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 be adapted to suit any circuit. The vector is used throughout the code in several functions. For example, in the “readCiruitFile” function, it is used as a function parameter to store the inputs.

Testing & Challenges

To test our simulator, we constructed five sample circuits to verify the validity of our code and check for errors. We would meet virtually on Zoom and occasionally on campus to work on and debug the code. 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 would not interpret gates when there were parentheses. If we entered an expression of the form ~(A^B), the program would only process A^B and neglect the inverter gate (~). To overcome this, we had to change the function that read and simulated the circuit file to make it check every input that was read from the file. This decreased the versatility of our code, as it became limited to gates of only two inputs.

C++ is not very flexible with files reading, so any white space or unexpected value would throw an error or make the data inaccurate. Consequently, we had to define and use functions such as trim() and safe_stoi() to ensure correct data handling.

After finalizing the code, we had to go through the debugging stage. We used a special int main() code to pinpoint the location of the error. The code displayed the output of every function in the program to help us determine which functions needed to be corrected. To complete this process, we had to go back and correct the simulation function several times. There were also challenges in using the CLion debugger; we encountered segmentation faults and memory access violations due to the usage of complicated data structures (e.g. maps of maps). As a result, we resorted to using vectors and unordered sets instead.

As this was our first time interacting with GitHub actions, we were not familiar with the feature. We had to watch YouTube tutorials, read documentations, and use ChatGPT to identify the errors. However, we could not overcome a fatal error in building the automation, preventing us from using this feature to its full potential.

Finally, the last major obstacle was generating a waveform for the output of our simulator. We tried to use C++, the language we were most comfortable with, for this task, but it did not work properly. Instead, we shifted our thinking to implement the function using Python. Python was a better fit because it is more optimized than C++ for graphing. None of our team members were experienced wiith the language, so we had to look for the syntax online and use AI software for help in the 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