Multi-Patch Generation for Multi-Error Logic Rectification by Interpolation with Cofactor Reduction[*]

Kai-Fu Tang[†], Po-Kai Huang[‡], Chun-Nan Chou[†], 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

For a design with multiple functional errors, multiple patches are usually needed to correct the design. Previous works on logic rectification are limited to either single-fix or partial-fix rectifications. In other words, only one or part of the erroneous behaviors can be fixed in one iteration. As a result, it may lead to unnecessarily large patches or even failure in rectification. In this paper, we propose a multi-patch generation technique by interpolation with cofactor reduction. In particular, our method considers multiple errors in the design simultaneously and generates multiple patches to fix these errors. Experimental results show that the proposed method is effective on a set of large circuits, including the circuits synthesized from industrial register-transfer level (RTL) designs.

1. INTRODUCTION

As VLSI technology continues to advance, late design modifications are nearly unavoidable in order to deal with the specification changes or to repair the design bugs. For the reasons of time-to-market and cost saving, it is unlikely that the designers would modify the design in register-transfer level (RTL) and re-run the flow from scratch. In contrast, designers would like to produce patch circuits in gate-level such that the design differences can be rectified with minimal netlist changes. This procedure is called *logic rectification.*

In this paper, we consider the multi-error logic rectification problem as follows. Given old and new circuits, where the *old* circuit is an erroneous gate-level implementation containing *multiple* errors, and the *new* circuit is a golden specification, we would like to generate *multiple* patches to fix the errors in the old circuit and make it functionally equivalent to the new one. The objective of the logic rectification problem is to minimize the total number of gates in the patches. Note that in the sequel, we will use the term *rectification function* instead of patch when we want to emphasize its functional counterpart more than its component part.

In the literature, there are a number of logic rectification methods. These methods can be classified into three categories: fault modeling, resynthesis, and structure-based methods. Fault modeling methods use common fault models to characterize the inconsistent behaviors between the old and new circuits [1, 6, 24]; however, when the old and new circuits are synthesized from RTL designs, these two circuits can be drastically changed due to logic synthesis. Therefore, fault modeling methods are not adequate to describe the differences. Resynthesis methods utilize symbolic techniques to fix erroneous designs [5, 9, 11, 14, 15, 16, 18, 27]. These methods are effective, but often with expensive runtime cost. Structure-based methods identify the differences of the old and new circuits by comparing their circuit structures [3, 4, 10, 12, 21]. These methods are efficient, but limited by the circuit similarity between the old and new circuits.

More recently, *interpolation* [7] is used to solve the logic rectification problem. First, prior works [17, 26] utilize satisfiability (SAT) solver and interpolation to synthesize *single-fix rectification function.* This approach, however, is only adequate to the rectification problem containing single error. Handling multi-error cases relies on heuristic techniques which may not be effective. Second, the work in [23] uses interpolation to generate *partial-fix rectification function.* This method performs multiple passes of partial rectifications to fix multiple errors in the design. Although this approach can handle the problem efficiently, it does not consider the correlations between multiple errors in the design. That is, each error in the design is considered *independently.* Therefore, this approach cannot effectively handle certain difficult cases.

This paper proposes a new algorithm to produce *multi-fix rectification functions* by interpolation. In contrast to prior work [23], our method considers multiple design errors *simultaneously.* When there are several differences between the old and new circuits, our approach is especially advantageous. Moreover, our method does not depend on any fault model or structural similarity. The main contributions of this paper include: 1) It derives the necessary and sufficient conditions for the existence of the multi-fix rectification functions. 2) It solves the multiple error rectifiability problem by SAT solver with cofactor reduction. 3) It generates multi-fix rectification functions by interpolation with cofactor reduction. Experimental results demonstrate that our method is effective in multi-error logic rectification. Compared to [23], the quality of our results substantially outperforms [23], and total patch size improvement can be up to 89%. Moreover, our algorithm is applicable to the circuits synthesized from industrial RTL designs.

The rest of this paper is organized as follows. Section 2 introduces terminologies and backgrounds. The proposed algorithm is presented in Section 3, and the experimental results are demonstrated in Section 4. Finally, Section 5 concludes our paper.

