**Final project report**

**Functionally Reduced And-Inverter Graph (FRAIG)**

**Course: NTU EE3011 Data Structure and Programming (DSnP)**

**Teacher: Ric Huang**

**Name: Shannon Lee (李尚倫)**

**ID: r07921001**

**Email:** [**r07921001@ntu.edu.tw**](mailto:r07921001@ntu.edu.tw)

**Full code:** [**https://github.com/shannon112/DSnPorygon/tree/master/fraig**](https://github.com/shannon112/DSnPorygon/tree/master/fraig)

**Course website:** [**https://github.com/ric2k1/DSnP.open**](https://github.com/ric2k1/DSnP.open)

**I. Introduction**

Functionally Reduced And-Inverter Graph (FRAIG) 是 Electronic Design Automation (EDA)的一環，主要功能是化簡重複功能的電路片段，讓整個電路變得更精簡，但保有和原本一樣的功能。可以從三個方面來看他的好處：

1.Area

- Reduce the number of gates

- Moreover, using library cells of smaller sizes, but they will have weaker driving capability 2. Timing

- Shorten the longest path

- Additionally, insert buffers and/or enlarge the cells to increase the driving capability

3. Power

- Reduce the switching activities

- Moreover, shutdown the sub-circuit that is not currently used

**II. Problem description**

****

**III. System overview**

****

主要的系統有分兩大塊，一塊是負責維持command line相關功能的程式以cmdMgr這個物件為中心來運作，於作業三以前已經完成。一塊是負責處理 AIGER (.aag) format電路的讀寫相關功能，以cirMgr這個物件為中心來運作，並於作業六中已經完成電路讀取和各種顯示的部分，本次project的內容則會以我的作業六為基礎疊加，加入編輯電路的功能，更新的原始碼集中於cir/中，其他未改，以達到運行以下功能的能力：

1. Unused gate sweeping (cirsweep)

2. Trivial optimization (constant propagation) (ciroptimize)

3. Simplification by structural hash (cirstrash)

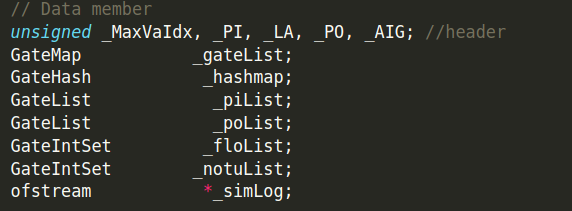
~~4. FRAIG: Equivalence gate merging (cirsimulate, cirfraig) 未完成~~

**IV. Date Structure**

****

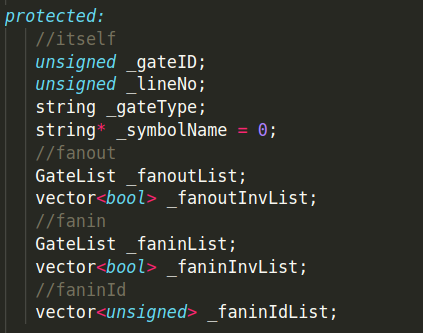
1. Cirmgr Class

裡面儲存的data member有aag的header以unsigned存，在電路編輯過程中AIG的數量會跟著變動，而主要儲存所有gate的大容器是\_gateList，是用map implement的以確保在搜尋時快速，且在traversal時不會因為改動而順序錯亂(相對於unodered\_map)，\_hashmap則是在cirstrash功能時會用到，用來儲存hash值和gate的pointer，是unodered\_map也就是hash table，可以快速find, insert和delete，\_piList和\_poList則是一般的vector，floating gates和unused gates存在\_floList和\_notuList中，用set存，方便儲存(消除重複儲存)和刪除(即使沒有，刪了也不會error)。

****

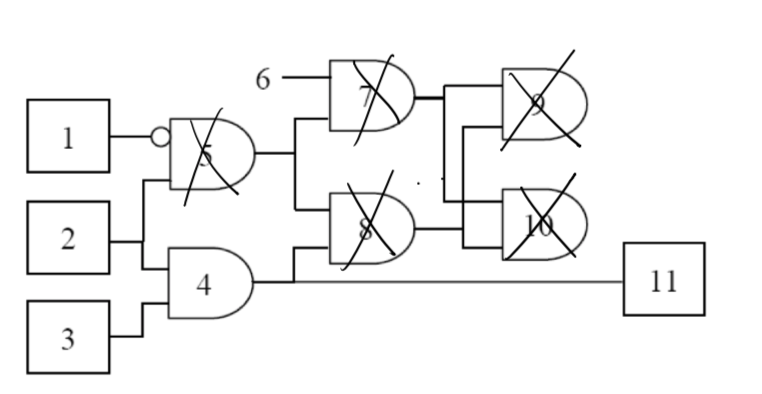
2. CirGate Class and its family

這個部分使用到Polymorphism的技巧，透過CirGate來製造出CirPiGate, CirPoGate, CirAigGate，每個Class都代表各自對應的gate (UNDEF和CONST0屬於CirPiGate)，而CirGate家族裡面儲存的data member有基本的gate資訊如：\_gateID, \_lineNo, \_gateType, \_symbolName，和剛開始讀入電路時需要的faninID和他的inverse list，以簡單的vector來儲存，等到讀完file要做電路connection的時候，再建構每個gate的fanout和fanin list和他們的inverse list，來儲存fanin和 fanout gate 的 pointer，並有多個access function配合，來對這四個vector list做編輯。

