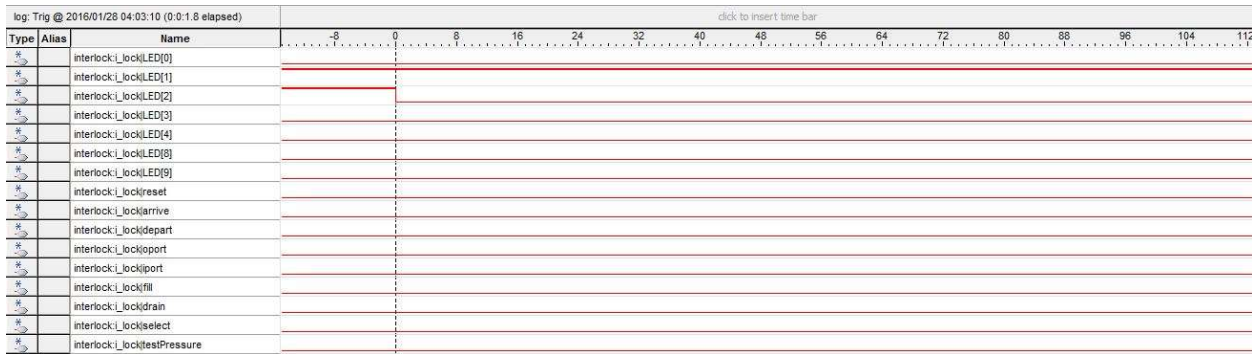


Figure 21. From Outer Port Open State to Outer Port Close State under Departure in Signal Tap.

5.2.7 Draining State

In Figure 22 and Figure 23, the interlock is depressurizing the interlock by draining all of the water out until the difference in pressure between the exterior of the inner port and the interlock is less than .1 atm. The chamber requires a draining period of 8 minutes before depressurization is done. After the chamber is drained, the system will wait for the inner port to be opened.

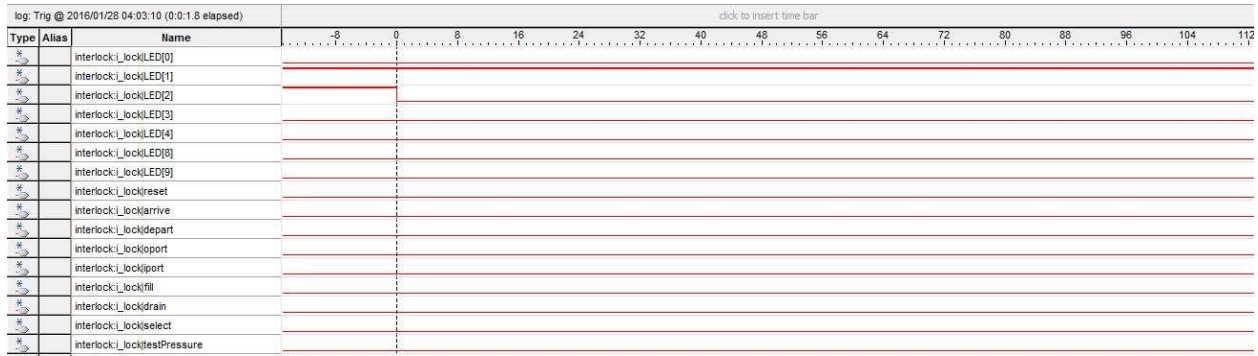


Figure 22. From Waiting to Drain State to Draining State under Arrival in Signal Tap.



Figure 23. From Waiting to Drain State to Draining State under Departure in Signal Tap.

5.2.8 Inner Port Open State

In Figure 24 and Figure 25, the inner port is opened. LED[3] lights up corresponding to the inner port's open position. If the bathysphere is in the interlock, then the passenger(s) will enter the underwater station. Once all passenger(s) have left the interlock, the system will wait for the inner port to close.

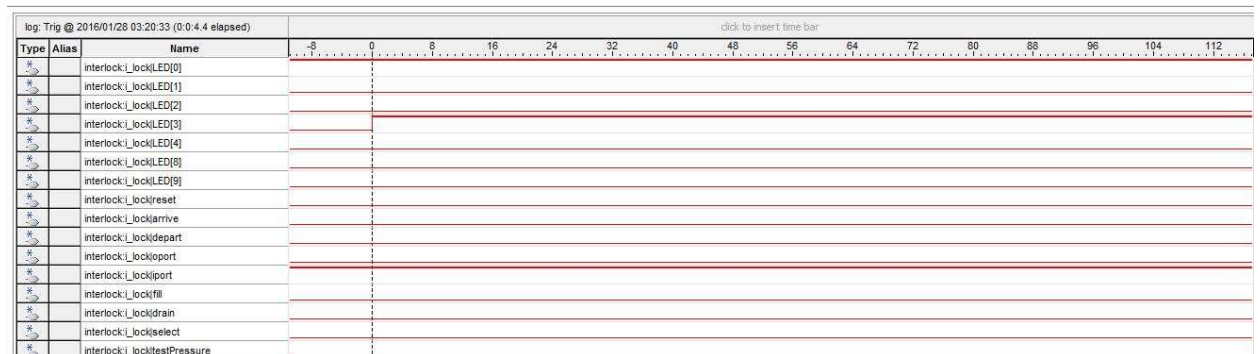


Figure 24. From Draining State to Inner Port Open State under Arrival in Signal Tap.

