Interpolation-Based Incremental ECO Synthesis for Multi-Error Logic Rectification*

Kai-Fu Tang†, Chi-An Wu†, Po-Kai Huang‡, and Chung-Yang (Ric) Huang†‡

†Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
‡Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan

ABSTRACT

To cope with last-minute design bugs and specification changes, engineering change order (ECO) is usually performed toward the end of the design process. This paper proposes an automatic ECO synthesis algorithm by interpolation. In particular, we tackle the problem by a series of partial rectifications. At each step, partial rectification can reduce the functional difference between an old implementation and a new specification. Our algorithm is especially effective for multiple error circuits. Experimental results show the proposed method is far superior to the most recent work and scales well on a set of large circuits.

Categories & Subject Descriptors:
B.6.3 [Logic Design]: Design Aids – Automatic synthesis

General Terms:
Algorithms, Design, Verification

Keywords
Logic rectification, engineering change order, satisfiability, interpolation

1. INTRODUCTION

In modern VLSI designs, late design changes are nearly inevitable to fix the design bug or to cope with the specification change. Due to the cost and time-to-market pressure, designers usually do not want to modify the register-transfer level (RTL) design and re-run the whole design flow from the beginning. Instead, they identify a patch circuit to rectify the gate-level design for the differences. This process is called *engineering change order* (ECO).

In this paper, the ECO synthesis problem is considered as follows. Given old and new circuits, where the *old* circuit is an original implementation which contains some errors, and the *new* circuit is a golden specification which defines the correct function, find a *patch* (or called *rectification function*) to fix errors in the old circuit and make the old circuit functionally equivalent to the new one. The objective of the ECO synthesis problem is to find a minimal patch such that the cost of the change can be minimized.

In the literature, there are a number of automated ECO synthesis algorithms. These automated methods can be classified into three categories. The first category is the *fault modeling approach*. This approach uses several common fault models to describe the differences between old and new circuits [1, 6, 19]. However, if the old and new circuits are synthesized from RTL, the old and new circuits usually have considerable differences. Using fault modeling approach is thus not affordable. The second category is called *diagnosis and resynthesis approach*. This approach first diagnoses the possible rectifiable locations and then synthesizes a new function to fix the rectifiable location [5, 9, 11, 15, 17, 18, 20, 21]. Nevertheless, previous approaches mainly rely on single-fix rectification techniques or binary decision diagrams (BDDs). The major disadvantage is that the approaches usually create large patches or suffer from the memory explosion problem. The third category is called *structural approach*. This method uses verification techniques to perform a structural comparison between the old and new circuits [3, 4, 10, 11, 12]. Although the structural approach is efficient, this method is limited by the similarity between the old and new circuits. There are also other ECO works, including ECO for FPGAs [14, 16, 17], and ECO for the transistor level [13]. The scalability of most existing methods is still a problem.

This paper proposes a method to synthesize multiple *strong partial-fix rectification functions* for the ECO synthesis problem by *interpolation*. Unlike prior works [17, 20] which can only synthesize a single-fix rectification function, our method gradually rectifies an old circuit by multiple partial-fix functions and guarantees that the functional differences between the old and new circuits become smaller after fixing. While there are several differences between the old and new circuits, single-fix rectification cannot properly solve the problem. Contrarily, our approach is particularly useful for these ECO cases. In addition, our method does not rely on any fault modeling or structural similarity.

The essential features of our method include: 1) It uses a SAT solver to identify equivalent signals between old and new circuits. 2) It reuses equivalent signals for effective search space reduction. 3) It adopts a novel counter-example-guided scheme for *strong partial-fix signal* selection. 4) It generates strong partial-fix rectification functions by interpolation. 5) It includes two flexible methods to handle multiple-output functions. Experimental results demonstrate that our interpolation-based method is effective to the ECO synthesis problem. Automatic ECO synthesis with hundred thousands of gates can be realized efficiently and the quality of the results are significantly superior to previous approaches. Compared with the most recent work in [20], our approach outperforms it and can be much more scalable.

The remaining part of this paper is organized as follows. Section 2 gives some essential backgrounds and Section 3 presents the main algorithms on interpolating strong partial-

*This work was partially supported by the National Science Council of Taiwan ROC under Grant No. NSC-99-2221-E-002-211-MY3.

DAC 2011, June 5-10, 2011, San Diego, California, USA.

fix rectification functions. Experimental results are demonstrated in Section 4. Finally, Section 5 concludes our paper.

