A Robust Functional ECO Engine by SAT Proof Minimization and Interpolation Techniques

Bo-Han Wu, Chun-Ju Yang, Chung-Yang (Ric) Huang, Jie-Hong Roland Jiang
Graduate Institute of Electronics Engineering/Department of Electrical Engineering, National Taiwan University

Abstract

Functional rectification in late design stages has been a crucial process in modern complex system design. This paper proposes a robust functional ECO engine, which applies SAT proof minimization and interpolation techniques to automate patch construction to make old implementation and golden specification functionally equivalent. The SAT proof minimization technique provides a sound and efficient way of fixing easy errors, and the interpolation technique provides a complete and robust way of fixing remaining errors. Experimental results show that our engine performs robustly to generate small patches in fixing various design rectification instances.

I. INTRODUCTION

In the modern VLSI design flow, a design typically goes through many synthesis and optimization stages. If the specification changes or a functional bug is found at a late stage, it is impractical to restart the design flow from the beginning as it would greatly affect the cost and time-to-market. This is when engineering change order (ECO) comes into play. The goal of the engineering change is to help the designer identify a part of the *old* implementation that should be modified such that the resulting circuit is functionally equivalent to the *golden* specification. In other words, we need a *patch* that, when applied to the old circuit, makes it equivalent to the golden one. Here the patch size should be minimized so as to lower the cost in consequent rectifications.

A simple and intuitive approach to ECO is to choose an internal signal in the old circuit, and replace it with a newly synthesized signal based on the golden circuit. With different formulations such as functional decomposition and synthesis by quantification, several algorithms under this idea have been proposed [1-2]. By the use of BDDs, these algorithms are relatively easy to implement. However, there are two major drawbacks of these algorithms. First, it is generally more expensive, if not impossible, to rectify the actual difference between two circuits by a single signal. For example, if there are two small, but yet far-from-each-other changes in the golden circuit, the single replacement would be limited in the common region of the transitive fanout cones of these two changes. The other drawback is that the size of the problem instance is bounded by the BDD capacity, which makes these algorithms impractical for modern designs.

To avoid these drawbacks, some researchers treat ECO as a design error correction problem [3-4]. By viewing the old circuit as an erroneous design, and treating the golden circuit as the corrected one, the patch generation is exactly an error correction process, to which many approaches exist already. Usually an error correction algorithm comes with an error model [5], which describes all the possible error types, including wrong gate-type, missing inverter, misplaced wire, and so on. The advantage of the error-model approach is that with the smaller search space it will allow us to find the solution within acceptable runtime. Nevertheless, its disadvantage is that these algorithms often fail to produce the patch when the golden circuit is too different from the old one or the difference cannot be represented by the error models.

Since the error-model approach possesses such possibility of failure, a technique called test-vector-guided approach is proposed to avoid the BDD capacity problem as well as to guarantee fixing the old circuit. For example, Co'Re [6] uses a set of error minterms to identify the signal to be fixed and then resynthesizes a replacing logic from the nearby signals. The circuit is guaranteed to be fixed under the chosen minterms, but not necessarily for the others. As a consequence, if a discrepancy still exists between the fixed old circuit and the golden one, more error minterms for the discrepancy will be added to the error minterm set and the whole process needs to be started over again. Although this algorithm seems efficient for some examples, in the worst case, it may end up fixing only a small set of minterms at a time and resulting in long runtime. If the design has many primary inputs, the number of error minterms can be enormous, and thus there is a chance that this algorithm will not converge in acceptable runtime.

Instead of performing ECO by examining the behavior of primary inputs and primary outputs, a recent work DeltaSyn [7] has been proposed to match the old and golden circuits in two phases. First, it seeks for functionally equivalent signals in the two circuits by both structural and functional information. With the location of these equivalent signals known, part of the two circuits is matched from primary inputs up to these signals. Then, the two circuits are matched from primary outputs by the Boolean matching technique in [8]. This algorithm would work perfect if the difference between the two circuits is small and close to primary outputs. However, for cases that the golden circuit is much different from the old one, it may result in large unmatched subcircuit and patches.

