Parser document

1031436 2A 呂昕恩

特殊資料結構：

1. 以 struct 存每個 id 資訊以及 id 之間的關聯，struck 內包含的：
   String id：gate 的 output $signal_id
   string type：logic gate type
   Gate* input1：每個 gate 的 input1 $signal_id pointer
   Gate* input2：每個 gate 的 input2 $signal_id pointer
   int value：logical gate 的值，初始化狀態為-1

gate 分三種存取狀況：

   1. input / output：將 input / output 視為 logic gate 的特例，僅使用 id、type、value，其餘狀態設為 NULL
   2. Buff、NOT：用到 id 、type、value 與 input1，input2 設為 NULL
   3. logic gate：剩餘所有的 logic gate，使用到所有 struct 結構

2. 利用兩個 vector 陣列，將 input / output 特別存取，以供後面計算方便

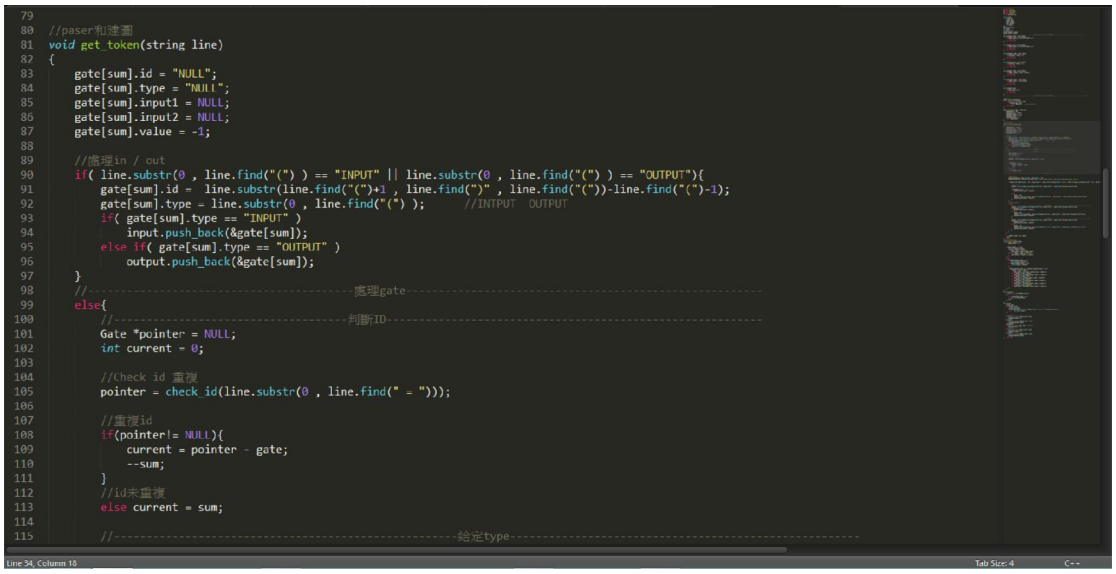
Functions：

1 Logical gate function：定義每個 gate 的功能，於計算時呼叫

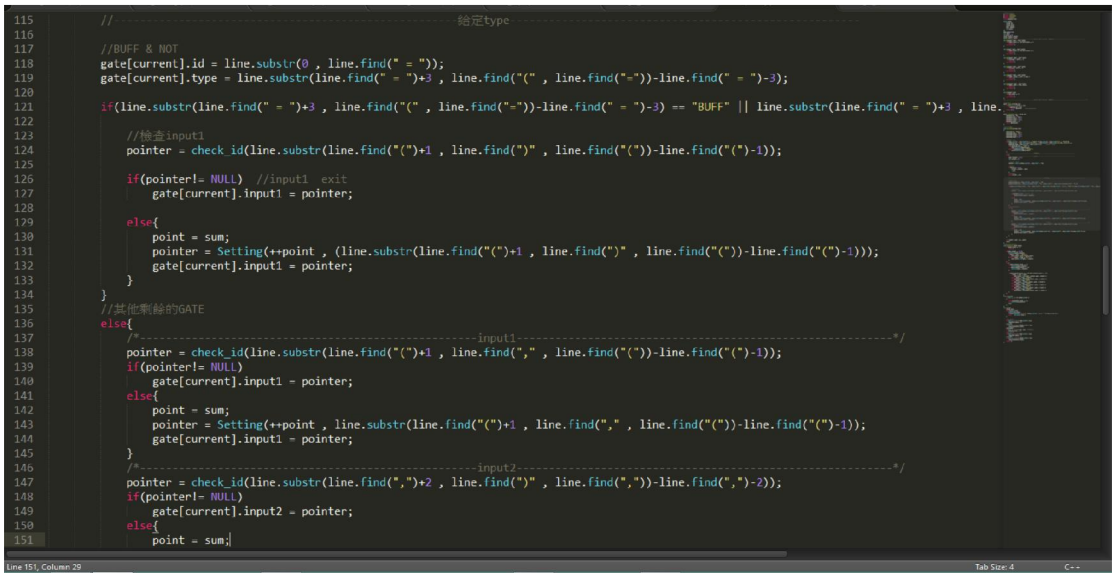
```
//-----------------------------------------Logical gate function，定義Gate-----------------------------------
int AND(Gate* right , Gate* left){
    if( right->value == 1 && left->value == 1)
        return 1;
    else return 0;
}

int OR(Gate* right , Gate* left){
    if( right->value == 0 && left->value == 0)
        return 0;
    else return 1;
}

int NAND(Gate* right , Gate* left){
    if( AND(right , left) == 1)
        return 0;
    else return 1;
}

int NOR(Gate* right , Gate* left){
    if( OR(right , left) == 0)
        return 1;
    else return 0;
}

int XOR(Gate* right , Gate* left){
    if( right ->value == left -> value )
        return 0;
    else return 1;
}

