EE6094
CAD for VLSI Design
Programming Assignment 1 Report

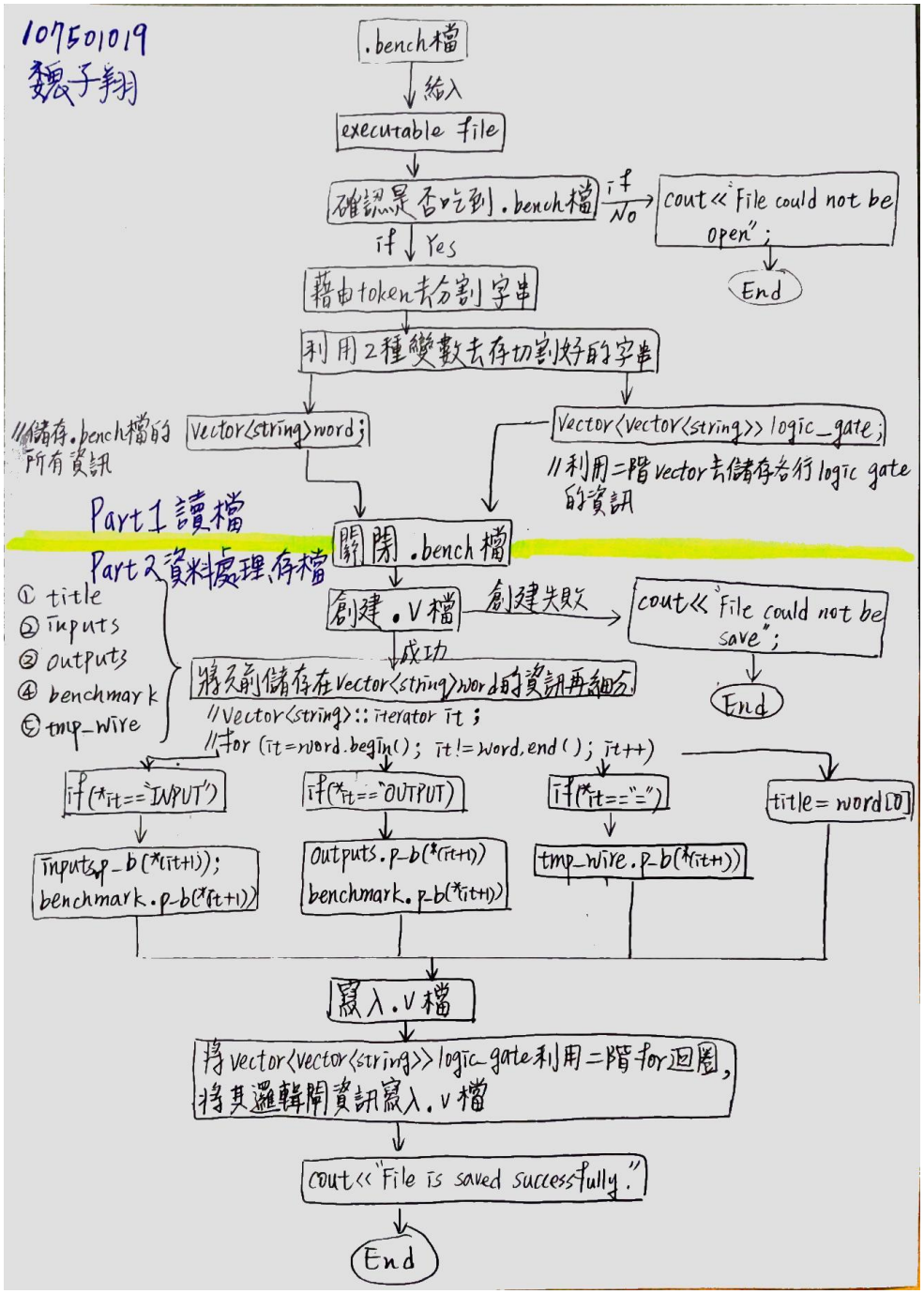
Student Name:魏子翔

Student ID:107501019

I. Problem Description

Verilog generator 在 EDA tool 中，是一項有利的工具，可以將 benchmark 自動地轉換成 verilog 形式，以便之後的 logic systhesis 及 verification。而本次作業就是要利用 C 或 C++寫出一個 verilog generator，之後將給定一個 bench 檔，藉由 verilog generator 讀取後，將其資訊整理後，生成出一個 verilog 檔。

