

LAB 2: AIRPLANE INDICATOR LIGHTS CONTROLLER

---

## **ECE327: PROJECT 2**

---

October 20, 2019

Bradley Selee  
Clemson University  
Department of Electrical and Computer Engineering  
[bselee@g.clemson.edu](mailto:bselee@g.clemson.edu)

## **Abstract**

Project 2 introduces a very important concept of Finite-state-machine's (FSM). The goal was to implement a state machine that keeps track of the current signal status. Within this lab, a Moore state-machine was designed to control two LED signals on an airplane, the "seat-belt" light and "no-electronics" light. The state of these lights were determined by the altitude of the plane, therefore, states were generated determining if the airplane has reached or maintained a specific altitude. This was implemented in a VHDL, tested through a test bench, and wrapped to a board file. After the VHDL implementation was created, the logic was mapped to OpenCL. FPGA display: the board begins with both LEDs high, when the reset is high (logic low) and the ivalid is high. When the proper transition sequence is entered and the clock pulses high, to detect the rising edge, the FSM will move to the next state and the LEDs will adjust correspondingly. Whenever the plane is off the ground, the "No Electronics" LED is turned off. After 5 consecutive, smooth-cycles, above 25 kilometers both LEDs will be off. When tested with OpenCL, the same output was given through the terminal [2].

# 1 Introduction

Lab 2 gives a more complex introduction to FSMs. The ultimate goal was to control two LED indicators of an airplane. This was accomplished by creating a Moore FSM in VHDL, testing it through the test bench simulation and FPGA board I/O, and mapping the design to OpenCL. The DE1-SoC, the lab FPGA, consists of 10 switches for input, and 10 LEDs and 6 7-segment displays for output; this lab uses these I/O devices to test the board implementation of the FSM. The FSM was simulated through a test bench, where the output waveform was monitored.

For this lab, a more complex FSM was created. Part I consisted of the FSM state-diagram design, VHDL implementation, and simulation and board testing. Part II required the design to be mapped to OpenCL. This involved creating the OpenCL VHDL implementation, uploading the necessary files to our SD card, and simulating the design through the terminal.

In order for the FPGA to be programmed, the board had to be set-up properly in Quartus, along with the board file being correct. Then, because the DE1-series has hardwired connections between the chip and its I/O, the pin assignment file *DE1.qsf* was included in the project. Finally, once all parts were completed and tested for correctness, the final board file was compiled and tested on the board.

## 2 VHDL Finite-State-Machine Design

In part I, a Moore FSM was designed for the following problem specified in the textbook:

**14.25 Airplane indicator lights,** I. Draw the state diagram and table for an FSM that controls the seat-belt and no-electronics signs for a commercial airliner. The state machine has three inputs: alt10k, alt25k, and smooth. Whenever the airplane passes 10 000 (25 000) feet moving in either direction, alt10k (alt25k) will pulse high for one cycle. If the plane is not climbing, descending, or experiencing turbulence, the smooth signal will be set to high. The state machine should set the no electronics signal to high when the plane is below 10 000 feet, and low otherwise. The seat belt signal should be low only when the plane is above 25 000 feet and smooth has been asserted for at least five cycles. Assume that the plane is initially on the ground. [1]

### 2.1 Architecture Design and Logic

First, the state-machine was defined in a basic VHDL file, then the logic was written. The first step was to draw the state diagram

In order to create the above FSM in VHDL, every possible state transition was defined by case-when and if statements, which was all wrapped inside a process statement. The state



## 2.2 Test Bench and Simulation

Once the architecture was created, the design needed to be simulated. This was done through test bench. A test bench allows the creation of test inputs and, in response, Quartus simulates the architecture by displaying the resulting input and output waveforms. For part 1, the test bench includes inputs for rst, ivalid, clk and alt. After simulating, the waveform shows the inputs rst, ivalid, clk, alt and the output lights at specific time intervals.

