**Data Structure and Programming**

**Final Project**

Name: 黃君灝

Student ID: r05631044

Email: [r05631044@ntu.edu.tw](mailto:r05631044@ntu.edu.tw)

1. CirGate

Gate的部分我是用inheritence寫的，先宣告一個pure class叫CirGate，再分別去繼承它，變成UndefGate、ConstGate、PiGate、AigGate、PoGate，CirGate裡面紀錄了一些重要的資訊如gate的id、symbol和一個做Depth First Traversal紀錄的變數ref。而判斷當前這個Gate是哪一種型態是依據兩個member function，一個印出字串，另一個回傳enum並且可作為switch的參數。

依據不同的類別，每個gate會有特定的member data，像是PiGate會有fanoutList而不會有faninList，這樣繼承的方式可以讓code看起來相較enum type的方式來的乾淨並且省一些記憶體。其中AigGate和PoGate的faninList是另外宣告成一個CirGateV類別的vector，這個CirGateV裡涵蓋一個size\_t的數字，可以將他cast成任意類別的指標，並且用前三個bit來紀錄當前這個gate與其fanin gate是否有inverse的關係。

1. myHashMap

寫一個類別用來存取gate的資訊，這是一個hash的資料結構，所以要達到搜索接近O(1)的複雜度，主要用以做strash以及FEC pairs的存取。存取的key是hashKey的class，裡頭涵蓋兩個size\_t，分別是AigGate的兩個fanin，而判斷key的存在只要兩個fanin都一樣即可不用順序也一樣，這邊在重載operator ==的時候實現。hashData則是之前寫的CirGateV的class，如此還可以運用這個類別裡面的member function。

1. CirMgr

管理整個cirgate的類別CirMgr裡主要紀錄一些相關的gate list，主要有piList、poList、totalList等在讀取電路aag時會建立起的list，以及後面建立電路時串起的dfsList。

* readCircuit

在讀取電路的部分寫了一個vector<string> inputs紀錄aag檔裡面每一行的內容，同時建立每一個gate存起來，遇到一個PiGate就new一個PiGate的class，並且幫其標上id、line number等等，另外AigGate & PoGate在建立的同時也有紀錄它們各自的fanin，建立好每一個gate以後變開始將每一個gate串接起來。

* constructNet

將建好的gate串接起來，就是將每一個gate的fanin接到自己身上，並且fanin的fanout也接回去。若是遇到不在totalList裡面的id，就new一個UndefGate的class，再把它加到totalList裡。

* dfsTraversal

將每一個gate串連好以後就開始作尋訪的步驟，因為PO是輸出，所以在尋訪的時候要從poList開始做起，這樣走到的每一個gate才是真的有功能有在運作的。演算法的部分是利用系統的stack幫我們做遞迴，將每個gate的fanin先做，做完以後才做自己本身再加到dfsList裡頭。

* sweep

這個指令用途為將沒有作用到的gate移除，意思就是不再dfsList裡的gate變將其從totalList刪去。我的做法是做尋訪，若不在dfsList便移除它，然而由於dfsList是有順序的是用vector寫成的，這樣一來我每次在移除一個新的gate時都得花上O(n)的複雜度去作搜索，確認是不是在dfsList裡，這樣會使得整個運作沒那麼有效，因此這邊我令了一個STL的unordered\_set，將所有dfsList中gate的id都放進這個set裡，由於unordered\_set是用hash所寫，所以之後再確認這個gate是否在dfsList裡就可以從這個set中作尋找，時間複雜度就變為O(1)，可以有效的提升效率。

在sweep過程中找到不在list裡的gate以後主要做一件事，將這個gate的fanin的fanoutList中對應到這個gate的指標移除掉。一般來說通常還要將這個gate的fanout的faninList對應到這個gate的指標也移除，不過因為是要移除dfsList中不存在的gate，所以亦即從poList中走不到的gate才要被移除，因此若是當前這個gate不在dfsList中，那同時也代表他的fanoutList也不會在dfsList，所以這裡我沒有作這個部分。最後只要將先前建立的set中所以id對應到的gate從totalList移除即可。

* optimize

在optimize的部分我也是創了一個unordered\_set，都先將要移除的gate id加到這個資料結構之中，最後在做完處理以後再將所以set的id刪去。主要將四種不同狀況的gate做優化的動作，因為都只會發生在AigGate上，所以我寫了兩個AigGate的member function分別為simplifyConst & simplifyFanin，並且也是再作一次dfsTraversal時加入。

simplifyConst主要將fanin中有ConstGate的AigGate做簡化，只要有const 0該gate便去除，將其fanout對應到該gate的fanin接上const 0，也在const 0的fanout接上該gate的fanout；再來若有const 1也一樣去除該gate，不過這邊不同的是不是接const 0，而是接該gate的另一個fanin gate。

simplifyFanin則是將fanin對應到同一個gate的AigGate移除，有兩種狀況，一種是同時有反向器或同時都沒有，另一種是一個fanin有另一個沒有。前者是將該gate移除，變將其fanin接到所有fanout中，若有反向則將fanout的CirGateV加入Inv()；後者是將其移除，並將const 0接到fanout。

