# SAT-RAR

B07901020劉昀昇

## Outline

- Introduction
- Backgorund
  - Fault model & Fault testability
  - Mandatory assignment
  - Necessary condition for redundant wire
  - General Idea
- Algorithm
- Implementation Issues

## Backgound

Fault Model

- s-a-0
- s-a-1

Fault Testability

- A fault is untestable if the fault can not be sensitized or be propagated to outputs

## Background

Mandatory asssignment (MA)

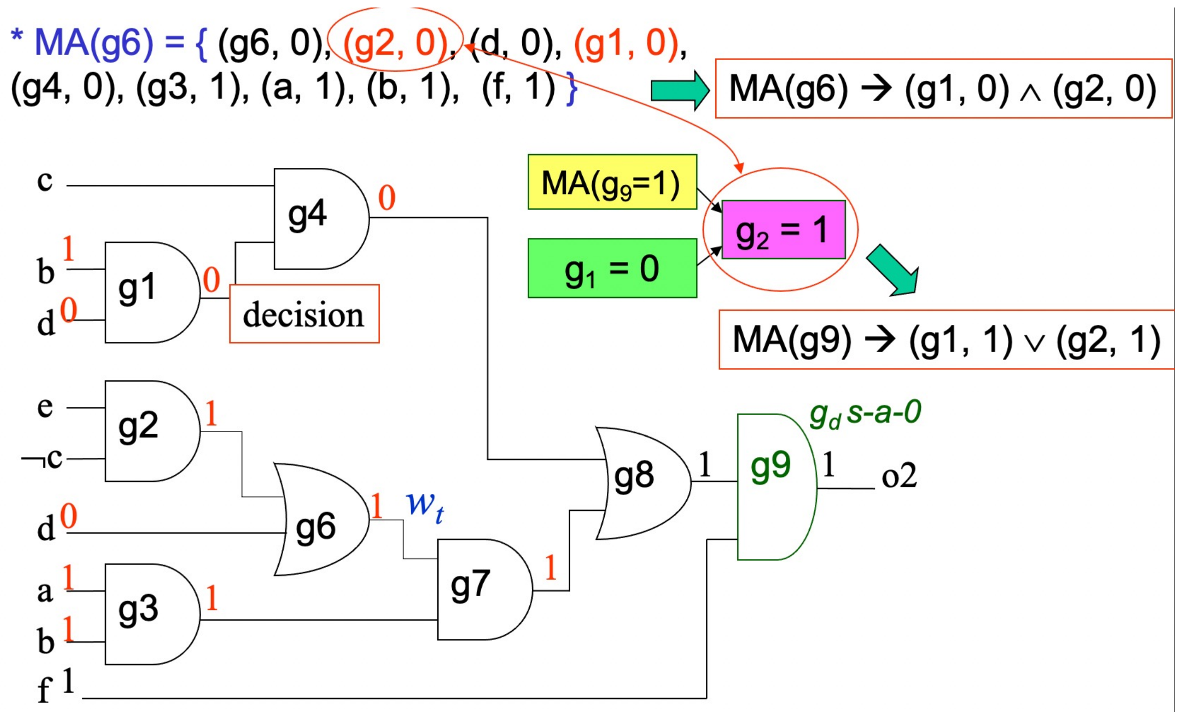
- Logic implication on (1. fualt sensitization, 2. fault propagation)
- MA is the union of (1.) and (2.)

