Data Structures and Programming

Functionally Reduced AND-Inverter Graph Report

B06901601 | 王廷峻 | 0976-209-552

I. Circuit Parser

Class : cirGate		
	Private Data Members	
Data Type	**Variable**	**Description**
unsigned	_id	Every gate has its distinct id
unsigned	_lineNum	Line of a read-in file
mutable unsigned	_ref	For graph traversal
size_t	_value	Simulation value
string	_symbol	Gate name
Var	_SATid	Endowed by SAT engine
vector<unsigned>*	_fecPos	Pointer to its FEC group if exists
vector<GateV>	_fanIn	FanIn list
vector<GateV>	_fanOut	FanOut list
Auxiliary Class : GateV		
	Private Data Members	
size_t	_gate	Containing address of a cirGate and its invert information
	Public Member Functions	
CirGate*	gatePtr ( ) const	Pointer to a cirGate
bool	is_invert ( ) const	A inverter between two gates
unsigned	getID ( ) const	ID of pointed gate
size_t	getSim ( ) const	Simulation value of pointed gate
Derived Class (Public Derived)		
PiGate	Primary-In	Input gates
PoGate	Primary-Out	Output gates
AIGate	And-Inverter	
FloGate	Floating	Undefined gate with no fanIn
ConstGate	CONST 0	Always gives 0

II. Circuit Manager

Class : CirMgr

Private Data Members

Data Type	**Variable**	**Description**
SatSolver	_solver	SAT engine
ofstream*	_simLog	
vector<CirGate*>	_GateList	Capacity is equal to the sum of all gates, acts like a hash. (Index corresponds to gate ID)
vector<unsigned>	_PiList	ID of primary-in gates
vector<unsigned>	_PoList	ID of primary-out gates
mutable vector<unsigned>	_FlList	ID of floating gates
mutable vector<unsigned>	_NUList	ID of defined but not used gates
mutable vector<unsigned>	_DFSList	ID of gates, which can arrive from POs, in the order of depth-first-search
mutable unordered_map <unsigned, unsigned>	_DFSmap	ID of gate is key, index of the _DFSList is its corresponding value. (Use to sort the vector in _fecGrps)
unordered_map <string, CirGate*>	_hash	Key : combination of its input literal (2*ID + invert) (eg. 25_31) Value : pointer to the gate with specific input pattern
vector<vector <unsigned> >	_fecGrps	Each vector corresponds to a distinct Simulation value. In a _fecGrps[i] contains **Literals of gates** with the same simulation value.
vector<size_t>	_CEXpattern	64 counter-example patterns derived from the SAT engine
mutable unsigned	_miloa [5]	Information of a circuit
mutable unsigned	_effAIGNum	Number of AIGs on DFS-path
mutable unsigned	_totalGate	Number of all gates
string	_header	Read-in file name

III. Sweep

Sweep( )	
Functionality	Delete gates which are defended but not used (i.e. gates with no fanout)
問題解析	刪除 Unused gates 之後需遞迴向前檢查是否變成 Unused gates

函式架構：

1. 利用 _NUList (defined but not used gates) 儲存的編號，遞迴呼叫 sweep(CirGate*)。
2. 判斷 Gate 類型：僅有 **_fanOut 的大小非零的AIGs** 或 **Floating Gates** 會進入 sweep(CirGate*)。
3. 若為 Floating Gates，則釋放 _GateList 中的指針；若為 AIGs，則執行以下步驟：
    - 將 gate 中的 _fanIn gates 的 _fanOut 更新，斷開兩個 gates 之間的連結。
    - 釋放 _GateList 中的指針。
    - 針對 gate 的 _fanIn gates 遞迴呼叫 sweep(CirGate*)，重複步驟二。

Relevant Member functions	
void sweep( )	提供 Commander 呼叫 Sweep( ) 的介面，並在 Sweep( ) 之後維護受更動的資料（_NUList, _FIList）。
void sweep(CirGate*)	提供遞迴呼叫的函式

IV. Trivial Optimization

Optimization( )	
Functionality	Substitute gates on DFS path with specific fanin (i.e. Identical fanin, Inverted fanin, const0, const1)

