ICCAD 2019 CAD

Contest

Problem D: Logic Synthesis using Programmable

Logic Gates

Taiwan Semiconductor Research Institute (TSRI), NARL

Contents

0. Announcement

October

- 2018-10--

August

- 2018-08-14- The FAQ of ProblemD is updated.
- 2018-08-02- The Beta Report of ProblemD is updated.

July

- 2018-07-09- The FAQ of ProblemD is updated.
- 2018-07-08- The Alpha Report of ProblemD is updated.

June

- 2018-06-19- The testcaseDv1 of ProblemD is updated.
- 2018-06-19- ProblemD description and FAQ are updated.
- 2018-06-18- The FAQ of ProblemD is updated.
- 2018-06-11- The FAQ of ProblemD is updated.

May

- 2018-05-24- The FAQ of ProblemD is updated.

April

- 2019-04-09- The FAQ of ProblemD is updated.
- 2019-04-08- The FAQ of ProblemD is updated.

March

- 2019-03-26- The FAQ of ProblemD is updated.
- 2019-03-19- The FAQ of ProblemD is updated.
- 2019-03-06- ProblemD updated.
- 2019-03-05- ProblemD updated.

Fabruary

- 2019-02-27- ProblemD updated.
- 2019-02-01- ProblemD announced.

Logic Synthesis using Programmable Logic Gates

Taiwan Semiconductor Research Institute (TSRI), NARL

1. Introduction

場域可程式化閘陣列 (Field Programmable Gate Array，FPGA)是透過預先建立的可程式化邏輯電路與路由(routing)電路，讓使用者可以藉由設定功能與連線的機制，設定邏輯電路模組的功能以及相互間的連線，以組合成所需要的電路。因為可以重新設定功能與可以直接在設計場域使用，如實驗室或研發單位，故能即時調整電路的效能與功能，以加速商品上市時間。因此，FPGA 被廣泛地使用於商品與雛型品(prototyping)驗證、仿真(emulation)驗證等應用。FPGA 是可程式化的邏輯電路，組合電路可以使用 FPGA 內之 RAM 區塊來做查表(LUT)而完成，也就是說將輸入當成記憶體區的位址，而預計得到的邏輯值則為記憶體的內容。例如位址 00 填 1，位址 01，10，及 11 都填 0，則此記憶體區塊即成為 NOR 閘之邏輯功能，位址的輸入則為 NOR 閘的輸入，記憶體區塊的輸出則為 NOR 閘的輸出。而另一種作法就是使用較複雜的 universal gate 來實現所需的邏輯，利用保險絲燒斷或是沒燒斷的機制來設定可程式邏輯的輸入訊號以及輸出連接的方式。本題目將針對複雜的 universal gates 開發邏輯合成的程式，將給定的邏輯函數合成以 universal gates 來實現的電路，並同時依據給定的 cost function 進行最佳化。

2. Problem Statement

本題目將給定三種複雜的 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，也就是將給定的邏輯函數轉換成只利用這三種 universal gates 來實現，而不能使用 assign 以及其他的邏輯閘。這三種 universal gates 的輸入也不能使用任何邏輯運算。為了簡化問題，三種 universal gates 的 area cost 都為 1，timing cost 也都為 1。參賽者需要讀入待合成的邏輯函數，以及三個 universal gates 的功能，並將此邏輯函數合成出只使用此三種 universal gates 的電路。

這三種 universal gates 都只有一個輸出，功能分別如下:

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=$\overline{(I_0 \bar{S_0} + I_1 S_0)\overline{(S_2 + S_3)} + (I_2 \bar{S_1} + I_3 S_1)(S_2 + S_3)}$

gate3:

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

題目固定使用這三種 universal gates，參賽者需要將給定的組合電路以此三種 universal gates 來實現，可以單獨選用一種或兩種，不需要全部的邏輯閘都使用，但也可以三種邏輯閘都使用。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。

以圖 1 的一個全加法器(full adder)為例，輸入為 A，B，C，輸出為 Sum，Carry，如果選擇所給定的 gate1 (4-to-1 MUX)來實現，則需要三個邏輯閘，所以 area =3，timing = 2，cost = area × timing = 6。

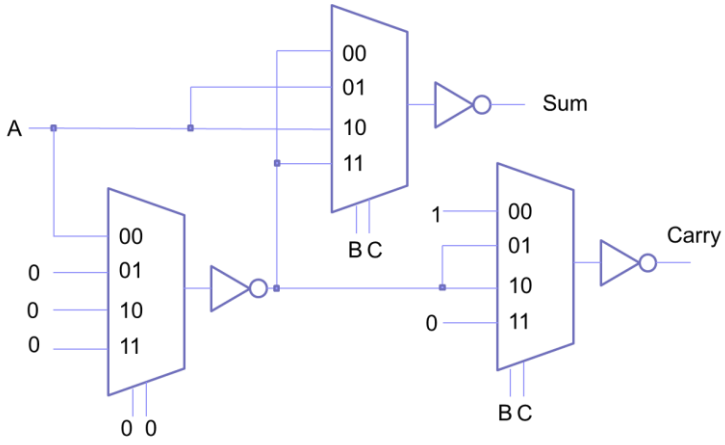

圖 1: 以 4-to-1 MUX 來實現之全加法器。

3. 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，兩個檔案格式與範例如下:

(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
```

(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
```

4. Requirements

需要提供 readme.txt 以說明環境設定與執行方法等資訊，而編譯好的程式需放在提供的目錄最上層，於提示符號下執行 "**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，也就是說，in.v 可以自由換成 in1.v，in2.v，…等，out.v 也可以換成 out1.v，out2.v，…等。其

中 in.v 是需要被合成的輸入檔，lib.v 則是三個 universal gates 的 Verilog 檔，而輸出 out.v 則是合成後以 gate1，gate2，或是 gate3 組合成的電路(netlist)檔。

5. 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 連結方式可以利用 connection by name 方式，需要連接宣告的名稱。另外，也可以利用 connection by order 方式，利用 module 宣告的順序連接，兩種方式都可以使用。底下範例是使用 connection by order 方式，將圖 1 的合成結果展現出來。注意，合成後的檔案不能有 assign 指令，也不能有其他邏輯閘存在，全部都只能利用 gate1，gate2，或是 gate3 組合而成。

```
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
```

6. Grading Criteria

- 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)。
- 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。
- 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。

7. Reference

[1] Logic Synthesis and Optimization Benchmarks User Guide Version 3.0.

[2] Actel ACT 1 Series FPGAs data sheet.

[3] Shun-Wen Cheng, “Configurable CMOS H-tree Logic Module,” Proc. of 2009 IEEE Int’l Conf. on Field-Programmable Technology (FPT’09), Sydney, Australia, Dec. 9-11, 2009, pp. 431-434.

VIII. Alpha Report

Rank	test1	test2	test3	test4	test5	test6	test7	Totoal	Time	Function Work
1	5863	4437	8310	10644	57400	11374	19899	117927	31.859	7
2	8952	4788	13520	12744	82812	13232	25018	161066	41874.04	7
3	9775	5568	16065	15181	105672	21252	32955	206468	12.56	7
4	7791	4396	11712	11056	73749	21056	0	129760	158.76	6

IX. Beta Report

Rank	test1	test2	test3	test4	test5	test6	test7	Totoal	Time	Function Work
1	3630	2366	4944	5190	31808	6800	11081	65819	1715.782	7
2	4788	3296	7028	8376	49960	8000	13264	94712	8505.558	7
3	6240	4466	9820	15100	48175	12404	19844	116049	39406.98	7
4	5678	3600	8322	9240	71421	10855	19140	128256	42737.92	7
5	12168	3744	9792	12960	51127	14940	35175	139906	1.507	7

X. FAQ

Q1. 請問題目給定的 test case 是否會出現以下的運算式/符：

Arithmetic Operators
Relational Operators
Bit-wise Operators
Reduction Operators
Shift Operators
Concatenation Operator
Replication Operator
Conditional Operator

A1. 為了簡化題目將重點放在邏輯合成演算法，故輸入的運算是只會包含 Bit-wise Operators，如題目所列輸入檔案的範例所示。

Q2. 請問 Problem D 中，題目給定的 test case 中，有可能會將 output 或是 wire，assign 成常數值嗎？

A2. Test case 內不會出現常數值或常數值的邏輯運算。

Q3. 請問所有的訊號 input、output、wire 是否都是單一個 bit 的訊號呢？

A3. 隱藏的測資會和公告的測資一樣，輸入、輸出與連線都是單一位元。

Q4. 在一組 bitwise 運算中，是否會出現在其中某部分的運算再經過 negation 的狀況，
例如以下運算
b&~(a|c)
又或者全部的 negation operator 都只會出現在一個訊號前面
b&(~a&~c)

A4. 邏輯運算的部分將不會限制某種格式，只要符合 verilog 表示格式，同一個布林函數，可能有因交換律、結合律、分配律等不同的表示方式，而這些都可能會出現。

Q5. 在一組 bitwise 運算中，&、^、|、~^、^~這些運算符號是否都會用括號區隔開來。
例如以下運算
b&a|c
是否一定會用括號來區隔部分運算，以辨明是
(b&a)|c 或是 b&(a|c)

A5. 同 Q4，都可能會出現。

Q6. 請問 bitwise 的 operator：&、|、~^、^~、^。這些 operator 是否有優先順序？

如假設有一個 expression 為

a|b&c

則這個邏輯將會是由左至右來看

也就是 a 先和 b 做 or，得到的結果再和 c 做 and。

又或者是&要先做

也就是 b 和 c 先做 and，得到的結果再和 a 做 or。

A6. 運算的優先順序在 verilog 格式裡，都有定義，單一運算會優先，再來才是位元運算 AND，位元運算 XOR 與 XNOR，接下來是 位元運算 OR，同樣優先度則會由左至右運算， 所以， a | b & c 會和 a | (b & c) 的功能相同。

Q7. 請問 assign 中的運算是否會出現 1'b1 或 1'b0 這類的常數呢?

