|  |
| --- |
| DSnP Fall 2019  Final Project  Functional Reduced And-Inverter Gate  (FRAIG) Report  Name: Edward Lee  Student ID: B05901119  Contact: [b05901119@ntu.edu.tw](mailto:b05901119@ntu.edu.tw) |

1. Outline

* Data Structure
* Algorithm
* Encounter Problem Description

1. Data Structure

CirGate

* Used as Inheritance
* CirPIGate, CirPOGate, CirCONSTGate, CirUNDEFGate, CirAIGate
* Property for DFTraversal, such as \_fanin, \_fanout, \_mark, static \_globalMark
* For Inverting problem, encode a bit in the pointer of CirGate\*

CirMgr

* Maintain gate defined priority in .aag with 5 vectors
* Maintain CirGate\* with 1 vector
* Maintain FECGroup by a class member
* DFSList is generated when used, I haven’t made it as a var.
* Provide CirSweep, CirOptimize, CirStrash and CirSimulate now.
* CirFraig still not finished, encounter the problem of bad performance and self-looping after merged.

1. Algorithm
2. Strash

I suggest that the key-point of accelerating strash is find out a good hash()that is satisfy:

For any a, b, c in positive integer,

hash(a, b) == hash(b, a)

hash(a, a) != hash(b, b)

hash(a, b) != hash(a, c)

Suggest by a classmate Gaodi Lee, I found a function is good for that, in website MathExchange, that is

In implementation, I have used Unordered\_map in STL. After the hash function are used, seldom meet the case that two gates are not in state Strash but any collision occurs.

1. Simulate

Used fixed 64 bits length to calculate because of convent for implementation. Hash is also used in this function. But it only need to maintain the bit patterns, therefore we don’t need to find a new Hash function.

For Random Simulation, Using rnGen provided by Prof in src/ to generate bit patterns. I haven’t add any extra function in this algorithm.

Pseudo Simulate(vector<size\_t>& pattern )

---------------------------------------------------

Initialize FECGroups with PIGate, CONSTGate, AIGate

Initialize HashTable, ptnContainer;

Generate DFSList;

while (PackPattern(ptnContainer) && KeepGoing())

{

Feed Pattern In Input;

FeedForward the pattern given DFSList;

GetPattern In Output;

For Group in gate

{

For gate in Group

{

if (first\_loop && gate.pattern > 2^63)

{ gate.pattern = NOT(gate.pattern) }

If (loop\_again && setAsFlip(gate.pattern))

{ gate.pattern = NOT(gate.pattern) }

if (!Hash.find(gate.pattern))

Open New Group with &gate

else

Insert &gate to exists Group;

}

// Split FECGroups

For pattern in Hash

{

If the Group size > 1

Keep in newFECGroups

}

}

FECGroups = newFECGroups // Replace

}

1. FRAIG

For me, FRAIG is still not completed, but I have first implemented some functions in here. To prove that whether some gates are equivalent, using the SATProblem solver hce.

The performance and correctness of FRAIG is not good, so I only provided some information about what I understand and implemented in FRAIG first.

Pseudo FRAIG() : take Simulate First

---------------------------------------------------

For Group in FECGroups

{

Initialize unSATGroup, SATGroup // SubGroup

For a, b in Group // Fix a first, sliding b

{

Build SAT(a,b) Problem;

Try to find solution;

If (find) -> SATGroup;

If (!find)-> unSATGroup;

// After searched all b in Group;

If (unSATGroup.size() > 1)

{

Merge all gates in unSATGroup;

Group <- SATGroup;

}

}

}