**Lab #5**

**ECE 4304 Spring 2021**

**Professor Aly**

**California State Polytechnic University, Pomona**

**Ahiezer Lopez ahiezerlopez@cpp.edu Bronco ID:012715521**

**Joe Si**

**joesi@cpp.edu**

**Bronco ID: 012636091**

**Sander Zuckerman sazuckerman@cpp.edu Bronco ID: 012644671**

**3/16/2021**

**Objective:**

Students will create an ALU/AU that outputs BCD values to connect into a Seven Segment Decoder. Students will have a clock, 3 switch/selector bits, 8 input bits (4 bits each number), a reset, and a go bit. The selector bits will determine which action is going to be done within the ALU/AU and the go bit will tie into the multiplication portion of the code. The maximum output will come from the multiplication which will output 8 bits.

**Materials:**

* FPGA (Nexys A7-100T)
* Vivado Software
* Computer

**Contributions:**

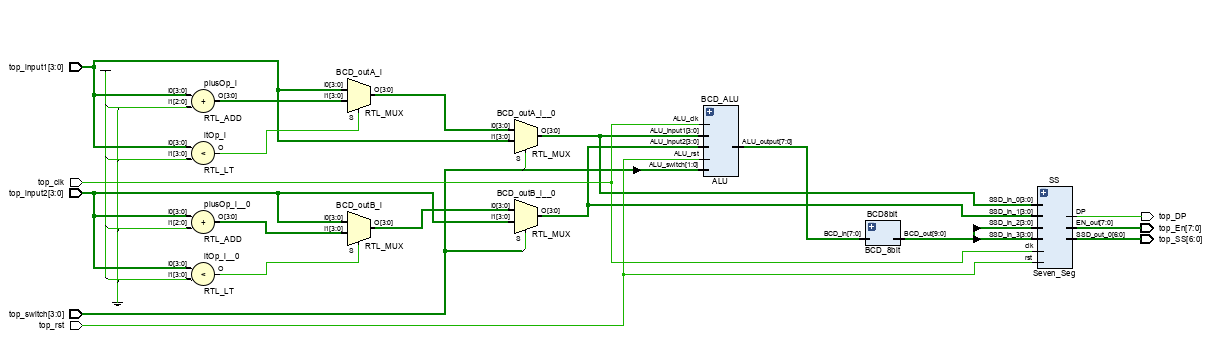
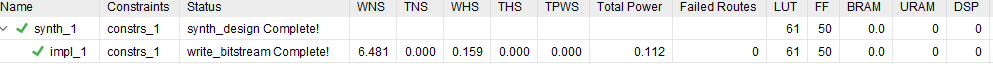
For this lab everyone had their own part in it. Ahiezer designed the adding and subtracting module which was done using a full adder mentioned in one of the class lectures. Joe was responsible for creating the multiplier portion of the ALU. Each member had their part within the multiplier as it created the most issues. Sander had helped with each portion of the code debugging and fixing any issues that may have happened throughout the design process.