The following images show various tests. Test 1, figure 2.3, demonstrates the necessity of the ivalid input, Test 2, also figure 2.3, demonstrates the entire flight pattern, showing all of the outputs. Test 3, figure 2.4, demonstrates interrupting signals, such as non-consecutive smooths above 25 kilometers, and the occurrence of a rest

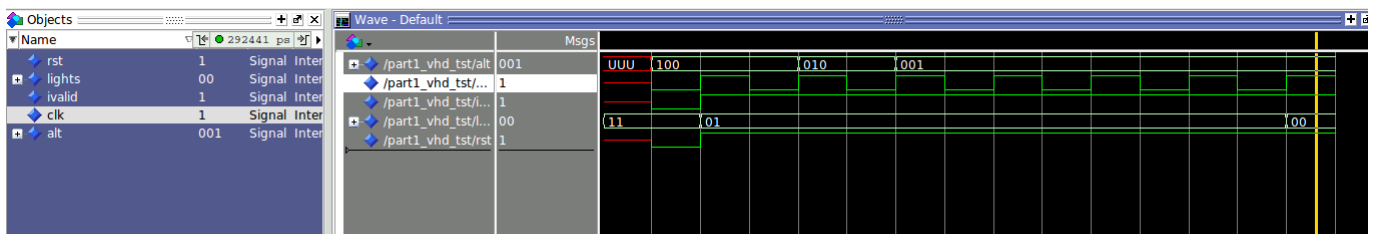


Figure 2.3: Part 1 Test Bench Test 1 & 2

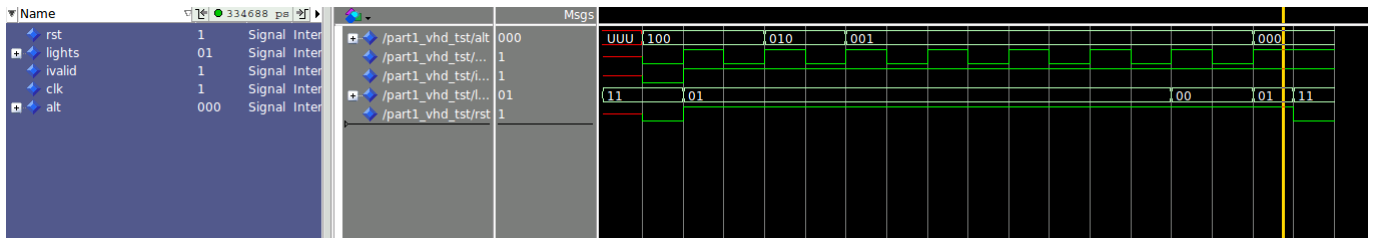


Figure 2.4: Part 1 Test Bench Test 3

## 2.3 Hardware Implementation

Once the FSM was verified through a simulation, the circuit was tested on the FPGA. To do this, a wrapper, board file, was created and Quartus was setup to program the board. Finally, the board file was compiled and the board was tested using the built in I/O. Most inputs were mapped to the switches and the outputs were mapped to LEDs; the clock input was mapped to the built in key because it includes a debouncer.

## 3 OpenCL

OpenCL is a framework used to write programs that execute across various types of hardware: CPUs, FPGAs, GPUs, and DPUs. For this project, OpenCL will be used with the lab FPGAs and our high-speed SD readers.

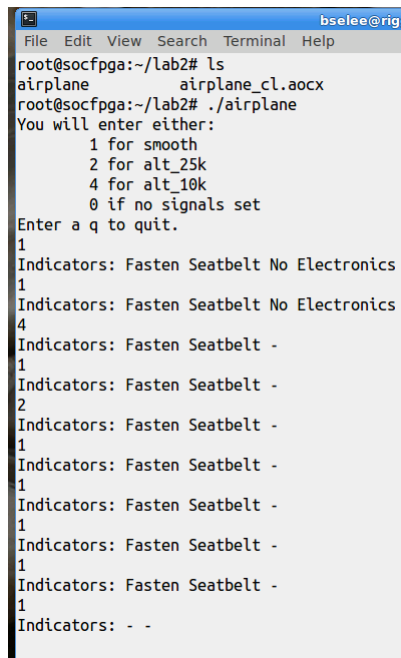
### 3.1 Port Mapping

Part 2 expands the scope of VHDL's capabilities. In this part, the Moore FSM was mapped to the given OpenCL project. The 3-bit input corresponding to (*alt<sub>10k</sub>*, *alt<sub>25k</sub>*, *smooth*) was mapped to *datain*(2 downto 0) respectively. The 2-bit output corresponding to (*seat belts*, *no electronics*) was mapped to *dataout*(1 downto 0)

### 3.2 Testing and Output

Once the FSM was mapped to the given OpenCL project, the SD card was placed into the FPGA and a serial communication was made. Once logging in to the FPGA, the OpenCL airplane executable was run to test the FSM.