一樣在最後的部分，將set中的id對應到的gate從totalList裡移除，變可以做到opt的功能，只是要在最後移除gate以後再做一次dfsTraversal更新dfsList。

* strash

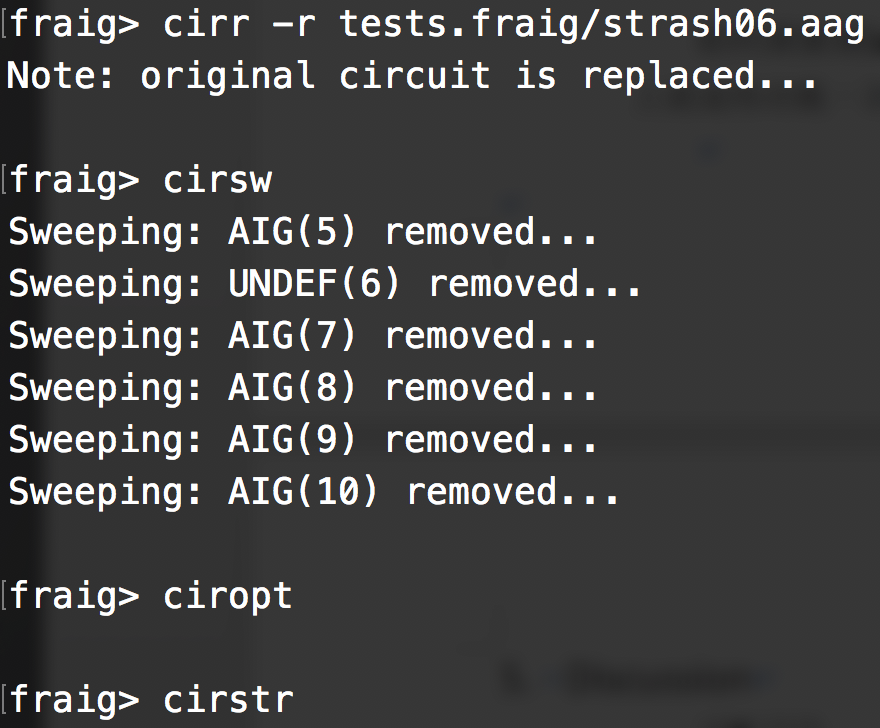
這裡要將有相同fanin的AigGate簡化，合併其中一個讓電路更簡潔，為了達到並且有效率的執行這個指令，這邊寫了一個hash的資料結構，以fanins pair作為hashKey，CirGateV作為hashData，從poList開始做dfsTraversal，美遇到一個AigGate就將它加進hash中，若遇到一樣的key就將這兩個gate做合併，並且更改合併gate的fanin fanout。然而一更新完可能會改變到其他的AigGate，因此更新後若是AigGate被更動，就在將這個AigGate做一次strash以確保不會漏掉新的hashKey & hashData。在最後時一樣將unordered\_set裡的id刪去，簡化成新的cirgate，也一樣要在更新dfsList。

