Logic Circuit Simulator: Project Report

Group 1
Hoda Hussein
900223388
Adham Salem
900222465
Youssef Aboushady
900223372
Department of Computer Science and Engineering, The American University in Cairo
CSCE 2301
Dr. Mohamed Shalan
March 23, 2024

Logic Circuit Simulator: Project Report

Used Data Structures and Algorithms

Unordered map:

Maps were used to map the data allocated in the files, such as the library file and the circuit file. Furthermore, they were used to map circuit aspects like components, propagation delays, and logic gates. This is shown in the functions: “readLibraryFile”, “readCircuitFile”, “readSimulationFile”, and ”generateSimulationOutput”.

Unordered set:

The unordered set is used to store outputs that are dependent on previous outputs or inputs. It was used in the function “generateSimulationOutput” to check if an output or input of a gate is dependent on a previous output/input, so it can be used to process the total delay of the circuit. This can be seen in: “if (dependencies.count(component.input1) > 0 ||
dependencies.count(component.input2) > 0) { // Calculate delay for the current component considering dependencies

```
        timestamp += component.delayPs;
.
.
.”
```

Vector:

Instead of using arrays, we opted for vectors to store any processed data. It is easier to add the data and pass the vector to other functions and it is dynamic, so it can better suit any circuit with any given size of inputs and to be processed data. The vector is used throughout the code in every function like in the “readCiruitFile” function as it is used as a parameter of the function to store the inputs, etc...

Testing & Challenges

1. Interpreting parentheses, NAND, and NOR gates
2. C++ file handling
3. Program debugging
4. Bonus 1: GitHub actions for ci
5. Bonus 2: generating graphical output

For testing, we used the 5 testing circuits we constructed to check the validity of our code and to check for errors. We would meet on Zoom and occasionally at Campos to work on the code and debug it. We faced several problems when testing.

The first challenge was in the evaluation expression function. We tried to evaluate boolean expressions using a stack data structure but the function did not interpret gates when there was "()", i.e if we had a circuit ~(A^B) the program would only process A^B and neglect the gate ~. To overcome this challenge, we had to change the function that read and simulated the circuit file, so we had to make the function check every input we read from the file to avoid this problem. This negatively affected the adaptability of our code, as it became limited only to gates of only 2 inputs.

C++ is not very lenient when it comes to files' readings, so any white space or unexpected value would throw an error or cause inaccurate data. So, we had to define and use functions such as trim() and safe_stoi(), in order to ensure correct data handling.

After finalizing the code, we had to go through the debugging step and we used a special int main() code to pinpoint where is the error, i.e, the code showed the output from every function in the program so that we could know if a function needs to be corrected or not and we had to go back and correct the simulation function several times.
Also, there were challenges in debugging using Clion because we encountered segmentation faults and memory access violations due to the usage of complicated data structures such as maps of maps. To overcome this, we used vectors and unordered sets instead.

The Github actions were very hard to understand, as we were not familiar with it before. We had to watch Youtube tutorials, read documents on GitHub, and use chatGPT to identify the errors, however, we could not manage to overcome a fatal error in the action. The error was in building the automation and we still cannot figure out how to resolve this issue.

Another struggle was in generating a waveform for the main program. We tried to use C++ for this task, but it did not work properly. So we shifted our thinking to implement the function using Python. In generating graphical output, we had to use python because c++ isn't really optimal for graphing, and none of the team members are experienced in python, so we had to look for the syntax online and use AI softwares for help in the syntax and installation of python packages.

Member Contributions:

We often worked together on campus and through zoom meetings, but the individual contributions of each member are listed below.

Hoda:

- Graphical representation of waveforms
- Implementation of unordered maps and sets
- Debugging using Clion
- Logic expression evaluation
- Generate simulation function
- Report

Adham:

- GitHub CI
- Test circuit design
- Debugging using debugging function
- Reading from library and circuit files
- Report

Youssef:

- Test case diagrams and truth tables
- Debugging
- Report
- GitHub CI