**Design Process:**

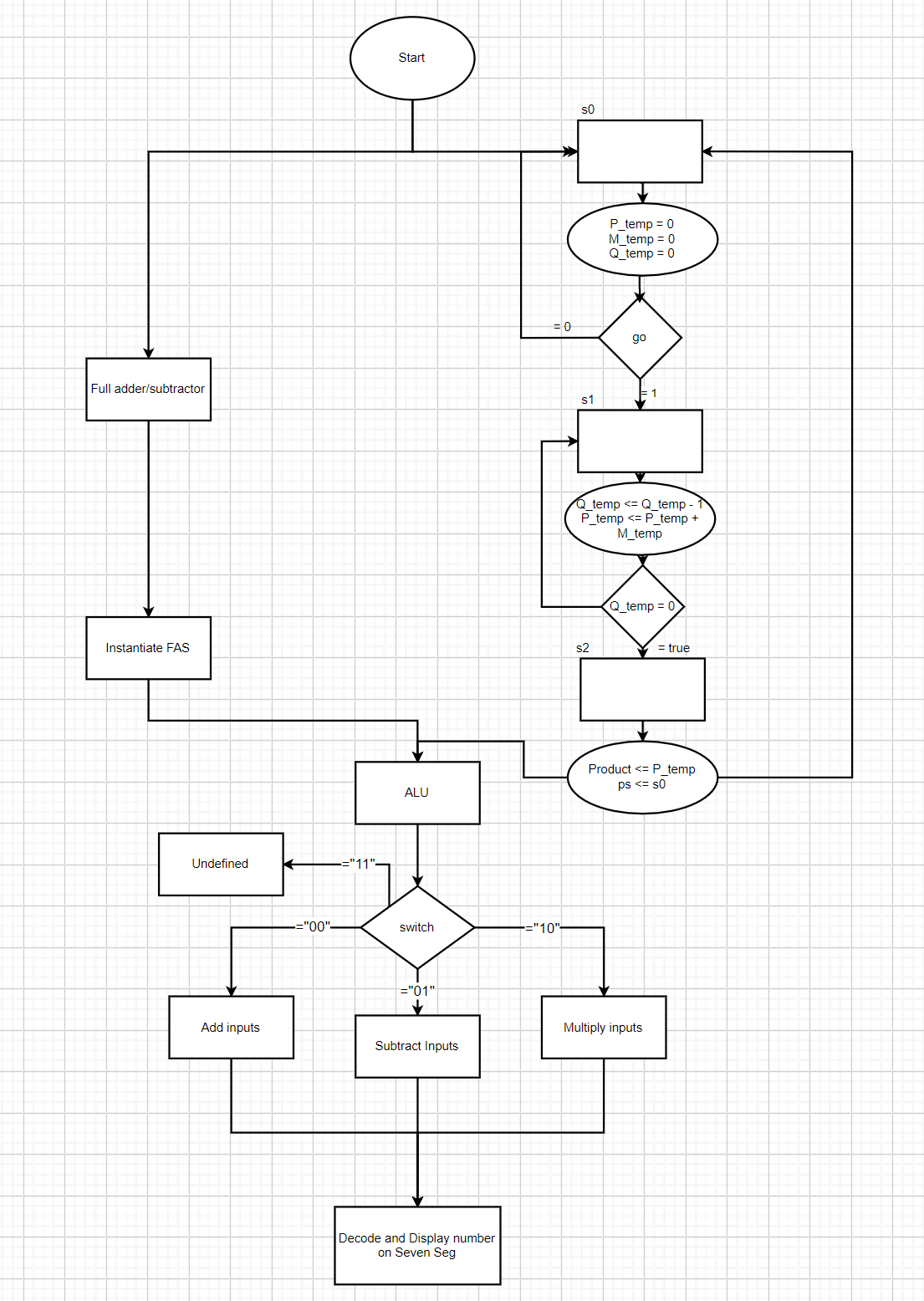
Using the flowchart from below, we implemented our idea on how to approach this lab. We ran into some difficulties with the Multiplication module as it has been a while since we’ve created a fully functioning multiplier using state machines. Certain numbers would not multiply properly. We were able to separate the modules and create another BCD module dedicated to reading values from the multiplication. We were also able to create was what needed and switch between bcd and hex values for our inputs. In the end we were able to create a working ALU with 61 LUTs and 50 FFs with a power consumption of 0.112mW.

**Design:**

Using the seven segment code from last week’s lab, as well as the adding and subtracting method, we were able to create the ALU with the addition of a multiplication component. The toughest part was the multiplication portion as there was some endless fine tuning to get the right number to output and trying different methods that worked best for multiplying. The multiplication was done with a counter and an adder, the numbers are loaded and will add to a sum until the counter hits zero.

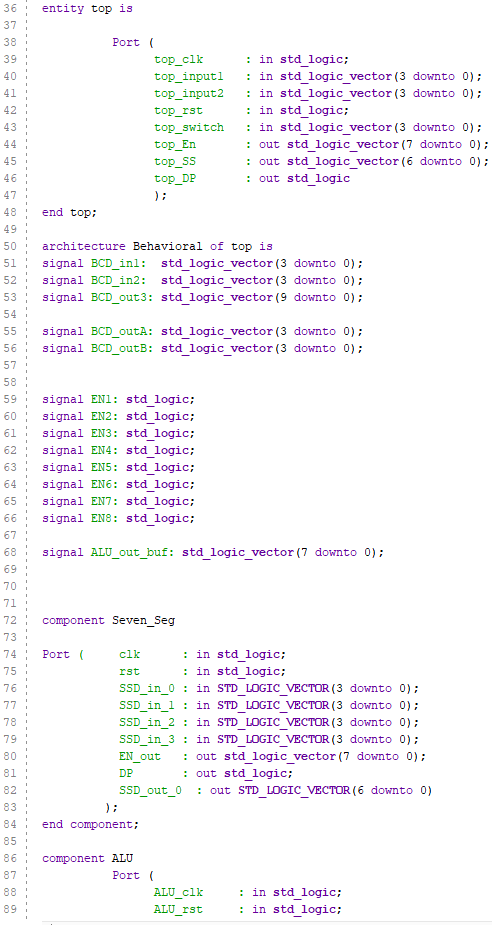
**Circuit Diagram:  
  
