資料結構與程式設計

Final Project Report (FRAIG)

姓名 : 吳岳

學號 : b02901093

Email : b02901093@ntu.edu.tw

手機 : 0972871931

一、功能設計

- **Sweep**

1. Key Element & Function:

- cirGate.h

bool _sw;
void setSw(bool s)
bool isSwList()

2. 描述

所有的gate在read_circuit時都會存一個_sw的標記，並預設成true，之後判斷該gate屬不屬於DFS List。若屬於DFS List，_sw = false，不屬於DFS List gateType != PI，將gate搜集入預設的gatelist。
(Sweep Gate List)

首先先將gate對應fanin跟它連線刪掉，重新整理GateList的排序，最後再將_sw = true 的 gate刪除。

- **Optimization**

1. Key Element & Function :

- cirOpt.cpp

faninJudge(CirGate g, GateList& opt)*
disConnect(CirGate g, unsigned& i)*
setNewConnect(CirGate g, CirGate* gi, size_t& phase)*

- cirGate.h

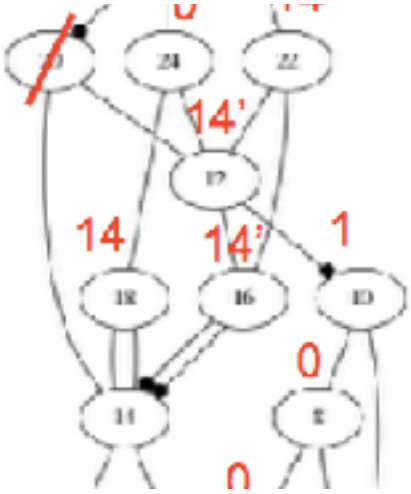
bool _optDel;
void setOptDel(bool s)
bool isOptDel()

2. 描述：

loop整個DFS List的gate，在**faninJudge()**裡，每個gate都會判斷4種trivial optimization的條件。若符合，**disConnect()**將連線拆掉。

再者，gate跟即將取代它的gate (ex: fanin, Const0)、以及phase都會被送到**setNewConnect()**，將gate的fanout跟要取代的fanin相連。

以老師簡報裡的例子說明，gate 18會被14取代，而我先將14連到18的fanout刪掉取代成24 (14 -> 24)，再將24連到18的fanin改成14(24 -> 14)。這樣做可以讓trivial optimization 一直沿著DFS List做下去。



最後重新整理 GateList的排序，將_optDel = true 的gate刪除。

- **Structural Hash**

1. Key Element & Function :

- myHashMap.cpp

  template <class HashKey, class HashData>

  ***class** HashKey, HashData, HashMap*

  vector<HashNode> _buckets;*

- cirGate.h

  bool _strHhDel

  void setStHDel(bool s)

  bool isStHDel()

2. 描述：

Hashkey裡面存兩個對應的fanin，**HashData**則是以size_t的資料型態包含phase存對應gate，”**HashNode**”是HashKey跟HashData的pair。

loop整個DFS List的gate，每個gate的fanin都會跟存在HashMap裡的HashNode比對，若兩個fanin不一樣，建構新的HashNode將fanin 跟 gate + phase存起來。若兩個fanin都一樣，就會將gate的 _strHhDel設為true。參照optimization後段同樣的方法先 **disConnect()** 再 **setNewConnection()**

- **Simulation**

1. Key Element & Function :

- **cirGate.h**

  void *getValue()*
  bool *getPreVal()*
  void *setValue(bool v)*
  void *setValToPre()*
  bool _value;
  bool _preValue;
  pair<unsigned, unsigned> _bukSl;

- **cirSim.cpp**

  piValSet(CirGate* g, char& pi)
  defValue(CirGate* g, int& it)
  setHash(CirGate* g, int& it, int& col)

- **cirDef.h**

  class FECGroup

2. 描述：

每次新pattern進來，都會用**piValSet()**將pi初始化，**defValue()**設定DFS List裡gate的 value。

每個get都會有**value & prevalue**，原理跟flip-flop很像，跑pattern時，會將上個state的值存到prevalue，新的值存成value。

