Erich Viebrock

ECPE 174

**Post Lab #6/7**

**ALU & Timing, Placement, Memory**

Problem Summary:

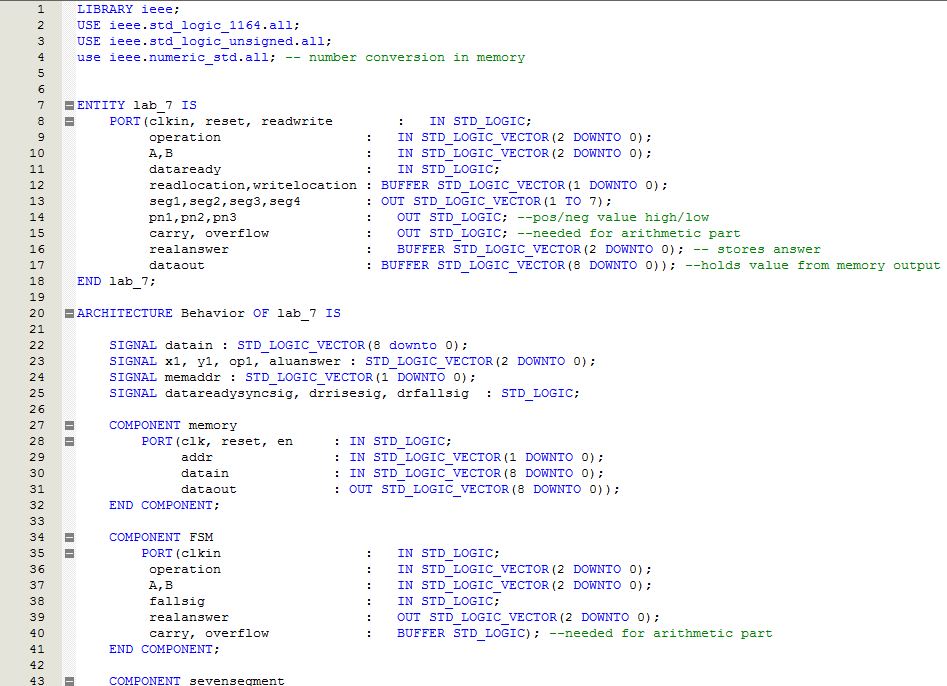
In this lab, our goal is to design an ALU. There will be two numbers entered into the system, “A” and “B”, and a command for which operation to execute. The ALU will be able to add, subtract, compare the two numbers, or check if A is 0. The two numbers, operation, and result will all be displayed on the seven segment display. Later, a memory component is added to the ALU, allowing the ALU to compute up to 4 different commands while reading from memory. The entire design will be hierarchical, as this exercises organized VHDL techniques.

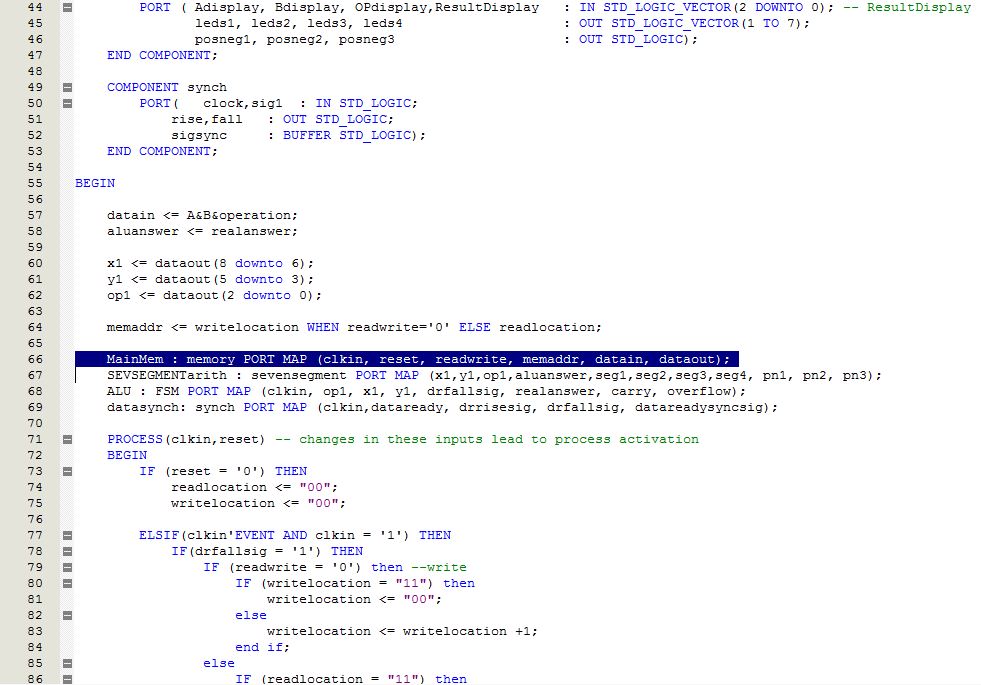
Assumptions:

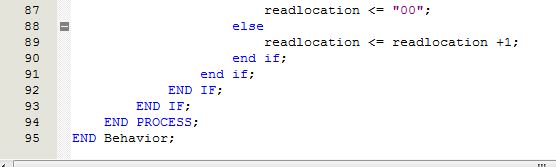
The adder/subtractor used for adding and subtracting two numbers, unfortunately outputs the result in a 2’s complement binary number. Also, we decided our inputs, A and B, would be entered in 2’s complement. Although this allowed us to represent negative numbers for inputs and outputs, there was a small flaw with our output. Our output was represented as a three bit 2’s complement number, which meant the result could only range from -4 to 3. This proposed an issue, as there are plenty of situations are result could be outside of these boundaries. Therefore, an “OVERFLOW” LED would turn on to indicate this situation was occurring.

VHDL:

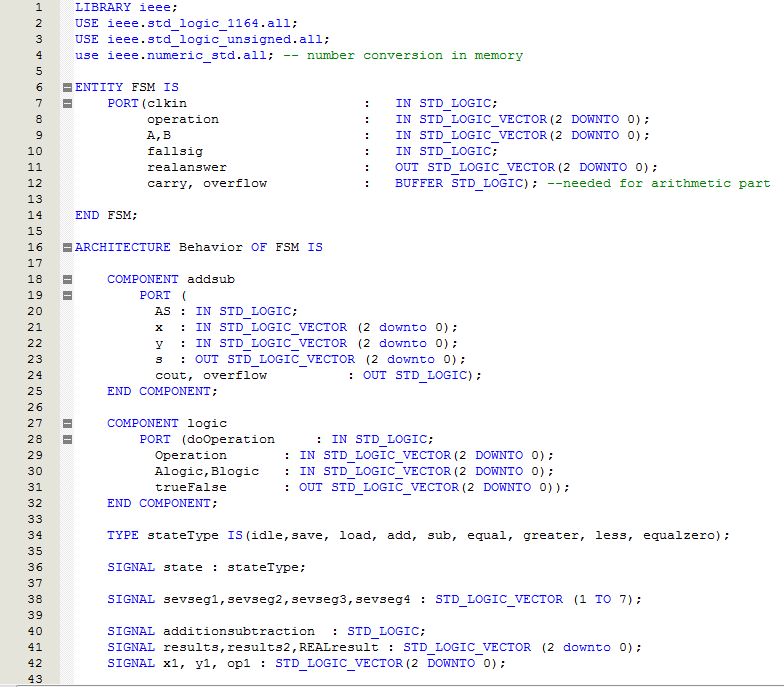
Main

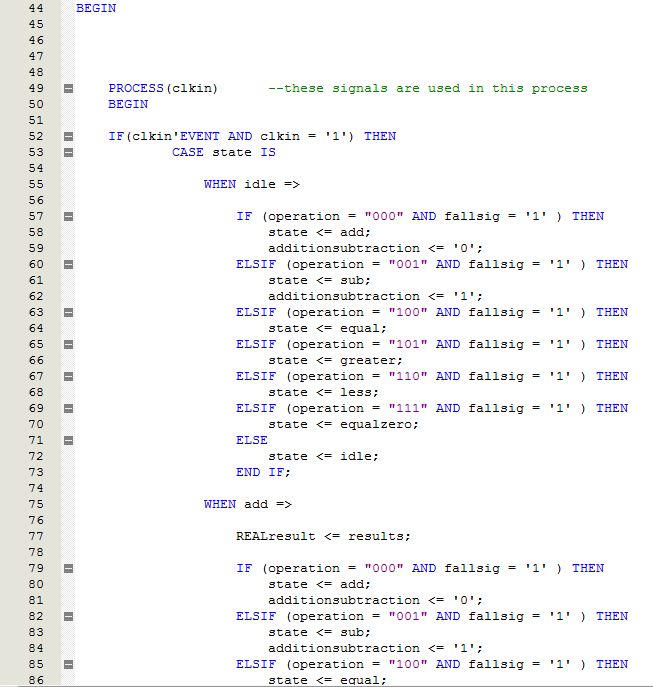


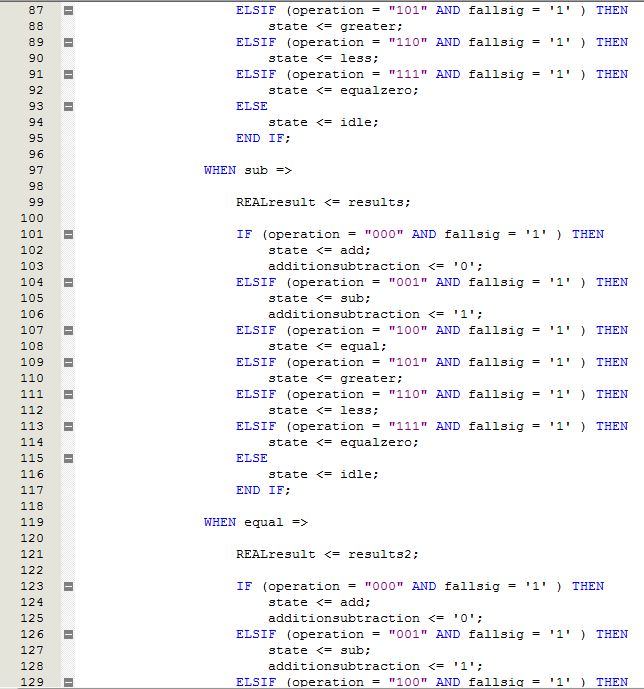


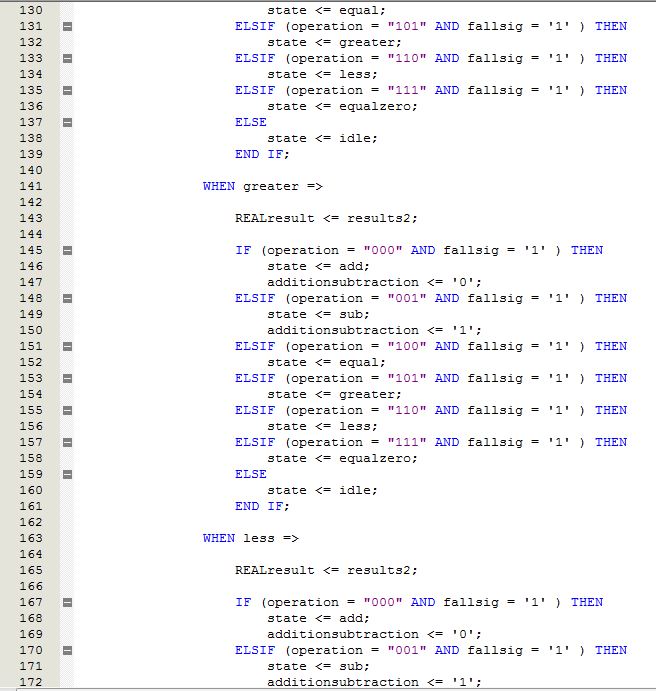


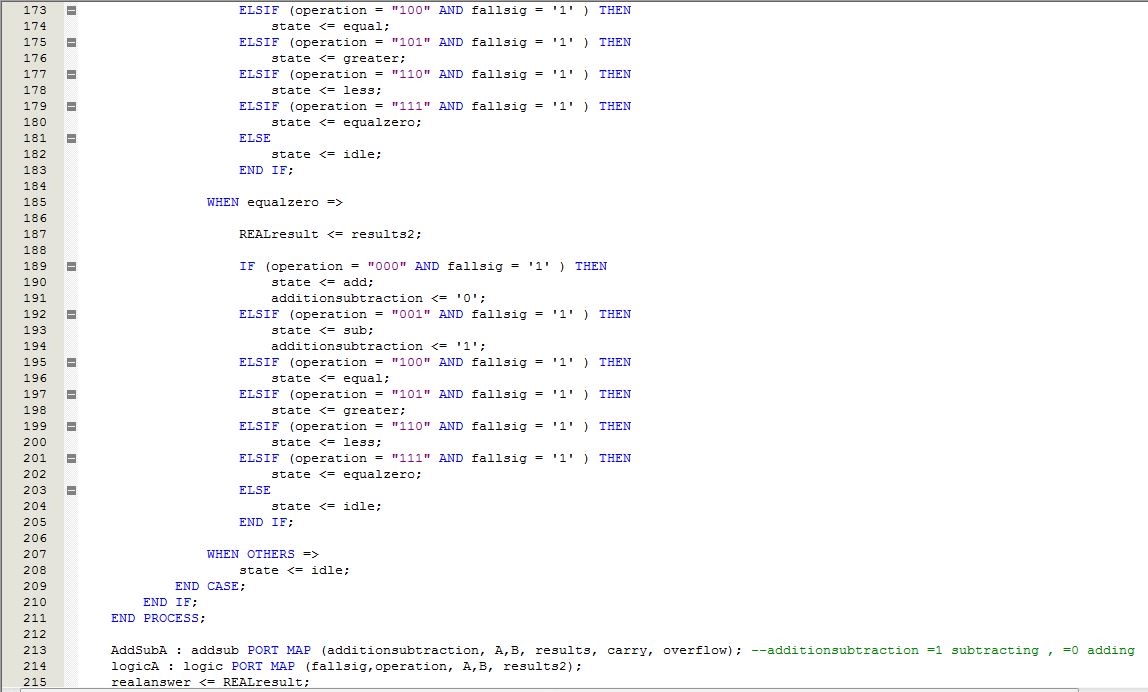
FSM





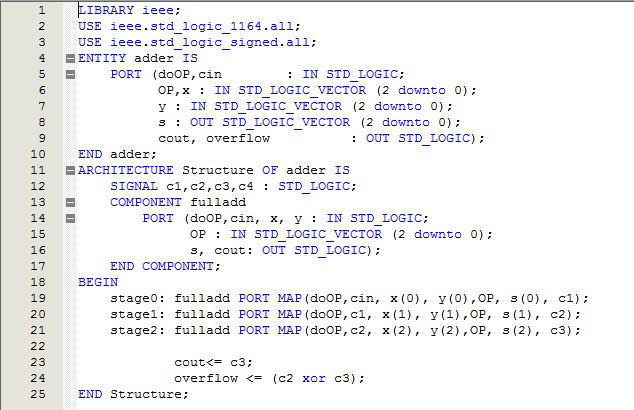




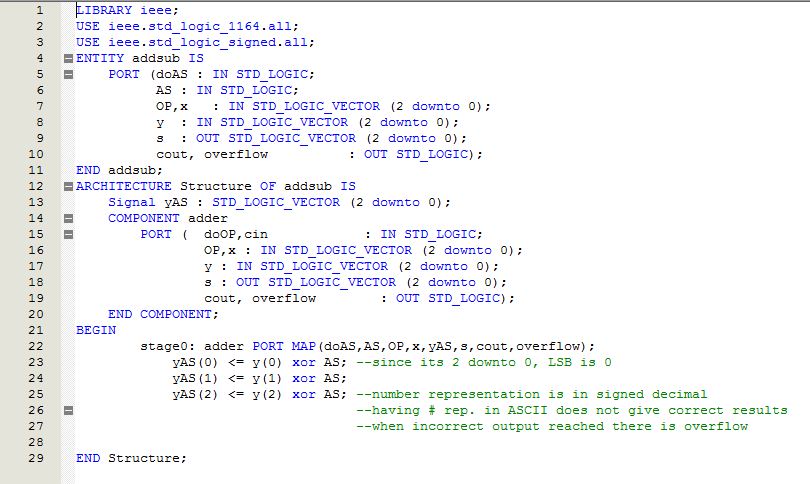


C:\Users\Erich Viebrock\Desktop\Classes\Fall 11\Advanced Digital\Lab 7\fsm6.JPG

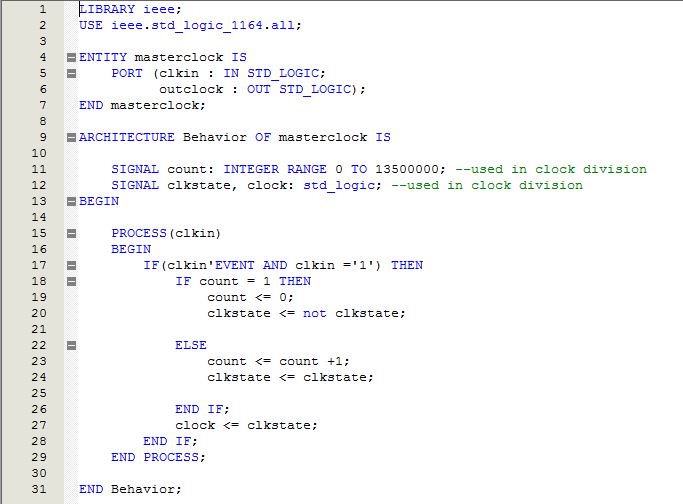
Adder



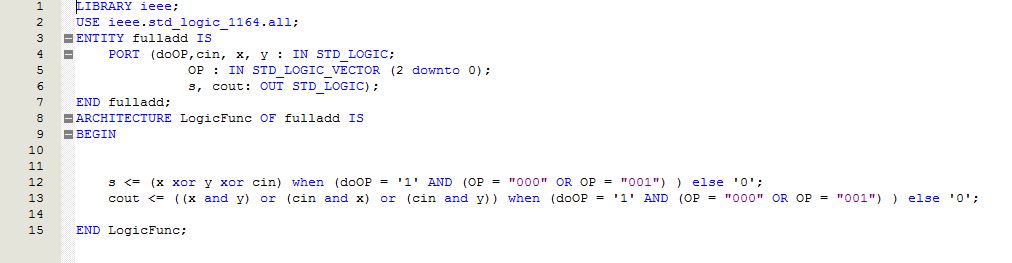
Add/Sub



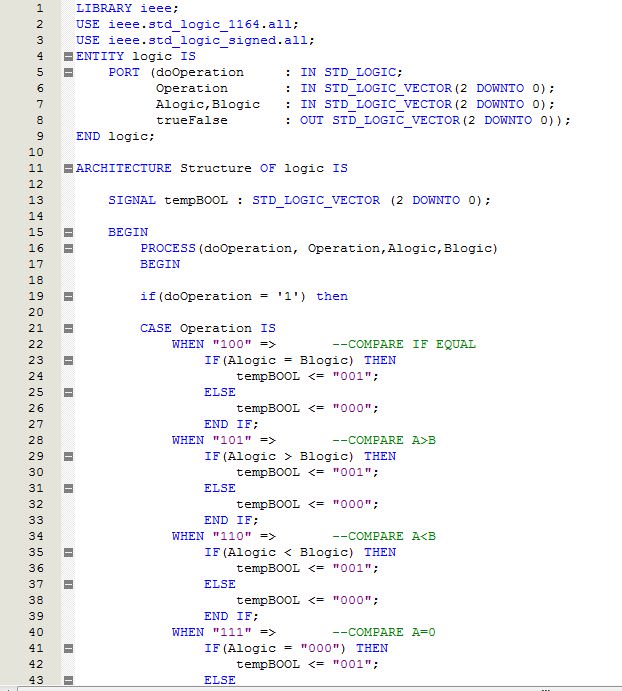
Clock

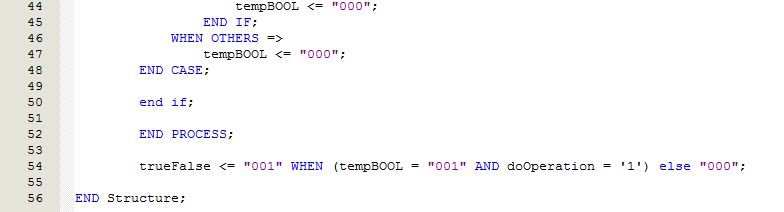


Full Adder

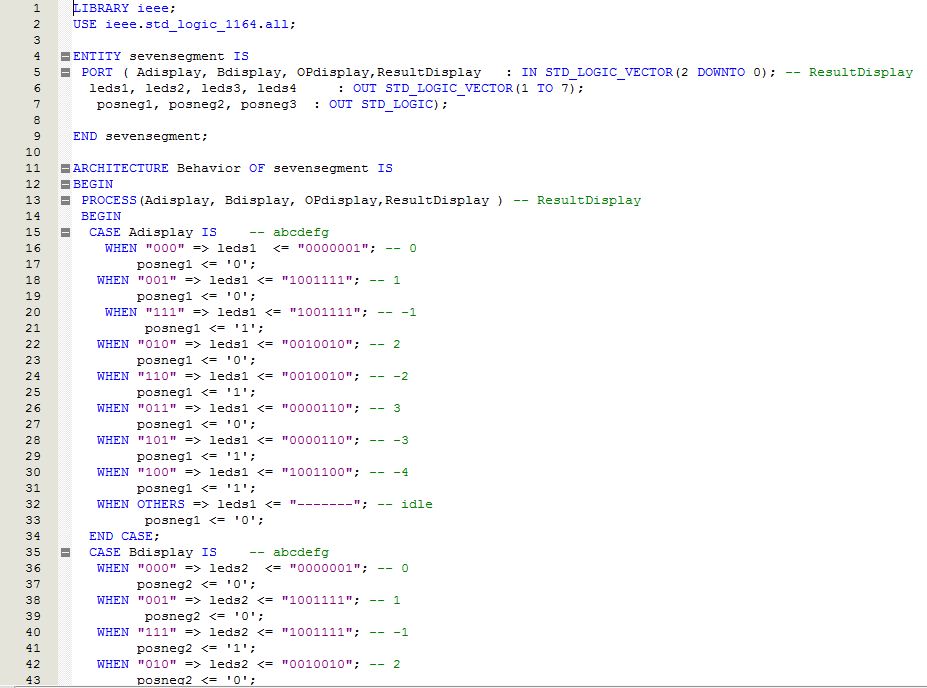


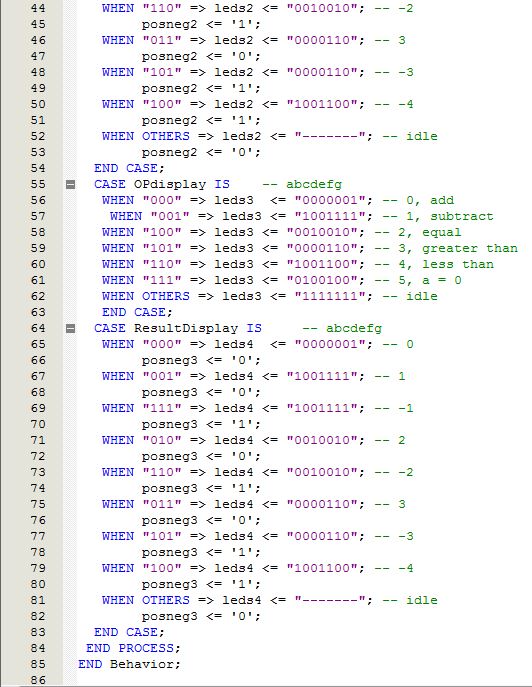
Logic



