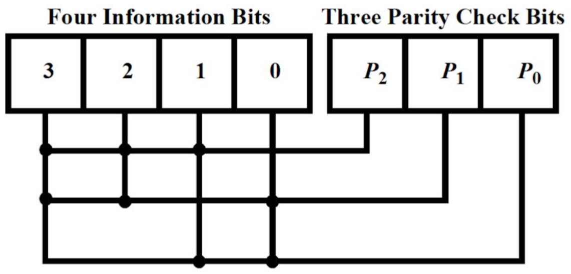
Justin Ngo
Digital Logic Design
Barry Johnson
16 October 2020

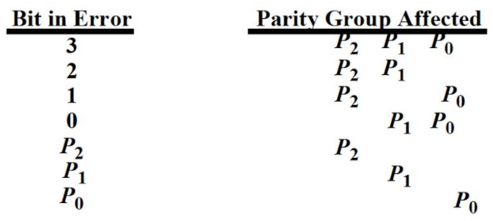
Laboratory Assignment 2

Problem Statement

The task for this laboratory assignment is to design an Error Detection and Correction (EDAC) unit. The input code word in the design was to read from memory 4 data bits and 3 parity check bits forming a 7-bit code word. Three 4-bit parity generators needed to be used to create the three required parity check bits. A gate-level design for the Syndrome Generator and the Corrector needed to be implemented in the design of the EDAC, as well as use of a Decoder.

Design Solution

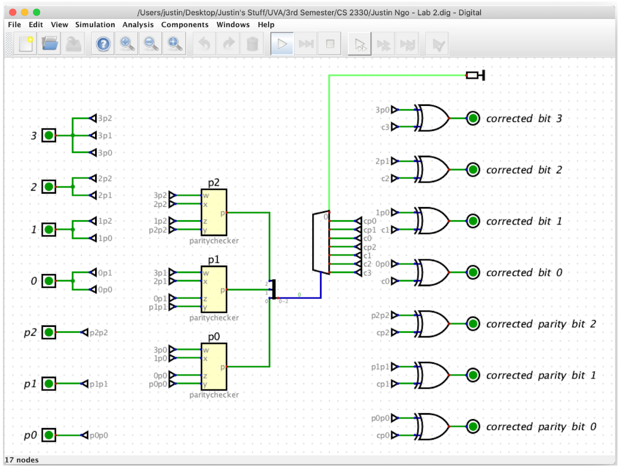


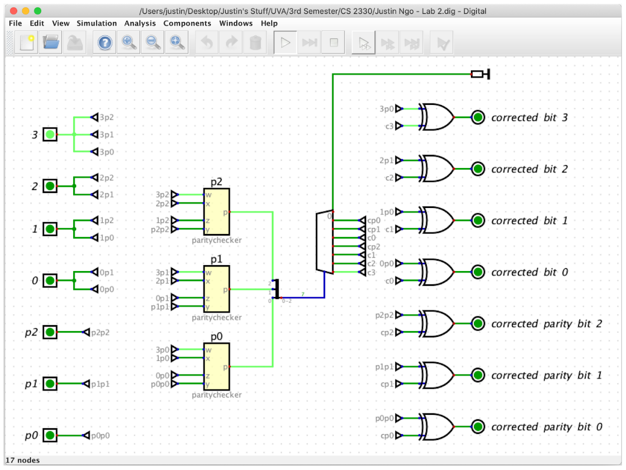


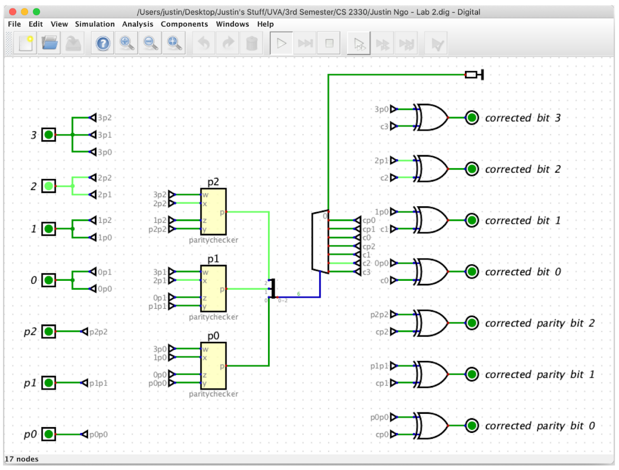
Bit in Error	Parity Group Affected
3	P_2 P_1 P_0
2	P_2 P_1
1	P_2 P_0
0	P_1 P_0
P_2	P_2
P_1	P_1
P_0	P_0

