ICCAD 2019 CAD Contest Problem D
Logic Synthesis Using Programmable Logic Gates

葉咸辰
Electrical Engineering
National Taiwan University
b05901178@ntu.edu.tw

***摘要*—面對給定的三種複雜的 universal gates，針對只有這三個 universal gates 所組成的元件庫(library)，使用 ABC 和 C++來邏輯合成，且不使用 assign 以及其他的邏輯閘。合成的最佳目標為合成後的電路和原先合成前的電路功能相同且盡可能降低 cost = area *timing 的值。**

Keywords—universal gates, ABC, C++, 邏輯合成, cost, area, timing

I. Introduction

A. Problem Statement

給定三種複雜的 universal gates，針對只有這三個 universal gates 所組成的元件庫(library)，開發邏輯合成程式，且不能使用 assign 以及其他的邏輯閘。三種 universal gates 的 area cost 都為 1， timing cost 也都為 1。合成後的電路，area 為所使用的邏輯閘總數，而 timing 則為從輸入端 到輸出端的最長路徑上的邏輯閘數目，也就是 longest path 上的邏輯閘數目。合成時的最佳化目標為降低 cost = area * timing 的值。

gate1:

4-to-1 MUX, Output=$\overline{S_0 S_1 I_3 + S_0 \bar{S_1} I_2 + \bar{S_0} S_1 I_1 + \bar{S_0} \bar{S_1} I_0}$

gate2:

Actel's ACT 1 logic module, Output=$\overline{(I_0 \bar{S_0} + I_1 S_0)(\bar{S_2} + \bar{S_3}) + (I_2 \bar{S_1} + I_3 S_1)(S_2 + S_3)}$

gate3:

H-bridge function, Output=$\overline{I_1(I_2 + I_4 I_5) + I_3(I_4 + I_2 I_5)}$

Fig. 1. 三種 universal gates

B. Inputs

輸入格式將採用 LGSynth91 的 Verilog 檔格式，以 assign 方式描述輸入電路，所有的輸入檔案都是組合邏輯電路，沒有序向電路。另外需要輸入三種 universal gates 所組成的元件庫

(1) Verilog 檔格式之輸入檔，範例如下:

```
module fa(a, b, c, sum, carry);
input
  a,
  b,
  c;
output
  sum,
  carry;
wire
  \[1] ;
assign
  \[1]   = (~b & a) | (b & ~a),
  sum = (~\[1] & c) | (\[1] & ~c),
  carry= (a & b) | (b & c) | (a & c);
endmodule
```

Fig. 2. 輸入格式

(2) Verilog 檔格式之元件庫，範例如下:

```
module gate1(s0, s1, i0, i1, i2, i3, o);
input
  s0,
  s1,
  i0,
  i1,
  i2,
  i3;
output
  o;
assign
  o = ~(( s0 & s1 & i3) | (s0 & ~s1 & i2) | (~s0 & s1 & i1 ) | (~s0 & ~s1 & i0));
endmodule

module gate2(s0, s1, s2, s3, i0, i1, i2, i3, o);
input
  s0,
  s1,
  s2,
  s3,
  i0,
  i1,
  i2,
  i3;
output
  o;
assign
  o = ~((( i0 & ~s0) | (i1 & s0) & ~(s2 | s3)) | ((((i2 & ~s1) | (i3 & s1)) & (s2 | s3)));
endmodule

module gate3(i1, i2, i3, i4, i5, o);
input
  i1,
  i2,
  i3,
  i4,
  i5;
output
  o;
assign
  o = ~((i1 & (i2 | (i4 & i5))) | (i3 & (i4 | (i2 & i5))));
endmodule
```

Fig. 3. 元件庫

C. Outputs

輸出檔案為 Verilog 檔格式， 輸入輸出和輸入檔設定相同，但是是由 gate1，gate2， 或是 gate3 所組成的結構化模型(structural modeling)方法。

```
module fa(a, b, c, sum, carry);
input
  a,
  b,
  c;
output
  sum,
  carry;
wire
  ab ;
  gate1 g0(1'b0, 1'b0, a, 1'b0, 1'b0, 1'b0, ab);
  gate1 g1(b, c, ab, a, a, ab, sum);
  gate1 g2(b, c, 1'b1, ab, ab, 1'b0, carry);
endmodule
```

Fig. 4. 輸出格式

D. Requirements

提示符號下執行 "time your_program_name –i in.v –l lib.v –o out.v"，其中 in.v，lib.v，out.v 不是固定的名稱，是要能被置換成其他的檔名。

