DSnP fraig Report

B07901021 潘世軒

email: b07901021@ntu.edu.tw

一、作業引用

在 HW6 時我的 code 在 cirg -fanin 和-fanout 的地方沒有拿滿，因此在這兩個函式之內使用了 **B07901016 朱哲廣** 同學 的 code 並進行修改，使其能在我的程式裡的結構運行，就開始做期末報告的部份了。

二、Cirsweep

在我的 CirGate 裏面存取是否在 DFS 的時候有沒有被走到，在 Cirsweep 時尋找那些沒被走到的然後刪除之，並對有存取 gate 的 list 更新，基本上這個部份算是非常簡單的。

三、Ciroptimize

- **Helper Function for Ciroptimize:**

void ban_fanout(CirGate* target, CirGate* banned)

會把 target 所有指到 banned 的 fanout 都清除掉，避免連結錯誤

void ban_fanin(CirGate* target, CirGate* banned)

會把 target 所有指到 banned 的 fanin 都清除掉，避免連結錯誤

void jump_connect(CirGate* now, CirGate* last, CirGate* next, bool inv)

用來執行跳過 now 然後把 last 和 next 連起來的部份，會先把 now 的 fanout 全部加到 last 裏面，然後 ban_fanin ban_fanout，避免連結錯誤

照投影片上面分的四種 case，注意一些小細節，再透過我寫的 helper function 的幫助就可以寫完了，這裡複雜度稍微高一點，多 segmentation fault 幾次後改一改就可以過了，弄完後重跑一次 DFS

四、Cirstrash

- **Helper Function for Cirstrash:**

void merge(CirGate* target, CirGate* banned, bool inv = false)

會把 banned 的 output 加進 target 的 output 內，然後 ban_fanin ban_fanout(extern)，避免連結錯誤，inv 用來判斷 target 和 banned 是不是反向的(fraig 才會用到)

把 input 的兩個 fanin(原始 aagfile 的那兩個數字)丟去 hash 之後拿到 key 放進

unordered map 裏面，key 選用 Cantor pairing function，如果已經在裏面了就把他和那個 key 對應到的 value merge 起來，弄完後重跑一次 DFS，再這裡之所以使用 unordermap 的原因是比起改作業 7，還不如上網看 unordered map 比較快 XD，但也隨之產生了一些問題(見後方)

```
// method:Cantor pairing function
// key = (a + b) * (a + b + 1) / 2 + a; where a, b >= 0, a>=b
```

終於要進入比較難的部份了...

五、Cirsimulate

- **Helper Function for Cirsimulate:**

bool readPattern(vector<size_t>& pattern, string buffer, size_t level)

讀這行的內容，然後判斷他是不是合法的，最後根據 level 的層數把 pattern+=(size_t)(1<<level)

void CirMgr::simulate(vector<size_t> pattern)

根據 dfslist 去 simulate，因為這個比較好寫，而且時間也不會差很多(？，跑完之後 update_FEC

void CirMgr::update_FEC()

如果第一次做的話，把 CONST0 和所有 DFS 走的到的 AIGgate 都加進去同一個 FECgroup，然後照著投影片的那欄 pseudo code 做，其實我覺得這部份最難的就是看懂 pseudo code(汗)，當初我還在想說到底要怎麼 hash 兩個 size_t 但不會爆掉 XD，但看懂投影片後就照著做，然後考慮反向的部份就好了~詳細會在下面說明

void CirMgr::set_FEC()

在每次 simulate 全部 pattern 之後把_FECgroups 裡面的各行 FEC 的指標設給那行裡面的每一個 gate

- **Some structure for Cirsimulate**

Class CirGateV

存了 Cirgate* 和 inv，用來表示他在 FECgroups 裏面是正向還是反向的存在，支援一些些 Cirgate 裏面有的功能，其實當初本來想要直接把這部份弄在 Cirgate 裏面，但是如果要反向的話好像要再創一個出來，Cirgate 裏面又包含了許多資訊，cout 過他的 size 在我的 code 裏面會是 200，因此後來我選擇寫了一個新的 classCirgateV 來實作這個部份

vector<map<size_t, CirGateV*>>* _FECgroups

_FECgroups 選擇使用 vector 來包 map，原先是使用 vector<vector>的，但是看到後面要 print_FECpairs 的時候就一個大改架構改成這樣了，因為 map 裡面是有

