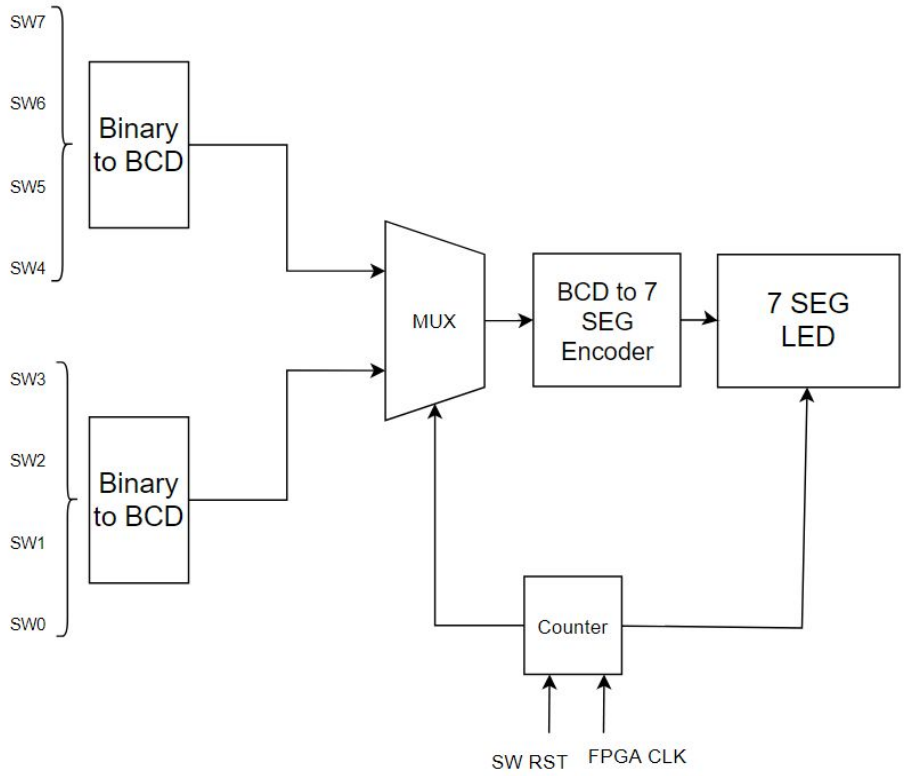
Lab 3 - BCD Seven Segment Display

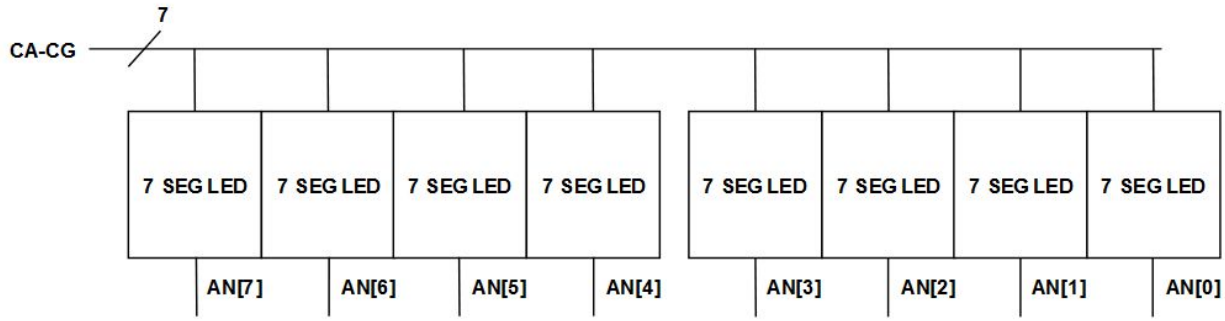
Group A - Yuta Akiya, Kyle Le, Megan Luong
Prof. Aly
ECE 4304

Architecture

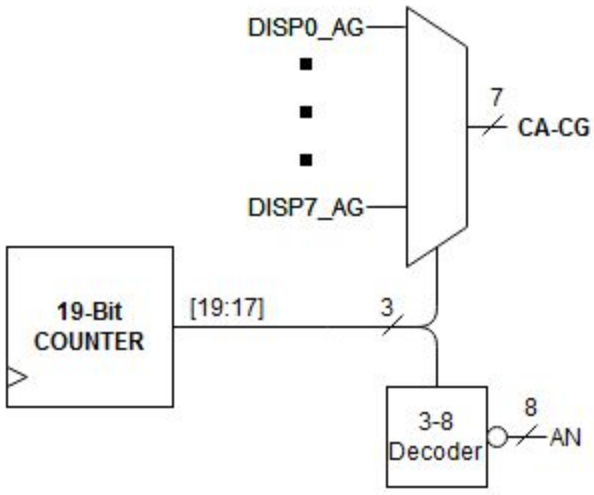
The architecture of this BCD to seven segment display features four main components. A binary to BCD converter, a BCD to seven segment encoder, a counter, and a mux. Two sets of four input switches are taken in by two BCD converters and the BCD output is then passed into the mux as inputs. The select bit of this mux is controlled by a counter. Every 1.31 ms, the counter will change the output of the mux as well which seven segment LED is activated on the board. Once the mux output is selected by the counter, the output is passed into a BCD to seven segment encoder, which takes a BCD value and encodes it into CA-CG which are the cathodes in each segment of the seven segment LEDs. With this architecture, we will be able to see two seven segment displays that each display numbers 0-9 based on 8 total input switches at what appears to be the same time using multiplexing.



Since the Nexys-A7 FPGA comes with 8 common-anode 7-segment displays, it had to be understood how to display values to them. The data bus CA-CG controls which part of the LED turns on is shared by all of the displays. This means that displays that are enabled via the AN pin will display the CA-CG value. This is important when figuring out how to display the appropriate value for the appropriate display.