2. PRELIMINARIES

In this paper, a set of Boolean variables $X = \{x_1, \ldots, x_m\}$ is used to denote the set of primary inputs (PIs). The functions of primary outputs in the old and new circuits are denoted by $F(X) = \langle f_1(X), \ldots, f_n(X)\rangle$ and $G(X) = \langle g_1(X), \ldots, g_n(X)\rangle$, respectively.

Let $f_i(X, r)$ be the function expressed in terms of X and an internal signal r. For a signal r and a function $f_i(X)$, the **care-set** is defined as

$$care_i^r(X) = f_i(X, r = t(X)) \oplus f_i(X, r = \neg t(X)),$$

where $t(X)$ is the original function at signal r. The care-set characterizes the set of input assignments for which any change at signal r can be observed at output function f_i. If F is a single-output function, the index i can be omitted, and the care-set is denoted as $care^r(X)$.

For an old function f_i and a new function g_i, the **diff-set** is defined as

$$diff_i(X) = f_i(X) \oplus g_i(X).$$

The diff-set characterizes the set of input assignments for which the functions f_i and g_i have opposite values. If F and G are single-output functions, the index i can be omitted, and the diff-set is denoted as $diff(X)$.

An input assignment $X^* \in \{0,1\}^m$ is an **error minterm** with respect to an output i if $diff_i(X^*) = 1$. On the contrary, an input assignment $X^* \in \{0,1\}^m$ is a **correct minterm** with respect to an output i if $diff_i(X^*) = 0$.

2.1 Rectification Functions

Definition 1. *Given old and new functions $F(X)$ and $G(X)$, we say that $F(X)$ is* **single-rectifiable** *with respect to $G(X)$ if there exists a Boolean function s, called the* **single-fix rectification function***, such that $F(X, r = s(X)) \equiv G(X)$, where variable r is an internal signal in the function F.*

The necessary and sufficient condition of the existence of the single-fix rectification function s is given as follows.

Proposition 1. *[15] For an old function $F = \langle f_1, \ldots, f_n\rangle$ and a new function $G = \langle g_1, \ldots, g_n\rangle$, there exists a single-fix rectification function s at signal r if and only if*

$$(\bigvee\nolimits_i f_i(X, r = 0) \oplus g_i(X)) \wedge (\bigvee\nolimits_i f_i(X, r = 1) \oplus g_i(X)) \quad (1)$$

is unsatisfiable. In this case, the characteristic functions of the on-set and off-set of s are

$$s_{on}(X) = \bigvee\nolimits_i f_i(X, r = 0) \oplus g_i(X), \text{ and}$$
$$s_{off}(X) = \bigvee\nolimits_i f_i(X, r = 1) \oplus g_i(X).$$

By Proposition 1, we can not only determine the existence of a single-fix rectification function, but also derive a feasible one.

Definition 2. *Given old and new functions $F(X)$ and $G(X)$, we say that $F(X)$ is* **multi-rectifiable** *with respect to $G(X)$ if there exists a set of Boolean functions $\{p_1, \ldots, p_k\}$, called the* **multi-fix rectification functions***, such that $F(X, r_1 = p_1(X), \ldots, r_k = p_k(X)) \equiv G(X)$, where variables r_i's are internal signals in the function F.*

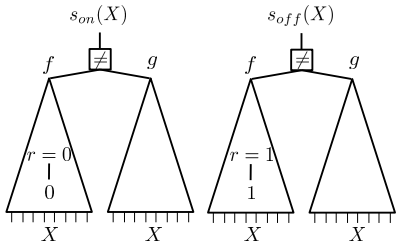

Figure 1: Boolean networks for single-fix rectification function.

Note that multi-rectification trivially holds if r_i's are the output signals of function F, and p_i equals g_i. The corresponding rectification functions p_i's are called *trivial*. This paper is concerned about non-trivial rectification functions. Moreover, we focus on finding restricted ones, called *partial-fix rectification functions*, which will be introduced in Section 3.

2.2 Interpolation and Single-Fix Rectification Functions

To introduce interpolation techniques behind our method, we review the following theorem.

Theorem 1 (Craig Interpolation Theorem). *[7] Given two Boolean formulae A and B, with $A \wedge B$ unsatisfiable, there exists a Boolean formula I such that 1) $A \Rightarrow I$, 2) $I \wedge B$ is unsatisfiable, and 3) I refers only to the common variables of A and B.*

The Boolean formula I is called the *interpolant* of A and B. The construction of interpolant can be derived from a refutation proof in linear time. The capability of modern SAT solvers [8] can be extended to support the interpolant construction.

