資料結構與程式設計 - Final Project (Fall 2016)

Functionally Reduced And-Inverter Graph (FRAIG)

B03901023 電機三 許秉鈞
指導教授：黃鐘揚

b03901023@ntu.edu.tw

@adrian_hsu

2017.01.18

please include your design of the data structure, your algorithms at least for simulation and FRAIG, and the results and analysis of the experiments. Comparison to the reference program is welcome, but please discuss the performance and bottleneck (if any) of your program.

摘要

本份report共分為以下幾段：

1. Design of the Data Structure & Algorithms
2. CirGate Class design & FecGrp Class Design
3. Sweep/Opt/Strash implementation
4. Simulation/Fraig implementation
5. Experiments: Performance & Bottleneck (Comparison to the reference)
6. Course Reviews （學期心得）

1. Design of the Data Structure & Algorithms

Class Hierarchy 架構如圖。我把CirGate 繼承出Pi, Po, Aig, Const, Undef 五種 Gates，然後有需要轉型（e.g. Pi 和 Po 有些時候有自己的name，而其他gate沒有）再做處理。CirGate* 內部細節將在2. 說明。
FecGrp 則是用以存取一組具有相同FECHashKey的Cirgates pointer，因此除了_gateList外，還會有SimValue，也就是 0010_1101…的值轉成32bits之後存儲。