They are only **necessary** conditions

## Background

Example

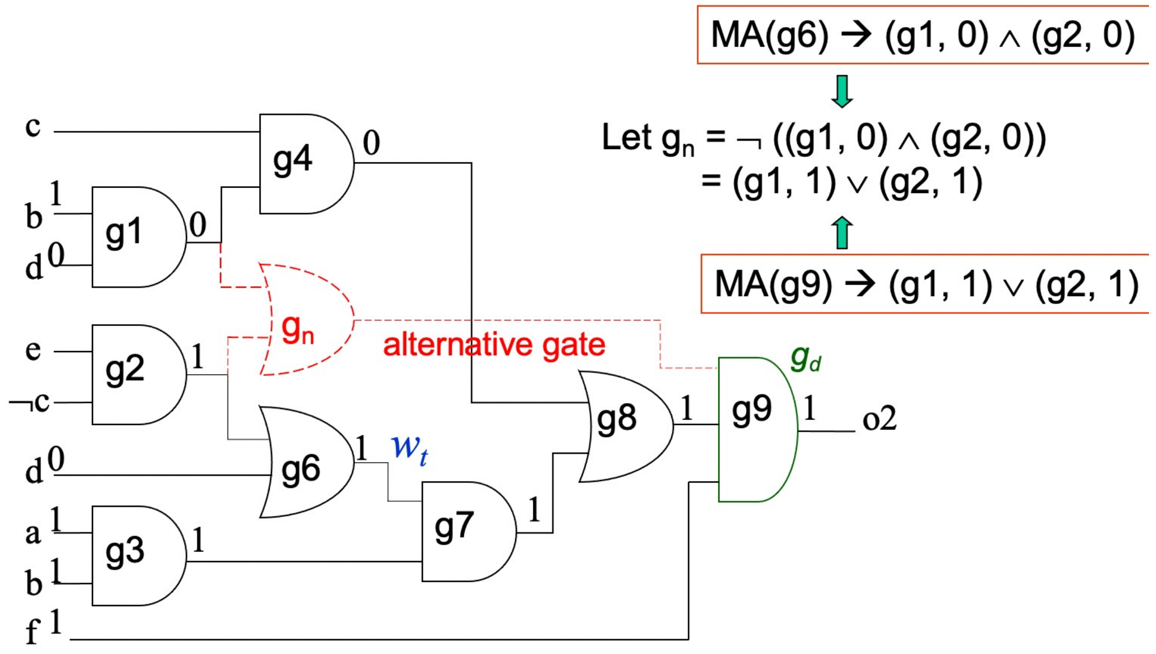
1. Given tow mandatory assignment MA(g6=0) and MA(g9=1)
2. Make decision on g1=0
3. Compute *MA(g6=0) considering the decision
4. Decision g1=0 after MA(g9=1) implicates g2=1, which contradicts g2=0 in *MA(g6=0)
5. Let $g_n$ be $\neg((g_1, 0) \wedge (g_2, 0))$, then $(g_n \rightarrow g_d)$ will make $w_t$ redundant


* MA(g6) = { (g6, 0), (g2, 0), (d, 0), (g1, 0),
(g4, 0), (g3, 1), (a, 1), (b, 1), (f, 1) }
MA(g6) → (g1, 0) ∧ (g2, 0)
MA($g_9$=1)
$g_2$ = 1
$g_1$ = 0
MA(g9) → (g1, 1) ∨ (g2, 1)
decision
$g_d$ s-a-0
$w_t$
g1
g2
g3
g4
g6
g7
g8
g9
c
b
d
e
¬c
a
f
o2

# Background

Reason

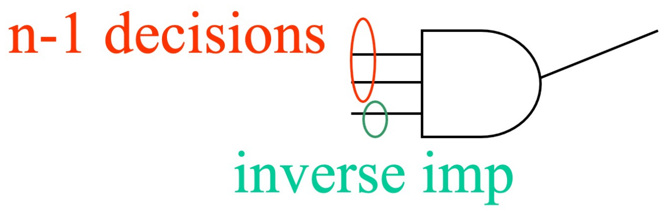
- g6 s-a-1 fault is untestable because there is conflict in MA(g6) after adding $(g_n \rightarrow g_d)$. Fault on $w_t$ can not propagate througth its dominator $g_d$ because it's blocked by the edge added
- Edge $(g_n \rightarrow g_d)$ is also redundant because $g_n$ can not be the only controlling variable of $g_d$

