Fraig報告

林展慶*

系級：物理二　學號：*b05202044*

(Dated: January 17, 2018)

I. 前言

在學期中，我們已經完成了hw6，是一個有關 AIG(And-Inverter Graph)讀寫的程式。在當時的程式中， 我們已經提供了下列有關AIG讀寫的commends:

- CIRRead：Read the circuit and store it.
- CIRPrint：Report circuit infromation in different ways.
- CIRGate：Report the gates in the circuit.
- CIRWrite：Write the netlist to an ASCII format AIGER file.

而在這次功課中， 我們需要為我們hw6寫的程式多提供以下幾個有關化簡電路的commends：

- CIRSWeep：Remove the gates that cannot be reached from POs.
- CIROPTimize：Perform trivial circuit optimizations including constant propagation andredundant gate removal.
- CIRSTRash：Merge the structurally equivalent gates.
- CIRSimulate：Perform circuit simulation to distinguish the functionally different signals and thus collect the FEC pairs/groups.
- CIRFraig：Based on the identified (I)FEC pairs/groups, perform fraig operations on the circuit.

II. DESIGN OF THE DATA STRUCTURE

首先，我們先來看一下在這次功課中我所設計的class 、當中存了哪些data以及用途是什麼：

- CirGate 用來儲存電路中gate的資訊(包括id, input, output)。 原先在hw6我並無使用繼承的方式來區分 不同type的gate。但是在最後幾堂課的時候， 看到老師寫的code寫得很漂亮就心動重新用繼承的方式 來重寫一次hw6了。比較值得注意的點有幾個：

- 我的Derived class中只有CirPOGate跟CirPIGate 有多存一個拿來紀錄symbol的string。
- 在紀錄input時，我是紀錄input的address， 並且將有沒有invert的information存在 這個address的第一位（老師上課時 有教的技巧），並且我在用一個class叫做 CirGateV將一些acess的function包在一起。
- 在寫sim跟fraig時需要儲存FECGroup， 但在這裡我做了一個蠻錯誤的決定， 我用List的方式儲存group， 而且我還沒有額外寫一個Group的class將整個group中的gate包起來，而是直接 在每個gate當中直接存兩個gate的pointer（一個是List的head，一個是下一個的pointer） 來將整個group串接在一起。當時會這樣做事因為在寫sim的時候，我並沒有 顧及到後面fraig要怎麼寫，想說還要 多寫一個Group和Groups的class 來implement很麻煩，再者直接將FEC的資訊 存在gate當中蠻方便的就直接寫了， 沒想到在後面寫fraig的時候 反而在處理group這方面的問題卻花費了很大的功夫。

接著我們來看一下，在CirGate中存了哪些data:

1. vector<CirGateV> _outputVList; 存放所有output的address。
2. CirGateV _in1V, _in2V; 存放兩個input的address。
3. size_t _simValue; 存放simValue。
4. Var _satVariable; 存放這個gate在satsolver中對應到的index。
5. CirGateV _FECBroV; 存放FEC group的head。
6. CirGate* _simDick; 用於連接FEC group的data。
7. mutable size_t _myFlag; 給DFSList使用的flag。
8. static size_t _trvlFlag; 給DFSList使用的flag。
9. size_t _lineNo; 紀錄讀取檔時的行數。
10. size_t _id; 紀錄gate的ID。

- CirMgr： 在這個class裡存放了所有的gate的address， 並且在外層(CirMgr)如果想要執行circuit 相關的commend都是藉由這個class的物件 去呼叫member function來完成。 以下提出幾個比較值得注意的點：
  - 我在讀取完aag檔的第一行後(知道電路的大小上限後) 我就會去new一個存放CirGate的位址的Array 並且在往下讀取檔案後會一一的把CirGate new 出來並依照id存到Array當中。

- 我會將所有有效的FEC group(大小大於一) 的head的address存在一個vector當中。
- 我有將從PO做DFS能夠搜尋到的gate 依照DFS的order來將他們存到一個vector裡面。 其實我在hw6的時候，並沒有這樣寫， 我當時的寫法是，每當遇到需要DFS的時候 我就會跑去CirGate這個class當中多寫一個 有關那個DFS需要做得member function。 這樣做其實真的很麻煩。 真的很感謝當時的教我這樣寫得那位同學， 不然我可能連fraig都寫不到。
- 在進行fraig的時候，我選擇將 要merge的資訊全部先丟到一個vector裡面（merge的資訊我用set包起來） 會這樣做得原因是因為，在做fraig時 我是找DFS順序中，第一個有有效group 的gate的group來跑sat。 因此我在做fraig的時候，會希望 沒有更動到DFS的順序(雖然這樣 會因為circuit的大小沒辦法動態改變 降低simulate的效率，不過其實也可以找一個 機制來讓circuit每reduced到某一個程度時 就跑一次merge+一次reset一次DFSList)