排序的，所以到時候要印_FECgroups 裡面的內容的時候就可以直接用 iterator 來 go through 整個 map 並印出來就好了，若是使用 vector<vector>就要行和每一列都 sort 一次，會比較花時間，所以後來使用 vector 來包 map，至於那個 size_t 是用來存後面的 cirgateV 的 ID，讓他排序~這邊還有一個讓我 debug 很久的地方(之所以 _FECgroups 是一個 vector 指標而不是一個 vector 的原因)，會在後面提到~

CirGate:: map<size_t, CirGateV*>* _fec;

這是 Cirgate 用來存和這個 gate 裡面是 FECpair 的 gate 的指標，這個指標就是對應到上方_FECgroups 裡面自己有出現的那行，存指標一來比較省空間，二來如果有變動的話只要動一下指標就好了，不需要把整個新的 map 乾坤大挪移(耗時又耗空間)

這部分的精華部份應該除了設計資料結構來方便完成功能之外就是如何判斷反向的_FECgroup 了吧~，在這裡我將作這部分的 pseudo code 打出來。

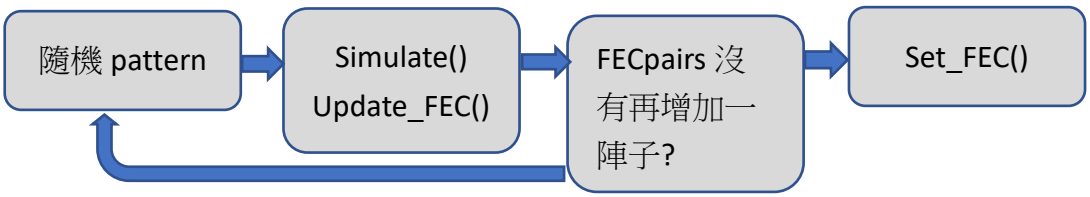
```
For every FECgroup in _FECgroups
        Hashtable<simvalue, fecgroups>
        for Gate in each FECgroup
            key =_simvalue
            if(inv) key = ~key;
            if(find key in Hashtable)
                add Gate into sim_map with key
            else if(find ~key in Hashtable)
                add ~Gate into sim_map with ~key
            else
                open and add Gate into sim_map with key
        for fecgroups in sim_map
            if contain more than one gate
                update _FECgroups
```

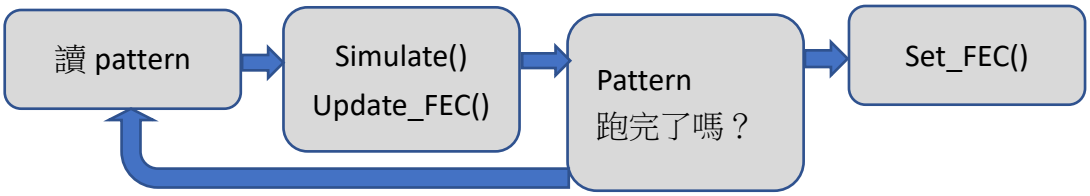
當初為了處理反向的部分花了很多時間在想，最主要就是卡在中段的地方，搞不清出到底是要加反的 gate 還是反反的 gate 進去，一直重複 try and error 的過程，到最後還是想不透就只好自己想了一個不小的情況用紙筆一步一步做，才弄出正確的判斷方法

只是還是有一個小問題就是我 fecgroup 裡面的第一個 gate 有可能是反的，所以如果有需要 print -FEC 的話要先判斷那個 FECgroup 的第一個是正向還是反向，如果

是反向的話後面全部都要再反向一次(好繞口 XD)

Simulate 分為 -file 和-random 兩種，畫一下各自的流程





我設定-random 模式下是否停止的判斷依據是連續 n 次_FECgroups 的數量沒有再變動就停止，n = (log(number of AIG gates))^2，基本上跑起來還 OK

六、Cirfraig

這部分我有寫一點點但是沒寫完...