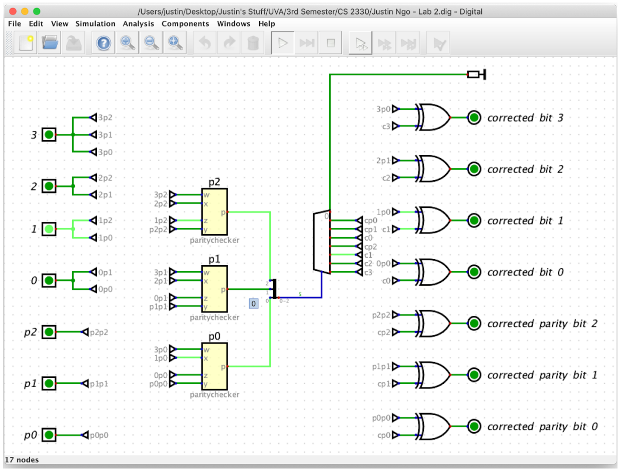
The design process of my EDAC was guided by the table above which demonstrates the concept of overlapping parity. The table organized the input bits into the parity groups it affected, revealing the parity checker that the bit in error needed to flow into. In addition, the table informs which specific output dot of the decoder to connect with the correct input bit in a XOR gate for the corrector, as well as the input bit that final output corrects. For example, bit 3 goes into all three parity checkers whereas bit 2 only goes into two of the parity checkers, P2 and P1. I reimplemented the 4-bit even parity code generator from laboratory assignment #1 and connected the outputs of each into the merger in order to build the syndrome generator. I then connected the single merged output from the merger to the decoder. There are eight red dots on the output of the decoder and they are counted starting at 0 and going up to 7. The 0 output dot of the decoder being triggered indicates that no bit is in error and thus in my design it flows into a pull down resistor. The 7th output dot of decoder being triggered represents that the bit 3 is in error and needs to be corrected because P2 P1 P0 turns into 1 1 1 in binary and therefore 7 in decimal. The 6th output dot of decoder being triggered represents that the bit 2 is in error and needs to be corrected because P2 P1 turns into 1 1 0 in binary and therefore 6 in decimal. The other outputs of the decoder were determined accordingly and XOR gates are used to correct the input bit in error because the output of XOR is 0 if any of the inputs is 1.