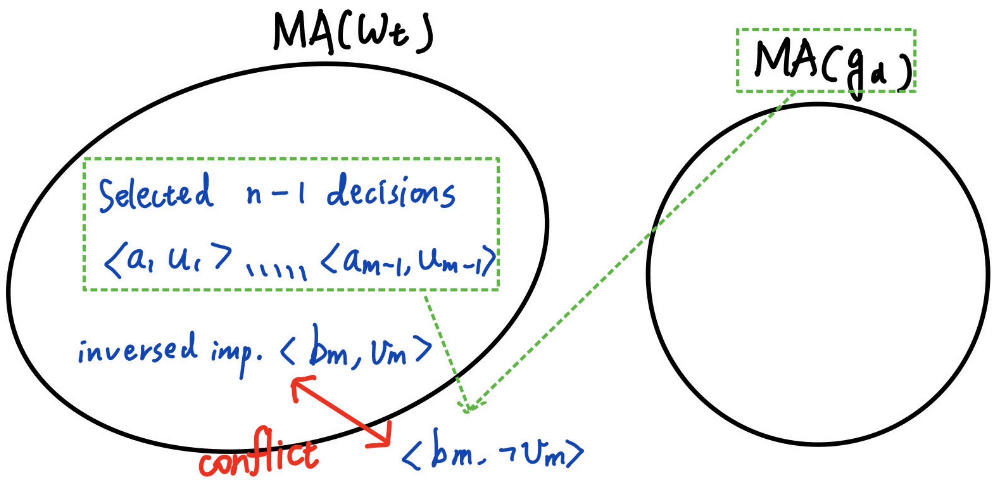


## Background

Multiple decision

- Given decisions $\{(g_1, v_1), (g_2, v_2), \ldots (g_m, v_m)\}$ belongs $MA(w_t)$, previous $m-1$ terms are decision on top of $MA(g_d)$, and $(g_m, v_m)$ belongs $MA(w_t)$. $MA(g_d)$ has no conflict with the decision but produce an implication $(g_m, \neg v_m)$
- Create the multiple-input AND gate $g_n$ with input $\{(g_1, v_1), (g_2, v_2), \ldots (g_{m-1}, v_{m-1}), (g_m, v_m)\}$
- $(g_n, \circ\, g_d)$ or $(g_n, g_d)$ is a valid edge, negating or not depending on gate type of $g_d$
- We can also treat $\{(g_1, v_1), (g_2, v_2), \ldots (g_m, v_m)\}$ as $m$ decisions, and there is conflict