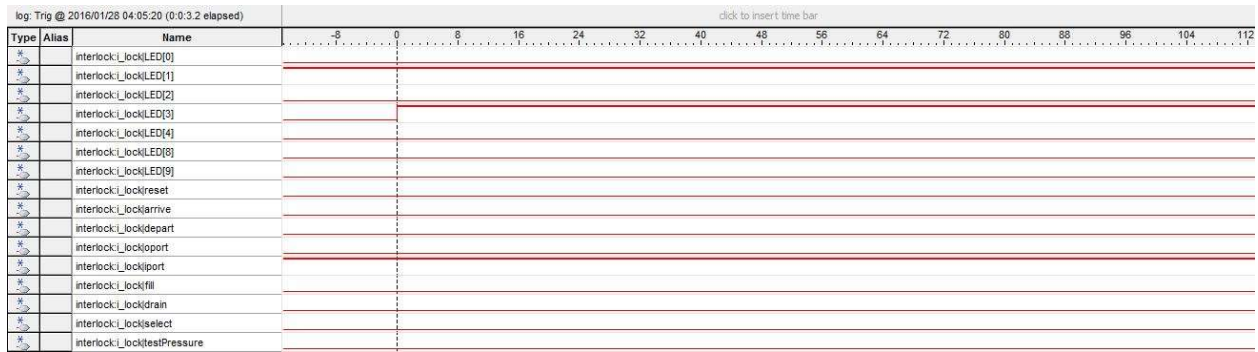


Figure 25. From Draining State to Inner Port Open State under Departure in Signal Tap.

5.2.9 Inner Port Close State

In Figure 26 and Figure 27, the inner port is closed. LED[3] is off now because the inner port is closed. Since the arrival and departure sequence is done, the LED[0] and LED[1] are both off as well. Now the system is back into the initial state and can go to either the arrival or departure sequence.

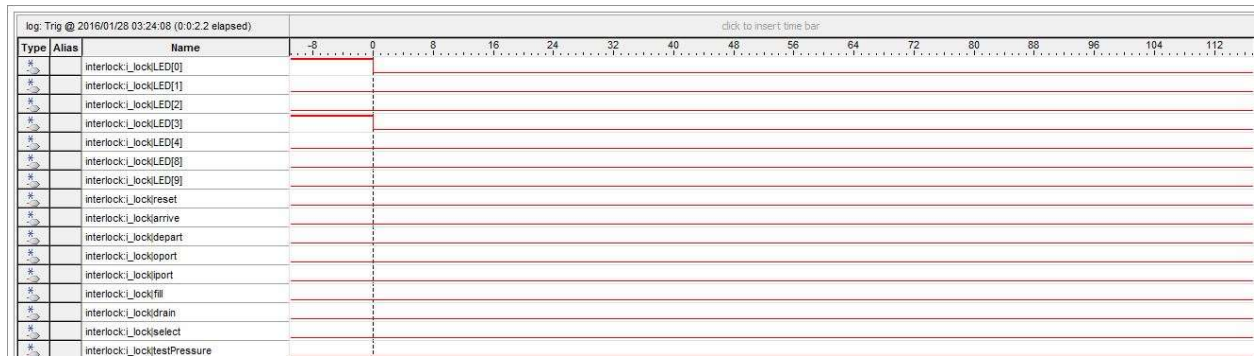


Figure 26. From Inner Port Open State to Inner Port Close under Arrival in Signal Tap.

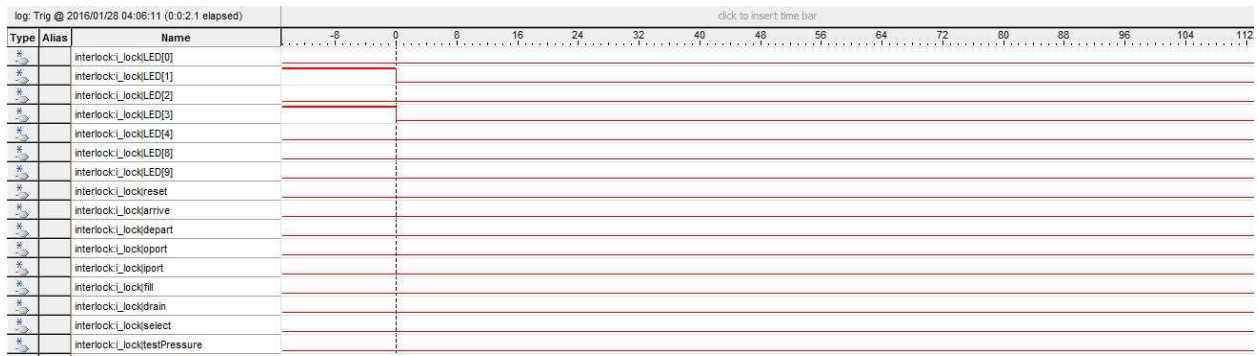


Figure 27. From Inner Port Open State to Inner Port Close under Departure in Signal Tap.

