Project 1

Logic Circuits Simulator

Group members:

Ahmed Farid Elaswar - 900211265
Mazen Ahmed Zaki - 900211593

Introduction

As the world advances in technology, the amount of technology we encounter increases day by day in addition to the competitors increasing day by day as well. Reason being, everyone is trying to own the most advanced piece of technology which gives them lots of opportunities and power especially with AI on the rise. However, in order for the engineers to build electronic devices, they have to design and test the digital circuitry placed into whatever device being built. This process needs to be time efficient and without the use of any expensive hardware so Logic Gates Simulators were created. In this project, our aim is to build our own simplified Logic Gate Simulator while learning the fundamentals of digital logic design along our journey.

How Data Structures and Algorithms Drive the Simulator

During our process of constructing the simulator, we were very precise on choosing the data structures and algorithms and they are key in allowing for a fast and efficient program. This section of the report focuses on the data structures used to make the simulator happen and the algorithms used in making its logic.

Data Structures

1. Classes

    a. **Component**
    The component class acts as a template for the simulator where it makes a component object for each line in the circuit library files **(.lib)** to define a gate type. Each object stores attributes of the gate such as its name, number of inputs, a boolean expression for its logic and its propagation delay.

    b. **Gate**
    The gate class serves as the building block of the simulated circuit where each object represents a specific logic gate type defined in the circuit library. It links a component object to its place within the circuit. Each gate object's attributes are the name, component, outVar and inputs.

c. **Variable**
   The variable class represents the wires, signals and nodes of the circuit. Each object contains a name, state and a list of gates.

2. Dictionaries

   a. **components**
      When analyzing the circuit file, the simulator uses a dictionary to look up the component object needed to create a specific gate.

   b. **variables**
      Stores the wires in the simulated circuit to track the state of every signal in the circuit and make sure updates work as supposed to.

Algorithms

1. Circuit Parsing **(getVars)**
   The algorithm parses the **(.lib)** file to create the gates based on the info provided in the file. It extracts the attributes for each component then creates instances of the component class.

2. Expression Evaluation **(evaluate)**
   The algorithm determines a gate's output where it matches the gate's name to the predefined ones allowing it to identify the type of logic the gate should perform. Based on the gate recognized, the correct Boolean logic is applied.

3. Event Scheduling and Propagation
   The algorithm allows the simulator to employ an event-driven approach for an efficient operation. When the state of variable to represent a signal changes, the update method is called. When the signal value is changed, it is compared to the previous value to change the variable object's state on whether a change is detected or not. The algorithm then sends information about the signal change to the output file **(.sim)** including the time of the change and the new signal value. The refresh function is called for each gate to check if the input change should trigger a new output for the gate to announce an update with the gate based on its propagation delay.

Challenges

1. **Debugging and Code Tracing**
   Although Python is well known for easily finding mistakes, it was difficult finding errors in this project. In this simulator, things occur all at once based on the events so we found difficulty keeping track of how signals changed and how that affected different parts of the circuit.

2. **Simulating Real-World Gate Behavior**
   Making sure the gates worked like real circuits were very important as gates are what make this simulator work. For example, an AND gate only outputs a 1 if all inputs are 1s. Therefore, the program had to have the same output based on the inputs given to make the gates work as in real life.

3. **Code Organization**
   As more features were added to the program and the amount of files increased, it was difficult keeping it organized so it looked messy and hard to understand what was written. This gave us a challenge when fixing bugs or adding features.

4. **Visualization (Graphing)**
   Adding a feature that would graphically represent the simulation results in the form of a waveform was something new for all of us which made it quite a challenge learning about it and implementing it for the first time. We had to research and integrate a python library to graph where we used Matplotlib. The development required a deep understanding of the project to be able to imagine how we would integrate the results in a graph which was very tricky and time consuming. Nevertheless, we were able to integrate it in time and we are confident in its results.

Contributions of Each Member

- **Mazen:** Responsible for the group project report ensuring its structure, clarity and alignment with project objectives. Mazen took the lead in coordinating the group's

work, maintaining smooth communication between group members and ensuring the completion of the project.

- **Ahmed:** Ahmed was responsible for designing and implementing the core codebase of the logic circuit simulator. In addition, he developed the simulator's graphing allowing for visualization of the simulation results.

ChatGPT prompts

- What is GitHub actions for CI?
- How to plot a digital signal in python
- How to compare string with regular expression with exact matching in python
- What is the python function that returns true when the list of arguments are all true
- How to declare a global variable in python
- How to remove the extra spaces from a string in python
- How to remove the line breaks from a line in python
- How does the xor gate behave when several inputs are passed to it
- How to pass arguments by reference in python

Project Meetings

Meeting #1
Date: March 8, 2024
Time: 11:00 PM

Finally we had our first meeting to discuss the group project at 11pm where we were able to cover the main project requirements to organize our tasks and set each member of the group a role. We all have assigned roles for the project and they are as following:

- **Mazen:** Project report, testing assistance (with Yehia), coding support if needed and researching GitHub CIs for a potential bonus grade.
- **Ahmed:** In charge of mainly all the Python coding.
- **Yehia:** Software testing all the code that Ahmed makes and coding support if needed as well.

This task breakdown allows for simultaneous work, maximizing our efficiency.