接著來來看一下在這個class中存了什麼data：

1. ofstream* _simLog;
2. CirGate** _gatePtrList; 存放CirGate的address。
3. size_t _size; 存放電路中input或是aig或是undef的id的上限。
4. size_t _iSize; 存放input的個數。
5. size_t _oSize; 存放output的個數。
6. size_t _aSize; 存放aig的個數。
7. vector<size_t> _iVec; 依照讀入順序存放input的id。
8. vector<size_t> _oVec; 依照讀入順序存放output的id。
9. vector<CirGate*>* _FECBroListPtr; 存放有效group的head。
10. vector<CirGate*> _DFSList; 依照DFS的order存放gate的address。
11. vector<pair<size_t, CirGateV>> _fraigMergePool; 給fraig存放merge的資訊。
12. bool _isSimFlag; 紀錄有沒有sim過了。

- CirGateV： 這個class其實不只是給input用而已，在很多地方 每當需要invert的資訊時，都會用到這個class。 在這個class當中只有存了一個型態為size_t 的data。

- StrashKey： 在strash的時候我是用hashSet來判斷兩個gate 的input是不是一樣的。而值得注意的點是，我的 hash function是將兩個input的address 轉成size_t後相乘。也就是說，兩個gate的 input在不管順序下，如果是一樣的，那麼他們的 hash值會是一樣的，不過這樣子因為沒辦法判斷 invert，所以在overload ==時就要記得implement。

- SimKey： 在sim的時候我是用hashSet來判斷兩個gate的 _simValue是不是相等的。而在這裡我的hash function 就只是回傳_simValue或是 _simValue，而 為讓simvalue完全相反的兩個gate也會變成 FEC所以我必須確保每一個hashvalue的某一個 bit的值固定為0或是1。但是這樣就有一個缺點， 就是如果a、b這兩個gate原先的_simValue是相反的，結果下一次的_simValue卻一模一樣， 在這裡的sim沒辦法抓出這兩個gate是不一樣的。

接著來看我在實行這次作業中所要新增的commend時所使用的的 data structure跟alogorith。

A. sweep

對於這個command，我只需要注意gate之間的連接不要壞掉就好 因此，我選擇在gate當中新增一些關於break連接的 的member function，來確保我能夠 比較輕鬆的來保持整個電路的connection不會壞掉。 接著，我只需要把DFS找不到的gate 都將他的input跟output的connection都break掉。 之後記得delete掉就好了。

B. optimize

跟剛剛的sweep很像，但這次我需要在CirGate多新增一些 關於connect的member function。 之後在寫一個在cirMgr中可以拿來merge的 member function(概念上就是跟sweep一樣 要注意connection不能壞掉)。 這樣子我只要依照DFS的順序 來將每個gate都check過一遍就好了， 一樣要記得將被merge掉的gate delete掉就好。

C. strash

為了能有效率的(希望是O(1))找到所有 有等價的input的gate，老師教我們使用hashSet的方式。 因此，我會先創一個hashSet<StrashKey>的物件， 並且依照DFS的順

序，將每個gate的address用StrashKey 的物件去query hashSet，如果有一模一樣的input的話， 那麼我就用先前寫過得merge這個member function 來將這兩個gate merge起來。不一樣的話就insert進去。 值得注意的是，我必須決定一個hash fucntion 來決定每個gate的key。 而因為這個問題其實是個對稱的問題，因此我覺得 使用對稱的hash function會比較好。所以我就使用了 前面講過得方法(將兩個input相乘)來當作我的hash key。

D. simulate

首先，我必須模擬整個電路， 為了達到較高的效率，老師教我們一次模擬64個bits。因此在-r這個指令中 我使用rand()這個std的隨機生成函數和一些bitwise的操作 來幫我產生一個型態為size_t的隨機數。 而在-f這個指令中，我則是一次讀一行， 並且使用一些bitwise的operator來將讀到的值加到 PI的_simValue上。 在進行模擬的時候，只需要按照DFS的順序來將每個gate 都模擬過一遍就好了。這樣子能夠確保每個可達到PO的gate都能正確的模擬。

接著我必須給FECgroup一個initialize 而我的設定必須是 大家都屬於同一個group才行，否則可能會有functional equivalent的兩個gate一開始在init時就被拆到不同 的group。而對於這一個init我選擇的是const gate 當作是一開始大家的head。

再來在我檢查我每個FECgroup之前，我會先new一個 新的vector來存等等每個group的head。new完之後 我需要檢查在我每個FECgroup當中有哪些gate應該是 屬於同一個FECgroup。因此仿造先前strash的作法，我使用 hashSet的方式，同樣的，我會將每個gate的Address用SimKey的物件去query，而在這裡的hash function，我就沒有 再多想了。(我在這裡並沒有打算要來寫能夠區分兩個gate 在這一次跟上一次simulate時的_simValue的值的關係 是不是不同(一次為FEC一次為IFEC)的SimKey，因為 要做到這點太麻煩了，我想說交給SAT去判斷就好。) 接著如果有一模一樣或是相反的_simValue的話，那麼我就 將兩個連起來，如果找不到就insert進去， 並且將這個key存的address丟進去我剛剛new的vector當中。在最後整個_FECBroVList中每個group都檢查 完成後，我就需要將原本的_FECBroVList delete掉，並 接上我新的_FECBroVList。 接下來就是重複以上動作數次。

