FRAIG REPORT

學號：B02901013

姓名：鄧傑方

手機：0955-387-923

信箱：b02901013@ntu.edu.tw

一、概述

與 Homework 6 相比，FRAIG 主要增加了五項新功能：CIRSWEEP, CIROPTMIZE, CIRSTRASH, CIRSIMULATE, CIRFRAIG，感覺功能強大了許多，也複雜了許多！

在最初 parse 檔案進來時，我利用 map 將 GateID 與 CirGate*連結在一起，而最後進行了一次 DFS，紀錄走過的 Gate。每次進行前，都先 setglobalref，將 globalref+1，就可以利用 globalref 來判斷是否被走過，建造出了一個 NetList 的 vector 來存整個 DFS 的順序。

二、CIRSWEEP

第一個部分先做 CIRSWEEP，每次 sweep 時先做一次 DFS，就可以判斷哪些 Gate 是沒走到的，將這些 Gate 的 fanin、fanout 進行與其他 Gate 的連結處理後就可以從 map 中移除，完成 sweep 的部分。

三、CIROPTIMIZE

第二部分的 optimize，我先設了一個 bool doOpt，決定在做 DFS 的時候要不要進行 optimize 的動作，若要進行 optimize 則判斷此 Gate 屬於四種狀況(fanin 是否為相同 Gate、是否有接 const…等)，若有符合，則將其 fanin、fanout 進行處理、連結，相較於 CIRSWEEP 處理 fanin、fanout 會複雜一些，最後將 optimize 產生的多餘 Gate 從 map 中移除，完成此一 Gate 的 optimize。當走完 DFS 即可完成此電路的 CIROPTIMIZE。

四、CIRSTRASH

第三部分的 strash，須由自己設計出一個 key，來對應到_buckets 中的位置，我使用的方法是將兩個 fanin 的 GateID 相乘，如果有 not 閘，則將 GateID+1 後再相乘，_numBuckets 則是 AIG 數量的兩倍，實際使用起來在 Gate 數量不多時比較容易分在同一個_buckets 中。與 CIROPTIMIZE 類似，我設了一個 bool doStr，決定在做 DFS 的時候要不要進行 strash，產生 HashNode，並存入_buckets 中，再存入之前，我會先判斷在_bucket 中是否已存有相同的 key，若有且其 fanin 也確實相同，則用_buckets 中存在的 Gate 將此 Gate 進行 merge 的動作，若否則將此 HashNode 存入_buckets 中。

五、CIRSIMULATE

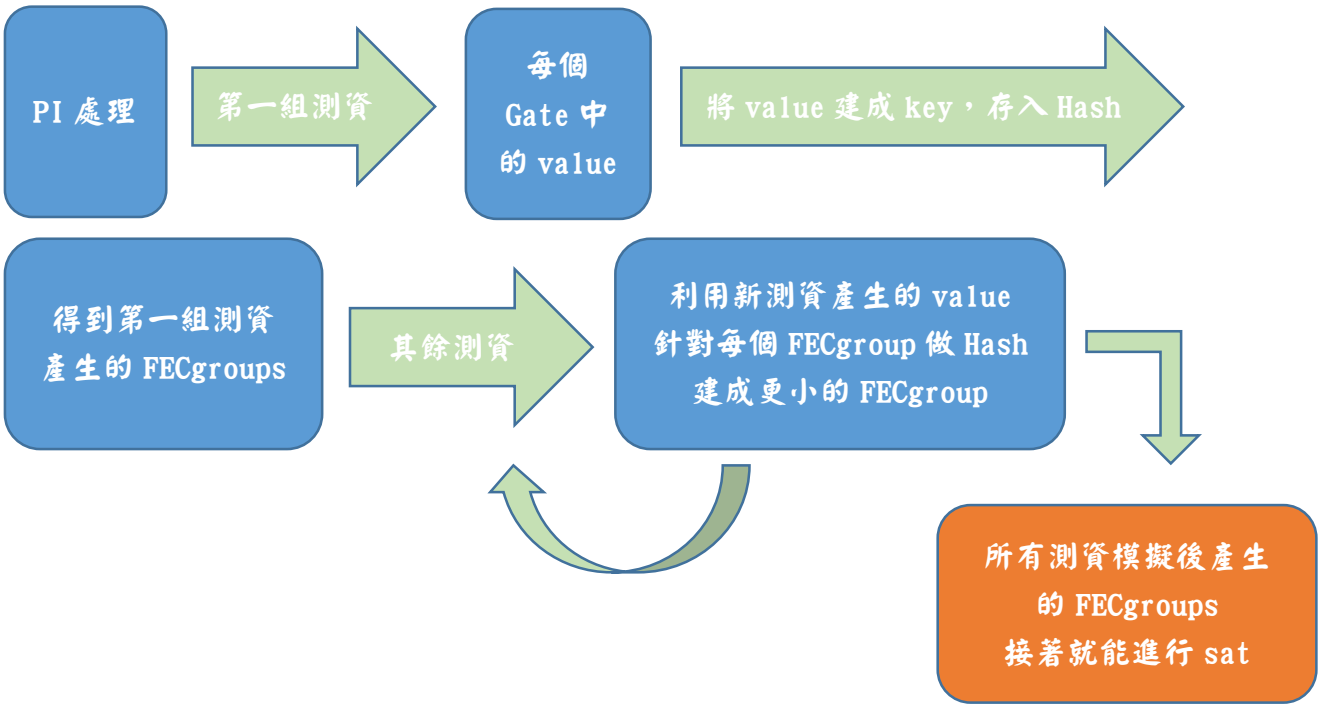
第四部份的 simulate 真的複雜了好多，一開始我先在 CirGate 新增了 size_t* value 的 member，並以二進制的形式來存-file 或-random 產生的模擬資料，每 32 個 pattern 為一組，這樣就可以直接使用&來加快模擬的速度，並將 PI 的 value 處理好之後，才真正進入電路的 simulate，產生 FECgroups。