II. Program Structure



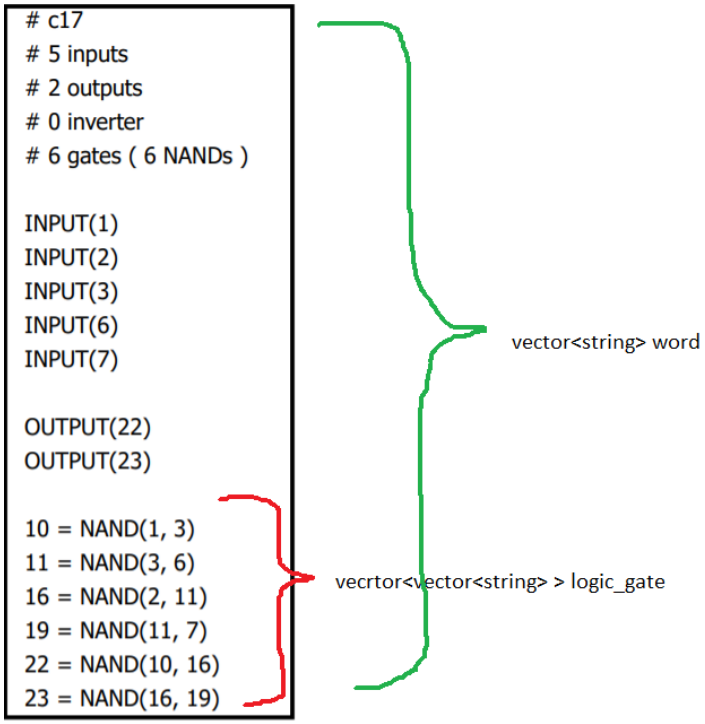
III. Data Structures / Algorithms Used

PART 1 讀取.bench：

根據” ,#()”字元去做檔案的切割

vector<string> word 儲存.bench 裡所有的字串

vecrtor<vector<string> > logic_gate 儲存邏輯閘電路資訊



PART 2 資料處理及輸出：

再將讀取後的 word 去做細分成

```
string title = word [0];
vector<int> inputs, outputs, benchmark, wire, tmp_wire;
```

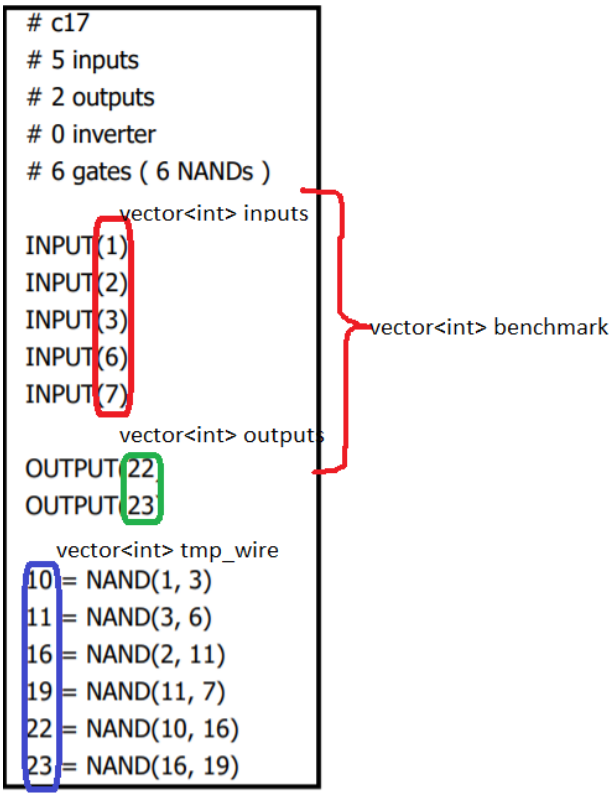
然後依條件去 push_back

```
        if(*it == "INPUT")
        {
            int tmp_num;
            tmp_num = atoi((*(it+1)).c_str());
            inputs.push_back(tmp_num);
            benchmark.push_back(tmp_num);
        }

        if(*it == "OUTPUT")
        {
```

```
            int tmp_num;
            tmp_num = atoi((*(it+1)).c_str());
            outputs.push_back(tmp_num);
            benchmark.push_back(tmp_num);
        }

        if(*it == "=")
        {
            int tmp_num;
            tmp_num = atoi((*(it-1)).c_str());
            tmp_wire.push_back(tmp_num);
        }
```



再將 tmp_wire 與 outputs 去做比對，將真正的 wire 找出來

IV. Difficulty Encountered

我在這次作業遇到的難點就是，一開始是用自己電腦的 VScode 編譯及 command line 去執行讀檔及存檔，在此存檔的結果都很順利的進行，輸出都是正確的，但是當丟到工作站後，存檔及輸出全部都亂掉了，因此花了很多時間去網路找資料，發現 windows 和 linux 吃檔案的方式不同，windows 每行的結尾不會有\n，而 linux 則有，因此整個程式碼的邏輯必須得花上一段時間去 debug。

後來 3 月 9 日課程結束後，助教提醒了要把.bench 檔把 Windows format 改成 Linux format，改完之後，原本的 code 編譯出來的執行檔可以順利的在工作站生成.v 檔。

V. How to execute

在工作站輸入 PA1_107501019.exe cXX.bench cXX.v

```
ifstream ifile(argv[1], ios::in);    //開檔
```