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

2. PRELIMINARIES

In this paper, we use the notation $\vec{v} = (v_1, \ldots, v_m)$ to denote a vector. The cardinality of a vector $\vec{v}$ is denoted by $|\vec{v}|$. For simplicity, we define two vector operators as follows. The **interval operator** $[\cdot, \cdot]$ of a vector $\vec{v}$ is defined by

$$\vec{v}_{[i,j]} = (v_i, v_{i+1}, \ldots, v_j).$$

The **concatenation operator** $[\cdot \parallel \cdot]$ of two vectors $\vec{u}$ and $\vec{v}$ is defined by

$$[\vec{u} \parallel \vec{v}] = (u_1, \ldots, u_{|\vec{u}|}, v_1, \ldots, v_{|\vec{v}|}).$$

For the logic rectification problem, we use multiple-output functions $F(\vec{x}) = \langle f_1(\vec{x}), \ldots, f_n(\vec{x})\rangle$ and $G(\vec{x}) = \langle g_1(\vec{x}), \ldots, g_n(\vec{x})\rangle$ to denote the functions of the old and new circuits, respectively. Moreover, we use $\vec{r}$ to denote a vector of signals with $|\vec{r}| = m$.

Let $F(\vec{x}, \vec{r})$ be the function expressed in terms of inputs $\vec{x}$ and internal signals $\vec{r}$. Given functions F and G with internal signals $\vec{r}$, we define the **rectification miter** as

$$RM(\vec{x}, \vec{r}) \triangleq F(\vec{x}, \vec{r}) \not\equiv G(\vec{x}).$$

We use the term **rectifying vector** to denote a truth assignment $\vec{v}$ to signals $\vec{r}$. Lastly, we define the **rectifying cofactor** of RM with respect to $\vec{v}$ as $RM(\vec{x}, \vec{r} = \vec{v})$.

2.1 Rectification Functions

In this subsection, we first review the definitions of *single-fix* and *multi-fix rectification functions* in terms of the notations defined above.

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

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

From the definitions, we can see that the single-fix rectification function is a special case of the multi-fix rectification functions when $m = 1$. While the necessary and sufficient conditions for the existence of the single-fix rectification function was given in [15, 18], in this paper, we derive the necessary and sufficient conditions for the existence of the multi-fix rectification functions in Section 3. Moreover, we use interpolation and cofactor reduction techniques to produce them.

2.2 Interpolation and Rectification Functions

To introduce the interpolation technique 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. Given a refutation proof of A and B, the interpolant can be derived in linear time [19, 20]. In this paper, we use $ITP(A, B)$ to denote the interpolation procedure proposed in [19].

In the literature, there are two types of rectification functions which can be constructed by interpolation. First, prior works [17, 26] use interpolation to produce a single-fix rectification function. Second, the work in [23] utilizes multiple passes of partial rectifications to fix multiple errors in the old circuit. However, these works generate one rectification function in one iteration without considering the rectification functions in other iterations. Therefore, these approaches can lead to unnecessarily large patches because the correlations between the rectification functions in different iterations are not considered. In this paper, we resolve the above shortcomings and propose an algorithm to produce multi-fix rectification functions.

3. CONSTRUCTING MULTI-FIX RECTIFICATION FUNCTIONS

3.1 Multiple Error Rectifiability Problem

Given old and new functions $F(\vec{x})$ and $G(\vec{x})$ with internal signals $\vec{r}$ in F, the multiple error rectifiability problem is to determine whether $F(\vec{x})$ is multi-rectifiable at signals $\vec{r}$. Intuitively, $F(\vec{x})$ is multi-rectifiable if for each minterm $\vec{x}$, there exists a truth assignment to signals $\vec{r}$ such that $F(\vec{x}, \vec{r}) \equiv G(\vec{x})$. Formally, we have the following proposition.

PROPOSITION 1. *For multiple-output functions $F = \langle f_1, \ldots, f_n\rangle$ and $G = \langle g_1, \ldots, g_n\rangle$, function $F(\vec{x})$ is multi-rectifiable at internal signals $\vec{r}$ with respect to $G(\vec{x})$ if and only if*

$$\forall \vec{x}. \exists \vec{r}. F(\vec{x}, \vec{r}) \equiv G(\vec{x}) \tag{1}$$

is evaluated to true.

The above single alternation $\forall\exists$-formula is a 2QBF formula. Solving this specific 2QBF formula is a Π_2^P-complete problem [25]. In the following subsections, we will demonstrate how we can effectively reduce this problem to a SAT problem, and use interpolation techniques to generate rectification functions as by-products. This is achieved by first conducting a translation to reduce 2QBF to SAT, and then perform a *cofactor reduction* technique for rectification function generation.