****

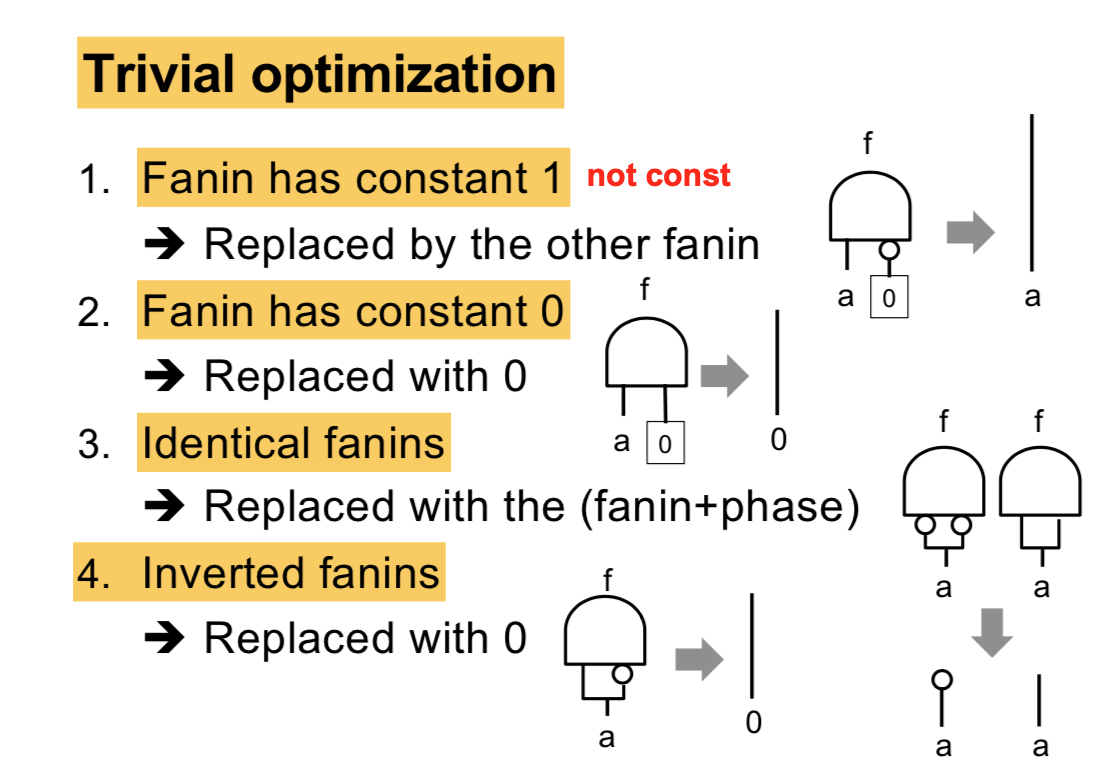
**V. Algorithm**

1. Unused gate sweeping (cirsweep)



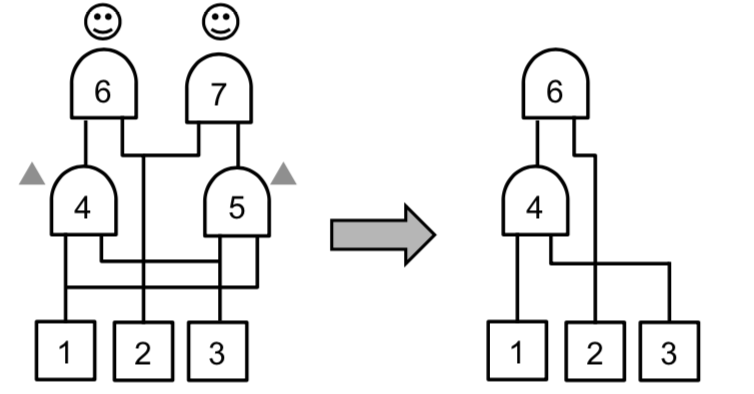
因為是要清掉不是從PO做DFS可以觸及到的gate，所以我的做法是先跑過一次DFS但並不清(reset)掉marked，然後直接traversal全部的gate，只要沒marked就通通刪掉，即可達成此功能，只是在實作時要特別注意刪掉gate時他的fanin gates和fanout gates的fanout和fanin的內容也要清乾淨。