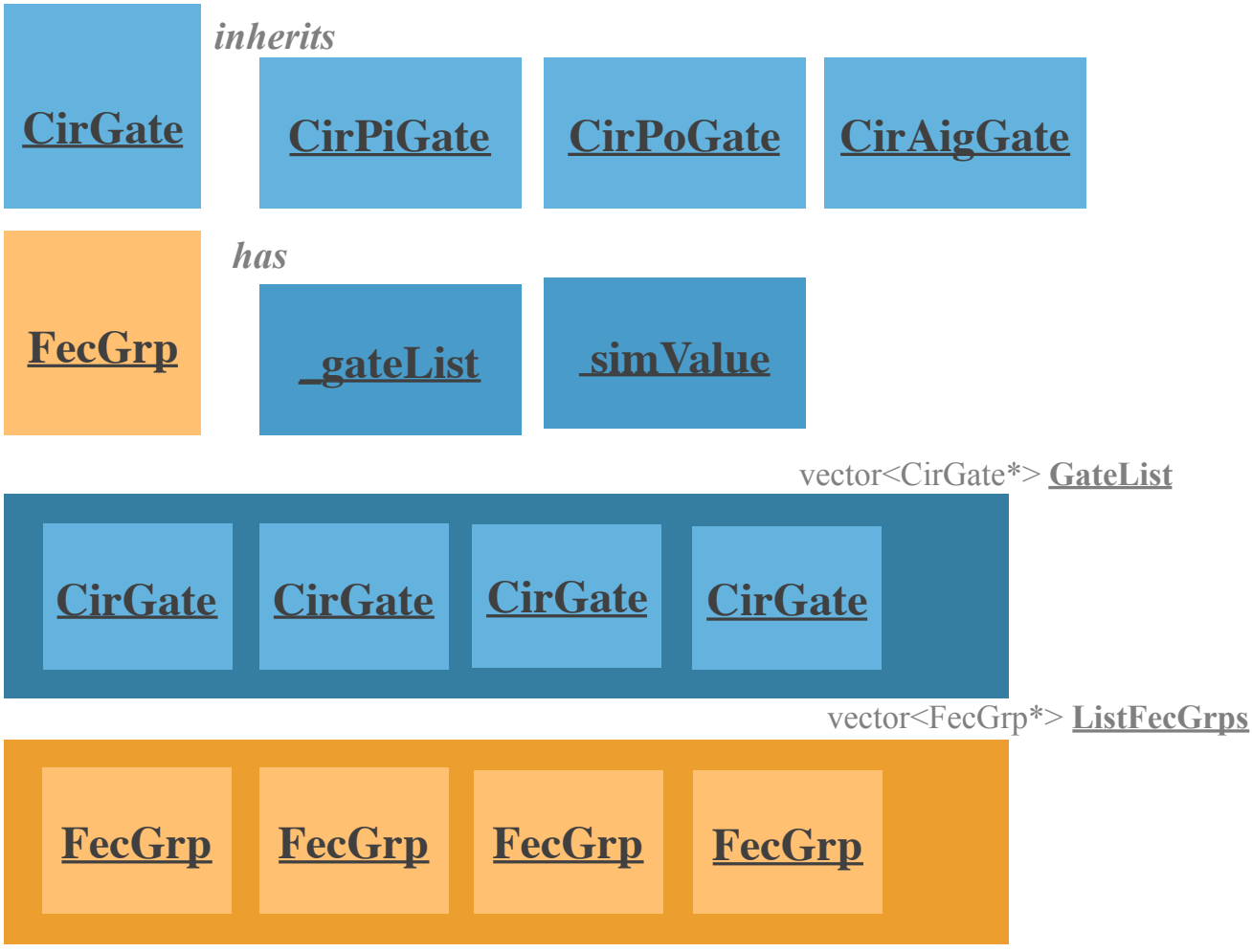


Figure 1. Class Hierarchy

2. CirGate Class design & FecGrp Class Design

```
CirGate:
        GateType type; // gate的類型，用enum存
        unsigned lineNo;// 用以紀錄剛讀入時是第幾行
        unsigned id; // variable id，乘上兩倍並加上 inv就是literal id
        mutable unsigned color; // 代表是否被踩過（DFS 會用到）
        unsigned simValue; // 32bit, 00100010011 etc
// _myFecGrp 要在Sim()之後才會有定義，若是被check Invalid 則會被強制 = 0
        FecGrp* _myFecGrp;
        bool fecInv; // 在第一次sim時，與同一grp第一人的Inv(訂為正)是否相反
        bool dead; // 是否已被fraig踩過
        Var _var; // SAT
        GateList faninList; // 用vector存方便移除與新增
        GateList fanoutList;

        static unsigned index; // 用在cirp -n，可以跨class查看當前印幾行
        static unsigned globalRef; // DFS 的判斷依據

FecGrp:
        GateList _gateList; // 存放這個grp的member
        size_t simValue; // 當前這次sim() 的Value（一次32bit）
```

