Fraig:

Functionally Reduced And-Inverter Graph

Name: 施昱安

School ID: B05901010
Email address: b05901010@ntu.edu.tw

Data Structures

1. Inheritance of class CirGate

   將 gate 分成 PI, PO, constant gate, Aig gate 四種，根據其特性執行不同的功能。

2. Several gate lists

   _gates: 依照_id 的大小，存放所有的 gate pointer。

   _PIList: 紀錄每個 PI 的_id，以便從_gates 中取得。

   _dfsList: 將所有能抵達 PO 的 gate 按照 depth first search 的次序排列。

   _unusedList: defined but not used gates.

   _undefList: referenced but not defined gates.

3. Class Hashmap

   (1) 執行 Strash 功能時，幫助配對 fanin 相同的 gates。

   (2) Simulate 電路時，用來存放 FEC groups。

4. For FEC groups

   _fecGrps: vector of FECGroup

   FECGroup: vector of pointer of SimKey

   SimKey: 每個 gate object 中有兩個 SimKey，分別代表 output signal 及 inverted output signal，在做 simulation 時作為 hash 的 key 使用。

Algorithm for Simulations

Random Simulation

1. 將 constant 0 gate 和_dfsList 中所有 Aig gate 的兩個 Sim Key 放入一個 FEC group 中。
2. Simulate the circuit: parallel-pattern simulation
   (1) Pattern generation: 利用 rnGen()生成亂數，決定 PI gate 的_value。
   (2) 依照_dfsList 的順序進行 simulation。
3. Divide FEC groups: 利用每個 gate output 的值，將原有的 FEC group 區分裂成更小的 groups。
4. 重複步驟 2, 3，直到連續 10 次無法切割任一個 FEC group。
5. Sort _fecGrps: 使其符合輸出的格式

File Simulation

1. 將 file 中所有的 pattern 存入 two-dimensional vector 中。
2. 將 constant 0 gate 和_dfsList 中所有 Aig gate 的兩個 Sim Key 放入一個 FEC group 中。
3. 檢查 pattern 的長度及內容是否有誤。
4. Simulate the circuit
   (1) 將一堆 pattern(size of size_t)包入 PI gate 的_value 中
   (2) 依照_dfsList 的順序進行 simulation。
5. Divide FEC groups
6. 重複步驟 3~5，直到所有的 pattern 使用完畢或任一 pattern 的格式錯誤。
7. Sort _fecGrps: 使其符合輸出的格式

Results and Analysis

1. Correctness of Sweep and Optimize: 使用 opt01.aag~opt07.aag 進行測試，結果都與 reference program 一致。然而，執行 Sweep 時刪除 unused gate 的順序可能與 reference code 不同(ex: opt07.aag)。
2. Correctness of Strash: 使用 strash01.aag~strash07.aag 進行測試，結果都與 reference program 一致。
3. Performance of Strash and Simulation:

   Strash:

Time usage(s)	sim06.aag	sim07.aag	sim08.aag	sim09.aag	sim10.aag
Reference	0	0	0	0.01	0
My code	0.01	0.01	0	0.01	0.01
Memory usage(MB)	sim06.aag	sim07.aag	sim08.aag	sim09.aag	sim10.aag
Reference	0.457	0.422	0	0.4491	0
My code	0.398	0.371	0	0.371	0

   觀察上表，可以發現我的 code 需要話較久的時間來執行 Strash，但需要的 memory 較少。

   Simulation:

Time usage(s)	sim06.aag	sim07.aag	sim08.aag	sim09.aag	sim10.aag
Reference	0	0.01	0	0.01	0
My code	0.01	0	0	1.71	0.22
Memory usage(MB)	sim06.aag	sim07.aag	sim08.aag	sim09.aag	sim10.aag
Reference	0	0	0	0.234	0
My code	0	0	0	0.539	0

   *紅字代表與 reference program 的輸出不同

   觀察上表，可以發現我的 code 不論是在所需時間還是記憶體的表現上都較 reference code 差。

4. 在跑 simulation 時，一開始採用 event-driven simulation，但因為還是得每個 gate 都看過，反而沒有 all-gate simulation 來的簡單快速，所以後來就改回使用 all-gate simulation 了。
5. 每使用一組 pattern 進行 simulation，都必須將每個 FEC group 中的元素放到 hash 中重新分類，應為整個 program 所需時間跟記憶體的主要來源。而這不僅與 gate 的數量有關，也受到 FEC group 數量的影響，因此挑選不同的 pattern 也會造成不同的結果。