**

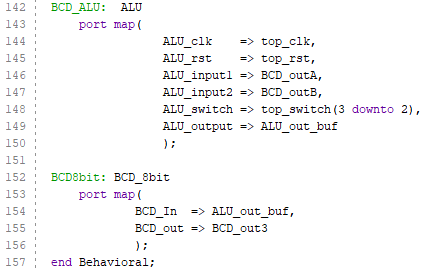
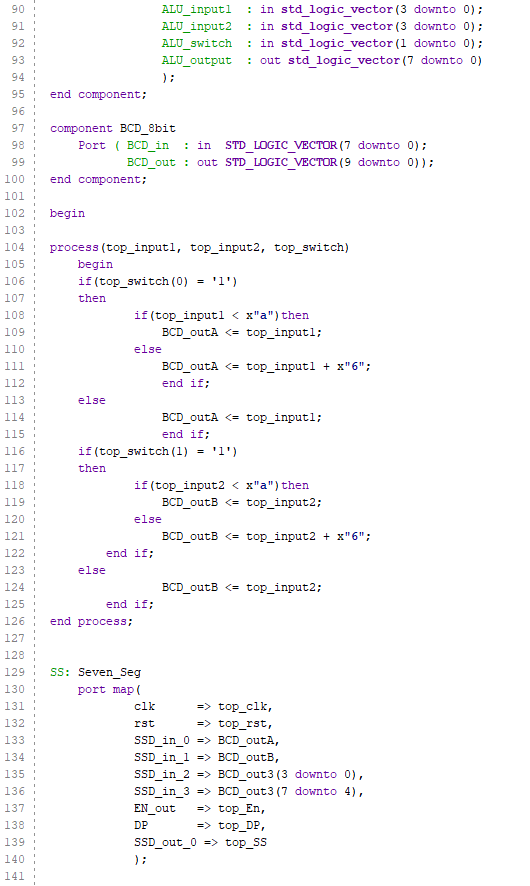
**Implementation:  
 Logic Flow Diagram**



