Rewiring Using IRredundancy Removal and Addition*

Chun-Chi Lin
Dept. of Computer Science
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.
Email: idk.ccling@gmail.com

Chun-Yao Wang
Dept. of Computer Science
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.
Email: wcyao@cs.nthu.edu.tw

***Abstract*—Redundancy Addition and Removal (RAR) is a restructuring technique used in the synthesis and optimization of logic designs. It can remove an existing target wire and add an alternative wire in the circuit such that the functionality of the circuit is intact. However, not every irredundant target wire can be successfully removed due to some limitations. Thus, this paper proposes a new restructuring technique, IRredundancy Removal and Addition (IRRA), which successfully removes any desired target wire by constructing a rectification network which exactly corrects the error caused by removing the target wire.**

I. Introduction

Redundancy Addition and Removal (RAR) is a technique for reconstructing a circuit by adding some redundant wires or gates, named *alternative wire/gates*, and results in the removal of given *target wires*. In the process of RAR, the addition and the removal of redundant wires will restructure a circuit without changing the functionality. This circuit transformation technique is applicable to achieving optimization objectives such as area, timing, power, or reliability [2] [3] [4] [6] [7] of VLSI circuits.

One of the most commonly used approaches to RAR is Automatic Test Pattern Generation (ATPG)-based algorithm. ATPG-based approaches can be divided into two-stage algorithms and one-stage algorithms. In two-stage algorithms [2] [3] [4] [6] [7], a set of candidate wires that make the target wire become redundant is built up by finding the *Mandatory Assignments* (MAs). Then, redundancy tests which require much effort on each candidate wire are performed. On the other hand, one-stage algorithms [1] [5] identify alternative wires without redundancy tests which can significantly reduce the CPU time. However, the rewiring capability is not as good as that of two-stage algorithms.

A target wire has an alternative wire if the target wire become redundant after adding a redundant wire to the circuit. The capability of ATPG-based RAR for finding alternative wire is limited by the identified MAs. If a target wire does not have an alternative wire since the condition of MA is unsatisfied, the target wire cannot be removed [3] [6].

In this work, the RAR technique is viewed in an opposite way where it removes an irredundant target wire first and then adds an irredundant wire to rectify the functionality of the circuit. From this point of view, a technique of *IRredundancy Removal and Addition* (IRRA) for circuit restructuring is proposed. IRRA is an ATPG-based technique that can remove any desired irredundant target wire and rectify the error by adding some wires/gates. These added wires/gates are named *rectification network*. Thus, RAR is a special case of IRRA where the rectification network is just an alternative wire.

II. Notations and background

A Boolean network is a Directed Acyclic Graph where each node ni is associated with a Boolean variable yi and a Boolean function fi. There exists a connection directed from node ni to node nj if the function fj depends on the variable yi. An input to a gate has a *controlling value* if the value of the gate's output is determined by the input regardless of the other inputs. The *noncontrolling value* is the inverse of the controlling value.

The dominators [8] of a wire w is the set of gates G such that all paths from w to any primary outputs have to pass through all gates in G. Consider the dominators of w, the *fault propagating inputs* of a dominator are the inputs in the transitive fanout of w, and the other inputs are $side\ inputs$ of the dominator. In the process of test generation for a stuck-at fault at a wire $w(gi \rightarrow gj)$, gi is assigned to a controlling value to activate the fault effect and all side inputs of w's dominators are assigned to noncontrolling values to propagate the fault effect.

The MAs are the unique value assignments required for a test to exist. The *logic implication* is a process of computing MAs for a test. The MAs for a stuck-at fault test on w can be computed from setting the fault-activating value and setting noncontrolling values on the side inputs of w's dominators. Then the MAs can be propagated forward or backward to obtain more MAs. Recursive learning [9] can be applied to find more MAs. *Forced MA* [4] is an MA that causes the target fault untestable while violating it.

III. Irredundancy removal and addition

For the process of IRRA, as seen in Fig. 1, it removes an irredundant target wire wt first, and then it adds another irredundant wire wr to rectify the functionality of the circuit.

*This work was supported in part by the National Science Council of R.O.C. under Grant NSC97-2220-E-007-042 and NSC97-2220-E-007-034.

