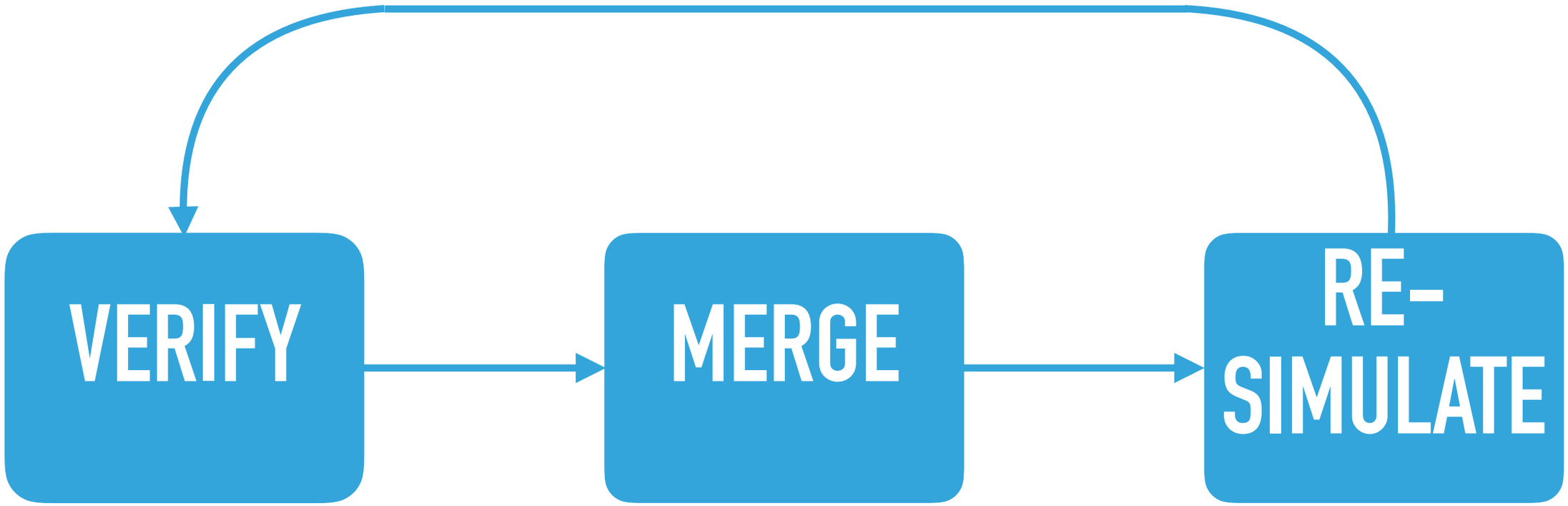
B03901052 王傑生

DSNP - FRAIG

FRAIG 整體流程



VERIFY – 分別處理FEC PAIR 及 FEC GROUP

- 對於FEC PAIR，直接建立proof model，若兩gate確實等價則丟入merge list中。
- 對於FEC GROUP，則分別將各個gate與group中的第一個gate組成pair進行驗證，最後將第一個gate與所有與其等價的gates組成一個group丟入merge list中。
- 剩下的gate數量若大於等於2個，則繼續留在FEC list中。
- merge list及FEC list的結構皆為list<vector<unsigned>*>，使用list及pointer便於插入、刪除以及搬移。

VERIFY 進行的優先順序（PRIORITY QUEUE）

- 建立priority queue決定verify進行的順序。
- 最初以DFS order作為priority。
- 可針對未驗證完的group，動態調整其priority。
- 具備擴充性，若有更好的priority依據，可進一步增加FRAIG的效能。

VERIFY – 中斷機制

- 當從SAT engine得到一定數量的可將FEC pair分開的input pattern時，則中斷verify，進行merge及re-simulate。
- 若為進行到一半的FEC group驗證時，則將group中已驗證等價的部分放入merge list，剩下的部分連同group中的第一個gate放回FEC list中。
- 設定一個驗證次數上限，當達到上限後，不論取得了多少pattern，都先中斷驗證，進入merge。

VERIFY – 利用SIMULATION 加速 FEC GROUP的驗證

- 在進行FEC group驗證時，每取得幾組有用的input pattern時便做一次simulate，利用simulate的結果，若兩gate simulate值仍相等才進行驗證。
- simulate一次所需的時間遠少於進行一次FEC pair的驗證。
- 不會更新FEC list，不會影響到group外的gate，雖會造成同樣的pattern被重複simulate，但整體仍有非常正面的效益。

MERGE – 利用HASH TABLE確保MERGING正確性

- 若merge的次序錯誤，可能會導致電路產生cycle。
- 仿造Strash的方法，只是改以各個gate所屬的pair/group在merge list中的index作為hash key，按照當下的DFS order將gate放入hash map中，若已存在中merge。
- 若為invert FEC，則將index變為~index，在check hash map時兩種情況皆進行檢查。

MERGE – 利用先前的TRIVIAL OPT. 以及STRASH

- 在將merge list中的pair/group皆merge完後，電路中可能產生許多const，或是不同的gate有相同input的狀況。
- 直接以trivial optimization以及Strash化簡這些情形，而不必再使用verify -> merge的方式。
- Trivial opt.及Strash所需的時間皆遠少於SAT engine進行verify所需的時間。

RE-SIMULATE - 清除FEC LIST中已不存在的GATE

- 在merge之後，部分FEC list中的gate可能已被清除
- 在進行re-simulate之前，先更新一遍DFS list後，利用hash map將FEC list 中不在DFS list的gate清除。
- 刪除group內的某個gate的方法為，將vector中的該element與最後一個element互換後pop back，避免array直接刪除O(n^2)的複雜度
- 全部清除後再將各group內的gate依ID重新排序

繼續改進的方向

- 找出更好的驗證priority
- 利用如BFS等方式，在不會造成cycle的前提下選擇更好的merge次序，而不單純只依照DFS order來merge。
- 進行更多電路的測試，對前述各步驟中可調整的部分，找出更好的參數。