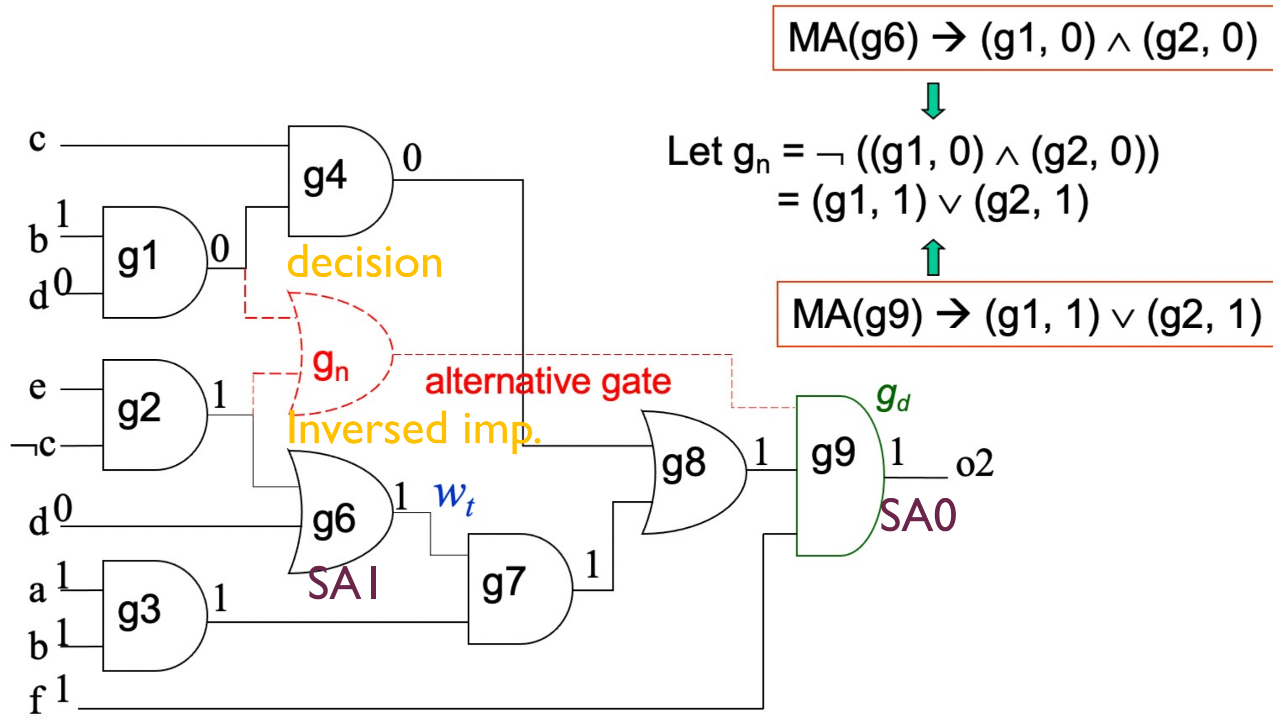




# Background

Explanation

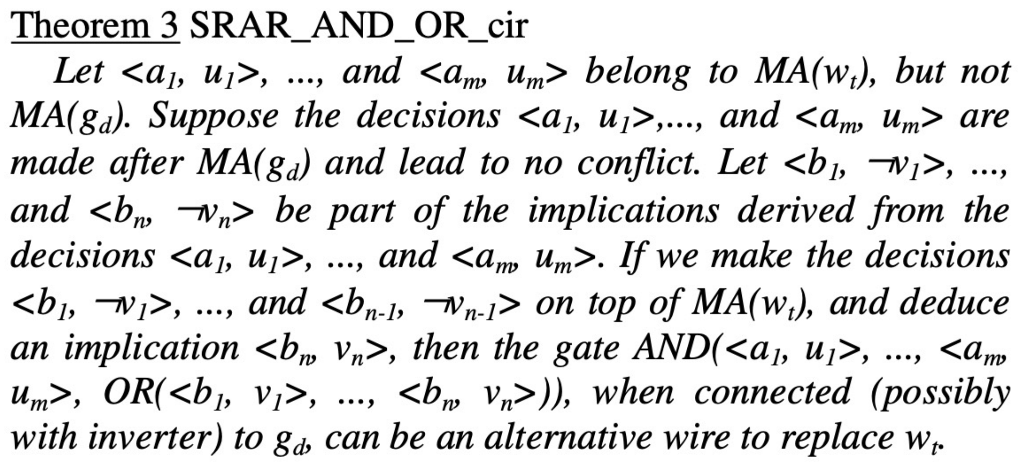
- Fault on $w_t$ can be activated, but can not propagate through $w_d$. The reason is that we can make the orignal input of $w_d$ non-controlling under assignment $MA(w_t)$
- Fault on $w_d$ can not be activated since it implies $(g_m, \neg v_m)$, however now $(g_m, v_m)$ is controlling assignment on added gate $g_n$, which leads to the conflict