```

```
int NXOR(Gate* right , Gate* left){
    if( right->value == left->value)
        return 1;
    else return 0;
}

int NOT(Gate* val){
    if( val->value == 0)
        return 1;
    else return 0;
}
/*-----------------------------------------Logical gate function，定義Gate-----------------------------------
```

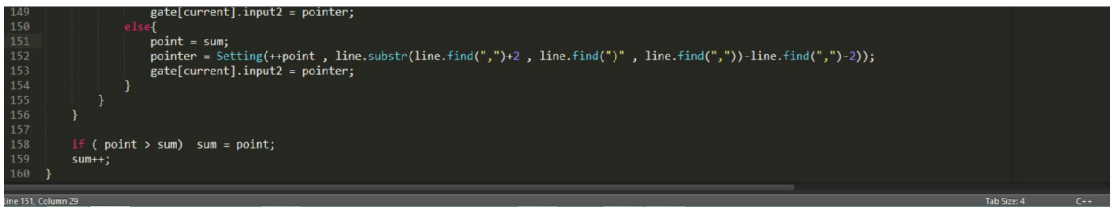
2 get_token()：在這個 function 中，主要進行電路圖的 paser，並且將切出來的 token 建構成圖。首先處理視為特例的 input/output，接著處理單一 input 的 gate，最後處理兩個 input 的 logic gate



```
79
80  //paser和建圖
81  void get_token(string line)
82  {
83      gate[sum].id = "NULL";
84      gate[sum].type = "NULL";
85      gate[sum].input1 = NULL;
86      gate[sum].input2 = NULL;
87      gate[sum].value = -1;
88
89      //處理in / out
90      if( line.substr(0 , line.find("(") ) == "INPUT" || line.substr(0 , line.find("(") ) == "OUTPUT"){
91          gate[sum].id =  line.substr(line.find("(")+1 , line.find(")" , line.find("("))-line.find("(")-1);
92          gate[sum].type = line.substr(0 , line.find("(") );      //INTPUT  OUTPUT
93          if( gate[sum].type == "INPUT" )
94              input.push_back(&gate[sum]);
95          else if( gate[sum].type == "OUTPUT" )
96              output.push_back(&gate[sum]);
97      }
98      //-------------------------------------處理gate-----------------------------------------------------
99      else{
100         //-------------------------------判斷ID-------------------------------------------------------
101         Gate *pointer = NULL;
102         int current = 0;
103
104         //Check id 重複
105         pointer = check_id(line.substr(0 , line.find(" = ")));
106
107         //重複id
108         if(pointer!= NULL){
109             current = pointer - gate;
110             --sum;
111         }
112         //id未重複
113         else current = sum;
114
115         //-------------------------------------------------給定type-----------------------------------------
Line 54, Column 18                                                                    Tab Size: 4    C++
```



```
115         //-------------------------------------------------給定type-----------------------------------------
116
117         //BUFF & NOT
118         gate[current].id = line.substr(0 , line.find(" = "));
119         gate[current].type = line.substr(line.find(" = ")+3 , line.find("(" , line.find("="))-line.find(" = ")-3);
120
121         if(line.substr(line.find(" = ")+3 , line.find("(" , line.find("="))-line.find(" = ")-3) == "BUFF" || line.substr(line.find(" = ")+3 , line.
122
123             //檢查input1
124             pointer = check_id(line.substr(line.find("(")+1 , line.find(")" , line.find("("))-line.find("(")-1));
125
126             if(pointer!= NULL)  //input1  exit
127                 gate[current].input1 = pointer;
128
129             else{
130                 point = sum;
131                 pointer = Setting(++point , (line.substr(line.find("(")+1 , line.find(")" , line.find("("))-line.find("(")-1));
132                 gate[current].input1 = pointer;
133             }
134         }
135         //其他剩餘的GATE
136         else{
137             /*-------------------------------------------------input1-------------------------------------------------*/
138             pointer = check_id(line.substr(line.find("(")+1 , line.find("," , line.find("("))-line.find("(")-1));
139             if(pointer!= NULL)
140                 gate[current].input1 = pointer;
141             else{
142                 point = sum;
143                 pointer = Setting(++point , line.substr(line.find("(")+1 , line.find("," , line.find("("))-line.find("(")-1));
144                 gate[current].input1 = pointer;
145             }
146             /*-------------------------------------------------input2-------------------------------------------------*/
147             pointer = check_id(line.substr(line.find(",")+2 , line.find(")" , line.find(","))-line.find(",")-2));
148             if(pointer!= NULL)
149                 gate[current].input2 = pointer;
150             else{
151                 point = sum;
Line 151, Column 29                                                                   Tab Size: 4    C++
```



```
149                 gate[current].input2 = pointer;
150             else{
151                 point = sum;
152                 pointer = Setting(++point , line.substr(line.find(",")+2 , line.find(")" , line.find(","))-line.find(",")-2));
153                 gate[current].input2 = pointer;
154             }
155         }
156     }
157
158     if ( point > sum)  sum = point;
159     sum++;
160 }
Line 151, Column 29                                                                   Tab Size: 4    C++
```