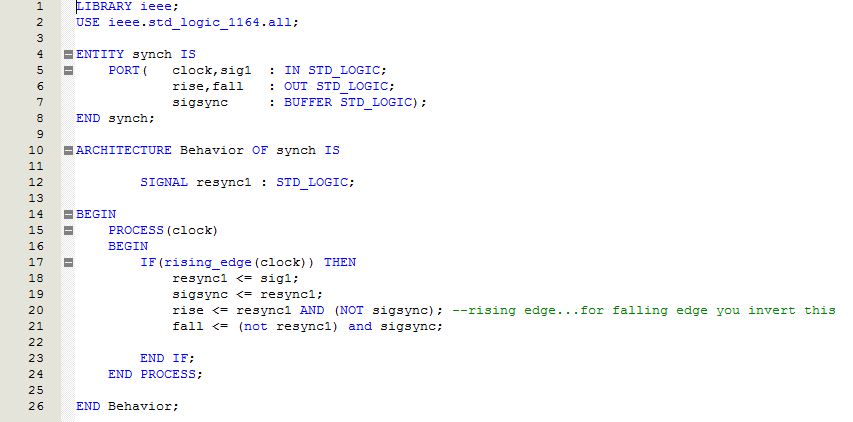


Seven Segment





Synchronizer:



Verification Procedure:

While implementing the design, the biggest problem we noticed was the ALU was computing the result during the write stage, rather than read. Although this did not propose an issue during the read stage, the ALU should not have been computing the results in the write stage. In order to fix this problem, creating two separate inputs for read and write would be required to allow the ALU to function as desired, while making the separate inputs equal. Therefore, it would be much easier to command the ALU to calculate the result of the set read inputs, and command the ALU to do nothing for write