有寫的：

修改 satTest.cpp 裡面的內容包括 genProofModel 還有如圖所示這部分

```
void CirMgr::get_SAT_result(pair<CirGateV*, CirGateV*>& target, SatSolver& solver, bool& result){
    Var newV = solver.newVar();
    solver.addXorCNF(newV, target.first->get_VarID() , target.first->get_inv()
                         , target.second->get_VarID(), target.second->get_inv());
    solver.assumeRelease();  // Clear assumptions
    solver.assumeProperty(newV, true);  // k = 1
    result = solver.assumpSolve();
        // reportResult(solver, iter->second);
}
```

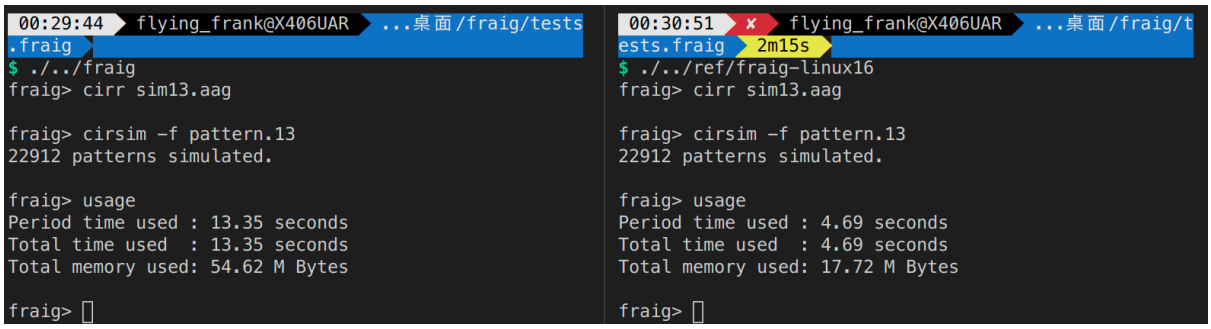
可以做到無腦跑每個 fecgroups 裡面 C N 取 2 做 SAT 然後 cout 出來每個 fecgroups 有哪兩個是 SAT 或 UNSAT 的並決定誰 merge 誰，但是問題是一來這樣有點粗糙，沒辦法把 FECgroups 減到最少，也沒有把該 delete 掉的東西 delete 掉，我這邊就是只要嘗試 delete 東西就會直接 segmentation fault，後來也來不及慢慢 debug，因此這邊就是半成品...

本來想到的作法：(沒有實作，也不知道有沒有用)

先做和 0 在一起的 FECgroup，並判斷是否可以和 0 merge 起來，假設 a 可以和 0 合起來的話，下面只要出現 a，就把那個 FECgroup 直接 ban 掉(都已經從 0 分出來了就不可能一樣)，下面只要出現！a，就直接把 a 從那個 FECgroup 裡面 ban 掉(因為他

是 1)，然後面剩下的每做完一個 GROUP 就把所有中被 merge 掉的那些 gate 改掉，然後一直跑跑跑 SAT~

七、performance



平台：linux16(comile->O3)

其他部分(除了 fraig)雖然有差異，但差異不大，就先不討論了

以最大的 sim13.aag 為例，跑起來的時間大約是 refernce 的三倍左右...，memory 也多用了蠻多的，memory 的問題在下面會討論到，時間的問題原本是非常久(約 5 倍左右)後來大改了讀取 pattern file 的方法，本來是全部讀進來到一個 vector<string>，然後再讀，但是這樣不僅要花很多記憶體存 pattern，讀起來也會比較慢，就改成一行一行邊讀邊跑，就達到了上面的狀況，後來發現是在也花了大概一個小時看到底哪裡可以再加速，但沒有找到...就讓他維持這樣的狀況了

八、已知問題

1. 檔案裏面有#include<unordered_map>，在 make 的時候 compiler 會說

error: #error This file requires compiler and library support for the ISO C++ 2011 standard. This support must be enabled with the -std=c++11 or -std=gnu++11 compiler options.

去 makefile 裏面察看也發現有

CFLAGS = -O3 -Wall -std=c++11 -DTA_KB_SETTING $(PKGFLAG)

CFLAGS = -g -Wall -std=c++11 -DTA_KB_SETTING $(PKGFLAG)

這樣的標示但 compile 就是不給面子....