## Background

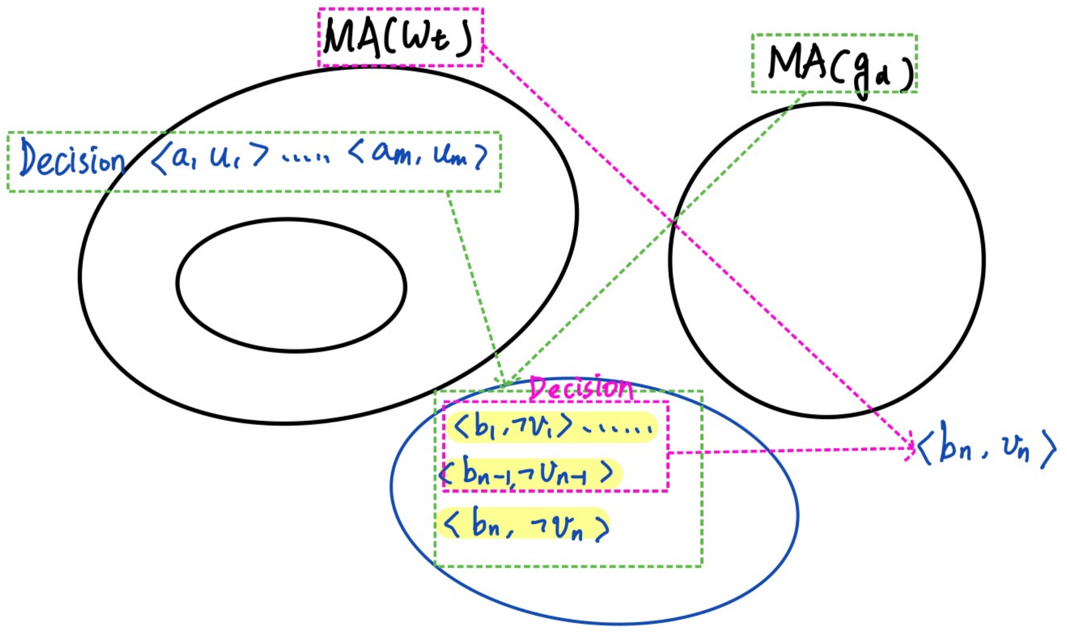
- Cosntruct an arbitrary circuit to replace target wire
- This may be expensive

General Case:



Theorem 3 SRAR_AND_OR_cir

*Let $<a_1, u_1>$, ..., and $<a_m, u_m>$ belong to $MA(w_t)$, but not $MA(g_d)$. Suppose the decisions $<a_1, u_1>$,..., and $<a_m, u_m>$ are made after $MA(g_d)$ and lead to no conflict. Let $<b_1, \neg v_1>$, ..., and $<b_n, \neg v_n>$ be part of the implications derived from the decisions $<a_1, u_1>$, ..., and $<a_m, u_m>$. If we make the decisions $<b_1, \neg v_1>$, ..., and $<b_{n-1}, \neg v_{n-1}>$ on top of $MA(w_t)$, and deduce an implication $<b_n, v_n>$, then the gate AND($<a_1, u_1>$, ..., $<a_m, u_m>$, OR($<b_1, v_1>$, ..., $<b_n, v_n>$)), when connected (possibly with inverter) to $g_d$, can be an alternative wire to replace $w_t$.*



$$g_n = (\langle a_1\ u_1\rangle \wedge \cdots\cdots \wedge \langle a_m . u_m\rangle) \wedge (\langle b_1\ \neg v_1\rangle \wedge \cdots\cdots \wedge \langle b_{n-1}, \neg v_{n-1}\rangle)$$

# Algorithm

Given $w_t$

Redundancy check and copy implications to set $\Phi_{wt}$

From $w_t$ to output (i=k down to 0), for each $g_{di}$, only backtrack to level $i$ to find MAs. If no conflict, generate $MA(g_{di})$

2-Way RAR

Select a gate from $\Phi_{wt}$ - $\Phi_i$ as the SAT decision

**SAT-controlled RAR**, based on selected set of decision,
If conflict, compute alternative wire
Conflict driven learning can prune the decision tree