A7. Test case 內不會出現常數值或常數值的邏輯運算。

Q8. 在 Question 6 中官方回答道：

a | b & c 會和 (a | b) & c 的功能相同。

然而根據 Verilog 的 Table of Operators Precedence

bitwise &的 precedence 應該是大於 bitwise |

所以 a|b&c，是否是跟 a|(b&c)的功能一樣才對呢？

Operator Precedence

Table below shows the precedence of operators from highest to lowest. Operators on the same level evaluate from left to right. It is strongly recommended to use parentheses to define order of precedence and improve the readability of your code.

Operator	Name
[ ]	bit-select or part-select
( )	parenthesis
!, ~	logical and bit-wise NOT
&, ~&, |, ~|, ^, ~^, ^~	reduction AND, OR, NAND, NOR, XOR, XNOR; If X=3'B101 and Y=3'B110, then X&Y=3'B100, X^Y=3'B011;
+, -	unary (sign) plus, minus; +17, -7
{ }	concatenation; {3'B101, 3'B110} = 6'B101110;
{{ }}	replication; {3{3'B110}} = 9'B110110110
*, /, %	multiply, divide, modulus; / and % not be supported for synthesis
+, -	binary add, subtract.
<<, >>	shift left, shift right; X<<2 is multiply by 4
<, <=, >, >=	comparisons. Reg and wire variables are taken as positive numbers.
= =, !=	logical equality, logical inequality
= = =, != =	case equality, case inequality; not synthesizable
&	bit-wise AND; AND together all the bits in a word
^, ~^, ^~	bit-wise XOR, bit-wise XNOR
|	bit-wise OR; AND together all the bits in a word
&&	logical AND. Treat all variables as False (zero) or True (nonzero).
||	logical OR. (7||0) is (TllF) = 1, (2||-3) is (TllT) =1, (3&&0) is (T&&F) = 0.
? :	conditional. x=(cond)? T : F;

A8. 的確，bitwise & 優先度大於 bitwise | ， 所以 a | b & c 會和 a | (b & c) 的功能相同，他們不是相同的優先度，將修正 Q6 的回覆，謝謝。

Q9. 想請問在今年的 D 題(Logic Synthesis using Programmable Logic Gates) 中能不能使用第三方的函式庫（如 abc），謝謝。
A9. 原則上不限制使用開源軟體的原始碼，加上不容易偵測比對，故不限制。

Q10. 請問 Problem D 中，提供的 verilog 檔範例元件庫(即 lib.v)， 和比賽時的檔案內容會一樣嗎？謝謝。
A10. 比賽驗證的元件庫檔案會和提供的測資裡的檔案一致。

Q11. 請問 TSRI machine 上，可以執行 testcaseD.tar 中，提供的驗證方式(即 lec 和 dcshell -f report.run)嗎？執行完都顯示 Command not found 是正常的嗎？謝謝。
A11. 執行驗證前，請先執行環境設定，
source /cad/cadence/CIC/CONFRML.csh
source /cad/synopsys/CIC/synthesis.csh

或是將上述設定加入 ~/.cshrc 檔內，登入會自動執行。

Q12.

```
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
```

檢驗用的 cadlib.v 的 gate2
這是剛剛官方新下載的測資
上面少打一組括號
會造成(i1& s0) & ~(s2 | s3)會先做運算
似乎會造成部分 gate 在 mapping 的檢驗上出錯
麻煩確認一下
A12. 經過驗證，的確會造成錯誤，將立即修改測資，謝謝。

Q13. 您好，我想要詢問 ProblemD Alpha test 的 2 個 hidden case 是否會公布呢？

A13. 為了維持公平性，隱藏的測資將會於 beta test 與 final test 使用，故不會公布。

Q14. 想請問 Problem D 在 final submission 時的評分標準，是否和 alpha 以及 beta test 一樣，以 total cost 最低的為優勝？另外想請問 final submission 所使用的測試資料，是否會和 alpha 以及 beta test 相同(5 筆公開測資，2 筆隱藏測資)，或是會再增加其他的隱藏測資？五筆公開的 test cases 在 final submission 時，同樣會計分嗎？能否詳述 final submission 的評分方式？謝謝。

A14.

(1) Final 的評分標準和 alpha 與 beta 的一樣，如題目描述的定義。

(2) Final 的測資和 alpha 與 beta 的一樣， 不會再增加其他隱藏的測資。

(3) 計分時包含公開的測資與隱藏的測資的結果。

Q15. 想請問在 problemD 中最後所輸出的 verilog 檔案中是否允許 variable 前面有"~"的存在，ex:gate1 g0(1'b0, 1'b0, ~a, ~b, 1'b0, 1'b0, ab);

謝謝您。

A15. 題目有提到: 合成後的檔案不能有 assign 指令，也不能有其他邏輯閘存在，全部都只能利用 gate1，gate2，或是 gate3 組合而成。

故變數參數內不能有~ & | 等邏輯運算。