最後因為我在一開始考慮FECGroup的資料結構時， 真的是考慮不周，導致麻煩的事情一一出現。 第一個出現的問題是，我的group如果一開始是依照 (0,1,2,3,4)的順序連接的話(這裡的數字是指index， 括號代表括號裡的數字屬於同一個group， 並且括號中的第一個index代表每個group的head)， 那麼有可能會有以下情形發生：
$(0,1,2,3,4) \rightarrow (0,2,3),(1,4) \rightarrow (2,3),(1,4)$
因此我的_FECBroVList中的順序並不定是會從 小排到大。因此在最後的時候還需要進行排序這個動作。

E. fraig

最後這個fraig也因為我FECGroup沒有選好資料結構， 而導致必須花費效率去換取正確率。 在寫fraig的時候，其實參照老師給的example，很容易 就可以寫出來了，就是一開始先創一個新的satsolver的 物件，接著將他initialize，然後把所有DFSList中的 gate都能在satSolver中正確的連接。接著就是要去 解sat problem直到沒有有效的FECGroup為止。在這裡 我選擇的演算法是根據DFS的順序下去檢查每個gate是沒是 待在一個有效的group，如果是的話，我需要先將他 變成整個group的head，接著將他和他的下一個functional equivalent candidate去解sat，如果無解的話， 那代表這兩個gate是可以merge的，那麼就把這個merge 的資訊丟到_fraigMergePool裡面； 如果有解的話，那麼代表可以被猜開，但是這不代表他下一個的functional equivalent candidate 並沒有functional equivalent的gate，因此我需要將這個gate先丟進去 一個容器，之後做一次模擬(將拿到這次有解的值 傳到input的的第一個bit當中(其他bit 會用rand()來生成，以免浪費)) ，等待這個head跟後面的所有的functional equivalent candidate都解完sat後，如果容器 內的gate能夠組成有效的group那麼就將他們連起來， 在丟回去_FECBroVList當中。這樣子一旦整個DFSList都跑完後，我就能確保 整個_FECBroVList已經是空的了。

III. 效率問題

A. sweep、optimal跟strash

對於sweep、optimal跟strash其實沒什麼效率問題， 這三個中比較有效率問題的可能會是strash。因為 其實我不曉得用相乘來產生key會不會不夠散，不過 經過一些測試(包掛兩個input做XOR)後，我還是決定要用 乘法。

B. simulate

對於simulate的效率問題，我覺得其實也沒什麼問題， 我在此提出幾個我比較有問題的點。第一、如果我的 SimKey能夠分辨前一次跟這次的_simValue的關係 是不是一樣，那麼我在Fraig解sat完後如果是有解， 我就不用將gate丟到容器後等等還要重新連，我只要 直接進行模擬就好了。第二、當然就是跟strash一樣 hash要開多大，還有-r時如果 沒有辦法用模擬分出全部的gate 那麼要模擬幾次？而最後我經過幾次 比對後，我選擇hash都開得跟要group的大小一樣， 這樣對於大的group來說可以提高分散的機率，對於小的 group也不用花太多時間在query上。至於要模擬 幾次，最後我選擇如果有10次沒有模擬成功(我這裡的成功有比較特別的定義方式，但是 deadline要到了，就先不做解釋)，我就 不模擬了，因為對於大的circuit要失敗，其實很困難， 所以自然可以做很多次，而對於小的circuit， 也可以很快的就結束，並不用等太久。再者，接下來的步驟 fraig裡面也會有sim，所以我認為並不用擔心sim的次數 不夠。

C. fraig

接著是fraig的效率。我覺得在問這個問題之前， 最主要還是要去了解一下sat.h裡面到底是怎麼運作的。 但是由於時間不足，並沒辦法這樣做，在此我先提出 我對於fraig效率的看法。 首先，我注意到在跑sim13.aag的fraig時 在一開始解sat時速度特別的慢。這可能是因為 sat有學習機制或是一開始因為是解const[0] ，又const[0]所待的group非常的大所以速度才會這麼慢 (因為隨後有在做sim)，但最後真正的原因 實在是不確定。再來是因為我merge是留到最後 才一次merge，這樣子會導致在做sim的時候速度不會變快。 (因為會有一個動作是做DFS，所以如果我有辦法在適當的時機 進行merge以及DFSList upload的動作，那麼 速度有可能可以提昇）。

D. coding

最後是時間分配，因為我期末考完後才開始趕這個功課， 所以導致一直寫code的時間很多，但是這麼作對我而言， 是會使我的效率大幅降低的，不管是在考慮要使用 什麼演算法或是資料結構時，都會很容易有誤判， 因此在寫完這次這麼大的功課後，我學到應該要 學習分散時間來寫code，而要做到這點，我覺得最重要的 就是code要能夠寫得易讀，否則如果像這次FECGroup 沒有用好的方式寫，在短時間內都很容易導致一堆小bug

handle不完了，更何況時間拉長。

---

* b05202044@ntu.edu.tw