I. Requirements: Restate the problem specification and any detailed requirements in your own words.

The goal of this homework assignment was to create an executable file that would read in two text files, one describing a circuit and the other giving a list of initial events. Each set of files would contain wires, gates, and delays. The output of the executable should show which wires are high, low, or unknown until the end of the instructions. We decided to implement the optional GUI.

II. Design: How did you attack the problem? What choices did you make in your design, and why? Show class diagrams for more complex designs.

A picture containing text

Description automatically generatedThe main premise behind our implementation of the project was an event-driven queue system. We have the event queue which tells us how the circuit will change at a given time. Once we make that change, we can then create new events based on those changes. Our class diagram is below.

III. Security Analysis: State the potential security vulnerabilities of your design. How could these vulnerabilities be exploited by an adversary? What would be the impact if the vulnerability was exploited?

The one major thing that could go wrong with this program is the user inputting incorrect data. If that happens, the worst thing that could happen is a segmentation fault. We do not use unsafe memory operations so there aren’t too many ways that a user would be able to exploit our program.

IV. Implementation: Outline any interesting implementation details in your solution.

One of the most interesting aspects of our solution is the Circuit class. Although this was not originally in the specifications for the project, we felt that this would be a beneficial class to have, especially once we moved to the GUI. The Circuit class manages everything to do with the circuit, making it easy to use since all the functionality is centered in one place.

V. Testing: Explain how you tested your program, enumerating the tests if possible. Explain why your test set was sufficient to believe that the software is working properly, i.e., what were the range of errors for which you were testing.

We ran this program through a rigorous set of test cases. To start out with we used the circuit files that Dr. Shomper provided. Comparing the results of these files with the sample programs that were provided gave us strong confidence that the program was indeed working as intended. After that we created a few of our own test cases and ran those through, once again getting the expected results.

VI. Summary/Conclusion: Present your results. Did it work properly? Are there any limitations? NOTE: If it is an analysis-type project, this section may be significantly longer than for a simple implementation-type project.

After much trial and error, the program does indeed work correctly. We started out with the program as a simple text-based console program. The modularity that we built into the code allowed it to easily be transferred to a GUI program. In the transition to GUI, we also added a few features that were not present in the console program. One of the sorely lacking features of the console program was the ability to step through the program one event at a time. We added this feature and then also added a stellar set of decorations that make the program even more fun to use.

VII. Lessons Learned: List any lessons learned. For example, what might you have done differently if you were going to solve this problem again?

Throughout the course of making this program we learned quite a few things. Although this problem seemed very complex at the start, we broke it down into manageable pieces and then divided the work between the two of us. Breaking the problem down into smaller manageable chunks allowed us to work asynchronously through the problem. We also learned about priority queues, which are very useful for sorting data.