E. *Grading Criteria*

1) 正確性: 合成後的電路和原先合成前的電路功能需要相同，我們將會驗證功能是否一致(functional equivalence)。

2) 程式需要在競賽單位指定的機器內，不能執行超過兩小時，超過兩個小時將強迫 終止，如果終止後仍有結果，將執行驗證作業並列入正常的成績計算。

3) 所有 test case 裡(包含給定與隱藏的 test cases)，功能正確數目最多的獲勝；當功 能正確的數目相同時，則比所有功能正確的 test case(s)之加總的 cost，最低者獲勝；當 cost 仍然相同時，則比加總的執行時間，越少者獲勝。

II. SOLUTION 1 : ABC

報告的時候才發現別組都是用 ABC，所以我也改用 ABC[註一]來做，確實方便許多。

A. *Download ABC*

從網路上下載 ABC 的檔案和 code(Github)[註二]，以建立可使用 ABC 的 Linux 環境。

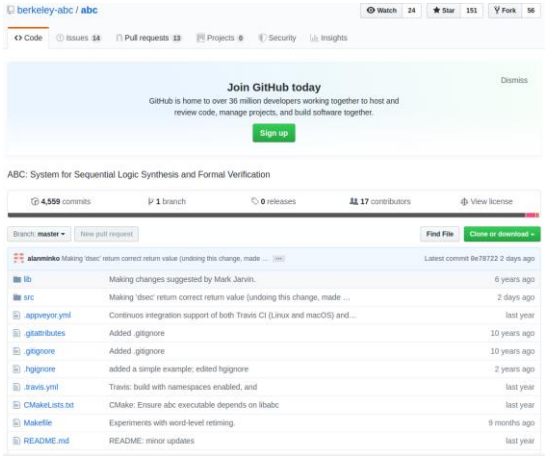

Fig. 5. Github for ABC code

B. *Compile ABC for Technology Mapping*

針對給定的三種 universal gates，建立一個 standard cell library，檔名為 mcnc.genlib[註三]。裡面使用的邏輯閘的 area cost 都改成以三種 universal gates 表示後的 area cost。在 compile ABC 的介面，read 要輸入的 Verilog 檔格式和 library 後，便進行 map，以得到以 library 裡所有的邏輯閘組成的電路。最後再 write 來得到輸出的 Verilog 檔格式。