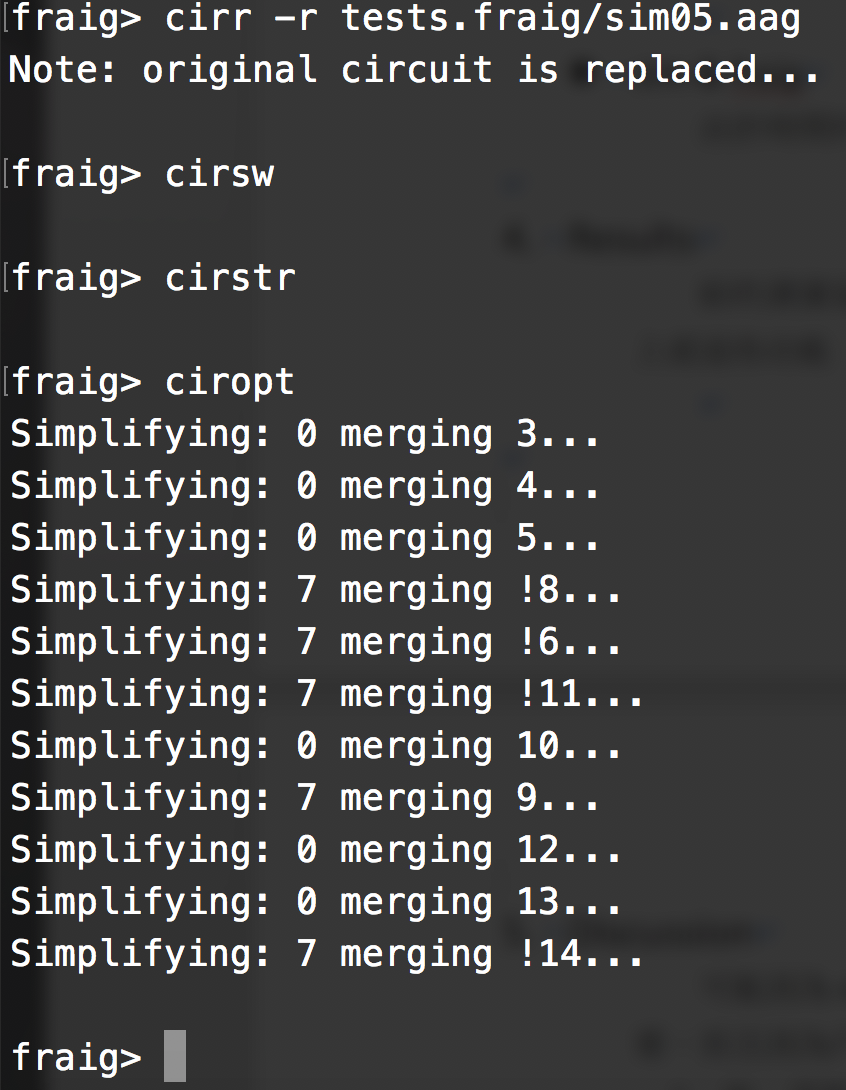
* sim & fraig

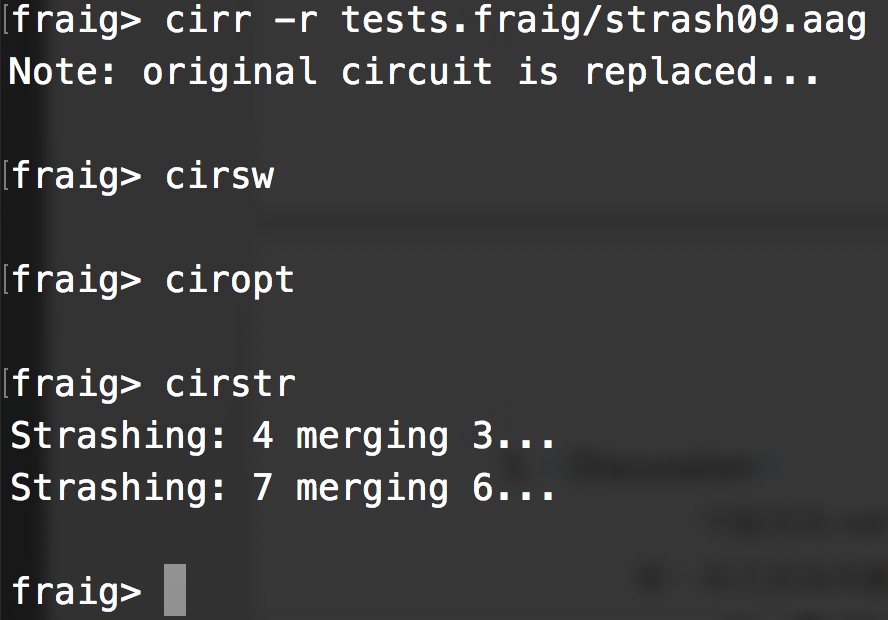
由於時間的關係目前只有做到前面的部分，所以這部份從缺。

1. Results

給的測資測試了幾個以後，應該在大部分的策資裡頭都可以運作上述這些功能，以下是一些測試的小結果。







1. Discussion

可能因為code沒時間做優化的動作，相對起來有點不是那麼好看，而且因為用繼承的關係，我做了很多的dynamic\_cast的動作並assign給一個新的pointer，我想可能就是這個部分的原因有要記憶體且複製一份，所以時間上跟refcode會有些差，gate數量少時沒很明顯，但我想若資料量一旦變得龐大時可能還是會差了一段，這部份是後續代改進的。另外因為沒有時間做完剩下sim & fraig的部分，所以主要該分析的部分都無法實現，這是這次final project的遺憾，後續若有時間會慢慢琢磨並且完成它。