2. Trivial optimization (constant propagation) (ciroptimize)



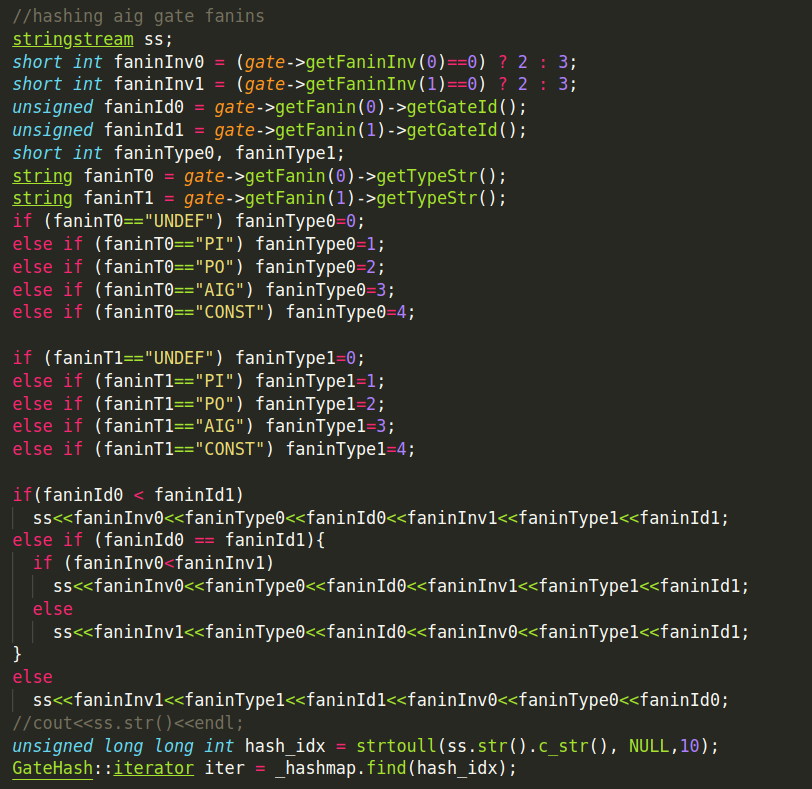
因為這個功能是化簡後前面的gate會影響到後面的gate，所以我的作法也是跑DFS，DFS從PO一路擴展到PI，然後在一路往回結束recursive function時(DFS演算法中把node塗黑的部分)，來用if else做這四種trivial optimization case的分析，即可達成此功能，只是在實作時也要特別注意刪掉gate時他的fanin gates和fanout gates的fanout和fanin的內容也要清乾淨，然後case1和case3會有inverse負負變成正的情況。

3. Simplification by structural hash (cirstrash)

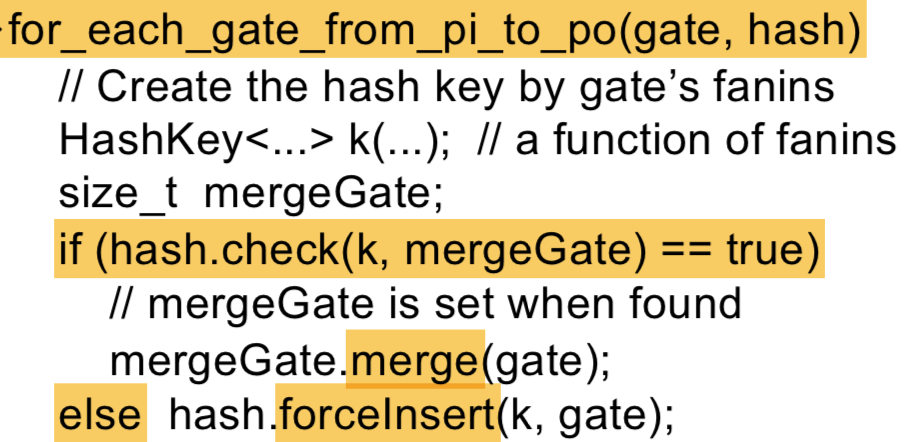


因為這個功能也是化簡後前面的gate會影響到後面的gate，所以我的作法還是跑DFS，DFS從PO一路擴展到PI，然後在一路往回結束recursive function時(DFS演算法中把node塗黑的部分)，來用hash table判斷這個gate的fanin以前有沒有出現過，有的話就合併，沒有的話就加入hash table，即可達成此功能，只是在實作時也要特別注意刪掉gate時他的fanin gates和fanout gates的fanout和fanin的內容也要清乾淨。