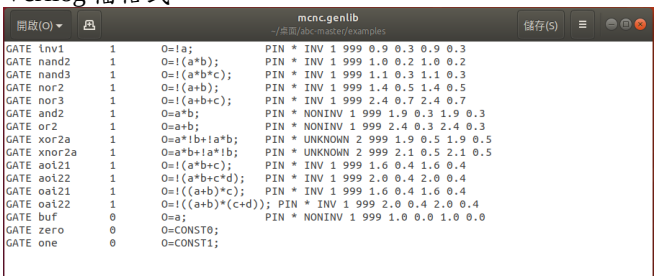

Fig. 6. mcnc.genlib

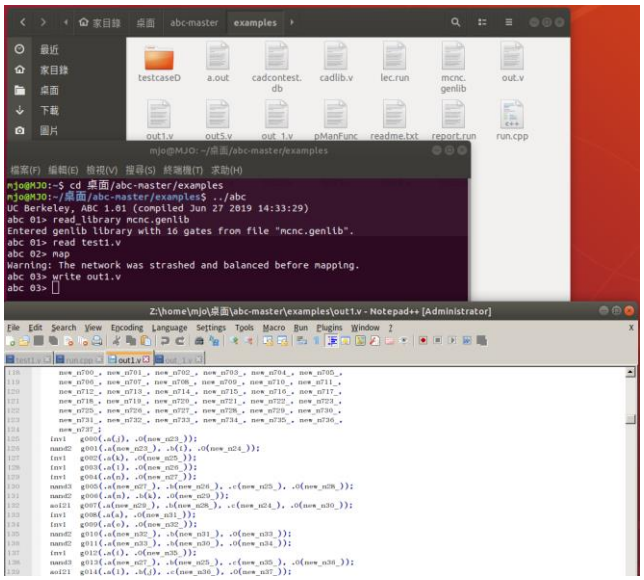

Fig. 7. Compile ABC for Technology Mapping

C. *Transfer to Gate 1, 2, 3 in C++*

由於輸出的 Verilog 檔格式是以 mcnc.genlib 裡的邏輯閘組成，所以用 C++來將邏輯閘表示轉換成 Gate 1, 2, 3 的表示方式。

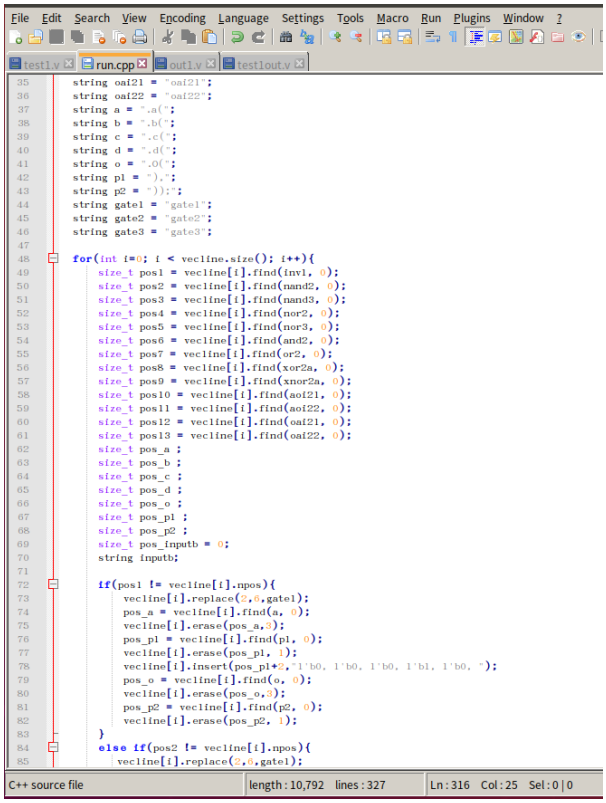

Fig. 8. Transfer to Gate 1, 2, 3 in C++

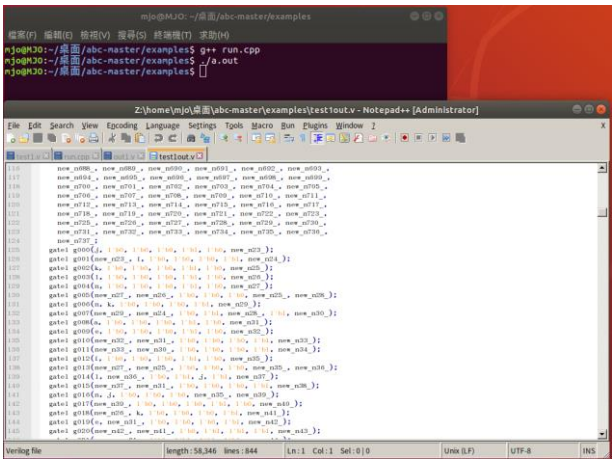

Fig. 9. Compile and After

D. Problem

1) 輸入指令

由於這個方法要先使用 ABC 來 compile，再轉回用 C++，所以對於輸入指令要執行 “time your_program_name –i in.v –l lib.v –o out.v”的部分我不知道要怎麼處理。

2) 檢驗正確性和 Cost

由於這個部分主辦方給的方式是用 cadlence lec 去檢查正確性和 synopsys dc_shell 去回報 area 和 timing，而我在使用 ubuntu 的情況下不知怎麼下載，所以沒有去檢驗。

3) 邏輯閘

由於時間關係，我確定能 Transfer to Gate 1, 2, 3 的邏輯閘偏少，之後可以增加一些去降低 Cost。

III. Solution 2 : C++

我在忘記能用 ABC 的情形下是先用 C++去硬爆。

A. Read In and Write Out

用 ifstream 和 ofstream 來讀入和寫出。

```
ifstream in("test1.v");
if (in.is_open())
{
  while ( getline (in,line) )
  {
      vecline.push_back(line);
  }
  in.close();
}
else cout << "Unable to open input file";
```

```
ofstream out("out.v");
if (out.is_open())
{
  for(int i=0; i < assign.size(); i++){
      out << assign2[i] << endl;
  }
  out.close();
}
else cout << "Unable to open output file";
```

Fig. 10. ifstream 和 ofstream

B. Divide

將讀入的檔案分成 input、output、wire、assign 4 種 vector<string>，再將 assign 分成等號左右 2 種 vector<string>，儲存同時將空白和尾端的逗號、分號去掉，以方便進行邏輯運算和 Technology Mapping。

```
for(int i=0; i < vecline.size(); i++){
  if(vecline[i] == "input")
      for(int j=i+1; j < vecline.size(); j++){
          if(vecline[j] != "output"){
              vecline[j] = removeSpaces(vecline[j]);
              vecline[j].pop_back();
              input.push_back(vecline[j]);
          }
          else break;
      }
  else if(vecline[i] == "output")
      for(int j=i+1; j < vecline.size(); j++){
          if(vecline[j] != "wire"){
              vecline[j] = removeSpaces(vecline[j]);
              vecline[j].pop_back();
              output.push_back(vecline[j]);
          }
          else break;
      }
```

```
  else if(vecline[i] == "wire")
      for(int j=i+1; j < vecline.size(); j++){
          if(vecline[j] != "assign"){
              vecline[j] = removeSpaces(vecline[j]);
              vecline[j].pop_back();
              wire.push_back(vecline[j]);
          }
          else break;
      }
  else if(vecline[i] == "assign")
      for(int j=i+1; j < vecline.size(); j++){
          if(vecline[j] != "endmodule"){
              vecline[j] = removeSpaces(vecline[j]);
              vecline[j].pop_back();
              assign.push_back(vecline[j]);
          }
          else break;
      }
}
```

