11/30/24
Timothy Ogg
VLSI 4334 Project 1 Report

For this project, I made my own implementation of the Quine-Mccluskey algorithm in python. The goal of this project was to design the algorithm in a way that takes in a PLA file, minimizes its logic using the Quine McClsukey algo. The output is the minimized PLA file. The sequence of the implementation is as follows: parse the PLA input, apply QM algorithm, and write the results minimized terms back into a new PLA file.

The core algorithm involves the following steps:
1.Grouping the minterms by hamming weight which is a minimization process in which the terms are grouped by the number of 1s in their binary form. In doing this the process is efficient for comparison and combination of terms that differ by one bit only.
2. Then combine minterms by iteratively merging pairs of terms that differ by one bit. In this process a “dont care” defined as “-” replaces the differing bit which results in a new term that covers more minterms. This is done until there are no more possible combinations.
3. We then generate the prime implicants after all possible combinations have been made thus resulting in the simplest form of the boolean function.

As far as challenges encountered, I initially had issues with trying to combine minterms with multiple differences. Initially there were terms with multiple differences incorrectly combined which resulted in an invalid output. The solution to this is to make a function called combine_terms that would check for more than one difference and return none if possible which would ensure that only valid combinations are made.

An additional challenge was properly grouping the minterms because of poor management of the multiple groupings. To solve this a dictionary was used to store the groups, with the key being the number of 1s in term and of course the value being a list of terms; the grouping. By sorting the groups by key there would be proper sorting by key before the processing hence the correct order.

As for the challenge with handling the writing of the pla file I struggled to place the .e directives. To fix this I had to omit the printing of the .e directive when I accessed a heading value that stored the directives from the input file. Once I print the minterms I print the .e directive.

To verify that my implementation worked I used an online Quine-McClusky Calculator.

At first I used multiple to verify that the calculator had no bugs but then I settled with the main one
being:https://atozmath.com/KMap.aspx?q=quine&q1=0%2c1%2c2%2c8%2c16%2c21%60%60a%2cb%2cc%2cd%2ce%601&do=1#PrevPart
Used a total of 10 test cases ranging in variable length up to 6 variables and determined that it was working 100%. Though further testing would need to be done for variables with greater range. All inputs and outputs are located in the Github repo. Change input(value).pla to run it and change the output(value).pla for the output file.

This project had its minor challenges but luckily there were enough online resources to help guide the way. The final result is a functional and limited QM algorithm implementation that generates a PLA file as output.