Almost Every Wire Is Removable: A Modeling and Solution for Removing Any Circuit Wire

Xiaoqing Yang, Tak-Kei Lam, Wai-Chung Tang, Yu-Liang Wu
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, Hong Kong
Email: {xqyang, tklam, wctang, ylw}@cse.cuhk.edu.hk

***Abstract*—Rewiring is a flexible and useful logic transformation technique through which a target wire can be removed by adding its alternative logics without changing the circuit functionality. In today's deep sub-micron era, circuit wires have become a dominating factor in most EDA processes and there are situations where removing a certain set of (perhaps extremely unwanted) wires is very useful. However, it has been experimentally suggested that the rewiring rate (percentage of original circuit wires being removable by rewiring) is only 30 to 40 % for optimized circuits in the past. In this paper, we propose a generalized error cancellation modeling and flow to show that theoretically almost every circuit wire is removable under this flow. In the Flow graph Error Cancellation based Rewiring (FECR) scheme we propose here, a rewiring rate of 95% of even optimized circuits is obtainable under this scheme, affirming the basic claim of this paper. To our knowledge, this is the first known rewiring scheme being able to achieve this near complete rewiring rate. Consequently, this wire-removal process can now be considered as a powerful atomic and universal operation for logic transformations, as virtually every circuit node can also be removed through repetitions of this rewiring process. Besides, this modeling can also serve as a general framework containing many other rewiring techniques as its special cases.**

I. Introduction

Following Gordon E. Moore's prediction, the size of IC transistors has been continually and drastically scaled down over the past decades. As a direct result, wire delay has become a much more dominating delay factor (compared with cell delay) in the timing closure problems of contemporary IC designs. Then, a situation is often encountered: how to remove a highly unwanted wire without changing the functionality of the circuit, where this wire, amongst many other problematic situations, may be located at a critical path, congested routing area, or partition boundary? The designer may want to remove it "at all other costs", as it may happen to be the last obstacle to the job completion. One of the solution options can be rewiring, a technique developed to deal with the problematic wires directly.

Rewiring is a circuit restructuring technique to reconnect wires without changing the circuit functionality. It has been widely applied in many EDA fields, such as circuit partition [1], circuit optimization [2] [3] . The major rewiring techniques include Automatic Test Pattern Generation (ATPG) based Redundancy Addition and Removal (RAR) [3] [4], Set of Pairs of Functions to be Distinguished (SPFD) [5], Graph Based Alternative Wiring (GBAW) [1] and Error Cancellation based Rewiring (ECR) [6] [7].

978-3-9810801-8-6/DATE12/

In RAR, for a given irredundant target wire w_t in a circuit C, w_a is an alternative wire for w_t if both w_a and w_t are redundant in the circuit $C + w_a$. Since w_t has become redundant, it can be removed and the following equation holds:

$$C = C + w_a = C + w_a - w_t \tag{1}$$

In [6], the authors suggested a symbolic based model of the following error cancellation concept. The removal of an irredundant wire w_t from the circuit C induces an error w_t_error. The addition of an originally non-existing wire w_a introduces an error w_a_error. If the w_t_error can be cancelled before it is propagated to any primary outputs by adding w_a_error, w_a is an alternative wire for w_t. In other words, w_t_error and w_a_error, where neither w_t nor w_a needs to be redundant, mutually and completely cancel one another before reaching any primary outputs.

The work in [6] has paved some theoretical foundation for error cancellation based rewiring, but the procedure is quite computation intensive and impractical for implementation. The authors in [7] proposed a low complexity heuristic for error cancellation based rewiring scheme. They re-examined the traditional view on dominators, which are derived from structural circuit traversal. The concept of dominators was refined to be not simply graph topology based but also error type dependent.

This method has significantly increased the rewiring capability, however, is still limited to the "one alternative wire for one target wire (1-to-1)" rewiring scenario. SPFD is a quite flexible scheme being able to transcend this constraint but tends to be too CPU intensive. The capability of all these known rewiring schemes is limited to a rewiring rate of 30-40% only for optimized circuits.