Prior works [17, 20] use Proposition 1 and Theorem 1 to produce a single-fix rectification function. We restate the result in Theorem 2.

Theorem 2. *For unsatisfiable Formula (1), with partition $A = s_{on}$ and $B = s_{off}$, the resultant interpolant I derived from a refutation proof implements a desired single-fix rectification function s.*

To illustrate the construction, consider the networks shown in Figure 1. The networks in Figure 1 represent the formulae stated in Proposition 1. If the instance is unsatisfiable, we can partition the clauses of left-hand and right-hand side networks as formulae A and B, respectively. Then the resultant interpolant is a single-fix rectification function for signal r.

3. INTERPOLATION-BASED INCREMENTAL ECO SYNTHESIS ALGORITHM

3.1 Incremental ECO Synthesis

In general, due to multiple errors in the old circuit, it may not be single-rectifiable. To overcome this limitation, we exploit partial-fix rectification [11] and then propose a solution by interpolation.

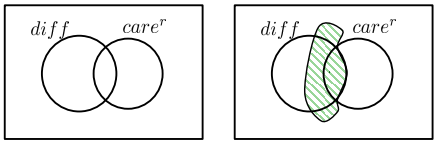

Figure 2: (a) Relation of $care^r$ and $diff$. (b) An over-approximation of $care^r \wedge diff$ which does not touch $care^r \wedge \neg diff$.

DEFINITION 3. *[11] Given old and new functions $F(X)$ and $G(X)$, we say that a signal r in $F(X)$ is a* **partial-fix signal** *if there exists a Boolean function p, called the* **partial-fix rectification function**, *such that 1) no error minterm is newly created, and 2) some error minterms for at least one output are rectified by p.*

With the concept of partial-fix rectification, the ECO synthesis problem can be solved by an incremental process. The process iteratively identifies a partial-fix signal and attempts to rectify some error minterms. At each step, it guarantees that some error minterms are fixed and no error minterm is newly created. In this fashion, the set of error minterms can be monotonically decreased. In the following, we discuss which error minterms can be fixed at a signal r from the functional standpoint.

With respect to a signal r, we first consider the relation between care-set and diff-set as shown in Figure 2(a) and discuss the meaning of three sets, $care^r \wedge diff$, $care^r \wedge \neg diff$, and $\neg care^r$, as follows.

For the ease of discussion, we assume F and G are single-output functions. Firstly, for a minterm in $care^r \wedge diff$, since this minterm belongs to the care-set, it means that if the value of signal r is inverted under this minterm, the output value will be changed. Moreover, this minterm belongs to the diff-set. Therefore, after changing the value of signal r under this minterm, the output values of old and new circuits will be equivalent under this minterm. Secondly, for a minterm in $care^r \wedge \neg diff$, since the minterm does not belong to the diff-set, output values of old and new circuits are the same under this minterm. Furthermore, the minterm belongs to the care-set. Therefore, the value of signal r cannot be inverted under this minterm. Otherwise, the output values of F and G will become nonequivalent, and new error minterms will be created. Thirdly, for a minterm in $\neg care^r$, since the minterm does not belong to the care-set, inverting the value of signal r does not affect the output of F. Hence, for a minterm in $\neg care^r$, inverting the value of signal r or not does not matter.

Based on the above discussion, for a signal r, minterms in $care^r \wedge diff$ are **correctable minterms** while minterms in $care^r \wedge \neg diff$ are the set of minterms for which we want to retain their values. Other minterms can be considered as don't-cares since they cannot affect primary outputs. From this point of view, *strong partial-fix rectification functions* can be defined as follows.

DEFINITION 4. *[11] Given old and new functions $F(X)$ and $G(X)$, we say that a signal r in $F(X)$ is a* **strong partial-fix signal** *if there exists a Boolean function p, called the* **strong partial-fix rectification function**, *such that 1) no error minterm is newly created, and 2) all correctable minterms for at least one output are rectified by p.*

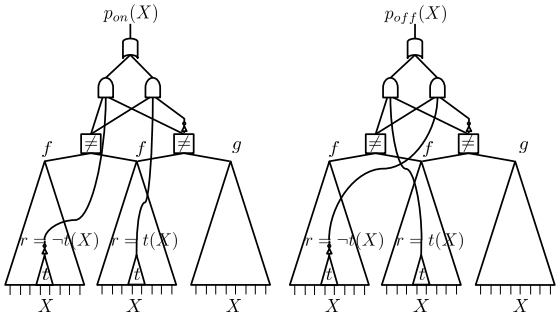

Figure 3: Boolean networks representing $p_{on}(X)$ and $p_{off}(X)$.