5.2.10 Differential Pressure Error Trigger to Error State

In Figure 28 and Figure 29, the differential pressure error is on. LED[8] corresponds to the differential pressure error. When the error signal is on, the rest of the LEDs are off and the system is in an error state. While in the error state, nothing can be done and the users will have to solve the problem before using the interlock again. This only occurs when the differential pressure is not under .1 atm and the system is about to open either the inner or outer port. With this error, the user would have to fix the differential pressure so there will not be any problems when the ports are opened.

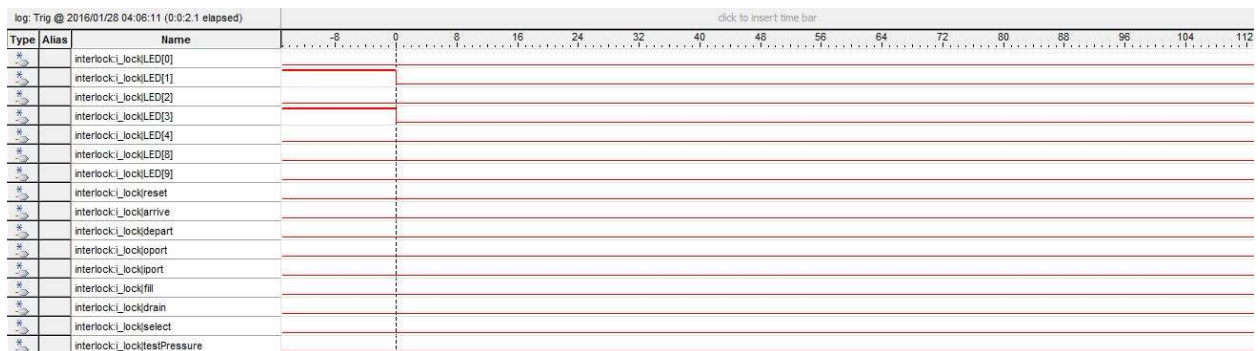


Figure 28. Differential Pressure Error and Inner Port Open State to Error State in Signal Tap.

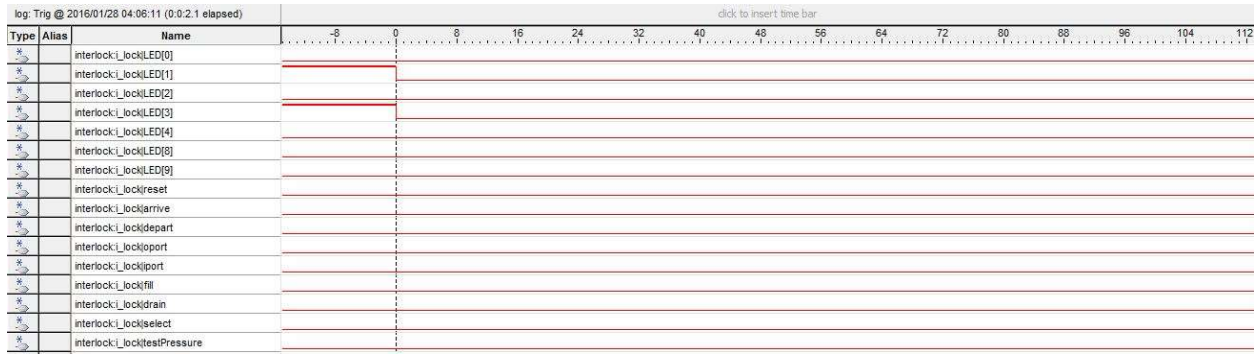


Figure 29. Differential Pressure Error and Outer Port Open State to Error State in Signal Tap.

5.2.11 Limit Pressure Error Trigger to Error State

In Figure 30 and Figure 31, the limit pressure error is on. LED[9] corresponds to the limit pressure error. When the error signal is one, the rest of the LEDs are off and the system is in an error state. The error state is the same as when the differential pressure error is on. The error occurs when the pressure inside of the interlock is not within the bound of 16,000 psi and 13 psi and the system is trying to pressurize or depressurize the chamber even more. With this error, the user would have to check the sensors or check the pressures outside the interlock to see if specs of the interlock is too limited.

