Digital Design

CSCE 2114-L007

Blake Fasse

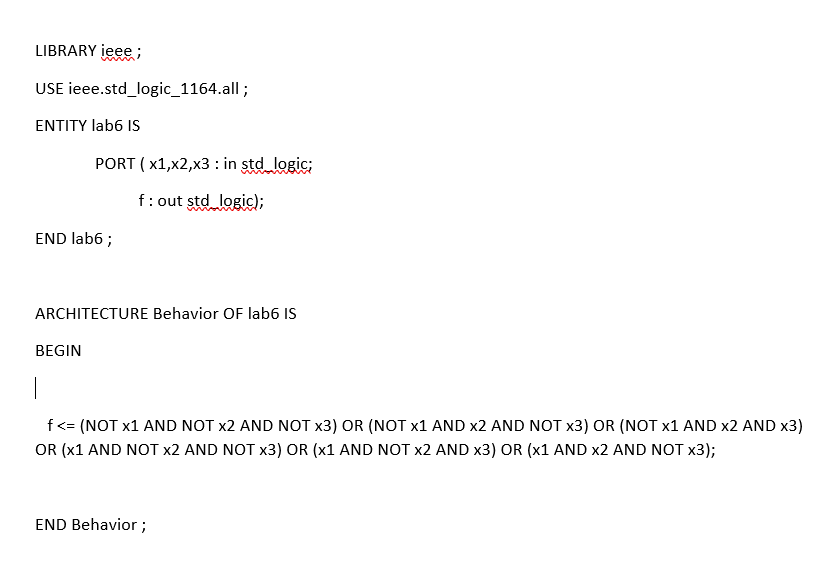
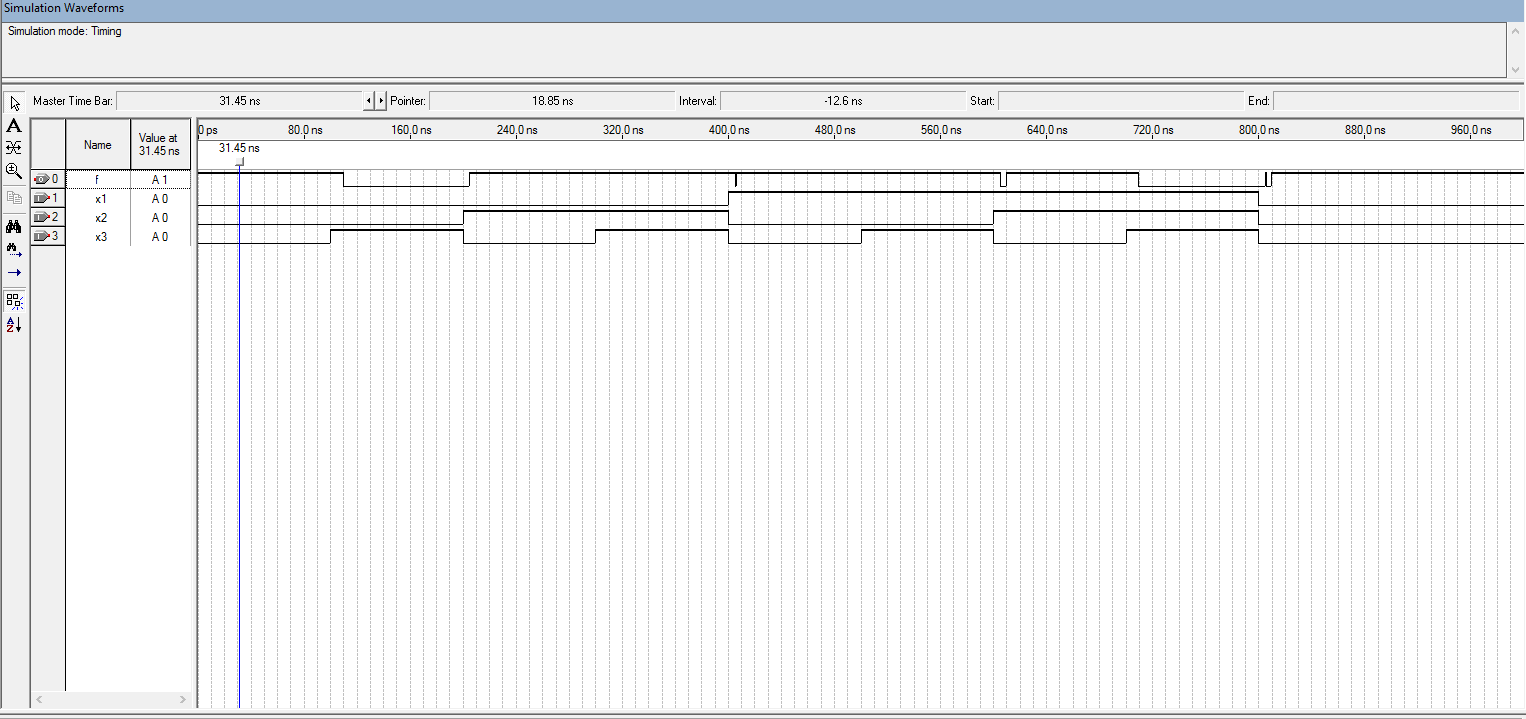
April 18, 2017