In Figure 2(b), we demonstrate a region of minterms which the strong partial-fix rectification signal can be allowed to rectify. The constraints of this region are that it has to include the correctable minterms and disjoint from $care^r \wedge \neg diff$. Other minterms are flexible to be covered or not.

3.2 Construction of Strong Partial-Fix Rectification Function by Interpolation

We first assume that F is a single-output function, i.e. $F = \langle f \rangle$. To derive a strong partial-fix rectification function p at a signal r, we consider the on-set and off-set of function p as follows. If function t, the original function of signal r, is evaluated to 0 under a minterm which belongs to $care^r \wedge diff$, then function p has to be evaluated to 1 under this minterm. Therefore, this minterm should belong to the on-set of function p. On the other hand, if function t is evaluated to 1 under a minterm which belongs to $care^r \wedge \neg diff$, then function p has to be evaluated to the same value as t. Therefore, this minterm should belong to the on-set of function p, too. Similarly, the off-set of function p can be derived in the same manner. Formally, we characterize the on-set p_{on} and off-set p_{off} of function p as follows.

$$\begin{aligned} p_{on}(X) &= (care^r(X) \wedge diff(X) \wedge \neg t(X)) \vee \\ &\quad (care^r(X) \wedge \neg diff(X) \wedge t(X)) \\ p_{off}(X) &= (care^r(X) \wedge diff(X) \wedge t(X)) \vee \\ &\quad (care^r(X) \wedge \neg diff(X) \wedge \neg t(X)) \end{aligned}$$

PROPOSITION 2. *[11] For single-output functions $F = \langle f \rangle$ and $G = \langle g \rangle$, there exists a strong partial-fix rectification function p at signal r if*

$$p_{on}(X) \wedge p_{off}(X) \tag{2}$$

is unsatisfiable. In this case, the signal r is a strong partial-fix signal.

As shown in Figure 3, functions $p_{on}(X)$ and $p_{off}(X)$ can be represented by Boolean networks. Note that we use and-inverter graphs (AIGs) as our underlying representation. Since most parts of $f(X, r = t(X))$ and $f(X, r = \neg t(X))$ are the same, these identical gates can be structurally hashed, resulting in effective circuit size reduction.

By translating circuits to conjunctive normal form (CNF), if formula $p_{on}(X) \wedge p_{off}(X)$ is unsatisfiable, a rectification function can be constructed by interpolation. To achieve this, we partition the clauses of left-hand side and right-hand

side networks in Figure 3 as formulae A and B, respectively. Thus we have the following theorem.

THEOREM 3. *For an unsatisfiable formula $p_{on}(X) \wedge p_{off}(X)$, with partition $A = p_{on}$ and $B = p_{off}$, the resultant interpolant I derived from a refutation proof implements a desired strong partial-fix rectification function p.*

PROOF. This can be easily deduced from the discussion above. Due to the space limitation, we omit the proof here. □

Since the common variables of A and B are primary inputs X, the resultant interpolant will be supported by variables X. However, for ECO purposes, we would like to reuse the old circuit as much as possible. In this case, we propose a method to generate a rectification function from internal signals.

Let $U = \{u_1, \ldots, u_k\}$ be a set of internal functions in F and $Y = \{y_1, \ldots, y_k\}$ be their corresponding internal variables. Let

$$p_{on}(Y) = \exists X.(\bigwedge\nolimits_i (y_i = u_i(X)) \wedge p_{on}(X))$$

$$p_{off}(Y) = \exists X.(\bigwedge\nolimits_i (y_i = u_i(X)) \wedge p_{off}(X))$$

Note that in the above formula, we quantifies out the primary input variables X and the resulting supports will be Y. In other words, we attempt to utilize Y as a set of supports for p.

To compute the function p from internal signals efficiently, we do not explicitly quantify out variables X. Instead, consider the Boolean network shown in Figure 4. We assert internal variables $y_i = y_i'$, and then convert the network to CNF. If the resulting SAT problem is unsatisfiable, it means that two sets $p_{on}(Y)$ and $p_{off}(Y)$ are disjoint and thus an interpolant, a strong partial-fix rectification function supported by internal variables Y, can be constructed.

Formally, we have the following corollary.

