ECE 6140 Project 1 Logic Simulator

Author Name: Zhao Anjie
GT ID: 903334981

Data Structure & Implementation

In this project, I choose Python as the programming language and the data structure used is Dictionary. How to run this program is written in the readme file. Dictionary is similar to the Map in C++/Java, which is the combination of the key-value pair. Following is the general description of this data structure.

Each key is followed by a colon (:) and after that is the associated value. Each key-value pair is separated by commas and the whole pairs set is enclosed in curly braces. Keys are unique in a dictionary but the values can be not. The values of a dictionary can be in any type while keys should be in an immutable type.

Following is a sample of dictionary:

dict = {'FirstName': 'Anjie', 'LastName': 'Zhao', 'Age': 23}

In this project, as each line/wire number of the circuit is unique, it is used as the key in the dictionary. The related value is the logic value of that line. So the dictionary is the collection of all the output of lines in the circuit. When input comes, find the corresponded key in the line dictionary and assign value to it. Then the program will work based on the algorithm and finally pop the output vector that is needed.

Using *.readlines()* to read the line one by one and then using *.split()* to split the line into gate name + input line number + output line number. Then using the input line number as the key to find its value in the dictionary. After calculation done by the algorithm, the output vector will be popped by searching in the dictionary. The details of the algorithm will be discussed in the next part.

Pseudo code

```
For each line in the circuit file:
        //if output already has a value,
        //may indicate there exists stuck-at
        //faults in the circuit, so do nothing
        //with that value, because input value
        //with not change the output stuck-at
        //fault value
        if gate name is INV:
                if output wire has a value already, do nothing
                else if output wire value is none, do inverter logic
                write value to the corresponded key in the circuit dictionary
        if gate name is BUF:
                if output wire has a value already, do nothing
                else if output wire value is none, do buffer logic
                write value to the corresponded key in the circuit dictionary
        if gate name is AND:
                if output wire has a value already, do nothing
                else if one of input values is 0, output is 0
                else if both of two inputs have values, do AND logic
                write value to the corresponded key in the circuit dictionary
        if gate name is NAND:
                if output wire has a value already, do nothing
                else if one of input values is 0, output is 1
                else if both of two inputs have values, do NAND logic
                write value to the corresponded key in the circuit dictionary
        if gate name is OR:
                if output wire has a value already, do nothing
                else if one of input values is 1, output is 1
                else if both of two inputs have values, do OR logic
                write value to the corresponded key in the circuit dictionary
        if gate name is NOR:
                if output wire has a value already, do nothing
                else if one of input values is 1, output is 0
                else if both of two inputs have values, do NOR logic
                write value to the corresponded key in the circuit dictionary

        if all of the required output nodes have values, end
```

Circuit Testcase Vector Table

s27.txt

Input	Output
1110101	1001
0001010	0100
1010101	1001
0110111	0001
1010001	1001

s298f_2.txt

Input	Output
10101010101010101	00000010101000111000
01011110000000111	00000000011000001000
11111000001111000	00000000001111010010
11100001110001100	00000000100100100101
01111011110000000	11111011110000101101

s344f_2.txt

Input	Output
10101010101010101111111	10101010101010101010101101
01011110000000111000000	00011110000000100001111100
11111000001111000111111	00011100000111011000111010
11100001110001100000000	00001101111001111111000010
01111011110000000111111	10011101111000001001000100

s349f_2.txt

Input	Output
101010101010101011111111	101010101010101101010101
010111100000001110000000	000111100000001010111110000
111110000011110001111111	000111000001110100011111100
111000011100011000000000	000011011110011100100011111
011110111100000001111111	100111011110000010100000100