資料結構 Final Report

機械四 鄭承昀 B03502106
b03502106@ntu.edu.tw

一、 Design for simulate

這裡大概可以分成三個部分:取得simulate的值(給PI)、進行simulate、決定FEC groups。

1. 取得simulate的值：

   在每個一個Gate裡都有一個unsigned long long _Sim的data member，用來存放其simulate的值。而作業要求有兩種simulate的方式，使用random：這邊我用random產生多組32bits的數字，並丟給PI，作為PI的初始值；而另一種從pattern file讀simulate的值，這邊方式一樣，先把string轉成unsigned long long再給到PI作為初始值。這邊有一個值得考慮的點就是在使用random時，要用多少個pattern進行simulate，太少會有simulate結果不準的問題，太多則會浪費時間。我採用的方式則是所要simulate的Gate數除2乘上64個pattern。

2. 進行simulate：

   我是採用All-Gate simulate的方式，依照_NetList的順序去進行simulate，以確保每一個AIG Gate再進行simulate前它的fanin都已經進行過新的simulate了。

   不過使用All-Gate simulate有個缺點就是，有的時候對同一個AIG Gate來說，其input可能根本沒變，但是卻還是要進行一次simulate，較浪費時間。但是我覺得這樣的演算法比較容易且直觀，所以還是採用這個方式。

3. 決定FEC group

   這裡我的想法很簡單，裡用HashMap對所有在_FECList裡面的Gate的_Sim進行分類。每加進一個新的Gate，就把它拿去HashMap裡面做check，如果HashMap裡面已經有這個值了，就將這個Gate push_back進那個FecGrp裡面；如果check的結果是沒有的話，就開一個新的FecGrp。

二、 Design for Fraig

Fraig的最基本的演算法就是藉由SatSolver對所有FecGrp裡面的Gate去證明是否真的為FEC。但是單只是這樣，在做fraig時的時間會拉得很長。我想了幾種方式：

1. 把每個 FecGrp 裡面的 Gate 依照_NetList 的順序排列。原因是離 PI 越近的 Gate 理論上應該越容易判斷出到底是不是 FEC，先證明出來的或許能幫助後面較複雜的進行證明。
2. 經過幾次證明後，可以做一次 resimulate，以把應該分開卻沒分開的 FEC 分開。

三、實驗

1. 希望比較不同的 pattern 數量對電路簡化的影響，例如：執行速度與簡化結果
2. 比較執行 Fraig 時，有無加入排序、有無 resimulate 對執行時間與結果之影響

以上是我對 simulate、fraig 的規劃設計，以及希望能做這兩個實驗去進行測試，但是最後這兩個部份我並沒有成功完成，所以我並沒有實驗結果...