**Algorithm: SAT-Controlled RAR**

```
SatRAR($w_t$) {  // $w_t$: target wire
  $A$ ← ∅;  // valid alternative wires
  decisionLevel ← 0;
  if (RedundancyCheck($w_t$) has conflict)
    return REDUNDANT($w_t$);
  Copy current implications as $\Phi_{wt}$;  // = MA($w_t$)
  for i = k Down To 0  // For each dominator of $w_t$
    Backtrack decisionLevel to i;
    if (BCP(CNTR($g_{di}$)) has conflict)
      return REDUNDANT($g_{di}$);
    Copy current implications as $\Phi_i$;// MA($g_{di}$)
    for_each ({$g_s | g_s \in$ inverseImp($\Phi_{wt}$, $\Phi_i$)})
      A += MakeAltWire($g_s$, $g_{di}$);
    for_each({$g_s | g_s \in \Phi_{wt} - \Phi_i \wedge g_s \notin$ fanoutCone($g_{di}$)})
      decisionLevel += 1;
      if (Decision($g_s$) has an inverse imp to MA($w_t$))
        Let $G$ = set of decisions that account for
            the inverse imp;
        $A$ += MakeAltGate(AndGate($G$), $g_{di}$);
      else if (Decision($g_s$) has a conflict)
        Let $G$ = set of decisions that lead to the
            conflict;
        $A$ += MakeAltGate(AndGate($G$), $g_{di}$);
        ConflictLearning();
  return $A$;
}
```

## Incorporate SAT Solver into the Algorithm

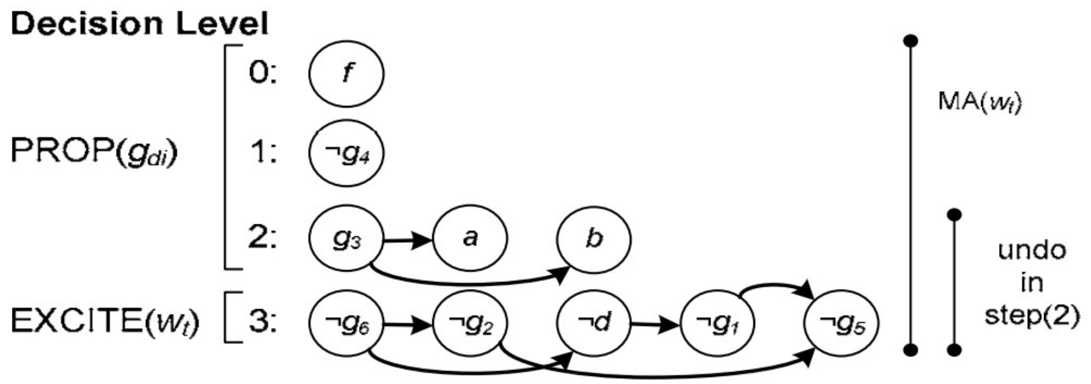
Variable

- Set each gate as a unique variable

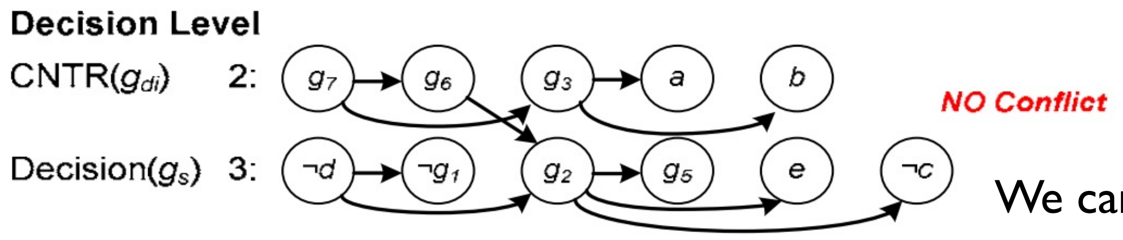
Constraint

- $MA(w_t)$ and $MA(g_d)$, for each decision level, combine the decision and $MA(w_t)$ (More mandatory assignments will be generated)
- If UNSAT, then conflicts exists in decisions