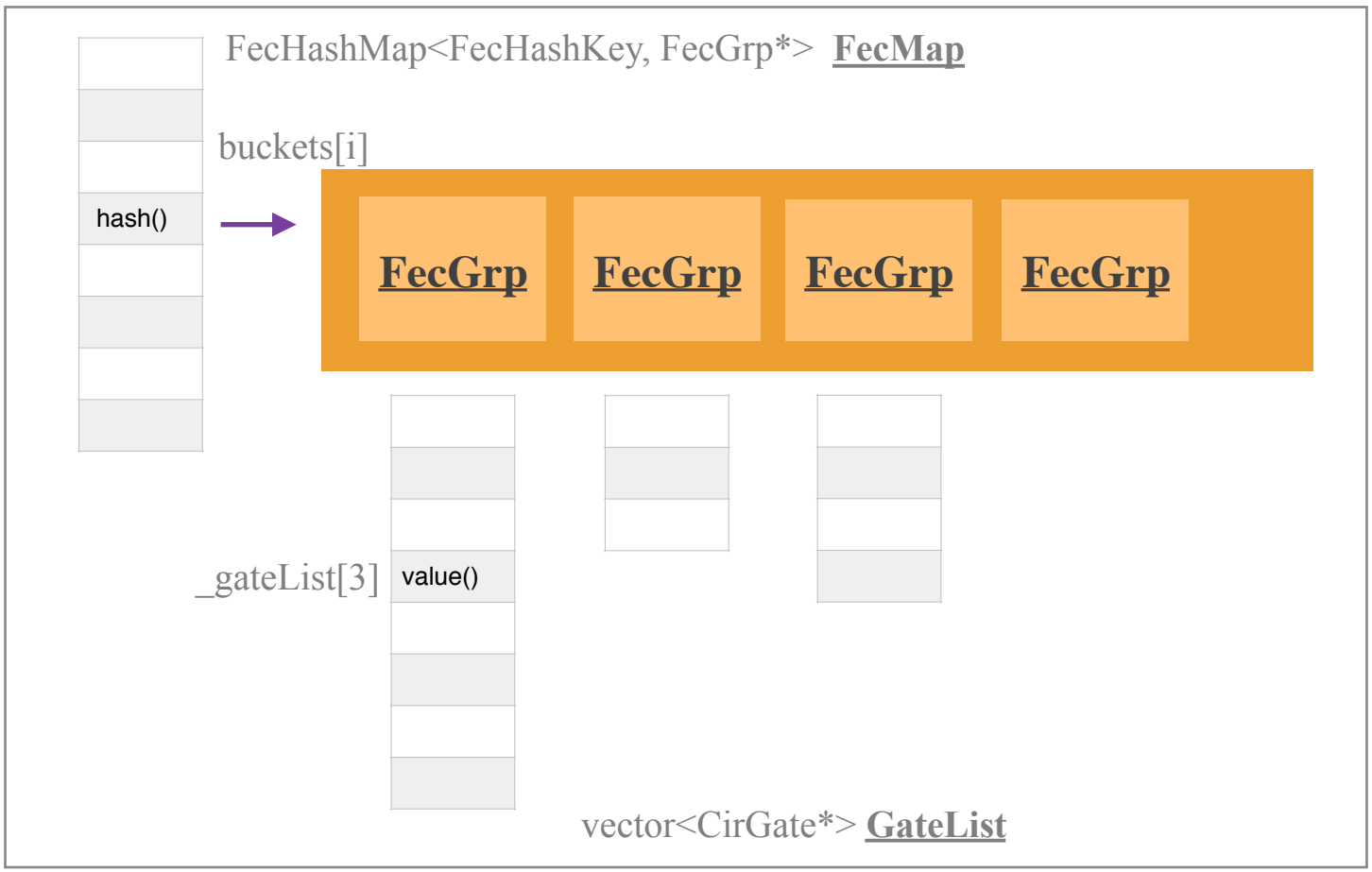


Figure 2. FecMap operation after simulate value is assigned to each Aig Gate*

3. Sweep/Opt/Strash Implementation

3.1 Sweep pseudo code:
先把buildDfsList() 建起來，然後走到的Gate，他GlobalRef() 就會++。接著把gateList 從 id=0 到底從頭掃一遍，如果GlobalRef() 跟當前的一樣的話代表他走過，否則沒走過、則進行 Sweep Operation。
比較需要注意的是，deleteGate()要記得把他前後的Gate的Fanin, Fanout都拔掉。

3.2 Opt:
Case分為四種、我個人將他的判斷順序稍微調換，順序如下：
若rhs0 == rhs1（即兩腳接在一起），則先判斷是SAME_FANIN還是INVERT_FANIN，然後回傳。若果rhs0 != rhs1，則接著判斷是否有接著CONST0，同樣需要考慮兩隻fanin。兩隻腳所做的操作相同。確認opt的種類後回傳，並把inv的結果以pass by ref的方式回傳到opt function。在optMerge()裡面，假設現在是g 要被刪掉、則g的fanouts 他們的fanin要進行移除，然後g 的 fanins 也要改成直接接到g的fanouts。確認都接上後、才把g刪除。