In this paper, we will further analyze the theory on error cancellation and propose a systematic Flow Graph Error Cancellation Rewiring (FECR) solution for our rewiring operation. FECR is a general 1-to-k (k alternative wires for 1 target wire) rewiring scheme where a flow graph is constructed to model error propagations such that the error caused by the target wire removal is to be cancelled at the minimum cut(s) of the graph. To cancel the error, a rectification network is temporarily added into the circuit. A node addition and removal (NAR) [8] based algorithm is then employed to find a source node,

which can be an existing node or a newly added node in the circuit, to replace the rectification network.

The rest of this paper is organized as follows: In Section II we briefly review the related concepts and related works. Flow-based ECR is detailed in Section III. Finally, experimental results and conclusion are presented in Section IV and V.

II. Preliminaries & Related Works

A Directed Acyclic Graph (DAG) can be used to represent a Boolean network. Logic gates are represented by vertices while wires are denoted by edges. The value that can determine a logic operator (gate) g's value is called the *controlling value* of g. Similarly, the *non-controlling* is the complement of the controlling value and cannot determine the logic output of g alone.

The *dominators* of a wire w are a set of gates G such that all paths from w to any primary outputs (PO) must go through these gates. To propagate w's stuck-at fault through a dominator, all the inputs of that dominator not lying in w's fault propagation path to POs, which are called *side inputs*, should be assigned with their non-controlling values. Let Γ be the set of all input vectors that can activate and propagate a certain stuck-at fault. A node k has a *mandatory assignment* (MA) m if the value of k is always m in all vectors in Γ. If the stuck-at fault is untestable when the value of k is changed to $\overline{m}$, m is a *forced mandatory assignment* (FMA).

Previous ATPG based rewiring schemes use dominators to find MAs and efficiently identify alternative wires. They commonly apply the concept of redundancy addition and removal: if a node is a dominator of the chosen target wire or has an FMA, it can be considered as a candidate alternative wire's destination. The source of an alternative wire would be another node with an MA. If the candidate alternative wire is originally redundant, and causes the target wire to become redundant, it is a valid alternative wire for the target wire. Note that all candidate alternative wires having destination nodes with FMAs must be *verified* to ensure that they are redundant. In some cases, however, the information at a traditional structural dominator is not sufficient to determine the alternative wires of certain target wires. The authors of [7] extended the concept and defined *testing dominators* as follows:

Definition 1: Given a network, a node set S is a blocking cut set of a wire w if every path from wire w to primary outputs must go through one and only one node in S.

Definition 2: Given a network, a node n_d is a testing dominator of the wire w's stuck-at fault if the fault must be propagated through n_d to primary outputs.

A testing dominator n_d is a special node in the blocking cut set D of w such that the stuck-at fault at w must be propagated through n_d and cannot be propagated through any other nodes in D. Beware that a testing dominator of a certain wire is fault-type dependent while a traditional dominator is not. And for a wire, its testing dominator set is a super set of its dominator set.

In Fig. 1, $\{g_7, g_8\}$ is a blocking cut set for wire $b \rightarrow g_1$. The stuck-at-0 fault for $b \rightarrow g_1$ can be propagated through

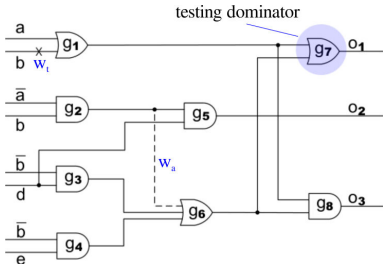


Fig. 1: General wire addition and removal

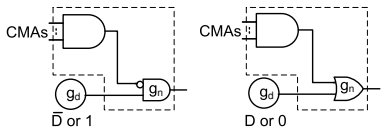


Fig. 2: Using rectification network for rewiring [4]

g_7 or g_8. To activate the stuck-at-0 fault for $b \rightarrow g_1$, b has to be set to 1. $b = 1$ implies $g_6 = 0$. Under this condition, the fault propagation path through g_8 is blocked. Since the fault can only be propagated through g_7, g_7 is a testing dominator in the stuck-at-0 fault test of $b \rightarrow g_1$.