函式架構：

1. 利用 _POList 儲存的編號，從 output 端遞迴呼叫 optimization(CirGate*, vector<bool>&)。
    - _DFS
2. 利用 vector<bool> 紀錄 gate 是否已經完成 optimization，完成後為 true。
3. 判斷 optimization cases : NA, identical, invert, const0, const1
    - case : identical, invert 僅需要針對一個 _fanin 遞迴呼叫optimization(CirGate*, vector<bool>&)
    - case : NA, const0, const1 需要對其他 _fanin 遞迴呼叫optimization
4. 依照五種類型做下述步驟
    - N/A   vector<bool> 將此位置改為 true
    - identical
        a. 更新 _fanOut 中的 gate 的 _fanIn，將原有 this gate 位置以 this gate 的 identical input 取代
        b. 更新 identical input 的 _fanOut（將原先在 this gate 的 _fanOut 新增到此）
    - invert, const0
        a. 更新 _fanOut 中的 gate 的 _fanIn，將原有 this gate 位置以 this gate 的 identical input 取代
        b. 更新 CONST0 的 _fanOut（將原先在 this gate 的 _fanOut 新增到此）
        c. 更新 _fanIn gate 中的 _fanOut（將 _fanOut 中的 this gate清除）
        d. 若 dropped gate 為 Floating gate，則需要清除。
    - const1
        a. 更新 _fanOut 中的 gate 的 _fanIn，將原有 this gate 位置以 this gate 的非 const1 gate取代
        b. 更新非 const1 gate 的 _fanOut（將原先在 this gate 的 _fanOut 新增到此）
        c. 更新 CONST0 的 _fanOut

   釋放 _GateList 中的指針，刪除this gate 。

Relevant Member functions

void optimization( )	提供 Commander 呼叫 optimization( ) 的介面，並在 optimization( ) 之後維護受更動的資料（_DFSList, _DFSmap, _NUList, _FIList）。
void optimization (CirGate*, vector<bool>&)	提供遞迴呼叫的函式
Relevant Data Members	
enum Opt_case	‣ NA. : default ‣ identical : fanin are identical, replace with its fanin. ‣ invert. : fanin are identical but inverted, replaced with CONST0. ‣ const0 : one of the fanin is CONST0, replaced it with CONST0. ‣ const1. : one of the fanin is 1, replaced it with the other fanin.

V. Structural Hash

Strash( )	
Functionality	Substitute gates on DFS path with the entirely same fanin combination.
問題解析	對於相同 Input Combination 的 Gates 而言，若採用一次全部合併到其中一個 Gate，會產生「合併之後又和其他合併過後的 Gates 有相同 Gates」的問題，需要不斷合併。因此我採用 **Depth First Search 的方式合併 gates**，DFS 能確保「任何具有相同 Input Combination 的 Gates 都會被最靠近 PI 的 Gate 合併」，**可避免部分 AIGates 因合併而產生相同 Input Combination 問題**。

函式架構：

1. 進入 strash(CirGate*) 針對 fanIn 做 strash(CirGate*)，維持 DFS 合併順序。
2. 將 AIGs 的 fanIn 按照 Literal 的大小排序（i.e. 16_31），轉換成 string，作為 CirMgr :: _hash 的 Key。
3. 利用 unordered_map(string, CirGate*) :: find(Key)，檢查此 AIGs 是否存在 CirMgr :: _hash 中。
    - 不存在 CirMgr :: _hash：Insert(make_pari(Key, CirGate*))
    - 存在於 CirMgr :: _hash：利用原本存在於 _hash 中的 CirGate* 呼叫 mergeGates 將此 AIGate 合併