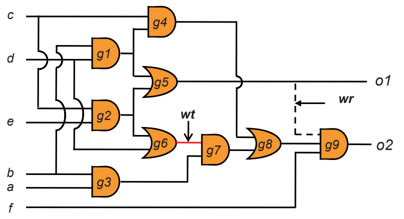

Fig. 1. An example of IRredundancy Removal and Addition.

The most important step is how to recognize the rectification network with respect to the injected error. To generate a test vector for the fault at wt, the MAs for wt are calculated. Among the MAs, Source MAs are defined as follows which will be used in our approach.

Definition 1: Given a set of MAs for a target fault, the *source MA (SMA)* is defined as an MA whose transitive fanin cone contains no other MAs.

After calculating MAs and SMAs, a destination gate gd is selected on which the rectification network is added. The destination is selected from the dominators of wt since a dominator is where the error must pass through. In this work, the gates of AND, OR, and INV are only considered and the error model is ”missing single wire”. Thus, the target wire contains at least one destination. To rectify the functionality of the circuit at gd, the differences between the good circuit and the faulty circuit at gd must be identified. The differences at gd are the minterms that changed from 0 to 1, and that changed from 1 to 0 after the removal of wt.

Definition 2: Given a Boolean network, a target wire, and a destination gate gd. The *Exact Addition Network (EAN)* at gd is defined as the network having minterms changed from 0 to 1 at gd after removing the target wire. The *Exact Removal Network (ERN)* at gd is defined as the network having minterms changed from 1 to 0 at gd after removing the target wire.

Theorem 1: Given a Boolean network, a target wire wt, and a destination gate gd. Suppose the cofactors of gd with respect to SMA in good/faulty circuits are denoted as $gd_{g(SMA)}$ and $gd_{f(SMA)}$, respectively, and the product of all SMAs is denoted as $AND(SMA)$. The Boolean function of EAN at gd is

$$AND(SMA) \cdot \overline{gd_{g(SMA)}} \cdot gd_{f(SMA)} \tag{1}$$

The Boolean function of ERN at gd is

$$AND(SMA) \cdot gd_{g(SMA)} \cdot \overline{gd_{f(SMA)}} \tag{2}$$

Theorem 2: Given a Boolean network, a target wire wt, a destination gate gd, the EAN at gd, and the ERN at gd. The functionality of $(gd+ERN)\cdot\overline{EAN}$ after removing wt is equivalent to that of the original Boolean network. The general scheme of the rectification network is as shown in Fig. 2(a).

Theorem 3: The EAN at gd can be simplified from (1) to

$$AND(SMA) \cdot \overline{gd_{g(SMA)}} \tag{3}$$

The ERN at gd can be simplified from (2) to

$$AND(SMA) \cdot gd_{g(SMA)} \tag{4}$$

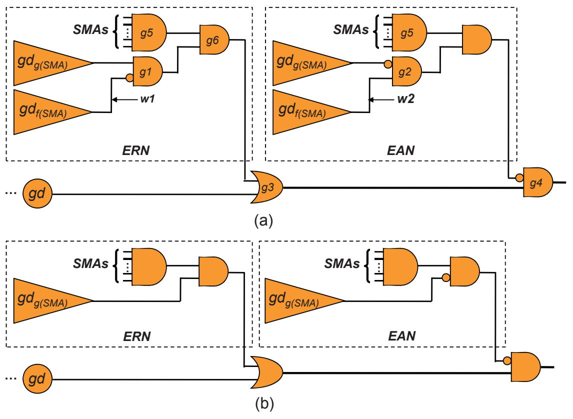

Fig. 2. (a) The general scheme of the rectification network. (b) The general scheme of the simplified rectification network.

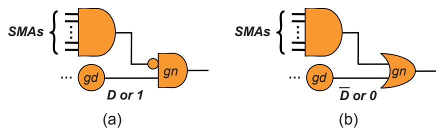

Fig. 3. IRRA schemes for alternative wires.

Proof: We prove this theorem by showing that the wires $w1$ in ERN and $w2$ in EAN of Fig. 2(a) are both redundant. Thus, the rectification network can be simplified as Fig. 2(b).

In Fig. 2(a), a s-a-0 test on $w1(gd_{f(SMA)} \rightarrow g1)$ is performed. $gd_{f(SMA)}$ has an MA of 1. All SMAs and $gd_{g(SMA)}$ are 1s. Since the SMAs are 1s and the current circuit is faulty, the value of gd is the same as $gd_{f(SMA)}$ and equals 1. However, gd has to be 0 for propagating the fault effect. Thus, $w1$ is a redundant wire, and the ERN formula in (2) can be simplified as (4). By the similar manner as $w1$, the wire $w2(gd_{f(SMA)} \rightarrow g2)$ in Fig. 2(a) is also a redundant wire. The EAN formula in (1) can be simplified as (3). ■

IV. Single alternative wire identification using the IRRA approach

In the RAR, a redundant wire wr is a *Forward Alternative Wire (FAW)* of wt if the addition wire wr blocks the fault effect. On the other hand, if the addition wire wr violates a forced MA, this wire is a *Backward Alternative Wire (BAW)* of wt.

In the IRRA, if the destination gate gd in Fig. 2(b) has an MA D $\{\overline{D}\}$, D $\{\overline{D}\}$ represents 0/1 in the good/faulty circuit, the value of $gd_{g(SMA)}$ will be 0 $\{1\}$. This causes the ERN $\{$EAN$\}$ be a constant 0 network and cause the EAN $\{$ERN$\}$ leave the $AND(SMA)$ term. Thus, the scheme becomes Fig. 3(a) $\{$Fig. 3(b)$\}$. The $AND(SMA)$ term in Fig. 3 can be seen as an MA 1 which blocks the fault effect at gn. Thus, the wire $(AND(SMA) \rightarrow gn)$ is the FAW of the target wire. On the other hand, if a gate which is not a dominator but has a forced MA 1 $\{0\}$, we can also violate the forced MA by adding the $AND(SMA)$ term. The scheme is also shown as Fig. 3(a) $\{$Fig. 3(b)$\}$. The new gate gn will have a forced MA 1 $\{0\}$. But $AND(SMA)$ term which has the MA 1 will cause the gn value become 0 $\{1\}$, this violates the forced MA at gn. Thus, the wire $(AND(SMA) \rightarrow gn)$ is the BAW of the target wire.

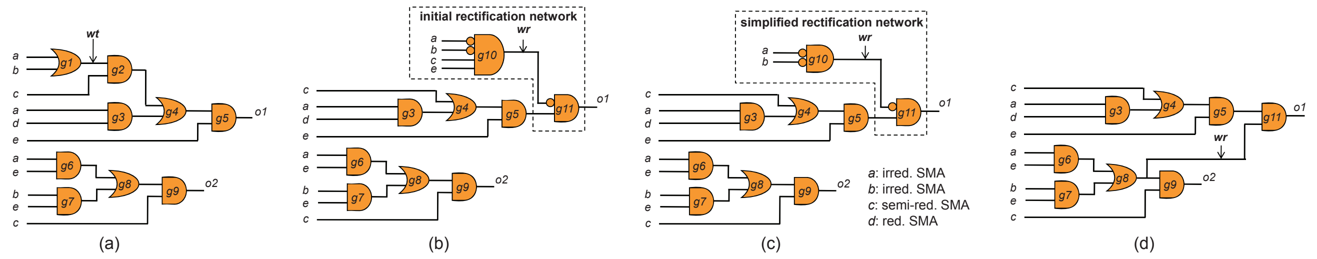

Fig. 4. An example of SMA classification and SMA substitution. (a) The original circuit with $wt(g1 \rightarrow g2)$. (b) After removing wt and adding the initial rectification network. (c) The simplified rectification network only remains irredundant SMAs. (d) The alternative wire $(g8 \rightarrow g11)$ for $wt(g1 \rightarrow g2)$ in (a).

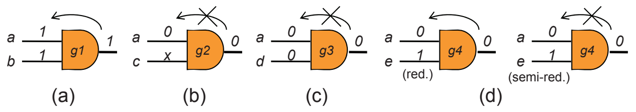

Fig. 5. Types of SMA substitution in an AND gate.

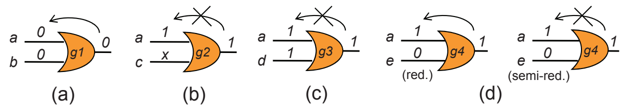

Fig. 6. Types of SMA substitution in an OR gate.

To find single alternative wires of a target wire, the $AND(SMA)$ term in Fig. 3 must be reduced to only one MA. This can be achieved by *SMA classification* and *SMA substitution* procedures.

A. SMA classification

Some of SMAs are redundant and can be removed in the IRRA schemes in Fig. 3. SMAs are classified into three types in this work. The first type is *irredundant SMA*, which is irredundant after removing wt. The second type is *redundant SMA*, which is redundant after removing wt. The irredundant SMAs in the $AND(SMA)$ term are don't care values when performing the redundancy test on this SMA. The last one is *semi-redundant SMA*, which is also redundant after removing wt, but the irredundant SMAs in the $AND(SMA)$ term have to be set specific values while performing the redundancy tests on the semi-redundant SMAs.

For example in Fig. 4(a), suppose the target wire is $wt(g1 \rightarrow g2)$. After calculating the MAs and the SMAs, $g5$ has D value, by referring Fig. 3(a), the initial rectification network is as shown in Fig. 4(b). The SMAs a and b are irredundant SMAs. For the s-a-1 test on the wire $(e \rightarrow g10)$, the SMA e is a redundant SMA. For the s-a-1 test on the wire $(c \rightarrow g10)$, the SMA c is a semi-redundant SMA. After SMA classification, Fig. 4(b) can be simplified as Fig. 4(c).

B. SMA substitution

To reduce the remaining irredundant SMAs to only one MA, we have to find an MA that can substitute for all irredundant SMAs. We say $g1$ can substitute for $g2$ if $g1$ implies $g2$. For each irredundant SMA, a set of MAs for substitution can be derived. Thus, an MA that is in the intersection of each irredundant SMA's substitution set can be identified for substituting for all SMAs. This MA is one end point of the alternative wire.

With the MAs computed from the wt and SMA classification, Fig. 5 lists all types of substitution relationships for an AND gate. Suppose a is an irredundant SMA and b, c, d, and e are known MAs. The purpose is to determine whether the value of a can be backward implied from the output of the gate. Fig. 5(a)-5(c) are the trivial cases. In Fig. 5(d), a is a controlling value 0 and e has a noncontrolling value 1. If e is a redundant SMA, $g4 = 0$ can imply $a = 0$, and $g4$ can replace a. On the other hand, if e is a semi-redundant SMA, we cannot ensure whether $g4 = 0$ can imply $a = 0$. This is because if e depends on the irredundant SMA a, e is unknown before $a = 0$ is implied. Thus, $a = 0$ cannot be implied from $g4 = 0$. The types of substitution relationships for an OR gate are summarized in Fig. 6.

For the last example in Fig. 4(c), by referring Fig. 5 and Fig. 6, the substitution set for the irredundant SMA a is $\{g6, g8\}$. By the same manner as a, the substitution set for the irredundant SMA b is $\{g7, g8\}$. The intersection of these two sets is $\{g8\}$. Thus, the wire $(g8 \rightarrow g11)$ is a single alternative wire for wt. The final circuit is as shown in Fig. 4(d).

V. Experimental Results

Two experiments are conducted to demonstrate the effectiveness of the proposed IRRA technique.

The proposed single alternative wire identification algorithm was implemented in C and the experiments were conducted over a set of ISCAS85 and MCNC benchmarks within SIS [10] environment on a Sun Blade 2500 workstation with 4GBytes memory. Since the circuits under consideration are only consist of AND, OR, and INV gates, we decompose the complex gates into these primitive 2-input gates by using *decomp_tech_network* command in SIS. Additionally, recursive learning technique is applied in these experiments with depth=1.

Table I shows the results for the single alternative wire identification that compared with the previous work [5] on the same platform. Nt represents the total number of target wires in each benchmark. % represents the percentage of target wires having alternative wires. Time(S) represents the CPU time measured in seconds. For example in c1908 circuit, the percentage of target wires having single alternative wires in [5] and ours are 47.95% and 66.15%, respectively. The CPU time needed are 169.85 seconds and 70.14 seconds, respectively.

According to Table I, our approach gets 16% improvement on the percentage of target wires having alternative wires,

TABLE I
COMPARISON OF EXPERIMENTAL RESULTS BETWEEN [5] AND OUR APPROACH FOR SINGLE ALTERNATIVE WIRE IDENTIFICATION.

Circuit	N_t	[5]		Ours		Circuit	N_t	[5]		Ours	
		%	Time (S)	%	Time (S)			%	Time (S)	%	Time (S)
C432	338	52.37	5.84	77.22	7	dalu	2904	65.94	3347.25	77	389.51
C880	692	38.29	10.74	54.48	8.54	example2	578	30.97	15.07	44.81	6.59
C1908	**1220**	**47.95**	**169.85**	**66.15**	**70.14**	frg2	3250	73.26	2984.58	77.85	323.63
C2670	1348	27.67	82.86	40.36	70.42	go	134	69.4	0.91	85.82	0.49
C3540	2336	40.33	299.3	50.86	290.43	i10	4620	44.22	2276.46	54.74	1179.46
C5315	4022	38.17	392.52	45.43	291.54	lal	374	74.06	11	81.55	3.49
C7552	4946	51.62	1137.04	53.19	615.61	mux	112	46.43	0.57	62.5	0.38
9symml	468	50	13.32	57.05	10.68	pair	3038	48.03	277.51	62.24	163.02
alu2	898	60.91	139.89	62.81	47.34	pcler8	156	26.92	0.88	31.41	0.59
alu4	1794	60.54	660.91	62.88	227.07	pm1	134	70.9	0.91	85.07	0.52
apex6	1332	22.15	24.27	32.81	22.79	rot	2242	61.82	415.75	73.6	188.04
apex7	512	48.24	7.26	64.26	5.05	sct	382	72.51	18.73	80.37	6.97
b9	258	60.85	2.02	87.6	1.41	term1	980	76.33	60.92	78.98	26.45
c8	436	57.11	16.25	66.06	5.25	ttt2	738	76.83	46.66	83.2	12.55
cc	154	61.69	1.28	77.92	0.53	unreg	224	35.71	1.03	50.89	0.67
comp	194	59.79	1.75	75.77	1.32	x3	2526	67.38	412.03	68.29	92.61
cu	142	75.35	2.21	76.06	0.68	x4	1364	63.64	209.58	76.32	38.58
Total	44846	52.91	13047.15	61.48	4109.35						
Ratio1	-	1	-	1.16	-						
Ratio2	-	-	1	-	0.31						

TABLE II
COMPARISON OF EXPERIMENTAL RESULTS BETWEEN [3] AND OUR APPROACH FOR AREA OPTIMIZATION.

Circuit	SIS	[3]		Ours	
	| gate |	| gate |	Time (S)	| gate |	Time (S)
9symml	209	-	-	179	360.73
c432	193	-	-	129	79.43
c880	357	-	-	310	119.9
c1908	400	-	-	360	977.68
c2670	654	-	-	560	1306.72
c3540	1057	938	5692.8	969	1099.96
c5315	1532	1321	2236.7	1391	3852.46
c7552	1892	1426	3668.6	1466	12526.04
alu2	**364**	**281**	**1127.4**	**255**	**289.01**
alu4	699	555	4171.5	549	1111.87
apex6	644	543	568.9	537	192.15
apex7	200	-	-	158	69.87
comp	118	84	51.9	78	37.64
dalu	1105	-	-	758	357
f51m	116	78	4.7	72	22.55
frg2	797	-	-	649	206.85
pcler8	71	64	29.7	64	9.43
rot	563	452	256	460	160.78
term1	202	113	56.2	97	24.68
ttt2	172	118	57.8	115	62.07
x3	614	552	472	554	204.56
x4	308	-	-	276	30.33
Total1	12443	-	-	10105	23247.01
Ratio1	1	-	-	0.81	-
Total2	-	6641	18423.3	6726	19704.36
Ratio2	-	1	1	1.01	1.07

and only requires 31% CPU time of [5]. It shows that our approach is more efficient and effective on single alternative wire identification as compared with [5].

The second experiment is to use IRRA for area optimization which contains two stages. The first stage greedily removes many target wires that have a common alternative wire. We add this common alternative wire to remove the target wires as many as possible. This stage straightforwardly minimizes the circuit area. When the first stage cannot further minimize the circuit size, the second stage begins. In the second stage, a target wire is selected to perform IRRA. This may result in significant change of the circuit structure by adding the rectification network. It acts as a stimulus for escaping from a local minimal point during the optimization process.

The area optimization was implemented on the same platform. The benchmark circuits are initialized by using *script.algebraic* command, and we compare the results of our approach against that of SIS and Perturb/Simplify algorithm in [3] in term of the number of 2-input gates.

In Table II, the results in Column SIS are the best results obtained by the commands *script.algebraic* or *script.boolean*. For example, alu2 circuit has 364 gates by using the script

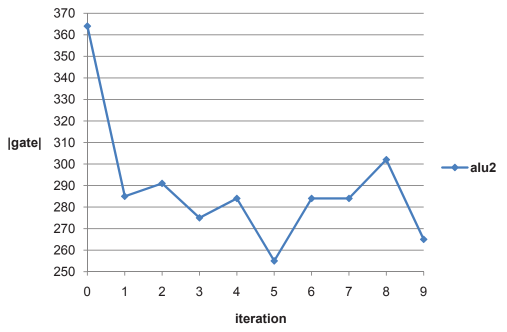

Fig. 7. The detailed analysis of area optimization for the circuit alu2.

of SIS. It is optimized to 281 gates by using Perturb/Simplify algorithm in [3] in 1127.4 seconds. With our approach, however, only 255 gates are left in 289.01 seconds. According to Row Ratio1 in Table II, our results are 19% smaller than that obtained by SIS. As compared with [3], the IRRA approach with this preliminary implementation is also competitive.

The detailed analysis of area optimization for the circuit alu2 is shown in Fig. 7. Iterations 1, 3, 5, 7, and 9 are the points to start the second stage which are the local optimal points. Iterations 2, 4, 6, and 8 are the points after the IRRA is applied. It can be seen that our approach optimizes the circuit area by escaping the local optimal points.

VI. CONCLUSIONS

This paper proposes a logic restructuring technique, IRRA. IRRA can remove any desired wire and rectify the error due to the removal of the wire. A single alternative wire identification procedure is also proposed from the IRRA technique. The experimental results show the effectiveness and efficiency of our approach as compared with the state-of-the-art work. The application of area optimization with the IRRA technique is also demonstrated in this paper. It is very promising that the characteristic of IRRA allows restructuring the circuits more widely such that different optimization objectives could be achieved.

REFERENCES

[1] C.-W. Jim Chang, et al, "A New Reasoning Scheme for Efficient Redundancy Addition and Removal," *IEEE TCAD.*, vol. 22, pp. 945-952, July 2003.
[2] S.-C. Chang, et al, "Postlayout Logic Restructuring Using Alternative Wires," *IEEE TCAD.*, vol. 16, pp. 587-596, June 1997.
[3] S.-C. Chang, et al, "Perturb and Simplify: Multi-level Boolean Network Optimizer," *IEEE TCAD.*, vol. 15, pp. 1494-1504, Dec. 1996.
[4] S.-C. Chang, et al, "Fast Boolean Optimization by Rewiring," in *Proc. ICCAD.*, pp. 262-269, 1996.
[5] Y.-C Chen, et al, "An Improved Approach for Alternative Wires Identification," in *Proc. ICCD.*, pp. 711-716, 2005.
[6] K. T. Cheng, et al, "Multi-level Logic Optimization by Redundancy Addition and Removal," in *Proc. Europ. Conf. Design Automation*, pp. 373-377, 1993.
[7] L. A. Entrena, et al, "Combinational and Sequential Logic Optimization by Redundancy Addition and Removal," *IEEE TCAD.*, vol. 14, pp. 909-916, July 1995.
[8] T. Kirkland, et al, "A Topological Search Algorithm for ATPG," in *Proc. DAC.*, pp. 502-508, 1987.
[9] W. Kunz, et al, "Recursive Learning: An Attractive Alternative to the Decision Tree for Test Generation in Digital Circuits", in *Proc. ITC.*, pp. 816-825, 1992.
[10] E. M. Sentovich, et al, "SIS: A System for Sequential Circuit Synthesis," Technical Report UCB/ERL M92/41, May 1992.