3.3 Strash:
個人認為自己的Strash操作還算簡潔且有效率。我的做法是先建一個HashMap，並且將aiggate 的 rhs0, rhs1 兩隻腳轉成size_t，然後利用**Cantor pairing function**這個方法（https://hbfs.wordpress.com/2011/09/27/pairing-functions/）將兩個size_t 雜湊出唯一的值。做出來後用 operator() 來overload這個功能。其他操作就跟HashMap一樣，把Key值丟進去、回傳一個Data 值，如果找不到則加進去Map內、找到了則回傳。我處理Collision發生（i.e. 剛好雜湊出相同的值）的方式是直接比對兩隻腳分別的size_t，如果相同則判斷掉。和Opt一樣，再刪掉Gate 的時候、前後腳的Gate的fanin、fanout要接到新的Gate上面。這邊較困難的是**fanout**。因為要找到自己是屬於這隻特定fanout的第幾隻腳，然後再補上，所以是兩層for。
code 如下：

```
bool is_inv = 0; //g是要取代this*的 CirGate*
   for(size_t i = 0; i < getfoutSize(); i++) {
      CirGate *fout = getOutput(i);
      size_t k = 0;
      for(; k < fout->getfinSize(); k++) {
         CirGate* me = fout->getInput(k);
         if(me == this) { // 我是這隻output的第k個input
            is_inv = fout->isInv(k);
            g->addOutput(fout); // 我的output變成g的out
            fout->insertInput(g, k, is_inv); //把g放在我原本的位置上
            break;
         }
      }// 出去後把我自己的前後腳拔掉
   }
```

4-1. Simulate() Implementation

in fecGrpsInit():

// 把每個Gate 塞到同一個Grp，inv 設為0
// 把Grp塞入listGrps內

in simulate():

// All-gate simulation:
// Perform simulation for each gate on the DFS list
在這做的就是，按照dfslist的順序把pattern吃進來，也就是從PI先吃、接著一路往PO吃，一直到每個gate都有他當下的SimValue為止。

in fecGrpsIdentify():（參考*figure 1.*，在第二頁）

```
// LET'S DO THE MOST IMPORTANT PART!
for_each(fecGrp, fecGrps):
   Hash<SimValue, FECGroup> newFecGrps;
   for_each(gate, fecGrp)
      grp = newFecGrps.check(gate);
      if (grp != 0) // existed
         grp.add(gate);
      else
         createNewGroup(newFecGrps,gate);
CollectValidFecGrp(newFecGrps, fecGrp,fecGrps);
```

以文字解釋figure 1. 的話，Identify()這個操作的大致流程如下：
先把當前的listGrps 整個丟進來、然後一個個grp檢查。在檢查這個grp時，我們開一個「暫時的」FecHashMap，用來當作配對搜尋工具。對於每個在此Grp的gate，去看他是否已經存在於這個map，若不存在、則丟進去hash、然後拿到key、進而把data存在最好的bucket位置；若已經存在、就只要把在目前map暫時開的空間(grp)、把gate丟進去就好。
做完操作後，最後要Collect 有效的Grp，做法很簡單、只要把size > 1的保留下來。然後再用iterator的方式把Map裡面的一群一群Grp找出來，進而塞到listGrps內，這樣就完成操作了。
其中，我在hashMap的check()，做了以下操作：

```
if(_buckets[n][i].first == k)
     inv = 0;
     return
n = bucketNum(~k);
 if(_buckets[n][i].first == ~k)
     inv = 1;
     return
```

也就是說，要確認這筆新進來的key，他是以k 還是 ~k 的形式雜湊、並丟進去bucket內的。
如果他不是以正(inv = 0)的方式、則上半部不會return掉，而是接著跑下半部、變成用~k去找雜湊與bucketNum。設定這inv = 1的目的是讓傳到 function外之後，他的gate自己要把他是inv與否設定好。

Q. 如何選擇randSim()何時停止？我的設計概念？

我的想法如下：

```
#define MAGIC_NUMBER 5.0
   MAX_FAILS = (size_t)(MAGIC_NUMBER * sqrt(in));
   if(MAX_FAILS == 0) MAX_FAILS++;
```