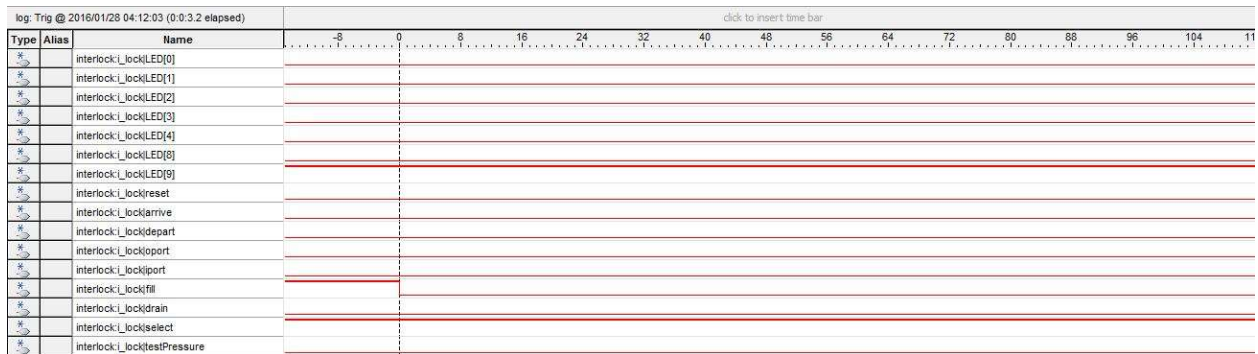


Figure 30. Limit Pressure Error and Filling State to Error State in Signal Tap.

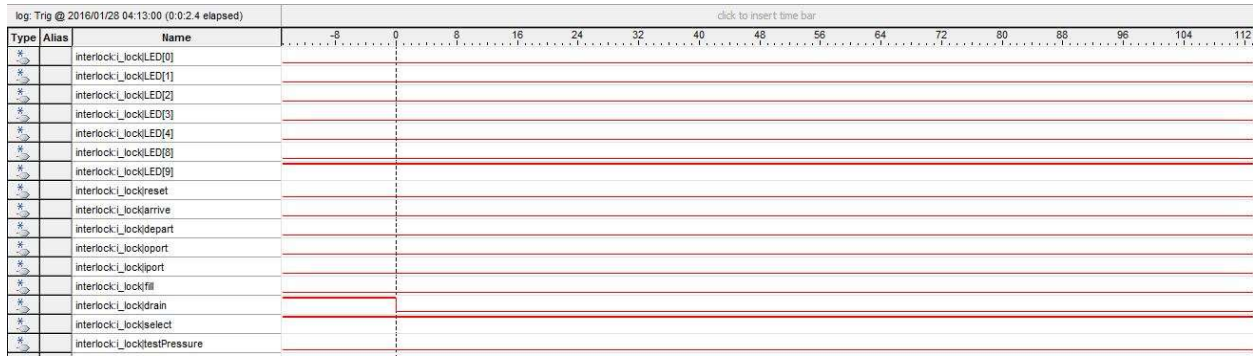


Figure 31. Limit Pressure Error and Draining State to Error State in Signal Tap.

5.3 Failure Mode and Analysis

The bathysphere as a whole is examined for failure mode analysis. The bathysphere has the signals for arrival, departure, filling chamber, draining the chamber, opening and closing the outer port and opening and closing the inner port for the regular functioning of the bathysphere. In addition to these signals, there is the difference and limit error signal which indicate the incompatible surrounding pressure conditions for the bathysphere to conduct its regular operations. In addition to these signals, the bathysphere takes in a reset and clock signals that control the system. The outputs of these signals and the results of their failures are mentioned below under each signal.

5.3.1 Arrive Signal

At SA0 for the outputs, the corresponding LEDs, LED[0] (which indicates the arrival stage) will not turn on. At SA1 for the outputs, the corresponding LEDs, LED[0] will continue to stay on and the LED[4], which indicates the Preparation stage will blink for 5 seconds and display the message of “Wait Fill” on the HEX Display indefinitely.

5.3.2 Departure Signal

At SA0 for the outputs, the corresponding LEDs, LED[1] (which indicates the arrival stage) will fail to turn on. At SA1 for the outputs, the corresponding LEDs, LED[1] will continue to stay on and the LED[4], which indicates the Preparation stage will blink for 5 seconds and display the message of “Wait Fill” on the HEX Display indefinitely.

5.3.3 Fill Signal

At SA0 for the outputs, the corresponding LEDs, LED[5] (which indicates the filling stage) will fail to turn on. At SA1 for the outputs, the corresponding LEDs, LED[5] will blink for about 7.5 seconds (assuming the counter is functional) and will display the “oPort Open” message on the HEX display- indicating the state is waiting for the oport to open. However, if the oport is opened when fill signal is SA1, there is no effect and the system and the HEX Display message is seen forever. Depending on what stage the bathysphere is in (arrival or departure) LED[0] or LED[1] will remain on correspondingly for both SA0 and SA1.

5.3.4 Drain Signal

At SA0 for the outputs, the corresponding LEDs, LED[6] (which indicates the filling stage) will fail to turn on. At SA1 for the outputs, the corresponding LEDs, LED[6] will blink for about 8.5 seconds (assuming the counter is functional) and will display the “iPort Open” message on the HEX display- indicating the state is waiting for the iport to open. However, if the iport is opened when fill signal is SA1, there is no effect and the system and the HEX Display message is seen forever. Depending on what stage the bathysphere is in (arrival or departure) LED[0] or LED[1] will remain on correspondingly for both SA0 and SA1.

5.3.5 Open Outer Port Signal

At SA0 for the outputs, the corresponding HEX Display ”oPort Open” (which indicates the waiting stage for the oport to open) will be displayed on the board forever. This is because the oPort is opened after the draining stage. At SA1 the HEX Display message “oPort Close”, will continue to stay on and the remaining stages will not execute. Depending on what stage the bathysphere is in (arrival or departure) LED[0] or LED[1] will remain on correspondingly for both SA0 and SA1.