3.2 Solving Rectifiability Problem by Satisfiability with Cofactor Reduction

According to Proposition 1, the multiple error rectifiability problem can be solved by QBF solver. However, since Formula (1) is a special 2QBF formula, we can simply translate it to a SAT formula. More importantly, by utilizing the unsatisfiability proof of a SAT solver, we can generate multiple interpolants which are the desired rectification functions. The following lemma reduces the multiple error rectifiability problem to a SAT problem.

LEMMA 1. *For multiple-output functions $F = \langle f_1, \ldots, f_n\rangle$ and $G = \langle g_1, \ldots, g_n\rangle$, function $F(\vec{x})$ is multi-rectifiable at internal signals $\vec{r}$ with respect to $G(\vec{x})$ if and only if*

$$\bigwedge_{\vec{v} \in \mathbb{B}^{|\vec{r}|}} RM(\vec{x}, \vec{r} = \vec{v}) \tag{2}$$

is unsatisfiable.

PROOF. By taking the negation of Formula (1), we obtain

$$\exists\vec{x}.\forall\vec{r}.F(\vec{x},\vec{r}) \not\equiv G(\vec{x})$$

is evaluated to false, which is equivalent to the unsatisfiability of Formula (2). □

The intuitive meaning of Lemma 1 can be explained as follows. If Formula (2) is unsatisfiable, then each input minterm $\vec{x}$ can make certain rectifying cofactor $RM(\vec{x},\vec{r}=\vec{v})$ unsatisfiable. That is, each minterm $\vec{x}$ can be rectified by the corresponding rectifying vector $\vec{v}$. Therefore, we can conclude that the old function F can be rectified at the internal signals $\vec{r}$.

However, evaluating Formula (2) in Lemma 1 involves a satisfiability solving of an exponential number of $RM(\vec{x},\vec{r}=\vec{v})$ terms. To simplify the computation, we propose a *cofactor reduction* technique in Algorithm 1 which can alleviate the computational cost for most practical cases.

Algorithm 1 CofactorReduction($F(\vec{x}), G(\vec{x}), \vec{r}$)

```
1: $rv\text{-}set \leftarrow \{(0,0,\ldots,0)\}$
2: $rc\text{-}set(\vec{x}) \leftarrow RM(\vec{x},\vec{r}=(0,0,\ldots,0))$
3: loop
4:    $result \leftarrow$ SatSolve($rc\text{-}set(\vec{x})$)
5:    if $result$ == UNSAT then
6:       return UNSAT
7:    else {with SAT assignment $\vec{x}^*$}
8:       $result \leftarrow$ SatSolve($\neg RM(\vec{x}=\vec{x}^*,\vec{r})$)
9:       if $result$ == UNSAT then
10:         return SAT
11:      else {with SAT assignment $\vec{v}^*$}
12:         $rv\text{-}set \leftarrow rv\text{-}set \cup \vec{v}^*$
13:         $rc\text{-}set(\vec{x}) \leftarrow rc\text{-}set(\vec{x}) \wedge RM(\vec{x},\vec{r}=\vec{v}^*)$
14:      end if
15:   end if
16: end loop
```

Algorithm CofactorReduction above presents an implementation of the satisfiability check for Formula (2) in an iterative manner. Let *rv-set* and *rc-set* be sets of rectifying vectors and cofactors, respectively. These two sets will grow synchronously in the algorithm. Initially, $rc\text{-}set(\vec{x})$ equals $RM(\vec{x},\vec{r}=(0,0,\ldots,0))$. After starting the loop, $rc\text{-}set(\vec{x})$ will be iteratively conjuncted with other rectifying cofactors. In the loop body of the algorithm, the satisfiability of $rc\text{-}set(\vec{x})$ is firstly determined. If the result is unsatisfiable, it means that each input minterm can make at least one rectifying cofactor in *rc-set* unsatisfiable. Therefore, the current *rv-set* can rectify all input minterms. The algorithm then returns UNSAT and terminates. On the other hand, if the result is satisfiable, we can obtain a satisfying assignment $\vec{x}^*$ that satisfies all rectifying cofactors in *rc-set* and conclude that the current rectifying vectors in *rv-set* cannot rectify the input minterm $\vec{x}^*$. Then, a SAT solver is invoked on formula $\neg RM(\vec{x}=\vec{x}^*,\vec{r})$. The purpose of this SAT solving is to check whether there exists another rectifying vector which can rectify the input minterm $\vec{x}^*$. If the SAT solver returns UNSAT, it means that *every* rectifying vector cannot rectify the input minterm $\vec{x}^*$. Hence, the algorithm returns SAT. Otherwise, the SAT solver returns SAT with a satisfying assignment $\vec{v}^*$ which can rectify the input minterm $\vec{x}^*$. Therefore, $rc\text{-}set(\vec{x})$ is conjuncted with a new rectifying cofactor $RM(\vec{x},\vec{r}=\vec{v}^*)$, and $\vec{v}^*$ is added to *rv-set*. The algorithm then goes to line 3 and iterates.

