資料結構與程式設計

Final Preject

系級：　電機二
姓名：　黃禹傑
學號：b04901146
Email : b04901146@ntu.edu.tw

一、資料結構

1. class CirMgr

   (1) int _header[5];

   記錄.aag file 之 M, I, L, O, A。

   (2) vector<unsigned> _gateId;

   以 gate Id 記錄.aag file 中 gate 定義的順序。

   (3) vector<unsigned> _floGateId1;

   以 gate Id 記錄 gates with floating fanin(s)。

   (4) vector<unsigned> _floGateId1;

   以 gate Id 記錄 gates defined but not used。

   (5) vector<unsigned> _DFSList;

   以 gate Id 依序記錄 DFS search 的 gates。

   (6) vector<FecGroup> _fecGroups

   記錄全部 FecGroup 的 vector。

   (7) map<unsigned, CirGate*> _gate;

   在 map 中透過 gate Id 取得 gate 位址。

   (8) mutable unsigned _printRefGlobal;

   呼叫 print 時使用的 ref。

   (9) unsigned _DFSRefGlobal;

   進行 DFS 時使用的 ref。

   (10) bool _simed;

   記錄是否 sim 過而尚未 fraig。

2. class CirGate

   (1) GateType _gateType;

   gate 的類型。

   (2) string* _gateName;

   gate 的名稱。若沒有名稱則為 NULL。

   (3) unsigned _lineNo;

   記錄 read 時 gate 是在 aag 中哪一行被定義。

   (4) unsigned _gateId;

   gate 的 ID。

   (5) unsigned _fanin[2];

   PI_GATE / UNDEF_GATE / CONST_GATE：皆不使用。

   PO_GATE：以_fanin[0]代表 output 來源。

   AIG_GATE：分別代表兩個 fanin。

由於是存取 fanin gate 之 ID，因此以兩倍 ID 作為基準，若有 invert 則+1。

(6) unsigned _DFSRef;

進行 DFS 時使用的 ref。

(7) unsigned _simValue;

最後進行的 sim 所得到的結果。

(8) unsigned _fecIdx;

目前被分在 CirMgr 中的_fecGroups 的哪一個 FecGroup 中，可以藉此找到其他同在一 group 中的 gates。

(9) mutable unsigned _printRef;

呼叫 print 時使用的 ref。

(10) vector<unsigned> fanout;

記錄輸出至哪些 gates。

(11) bool _inDFS;

代表是否在_DFSList 中。

3. class GateNode

在 strash 時將 gate 包起來丟進 hash。因此需 overload operator==、operator()…等。

(1) CirGate* ptr;

單純記錄著 gate 的位址。

4. class FecGroup

將目前為止 sim 結果相同的 gates 包起來，在 hash 中操作較方便。

(1) vector<CirGate*> _fecGate;

記錄著至目前為止 sim 結果皆相同的 gates 們。

(2) unsigned _simValue;

至目前最後一次 sim 所得到的 value。

二、演算法

1. Sweep

做一遍 DFS 並 mark 有走過的，最後沒被 mark 到的都刪掉。

2. Optimize

對於所有 AIG_GATE A：

(1) 若 A 之任一 fanin 為 const 0，或是兩 fanin 互為反向，則將 A 的 fanout gates 相對應之 fanin 設為 const 0;

(2) 若其中一 A 之 fanin 為 const 1，或是兩 fanin 相同，則將 A 的 fanout gates 相對應的 fanin 設為 A 的 fanin。

3. Strash

   按照_DFSList 中的順序檢視，以 gate 的兩個 fanin 為 key 丟入 hash，若有一樣的便可以將兩者簡化，並選擇_DFSList 中較前面者保留，後者刪除。

4. Simulate

   (1) pattern 使用的是(PI 個數)個 unsigned int，將其餵給 PI_GATE，接者以 DFS 跑完其他 AIG_GATE，並將 simulate 的結果存在 gate 中。

   (2) 接著抓出原本就在同一 FecGroup 中的 gate 來檢視，一一包成新的 FecGroup 後丟進 hash，若是遇見相同的便將兩者 merge，如此一來便可以得到新的_fecGroups。

5. Fraig

   (沒有完成)

三、測資

Sweep、Optimize 以及 Strash 的測資都測過並通過了。Simulate 的部分的測資寫出檔案也都和 reference code 一樣，但是分 fecgroup 的速度有點慢。

四、瓶頸

1. 在 sim13.aag 中，可以發現 simulate 的效率不太好，要跑很久，仍在加速中。
2. 由於效率不太好似乎也無法跑很多遍所以 fec groups 的數量會比較多…
3. fraig 無法完成…

五、參考

都自己寫的沒參考別人的。