5.3.6 Open Inner Port Signal

At SA0 for the outputs, the corresponding HEX Display ”iPort Open” (which indicates the waiting stage for the iport to open) will be displayed on the board forever. This is because the oPort is opened after the draining stage. At SA1 the HEX Display message “iPort Close”, will continue to stay on and the remaining stages will not execute. Depending on what stage the bathysphere is in (arrival or departure) LED[0] or LED[1] will remain on correspondingly for both SA0 and SA1.

The signal for iport is also seen in the beginning of the depart stage when passengers are allowed to board the bathysphere. In this case, at the signal SA0 the HEX Display would be stuck at “BOARD” and the LED[1], indicating the departure stage will stay on. At SA1, the same message will be stuck on the HEX Display. The bathysphere will be stuck on the “boarding” state of the departure.

5.3.7 Difference Error

At SA0, the system has no difference error. This means that the LEDs indicating a difference error in the system (LED[7]) will be off and the bathysphere will continue its regular stages of execution in both the arrival and departure stage. At SA1, the LED[7] will remain on and an error message “ERROR” will be displayed on the HEX Display after the Waiting for IPort or Waiting for Oport stage (depending on what stage it got stuck) and will go into an ‘error state’ where it will stay indefinitely.

5.3.8 Limit Error Signal

At SA0, the system has no limit error. This means that the LEDs indicating a difference error in the system (LED[8]) will be off and the bathysphere will continue its regular stages of execution in both the arrival and departure stage. At SA1, the LED[8] will remain on and an error message “ERROR” will be displayed on the HEX Display after the Filling or Draining stage (depending on what stage it got stuck) and will go into an ‘error state’ where it will stay indefinitely.

5.3.9 Clock and Reset Signal

The counter for our system, is used for measuring time intervals by matching the bit patterns (as mentioned in Section 3 above). The counter is 27 bits long and increments by 1 every clock cycle. For the purposes of displaying the LED, we have used a slower clock so changes in state, indicated by the LEDs can be seen by the human eye. If the signal is SA0 or SA1 mode, this means the clock will not be oscillating from posedge to negedge and the signal will be stuck on 1 or 0. This means the counter will be stuck on the previous state since the components will not see the next rising edge to change its values. If the reset signal is SA0, the bathysphere will never be able to reach its initial state if it reached the error state. At SA1, the bathysphere will keep on resetting and not move forward to any of the other states in the bathysphere. Analysis of Errors

5.4 Analysis of any errors

The first error we encountered was the inconsistency of the signals from the switch. Since all of the available keys on the board were used, the switches were the only easily accessible inputs available. The switch would oscillate between on and off when it is switched on because of the transient state of the electricity at the moment of the flick of the switch. The time to settling time much longer than a key, which already has a debounce. Without a stable signal from the inputs, the sequence that the interlock system must go through was erratic and not what was expected. Ultimately, the switch was still used, but the clock was slowed down. The slower clock allowed the switch response to be more stable with given extra time.

Another error occurred while simulating the interlock system. The system was working fine on the board, but the gtkwave waveforms were being stuck in the initial state of the system. The initial state occurs right when the reset signal is switched off. The problem was that the always block in the code controlling the next state was only looking at the changes of the present state. When there were no changes to the present state, the next state were never updated. This causes the next state to stay the same and so present state will never change. The solution to the problem was to have the always block to look at all of the inputs that next state depended on to update its value. After making the change in the always conditions, the simulations ran smoothly.

The last major error that occurred was using the HEX displays. Hex displays were ideal to show the state that the system is currently on. The displays were troublesome because there were 42 lights to connect to in our code. Knowing the orientation and how the bits aligned to each of the LEDs of the displays were confusing. The problem was that these LEDs were active low so all of the patterns on the displays were a mess and made no sense. The solution was to place an inverter on each of the 7-bit values to get the correct pattern for each of the states.

5.5 Analysis of possible errors

In this lab, there is a high chance of encountering errors because of all all of the outputs and inputs of the interlock system. One likely error is forgetting to update the next state in any of the case statements will instantly produce the wrong output. The state machine would most likely stay in that same state forever until the reset is hit. This will cause a freeze in the interlock system. Another error is messing up one of the pin assignments for one of the inputs and outputs. If one of the switches were not connected correctly, there are many possible errors that would have to be checked. Wrong pin assignments can be common since the pins for the DE1-Board do not have a logical order. Another error can occur during the signal tap part of the lab. The clock used in the

tool can change the width of the waveforms. If a project contains multiple modules using different clocks, the chosen clock might be incorrect and produce an output that is not close to the expected waveforms.