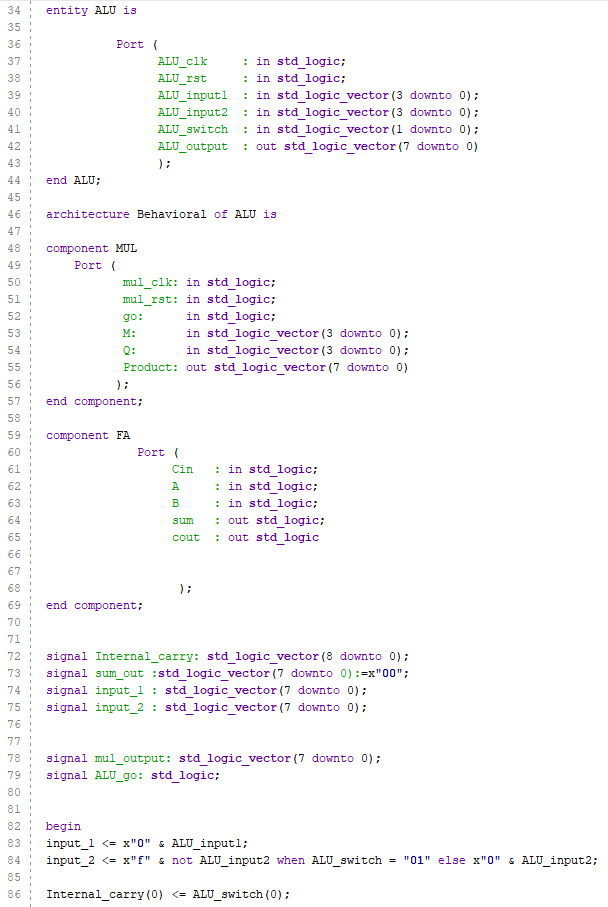
**Design:**

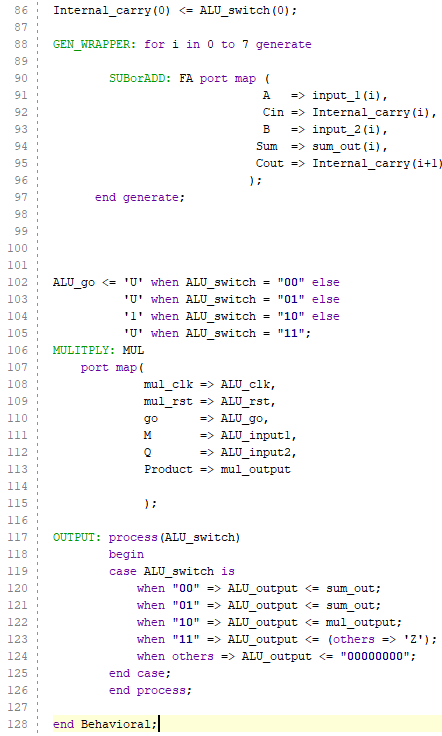
***Top Module***

****

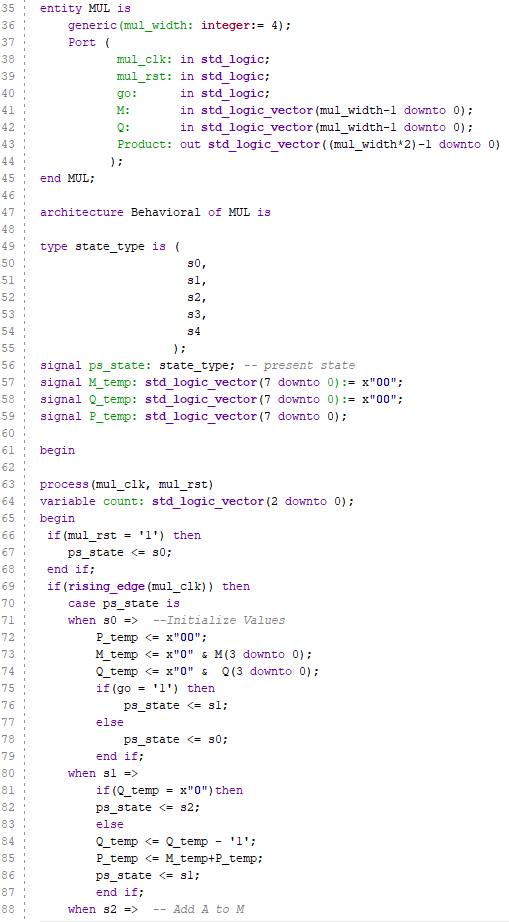
****

***ALU Module:***

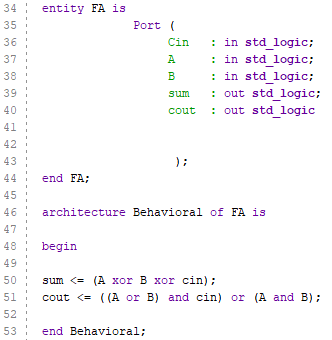
****

****

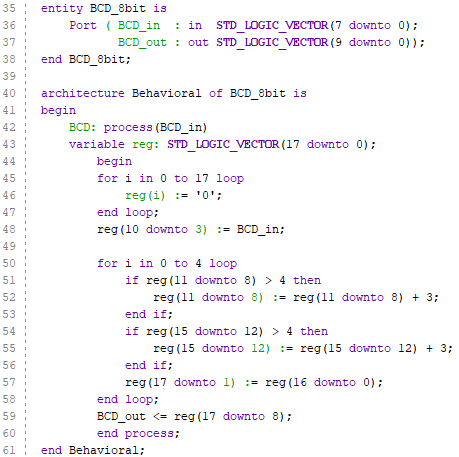
***Multiplier Module:***

****

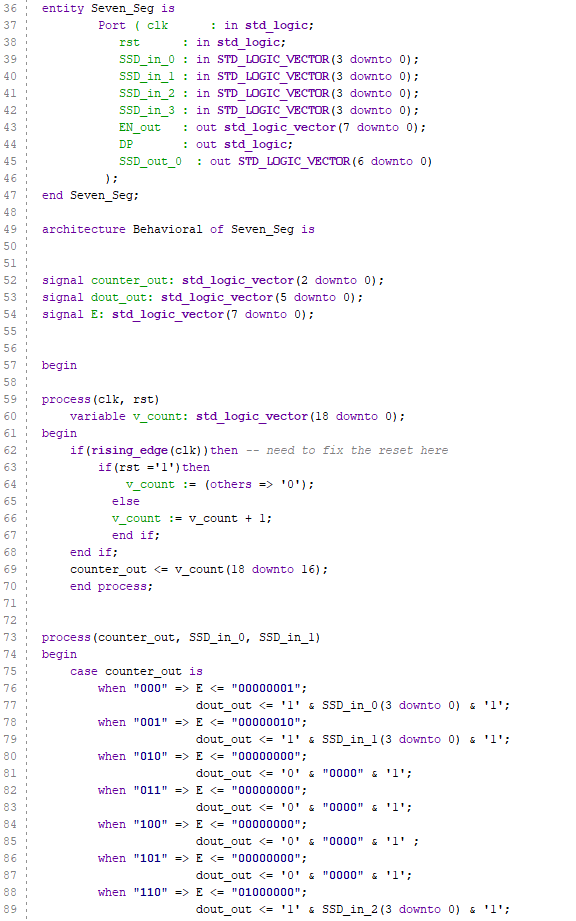
***Full Adder Module:***

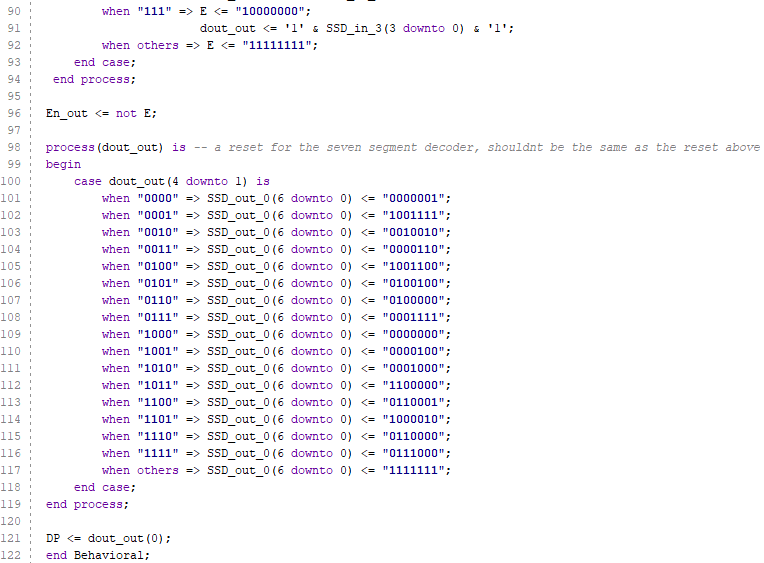
****

***8-Bit BCD Module:***

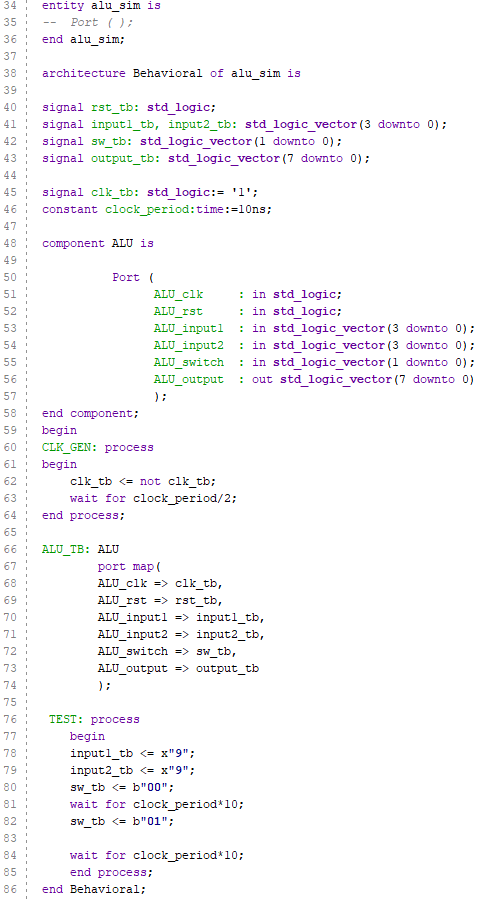
****

***Seven-Segment Display Module:***

****

****

**Test Bench:**

****

**Analysis:**

The main part of this lab was implementing the ALU multiplication portion. With the we had five different states for our multiplication process. State zero was to initialize all values to its default, which will either be zero or the input value we choose. The State two does the initial addition to the accumulator register. State three does the repeated step of adding to the accumulator. State 4 is where we shift the values of the register and the multiplier. And the Final state is where we would load the final value, the accumulator register and the multiplier, to the output. Unfortunately, this method was not able to work properly and changed each state to accommodate a down-counter loaded with the input and continuously adding until the counter hits zero. Similar to how actual multiplication works

**Conclusion:**

In this lab, we were successfully able to create an ALU to BCD circuit. Since we already had the BCD converter and the Seven Segment modules from previous labs, our main focus was the ALU. We came to the conclusion to create a multiplier utilizing state machines, more specifically a moore machine. There were some issues with the add shift method as there were some faults with multiplying certain numbers. However, we were able to overcome these issues by redesigning the method completely and doing a down-counter loaded with one of the digits and continuously adding until it hits zero . This lab allowed us to further understand the functions of ALUs and gave us more experience in coding state machines, as well as registers.