田昀曜 B07505026 b07505026@ntu.edu.tw cellphone: 098896567

期末專題報告

Data Structure & Algorithms:

我在 CirMgr 裡使用 vector<CirGate*> CirCuit 儲存全部 gate 的指標。並且有另一個 Map 來對應 gate 的 Id。

而在 CirMgr 裡面有 vector<vector<int> > fecGroup，用來儲存 FEC 和 IFEC 的資訊，並且在 CirGate 裡存了這個 gate 在 fecGroup 裏頭的 index，可以快速找到這個 gate 有哪些 FEC 和 IFEC。

在需要刪除很多 gate 的情況時，為了維持 CirCuit 的正確性，如果一直使用 erase 太慢了，所以我有先將要刪除的 id 記下來，再從 CirCuit 的最後面開始跑， 並且使用 swap 和 pop_back 來刪除。而我有在 cirGate 中設計幫助刪除的函式，用來處理這個 gate 的 fanin 和 fanout 的連接。

Simulation:

由於 fileSim 和 randomSim 都要使用類似功能，因此利用一個 simulate 函式來處理。我是使用 bitset<PatternSize>來當作 parallel pattern。並且在讀取滿了時候進入 simulate 處理。

而在 simulate 中，我進行 update value -> hash into tmpGroup -> push_back into fecGroup 的流程。

其中 hash into tmpGroup 我會先判斷是第一次或不是第一次。如果是第

一次的話我會從 netList 來 hash，為了處理同一個 group 但要區分 FEC 和 IFEC 的情況，我讓 hashkey 為 FEC 的值，IFEC 則是使用加一個大數(假設不會超過 10 億個 gate)來區分。

不是第一次的情況，就要利用 FEC 或 IFEC 的判斷來看要怎麼 hash。我先讓 FEC 的值 hash 進 tmpGroup，再用 IFEC 的相反值來加進 tmpGroup，如果沒找到就先不要 hash 並且存起來。完成之後先把 tmpGroup 丟進 fecGroup，然後清空 tmpGroup 再來處理 IFEC 的 hash。這樣來避免原本是 IFEC 後來卻又剛好一樣的值的情況。(這個 bug 找了兩小時 QQ)

而在跑完一次 simulate 時，會重新把 fecGroup 裡 Id 的值變回正常的。在需要判斷是 FEC 還是 IFEC 的時候只要用利用我存在每個 gate 中的 gateValue 來判斷即可。

FRAIG:

在遍歷 fecGroup 的每個 group 時，我使用 visited 的來記錄是否已經檢查過了，然後在每個 group 中每兩個 gate 進行比對，需要刪除時我也是先將 Id 存起來，並且特別紀錄需要這個 gate 已經被刪除了。最後利用 swap 和 pop_back 來刪除。不過因為一值找不到 bug，所以我沒有寫完這個功能，只有一些測資會過。

Final ProJect 心得:

我覺得我在 simulation 上花了太多時間，因為 sim13 一直會錯，我花在

sim13 的時間就有兩天了。如果先來把後面的 fraig 弄好，成績應該會好很多。不過那時候實在是太想找出哪裡有問題了，所以就一頭栽進去沒有注意到剩餘的時間。

Comments:

我覺得老師在課堂中講的一些寫 code 的技巧和觀念都非常厲害，如果以後只有 github 上的教材自學的話，可能就聽不到這些了，有點可惜。