COROLLARY 1. *For single-output functions $F = \langle f \rangle$ and $G = \langle g \rangle$, there exists a strong partial-fix rectification function p at signal r if*

$$(\bigwedge_i (y_i = u_i(X)) \wedge p_{on}(X)) \wedge (\bigwedge\nolimits_i (y_i' = u_i(X')) \wedge p_{off}(X')) \wedge (\bigwedge\nolimits_i (y_i = y_i'))$$

is unsatisfiable. In this case, with partition

$$A = (\bigwedge\nolimits_i (y_i = u_i(X)) \wedge p_{on}(X)), \text{and}$$

$$B = (\bigwedge\nolimits_i (y_i' = u_i(X')) \wedge p_{off}(X')) \wedge (\bigwedge\nolimits_i (y_i = y_i')),$$

the resultant interpolant I derived from a refutation proof implements a desired strong partial-fix rectification function p which is supported by internal variables Y.

REMARK 1. *For single-output functions F and G, if rectification function p is constructed from X, Formula (2) is necessarily unsatisfiable, i.e., $p_{on}(X)$ and $p_{off}(X)$ are disjoint. However, in the case where the function p is constructed from internal functions Y, their on-set and off-set is not necessarily disjoint. This assures that we can always find a strong partial-fix rectification function for single-output functions F and G, and yet we try to optimize it by utilizing certain internal functions.*

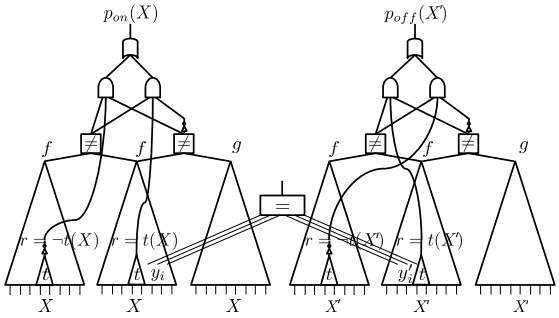

Figure 4: Boolean networks utilizing internal signals.

3.3 Extension to Multiple-Output Functions

We now turn attention to seek a rectification function for multiple-output functions, i.e. $F = \langle f_1, \ldots, f_n \rangle$ and $G = \langle g_1, \ldots, g_n \rangle$. In essence, our intention is to produce strong partial-fix rectification functions by generalizing the previous concept. In the following, we study two methods with different ways of fixing multiple-output functions. One is through *all outputs*; the other is through an *output subset*.

3.3.1 Rectification for All Outputs

For a multiple-output function, all-output rectification attempts to fix a signal such that all primary outputs are considered. That is, for a signal r, all correctable minterms with respect to *every primary output* need to be fixed. At the same time, no error minterm with respect to *every primary output* is newly created. Therefore, similar to the case of single-output functions, the on-set p_{on}^{all} and off-set p_{off}^{all} of a strong partial-fix rectification function p^{all} with respect to all outputs can be derived.

$$\begin{aligned} p_{on}^{all}(X) &= \bigvee\nolimits_i ((care_i^r(X) \wedge diff_i(X) \wedge \neg t(X)) \vee \\ &\quad (care_i^r(X) \wedge \neg diff_i(X) \wedge t(X))) \\ p_{off}^{all}(X) &= \bigvee\nolimits_i ((care_i^r(X) \wedge diff_i(X) \wedge t(X)) \vee \\ &\quad (care_i^r(X) \wedge \neg diff_i(X) \wedge \neg t(X))) \end{aligned}$$

Similar to the derivation in Theorem 3 and Corollary 1, we can utilize internal variables $Y = \{y_1, \ldots, y_k\}$ as the support set of a strong partial-fix rectification function. Thus we have the following corollary.

COROLLARY 2. *For multiple-output functions $F = \langle f_1, \ldots, f_n \rangle$ and $G = \langle g_1, \ldots, g_n \rangle$, there exists a strong partial-fix rectification function p^{all} with respect to all outputs at signal r if*

$$(\bigwedge_i (y_i = u_i(X)) \wedge p_{on}^{all}(X)) \wedge (\bigwedge\nolimits_i (y_i' = u_i(X')) \wedge p_{off}^{all}(X')) \wedge (\bigwedge\nolimits_i (y_i = y_i'))$$

is unsatisfiable. In this case, with partition

$$A = (\bigwedge\nolimits_i (y_i = u_i(X)) \wedge p_{on}^{all}(X)), \text{and}$$

$$B = (\bigwedge\nolimits_i (y_i' = u_i(X')) \wedge p_{off}^{all}(X')) \wedge (\bigwedge\nolimits_i (y_i = y_i')),$$

the resultant interpolant I derived from a refutation proof implements a desired strong partial-fix rectification function p^{all} which is supported by internal variables Y.

3.3.2 Rectification for an Output Subset

Unlike all-output rectification, output-subset rectification only fixes the correctable minterms of a specific output subset. In this manner, the sufficient condition in Corollary 2 can be relaxed, and thus the chance of finding a strong partial-fix signal becomes larger. Moreover, since output-subset rectification is more target-oriented, it can fix a specific output subset more effectively. Therefore, for a specific output subset $O' \subseteq O$, where $O = \{1, \ldots, n\}$ is the primary output indexes, the on-set p_{on}^{sub} and off-set p_{off}^{sub} of a strong partial-fix rectification function p^{sub} with respect to an output subset O' is shown as follows.

$$\begin{aligned} p_{on}^{sub}(X) &= \bigvee_{i \in O'} (care_i^r(X) \wedge diff_i(X) \wedge \neg t(X) \vee \\ &\quad \bigvee_{j \in O} (care_j^r(X) \wedge \neg diff_j(X) \wedge t(X))) \\ p_{off}^{sub}(X) &= \bigvee_{i \in O'} ((care_i^r(X) \wedge diff_i(X) \wedge t(X)) \vee \\ &\quad \bigvee_{j \in O} (care_j^r(X) \wedge \neg diff_j(X) \wedge \neg t(X))) \end{aligned}$$

Similarly, we state the following corollary.

COROLLARY 3. *For multiple-output functions $F = \langle f_1, \ldots, f_n \rangle$ and $G = \langle g_1, \ldots, g_n \rangle$, there exists a strong partial-fix rectification function p^{sub} with respect to an output subset O' at signal r if*

$$(\bigwedge_i (y_i = u_i(X)) \wedge p_{on}^{sub}(X)) \wedge (\bigwedge_i (y_i' = u_i(X')) \wedge p_{off}^{sub}(X')) \wedge (\bigwedge_i (y_i = y_i'))$$

is unsatisfiable. In this case, with partition

$$A = (\bigwedge_i (y_i = u_i(X)) \wedge p_{on}^{sub}(X)), \text{and}$$

$$B = (\bigwedge_i (y_i' = u_i(X')) \wedge p_{off}^{sub}(X')) \wedge (\bigwedge_i (y_i = y_i')),$$

the resultant interpolant I derived from a refutation proof implements a desired strong partial-fix rectification function p^{sub} which is supported by internal variables Y.

3.4 Overall Flow of Our Algorithm

In this section, we introduce the main flow of our algorithm. Given old and new circuits, our flow iteratively constructs strong partial-fix rectification functions to rectify the old circuit. In the end, output functions of old and new circuits are equivalent and the process terminates. As shown in Figure 5, our flow includes equivalent signal identification, counter-example-guided strong partial-fix signal selection, and strong partial-fix rectification function generation.

To identify equivalent signals, *SAT sweeping* [22] is performed on old and new circuits. SAT sweeping consists of simulation and SAT solving. First, random simulation vectors are assigned to the inputs of old and new circuits. If two signals have same values under all random vectors, then we will use the SAT solver to prove the equivalence of these two signals. In this manner, equivalent signals between old and new circuits can be efficiently identified.

In trivial cases that old and new circuits are equivalent, the process terminates. Otherwise, some pairs of primary outputs are nonequivalent and SAT solver will return a counter-example, denoted by cex, which makes $F(cex) \neq G(cex)$. This counter-example can be served as a guidance for strong partial-fix signal selection. We choose a signal r as a candidate of strong partial-fix signal if the following formula

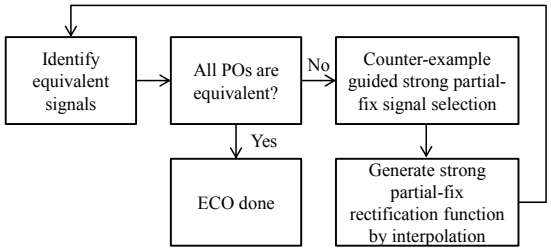

Figure 5: The flow of our algorithm.

holds:

$$\bigvee_i (care_i^r(cex) \wedge diff_i(cex)) \wedge \bigwedge_i (\neg care_i^r(cex) \vee diff_i(cex)) = 1$$

Intuitively, the above formula asserts that under the input assignment cex, if the value of signal r is inverted, at least one incorrect output value can be fixed while all correct output values retain the same. After finding a signal that satisfies the above condition, the flow proceeds to generate a strong partial-fix rectification function by interpolation. The strong partial-fix rectification function can rectify all correctable minterms and create more equivalent signals in its fan-out cone. Therefore, the procedure goes to the first step and iterates.

4. EXPERIMENTAL RESULTS

Our algorithm was implemented in C language in ABC [2] using MiniSAT [8] as the SAT solver. We conducted experiments on a Linux machine with Intel Xeon 2GHz CPU and 16GB memory.

The testcases were chosen from circuits in ISCAS89 and ITC99 benchmarks. Sequential circuits were converted to combinational circuits by cutting off sequential elements and treating their inputs and outputs as POs and PIs. The old and new circuits in the benchmark suites were designed for the ECO synthesis problem in the following ways. The new circuit was changed from the old one by rewiring and cube modifications. To perform the modifications, we implement a program to reconnect wires and change cubes from the functions of selected gates. Then both circuits were minimized by ABC [2] with command `resyn2`. After that, the old and new circuits were both structurally and functionally different.

We compare our results with [20], a recent publication in ICCAD 2010. Table 1 summarizes our experimental results. Column 2 shows the types of modifications on the old circuits. Columns 3 and 4 show the number of nodes in the old and new circuits, respectively. From Columns 5 to 7 are the results of the ICCAD 2010 method, which includes the patch sizes, runtime, and the memory usage. Columns 8 to 10 present our results.

Among all of these 20 test-cases, our method can consistently produce high quality results. Compared to the ICCAD 2010 method, the runtime of our algorithm is significantly faster. For the cases of `s3384`, `s35932`, `b14`, and `b21`, their tool were aborted while our algorithm can complete the tasks within three seconds and generate reasonably good patch (sizes are 106, 2, 1, 1, respectively). Moreover, for the cases `b19` and `b20`, their runtime data are unavailable due to time-out at the limit of 16 hours. In contrast, our

Table 1: Experimental results on ISCAS89 and ITC99 benchmarks.

Circuits	Error Type	#Nodes		ICCAD 2010 [20]			Ours		
		Old	New	Patch Size	Time (s)	Mem (MB)	Patch Size	Time (s)	Mem (MB)
s3271	Rewire	2662	2214	46	6.72	31.64	5	0.02	26.82
s3384	Cube	2761	2307	–	–	Abort	106	0.59	27.42
s6669	Rewire	5669	4781	99	14.49	35.17	2	0.13	27.15
s9234.1	Cube	4227	3187	146	14.83	84.12	2	0.43	26.6
s13207	Cube	6234	5138	69	90.65	151.9	3	4.69	31.38
s13207.1	Rewire	6234	5112	73	80.79	154.5	8	5.66	30
s15850	Cube	8084	6505	116	90.53	201.8	3	4.17	29.54
s15850.1	Cube	8084	6516	118	91.02	201.1	2	2.7	29.53
s35932	Cube	29176	21305	–	–	Abort	2	3.4	32.05
s38584	Cube	27237	22267	209	1121	421.9	8	213.29	32.05
s38584.1	Cube	27237	22226	180	997.5	397.3	4	154.31	32.05
s38417	Cube	22252	19266	20	278.2	198.7	2	4.59	32.05
b14	Cube	13302	13303	–	–	Abort	1	0.14	31.05
b15	Cube	14782	16595	8843	794.5	660.7	2	2.11	40.2
b17	Cube	54428	54427	624	1018	219.7	3	6.08	54.67
b18	Cube	165829	165835	142	1654	483.6	13	71.56	105.68
b19	Cube	333969	333974	–	T.O. (16 hr)	–	39	3193	190.68
b20	Cube	27432	27437	–	T.O. (16 hr)	–	19	0.27	39.23
b21	Cube	28500	28500	–	–	Abort	1	0.45	38.65
b22	Cube	41390	41398	1	507.6	97.18	1	0.44	45.89

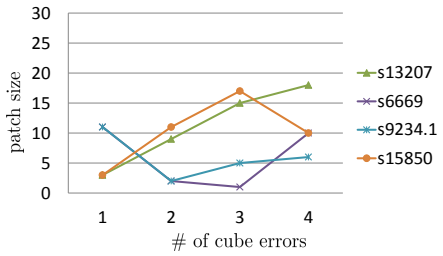

Figure 6: Patch size for different numbers of cube errors.

algorithm is scalable to these two large cases.

Furthermore, another set of experiments were designed to demonstrate the usefulness of our approach. Figure 6 shows experimental results for circuits `s6668`, `s9234.1`, `s13207`, and `s15850` with injected errors. The error types are cube modifications ranging from one to four cubes. From the figure, we can see that our method is scalable with respect to the number of errors. It also suggests that our method is inherently superior to the previous work for solving the multi-error ECO synthesis problem.

5. CONCLUSIONS

We have presented a novel ECO synthesis algorithm by interpolation. Our incremental ECO synthesis algorithm iteratively find strong partial-fix rectification functions to fix old designs, and the rectification functions can be maintained in a small size. To search the strong partial-fix signal effectively, SAT sweeping is performed for effective search space reduction. Experimental results demonstrate that our algorithm is very efficient and can produce high quality rectification functions for large circuits.

6. REFERENCES

[1] M. S. Abadir, J. Ferguson, and T. E. Kirkland. Logic design verification via test generation. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 7(1):138–148, 1988.

[2] Berkeley Logic Synthesis and Verification Group. *ABC: A system for sequential synthesis and verification.*

[3] D. Brand. Verification of large synthesized designs. In *Proc. ICCAD*, pages 534–537, 1993.

[4] D. Brand, A. D. Drumm, S. Kundu, and P. Narain. Incremental synthesis. In *Proc. ICCAD*, pages 14–18, 1994.

[5] K.-H. Chang, I. L. Markov, and V. Bertacco. Fixing design errors with counterexamples and resynthesis. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(1):184–188, 2008.

[6] P.-Y. Chung and I. N. Hajj. ACCORD: Automatic catching and correction of logic design errors in combinational circuits. In *Proc. ITC*, pages 742–751, 1992.

[7] W. Craig. Linear reasoning: A new form of the Herbrand-Gentzen theorem. *J. Symb. Log.*, 22(3):250–268, 1957.

[8] N. Eén and N. Sörensson. An extensible SAT-solver. In *Proc. SAT*, pages 502–518, 2003.

[9] M. Fujita, Y. Tamiya, Y. Kukimoto, and K.-C. Chen. Application of Boolean unification to combinational logic synthesis. In *Proc. ICCAD*, pages 510–513, 1991.

[10] S.-Y. Huang, K.-C. Chen, and K.-T. Cheng. Error correction based on verification techniques. In *Proc. DAC*, pages 258–261, 1996.

[11] S.-Y. Huang, K.-C. Chen, and K.-T. Cheng. AutoFix: A hybrid tool for automatic logic rectification. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 18(9):1376–1384, 1999.

[12] S. Krishnaswamy, H. Ren, N. Modi, and R. Puri. DeltaSyn: An efficient logic difference optimizer for ECO synthesis. In *Proc. ICCAD*, pages 789–796, 2009.

[13] A. Kuehlmann, D. I. Cheng, A. Srinivasan, and D. P. LaPotin. Error diagnosis for transistor-level verification. In *Proc. DAC*, pages 218–224, 1994.

[14] Y. Kukimoto and M. Fujita. Rectification method for lookup-table type FPGA's. In *Proc. ICCAD*, pages 54–61, 1992.

[15] C.-C. Lin, K.-C. Chen, and M. Marek-Sadowska. Logic synthesis for engineering change. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 18(3):282–292, 1999.

[16] C.-H. Lin, Y.-C. Huang, S.-C. Chang, and W.-B. Jone. Design and design automation of rectification logic for engineering change. In *Proc. ASP-DAC*, pages 1006–1009, 2005.

[17] A. C. Ling, S. D. Brown, J. Zhu, and S. Safarpour. Towards automated ECOs in FPGAs. In *Proc. FPGA*, pages 3–12, 2009.

[18] J. C. Madre, O. Coudert, and J. P. Billon. Automating the diagnosis and the rectification of design errors with PRIAM. In *Proc. ICCAD*, pages 30–33, 1989.

[19] A. G. Veneris and I. N. Hajj. Design error diagnosis and correction via test vector simulation. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 18(12):1803–1816, 1999.

[20] B.-H. Wu, C.-J. Yang, C.-Y. Huang, and J.-H. R. Jiang. A robust functional ECO engine by SAT proof minimization and interpolation techniques. In *Proc. ICCAD*, pages 729–734, 2010.

[21] Y.-S. Yang, S. Sinha, A. G. Veneris, and R. K. Brayton. Automating logic rectification by approximate SPFDs. In *Proc. ASP-DAC*, pages 402–407, 2007.

[22] Q. Zhu, N. Kitchen, A. Kuehlmann, and A. L. Sangiovanni-Vincentelli. SAT sweeping with local observability don't-cares. In *Proc. DAC*, pages 229–234, 2006.