2.1 check_id()：因為在 parser 過程中同時建圖，而圖中只會有一個 id。為保證每個 id 都只代表一個 node，此 function 用來檢查 id 是否已經存在，保證 id(node)不重複

```
63  //new GATE
64  Gate* check_id(string id){
65      for(int i = 0 ; i<sum+1 ; i++)
66          if(gate[i].id == id)
67              return &gate[i];    //存在id直接指過去
68      return NULL;
69  }
70
```

2.2 Setting()：當使用 check_id()進行檢查時，若 id 存在則 return 該 id 在 gate[]的位置，若 id 不存在則呼叫 Setting()創建新 id(node)

```
Gate* Setting(int pos , string id){
    gate[pos].id = id;
    gate[pos].type = "NULL";
    gate[pos].input1 = NULL;
    gate[pos].input2 = NULL;
    gate[pos].value = -1;
    return &gate[pos];
}
```

3 BackTrack ()：以遞迴方式，從 output 開始回溯到到最底層(即 input)，開始呼叫 Logical gate function 計算每個 gate 的 output 值

```
//遞迴回溯計算output
void BackTrack (Gate* gate){
    if (gate->value != -1)
        return ;

    if(gate->input2 == NULL){
        BackTrack(gate->input1);
        if(gate -> type == "BUFF") {
            gate->value = gate->input1->value;
        } else if(gate -> type == "NOT") {
            gate->value = NOT(gate -> input1);
        }
    }
    else{
        if(gate->input1->value == -1)
            BackTrack(gate->input1);
        if (gate->input2->value == -1){
            BackTrack(gate ->input2);
        }

        if(gate->input1->value != -1 && gate->input2->value != -1){
            if(gate -> type =="AND")
                gate->value = AND( gate ->input1 ,gate ->input2 );
            else if(gate -> type =="NAND")
                gate->value = NAND(gate->input1 ,gate -> input2 );
            else if(gate -> type =="OR")
                gate->value = OR(gate->input1 ,gate -> input2 );
            else if(gate -> type =="NOR")
                gate->value = NOR(gate->input1 ,gate -> input2 );
            else if(gate -> type =="XOR")
                gate->value = XOR(gate->input1 ,gate -> input2 );
            else if(gate -> type =="NXOR")
                gate->value = NXOR(gate->input1 ,gate -> input2 );
        }
    }
}
```

3.1 answer()：用來傳送 output 給 BackTrack();

```
void answer(){
    for(int i = 0 ; i< output.size(); )
    {
        while(output[i]->value == -1)
            BackTrack(output[i]);
        i++;
    }
}
```

4 maim()：進行電路圖輸入，送入 get_token()進行 parser，並賦予 input 初始值，進行 answer()計算 output 結果。

```
int main(){
    string line;
    while(!cin.eof()){
        getline(cin,line);
        if(line.substr(0 , 1) =="#" ||line.substr(0 , 1) == " " || line.size() <3) ;
        else    get_token( line );
    }

    //第一組測資
    for(int i = 0 ; i < input.size() ; i++)
        input[i]->value = 0;
    answer();
    for(int i = 0 ; i < output.size() ; i++)
        cout<<output[i]->value;
    cout<<endl;
    for(int i = 0 ; i < sum ; i++)  //清空gate
        gate[i].value = -1;
    //第二組測資
    for(int i = 0 ; i < input.size() ; i++)
        input[i]->value = 1;
    answer();
    for(int i = 0 ; i < output.size() ; i++)
        cout<<output[i]->value;
    return 0;
}
```

OUTPUT

```
[root@wristband ~]# ./a.out < design_02.isc
11111011011101000000101000110110
00001110101001110111000110101111[root@wristband ~]# ./a.out < design_01.isc
00111001011110000011011001110110
10011010000111000110100010111000[root@wristband ~]# ./a.out < design_00.isc
00
01[root@wristband ~]#
```