這是我訂定MAX_FAILS的方法，（in 是input gate數量）。我是用一個調參數的方式、測了log()與sqrt()、並且也測了以aig, 以in、甚至兩個相加的方式來調整。最後覺得這個MAX_FAILS是最合理的值、跟ref也較為接近。

我的rand pattern生成的方式是，2 * rnGen(MY_RAND_MAX) 其中MY_RAND_MAX是**2147483647**，也就是INT最大的值，乘以兩倍後就是UNSIGNED INT的最大值，也是32 bit能容許的最大長度。

Q. MAX_FAILE何時會增加？我的設計概念？

那麼、MAX_FAILS怎麼增加呢？我的想法是，在listGrpsSize經過兩個接連的patterns simulate後仍不變的時候，則有可能他的grp member是已經固定下來的，雖然機率不是太高，但多次以後就可以蠻確認已經固定下來，所以我才會用這個方式：

```
if(IS_RANDSIM && _listFecGrps.size() == _tmpListFecGrps.size())
      CURRENT_FAILS++;
```

4-2. Fraig() Implementation

in genProofModel():

我把所有DfsList內的Gate都add進去solver、並特別把AIG挑出來、push進去fecAigList這個vector內。

in prove():

簡單來說我的做法是：把fecAigList的Gate依剛剛Dfs的順序一個一個吃進來，然後檢查她的fecGrp是否存在、存在的話把fecGrp裡面其他的所有Gate都取出來、和他自己的Var做assumpProve()，如果prove出的結果是SAT、只要把自己從fecGrp移除並且break就好；如果是UNSAT、則把這個其他人的Gate暫存到die這個vector內，而自己的Gate則存進live這個vector，依序做到整個grp都檢查完畢；出了for loop後，先以die的id「用DFS先遇到先處理」的順序做sort、然後將這些預期要die的Gate從當前fecGrp移除，並且call fraigMerge() 來處理與其他人的腳位merge問題。有個特例是，如果出現CONST0的話、則要把調換live與die的順序。

Q. 我怎麼設計如何停止？每個*pair*都證明嗎？

我的做法蠻直覺的：當這些Gate都被清空時（因為都是和dfsList內遇到的第一個merge），有點像是有個main branch而其他小branch 共同merge到這條main branch的概念、而不是一部分一部分的小branch 分邊merge起來、最後才merge進去mainbranch。這個作法壞處是，他的停

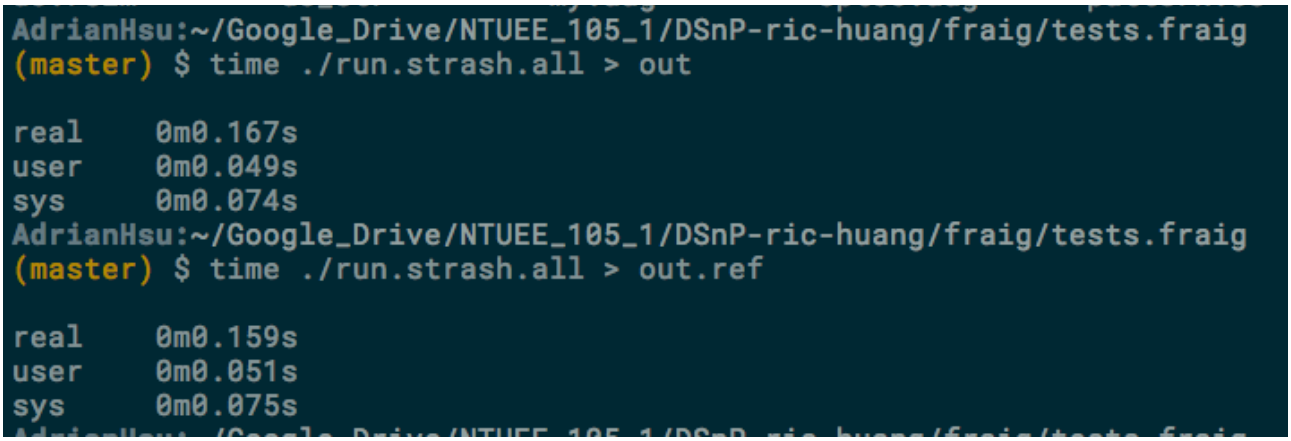
止條件就是要清空這個group、而因為我沒有做pattern的存儲，所以他也不會部分的merge；但有個很大的好處是、當grp清空就會merge完畢、後面都不會遇到。