6. Summary and Conclusion

The purpose of the lab was to understand how to design a bathysphere system using the concept of mealy and moore state machine diagrams. The bathysphere system was designed using spec guidelines and state diagrams to design the different states taken by the bathysphere. We implemented the functionality of the interlock chamber (part of the bathysphere) during the arrival and departure modes of the bathysphere. The interlock contains 2 bi-directional ports: the inner and the outer port. We provided the following functionality to the 2 ports: (1) filling and (2) draining the interlock chamber. We also implemented the functionality of detecting errors in pressure (limit & differential) while executing the draining and filling stages. We achieved the resulting system using various control signals for each of these functions. To top it off the C program we wrote furthered our knowledge of pointers in C and helped understand their use cases.

In conclusion, overall, the lab covered many things that have previously been covered and new things that are just being introduced. The most important thing we learned was how to truly think about a project in terms of a good design process for making it accessible to the end-user. In this case, we went through multiple state-diagrams for modeling the different states until we found the user-friendliest one for the bathysphere. It also gave us an opportunity to think of ways we could improve our code to make it cleaner and user easier to use. After our first version of the Verilog code, we introduced an interactive HEX Display and LED chunk of code in an always block. This display showed the user the current steps the user should take to control the bathysphere (for instance, if the outer port was to be opened, the HEX Display would indicate that). The design reviews were also useful for helping us to brainstorm various ways of implementing the logic of arrival and departure of the bathysphere. The spec of this lab did a great job of giving us just enough information for implementing the bathysphere as well as giving us space for being creative for coming up with implementation details. We took advantage of this and made use of more components of the board which weren't necessarily specified to use (such as the HEX Display and the different Switches and Keys). Learning the process of design creation is an important aspect of engineering. You can be a great engineer, but if you don't give what the user what they want, you can't make a great product. Learning new tools will be essential in the industry since not all places will have familiar tools or some places will have a superior tool. The lab also demonstrated that even if simulations are producing the right output, the actual results on the board might differ. Using the Signal Tap tool, we determined that the key produces a better input signal than the switch.

7. Authors and Contributions

| NAME | CONTRIBUTIONS |
|------------------|---|
| Denny Ly | Interlock Implementation. Analysis, Differential Error Module, Lab Report |
| Minhhue H. Khuu | Interlock Implementation, Testing, Input Handler Module, Lab Report |
| Ruchira Kulkarni | Interlock Implementation, C Program, Limit Error Module, Lab Report |

8. Appendix

8.1 Interlock System

Table 10. Signal to FPGA Pin Pairing.

| Signal Name | FPGA Pin | Description | I/O Standard | Interlock Name |
|-------------|----------|--------------------------|--------------|----------------------|
| CLOCK_50 | PIN_AF14 | 50 MHz clock input | 3.3 V | clock |
| LEDR[0] | PIN_V16 | LED[0] | 3.3 V | arriving |
| LEDR[1] | PIN_W16 | LED[1] | 3.3 V | departing |
| LEDR[2] | PIN_V17 | LED[2] | 3.3 V | outer port open |
| LEDR[3] | PIN_V18 | LED[3] | 3.3 V | inner port open |
| LEDR[4] | PIN_W17 | LED[4] | 3.3 V | waiting |
| LEDR[8] | PIN_W21 | LED[8] | 3.3 V | diff. pressure error |
| LEDR[9] | PIN_Y21 | LED[9] | 3.3 V | limit pressure error |
| HEX0[0] | PIN_AE26 | Seven Segment Digit 0[0] | 3.3 V | HEX0 |
| HEX0[1] | PIN_AE27 | Seven Segment Digit 0[1] | 3.3 V | HEX0 |
| HEX0[2] | PIN_AE28 | Seven Segment Digit 0[2] | 3.3 V | HEX0 |
| HEX0[3] | PIN_AG27 | Seven Segment Digit 0[3] | 3.3 V | HEX0 |
| HEX0[4] | PIN_AF28 | Seven Segment Digit 0[4] | 3.3 V | HEX0 |
| HEX0[5] | PIN_AG28 | Seven Segment Digit 0[5] | 3.3 V | HEX0 |
| HEX0[6] | PIN_AH28 | Seven Segment Digit 0[6] | 3.3 V | HEX0 |
| HEX1[0] | PIN_AJ29 | Seven Segment Digit 1[0] | 3.3 V | HEX1 |
| HEX1[1] | PIN_AH29 | Seven Segment Digit 1[1] | 3.3 V | HEX1 |
| HEX1[2] | PIN_AH30 | Seven Segment Digit 1[2] | 3.3 V | HEX1 |
| HEX1[3] | PIN_AG30 | Seven Segment Digit 1[3] | 3.3 V | HEX1 |
| HEX1[4] | PIN_AF29 | Seven Segment Digit 1[4] | 3.3 V | HEX1 |
| HEX1[5] | PIN_AF30 | Seven Segment Digit 1[5] | 3.3 V | HEX1 |
| HEX1[6] | PIN_AD27 | Seven Segment Digit 1[6] | 3.3 V | HEX1 |
| HEX2[0] | PIN_AB23 | Seven Segment Digit 2[0] | 3.3 V | HEX2 |
| HEX2[1] | PIN_AE29 | Seven Segment Digit 2[1] | 3.3 V | HEX2 |
| HEX2[2] | PIN_AD29 | Seven Segment Digit 2[2] | 3.3 V | HEX2 |
| HEX2[3] | PIN_AC28 | Seven Segment Digit 2[3] | 3.3 V | HEX2 |