Fig. 11. Devide

```
string removeSpaces(string str)
{
    str.erase(remove_if(str.begin(), str.end(), ::isspace), str.end());
    return str;
}
```

Fig. 12. 去空白

C. Logic Operation and Technology Mapping

我的想法是檢視每一行 assign 等號右邊的 vector<string>，若裡面出現的只有 input 和已邏輯運算後的 wire，便對它進行邏輯運算，結果是由哪些邏輯閘組成會存入另一個 vector<string>，也是透過此 vector<string>來得知此 wire 是否有邏輯運算過。因為 assign 並不是照順序排好，所以會用雙層 for loop 去做類似 sort，就是內層檢驗使否出現的只有 input 和已邏輯運算後的 wire，對它進行邏輯運算後去跟最前面有包含尚未做邏輯運算的 wire 做 swap，並每 swap 一次就 count+=1，最外層就是從 i = 0，然後 i+ = count 一直上去。

運算方面則是存取括號、邏輯符號、input 和 wire 的位置，在運算並記錄相對應的邏輯閘使用。

```
int count = 0;
for(int i=0; i < assign.size(); i+=count){
  for(int j=0; j < assign.size(); j++){
      string positions_w;
      string positions_p1;
      string positions_p2;
      string positions_n;
      string positions_a;
      string positions_o;
      string positions_t;
      vector<string> wire2;
      vector<string> input1;
      for(int k=0; k < wire.size(); k++){
          size_type pos = assign2[j].find(wire[k], 0);
          int unmatch = 0;
          while(pos != assign2[j].npos){
              if(wire1[k] == "0"){
                  unmatch+=1;
                  break;
              }
              else{
                  positions_w.pushpack(pos);
                  pos = assign2[j].find(wire[k], pos+1);
              }
              wire2.push_back(wire[k]);
              positions_w.pushpack("break");
          }
          if(unmatch != 0)
              break;
      }
```

```
      if(unmatch == 0){
          pos = assign2[j].find("(", 0);
          while(pos != assign2[j].npos){
              positions_p1.pushpack(pos);
              pos = assign2[j].find("(", pos+1);
          }
          pos = assign2[j].find(")", 0);
          while(pos != assign2[j].npos){
              positions_p2.pushpack(pos);
              pos = assign2[j].find(")", pos+1);
          }
          pos = assign2[j].find("~", 0);
          while(pos != assign2[j].npos){
              positions_n.pushpack(pos);
              pos = assign2[j].find("~", pos+1);
          }
```

Fig. 13. Logic Operation and Technology Mapping

D. Transfer to Gate 1, 2, 3

在上述的 Technology Mapping 完成後，將記錄下來邏輯閘組成轉換成由 Gate 1, 2, 3 的表示方式。並配合最後的輸出檔案。

E. Problem

1) 輸入指令

由於時間關係，所以對於輸入指令要執行 “time your_program_name –i in.v –l lib.v –o out.v”的部分我還沒處理。

2) 邏輯運算

由於某些 assign 會出現由一堆括號、邏輯符號、input 和 wire 組成，因此在做邏輯運算會使用很多迴圈，可能會很沒效率。

因為時間關係，我還沒完成整個檢驗及運算過程。

3) 邏輯閘

由於這個部份我是在做前就先做，所以我做的邏輯閘偏少也有幾個好像用不太到。同時還沒 debug 過也不知有沒有錯。

IV. Conclusion

ABC 真的是一個好工具，能幫忙做 Technology Mapping，就不用去用 C++去硬爆，不過用 ABC 的話輸入指令會比較難完成。

A. 輸入指令

不論是哪種方法我都還沒對輸入指令去做設定，以後再補完成。

B. 檢驗正確性和 Cost

這部分我還要稍微研究一下，暫時無法解決。

因為還沒檢驗正確性的關係，我不確定也沒辦法 po 出我是否成功及我的 Cost 是多少。

REFERENCES

[註一]
https://www.dropbox.com/s/qrl9svlf0ylxy8p/ABC_GettingStarted.pdf
[註二] https://github.com/berkeley-abc/abc
[註三] http://www.ecs.umass.edu/ece/labs/vlsicad/ece667/links/abc.html
http://www.ecs.umass.edu/ece/labs/vlsicad/ece667/links/mcnc.genlib