發生一個很奇怪的狀況就是只要 make clean 後再 make，就可以成功 make，但每次改東西後就要重新 make 好像有點麻煩，所以後來也發現另外一個替代的解決方法，make->噴錯->make->成功，至少這樣比較快 XD。

2. HW6 我讀.aagfile 的方法是把.aag 的內容全部存進一個 vector<string>再一行一

行 parse，但如同上面所說的這樣的方法會造成 memory 用量比較多，在讀進 sim13.aag 的時候就會已經用掉 3,40M 了，本想在最後用空間時間把這個地方改掉，但時間不構只好作罷…

3. 一開始在做 HW6 和這個作業前半的時候就是直接無腦做，比方說 cirgate 裡面需要什麼 datamember，就直接加進去，也沒有一些取用這個 member 的條件，然後再加上 set 和 get 的函式讓外界存取，造成後面在 debug 的時候會變得有些困難，在作業後半也才開始學會用 assert()這個功能，雖然之前的作業就常常看老師幫忙寫好的 code 裡面有用到這個東西，但很後面才開始用他，有時候多寫一些 assert 和用 debugger 的輔助就能夠比較快速的找到 bug，所以 code 中有一部分看起來很落漆 QQ

4. 一開始在 cirmgr 裡面有太多東西來存放相似的東西，比方說我不只有一個所有 gate 的 vector，還有各種 gate 的 vector，造成在刪東西的時候容易產生 double delete error，搞得我 debug 很煩，好在開始的第二天我就狠下心來去蕪存菁把 cirmgr 裡面大改了。

5. 用 cirgateV 的地方和 cirgate 之間有點錯綜複雜，在要寫 fraig 的時候發現我這個地方的結構有點鬆散，所以讓我在那個地方 delete 或者 merge 的時候就會產生 segmentation fault，最後也沒時間去重改這個地方，fraig 也因此沒有實作完 QQ

6. 一開始我在 cirmgr.h 裏面 cirmgr 的成員加了一個 vector<vector<CirGate*>> _all 想要用他來存所有的 FECgroup，但是我想要在 cirsim 裏面 call 他的時候（這之前完全沒動過這個東西）就會產生一些奇怪的東西
第一個很奇怪的點是_all.size()是一串很大的數字而不是 0
第二個很奇怪的地方是只要我想要對_all 做會動到他的事情，比方說= 別人，或者 resize()和 push_back()的時候他都會直接 segmentation fault
這個問題卡了我四五個小時.....上網查查不到原因，寄信問了老師老師和我說可能是哪個地方讓他的 memory crash 掉了，我找了找也還是不知其所以然，所以就改成 _FECgroups 是一個指標，在 cirsim.cpp 裡面 initialize 的時後再把它 new 出來，用不到再刪掉，這樣就不會有問題~算是個知其然但不知其所以然的情況。

7. 沒有做 error handeling，我只有辨別出 error 然後 return false 而已…

九、修課心得

雖然最後沒能完成 fraig 的部分有點可惜但是我在這堂課也學到了很多東西，老師在第一堂課的時候說過 ***心中要有記憶體*** ，這句話實在是刻骨銘心阿，每個禮拜都要花一堆時間在處理 segmentation fault，我也是到很後面才開始用 debugger 和 assert 來減少 debug 的負擔，如果前面就開始用的話應該可以更快寫完前面的作業吧 XD，同時也親身體驗了一個好的結構可以讓 code 功能的運行簡單而明瞭，但一個沒搞好的資料結構卻會送你一堆 segmentation fault，我到後面寫 fraig 的時候真的是悔不當初阿 QQ，先前作業都是在老師給的模板上面作業，通常都只要弄一些 function，直到最後 fraig 的時候才知道這個的重要性。

大一在寫 c++的 final project 的時候就常常需要什麼變數亂加在 global 然後一些東西就亂弄，現在回去看那時寫的 code 真的是有點好笑，希望我在這堂課之後寫 code 可以妥善運用所學，寫出一個「漂亮」的程式。