註：因為是 DFS，所以存在於 _hash 的 AIGate 會是最靠近 PI 的 AIG，可避免上述問題。

Relevant Member Functions	
void strash( )	提供 Commander 呼叫 strash( ) 的介面，並在 strash( ) 之後維護受更動的資料（_DFSList, _DFSmap, _NUList, _FIList, _hash）。
void strash (CirGate*)	提供遞迴呼叫的函式
void mergeGates (CirGate* merger, CirGate* merged bool)	Cirgate* merger : gate which merges other Cirgate* merged : gate be merged bool. : relation between merger and merged(i.e. invert | non-invert) 需要完成的更新： 1. merger 需繼承 merged 的 _fanOut，依照 bool 建構 GateV 2. merged _fanOut 需要更新 _fanIn，依照 bool 建構 GateV 3. merged _fanIn 須將 merged 從 _fanOut 移除

Relevant Data Members		
unordered_map <string, CirGate*>	_hash	Key : combination of its input literals (2*ID + invert) (eg. 25_31) Value : pointer to the gate with input pattern same as the key

VI. Simulation

Simulation	
Functionality	Given a 64-bit input pattern to each Primary input gate, each gate (i.e. PI, PO, AIG, etc.)will have a distinct simulation value. Each AIGs and CONST0 will be hashed into a particular _fecGrp (i.e. vector<unsigned) according to its simulation values.
Relevant Member Functions of CirMgr	
void randomSim( )	▸ Case 1 : Not yet calling CirMgr :: fraig( ) 依照全部的 gate 數量隨機產生 input patterns 並傳入 CirMgr :: doSimulation( )。 ▸ **Case 2 : Has already called CirGate :: fraig( )** 將 _CEXipPattern (vector<vector<size_t>>) 的每個 vector 元素傳入 CirMgr :: doSimulation( )。 **設計主因**：透過 SAT 證明兩個 gates 得到的 Input patterns 同時亦可使這兩個 gates 以前的部分 gates 分離，得到分散且量多的 FEC groups。相較於隨機產生的數列，為相當有效的 input patterns。
void fileSim (ifStream&)	1. 利用 local variable : vector<size_t> pattern，作為儲存讀檔後的輸入 **2. 利用 pattern[] 和 0, 1 做 bit-wise-OR，將 0, 1 從 pattern[] 的 LSB 開始存放** 3. 當存放 64 次後將 pattern 傳入 CirMgr :: doSimulation( ) 4. 因為 _fecGrps 改變，故維護在 _fecGrps 的 gates 的 _fecPos
void initFecGrps( )	1. 每次 simulation 之後需要維護每個 gate 的 _fecPos 2. 若 _fecGrps 為空，則須將 _DFSList 中的所有 AIGs 還有 CONST0 放入 _fecGrps，作為 initialization，以供 CirMgr :: doSimulation( ) 使用。 3. 若 fecGrps 非空，則不須初始化 fecGrps。因為 _fecGrps 會隨不斷 simulation 改變大小和內容。
void doSimulation (const vector<size_t>&)	Given a collection of 64-bit input patterns of each gate (i.e. vector<size_t>, size of vector<size_t> is equal to number of primary inputs)
1. 建立 _DFSList	按照 _DFSList 的順序分別對每個 gate 做 CirGate :: simulation( )
2. 針對原有的 _fecGrps (vector<vector<unsigned>>) 更新	針對每一個 _fecGrp (vector<unsigned>_fecGrp。儲存具有相同 simulation value 的 gate 的 literal) 創建 unodered_map<size_t, vector<unsigned>> hash。
	將每個更新 simulation value 的 gate 依照 simulation value (size_t) hash 進入 newFecGrp(vector<unsigned>)
	搜集所有 size 大於 2 的 newFecGrp，並裝入 _fecGrps (vector<vector< unsigned>>)
	刪除進入迴圈更新前 _fecGrps 的部分
void writeSimulation (const vector<size_t>&, size_t)	Given a 64-bit input pattern (i.e. vector<size_t>) and number of pattern ought to be write out (i.e. size_t), wirteSimulation ( ) will write input patterns and output patterns in the required format.
Relevant Member Functions of CirGate	

void Simulation( )	Updating its simulation value by bit-wise-AND operation on its _fanIn.	
	Relevant Data Members of CirMgr	
vector<vector<unsigned>>	_fecGrps	Each vector has a distinct simulation value, which size is more than 2.
vector<vector<size_t>>	_CEXipPattern	Collection of counter example patterns of primary inputs ,generated by SAT engine when proving SAT.
ofstream*	_simLog	Pointer to the output file
	Relevant Data Members of CirGate	
size_t	_value	Simulation value, contains at most 64 patterns
vector<unsigned>*	_fecPos	Pointer to _fecGrp that contains this gate