## Algorithm



(a) Implication graph for MA($w_t$) in Figure 1



(b) Implication graph for MA($g_7$) and decision($\neg d$)

We can remove $g_1$ from the decision list

## Implementation Issues

- Parsing

  Represent circuit in aiger format (convert a OR gate into a NAND gate with two inverted input)

- Preprocess

  Find dominators of all nodes in $O(V \cdot \lg(V))$

  Find transitive closure of the circuit in $O(V^2)$

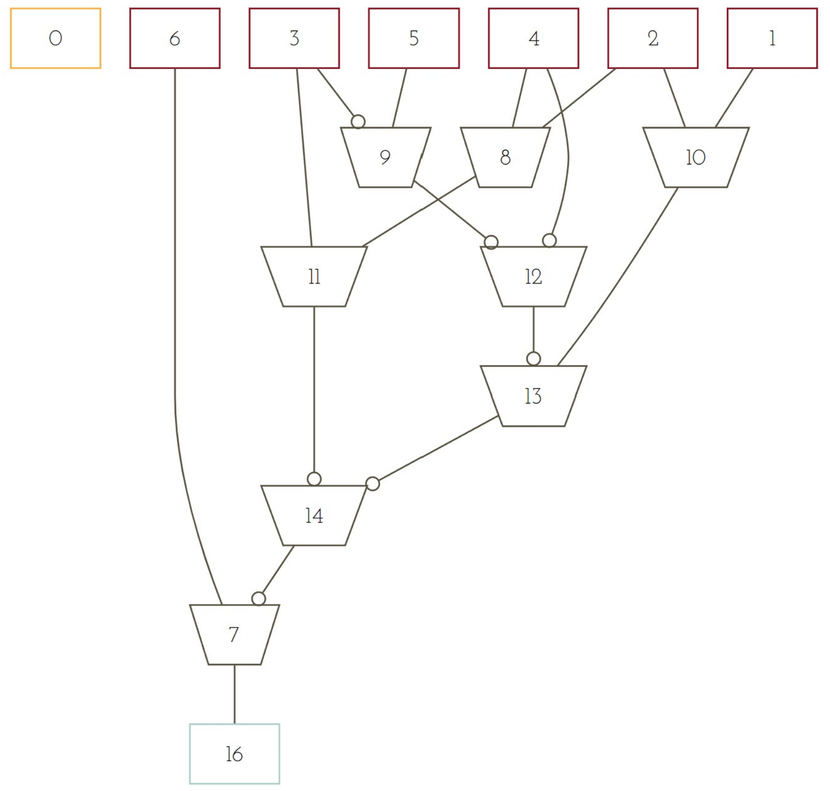
- SAT-RAR algorithm

  Modify MINISAT, use function "propagate" to compute mandatory assignment

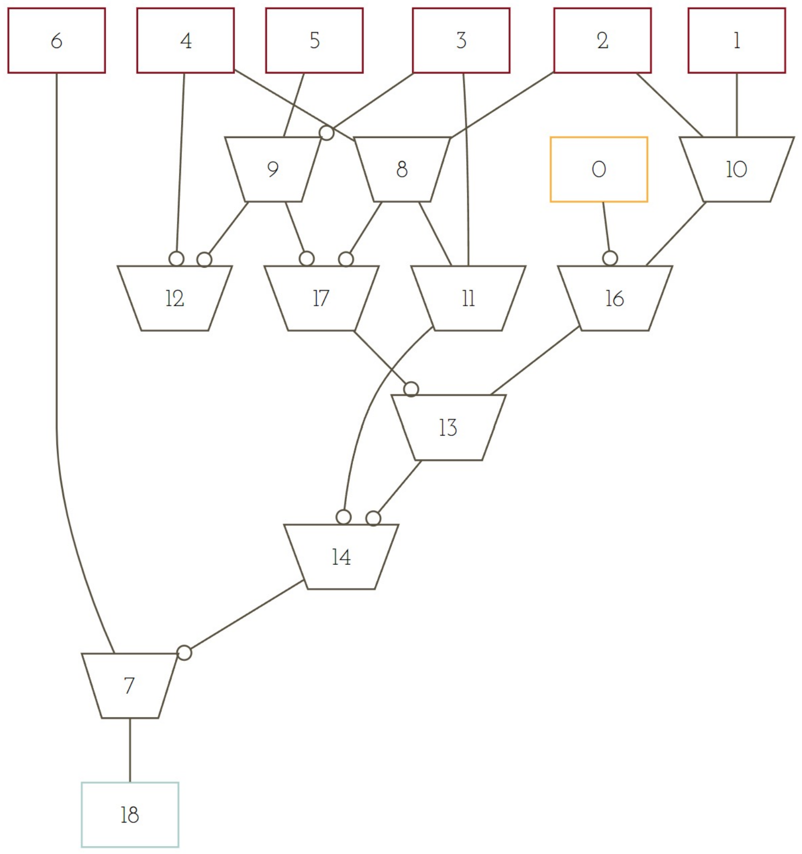
  Use "CancelUntil()" to back trace to desired level

- Construct a repaired circuit for verification

## Examples



w_t(12, 13)
level 1: 6
level 2: !11
level 3: 1 2 10
level 4: !4 !8 !9
  g_d = 13
  level 1: 6
  level 2: !11
  level 3: 1 2 10 !12
    decide !8
      conflict gid 9
  g_d = 14
  level 1: 6
  level 2: !11 !13
    decide !8
    decide 10
  …..



Ref: AAG-Visualizer (byronhsu.github.io)

## Results

| | #wire | #tar | #alt | #T | 2-Way-rar | Rar-wire | Rar-gate |
|---|---|---|---|---|---|---|---|
| **C17** | 8 | 3 | 3 | 0 | 2 | 0 | 1 |
