Digital Design Report - Project 1

Project name: Simulation For Logic Circuits
Section: 02
Team members:

- Yousuf Badawy
- Jana Fadl
- Maha Shakshuki

• Used data structures and algorithms:

1- Classes for

A- Multi-Input gate.

B- XOR, NAND, AND, OR, NOT:

- Organizing data and accessing the logic of the gates towards the other files.

2- Vectors to store the gate input:

- Dynamic access to the inputs in the other functions, such as parsing and the main.

3- Inheritance and polymorphism.

4-Construcutres and destructures in order to initialize inputs vector with specified size.

5- Different types of functions (virtual, setters, getters, structs), also to evaluate gates.

6- Tuples in order to group the gates into a single unit and return multiple values.

7- Ifstream for the different types of files:

- Read files and generate their content in the code.

8- If statements, while loops, and expectation handler:

- To manage the functionality of the gates.

9- Vectors to the time stamp:

- To ensure that each gate has individual delay as specified in the files

10-Maps to access the value of the gate.

11- Functions to calculate the time stamp.

12- Sorting function

-Sorts tuples in ascending order based on timestamp (used for output sorting)

Note :

ChatGPT has contributed to developing some functions in the code.
The prompts were :
vector<tuple<string, string, string, vector<string>>> connections

• **Testing:**

Part 1:

We tested each of these operations separately to ensure that they work probably:

1- Testing for classes.

2- Testing for parsing library files.

3- Testing for parsing stimulates file.

4- Testing for circuit file.

Part 2:

We combined all of them to test, and we added the testing element for the time diagram.

• **Challenges:**

The team has faced various challenges regarding the work environment, harmony, and the project work. For the work environment, we tried to find a time that suited the 3 of us in order to brainstorm and start dividing the tasks. For the first task, each team member worked on a different part as well as for the other tasks. We were meeting via Google Meet.

The challenges regarding the project were that we came up with different ideas, but most of them were not visible. We thought of implementing the project with Verilog because it is designed for the purpose of logical circuits. Still, none of us had a strong background in using it, so the only visible solution was using C++.

C++ has various operations and features that allow us to implement the requirements. However, there were plenty of errors regarding the classes, and we solved them by adding keywords such as virtual and override. For parsing the files, we faced problems with the extension and the compiler. We solved them by making sure that both the code and the file have the same extension. Also, the online compiler didn't accept the online files (.cir and .stim), so we used Visual Studio, and it worked. We also faced some challenges regarding printing the time stamp, but we fixed it by fixing the functions and adding maps, tuples, sort, and vectors. Overall, we overcame the challenges and produced the final product.

• **The contributions of each member:**

Yousuf Badawy :

- Worked on crafting the circuits and some parts of the timing diagram (Phase 1)
- Designed the .lib . cir and .sim files(Phase 1)
- Designed the algorithm of the code(Phase 2)
- Tested Classes and circuit files and fixed the errors. (Phase 3)
- Designed the Parsing functions
- Designed the Gates functions
- Wrote the code(Phase 3)
- Worked on debugging the whole final version of the code. (Phase 3)
- Report (Phase 3)

Jana Fadl:

- The initial code to test circuits. (Phase 1)
- Contrbuited to the semi-fuctional code. (Phase 2)
- Worked on the classes of each gate type. (Phase 3)
- Tested the stimuli file and identified any errors that were found (Phase 3)
- Identified any errors in the parsing files (Phase 3)
-

Maha Shakshuki:

- Created the GitHub-repo and was responsible for it. (Phase 1)
- Drew the time diagram for the circuits in the first task (Phase 1)
- Contrbuited to the semi-fuctional code. (Phase 2)
- Tested Library files and fixed the errors (Phase 3)
- Tested circuit files and fixed the errors (Phase 3)
- Report (Phase 3)