Logic Circuit Simulation

By: Mostafa Gaafar (900214463)
Ahmed Elkhodary (900213472)
Malek Mahmoud (900222057)

Department of Computer Science and Engineering

Course: Digital Design 1
Project 1 - *Report*
Supervised by: Dr. Mohamed Shalaan

SPRING 2024

1 Introduction

Logic circuits serve as fundamental components in today's world of modern technology. Through them it is possible to process and manipulate digital information, which is quite essential for a variety of tasks. For example, computer algorithms and data analysis. The digital revolution and developments in industries such as robots, telecommunications, and artificial intelligence would not take place without the existence of logic circuits. Therefore, The objective of this project is to develop a logic circuit simulator that accurately replicates actual logic circuits. This requires utilization of advanced tools such as data structures, algorithms, and a comprehensive understanding of logic circuits. We aim to create a platform that will serve to help users master digital logic circuits. Moreover, provide users with the ability to study the value and time delay at any given point in the circuit. Through this, users will be able to trace logic circuits and enhance their proficiency in dealing with digital logic design. Therefore, our project's goal to simulate logic circuits is crucial for unlocking innovation and efficiency in the field of technology.

2 Specifications of Data Structures

Storing the data was an integral part of our algorithm. Initially we had the idea to store them in areas however the problems with such implementation quickly disregarded such approach. After discussing this we agreed to use vectors extensively due to its dynamic property. We also used classes to read and store data from the 3 files (stim, cir, and lib). The classes provided a level of abstraction that kept everything separated which made it easier to troubleshoot and debug any problems that rose. The way we implemented the class is that it takes the 3 file locations and read everything by itself, storing the data along the way. We also used structs and created functions inside them to reduces the lines of code in the main and to simplify the code overall. Analyzing the data and extracting what we want from it was the hard part and it took place over in the main. Our implementation goes over the vector of logic gates, evaluating if any input was changed and if the output of such logic gate was different from its previous value, we push it to another vector that have the inputs. At the end we sort that vector

according to timeStamp. We have a corresponding vector called "changed" that keeps track of whether a change happened at that time stamp. And if it did, the program writes it into the output file. As for the algorithms we used merge sort and searching which will be discussed in depth later on in the report.

3 Data Specification:

Apart from the LogicGateExpressionEvaluator, we also prepared 5 test circuits which were inclusive of a wide variety of logic gates and number of inputs to test out programs on different cases and scenarios. We included OR, XOR, AND, NAND, and NOT each taking inputs varying from 1-3. However, our program can work with any unspecified circuit and any number of inputs as long as the logic expression for it is provided. We also produced corresponding sim files to check our output compared to the produced output

4 Specifications of Algorithms:

The LogicGateExpressionEvaluator class plays an essential role in the implementation of the logic circuit simulator. It serves as the backbone as its main target is to imitate a logic gate given the input and producing output. The input is given in a form of expression in the format of (i1& i2)—(ĩ1&i2) where & represents AND gate, — represents OR gate, and ĩepresents NOT gate. In this example, i1 and i2 are the variables. Each variable stores a boolean value. The approach we saw best appropriate was to take the input expression and convert it from infix to postfix expression by utilizing stacks. This implementation is seen as efficient and professional as it handles the string expression and the vector of boolean values well and translates it into a boolean output. In order to make our function compatible with any given test circuit, our logic gates can take as many inputs as the user wishes. This is implemented by using a vector of type boolean to store the value of the variables in the expression. The function precedence objective is to order the precedence of the operator. Moreover, the function isOperator returns true if it's given a character that is an operator. The function infixToPostfix is responsible for changing the input infix expression we are given to postfix.