Testing dominators, like dominators, can be considered as potential alternative wire destinations. In ECR, the authors suggested a way to identify alternative wires having testing dominators or nodes with FMAs as the destinations. The related definitions and formulations [4] [7] are to be presented again for completeness.

Definition 3: Given a set of MAs for a target fault, a *core MA* (CMA) is defined as a subset of MAs which can completely represent all the MAs.

All CMAs represent the vectors that can activate and propagate the target wire's stuck-at fault. *AND(CMA)* is used to represent such vectors (the product of all CMAs). Since only under those test vectors in AND(CMA) could the w_t_error be activated and propagated to primary outputs. w_a_error must also be activated and propagated under the same condition, AND(CMA).

Given a Boolean network, a target wire w_t, and a dominator g_d, suppose the cofactors of g_d with respect to the CMAs are denoted as $g_{d_{g(CMA)}}$ and $g_{d_{f(CMA)}}$ in the good and faulty circuits respectively. The *Exact Addition Network (EAN)* at g_d is:

$$EAN = AND(CMA) \cdot \overline{g_{d_{g(CMA)}}} \tag{2}$$

The *Exact Removal Network (ERN)* at g_d is:

$$ERN = AND(CMA) \cdot g_{d_{g(CMA)}} \tag{3}$$

Then, the functionality of the node $(g_d + ERN) \cdot \overline{EAN}$ after removing w_t is equivalent to that of g_d in the original circuit.

The authors in [4] makes use of a rectification network, which consists of the node g_n and the AND gate (AND(CMA)) having CMAs as its inputs (Fig. 2), to correct error caused

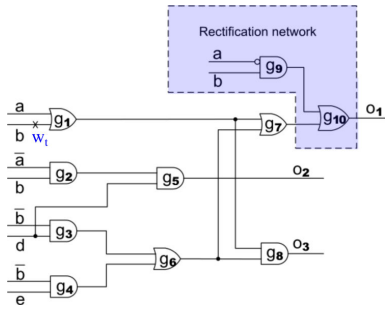


Fig. 3: Error being rectified at a testing dominator with rectification network

by w_t removal. After identifying the CMAs and then the rectification network as the destination of the alternative wire, the source of the alternative wire is found by node substitution techniques. A similar but improved scheme was proposed in [7]. Besides dominators, testing dominators and nodes with FMA can also be candidate alternative wires' destination nodes. For instance, we can construct a rectification network based on the computed CMAs ($\{a = 0, b = 1\}$ and g_7 = $D(1/0)$, as shown in Fig. 3.

Being more general than the traditional RAR schemes, this approach can consider irredundant wire/gate additions, and many more valid candidate alternative wire destinations can be identified through the use of testing dominators.

III. Flow Graph Based ECR

Although the approach of ECR [7] is already more general than traditional RAR schemes, it is still limited to the "one alternative wire for one target wire" constraint. To lift this constraint, partly inspired by the ideas introduced in Global Flow Optimization (GFO) and Implication Flow Graph (IFG) [9], a Flow Graph Error Cancellation Rewiring algorithm (FECR) is proposed to explore a much broader solution space. In GFO, the problem of wire re-connections is modeled as a flow graph optimization that can be solved by a maxflow-mincut algorithm. Implication flow graph is applied to reformulate the problem of fanout/fanin reconnections.

In FECR, the error propagation paths are firstly identified. Rectification networks are then added into the circuit to cancel the errors found in the error propagation graph. Min-cuts of the error propagation graph are then found for efficient searching of destination nodes. Below, we will introduce the flow in more details with illustrative examples.

A. Error Flow Graph Construction

In order to activate and propagate the w_t_error, the target wire and all the side inputs of dominators are assigned with suitable MAs. Other MAs are then obtained by logic implications. With all MAs collected, we construct the *error flow graph* (also called *error propagation graph*) as follows:

1) The target wire's sink node is added as the source node (root) s of the error flow graph. This node is given infinite (inf) weight.

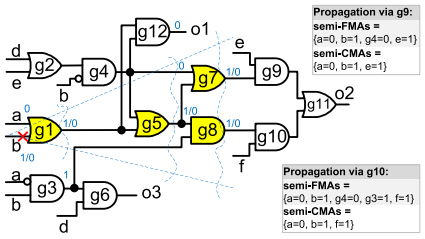
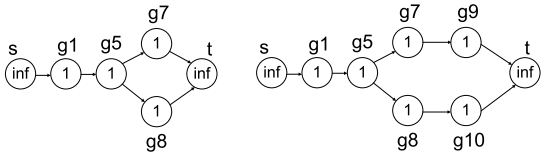


Fig. 5: Error propagation of w_t_error



(a) Error flow graph without Semi-MA (b) Error flow graph with Semi-MA

Fig. 6: Error flow graphs

2) All nodes having implied values 1/0 or 0/1 are added. Initially, all such nodes have weight = 1.
3) A sink node t with infinite weight is added to the graph.
4) Edges are inserted between nodes following the error propagation path of the circuit. A node with at least one fanout without MA is connected to the sink directly.

For example, suppose that $b \rightarrow g1$ is the target wire in the circuit in Fig. 5. To activate the w_t_error and propagate it through $g1$, the logic values at wire $b \rightarrow g1$ and a are set to 1/0 and 0 respectively. After implications, the following MAs are obtained: $\{a = 0, b = 1/0, g1 = 1/0, g3 = 1, g4 = 0, g12 = 0, g5 = 1/0, g7 = 1/0, g8 = 1/0\}$. The FMAs are $\{a = 0, b = 1, g4 = 0\}$ and the CMAs are $\{a = 0, b = 1\}$. Fig. 6a illustrates the error flow graph created according to the construction steps. Although $g12$ is in the transitive fanout cone of w_t_error and has an MA, it does not exist in the error flow graph because the value at $g12$ remains 0 in both the good circuit and the faulty circuit. This means the error cannot be propagated through $g12$, and therefore it should be excluded from the error flow graph.

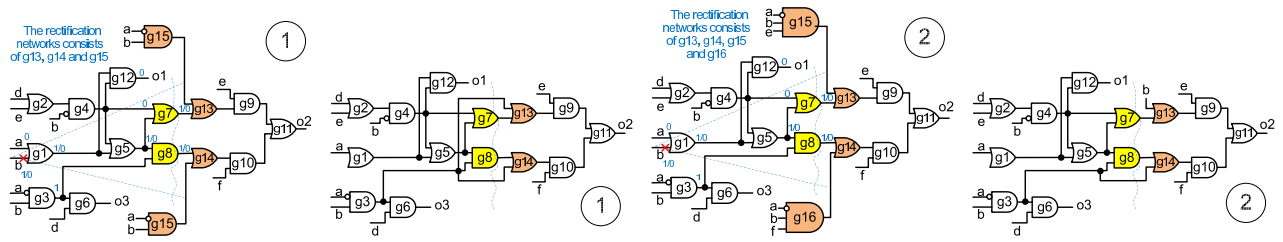
In the next subsection, we will explain how to use min-cuts of sizes 1 and 2 from the constructed error flow graph to identify candidate destination nodes.

B. Destination Node Identification

Definition 4: Given a network, a node set S is an *E-frontier* of a wire n's stuck-at-1 or stuck-at-0 fault if:

1) Every node in S must have an MA.
2) For each path from node s_i ($s_i \in S$) to any primary output o: $s_i \rightarrow n_1 \rightarrow n_2 \rightarrow ... \rightarrow n_k \rightarrow o$, any node n_j ($j \in [1, k]$) is not in S.
3) Every erroneous path from wire n to any primary output must go through at least one node in S.
4) Each node in S is reachable from n.

Definition 5: Given a network C, a stuck-at fault s, and a node t which is in the transitive fanout cone of s, a node n has a *semi-MA* x (x is 0 or 1) with respect to t if:



(a) Addition of the rectification network - first approach (b) Rewired circuit - first approach (c) Addition of the rectification network - second approach (d) Rewired circuit - second approach

Fig. 4: Example of Flow Graph Error Cancellation Rewiring (FECR)

1) Γ is the set of test vectors which can activate s and propagate it to primary outputs via node t.
2) n has the same value x in all test vectors in Γ.

