The new ALU is definitely more complex but has the same core functionality and should utilize a level structure. Since we've essentially increased the number of cases, we've increased the number of inputs and outputs of the ALU which acts sort of like a LUT. For the adder, I've chosen the carry select adder due to its simplicity and ease of understanding and implementation. Since you're essentially performing two calculations in parallel and simply using MUXs to choose the final values, it's fast and lightweight compared to the regular ripple adder. When comparing the ALU and decoder, I'd say that the decoder is faster due to its simple logic design. It's essentially a MUX that outputs a specific value based on some inputs. Compared to the ALU, which may have less cases to check but must perform many calculations alongside those, the decoder is simple and is thus faster. Overall, the design is very lightweight using about 70 LUTs (about 33 for the ALU (1 + 8 functions * 5 bits each which includes c_out), 8 for the 8-bit decoder output, 8 for the 8-bit display module, and about 13 LUTS for the carry select adder (8 full adders + 5 MUXs)). Assuming we have 2 slices per CLB and 4 LUTS per slice, we are using about 18 slices and 9 CLBs on the FPGA. I'd say this design could get smaller but it's already fairly optimized for its use case as some of the modules are almost LUTS themselves. The main part that I could see being optimized is the ALU with all of its operations and the adder. Since the ALU is the largest part of the design due to it needing to work with many inputs, and perform many calculations that require sub modules or extra logic.