# Readme

電機碩一　r06921048　李友岐

**(1) Submitted Files:**

file:

abc.h: please use it to replace the original src/base/abc/abc.h

folder:

lsv : please use it to replace the original src/lsv

**(2) How to run:**

To compile the code, please type “make” .

To read the circuit, please type “read <circuit name>” . (ex: “read c432.blif” )

To run the command, please type “1subfind” .

**(3) The method:**

For Pi and Node, I use different ways to achieve the goal.

If the replaced target is Pi, then we build the miter of the original and revised circuit, which are basically the same. Next, we bind all the Pis of two circuits except for the target Pi. If a Pi can be replaced by others, then this Pi must be redundant, which means the value of this Pi doesn’t affect the output result. Thus, after we build the miter and bind all Pis except for target Pi, we check whether this miter is UNSAT. If it is UNSAT, then no matter we give target Pi value 1 or value 0, the output won’t be affected. As a result, it can be replace by any other Pi or Node except for the fanout cone of this target Pi.

If the replaced target is Node, then we build the miter of the original and revised circuit, which are basically the same. However, for the revised circuit, we need to view the target node as a new Pi, which means it doesn’t have fanin constraint. Next, we bind all the Pis of two circuits and check whether it is UNSAT. If it is UNSAT, then this node is redundant, since no matter we give this target node value 1 or value 0, the output won’t be affected. As a result, it can be replace by any other Pi or Node except for the fanout cone of this target Node. If the miter is SAT, then we need to check which Pi or Node can replace the target Node. To replace it, we add a constraint that candidate Node is equal to target node, which is formulate as (C xnor T). If the result become UNSAT, then this candidate node is what we want. To check the complement case, we can change the constraint to (C xor T). Iterate through all candidate nodes, then we are done with this target node.

The main problem is that there are too many iterations. However, there are many of them unnecessary.

**(4) How to improve computation efficiency:**

1. Select affected Po:

In our miter, we don’t need to check every Po. If a Po is not in target’s fanout cone, then no matter how we change the values of target, it doesn’t affect Po. Thus, we first find the affected Pos of each target by recursively getting fanout nodes. Our miter only need to check the affected Po, other Po can be neglect. By this method, the number of CNF clauses largely reduce, which saves a lot of time for us.

2. Select the true candidate Pi and Node:

Next, we reduce the candidates of each target. If a target isn’t redundant, then the candidate node’s fanin cone must contain some of the Pi target node contains. Thus, we find the Pis of target node by recursively getting fanin nodes. After that, we find the fanout cone of these Pis. This fanout cone is our candidates except for the nodes that target node’s fanout cone contains. By this method, we don’t need to iterate through every Pis or Nodes. The number of total iterations largely reduce.

3. Incremental Sat solving:
The re-computation of same clauses wastes lots of time, so we use the method proposed by Minisat. We add an activate variable in the clauses. If we want this constraint , then we activate the clause. Otherwise, we don't activate the clause. This method saves lots of computation time.

**(5) The algorithm:**

```
For each Pi{
        select affected Po
        build the miter
        bind all Pis except for itself
        if (UNSAT){ all candidates can replace it except the fanout cone of it}

}
For each Node{
        select affected Po
        select true candiates
        build the miter (view target node in revised circuit as no fanin)
        bind all Pis
        if (UNSAT){ all candidates can replace it except the fanout cone of it}
        else {
              For each true candidates{
                    activate the constraint that candiate xnor target
                    if (UNSAT) { the candidate can replace target }
                    activate the constraint that candiate xor target
                    if (UNSAT) { the complement of candidate can replace target }
              }
        }
}
```