To display the correct value to the correct display, a counter was used to cycle through the 8 displays and to drive the appropriate CA-CG values. The counter acted as the refresh rate for the displays. Being too slow, the displays would update too slow and the cycling would be obvious to the human eye. A decent middle ground was using a refresh rate of 2^{17} clock cycles. With a 10ns clock cycle this means the displays would cycle between every 1.31 ms. With 8 displays this means the entire board refreshes every 10.48ms, or ~95 Hz.



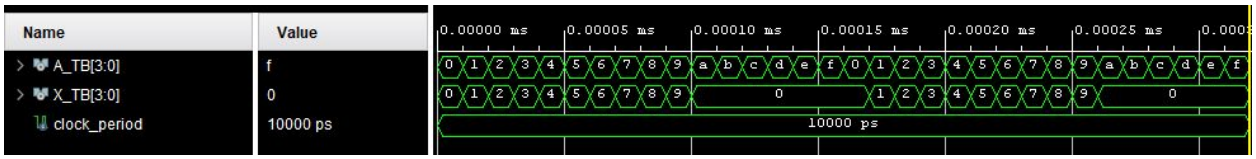
Code Details

Overall, the code, compared to previous labs, was simple and had no complicated tricks. One simple trick, multiplexing, was implemented in order to deceive the eyes into seeing both seven segment LEDs activated at the same time. With multiplexing, we are toggling each seven segment display on and off so fast that we cannot see it flickering. The displays used in the code rotate every 1.31ms.

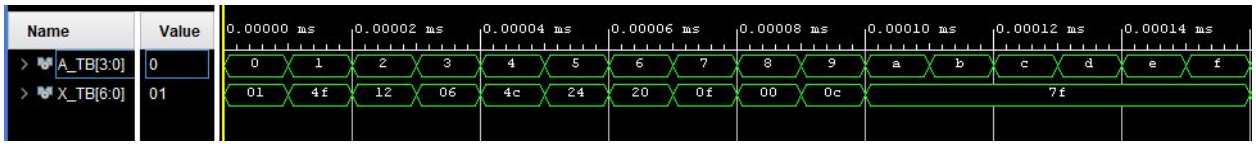
Since only displays 0 and 4 are used in the lab, it was simpler to focus the cases to only display values for them specifically. This meant that instead of writing a case for every display 0-7, it took up less lines to do a case for 0 and 4 only, leaving the rest as a default value.

Corner Cases

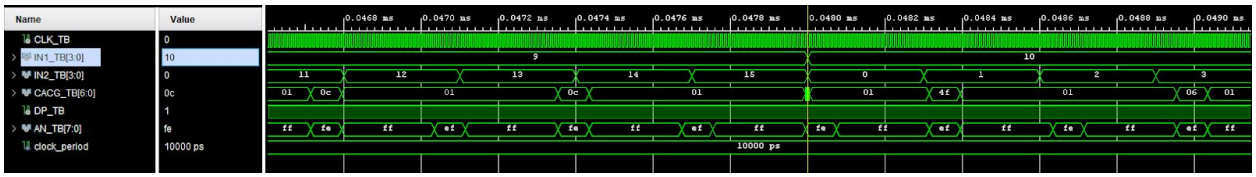
The main corner case to consider is what happens when the input was not a valid BCD digit. Since we are using 4 switch inputs, the values 10-15 would not be valid BCD and so a catch would have to be made for them.



In addition, the case and if statements needed to be checked to make sure that all cases were covered. This way, latches or errors would be prevented. For the BCD to 7-segment decoder, any value outside of the BCD range is displayed as 0.

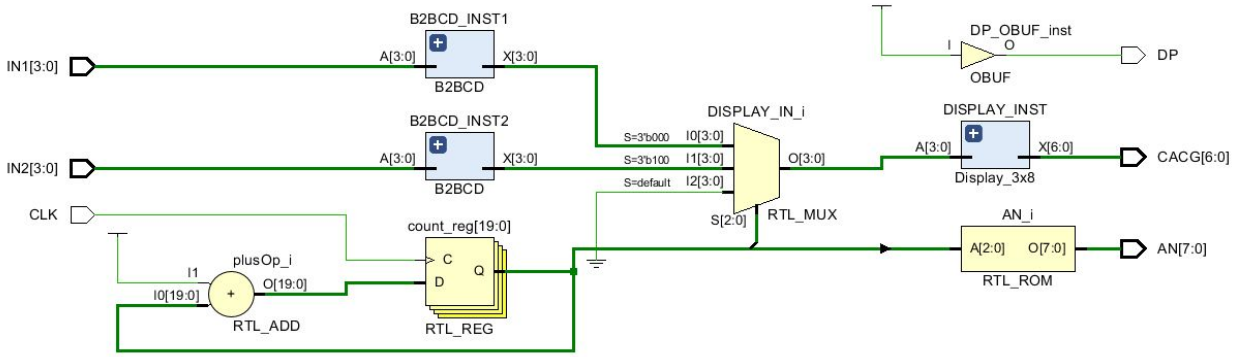


Final simulation of the entire system



Area/Resource Information

Elaborated Design of Entire Design



Resource Usage of Entire System

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP
synth_1	constrs_1	Synthesis Out-of-date								16	20	0.0	0	0
impl_1	constrs_1	Implementation Out-of-date	7.665	0.000	0.254	0.000	0.000	0.101	0	16	20	0.0	0	0

Power Usage Details

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.101 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.5°C**
Thermal Margin:	59.5°C (12.9 W)
Effective ϑJA:	4.6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity