SAT-Controlled Redundancy Addition and Removal — A Novel Circuit Restructuring Technique

Chi-An Wu, Ting-Hao Lin, Shao-Lun Huang, Chung-Yang (Ric) Huang

Graduate Institute of Electronics Engineering, National Taiwan University, Taiwan

Abstract - We proposed a novel Boolean Satisfiability (SAT)-controlled redundancy addition and removal (RAR) algorithm to resolve the performance and quality problems of the previous RAR approaches. With the introduction of modern SAT techniques, such as efficient Boolean constraint propagation (BCP), conflict-driven learning, and flexible decision procedure, our RAR engine can identify 10x more alternative wires/gates while achieving 70% reduction in runtime.

I. Introduction

Redundancy addition and removal (RAR) is a popular and powerful circuit restructuring technique. It has been successfully applied to various VLSI design problems such as logic synthesis [1], post-layout performance optimization [2], design debugging [3], ECO [4], FPGA routing [5], and power analysis[6], etc. Given a non-redundant target wire(w_t) in a circuit, RAR technique deliberately inserts a redundant wire to the netlist and makes the originally non-redundant wire redundant. Taking the circuit in Figure 1 as an example, the target wire $w_t(g_6 \rightarrow g_7)$ is not redundant. After the redundant

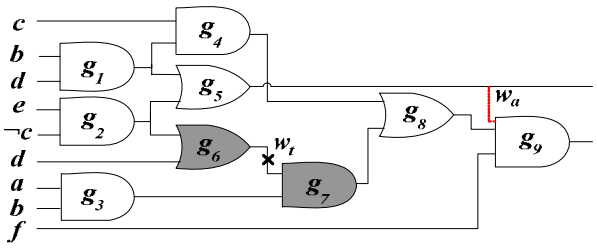


Fig.1: An example of redundancy addition and removal.

wire $w_a(g_5 \rightarrow g_9)$ is added, the wire w_t as well as its surrounding gates g_6 and g_7 will become redundant and thus removable. Therefore, the circuit can be simplified.

As we will describe in Section II, previous RAR algorithms have drawbacks in either runtime efficiency or alternative wire quality. The first RAR algorithm, RAMBO [1], as well as its followed works (e.g. [7]), usually suffer in long runtime because of the need to perform a large number of redundancy checks. Entrena *et. al.* in [8] proposed a "2-way" RAR algorithm to alleviate the runtime problem of the original RAMBO. However, due to the limitation of logic implications, it usually finds fewer alternative wires. On the other hand, if we apply more sophisticated implication methods like recursive learning to improve the implicability in the 2-way RAR, it may identify more alternative wires but the runtime will increase substantially. In [9], Chang *et. al.* implemented a single-pass RAR technique by the non-controllability/observability logic of FIRE [10]. However, their algorithm also requires the expensive implication methods to achieve satisfactory results

In this paper, we proposed a Boolean Satisfiability (SAT) controlled RAR technique to address the above issues. Through the seamless integration of powerful SAT solving and RAR techniques, our RAR algorithm can obtain a high quality and quantity of alternative wires/gates, and at the same time achieve great improvements on the runtime efficiency. It can also be extended with different types of optimization filters so that users can *control* the optimization strategy to suit their needs. The experimental results show that our algorithm can identify 10x more alternative wires/gates while achieving more than 70% reduction in runtime.

The rest of the paper is organized as follows: in Section II, we first review the algorithms of the previous RAR approaches. Our proposed SAT-controlled RAR algorithms and their detailed implementations are described in Sections III and IV, respectively. We present our experimental results in Section V and conclude the paper in Section VI.

II. Preliminaries

To facilitate the reader's understanding of our new RAR algorithm, we first review the background knowledge of the RAR techniques. Two previous RAR algorithms [1][8] are discussed and compared. By observing the causes of their performance deficiency, we come up with the idea of utilizing SAT engine for a superior RAR algorithm.

A. Redundancy Test and Mandatory Assignment

A wire in a combinational circuit is redundant if and only if its corresponding stuck-at fault is untestable [11]. While completely verifying the testability (or untestability) of a fault, called *fault test* (or *redundancy test*), is a NP-complete problem [12], in practice, people usually compute the set of *mandatory assignments (MAs)* instead. As its name suggests, the MAs consists of the necessary signal values for a fault to be testable.

[Example 1] Consider the stuck-at-1 fault test on the wire $w_t(g_6 \rightarrow g_7)$ in Figure 1. For the fault to be testable, we must assign a '0' on the gate g_6 so that its values in good and faulty circuits can be differentiated. With logic implications, we have:

- Fault excitation(w_t): { $\neg g_6, \neg g_2, \neg d, \neg g_1, \neg g_4, \neg g_5$ } (1)

On the other hand, for the fault to be observed at any of the circuit outputs, the difference on w_t must be propagated through its dominators g_7, g_8, and g_9. This requires the side inputs of the dominators, g_3, g_4, and f, to have non-controlling values '1', '0', and '1', respectively. With logic implications, we have:

- Fault propagation(w_t): { $g_3, \neg g_4, f, a, b$ } (2)

MA(w_t) is the union of (1) and (2).

- MA(w_t): { $\neg g_6, \neg g_2, \neg d, \neg g_1, \neg g_4, \neg g_5, g_3, f, a, b$ } (3)

□

B. Original RAMBO Algorithm

By definition, MAs contain the necessary signal assignments for a fault to be testable. Therefore, if there is a conflict in MAs, that is, a contradictory implication on certain gate, the fault will be untestable and thus the corresponding wire under test must

978-1-4244-2749-9/09/$25.00 ©2009 IEEE

Authorized licensed use limited to: National Taiwan University. Downloaded on December 03,2021 at 03:33:23 UTC from IEEE Xplore. Restrictions apply.

be redundant.

On the other hand, for a non-redundant wire, there should be no conflict in its MAs. However, if we can somehow change the circuit netlist (e.g. add a wire) to *perturb* the logic implications of its redundancy test, we can create a conflict in MAs and thus make it redundant. Entrena *et. al.* in [1] observed that this can be easily achieved by adding a connection from a mandatory assignment gate of the target wire, to the fanin of one of its dominators.

[Example 2] For the target wire $w_t(g_6 \rightarrow g_7)$ and the MAs of its stuck-at-fault test in Example 1, the gate g_5 has the input controlling value '0' for the dominator g_9, and g_1 has the non-controlling value '0' for g_8. Therefore, the connections $g_5 \rightarrow g_9$ and $g_1 \rightarrow\circ\ g_8$ can both block the fault propagation path and thus make w_t redundant (note: "$\rightarrow\circ$" means a wire connection with an inverter). □

Please note that one of the above additional wires, $g_1 \rightarrow\circ\ g_8$, is not redundant. Adding it to the circuit will change the circuit functionality. To avoid this problem, the original RAMBO algorithm goes through all the potential alternative wires and performs corresponding fault tests to screen out the invalid ones. For larger designs, the number of potential alternative wires can be huge and thus the RAR performance will be severely deteriorated.

Chang *et. al.* in [7] proposed several rules and theorems to prune out the impossible alternative wire candidates in the RAMBO algorithm. They showed that the number of fault tests can be successfully reduced. However, according to the survey in [13], the improvement is still limited (22% in runtime).

C. 2-Way RAR

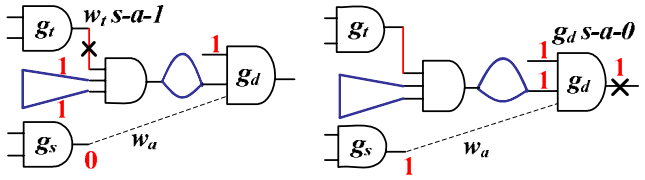
The drawback of the original RAMBO is its vast amount of redundancy tests on the potential alternative wires. Entrena *et. al.* in [8] proposed a "2-way" RAR to resolve this problem. Basically they started from a reverse thinking: "Can we deliberately add a redundancy to the circuit first?" Given a destination gate g_d, they observed that a redundancy can be easily created by connecting any gate g_s, which is in the MAs of g_d's output stuck-at-fault test, to the g_d itself.

[Example 3] Let's consider the gate g_6 in Figure 1 as the destination gate. The MAs of g_6 stuck-at-1 are: { $\neg g_6$, $\neg g_2$, $\neg d$, $\neg g_1$, $\neg g_4$, $\neg g_5$, g_3, f, a, b }. Excluding the direct fanins and the gates in the fanout cone of g_6, we have { $\neg g_1$, $\neg g_4$, $\neg g_5$, g_3, f, a, b }. Then a wire (possibly with inverter) from any of these gates to g_6, for example, $g_4 \rightarrow g_6$ or $g_3 \rightarrow\circ\ g_6$, is a redundant wire. □

To verify the above example, let's examine the newly added wire $w_a(g_4 \rightarrow g_6)$. Its *fault propagation* requirement includes the non-controlling value assignments to the original inputs of g_6 and the side inputs of g_6's dominators. It is *exactly the same* as the MAs of the output stuck-at-1 fault test on g_6. However, this set of MAs contains an assignment on g_4 with value '0'. This will in turn contradict the *fault excitation* requirement of $w_a(g_4 \rightarrow g_6)$ stuck-at-0 fault. Therefore, w_a must be redundant.

Example 3 shows us how to create a redundancy in a circuit, and Example 2 tells us how to make an originally non-redundant wire redundant. Without loss of generosity, we can illustrate the concepts behind these two examples using AND gates as in Figure 2. If we compare both figures carefully, we can find that they only differ on the implied value of g_s. This observation leads to the basic idea of the "2-way" RAR algorithm:

Let w_t be the target wire and g_d be one of its dominators. If a gate g_s has inverse implication values between MA(w_t) and MA(g_d), then the connection $g_s \rightarrow g_d$ or $g_s \rightarrow\circ\ g_d$, depending on the gate type of g_d and value of g_s, will be a valid alternative wire to replace w_t. No individual redundancy check is necessary as in the original RAMBO algorithm.



(a) Add a wire w_a to make w_t redundant (b) Add a wire w_a which is redundant by itself

Figure 2: Basic idea behind the 2-way RAR algorithm

[Example 4] Consider the target wire $w_t(g_6 \rightarrow g_7)$ and one of its dominator $g_d(g_9)$ in Figure 1. The MAs for w_t stuck-at-1 fault are: { $\neg g_6$, $\neg g_2$, $\neg d$, $\neg g_1$, $\neg g_4$, $\neg \mathbf{g_5}$, g_3, f, a, b } and MAs for g_d stuck-at-0 are: { g_9, g_8, f, g_6, b, $\mathbf{g_5}$ } (Note: *Recursive learning* is required to derive MA(g_d); more details below). Intersecting both MAs, we have the gate g_5 with opposite values. According to the 2-Way RAR algorithm, the wire $w_a(g_5 \rightarrow g_9)$ must be a valid alternative wire. No redundancy check is needed. □

Intuitively, the 2-way RAR algorithm can alleviate the performance problem of the original RAMBO because it does not require iterative redundancy checks on the vast number of potential alternative wires. However, in reality, there are rarely conflicting assignments from the intersection of these two MAs, unless some expensive method such as recursive learning [14] is applied.

For the case in Example 4, level-2 recursive learning is needed (i.e. recur from g_8 and then from g_6). There will be no alternative wire identified if we limit the recursion depth to 0 or 1. However, for larger testcases, deeper recursion depth will result in significant runtime degradation.

D. Motivation of Our SAT-Controlled RAR Technique

In summary, the performance problem of the original RAMBO is due to the large number of fault tests on the potential alternative wires. However, if we examine the testability requirements among these faults, we can find that there are substantial overlaps among their MAs, especially originated from the fault propagation conditions. Therefore, one interesting idea for RAMBO improvement will be to avoid the repeats in the logic implication process. We will demonstrate our realization of this idea with an incremental SAT solver.

On the other hand, the bottleneck of the 2-way RAR algorithm is mainly on the use of recursive learning. We will show that by using SAT decision procedure and conflict-driven learning, our new RAR algorithm can identify more alternative wires than the previous approaches, and at the same time achieve significant performance improvement.

III. SAT-controlled RAR Algorithm

We propose a new RAR approach, called SAT-controlled RAR (abbreviated as SatRAR, or SRAR), by incorporating several novel redundancy identification techniques into the SAT

Authorized licensed use limited to: National Taiwan University. Downloaded on December 03,2021 at 03:33:23 UTC from IEEE Xplore. Restrictions apply.

solving procedure. We name it "*controlled*" because our RAR algorithm can provide the flexibility to control the quality and quantity for the circuit restructuring.

We assume that the readers should be familiar with the basic SAT algorithms, including the DPLL-based decision procedure, Boolean constraint propagation (BCP), and conflict-driven learning, etc. We will not go through them due to the space limit.

Our SatRAR techniques can identify not only alternative wires, but also alternative gates and, in general, alternative sub-circuits. We will name them SRAR_Wire, SRAR_Gate, and SRAR_Cir, respectively. We will first present the theorems behind them followed by some examples. The detailed SatRAR implementation will be explained in the next section.

A. *Single alternative wire by SAT-controlled RAR*

Theorem 1 SRAR_Wire

Let MA(w_t) and MA(g_d) be the mandatory assignments for the fault tests of the target wire w_t and its dominator g_d, respectively. Let <g_s, v> belong to MA(w_t) but not MA(g_d), and g_s be not in the fanout cone of g_d. If we make a decision <g_s, v> on top of MA(g_d) and encounter a conflict, then (i) MA(g_d) ⇒ <g_s, ¬v> and (ii) (g_s→ g_d) or (g_s→◦ g_d) must be a valid alternative wire for w_t.

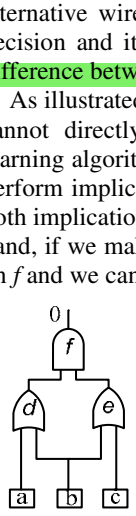
Proof:

Since there is a conflict between MA(g_d) and <g_s, v>, we have MA(g_d)∧(g_s=v) = false, and therefore, MA(g_d) ⇒ <g_s, ¬v>. Because <g_s, v> belongs to MA(w_t), MA(w_t) ⇒ <g_s, v>. According to the theorems behind 2-way RAR [8], (g_s→g_d) or (g_s→◦ g_d) must be a valid alternative wire for w_t. □

[Example 5] Consider the target wire w_t(g_6→g_7) and one of its dominator g_d(g_9) in Figure 1. The MAs for w_t stuck-at-1 and g_d stuck-at-0 faults are: { ¬g_6, ¬g_2, ¬d, ¬g_1, ¬g_4, ¬g_5, g_3, f, a, b } and { g_9, g_8, f }, respectively. If we pick the gate g_5 which belongs to MA(w_t) but not MA(g_d), and make a decision <g_5, 0>, we will encounter a conflict in its implications. Therefore, the wire (g_5 → g_9) must be a valid alternative wire to replace (g_6→g_7). □

Compared to the 2-Way RAR case in Example 4, where the level-2 recursive learning is required to identify the same alternative wire as in Example 5, our approach needs only a decision and its logic implications. This is mainly due to the difference between "recursive" and "conflict-driven learning".

As illustrated in Figure 3, the gate f has implied value '0' and cannot directly imply on any of its fanins. The recursive learning algorithm will first split on two cases, d = 0 and e = 0, perform implications and then conclude that the intersection of both implications, b = 0, is the learned implication. On the other hand, if we make a decision on b = 1, it will result in a conflict on f and we can learn that f = 0 implies b = 0.



- Recursive learning:
  f = 0 ⇒ d = 0 or e = 0
  ⇒ { a=0, b=0 } or { b=0, c=0 }
  ⇒ b = 0
- Conflict-driven learning:
  f = 0; decision b = 1 results in conflict

f = 0 ⇒ b = 0

Figure 3: Recursive vs. conflict-driven learning

It is clear that recursive learning needs to conduct more implications and has the overhead in checking the intersections of the implications. What is worse, there are usually a lot of gates to recur but many of them either lead to no learned implication or produce the implications that are irrelevant to the RAR target. Besides, it is not easy to reuse its learned implications because of the dependencies on the implication context of previous recursive level. Therefore, as we will demonstrate in the experimental results, our proposed SatRAR algorithm is much more efficient than the 2-Way RAR.

However, one may ask: "Which gate should we choose to make decision first"? If we iteratively pick the gates from the set { MA(w_t) – MA(g_d) } for decisions, this may sound very similar to the original RAMBO algorithm where the redundancy test on each potential alternative wire is performed one by one. Then where is the improvement?

The difference is, while in the original RAMBO the redundancy tests are executed separately, in SatRAR, we convert the potential alternative wires as "decision variables" and apply the concept of *incremental SAT* to share the most of the implication efforts. In addition, with the assistance of conflict-driven learning, we can identify multiple alternative wires under the same conflict and record the learned information for the future implication. Moreover, as we will present in the following sections, we can extend our algorithm to identify alternative gates and even alternative sub-circuits.

B. *Single alternative gate by SAT-controlled RAR*

We will now present our algorithm for identifying single alternative gates. This is corresponding to the cases where the decision leads to no conflict.

Theorem 2 SRAR_gate

Let MA(w_t) and MA(g_d) be the mandatory assignments for the fault tests of the target wire w_t and its dominator g_d, respectively. Let both <g_s, u> and <g_t, v> belong to MA(w_t), and be not in the fanout cone of g_d. Suppose we make the decision <g_s, u> after MA(g_d) and result in an implication <g_t, ¬v>. Let a gate g_n = AND(<g_s, u>, <g_t, v>). Then (i) MA(g_d) ⇒ ¬ g_n, and (ii) g_n or ¬ g_n, when connected to g_d, must be a valid alternative gate for w_t.

Proof:

Since <g_s, u> and <g_t, v> belong to MA(w_t), we have: MA(w_t) ⇒ <g_s, u> ∧ <g_t, v>. Let g_n = AND(<g_s, u>, <g_t, v>), and then MA(w_t) ⇒ g_n. (1)

Similarly, we have MA(g_d) ∧ <g_s, u> ⇒ <g_t, ¬v>. Rearrange the terms we will get: MA(g_d) ⇒ <g_s, ¬u> ∨ <g_t, ¬v>. Then by DeMorgan's Law, we know MA(w_t) ⇒ ¬g_n. (2)

From (1), (2), and 2-Way RAR theorem [8], the wire (g_n→g_d) or (g_n→◦ g_d) can be added to replace w_t. □

[Example 6] Consider the target wire w_t(g_6→g_7) and one of its dominator g_d(g_7) in Figure 1. The MAs for w_t stuck-at-1 and g_d stuck-at-0 faults are: { ¬g_6, **¬g_2**, ¬d, ¬g_1, ¬g_4, **¬g_5**, g_3, f, a, b } and { f, ¬g_4, g_7, g_6, g_3, a, b }, respectively. If we pick the gate d and make a decision <d, 0>, we will derive the implications: { ¬g_1, **g_2**, **g_5**, e, ¬c }. Clearly, the gates g_2 and g_5 have inverse implication values from their counterparts in MA(w_t). Therefore, both AND(¬g_2, ¬d) and AND(¬g_5, ¬d) are valid alternative gates (with inverted wire connection to g_7) to replace the target wire. □

Authorized licensed use limited to: National Taiwan University. Downloaded on December 03,2021 at 03:33:23 UTC from IEEE Xplore. Restrictions apply.

Please note that one of the alternative gates "g_n: AND($\neg g_2$, $\neg d$)" is actually equivalent to NOR(g_2, d), and the wire ($g_n \rightarrow\circ\ g_7$) is exactly the same as the target wire. We have excluded this kind of trivial cases in our experimental results. For more complicated circuit, we can identify many non-trivial alternative gates as in Figure 4. Although the names of the gates are not clear due to the resolution of the schematic print out, we can see that the alternative gate is composed by the gates far from the target wire.

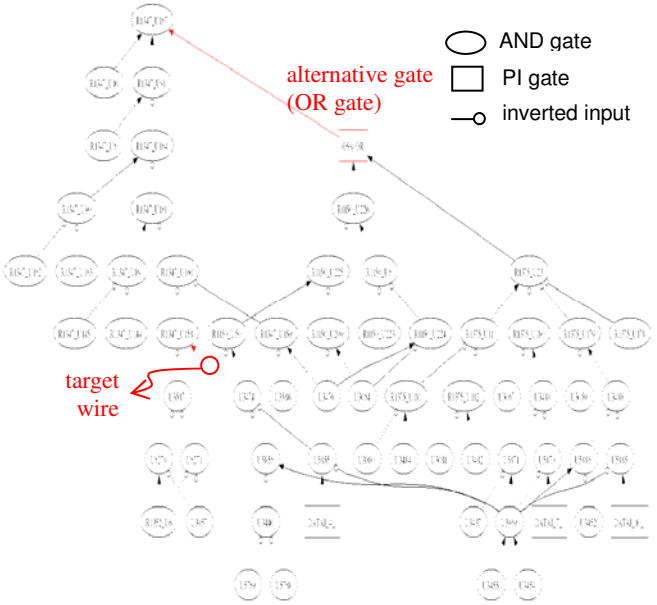


Figure 4: A non-trivial alternative gate example from circuit B14

It is worthwhile to note that the single alternative gate can also be derived when there is an implication conflict under multiple decisions (Corollary 2.1). This may sounds like an extra, negligible case since we have checked the inverse implications in Theorem 2. That is, let a and b be two MAs in MA(w_t). If the decision on a on top of MA(g_d) does not imply any value on b, and if we then continue making decision on b, will there be any conflict? If yes, shouldn't the decision on a implies $\neg b$ in the first place?

The answer is "no", because the implication power is limited. The implications may go only from one direction (e.g. $a \wedge b \Rightarrow$ *conflict*), but not necessarily from the other (e.g. a $\Rightarrow \neg$b). Therefore, we should fortify Theorem 2 with the following corollary.

Corollary 2.1 SRAR_gate with implication conflicts

Let $<g_s, u>$ *and* $<g_t, v>$ *belong to MA*(w_t). *If the decisions* $<g_s, u>$ *and* $<g_t, v>$ *on top of MA*(g_d) *lead to a conflict, and if both are contributors to the conflict, then the gate AND*($<g_s, u>$, $<g_t, v>$) *is an alternative gate to* w_t. □

In addition, Theorem 2 can also be extended to find the multi-input alternative gates.

Corollary 2.2 Multi-input alternative gates

Let $<g_1, v_1>$, $<g_2, v_2>$,... *and* $<g_n, v_n>$ *belong to MA*(w_t). *If the decisions* $<g_1, v_1>$, $<g_2, v_2>$,... *and* $<g_{n-1}, v_{n-1}>$ *on top of MA*(g_d) *lead to no conflict but together produce an implication* $<g_n, \neg v_n>$, *then the multi-input gate AND*($<g_1, v_1>$, $<g_2, v_2>$,... $<g_n, v_n>$) *is an alternative gate to* w_t. □

The proofs for the corollaries should be quite straightforward. Due to the space limit, we omit them here.

C. *Alternative sub-circuit by SAT-controlled RAR*

In the single alternative wire and gate identification algorithms (i.e. Theorems 1 and 2), we pick the gates in the set of { MA(w_t) – MA(g_d) } for SAT decisions. In the following, we will show that by alternatively making decisions from { MA(w_t) – MA(g_d) } and { MA(g_d) – MA(w_t) }, we can construct an arbitrary circuit to replace a target wire. Although this algorithm may seem to be too expensive in most of the practical cases, it provides the theoretical soundness and can be evoked when a must-removed wire cannot be removed by other approaches. We will first introduce the 2-level AND-OR alternative circuit identification theorem.

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.

Proof:

From the assumptions, we have:

- MA(w_t) $\Rightarrow <a_1, u_1> \wedge ... \wedge <a_m, u_m>$ (1)
- MA(w_t) $\wedge <b_1, \neg v_1> \wedge ... \wedge <b_{n-1}, \neg v_{n-1}> \Rightarrow <b_n, v_n>$
  $\equiv$ MA(w_t) $\Rightarrow <b_1, v_1> \vee ... \vee <b_{n-1}, v_{n-1}> \vee <b_n, v_n>$ (2)
- MA(g_d) $\wedge <a_1, u_1> \wedge ... \wedge <a_m, u_m>$
  $\Rightarrow <b_1, \neg v_1> \wedge ... \wedge <b_{n-1}, \neg v_{n-1}>$ (3)

From (1) and (2),

- MA(w_t) $\Rightarrow <a_1,u_1> \wedge ... \wedge <a_m,u_m> \wedge (<b_1,v_1> \vee ... \vee <b_n,v_n>)$(4)

From (3),

- MA(g_d) $\Rightarrow (<a_1, \neg u_1> \vee ... \vee <a_m, \neg u_m>) \vee$
  $<b_1, \neg v_1> \wedge ... \wedge <b_{n-1}, \neg v_{n-1}>$ (5)

Let $g = <a_1, u_1> \wedge ... \wedge <a_m, u_m> \wedge (<b_1, v_1> \vee ... \vee <b_n, v_n>)$.

Then MA(w_t) $\Rightarrow g$, and MA(g_d) $\Rightarrow \neg g$. From the 2-Way RAR Theorem [8], the connection ($g \rightarrow g_d$) or ($g_n \rightarrow\circ\ g_d$) can be added to replace the target wire w_t. □

From the proof above, we can learn that if we allow more iterations between the decisions from MA(w_t) and from MA(g_d), we will be able to construct arbitrarily level of circuits as the alternative logic. An informal way to understand this idea is that we are exploring the functional relationship between MA(w_t) and MA(g_d) by picking up an implication on one side to make the decision on the other, and perform logic implications to bring the relation closer. Once there is any inverse implication from both sides, an alternative circuit can be derived from the decisions.

IV. Implementation Details

We implemented our SatRAR algorithms on a circuit-based SAT solver. A circuit-based SAT solver implements all the logic implications, decision procedure and conflict-driven learning on the circuit data structure directly. Although it is also possible to realize our algorithms on a Conjunctive Normal Form (CNF)

Authorized licensed use limited to: National Taiwan University. Downloaded on December 03,2021 at 03:33:23 UTC from IEEE Xplore. Restrictions apply.

SAT solver, yet the transformation between circuit netlist and the CNF formulae will create too much overhead. Besides, recent work on circuit-based SAT solvers [15] shows that the circuit SAT can be as efficient as the state-of-the-art CNF ones.

The pseudo code of our SatRAR algorithm is as below:

Algorithm: SAT-Controlled RAR

```
1. SatRAR(w_t) {  // w_t: target wire
2.   A ← ∅;  // valid alternative wires
3.   decisionLevel ← 0;
4.   if (RedundancyCheck(w_t) has conflict)
5.     return REDUNDANT(w_t);
6.   Copy current implications as Φ_wt; // = MA(w_t)
7.   for i = k Down To 0  // For each dominator of w_t
8.     Backtrack decisionLevel to i;
9.     if (BCP(CNTR(g_di)) has conflict)
10.      return REDUNDANT(g_di);
11.    Copy current implications as Φ_i;// MA(g_di)
12.    for_each ({g_s|g_s ∈ inverseImp(Φ_wt, Φ_i)})
13.      A += MakeAltWire(g_s, g_di);
14.    for_each({g_s|g_s ∈ Φ_wt−Φ_i ∧ g_s ∉ fanoutCone(g_di)})
15.      decisionLevel += 1;
16.      if (Decision(g_s) has an inverse imp to MA(w_t))
17.        Let G = set of decisions that account for
18.                the inverse imp;
19.        A += MakeAltGate(AndGate(G), g_di);
20.      else if (Decision(g_s) has a conflict)
21.        Let G = set of decisions that lead to the
22.                conflict;
23.        A += MakeAltGate(AndGate(G), g_di);
24.        ConflictLearning();
25.  return A;
26. }
```

Figure 5: The basic SAT-Controlled RAR algorithm

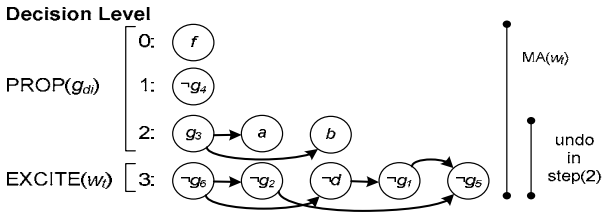
We first conduct the redundancy check on the target wire w_t and copy the implications to a set Φ_{wt} (lines 3 to 6). Please note that we name the dominators of w_t as g_{d0}, g_{d1}, ... and g_{dk}, where g_{d0} is closer to circuit output and g_{dk} is the direct fanout of w_t. By storing the implications derived from the fault propagation requirements on g_{d0}, g_{d1},... and g_{dk}, and the fault excitation condition on decision levels 0 to k+1, respectively, we can maximize the implication sharing in the later steps.

For example, as shown in Figure 6(b), to compute the MAs for the first dominator $g_{d2} = g_7$, we backtrack to level 2 and apply the g7 s-a-0 fault excitation ($g_7 = 1$) on this level. In this way, the implications before this level (i.e. levels 0 and 1) can be reused. In general, to compute the MAs for the output controlling fault of the i-th dominator g_{di}, we only need to backtrack to level i and all the implications before this level can be kept (lines 8 to 11). If there is any conflict, the dominator itself is redundant. Otherwise, we store the current implications as Φ_i in order to perform the set difference with Φ_{wt} later.

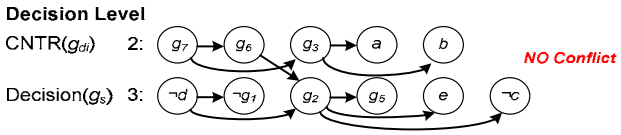
Assumption list can remains

Lines 12 and 13 are the original 2-way RAR algorithm, with no recursive learning called. In line 14, we select a gate from Φ_{wt} - Φ_i as the SAT decision. Please note that this is where we control the quality of SRAR. Intuitively, for timing optimization, we can pick the one with smaller arrival time. This will ensure the alternative wire with better timing. On the other hand, if we are working on post-layout optimization, we can choose the gate closer to g_d so that the delay of the alternative wire can be minimized.

In line 16 we make a decision on the selected gate (e.g. Level 3 in Fig. 6(b)). If the decision results in a inverse implication when compared to MA(w_t), we apply Theorem 2 and Corollary 2.2 to derive the alternative gate (lines 17-19). Otherwise, if there is an implication conflict (line 20), we resort to Theorem 1 and Corollary 2.1 to generate the alternative wire or gate (lines 21-23). Conflict-driven learning is performed at the end to record the learned implication (line 24).



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



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

Figure 6: An example for SAT-controlled RAR

The performance of the SatRAR algorithm can be further improved by the "*implication filter*". Please refer to the implication graph in Figure 6(b), where the decision <d, 0> results in no conflict with MA(g_d). Any implication of <d, 0>, say <g_1, 0>, should produce less implications than <d, 0> itself under the same implication context (i.e. MA(g_d)). Therefore, if we make decision on this implied gate (g_1), we will not encounter conflict either and thus can remove it from the decision list.

V. Experimental Results

We conducted two sets of experiments: one was to compare SatRAR and 2-way RAR with differnt recursive learning levels, and the other was to examine the varied options of our SAT-controlled RAR algorithm. We implemented both SatRAR and 2-Way RAR on the same circuit SAT solver so that the comparison can be fair. All the experiments were executed on a Linux 64-bit machine with Intel Xeon CPU 5130 2GHz and 16GB memory. The testcases were chosen from the larger circuits in ISCAS85 and ITC99 benchmarks.

The column "#wire" in the table refers to the total number of wires in a circuit. In our experiments, we go through each wire of the circuit to generate alternative wires/gates. However, to avoid the trivial cases, we do not generate alternative logic connected to the output of the target wire (i.e. g_{dk}). The columns "#tar" and "#alt" are the total number of target wires that can be replaced and the total number of alternative wires/gates generated for these target wires, respectively. The column "#T" records the runtime.

As shown in Table I, our SRAR_Wire engine is more robust than the 2-way RAR. — it can identify the largest numbers of alternative wires in almost all of the testcases. When compared to the 2-way RAR algorithms with level-1 and level-2 recursive learning (2WRAR_L1, 2WRAR_L2), our algorithm is 8.52 and 166 times faster (from Figure 7) and is the only one that can handle the larger cases. On the other hand, although SRAR_Wire runs about twice slower than the 2-way RAR without recursive learning (2WRAR_L0), we can identify 36%

Authorized licensed use limited to: National Taiwan University. Downloaded on December 03,2021 at 03:33:23 UTC from IEEE Xplore. Restrictions apply.

more of alternative wires. Please note that we can easily switch our SatRAR to 2WRAR_L0 by commenting out lines 14 to 24 in our algorithm (Figure 5).

Table II shows that the SRAR_Gate algorithm is much more powerful than SRAR_Wire. It can identify 12.6x more alternative logic with 34% overhead in runtime. Please note that the runtime can be easily controlled if we choose to make fewer decisions (e.g. focus on the ones with better optimization capability first). The "SRAR_Gate-Lm" and "SRAR_Gate-Imp" are two options to turn off the conflict learning and implication filter, respectively. We can see that it is disadvantageous to turn off the conflict learning. However, we will identify more alternative wires/gates if switching off the implication filter, but the runtime is almost tripled. On the average, the SRAR_Gate algorithm can identify alternative logic for 56.6% of wires in the testcases (#tar/#wire) (i.e. 56.6% wires can be replaced).

TABLE I: Comparison of SRAR_wire with 2-way RARs

Alg.		SRAR_wire		2WRAR_L0		2WRAR_L1		2WRAR_L2	
Cir.	#wire	#tar	#alt	#tar	#alt	#tar	#alt	#tar	#alt
C432	336	102	165	55	91	80	139	105	168
C499	408	40	40	40	40	40	40	40	40
C3540	2939	1272	3615	1195	3094	1236	3537	1278	3853
C6288	4800	1011	1043	100	116	1011	1043	1011	1043
C7552	6144	2262	7647	2063	6296	2257	7685	2272	7921
b01	80	12	17	4	4	12	17	12	17
b13	558	139	253	72	92	133	226	143	259
b14	1.9e4	9677	1.44e5	9238	1.28e5	9835	1.47e5	--	--
b15	1.7e4	5201	18435	4055	12128	5054	19704	--	--
b17	6.2e4	23111	1e5	19372	68339	22914	1.03e5	--	--
b18	2.2e5	97038	8.52e5	85705	6.82e5	95912	8.53e5	--	--
b19	4.3e5	1.95e5	1.51e6	1.72e5	1.13e6	--	--	--	--
Ratio		1	1	0.72	0.64	0.97	0.98	1.01	1.03

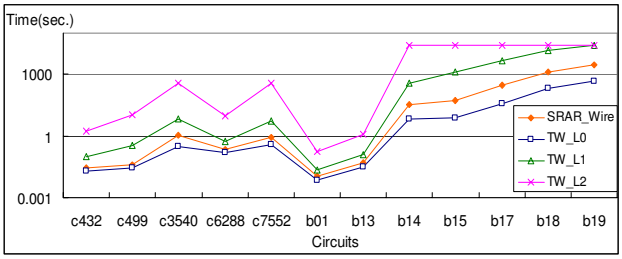


Figure 7: Runtime comparison of SRAR and 2-Way RAR

VI. Conclusions

In this paper we proposed a novel SAT-controlled RAR technique. We fully utilized the power of modern SAT engines so that the new RAR algorithm can be very efficient. Compared to previous RAR approaches, our RAR engine can significantly outperform them in both runtime and number of alternative wires/gates. Our RAR algorithm is also very flexible — we can choose the order of decision variables to gear the generation of alternative wires/gates toward the specific optimization objectives. In the near future, we will apply the SRAR engine for the post-layout optimization as well as the automatic engineering change order (ECO) problems.

References

[1] L. A. Entrena and K.-T. Cheng, "Combinational and Sequential logic Optimization by Redundancy Addition and Removal," in IEEE TCAD. vol. 14, no. 7, pp. 909-916, July 1995.

[2] C-Y. Huang, Y. Wang, and K.-T. Cheng, "Libra – A Library-Independent Framework for Post-Layout Performance Optimization," in ISPD, pp. 135-140, Apr. 1998.

[3] S-Y. Huang and K-T. Cheng, "Formal Equivalence Checking and Design Debugging," Kluwer Academic Publishers, 1998.

[4] C.-C. Lin, K.-C. Chen and M. Marek-Sadowska, "Logic Synthesis for Engineering Change," in IEEE TCAD, Mar. 1999.

[5] S.C. Chang, K.-T. Cheng, N.-S. Woo, and M. Marek-Sadowska, "Layout driven logic synthesis for FPGA," in Proc DAC, 1994.

[6] R. Bahar, M. Burns, G. Hachtel, E. Macii, H. Shin, F. Somenzi, "Symbolic Computation of Logic Implications for Technology-Dependent Low-Power Synthesis," in Proc ISSLPED, 1996.

[7] S.-C. Chang, M. Marek-Sadowska and K.-T. Cheng, "Perturb and Simplify: Multilevel Boolean Network Optimizer," in IEEE TCAD, vol. 15, no.12, pp 1494-1504, Dec. 1996.

[8] L. A. Entrena, J. A. Espejo, E. Olias and J. Uceda, "Timing Optimization by an Improved Redundancy Addition and Removal Technique," in Proceedings of EURO-DAC, 1996.

[9] C-W (Jim) Chang and M. Marek-Sadowska, "Single-Pass Redundancy Addition And Removal," in Proc. ICCAD, 2001.

[10] M. A. Iyer and M. Abramovici, "FIRE: A Fault-Independent Combinational Redundancy Identification Algorithm," in IEEE Trans. on VLSI, vol. 4, pp. 295–301, June 1996.

[11] M. Abranovici, M.A. Breuer, and A. D. Friedman, "Digital Systems Testing and Testable Design", Wiley Press, Rev. Ed, 1994.

[12] Oscar H. Ibarra, Sartaj Sahni, "Polynomially Complete Fault Detection Problems," in IEEE TComp. vol. 24, no. 3, 1975.

[13] W. C. Tang, W. H. Lo, T. K. Lam, K. K. Mok, C. K. Ho S. H. Yeung, H. B. Fan, and Y. L. Wu, "A Quantitative Comparison and Analysis on Rewiring Techniques," in Proceedings of International Conference on ASIC, vol.1, pp. 242-245, Oct. 2003.

[14] W. Kunz and D. K. Pradhan, "Recursive Learning: An Attractive Alternative to the Decision Tree, Test Generation in Digital Circuits," in Proc. of ITC, pp. 816-825, Oct. 1992.

[15] C-A. Wu, T-H. Lin, C-C. Lee, and C-Y (Ric) Huang, "QuteSAT: A Robust Circuit-based SAT Solver for Complex Circuit Structure", Proc. DATE, March 2007.

TABLE II: Comparison of different SAT-Controlled RAR options

Options		SRAR_Wire			SRAR_Gate			SRAR_Gate - Lrn			SRAR_Gate - Imp		
Cir.	#wires	#tar	#alt	#T	#tar	#alt	#T	#tar	#alt	#T	#tar	#alt	#T
C432	336	102	165	0.03	158	648	0.03	157	569	0.03	158	675	0.04
C499	408	40	40	0.04	64	136	0.04	64	136	0.04	64	136	0.05
C3540	2939	1272	3615	1.12	1965	24972	1.45	1964	22816	1.38	1965	31189	3.79
C6288	4800	1011	1043	0.21	1922	10910	0.36	1922	8896	0.30	1922	11029	0.39
C7552	6144	2262	7647	0.9	3734	52745	1.6	3734	49657	1.48	3734	61490	2.28
b01	80	12	17	0.01	47	314	0.01	47	327	0.01	47	336	0.01
b13	558	139	253	0.05	308	2678	0.08	305	2615	0.08	308	2911	0.11
b14	1.9e4	9677	1.44e5	32.1	14121	5.72e5	40.6	14117	5.16e5	37.8	14121	9.16e5	128
b15	1.7e4	5201	18435	54.4	10337	3.57e5	61	10331	2.75e5	55.1	10337	5.49e5	319
b17	6.2e4	23111	1e5	288	40464	2.95e6	353	40411	2.12e6	314	40464	4.06e6	1986
b18	2.2e5	97038	8.52e5	1337	1.46e5	1.57e7	1719	1.46e5	1.12e7	1541	1.46e5	2.1e7	9017
b19	4.3e5	1.95e5	1.51e6	2845	2.94e5	3.09e7	3656	2.94e5	2.20e7	3260	2.94e5	4.27e7	18785
Ratio		1	1	1	1.88	12.6	1.34	1.88	10.35	1.23	1.88	16.16	3.68

Authorized licensed use limited to: National Taiwan University. Downloaded on December 03,2021 at 03:33:23 UTC from IEEE Xplore. Restrictions apply.