| | | | | |
|----------------|----------|--------------------------|-------|---------------|
| HEX2[4] | PIN_AD30 | Seven Segment Digit 2[4] | 3.3 V | HEX2 |
| HEX2[5] | PIN_AC29 | Seven Segment Digit 2[5] | 3.3 V | HEX2 |
| HEX2[6] | PIN_AC30 | Seven Segment Digit 2[6] | 3.3 V | HEX2 |
| HEX3[0] | PIN_AD26 | Seven Segment Digit 3[0] | 3.3 V | HEX3 |
| HEX3[1] | PIN_AD27 | Seven Segment Digit 3[1] | 3.3 V | HEX3 |
| HEX3[2] | PIN_AD25 | Seven Segment Digit 3[2] | 3.3 V | HEX3 |
| HEX3[3] | PIN_AC25 | Seven Segment Digit 3[3] | 3.3 V | HEX3 |
| HEX3[4] | PIN_AB28 | Seven Segment Digit 3[4] | 3.3 V | HEX3 |
| HEX3[5] | PIN_AB25 | Seven Segment Digit 3[5] | 3.3 V | HEX3 |
| HEX3[6] | PIN_AB22 | Seven Segment Digit 3[6] | 3.3 V | HEX3 |
| HEX4[0] | PIN_AA24 | Seven Segment Digit 4[0] | 3.3 V | HEX4 |
| HEX4[1] | PIN_Y23 | Seven Segment Digit 4[1] | 3.3 V | HEX4 |
| HEX4[2] | PIN_Y24 | Seven Segment Digit 4[2] | 3.3 V | HEX4 |
| HEX4[3] | PIN_W22 | Seven Segment Digit 4[3] | 3.3 V | HEX4 |
| HEX4[4] | PIN_W24 | Seven Segment Digit 4[4] | 3.3 V | HEX4 |
| HEX4[5] | PIN_V23 | Seven Segment Digit 4[5] | 3.3 V | HEX4 |
| HEX4[6] | PIN_W25 | Seven Segment Digit 4[6] | 3.3 V | HEX4 |
| HEX5[0] | PIN_V25 | Seven Segment Digit 5[0] | 3.3 V | HEX5 |
| HEX5[1] | PIN_AA28 | Seven Segment Digit 5[1] | 3.3 V | HEX5 |
| HEX5[2] | PIN_Y27 | Seven Segment Digit 5[2] | 3.3 V | HEX5 |
| HEX5[3] | PIN_AB27 | Seven Segment Digit 5[3] | 3.3 V | HEX5 |
| HEX5[4] | PIN_AB26 | Seven Segment Digit 5[4] | 3.3 V | HEX5 |
| HEX5[5] | PIN_AA26 | Seven Segment Digit 5[5] | 3.3 V | HEX5 |
| HEX5[6] | PIN_AA25 | Seven Segment Digit 5[6] | 3.3 V | HEX5 |
| SW[0] | PIN_AB12 | SW[0] | 3.3 V | arrive signal |
| SW[1] | PIN_AC12 | SW[1] | 3.3 V | depart signal |
| SW[2] | PIN_AF9 | SW[2] | 3.3 V | outer port |
| SW[3] | PIN_AF10 | SW[3] | 3.3 V | inner port |
| SW[4] | PIN_AD11 | SW[4] | 3.3 V | error select |
| KEY[0] | PIN_AA14 | KEY[0] | 3.3 V | reset |
| KEY[1] | PIN_AA15 | KEY[1] | 3.3 V | fill |

| | | | | |
|--------|---------|--------|-------|---------------|
| KEY[2] | PIN_W15 | KEY[2] | 3.3 V | drain |
| KEY[3] | PIN_Y16 | KEY[3] | 3.3 V | error trigger |

8.2 C Program (Pointers)

8.2.1 Design Requirements

8.2.1.1 Abstract

In this part of the lab, we were introduced to the concept of pointers in the C programming language. This process was divided into two sections:

- (1) Part 1: Getting to know Pointers
- (2) Part 2: Working with Pointer Variables

As a result of writing this C program, we found that the dereferencing integer pointer for multiplication is the same thing as multiplying by integers.

8.2.1.2 Introduction

C, unlike higher level programming languages is used for writing microprocessor code. For this reason, it is important that we understand the different ways we can access the address memory of variables in the program. In C, addresses are accessed with the help of pointers. We can change the value of a variable by making it point to a different address in memory with a pointer.

8.2.1.3 Inputs to Pointer Program

There were no inputs to the pointers.c program.

8.2.1.4 Outputs

The outputs produced by the pointers.c (part 2) are as follows:

1. The result of the multiplication by pointer dereferencing the int pointers, which is 5.
2. The result of the integer multiplication, which is 5.