[bafasse@uark.edu](mailto:bafasse@uark.edu)

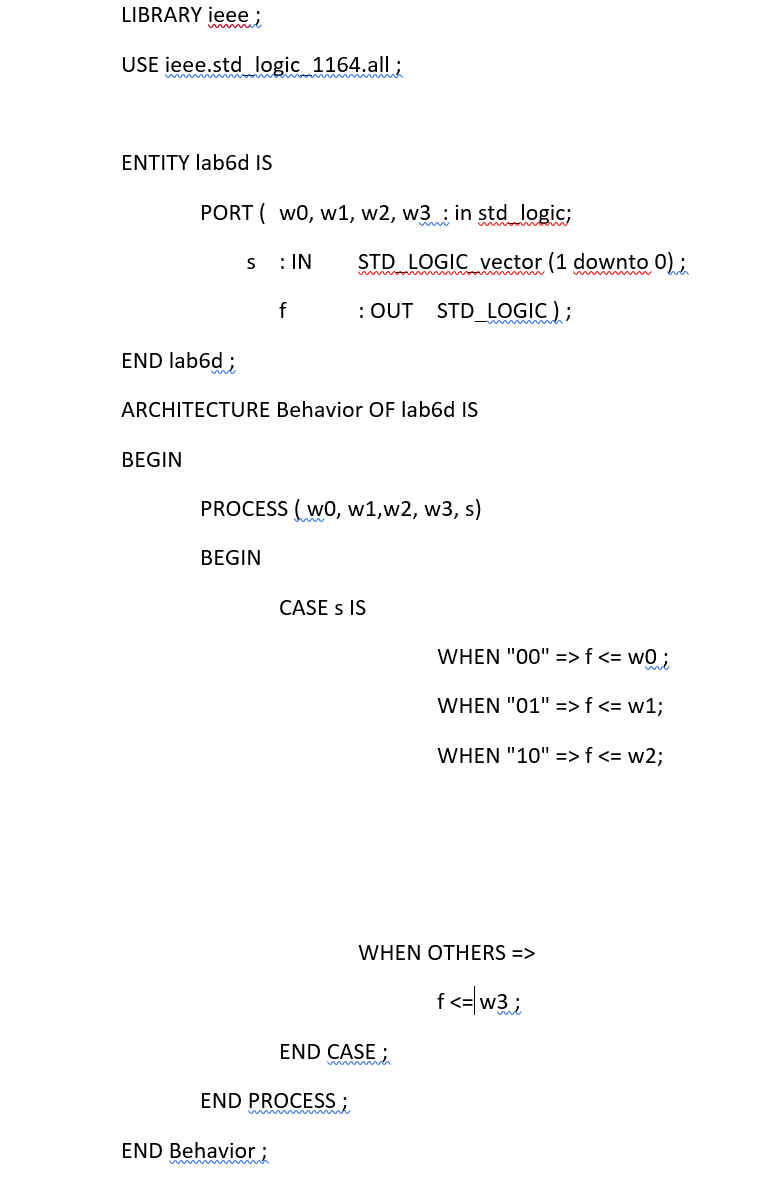
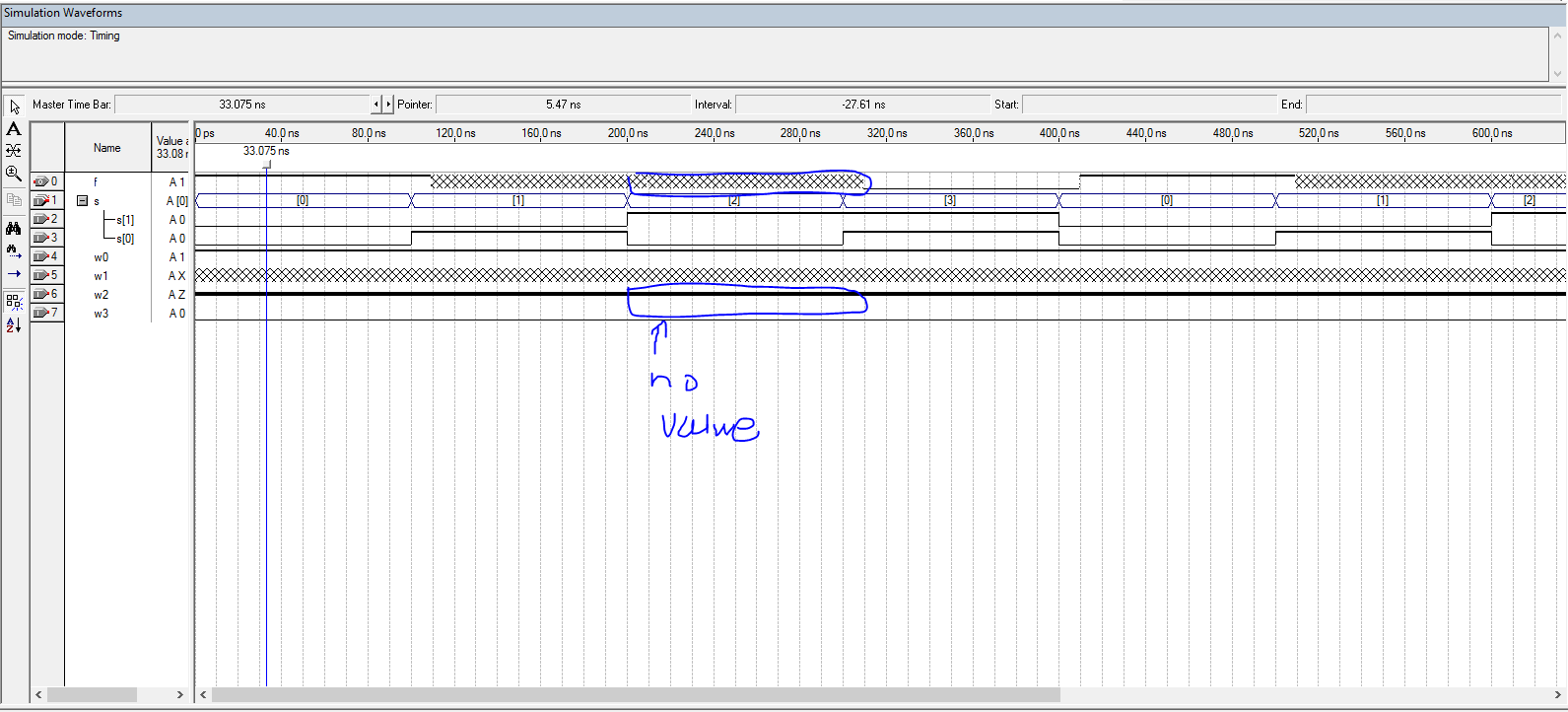
**Introduction**

This lab was split in two due to how much work was needed to be done on it. The first part of the lab consisted of building a 3-input function and a 4-1 multiplexer while the second part of the lab consisted of making a 4-bit adder/subtractor. It is unclear what to expect with the first part of the lab with the 3-input function but with the 4-1 multiplexer it is expected to simulate the outputs of a 4-1 multiplexer and with the 4-bit add/sub it is expected that any time the add/sub is set to active high then it will add the values together and when its active low it will subtract them.