開啟 cXX.bench 檔案

```
while (getline(ifile, line))
```

利用 getline()將每行的資訊讀出來

```
    vector<string> tmp_gate;     //暫時儲存邏輯閘每行的資訊
        char *p;         //分隔後的字串 token
        p = strtok(lineCharArray, cut);
        // 將資料加入 vector
        while (p)
        {
            if(line_cnt == 3)   //line_cnt == 3 時，代表已經讀到 3 個空行，
                                 接下來會進入邏輯閘的資訊
                tmp_gate.push_back(p);

                word.push_back(p);
                p = strtok(NULL, cut);
        }

        if(line_cnt == 3)
            logic_gate.push_back(tmp_gate); //將 tmp_gate 丟進二階 vector

        if(line.size() == 0)    //遇到空行，line_cnt++
            line_cnt++;
```

並利用 token 的方式去將每行切割，並儲存於 vector<string>word;中，而 line_cnt 是去檢查是否已經讀到邏輯電路的資訊，因為在讀到邏輯電路資訊前，一定會有 3 個空行(可以參考 III. Data Structures / Algorithms Used 中，.bench 的圖片)，因此當 line_cnt=3 時，就要開始進入邏輯電路的部分，並藉由 tmp_gate 將其儲存於 vecrtor<vector<string> > logic_gate

```
 ofstream ofile(argv[argc-1], ios::out); //存檔
```

建立 cXX.v 檔

```
    vector<string>::iterator it;
        for(it = word.begin(); it != word.end(); it++)  //利用 for 迴圈去讀 word 的字串
        {
            if(*it == "INPUT")  //儲存 inputs
            {
                int tmp_num;
                tmp_num = atoi((*(it+1)).c_str());  //string 轉 int
                inputs.push_back(tmp_num);
                benchmark.push_back(tmp_num);
            }

            if(*it == "OUTPUT") //儲存 outputs
            {
                int tmp_num;
                tmp_num = atoi((*(it+1)).c_str());
                outputs.push_back(tmp_num);
                benchmark.push_back(tmp_num);
            }

            if(*it == "=")  //儲存可能是 wire 的資訊
            {
                int tmp_num;
                tmp_num = atoi((*(it-1)).c_str());
                tmp_wire.push_back(tmp_num);
            }

        }
```

接著將存於 word 中的字串，依上方 3 個條件整理到對應的變數裡。

開始輸出 module、input、output 於 cXX.v 檔中。

```
    for(int i = 0; i<tmp_wire.size(); i++)  //將 tmp_wire 與 outputs 比較，將真正的 wire 存起來
        {
```

```
            int save_wire = 0;
            for(int j = 0; j<outputs.size(); j++)
            {
                if(tmp_wire[i] == outputs[j])
                {
                    save_wire = 1;
                }
            }
            if(save_wire == 0)
                wire.push_back(tmp_wire[i]);
        }
```

因為存於 tmp_wire 中的東西並非全然是真正的 wire，因此利用上方的程式碼將真正的 wire 找出。

輸出 wire 於 cXX.v 檔。

將存於 vecrtor<vector<string> > logic_gate 中的電路資訊存到 cXX.v 檔。

VI. Experimental Results

```
[107501019@eda359_forclass ~/hw1]$ g++ hw1.cpp -o hw1_test.exe
[107501019@eda359_forclass ~/hw1]$ ./hw1_test.exe c17.bench c17_test.v
File is saved successfully..
```

生成的.v 檔的資訊：

```
1  `timescale 1ns/1ps
2
3  module c17 (N1,N2,N3,N6,N7,N22,N23);
4
5  input N1,N2,N3,N6,N7;
6
7  output N22,N23;
8
9  wire N10,N11,N16,N19;
10
11 nand NAND2_1 (N10, N1, N3);
12 nand NAND2_2 (N11, N3, N6);
13 nand NAND2_3 (N16, N2, N11);
14 nand NAND2_4 (N19, N11, N7);
15 nand NAND2_5 (N22, N10, N16);
16 nand NAND2_6 (N23, N16, N19);
17
18 endmodule
```

將生成的.v 檔丟入 testbench 驗證：

```
ncsim> source /usr/cad/cadence/INCISIV/cur/tools/inca/files/ncsimrc
ncsim> run
 input pattern = 00000 --> golden value = 00
 your answer = 00
You're correct1
 input pattern = 10101 --> golden value = 11
 your answer = 11
You're correct2
 input pattern = 01010 --> golden value = 11
 your answer = 11
You're correct3
 input pattern = 11011 --> golden value = 11
 your answer = 11
You're correct4
 input pattern = 11111 --> golden value = 10
 your answer = 10
You're correct5
ncsim: *W,RNQUIE: Simulation is complete.
ncsim> exit
[107501019@eda359_forclass ~/hw1]$
```

驗證結果顯示正確!!!

VII. Reference

1. C++ How to programing
2. 大一計算機概論 李龍豪教授 自編上課講義