PI 有了模擬資料後，我一樣是設了一個 bool doSim 先進行 DFS，將每個 Gate 的 value 模擬出來，並使用第一組測資(前 32 個 pattern)模擬的結果，也就是每個

Gate 中的 value 設計成 key，存入 Hash 中，方便我直接找到 FEC pairs，因為要考慮 IFEC 的問題，所以每個 Gate 我都將其 key,~key 存入 Hash 中，使用上比較直覺，可是會有一些要處理的問題，在速度上也會慢一些。

接著用 vector< vector < pair <HashKey , CirGate* > > >作為 FECgroups，若在 _buckets 中的 FEC pairs 小於兩組，就可以直接砍掉，最後就可以得到用第一組測資(前 32 個 pattern)模擬出來的 FECgroups。

之後就繼續進行其他組測資的模擬，一樣是每 32pattern 為一組來模擬。此時就只要再針對 FECgroups 中的每個 FECgroup 進行，方法類似做第一組測資的方式，將此組測資產生的 value 一樣作為 key，存入 Hash 中，並判斷是否可以將此 FECgroup 分成更小的 FECgroup，之後將單獨 FECpair 砍掉，這樣就大致完成最終所有測資模擬後產生的 FECgroups。



比較值得注意的是模擬 pattern 的數量吧，因為在測試 sim13.aag 時，光是跑 4xxx 個 pattern 就要花上 15 秒左右，跟老師的 20000 出頭 pattern 花 4~5 秒相比慢了許多，所以我 pattern 的數量並不是由存在的 FECgroups 數量來決定，而是由 PI 的數量經過數學運算來取得一個合適的 pattern 數量，不過當然 pattern 跑得少，最後 cirp –fec 的數量就會比較多，跑出來的結果我的有 44xx 左右的 FECgroups，而 ref 則只有 38xx 左右！

Cirgate&Cirp –fec 算是經過 simulate 之後需要特別去處理的功能吧！Cirgate 印出 value 值時，我是把最後模擬的那組 value 印出來，若是-file 的 pattern 倍數並不為 32 時，在 PI 的時候則會補上 0。而因為 Cirp –fec 有要求要按照 GateID 順序排列，所以每次 insert 到 Hash 中時，除了找到對應的_buckets，我還由後往前比 GateID 的大小決定其位置，之後用 vector < pair <HashKey , CirGate* > >裝 FEC

pairs 時比較方便。

六、CIRFRAIG

之前一直想說 simulate 做完之後，fraig 應該很快，真正做的時候才發現不是這樣，光是研究 sat 就花了不少時間，之後就可以利用 sat 幫我們證明 FECgroup 內的 Gate 是否一樣。基本步驟就是對每一個 FECgroup，將其中的 Gate 兩兩抓出來做 sat 比較。因此就每一個 FECgroup 我先用他的第一個 Gate 來與其他 Gate 比較，若是 UNSAT 則代表這兩個 Gate 完全一樣，先繼續留在此 FECgroup 中，若是 SAT 則將 Gate 移除，並存到新的一個 FECgroup(tmp)中，最後存在此 FECgroup 中的就會只剩完全一樣的 Gate 了，而若是 tmp 中的 Gate 大於一個，則將此 tmp push_back 到 FECgroups 中，最後 FECgroups 中的每個 FECgroup 就只會存在完全一樣的 Gate，接著就可以進行 merge 的動作。