VII. Functionally Reduced AND-Inverter Graph

fraig( )	
Functionality	Make the use of SAT solver to prove two potentially functionally-equivalnet gates.
問題解析	若直觀地將同個 _fecGrps 的所有 potential gates 使用 _solver 證明會產生以下問題：**1. 證明次數過多曠日費時 2. 合併的 gate 會可能形成迴圈** 要解決此問題的方法如下 1. 善用證明後產生的 counter examples 降低證明次數 2. 利用 DFS 的順序從最靠近 PI 的 gate 去合併其他可能等價的 gate
Algorithm	依照 DFS 的順序針對 gate 做 fraig( )
	若此 gate. 是 _fecGrp 的第一個 gate，則為 merger 若非，則為 potential merged gate，和 merger 一同傳入 CirMgr :: proveFEC( ) 做證明
	證明後若為 UNSAT.，則刪除 potential merged gate， 若為 SAT，則收集所有 gate 的 Counter Example 並更新 Simulation value
Relevant Member Functions of CirMgr	
void fraig( )	**1. Sort each _fecGrp with respect to its position on the _DFSList** 2. Calling CirMigr :: fraig(Cirgate*, int&) recursively, and pass its fanIn 3. Collect the remain counter-example generated by SAT solver 4. Clear all lists since circuit structure is altered 5. Calling CirMgr :: strash( )
void fraig(CirGate*, int&)	
1. Every 64 CEXs were collected	1. Collect 64-bit patterns of Primary Inputs with _CEXipPattern (vector<vector<size_t> >), used for the next random simulation ( ). 2. Renew each FEC group, delete the gates ,which simulation value isn't equal or complement to that of the first gate
2. Recursively calling fraig(CirGate*, int&)	Always use gate, which is closest to primary inputs to merge other FEC in the _fecGrp
3. Determine whether the gate has to be merged	‣ **Case 1 : The gate is the first entry of the _fecGrp. It will be the only merger to merge other gates in the group.** ‣ **Case 2 : If potential merged gate's simulation value isn't equal or compliment to that of the merger, it won't call CirMgr :: proveFEC( ).** Note : CirMgr :: proveFEC( ) can make the number of proof greatly reduced!!
void initSolver( )	1. Give all gate on the _GateList a distinct SAT value 2. Calling CirMgr :: buildFanInCNF( ) and pass PO id to it (DFS order)
void buildFanInCNF (const unsigned&)	1. Recursively calling buildFanInCNF( ) and passing _fanIn ID to it 2. _solver calling SatSolver :: addAig(Var, Var, bool, Var, bool) to build connection

void proveFEC (const unsigned& g1, const unsigned& g2, int&, bool ivt)	const unsigned& g1, g2 : ID of gates, bool ivt : potential relation of two gates 1. Calling _solver to do proving progress 2. The result will be SAT or UNSAT ▸ UNSAT calling CirGate :: mergeGates(), and merge g2 with g1 ▸ **SAT** **a. Collect all AIGs and PIs patterns (counter-examples)** **b. Left-shift one bit simulation values of all gates and bit-wise-OR with the collected pattern (note : simulation value change !! The same FEC group won't guarantee the same simulation value anymore !!)**	
void mergeGates (CirGate* , CirGate*, bool)	As mentioned above.	
	Relevant Data Members of CirMgr	
SatSolver	_solver	SAT engine object
vector<vector<unsigned>>	_fecGrps	Each vector has a distinct simulation value, which size is more than 2.
vector<vector<size_t>>	_CEXipPattern	Collection of counter example patterns of primary inputs ,generated by SAT engine when proving SAT.
	Relevant Data Members of CirGate	
vector<unsigned>*	_fecPos	Pointer to _fecGrp that contains this gate