| **C432_r** | 523 | 374 | 1039 | 0.25 | 1030 | 0 | 109 |
| **C432** | 531 | 355 | 1050 | 0.31 | 1024 | 0 | 26 |
| **C499_r** | 684 | 436 | 485 | 0.19 | 317 | 0 | 68 |
| **C499** | 1004 | 732 | 942 | 0.5 | 856 | 0 | 86 |
| **C880** | 699 | 261 | 355 | 0.23 | 181 | 0 | 174 |
| **C1355** | 1036 | 620 | 830 | 0.45 | 648 | 0 | 182 |
| **C1908** | 2304 | 1238 | 2249 | 18 | 1399 | 0 | 850 |
| **C3540** | 4038 | 2673 | 7274 | 31.69 | 2784 | 0 | 4490 |
| **C5315** | 5884 | 3926 | 6484 | 9.47 | 4688 | 0 | 1796 |
| **C6288** | 4320 | 2417 | 2417 | 33.07 | 1923 | 0 | 494 |
| **C7552** | 8746 | 5641 | 9284 | 21 | 6633 | 0 | 2651 |

## Issues

- A successful repairment
  1. Fault on replaced wire can be activated, but can not propagate
  2. Fault on replacing wire can not be activated

(1) Can be handled by replaced $w_t$ with either constant one or zero depending whether there is a inverter on this wire

(2) Miter still return SAT even if we do not replace $w_t$ with constant, that means the added wire is not redundant. To further explain, there exists cases that $g_d$ ‘s both two original inputs are 1 (MA), and AND gate $g_n$ is 1. $g_d$ is controlled by $g_n$.