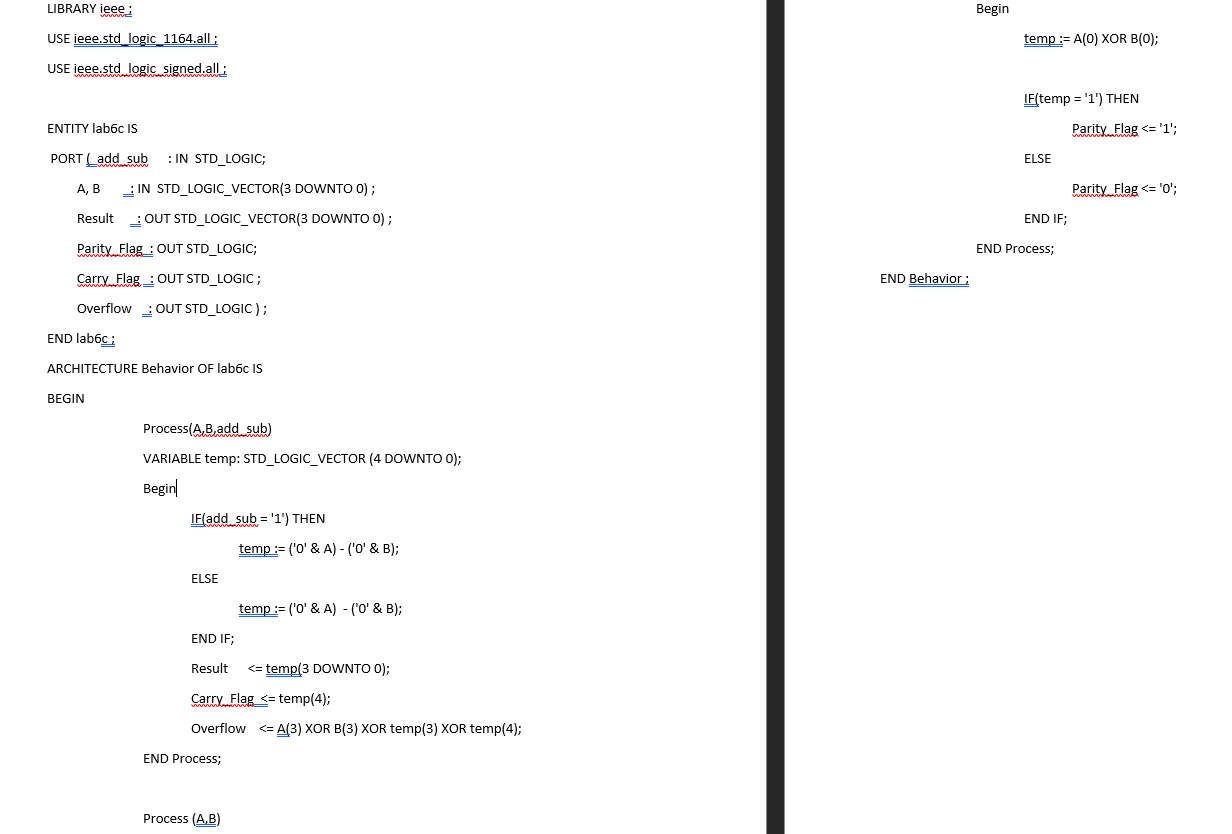
**Design**

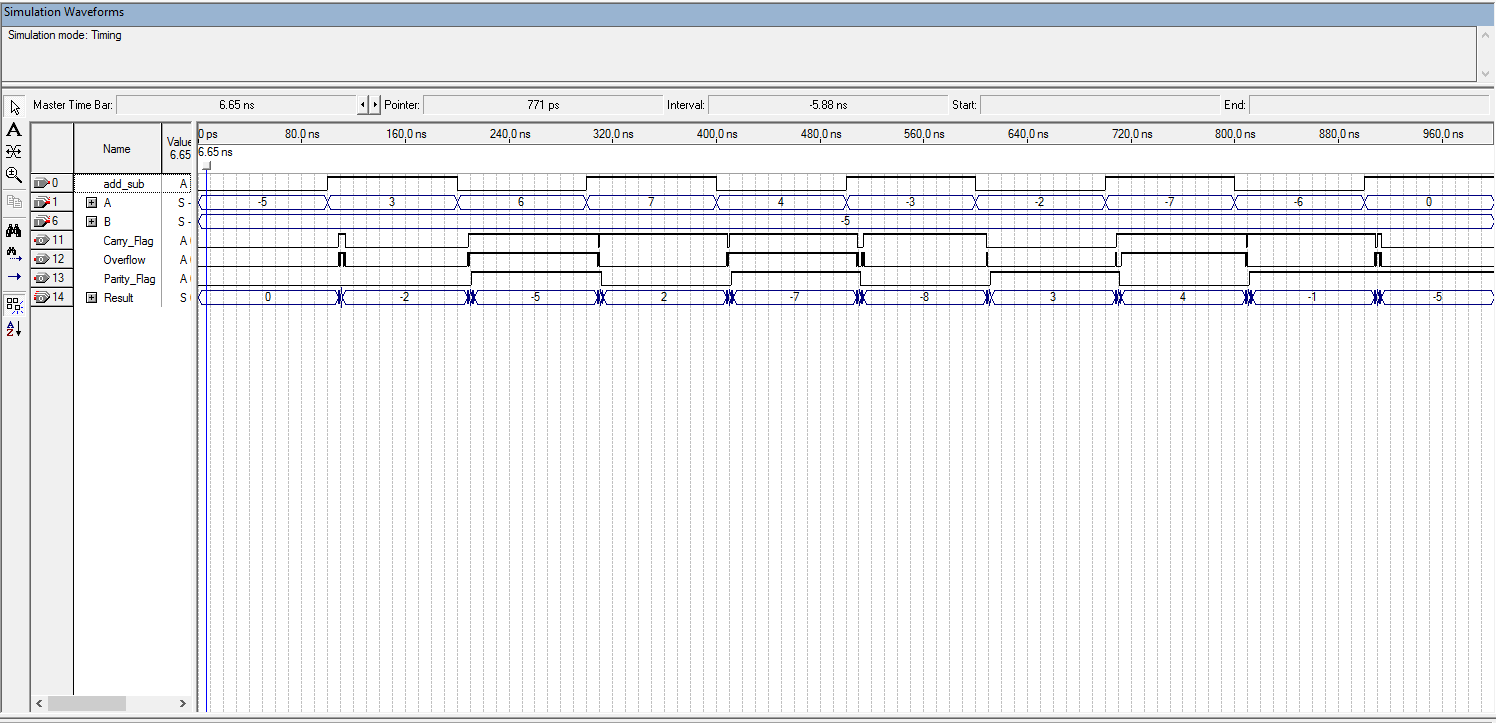
Most of the VHDL code that was written for all the parts of the lab was not difficult because a lot could be referenced from the VHDL notes provided online. The first part of the lab was to build a 3-input function in sum of products form. The provided function that was to be implemented was F (x1, x2, x3) = ∑m (0,2,3,4,5,6). Going into this it was unclear what to expect the output of this function to be but upon further inspection of the question at hand it became clear what the output should be. Since function provided is in sum of products form that means every output listed in the function is going to be set to active high while the rest will be active low. Also the waveform file goes from 0-1000 nanoseconds and with there only being eight outputs in 100ns intervals everything past 800ns does not count. Shown below is a copy of the code and output of this part of the lab. 

The second part of the lab was to build a 4-1 multiplexer. Using the truth table that was shown on the lab document it was easy to understand the output. The 4-1 mux consists of two inputs, s1 and s2, and four outputs, w0, w1, w2, and w3. Each of the four outputs is assigned a specific value with w0 being assigned 1, w1 being assigned X, w2 being assigned Z, and w3 being assigned 0. According to the truth table, depending on what the inputs are for s1 and s0 determines which output is put into effect. Whenever the output is 00 (s1, s0) then w0 is implemented, 01 implements w1, 10 implements w2, and 11 implements w3. It should be noted that anytime w1 or w2 are implemented then there is no output because the values assigned to them have no value. Below is picture of the code and the waveform output.

The third part of the lab, which took place the following week, had the user build an adder/subtractor. There are three inputs, A, B and add/sub, and four outputs, result, parity flag, carry flag, and overflow. Since this is only 4-bits and the values are signed that means that only values between -8 and 7 and be displayed without overflow occurring. The input was assigned multiple values -5, 3, 6, 7, 4, -3, -2, -7, -6, and 0 at 100ns intervals and all of B was assigned the value -5. The add/sub was set to a clock signal of 200ns. The results were as expected with values of A and B being added together when add/sub was set to active high and being subtracted when set to active low. The overflow, parity flag, and carry flag signals worked and showed whenever overflow had occurred, when the number of ones in a binary number are even, and if there is a carry-in value, respectively. Below is a picture of the code and waveform output.





**Conclusion**

This lab was a lot easier than originally expected. Building this code from scratch showed that VHDL is similar to C/C++ code with the only difference being the layout.