不過我卻在這裡崩潰了，一直 crash…，紀錄一下現在時間 2015.1.20 04:03，先去睡個覺好了，大概一兩年沒這麼晚睡了…，希望早上醒來可以把它完成。

現在時間 2015.1.20 11:55，大致上把之後 merge 的動作完成了。在得出 FECgroups 中的每個 FECgroup 都只存在完全一樣的 Gate 之後，利用先前得到的 DFS 時 AndGate GateID 的順序，進行 merge 的動作，每走一個 AndGate 就判斷要用哪個 Gate 進行 merge，用的是在此 FECGroup 中 DFS 最早走到的那個 Gate，我想之前會 crash 可能就是沒用 DFS 最早走到的 Gate 來 merge，然後亂連接，導致電路爛掉。這個方法感覺有很多缺點，因為每次都要在 FECGroup 中找到 DFS 最早走到的 Gate，再加上 FEC、IFEC 皆會存在 FECGroups 中，使 sat 證明的數量變成兩倍，也沒有記錄重要的 pattern，不過死線在即，只能先求有部分功能，之後來得及的話再來慢慢修！

後來又改了一些地方，改成先不證明，而是利用先前得到的 DFS 時 AndGate GateID 的順序，每走一個 AndGate 也是會進行判斷，若此 Gate 為其 FECgroup 中進行 DFS 最早走到的那個 Gate 時，則在此時進行這組 FECgroup 證明的動作，即可少掉大概一半的證明數量，不過做完剩餘的 AIG 數量總是跟 ref 不一樣，真是奇怪！

七、performance 比較

(1) CIRSWEEP

Cirsweep 感覺起來跟 ref 的差不多，不過因為我要再跑一次 DFS 來判斷哪些 Gate 是走不到的，所以在 sim13 時，雖然沒 remove 任何 Gate，可是也會花上 0.81 秒 DFS 的時間。

(2)CIROPTMIZE

感覺也是跟 ref 差不多，應該差在 DFS 的時間。

(3)CIRSTRASH
測 sim13.aag 時雖然沒有任何可以 strash 的 Gate，但還是花了 1.5 秒，主要可能還是 DFS 的時間。

(4) CIRSIMULATE
因為我的設計方式是每模擬完一筆測資(32pattern)就會存入 Hash 一次，然後再得到比較小的 FECgroup，可能就是在重複進行這個動作時把時間都耗掉了，用 usage 查看的結果，記憶體使用量也暴增了不少，有時間的話應該這個地方有很多可以改進的地方,目前有個想法,若是連續跑個 10 筆左右的測資,若 FECgroups 的數量一直沒有減少，應該就能推測極難找到可以分辨他們的 pattern，此時就可以進行 fraig 以減少時間。測 sim13 的結果就如同上面我所述。

(5) CIRFRAIG
一開始用 DFS 幫 SatSolver 建電路。接著對每一個 FECgroup 之中的第一個 Gate 與其他 Gate 進行證明，時間複雜度 O(N)。UNSAT 的 Gate 就繼續放在此 FECgroup 中，最後才會 merge；SAT 的話，就會移除並存入新的一個 FECgroup(tmp)。整個 FECgroup 比對完，tmp 的 size 有大於 1，就將 tmp push_back 到 FECgroups。
跑比較小的電路都能證出來，在測 sim13 時則直接卡住了，有太多可以加速的小技巧都沒用，也難怪會卡住，有時間一定要把它修到好！
改成新的方式後，又測了一次 sim13，可以看的到他在緩慢進行 merge 了！

八、心得

上完了這一學期的資結，我想增進的不只是程式的能力，更重要的還有查資料、自我學習的能力，尤其是一開始的幾個 homework，很多東西都是大一計程沒有用過的，像是之前都沒用過 vector、map……等，這個時候就要一直爬文，研究如何使用，過程真的學到很多。此外就是如何妥善安排時間，以及那種真的花 20 小時 up 的時間完成一件 homework 的決心及毅力吧,這真的很磨練人的耐心,對於電機系的學生也很重要，以後想必會有很多 project 可能要花更長時間，尤其我覺得我沒什麼天分，前幾個 homework 都是將時間砸下去，努力完成所有功能，將遇到的 bug 處理掉，所以分數都能拿到八成以上，過程真的很艱辛，不過看到最後寫出來的作業在功能上幾乎都能與老師相同時的那種喜悅感(速度另當別論)，真的都快喜極而泣了，也很有成就感。

FRAIG 果然不是徒具虛名，在過年的那幾個連假我就寫完了 CIRSWEEP、CIROPTIMIZE、CIRSTRASH，想說利用考完的五晚上、六日一二整天將整個 FRAIG 完成，豈知 CIRSIMULATE 的功能一下子複雜了好多，在處理效能上也花了很多時間，而 CIRFRAIG 更是崩潰了好久，只能完成一部份的功能，在效能的處理上還是無法改進，比較大的電路都證不出來，最後沒能將 CIRFRAIG 做到完整，還滿遺憾的，感覺就是有種缺陷，還有很多進化的空間。

很高興在大二就修了資結這們課，比較能感受到身處電機系的那種感覺了，很累但是我很喜歡，每次作業都能感受到自身能力的進化，真的很謝謝老師，超級扎實的，好課大推 XD！