This was tested through a simple command-line program. Specific command-line arguments were entered and the program responded with outputs based on the designed FSM. Below are the correct outputs given by OpenCL



```
bselee@rigg
File Edit View Search Terminal Help
root@socfpga:~/lab2# ls
airplane      airplane_cl.aocx
root@socfpga:~/lab2# ./airplane
You will enter either:
    1 for smooth
    2 for alt_25k
    4 for alt_10k
    0 if no signals set
Enter a q to quit.
1
Indicators: Fasten Seatbelt No Electronics
1
Indicators: Fasten Seatbelt No Electronics
4
Indicators: Fasten Seatbelt -
1
Indicators: Fasten Seatbelt -
2
Indicators: Fasten Seatbelt -
1
Indicators: Fasten Seatbelt -
1
Indicators: Fasten Seatbelt -
1
Indicators: Fasten Seatbelt -
1
Indicators: Fasten Seatbelt -
1
Indicators: - -
```

Figure 3.1: Part 2 OpenCL Outputs

## 4 Discussion and Results

Lab 2 gave a more complex example of designing Finite-State-Machine, through the use of airplane indicator lights. This lab seemed very complex in the beginning, as I had very little experience with state machines. However, adhering to the design steps given in the class lectures, the lab was tremendously easier. To me, the hardest part was drawing the state diagram. After several drafts, I finally came up with the state diagram shown in figure 2.1. When the final draft was completed, the logic created in VHDL was somewhat simple, consisting of one process statement containing case-whens and ending with two concurrent statements to assign the output. Once the logic was defined, creating a wrapper was fairly easy because it was very similar to lab 1; this holds true with the test bench as well. After, mapping the design into OpenCL was straight forward, mostly because the lab manual is very clear when implementing this feature. The biggest struggle with OpenCL was having slight syntax errors, then having to wait for the files to compile. Upon the completion of this project, the FPGA board will mimic the designed state diagram. The board will begin with both output LEDs on, then, 5 switches and 1 key will control the inputs. The reset was configured as active low to match OpenCL, and ivalid was used to ensure valid input was received. The clock was mapped to one of the keys to detect the rising edge. Whenever the reset and ivalid was high, and a valid transition state was given, the current state would move to the next state. The airplane begins at GROUND and both LEDs are on, when a 100 is received, the state is transitioned to  $ALT_{10k}$  with the "No Electronics" light off and the "Seat Belt" light on. When a 010 is received, the state is transitioned to  $ALT_{25k}$  the "No Electronics" light off and the "Seat Belt" light on. Then, upon 5 consecutive smooths, 001, the state stays at  $ALT_{25k}$ , but the output LEDs are both off. Whenever the reset is low, the state will immediately be transitioned to the ground state. OpenCL gives the same output as the board, however, the outputs are displayed through software by the terminal.

## 5 Conclusion

VHDL is a powerful hardware description language (HDL), that provides a somewhat-simple, interface which helps users quickly and efficiently design, simulate, and test circuits. Because FPGAs are re-programmable and the user can design nearly any circuit, this makes these tools very expensive. For this lab, the main focus was using process statements to control the state transitioning. Upon successful logic design, Quartus was able to generate the Moore FSM shown in figure 2.2 and display the proper LED outputs on the FPGA. When the design was mapped to OpenCL, the terminal showed the proper outputs. Overall, I believe this lab was completed successfully and I feel more confident with FSMs. If I could redo this experiment, I would immediately recognize the method for implementing state machines in VHDL and waste less time coding the logic.

## References

- [1] W. J. Dally, R. C. Harting, and T. M. Aamodt. *Digital Design Using VHDL: A Systems Approach*. Cambridge University Press, Cambridge, 2015.
- [2] M. Smith. *Lab 2: Lab 2: Airplane Indicator Lights Controller*. Clemson University, 2019.

## 6 Appendix

No further references were needed for this project.