Name: Jilin Zheng

UID: U49258796

Discussion about ALU:

I chose the carry-select adder because it seemed relatively easy to implement as long as I could figure out all the inputs and outputs. Doing some research on the carry-select adder allowed me to easily determine these inputs and outputs. The MUXes used in the adder are also pretty intuitive and easy to implement.

I believe my decoder is faster than my ALU. In my implementation, I left the 7-segment display's program to the 'fullSystem' module that instantiates the decoder and the ALU. In my case, my decoder simply took the 12-bit instruction and "chopped" it into a 4-bit operation code (opCode), 4-bit input A, and 4-bit input B, and outputted the three 4-bit values. On the other hand, my ALU performed both of the logical and arithmetic 3-bit operations with the two inputs (the ALU's inputs were the outputs of the decoder, excluding the carry-in input). I used a MUX after the operations to choose the resulting output based on the most significant bit of the opCode input. Since the ALU is performing many more operations than the decoder, I believe my decoder is faster than my ALU.

My design utilizes 47 LUTs. Looking at the Artix-7 documentation, with each slice containing four LUTs, I am using just under 12 slices. I believe it can be smaller, but not by much. I used two wires in my fullALU for the mode and for the operation for clarity, but I believe I can just directly read the opCode and avoid the new wires. Aside from that though, I believe most of my implementation is pretty space-efficient. The biggest parts of my design are the ALU’s logical and arithmetic operations, as well as the seven-segment display’s cathodes’ control.