Similar to the semi-FMA, a semi-CMA can be defined accordingly. For example, in Fig. 5, we can compute both sets of semi-FMAs and semi-CMAs with respect to $g9$ and $g10$ respectively.

The product of semi-CMAs, AND(semi-CMA) with respect to a node t represents the test vectors that can propagate the error via t. Each min-cut in the error flow graph is an E-frontier. Then, an error is not observable if it is stopped from propagating through an E-frontier without injecting any new error effects outside the error propagation graph. All E-frontiers of various sizes are determined by iterations of the maxflow-mincut algorithm. For example, the E-frontiers in Fig. 5, are $\{g1\}$ and $\{g5\}$ when cut size is 1, and $\{g7, g8\}$ when cut size is 2. The error can thus be cancelled at $\{g1\}$, $\{g5\}$ or $\{g7, g8\}$. When the number of nodes in an E-frontier is more than one, multiple errors may need to be injected into the circuit to cancel the w_t_error. In our example, the FMAs are $\{a = 0, b = 1, g4 = 0\}$. Let us assume that the E-frontier is $\{g7, g8\}$. If the w_t_error is to be propagated via $g9$, e has to be assigned with 1. Semi-FMAs are found to be $\{a = 0, b = 1, g4 = 0, e = 1\}$. The corresponding semi-CMAs are then $\{a = 0, b = 1, e = 1\}$. On the other hand, if the w_t_error is to be propagated via node $g10$, the node f has to be 1. Semi-FMAs are found to be $\{a = 0, b = 1, g4 = 0, g3 = 1, f = 1\}$, and the corresponding semi-CMAs are $\{a = 0, b = 1, f = 1\}$. The error flow graph is shown in Fig. 6b.

Rectification networks, which are constructed as AND(semi-CMA), can then be added at E-frontiers or semi-FMAs to cancel the error. Candidate alternative wire destinations can be:

1) nodes in the E-frontiers
2) nodes that have been assigned semi-FMAs during the error propagation via E-frontier nodes

If there is only one node in the E-frontier, this destination is a valid destination node. Otherwise, additional consistency check must be done to ensure the validity of the rewired circuit. Theorems related to the destination node identification are discussed below.

Theorem 3.1: Suppose that a node m has semi-FMAs in different propagation paths via the nodes in the same E-frontier, then m is not a unique candidate destination (more than one candidate destinations are necessary) for cancelling the error under the following conditions:

1) m has different semi-FMA values for different propagation paths.
2) The intersection Γ of the test vectors corresponding to the different propagation paths is non-empty.

Proof: Suppose the semi-FMA for m is 0 when the corresponding E-frontier node is v. And in another propagation path, when the corresponding E-frontier node is u, m's semi-FMA is 1. In other words, when the w_t_error is propagating through node v, the value of m is 0; whereas when the w_t_error is propagating through node u, the value of m is 1. Let Γ_u and Γ_v denote the test vectors for the propagation path through node v and u respectively. Then consider any test vector $t \in \Gamma = \Gamma_u \bigcap \Gamma_v$.

When t is applied to test w_t_error, if rectification is made at node m only, the error cannot be cancelled no matter what the value of m is. This is because when the value of m is 0, the w_t_error can always be propagated through the node v or otherwise through the node u. ■

More than one candidate destinations are required to correct the w_t_error even if the E-frontier has only one node. Theorem 3.1 gives the conditions for filtering out impossible single candidate alternative wire destinations. For instance, if the E-frontier is $\{g7, g8\}$ in Fig. 5, each of the $g7$ and $g8$ cannot be the only single candidate alternative wire destination alone for rectifying the w_t_error.

Theorem 3.2: Suppose there are two nodes u and v in the same E-frontier. The corresponding semi-MAs for each of these E-frontier nodes are found by propagating the w_t_error through u and v respectively. Denote the AND(semi-CMA) of a node n as AND(semi-CMA(n)). If w_t_error is to be corrected at both nodes u and v at the same time, the necessity of consistency check is based on the following conditions:

1) If AND(semi-CMA(u)) $\bigcap$ AND(semi-CMA(v)) $= \Phi$, consistency check can be exempted.
2) If AND(semi-CMA(u)) $\bigcap$ AND(semi-CMA(v)) $\neq \Phi$, consistency check is necessary for AND(semi-CMA(u)) $\bigcap$ AND(semi-CMA(v)).

Proof: Suppose that the intersection is empty. Since AND(semi-CMA(u)) represents the necessary condition for all the test vectors that can distinguish between the good circuit and rewired circuit when the w_t_error is propagated through u, it is impossible for AND(semi-CMA(v)) to affect the error propagation through u. The two test vectors cannot be effective simultaneously and therefore a consistency check is not necessary. ■

For instance, a consistency check is necessary if the w_t_error is rectified at both $\{g7, g8\}$ in the example shown in Fig. 5. Our FECR algorithm is summarized in Algorithm 1.

C. Source Node Identification

After identifying candidate destination nodes, the next task is to figure out how the rectification networks should be constructed.

1) First approach: For all the candidate destination nodes, a rectification network can be constructed using the AND(CMA) obtained from the stuck-at fault test of the w_t_error. Then, substitution nodes for the rectification network are identified as the alternative wires' sources using the techniques presented in [10] and [8].

Referring to the example in Fig. 4, if $\{g7, g8\}$ are the candidate destination nodes, the circuit after adding the rectification network is shown in Fig. 4a. The substitution node for the rectification network is found to be $g3$. The target wire $b \rightarrow g1$ is removed. Fig. 4b shows the rewired circuit. The circuit can be further simplified. For example, the gated $g8$ is obviously redundant after the addition of the rectification network.

2) Second approach: In FECR, since it is not limited to have only one candidate destination node for an error cancellation, the rectification condition in FECR is not as restrictive as that in the ECR approach. It may be advantageous to relax the AND(CMA) rectification scheme.

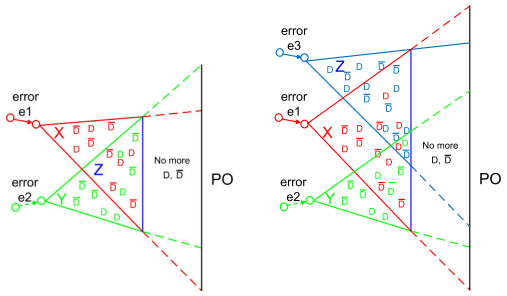
AND(semi-CMA) is a subset of AND(CMA). From the example in Fig. 5, the CMAs are $\{a = 0, b = 1\}$ in the ECR approach. The rectification network built upon the CMAs, and the corresponding single alternative wire, are valid only if none of the test vectors in $\{a = 0, b = 1\}$ can distinguish the good circuit and the rewired circuit. Unlike ECR, the test vectors in FECR are divided into three parts: *a*) $\{a = 0, b = 1, e = 1\}$ and *b*) $\{a = 0, b = 1, f = 1\}$ and *c*) the don't cares: $\{a = 0, b = 1, e = f = 0\}$. The rewired circuit is functionally equivalent to the original circuit only if none of the test vectors in part (a) and part (b) can distinguish between these two circuits.

Rectification networks for each of the candidate destination nodes can be therefore constructed using the corresponding AND(semi-CMA) instead. Fig. 4c depicts this approach and shows the rewired circuit after rectification network substitutions.

Based on the definitions in Definition 4, an E-frontier is a set of nodes such that the w_t_error can be cancelled before being propagation through these nodes. A single node in an E-frontier is essentially a testing dominator. Therefore, the circuit transformations found by ECR is a subset of FECR. Since RAR is a special case of ECR [7], both RAR and ECR are subsets of FECR.

Algorithm 1: Flow Graph Error Cancellation based Rewiring (FECR)

```
   input : circuit C, target wire w_t
   output: rewiring solution RS
1  begin
2      MA = stuckAtFaultTest(w_t);
3      efg = constructErrorFlowGraph(MA);
4      S = findEFrontierSets(efg);
5      foreach E-frontier S_i in S do
6          foreach E-frontier node S_ij in S_i do
7              semiFMA = findForcedSemiMAs(S_ij);
8              insert S_ij, semiFMA into CD_ij;
9              foreach candidate destination node C_d in CD_ij do
10                 if singleDestinationValid(w_t, S_i, C_d) then
11                     insert C_d into SD_ig;
12             insert SD_ij into SD_i;
13         D_i = validateConsistentDestinationGroupSet(SD_i);
14         foreach consistent destination group D_ij in D_i do
15             SRS = identifySourceNode(D_ij);
16             insert SRS into RS;
17     return RS;
18 end
```



(a) 1-to-1 Error Cancellation (b) Multi-Error Cancellation

Fig. 7: Structural View on Error Cancellation

The structural views on 1-to-1 and 1-to-k rewirings are shown in Fig. 7. As depicted in Fig. 7a, the error effects of e_1 and e_2 mutually cancel each other in region Z. No error effects can be observed at any primary output. As illustrated in Fig. 7b, the errors that are caused by target wire removal can be cancelled by adding multiple errors into the circuit.

Observation 3.3: A target wire in a Boolean network is removable by the error cancellation scheme proposed in this paper, as long as it is neither (i) connected to a primary output directly nor (ii) connected to a logic gate connecting to a primary output.

Due to space limit, here we simply give an intuitive explanation on what we can conclude from our studies and experiments: given that the wire is not connected to close to the primary outputs, we are always able to construct an error propagation graph. The mincut(s) from the graph can identify proper location(s) at which the rectification network, which is built using CMAs, can be added to cancel the error injected by the removal of the target wire.

Hence, we believe that most of circuit wires are removable, and their alternative wires can be identified by our proposed scheme. It should be noted that our observation *only* provides

TABLE I: Comparison between ECR and FECR

Benchmark	# TW	ECR				FECR					
		# r.TW	r.TW%	# AW	Time(s)	# r.TW	r.TW%	# AW *	Time(s)	Single	Pair
9sym-hdl	96	25	26.04	58	0.01	78	81.25	55	0.07	989	2962
pcler8	126	36	28.57	106	0.03	102	80.95	150	0.07	2017	0
f51m	190	94	49.47	441	0.04	188	98.95	445	0.96	20589	334995
comp	184	78	42.39	610	0.15	137	74.46	334	1.01	1835	8875
5xp1	178	90	50.56	452	0.04	176	98.88	460	0.79	21141	269695
b9_n2	166	78	46.99	240	0.03	157	94.58	306	0.16	2327	3097
my_adder	256	81	31.64	134	0.02	222	86.72	136	0.11	2669	2677
ttt2	274	120	43.80	521	0.05	261	95.26	548	0.27	18064	47508
term1	314	142	45.22	699	0.1	300	95.54	793	1.24	19399	179782
sao2-hdl	324	96	29.63	428	0.16	301	92.90	343	2.41	34558	392812
C432	320	131	40.94	545	0.14	292	91.25	732	0.3	7390	1841
C1908	706	162	22.95	291	0.21	685	97.03	302	0.69	6677	11156
C880	648	193	29.78	458	0.19	610	94.14	551	0.4	7067	2598
C1355	772	172	22.28	228	0.16	760	98.45	260	0.55	4506	8002
rot	942	398	42.25	1046	0.24	882	93.63	1146	0.79	13342	19948
x3	1120	341	30.45	1287	0.36	1089	97.23	1247	0.97	27766	57594
apex6	1188	379	31.90	2099	0.4	1158	97.47	2196	1.67	41408	112536
Total	7804	2616	**33.52**			7398	**94.80**			1.00	2.74
Average			36.17				92.27				

TW: the number of target wires in the circuit; # r.TW: the number of *removable* target wires; r.TW%: the percentage of removable target wires
AW (for ECR): the number of total alternative wires found; # AW* (for FECR): the number of all the single (1-to-1) alternative wires
Time(s): average CPU time which is measured in seconds for each target wire; Single/Pair: the number of solutions with single/pair destination

the sufficient, but not necessary, condition for a wire to be removable. The wires connected close to primary outputs may also be removable, despite the fact that its error propagation graph cannot be constructed.

IV. Experimental Results

Table I shows the result of the comparison between ECR and FECR. Our algorithm is implemented in C++ and tested with ISCAS and MCNC benchmarks which were first optimized by ABC [11]. Previously, the experiments with non-optimized circuits in [12] and [4] reported that around 35% and 60% of wires can have alternative wires found respectively. In our experiment, the percentage was found to be 34% by using ECR and impressively 95% by using FECR (the percentage is calculated using the total to account for the *wire* average but not *circuit* average), compared to 20% in [4]. The number of 1-to-1 alternative wires found by FECR matched that of ECR and in practice most of the rest target wires can be removed by 1-to-2 rewiring. For most benchmarks, the CPU time cost of FECR was times to an order of magnitude higher than that of ECR due to its much broader searching space. In practice, an upper bound can be imposed on the total number of alternative logics searched if less CPU time is demanded. It was found that the ratio of the number of solutions with pair destination to that with single destination is 2.74.

V. Conclusion

In this work, we have theoretically and experimentally shown that in a circuit, virtually every single wire is removable and its alternative wires can be found within a reasonable amount of time. We have proposed a Flow Graph Error Cancellation Rewiring (FECR) scheme which can untie the 1-to-1 limitation of most previously developed rewiring engines. The experimental results show that a (first known) near complete (95%) rewiring rate can be practically achieved even for optimized circuits. This significant progress makes rewiring a much more flexible logic synthesis technique that can provide today's various challenging EDA problems one extra last resort to explore.

References

[1] Y. L. Wu, C. C. Cheung, D. I. Cheng, and H. Fan, "Further Improve Circuit Partitioning Using GBAW Logic Perturbation Techniques," *IEEE Transactions on VLSI Systems*, vol. 11, pp. 451–460, June 2003.

[2] S.-C. Chang, L. Van Ginneken, and M. Marek-Sadowska, "Circuit Optimization by Rewiring," *Computers, IEEE Transactions on*, vol. 48, pp. 962–970, Sep 1999.

[3] K.-T. Cheng and L. Entrena, "Multi-level Logic Optimization by Redundancy Addition and Removal," in *Design Automation, 1993, with the European Event in ASIC Design. Proceedings. [4th] European Conference on*, pp. 373 –377, Feb. 1993.

[4] C.-C. Lin and C.-Y. Wang, "Rewiring using IRredundancy Removal and Addition," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pp. 324 –327, april 2009.

[5] J. Cong, J. Lin, and W. Long, "A New Enhanced SPFD Rewiring Algorithm," in *Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on*, pp. 672 – 678, Nov. 2002.

[6] C. Chang and M. Marek-Sadowska, "Theory of Wire Addition and Removal in Combinational Boolean Networks," *Microelectronic Engineering*, vol. 84, no. 2, pp. 229–243, 2007.

[7] X. Yang, T.-K. Lam, and Y.-L. Wu, "ECR: A Low Complexity Generalized Error Cancellation Rewiring Scheme," in *DAC '10: Proceedings of the 47th Design Automation Conference*, (New York, NY, USA), pp. 511–516, ACM, 2010.

[8] Y.-C. Chen and C.-Y. Wang, "Node Addition and Removal in the Presence of Don't Cares," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pp. 505 –510, june 2010.

[9] Z. Wu and S. Chang, "Multiple Wire Reconnections based on Implication Flow Graph," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 11, no. 4, pp. 939–952, 2006.

[10] Y.-C. Chen and C.-Y. Wang, "Fast Detection of Node Mergers using Logic Implications," in *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pp. 785 –788, Nov. 2009.

[11] Berkeley Logic Synthesis and Verification Group, "ABC: A System for Sequential Synthesis and Verification, release 70911."

[12] C. Chang and M. Marek-Sadowska, "Who are the alternative wires in your neighborhood?(alternative wires identification without search)," in *Proceedings of the 11th Great Lakes symposium on VLSI*, pp. 103–108, ACM, 2001.