inputs. Although in reality this will most likely not be the case as this is a practice of slow computation, this is the method required for the lab.

Post Lab Questions:

Our test procedure was to first construct each block of the ALU separately, then combine the ALU later. These initial tests ran fine, although combining the logic proved to be a little more complicated. Because the adder outputted 2’s complement numbers and the other pieces of logic did not, this meant a conversion was necessary in order to use the seven segment decoded effectively.

Each unit, inputs and outputs, required 3 bits. There was a design choice in choosing how these three bits would represent our numbers. In our choice, we chose the three bits to resemble 2’s complement, as this method would allow us to represent negative numbers, while representing a wider range than 1’s complement or sign magnitude. Although a clock divider was used to slow down the output to 1 Hz, theoretically the output could be displayed up to 27 MHz. However, I still feel the legitimacy of this figure is inaccurate, due to other digital displays being much slower than this.

The time requirements for the add/subtract unit requires a much larger time requirement than the logic unit, because of adder/subtractor requires a larger quantity of gates, and flip flops. The physical design of an adder is fairly complex, while the logic unit merely compares numbers, and results in true or false. Therefore, the adder/subtractor would have a fairly larger time requirement.

The biggest flaw our ALU possessed was lacking patience. The ALU would accept inputs for A, B, and Operation. However, while these inputs were entered into the system, the ALU would already calculate the result before commanded to do so. Therefore, if this issue was fixed, this would be an improvement to the system.

Of course, the more hierarchical the ALU design is, the faster the ALU will be able to run. This makes logical sense, as the compiler is able to search through smaller amounts of data to be able to compute the result.