Then the postfix expression is passed on to the function named evaluatePostfix, which implements the expression and evaluates its value. To ensure the function's validity we ran a test case written by chatgpt in the int main in Malek's Branch and it passed the test. Apart from the logic gate evaluator, we implemented merge sort with the help of chat gpt and changed it a bit so that it can accommodate the fact that whenever we swap values we swap the corresponding indexes at other vectors. We also used an instance of the search algorithm that searches for the logic gate name through the vector used to store data from lib files. At the end, our program creates a file called CircuitOutput.sim with timestamps and logic values of every component when it changed

5 Analysis and Critique:

Several issues roose from the fact that we were using vectors. One of the most prominent of them was out of range accessing which was repeated in multiple parts of the program. However with enough code revisions they were all fixed. Moreover, the implementation sometimes uses integers for truth values and sometimes use bool (according to which struct you are using) which also created some problems leading to standarizing the usage of only bool vectors and bool for truth values. One of the biggest problems with the code is efficiency as it runs multiple loops concurrently which explodes the complexity however, some measures were taken to reduce the complexity like terminating the loops when the required functionality was obtained from them. However, there is still room for improving the complexity especially when searching for something through the struct's vectors. An Idea we had was to remake the vectors into a hashtble which will dramatically improve the search time however it will use more space. Therefore, the implementation could be improved into using more space and less time.

6 Experimental Results:

6.1 Screen Output

The screen was used extensively for debugging and running the code will show a lot of data relating to how everything is stored, and how everything is evaluated, as long as multiple counters and bool that ensures the vectors doesnt go out of bound and the program doesnt try to access data beyond whats allowed. Several mistakes occurred in the code which were resolved after noticing abnormalities in the expected data displayed on screen

6.2 File Output

The final output is a file, a file called CircuitOutput.sim which includes the names, time stamps, and truth values for all the components everytime their truth values changed. This step was done only once we made sure everything works correctly.

6.3 Bonus

The implementation of the waveform visualization system will be implemented using JSON, with the additional assistance from LLMs such as Chat GPT. The system takes the input of 5 .sim files for the 5 test circuits, processes the data, converts it into JSON format using the assistance of the LLM, and then visualizes the waveform using the WaveDorm tool in a .SVG file format that can be visualized by web browsers, graphics softwares, and other applications.

Input Data:

The input data for the waveform visualization system consists of .sim files representing different test circuits. These .sim files contain timestamped data representing the states of signals over time. Each line in the .sim file consists of three components separated by commas: timestamp, signal name, and state.

Output Data:

For each circuit outputs 1 through 5, the x-axis of the waveform represents the time, measured in seconds. Also, the y-axis of the waveform represents the logic state of signals at different points in time. The logical states are either "0" or "1", with discrete transitions between these states indicated by vertical lines. Regarding the colors in a waveform shown above in all test circuit waveforms, they serve various purposes and provide meaningful information about different aspects of the signals being represented. They represent the logic levels, signal colors, as well as the signal state transitions. Screenshots of the output can be found in the Github repo.

7 Future Improvement:

- Automated .sim File Upload: Implement functionality to allow us to upload .sim files directly to the waveform visualization system without the need for manual conversion of the JSON code through LLM assistance. This enhancement aims to ease the process without the need to manually generate JSON code and then generate the waveform automatically.
- Support for Any .sim Circuit: Ensure that the system can handle any of the .sim circuits.
- Integration with External Tools: Consider integrating the waveform visualization system with external tools and platforms to extend its functionality and compatibility. Integration with simulation software can enhance the system's capabilities.
- Support reading from multiple files concurrently to simulate multiple circuits working at the same time

- Allow for multiple entries for each variable at the stim file to implement changing logic circuits with different inputs at different time stamps

8 Conclusion

In conclusion, this project's main objective is to provide the user with a platform which can simulate any logic circuit. Through this design, the user can study time delay and output values at any point the user wishes. As the project might have come short handed in some aspects such as complexity or handling outputs happening at the same time, it still sufficiently accomplished its goal. In addition, a lot of learning lessons have been transcended to the team by facing these blocks. The team does not only hope to simulate logic circuits, but it anticipates to continue to adapt and leverage new emerging technologies in order to utilize them as catalysts in order to build breakthroughs in this significant field.