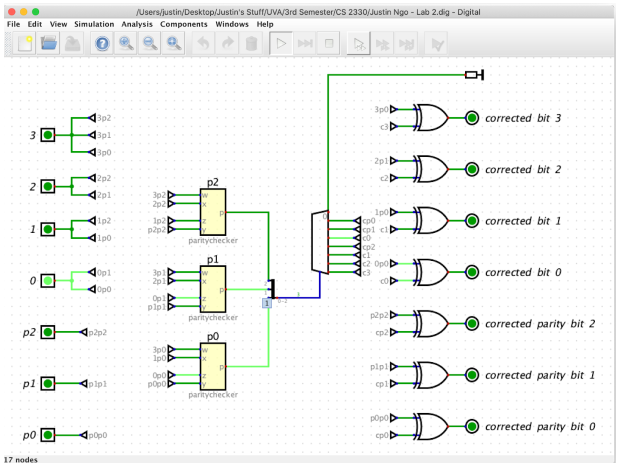
Test Results

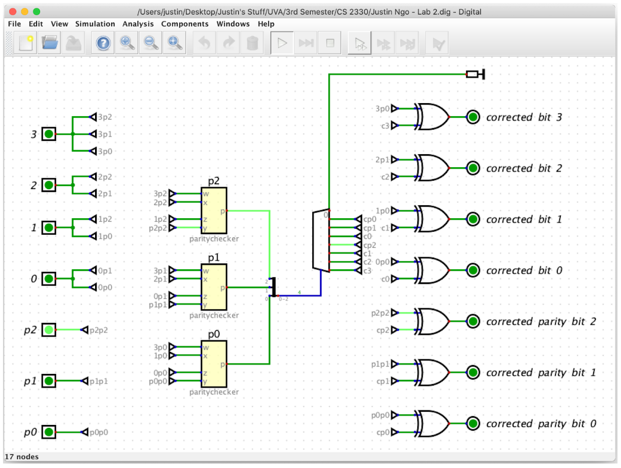


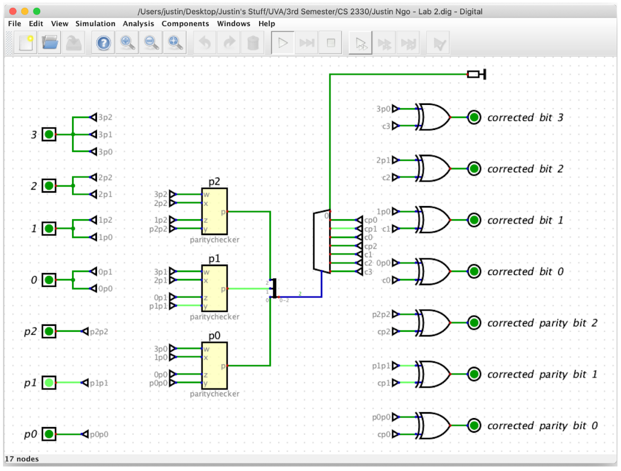


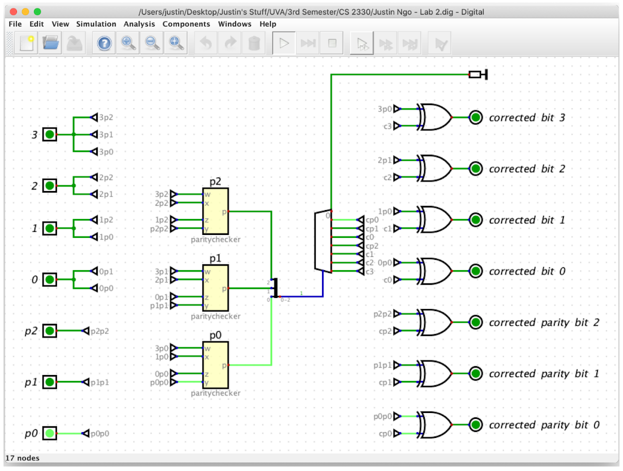












Summary and Conclusions

I began the laboratory assignment by researching and studying the different components of the EDAC from resources such as the online textbook on how the syndrome generator, merger, and decoder would be implemented in a circuit. I implemented the 4 data bits and 3 parity check bits forming a 7-bit code word as seven inputs. Following the given instructions "figure 2: concept of overlapping parity" table I organized and connected the input bits as tunnels into the three parity checkers. For example, when bit 3 is in error all three parity checkers p2, p1, and p0 would receive the input from bit 3. Instead of having one tunnel for each of the input bits that affected more than one parity group, I separated them into multiple as it made more logical sense to me in seeing how many and which of the parity checkers each input bit went into. I reimplemented the 4-bit even parity code generator from laboratory assignment #1 and connected the outputs of each into the merger in order to build the syndrome generator. I then connected the single merged output from the merger to the decoder. The output of the decoder would go into the pull-down resistor if none of the input bits were incorrect and would correct the incorrect input bits by comparing the output from the decoder to the original input using a XOR gate. The XOR gate takes in both inputs and corrects the bit accordingly. I finished the assignment by checking that the outputs of my circuit were corrected using the XOR gates and corresponding input bits. A challenge I faced in completing the assignment was that the parity groups affected in the given table did not translate directly to 7 6 5 4 3 2 1 in decimal after converting the parity bits to binary, but instead translated to 7 6 5 3 4 2 1 and thus had to be slightly readjusted in designing the final corrector of the circuit.