5. Performance & Bottleneck (Comparison to the reference)

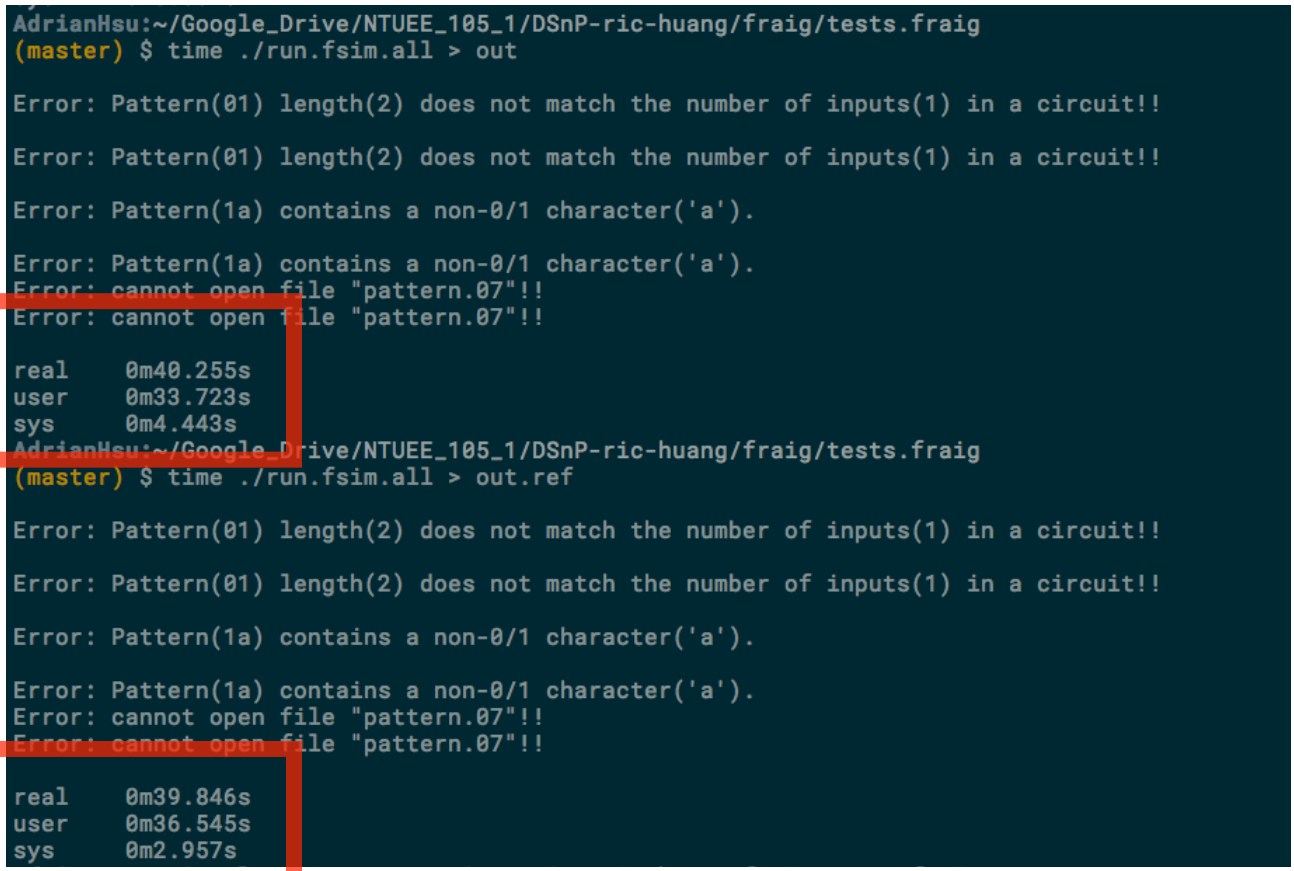
註：上面都是我的結果、下面都是ref的結果

./run.opt.all 秒數和ref都在0.3秒內

./run.strash.all 秒數和ref都在0.3秒內，記憶體用量也差不多



./run.fsim.all 可以發現我的表現跟ref只差1秒以內、應算還不錯。



./run.rsim.all 可以發現我的表現跟ref差了不少秒數、但我的正確率高蠻多的、FEC切很開。

```
AdrianHsu:~/Google_Drive/NTUEE_105_1/DSnP-ric-huang/fraig/tests.fraig
(master) $ time ./run.rsim.all > out

real    0m14.113s
user    0m13.089s
sys     0m0.497s
AdrianHsu:~/Google_Drive/NTUEE_105_1/DSnP-ric-huang/fraig/tests.fraig
(master) $ time ./run.rsim.all > out.ref

real    0m5.525s
user    0m4.821s
sys     0m0.374s
```

./run.fraig.all 可以發現我的表現跟ref差了約1.5倍（時間），空間倒是差蠻多的（因為我沒有處理空間、而是選擇優化時間），然後Gate的數量有差一些，因為我沒有做%32 的re-sim()

```
fraig> cirp

Circuit Statistics
==================
  PI        3357
  PO        3343
  AIG      77622
------------------
  Total    84322

fraig> usage
Period time used : 179.3 seconds
Total time used  : 179.3 seconds
Total memory used: 342.9 M Bytes

fraig> q -f
```

<我的do.fraig>

```
fraig> cirp