PROPOSITION 2. *Algorithm* CofactorReduction *decides the satisfiability of Formula (2).*

With the cofactor reduction technique, evaluating Formula (2) only needs to consider the set *rc-set* of rectifying cofactors instead of expanding all rectifying cofactors. The following lemma utilizes the set *rv-set* of rectifying vectors recorded in Algorithm CofactorReduction to solve the multiple error rectifiability problem.

LEMMA 2. *For multiple-output functions $F=\langle f_1,\ldots,f_n\rangle$ and $G=\langle g_1,\ldots,g_n\rangle$, function $F(\vec{x})$ is multi-rectifiable at internal signals $\vec{r}$ with respect to $G(\vec{x})$ if and only if*

$$\bigwedge_{\vec{v}\in rv\text{-}set} RM(\vec{x},\vec{r}=\vec{v}) \qquad (3)$$

is unsatisfiable.

PROOF. This lemma can be proved by Lemma 1 and Proposition 2. □

3.3 Generating Rectification Functions by Interpolation with Cofactor Reduction

To perform the rectification function construction with cofactor reduction, we first define $rv\text{-}set_0(i)$ and $rv\text{-}set_1(i)$ for the i-th value of a given *rv-set* of rectifying vectors as follows.

$$rv\text{-}set_0(i) = \{\vec{v}_{[i,m]} \mid \vec{v}\in rv\text{-}set \text{ and } v_i = 0\}$$
$$rv\text{-}set_1(i) = \{\vec{v}_{[i,m]} \mid \vec{v}\in rv\text{-}set \text{ and } v_i = 1\}$$

EXAMPLE 1. *Let $rv\text{-}set=\{(0,0,0),(0,1,1),(1,0,0),(1,1,1)\}$. Then $rv\text{-}set_0(1)=\{(0,0,0),(0,1,1)\}$, $rv\text{-}set_1(1)=\{(1,0,0),(1,1,1)\}$, $rv\text{-}set_0(2)=\{(0,0)\}$, $rv\text{-}set_1(2)=\{(1,1)\}$, $rv\text{-}set_0(3)=\{0\}$, and $rv\text{-}set_1(3)=\{1\}$.*

Next we show that given old and new functions F and G with internal signals $\vec{r}$ in F and the set *rv-set*, when Formula (3) is unsatisfiable, we can derive a set of multi-fix rectification functions $\vec{p}$ at signals $\vec{r}$. These rectification functions p_i's can be obtained through the following iterative derivation.

$$\begin{aligned}
p_1^{on}(\vec{x}) &= \bigwedge_{\vec{v}\in rv\text{-}set_0(1)} RM(\vec{x},\vec{r}=\vec{v})\\
p_1^{off}(\vec{x}) &= \bigwedge_{\vec{v}\in rv\text{-}set_1(1)} RM(\vec{x},\vec{r}=\vec{v})\\
p_1(\vec{x}) &= ITP(p_1^{on}(\vec{x}), p_1^{off}(\vec{x}))\\
&\vdots\\
p_i^{on}(\vec{x}) &= \bigwedge_{\vec{v}\in rv\text{-}set_0(i)} RM(\vec{x},\vec{r}=[\vec{p}_{[1,i-1]} \parallel \vec{v}])\\
p_i^{off}(\vec{x}) &= \bigwedge_{\vec{v}\in rv\text{-}set_1(i)} RM(\vec{x},\vec{r}=[\vec{p}_{[1,i-1]} \parallel \vec{v}])\\
p_i(\vec{x}) &= ITP(p_i^{on}(\vec{x}), p_i^{off}(\vec{x}))\\
&\vdots\\
p_m^{on}(\vec{x}) &= RM(\vec{x},\vec{r}=[\vec{p}_{[1,m-1]} \parallel 0])\\
p_m^{off}(\vec{x}) &= RM(\vec{x},\vec{r}=[\vec{p}_{[1,m-1]} \parallel 1])\\
p_m(\vec{x}) &= ITP(p_m^{on}(\vec{x}), p_m^{off}(\vec{x}))
\end{aligned}$$

LEMMA 3. *Given a function F which is multi-rectifiable at internal signals $\vec{r}$ with respect to G, then*

$$p_i^{on} \wedge p_i^{off}$$

is unsatisfiable, where $i = 1, \ldots, m$.