Hash方式的部分，我把每個gate的fanin hash成一串unsigned long long int：

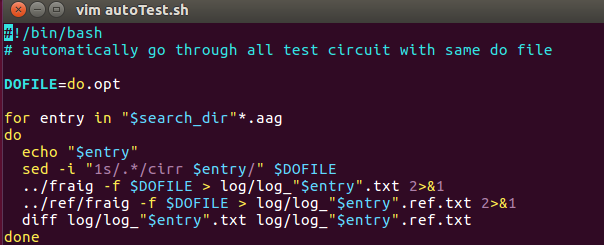


Pseudo code的部分：

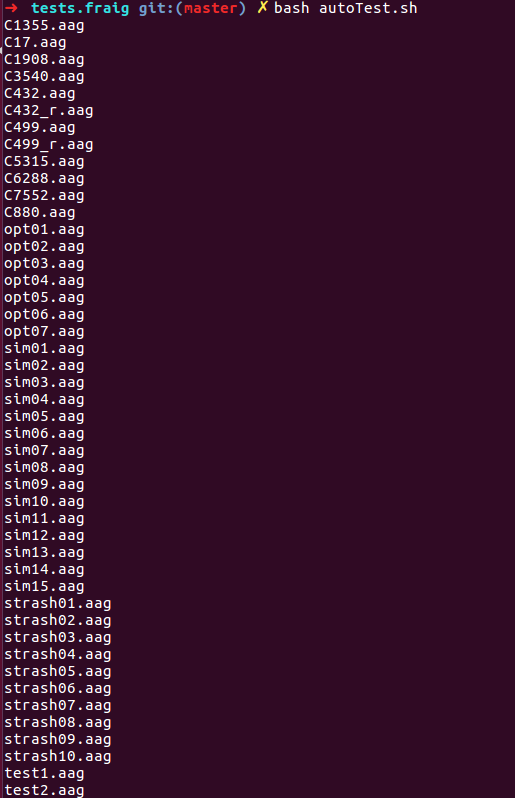


**VI. Experiment & Result**

根據作業給的電路，tests.fraig裡面的所有電路都為測試對象，因此我自己寫了一個bash script來對每個電路都自動跑過同一份的dofile，並且每個電路都要reference program和我自己的program做diff比較差異，以確認correctness。autoTest.sh如下：

****

而下圖則是全部pass時會有的輸出，因為diff在兩個檔案完全一致的時候不會有output，所以output就只剩下檔名。

****

1. 單測試cirsweep

dofile的設計上，簡單讀入電路後，看cirp的所有輸出方式，看cirg挑幾個gate做fanin和fanout，就進行cirsweep，然後再看cirp的所有輸出方式，再看cirg挑前面挑的幾個gate做fanin和fanout。

2. 單測試ciroptimize

dofile的設計上，簡單讀入電路後，看cirp的所有輸出方式，看cirg挑幾個gate做fanin和fanout，就進行ciroptimize，然後再看cirp的所有輸出方式，再看cirg挑前面挑的幾個gate做fanin和fanout。

3. 單測試cirstrash

dofile的設計上，簡單讀入電路後，看cirp的所有輸出方式，看cirg挑幾個gate做fanin和fanout，就進行cirstrash，然後再看cirp的所有輸出方式，再看cirg挑前面挑的幾個gate做fanin和fanout。

4. 整合測試

dofile的設計上，簡單讀入電路後，看cirp的所有輸出方式，看cirg挑幾個gate做fanin和fanout，就進行cirsweep, ciroptimize, cirstrash，然後再看cirp的所有輸出方式，再看cirg挑前面挑的幾個gate做fanin和fanout。

**VII. Discussion**

根據實驗結果，大致上可以保證這次有完成的三大功能，在correctness上沒有問題，但因為是沿用自己上次作業的架構和function，除了error detection和circuit write(忘了真的輸出成檔案)沒拿到分數外，我還有在fanin和fanout的顯示上並沒有拿到全部的分數，所以fanin fanout的功能上可能還有瑕疵，導致如果在別的測資上有測到與reference program不符，也有可能是作業六就已經存在的bug( 雖然部分function也有在這次做修改，但不知道有沒有改到)，並不一定是這次實作的功能本身的bug。

而會有這樣全部通過的結果，也是經過一次一次的測試和修改而來的，像是實驗中提到的三個功能分開來測試時，就已經跟對個別function修改到沒有錯，但在同一份dofile一起側的時候，又會發現之前沒發現的bug，如此一直修改下來才有最終的結果，詳細修改的bug可以參閱github上的commit，數量眾多就不在此一一列出。

**VII. Conclusion**

這次因為時間不太夠，只成功implement了sweep, optimization, strash並驗證他們的correctness，在我設計的dofile下都有成功通過所有可取得的側資，期望之後有時間還可以完成另外cirsimulate, cirfraig的功能，並驗證correctness和大家的performance。