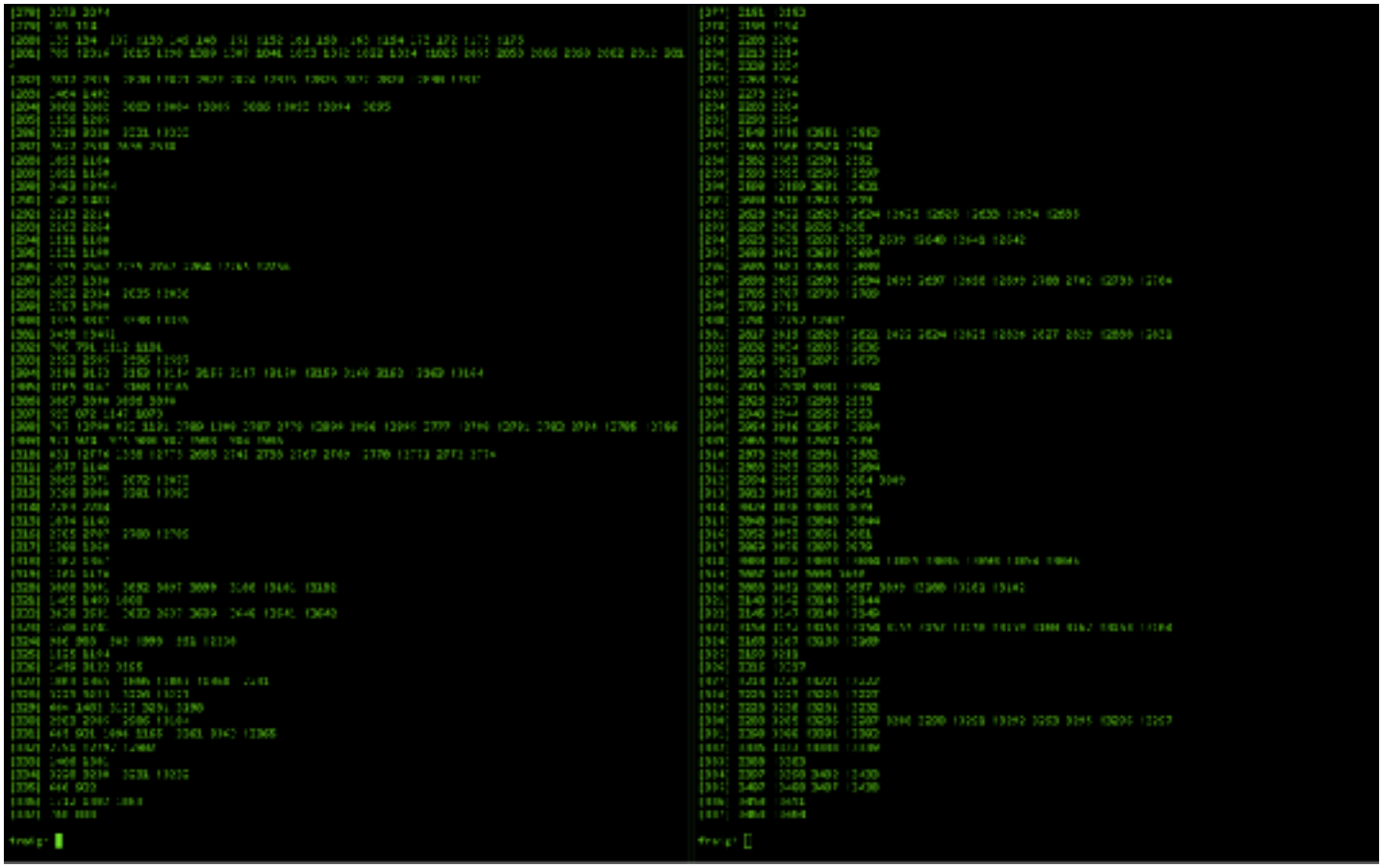
之所以需要preValue跟Value，是因為**要判斷FEC & iFEC group**，第一次跑pattern時，所有的gate都會存在hash的第一個bucket裡面。

從第二次pattern開始，每個bucket的第一個slot是參考gate，其他的bucket都要跟它比較preValue & Value。

如果B.value == A.value && B.Prevalue == A.Prevalue 合理猜測B可能跟A是FEC pair，相反的如果B.value != A.value && B.Prevalue != A.Prevalue，那他很有可能彼此是iFEC pair。

如果其中一個相等而另外一個不等，那它將不屬於A的FEC group。將拆夥產生新的bucket，同樣的，第一個slot的gate是參考點，之後進來的gate在每次有新pattern時都會跟它比較。

最後print FEC pair 得到的結果跟Reference Code一樣，(但ID的排列並沒有按照順序, 如下圖操作sim09.aag範例，得到的總FEC pair數目一樣、pair的成員也相同)



- **Fraig**
  1. Key Element & Function :
     - **cirFraig.cpp**

       fraigLoop(CirGate* g0, GateList& fecList, SatSolver& solver)

       SATSetHash(CirGate* g, bool result)

       genProofModel(SatSolver& s)
  2. 描述

     我們是按照Dfs search的方式去判斷FEC group的成員跟第一個成員之間有沒有SAT，因為有範例的code，在實作上並沒有發生太大的困難。但因為CONST_Gate並不在dfs list裡面，需要額外將它加到hast。

     對於那些SAT的成員，我的做法是將他們都放到新的hashSet裡面。讓成員在跟第一個成員

(**simHach[i]->getGate(0)**)判斷有沒有SAT。跟老師的code比起來，我的code會慢很多，worst case狀況下，第一個成員跟其他成員都是SAT，將會有n!的複雜度，但可以確保不有漏網之魚。

二、修課心得

大三上時就很想修姿婕了，但害怕之下卻步了。那一年修了不少的實驗&選修，每到要寫程式總有種使不上力的感覺，後來心一橫，決定在大四跟姿婕好好大戰一場。

然而，修完之後並沒有戰勝的感覺，上課教過的很多內容老師說很多都忘了，除非作業有做到，不然它就在檔案夾裡面封塵。對於自己之後會不會走軟體相關這條路，也沒有明確的方向，只覺得程式是個我不可或缺的技能，從喜歡的技能去分類自己要做的事情太膚淺了，目前也沒有找到什麼很關注的議題，感覺都很新穎但都不是志業。

多年後，我可能忘記fraig的定義，但我一定不會忘記戰勝這門課前，老師的姿婕與人參，那不是無戰事下的嘴砲，而是大學被重課烙印後的教誨，更有歷練的鹹味。