PROOF. This lemma can be proved by induction on i. Due to the space limitation, we omit the proof here. □

Since $p_i^{on} \wedge p_i^{off}$ is unsatisfiable, the interpolant of p_i^{on} and p_i^{off}, i.e., the function p_i, is guaranteed to exist. In addition to the existence of functions p_i's, the following theorem shows that these functions indeed constitute a set of multi-fix rectification functions which can correct the old function $F(\vec{x})$.

THEOREM 2. *Given a function F which is multi-rectifiable at internal signals $\vec{r}$ with respect to G, we have $F(\vec{x}, r_1 = p_1(\vec{x}), \ldots, r_m = p_m(\vec{x})) \equiv G(\vec{x})$ for rectification functions p_i's obtained by the above derivation.*

PROOF. By Lemma 3 and interpolation theorem, function p_m is guaranteed to exist. Thus, the equation

$$p_m^{on}(\vec{x}) \leq p_m(\vec{x}) \leq \neg p_m^{off}(\vec{x}) \qquad (4)$$

holds. Consider a minterm $\vec{x}^*$ which makes $p_m(\vec{x}^*) = 0$. By definition and Equation (4), we have

$$\begin{aligned} & F(\vec{x}^*, r_1 = p_1(\vec{x}^*), \ldots, r_m = p_m(\vec{x}^*)) \not\equiv G(\vec{x}^*) \\ &= F(\vec{x}^*, r_1 = p_1(\vec{x}^*), \ldots, r_m = 0) \not\equiv G(\vec{x}^*) \\ &= RM(\vec{x}^*, \vec{r} = [\vec{p}_{[1,m-1]} \parallel 0]) \\ &= p_m^{on}(\vec{x}^*) \\ &\leq p_m(\vec{x}^*) \\ &= 0 \end{aligned}$$

Similarly, for a minterm $\vec{x}^*$ which makes $p_m(\vec{x}^*) = 1$, we can also derive the same result. Therefore, after attaching all rectification functions p_i's to signal r_i's, there is no error minterm between F and G. □

EXAMPLE 2. *Continue Example 1. With cofactor reduction, we need to consider the following rectifying cofactors during the construction.*

$$\begin{aligned} p_1^{on}(\vec{x}) &= RM(\vec{x}, \vec{r} = (0,0,0)) \wedge RM(\vec{x}, \vec{r} = (0,1,1)) \\ p_1^{off}(\vec{x}) &= RM(\vec{x}, \vec{r} = (1,0,0)) \wedge RM(\vec{x}, \vec{r} = (1,1,1)) \\ p_2^{on}(\vec{x}) &= RM(\vec{x}, \vec{r} = [p_1 \parallel (0,0)]) \\ p_2^{off}(\vec{x}) &= RM(\vec{x}, \vec{r} = [p_1 \parallel (1,1)]) \\ p_3^{on}(\vec{x}) &= RM(\vec{x}, \vec{r} = [\vec{p}_{[1,2]} \parallel 0]) \\ p_3^{off}(\vec{x}) &= RM(\vec{x}, \vec{r} = [\vec{p}_{[1,2]} \parallel 1]) \end{aligned}$$

The number of RM terms for constructing the rectification functions p_i's is 8. Without cofactor reduction, rv-set will include all rectifying vectors. Therefore, we need to consider the following rectifying cofactors.

$$\begin{aligned} p_1^{on}(\vec{x}) = {} & RM(\vec{x}, \vec{r} = (0,0,0)) \wedge RM(\vec{x}, \vec{r} = (0,0,1)) \wedge \\ & RM(\vec{x}, \vec{r} = (0,1,0)) \wedge RM(\vec{x}, \vec{r} = (0,1,1)) \\ p_1^{off}(\vec{x}) = {} & RM(\vec{x}, \vec{r} = (1,0,0)) \wedge RM(\vec{x}, \vec{r} = (1,0,1)) \wedge \\ & RM(\vec{x}, \vec{r} = (1,1,0)) \wedge RM(\vec{x}, \vec{r} = (1,1,1)) \\ p_2^{on}(\vec{x}) = {} & RM(\vec{x}, \vec{r} = [p_1 \parallel (0,0)]) \wedge RM(\vec{x}, \vec{r} = [p_1 \parallel (0,1)]) \\ p_2^{off}(\vec{x}) = {} & RM(\vec{x}, \vec{r} = [p_1 \parallel (1,0)]) \wedge RM(\vec{x}, \vec{r} = [p_1 \parallel (1,1)]) \\ p_3^{on}(\vec{x}) = {} & RM(\vec{x}, \vec{r} = [\vec{p}_{[1,2]} \parallel 0]) \\ p_3^{off}(\vec{x}) = {} & RM(\vec{x}, \vec{r} = [\vec{p}_{[1,2]} \parallel 1]) \end{aligned}$$

In this case, we need 14 rectifying cofactors.

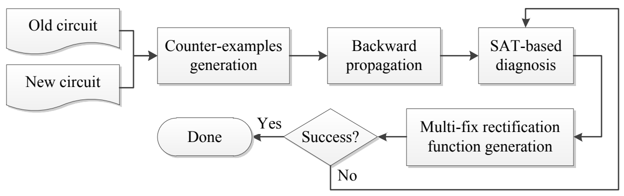

Figure 1: The integrated logic rectification flow.

3.4 Integrated Logic Rectification Flow

In this section, we introduce the main flow of our algorithm. Given old and new circuits, our flow iteratively diagnoses a set of error locations. Based on these error locations, multi-fix rectification functions are generated by interpolation to rectify the old circuit. Once the rectification procedure succeeds, the process terminates. As shown in Figure 1, our flow includes counter-example generation, backward propagation, SAT-based diagnosis, and multi-fix rectification function generation.

To facilitate the diagnosis process, a set of counter-examples which are the nonequivalent assignments of $F(\vec{x})$ and $G(\vec{x})$ is generated by combinational equivalence checker. Then, backward propagation [13] is performed for each counter-example to mark the signals that, when the value of any one of them is changed, the erroneous output response can be rectified. Intuitively, these marked signals can be treated as error location candidates, which will be later used for inserting multiplexers at the SAT-based diagnosis stage.

The SAT-based diagnosis technique used in our flow is based on the work by Smith *et al.* [22]. Given old and new circuits with a set of counter-examples, and a set of correct output responses, we first insert multiplexers at the error location candidates. Then, the diagnosis procedure will return a set of error locations with predetermined size. The predetermined size can be enforced by cardinality constraints [22], which can be dynamically adjusted. If the diagnosis procedure fails to find the error locations, we can enlarge the predetermined size by adjusting the cardinality constraints. Note that we adopted Smith's technique [22] due to its scalability. Any other efficient diagnosis method can be used as well.

After finding a set of error locations, the flow proceeds to generate multi-fix rectification functions by interpolation. If the rectification process fails, the main procedure goes to the diagnosis 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 for both benchmark circuits and industrial designs on a Linux machine with Intel Xeon 2GHz CPU and 16GB memory. The correctness of the experiments was verified by the ABC command `cec`.

The first set of our experiments are on the circuits in ISCAS89 and ITC99 benchmarks. Sequential circuits are converted to combinational by cutting off sequential elements and treating their inputs and outputs as primary outputs and inputs, respectively. The old and new circuits in the benchmark suites are created for the experiments as follows. The old circuit is changed from the original one by cube modifications. To perform the modification, we implement a program to change cubes randomly from the functions of selected gates. Then the modified and original circuits are optimized and mapped by ABC with command `map` using `cadence.genlib` technology library to produce old and new circuits, respectively. After that, the old and new circuits are both structurally and functionally different.

We compare our results with [23], a most recent work in the literature which uses interpolation to generate partial-fix rectification functions. Table 1 and Table 2 summarize the first set of experimental results. For each circuit, we conducted experiments with three settings including one cube modification (1C), two cube modifications (2C), and three cube modifications (3C) on the old circuit. For each circuit and for each setting, we randomly ran the experiments three times and the result was averaged over these three individual experiments. Table 1 shows the number of cells on the old and new circuits in each setting. For Table 2, Columns 2 to 4 are the patch sizes of our method and Columns 5 to 7 are the patch sizes of [23]. The corresponding runtime of our method and [23] are shown in Columns 9 to 11 and Columns 12 to 14, respectively. Column 8 and Column 15 are the average values of the ratios which are defined below:

$$Ratio_p = \frac{\frac{C_2}{C_5} + \frac{C_3}{C_6} + \frac{C_4}{C_7}}{3}, \text{ and}$$

$$Ratio_t = \frac{\frac{C_9}{C_{12}} + \frac{C_{10}}{C_{13}} + \frac{C_{11}}{C_{14}}}{3},$$

where C_i represents the value of Column i. Among all of these testcases, our method can consistently produce much better results and the total average ratio of patch size is about 11%. This means we can use only one tenth of gate counts in [23] to fix the old circuit. For the cases of `s15850`, `s15850.1`, and `s38417`, the patch sizes in our approach are less than 5% of the patch size in [23]. Although there are some testcases which spend more time in our method, the overheads are acceptable with the benefit of much smaller patch sizes.

Furthermore, another set of experiments are designed to demonstrate the applicability of our approach. Table 3 shows the experimental results for RTL designs. `SDRAM` implements the finite state machine in a SDRAM controller. `I2C` is a two-wire and bidirectional serial bus that provides a simple and efficient method of data exchange between devices. `VGA_PGEN_M` represents the OpenCores VGA/LCD controller core. This set of experiments are conducted as follows. We perform two types of modifications to alter the RTL designs - replacing the logic operations (logic), and changing the wire connections (rewire). After being converted from RTL into the blif format by using our internally developed synthesis tool, the old and new circuits are optimized and mapped by the same way as the first set of our experiments. The results show the patch sizes in our approach are much smaller than the patch sizes in [23], so our approach is also applicable for RTL designs.

Finally, we conduct another set of experiments to demonstrate the effectiveness of the cofactor reduction technique. The results are plotted in Figure 2. Both x-axis and y-axis units are in log scale. If a point is above the dotted line, it indicates that our approach with cofactor reduction outperforms the one without cofactor reduction. According to Figure 2, we can see that for different sizes of circuits, the cofactor reduction technique helps in reducing the runtime in most of the cases.

5. CONCLUSIONS

We have presented a novel logic rectification algorithm by interpolation. By deriving the necessary and sufficient conditions for the existence of multi-fix rectification functions, our method generates these functions to fix the erroneous gate-level implementation. In addition, we propose a cofactor reduction technique to alleviate the computational cost. Experimental results demonstrate that our integrated logic rectification algorithm is efficient and effective, and the generated patches can be maintained in a small size. Moreover, the results show that our algorithm is applicable to the circuits synthesized from real RTL designs.

Table 1: Profile of ISCAS89 and ITC99 benchmarks.

Circuits	Old #Cells			New #Cells
	1C	2C	3C	
s3271	705	704	705	705
s3384	671	674	669	672
s6669	1468	1468	1467	1468
s9234.1	1188	1206	1205	1205
s13207	1781	1789	1789	1789
s15850	2011	2008	2008	2012
s15850.1	2014	2020	2017	2019
s35932	4645	4644	4647	4646
s38584.1	6650	6651	6647	6651
s38417	5286	5286	5287	5286
b14	3478	3479	3480	3477
b15	4924	4928	4923	4929
b17	15237	15255	15269	15237
b20	7063	7066	7061	7064
b21	7226	7219	7217	7217
b22	10819	10821	10818	10817

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. Symbolic Logic*, 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.

Table 2: Experimental results on ISCAS89 and ITC99 benchmarks.

Circuits	Patch Size #AIG							Time (s)						
	Ours			DAC 2011 [23]			$Ratio_p$	Ours			DAC 2011 [23]			$Ratio_t$
	1C	2C	3C	1C	2C	3C		1C	2C	3C	1C	2C	3C	
s3271	4	5.7	9	115.3	59	111.7	0.07	0.6	0.6	1.6	0.6	0.4	0.8	1.50
s3384	1	4.7	25.3	20.3	83.3	53.7	0.19	0.1	0.2	1.1	0.1	0.4	0.3	1.72
s6669	2.3	5.3	29.3	38.3	137.7	117.7	0.12	1.5	0.7	30.3	0.3	0.7	33.6	2.30
s9234.1	104.7	3.3	57	330.7	95.3	158.3	0.24	6.3	0.2	3.1	3.3	1	1.7	1.31
s13207	5	2	3.3	64.3	13	20.7	0.13	2	1	1.3	2.8	2	2.6	0.57
s15850	3.3	5	4.7	226.3	232.3	290.7	0.02	3.5	3.3	2.4	5.2	5.2	6.2	0.56
s15850.1	2	3	5	337.7	48.7	105	0.04	5.3	0.4	0.5	8.6	1.4	2.2	0.38
s35932	1.3	3	7	38	56	67	0.06	0.2	0.3	0.4	0.6	0.7	1	0.39
s38584.1	4.3	5.7	22	133.3	54	75.7	0.14	40.7	0.5	47.5	15.1	12.1	14.7	1.99
s38417	2.3	3	5	65	83	248.7	0.03	0.5	0.5	4	3.1	4.1	17.6	0.17
b14	2	24.7	260.3	10.3	370	1059.3	0.17	52.9	150.5	246.1	1.3	135.4	277.1	14.23
b15	36	34	36	161	207.3	522.7	0.15	510.4	198.8	775.2	28.1	6.4	33.3	24.17
b17	2	59	551.7	587.3	506.7	1760.3	0.14	14.7	168.1	787.5	514.8	492.5	1717.5	0.28
b20	4.3	11	75.7	591.7	886	498.3	0.06	120.9	768.7	102.8	301.1	481.3	18.1	2.56
b21	3.7	6	19.3	230	595.3	205.3	0.04	159.6	748.2	244.8	10.3	450.3	10.3	13.64
b22	1.3	6.3	184.7	69.7	158	561.3	0.13	12.1	15.6	693.9	5	11	253.8	2.19
Average							0.11							4.25

Table 3: Experimental results on RTL designs.

RTL Circuits	Error Type	Ours		DAC 2011 [23]	
		Patch	Time	Patch	Time
SDRAM_1	2 logic	6	0.42	47	0.22
SDRAM_2	3 logic + 1 rewire	47	0.62	128	0.31
SDRAM_3	4 logic + 2 rewire	61	1.24	82	0.25
SDRAM_4	5 logic + 1 rewire	85	12.09	211	0.66
I2C_1	3 logic	32	0.94	98	0.43
I2C_2	4 logic	80	3.1	243	1.14
I2C_3	1 logic + 2 rewire	35	1.93	61	0.26
I2C_4	1 logic + 2 rewire	35	1.86	63	0.27
I2C_5	2 logic + 3 rewire	126	21.86	303	1.06
VGA_PGEN_M_1	2 logic	65	1.9	185	1.11
VGA_PGEN_M_2	3 logic	65	1.33	186	1.14
VGA_PGEN_M_3	4 logic	70	2.25	187	1.15
VGA_PGEN_M_4	5 logic	74	1.49	190	1.17

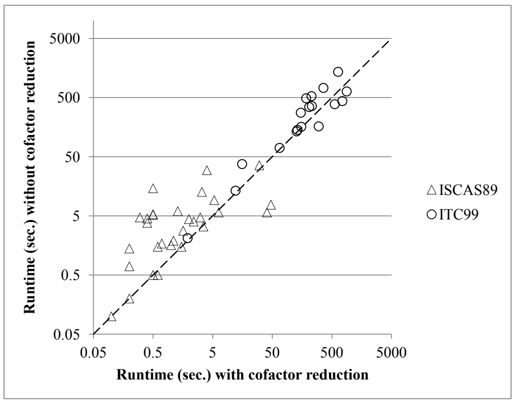

Figure 2: Effects of cofactor reduction.

[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] K. L. McMillan. Interpolation and SAT-based model checking. In *Proc. CAV*, pages 1–13, 2003.

[20] P. Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *J. Symbolic Logic*, 62(3):981–998, 1997.

[21] T. Shinsha, T. Kubo, Y. Sakataya, J. Koshishita, and K. Ishihara. Incremental logic synthesis through gate logic structure identification. In *Proc. DAC*, pages 391–397, 1986.

[22] A. Smith, A. G. Veneris, and A. Viglas. Design diagnosis using Boolean satisfiability. In *Proc. ASP-DAC*, pages 218–223, 2004.

[23] K.-F. Tang, C.-A. Wu, P.-K. Huang, and C.-Y. Huang. Interpolation-based incremental ECO synthesis for multi-error logic rectification. In *Proc. DAC*, pages 146–151, 2011.

[24] 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.

[25] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theor. Comp. Science*, 3(1):23–33, 1976.

[26] 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.

[27] 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.