Circuit Statistics
==================
  PI        3357
  PO        3343
  AIG      77311
------------------
  Total    84011

fraig> usage
Period time used : 102.7 seconds
Total time used  : 102.7 seconds
Total memory used: 44.27 M Bytes

fraig> q -f
```

<ref的do.fraig>

另外，在fraig的正確性方面，我已經將提供的所有測資用ref的cirmiter測過，都能把aig數量化為0，也就是確認兩個cirquits做miter後是 equivalent的。（超感動的ＱＱ）

6. Course Reviews

這堂課雖然作業很重，但說真的學到不少！HW#1 教的是最簡單的.csv 格式處理與parsing，我記得這部分寫得還算快，應該在一天內就寫完了；HW#2 是有點麻煩的cmdReader，我記得我因為沒有仔細看code、所以少用了幾個老師寫好已提供的function、因此自己多寫了好幾十行；HW#3 是mydb、需要自己設計dbTable、然後進行db的操作；HW#4是memTest、我記得這幾個作業的 cmd Parsing都頗為複雜、還有最經典的「用size_t 直接取出下一個的pointer」。HW#5 是我最喜歡的作業、也就是實作ADT（DLIST、ARRAY、BST），因為我

的BST寫得還不錯、而且把MIT press那本演算法課本的BST做出來、很有成就感；HW#6...蠻累的，印象中寫了頗久、而且我沒做error handling 被扣了分...；HW#7的hashset很好玩、而且難度不高，蠻棒的。最後...Fraig，我從段考前兩個禮拜、花了約5天的時間把他的sweep、opt、strash一次寫完，然後愉快地面對期末考...。期末考後因為還有嵌入式實驗demo、所以一直到上禮拜五能開始寫fraig，然後一邊寫才發現還有不少問題沒有解決，像是writeGate()、印出simlog...等，最後終於在deadline前一個晚上跑完了sim13.aag的miter結果，看到cirp 的aig 數量為0的那一刻真的很感動很有成就感。

謝謝老師、謝謝助教這學期的教導、辛苦了！