8.2.1.5 Major Functions

This program contains one major function:

```
int main (int argc, char** argv)
```

8.2.2 Design Specifications

8.2.2.1 Abstract

Our task comprised of using the Command-Line Tool Interface to run a C program file as well as compile and run the C program for calculating a simple mathematical equation using the properties of pointers. The equation we were to compute was as follows:

$$\text{result} = ((A - B) * (C - D)) / E$$

where A, B, C, D and E are int variable were given to us.

The result of multiplication by dereferencing integer pointer for multiplication is the same thing as multiplying by integers.

8.2.2.2 Introduction

In part 1 of the C assignment, we created different “int” and “float” pointers as specified in the assignment write-up. We did this by declaring various int and float variables and then saving the address of these variables by using the ‘&’ operator which indicates “address of value” into int pointer (int *) and float pointers (float *) respectively.

In part 2, we implemented the concepts we learned in part 1 and computed an integer multiplication by using the dereferencing properties of pointers.

8.2.2.3 Inputs to Program

There are no required inputs in the C program. This is because there is only one formula (that was shown above in Abstract) which needed to be implemented. Moreover, the integer variables A, B, C, D and E were already provided to us as well. As the program was meant to be executed for a very specific formula for a set of values, no arguments/inputs were necessary.

The values given to the int variables are as follows:

A = 22

B = 17

C = 6

D = 4

E = 9

8.2.2.4 Outputs

The program prints the result of the multiplication by pointer dereferencing the int pointers, which is 5 to the command line. It also prints the result of the integer multiplication, which is 5. Both these results were used to check the correctness of the program.

8.2.2.5 Major Functions

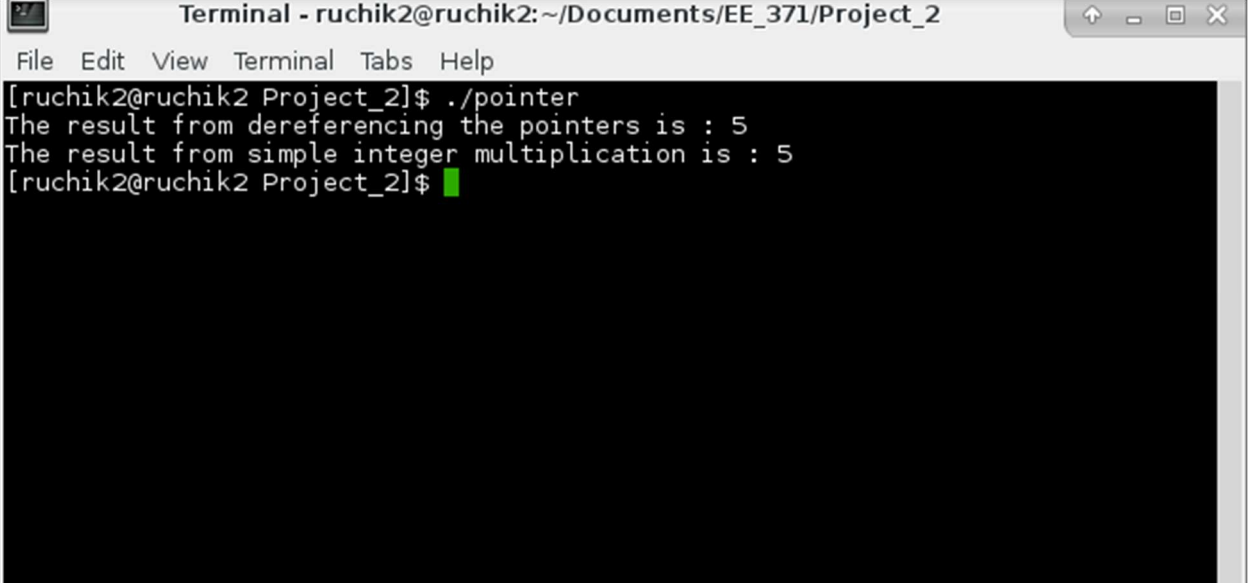
This program contains four major functions:

int main (int argc, char argv) :** This program only consists of a main method, which is the start of execution in a C program. The main method takes in an int argc, which represents the number of arguments in the command line interface and char ** argv which is a double character pointer which represents the strings in the command line.

The main function declares the 5 different ints and then creates their respective int pointers. Then, it dereferences these pointers and uses these values in the equation specified above in the abstract.

8.2.3 Results and Analysis

The results of the C program behaves as expected. When the user inputs valid arguments that follows the design specification, the program outputs a correct and corresponding result (see Figure 32).

A screenshot of a terminal window titled "Terminal - ruchik2@ruchik2:~/Documents/EE_371/Project_2". The terminal has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The command prompt shows the user is in the "Project_2" directory. The user has executed the command `./pointer`, which has produced two lines of output: "The result from dereferencing the pointers is : 5" and "The result from simple integer multiplication is : 5". The prompt is now ready for the next command.

```
Terminal - ruchik2@ruchik2:~/Documents/EE_371/Project_2
File Edit View Terminal Tabs Help
[ruchik2@ruchik2 Project_2]$ ./pointer
The result from dereferencing the pointers is : 5
The result from simple integer multiplication is : 5
[ruchik2@ruchik2 Project_2]$
```

Figure 32. Demonstration of the C pointer program