Recent work [9] also exploited SAT solving and interpolation as the underlying ECO techniques. The authors utilized MAX-SAT to locate the potential signals to rectify design discrepancy. Then SAT-based function dependency check [10] and function decomposition [11] are used to derive replacing signals and to fit in FPGA look up tables, respectively. Although this algorithm does not suffer from the problem mentioned above, it may probably produce a patch far from optimal when MAX-SAT fails to identify the real positions of the difference between the two circuits. We improve the work by providing an equivalent yet simpler interpolation formulation.

In this paper, we propose a robust ECO engine that can efficiently work on complicated as well as simple ECO problems. We enhance the error-model and the test-vector-guided approaches by recording the potential fixes on a MUX-remodeled circuit. Then the SAT proof-core minimization technique is applied to minimize the patch size. If the above method does not work well, or if it can only fix partial differences, we will resort to the interpolation-based technique to resynthesize the (remaining) functional difference between the old and golden circuits. Our formulation implicitly takes the observability don't-cares into consideration and the interpolation engine will utilize it to minimize the patch circuit. Experimental results show that our ECO engine can consistently identify the patches with small sizes efficiently.

The rest of this paper is organized as follows. In Section II we first outline our algorithm. Sections III and IV respectively describe the MUX-remodeled SAT proof minimization solution and the interpolation technique in details. Experimental results are presented in Section V, and Section VI concludes this paper.

II. OVERVIEW OF OUR ECO ALGORITHM

Fig. 1 outlines our ECO algorithm. In the beginning, the inputs of the old and the golden circuits are merged together by their pin names. Then FRAIG [12] is applied to identify the functionally equivalent part of the circuits. Please note that we merge the functionally equivalent sub-circuit only when there is exactly one of the equivalent signals in the old circuit. This way, we can ensure that the signal in the old circuit is reused and the equivalent part of the golden circuit is removed from the problem. It can always lead to a smaller patch size.

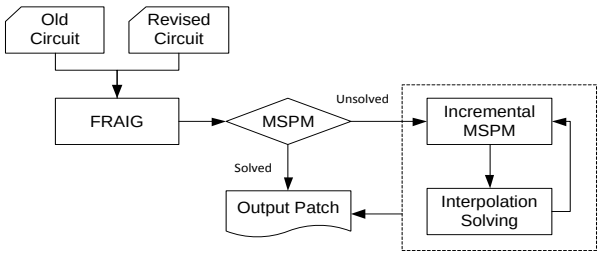


Figure 1. Flowchart of our algorithm

Next, a MUX-remodeled SAT proof minimization (MSPM) approach tries to solve the problem with an error-model library containing different types of modifications. This approach makes use of simulation and equivalent checking to recursively identify possible modifications one at a time. Then a collection of individual modifications are remodeled by MUX gates on the old circuit and if there exists an assignment on the selecting signals of the MUXes that can make the old and golden circuits equivalent, we find a feasible patch of this problem. We then apply SAT proof minimization technique to minimize the patch.

However, in some cases, the difference between the two circuits is so complex that the MSPM is unable to rectify it. To solve this problem, we first turn to the incremental MSPM, which tries to fix as many primary outputs as possible. The remaining unfixed part is then passed to the interpolation technique. Since the interpolation actually synthesizes the functional difference between the old and the golden circuits, it is guaranteed to find a solution. In addition, during the interpolation solving process, we can still check whether there are primary outputs that can be fixed by incremental MSPM after one or more interpolants have been replaced into the old circuit. As this iteration goes on, we can always find a patch, which in the extreme case, is generated entirely by interpolation.

III. MUX-REMODELED SAT PROOF MINIMIZATION

MSPM is a recursive algorithm with the capability of finding small modifications for the error cubes and then minimizing the resulting patch. The flowchart is as shown in Fig. 2. At each recursion level, the merged circuits are first passed to the equivalence checker to find an error vector. If an error vector is found, the circuits are simulated with this vector and the simulation result is then used to identify the remodeling candidates. When such a candidate is found, it is remodeled by our MUX model for later patch minimization. Then the resulting modified circuits are solved recursively as a sub-problem by the same procedure. If the solving of the sub-problem fails, that is, if we cannot find a remodeling candidate from the error-model library, the modification is reverted and we continue to find another potential error.

Since we do not need this algorithm to be a complete solution, we can set a limit on the recursion depth and the number of modifications tried at each recursion. Therefore, we can easily change the size of the search space as a tradeoff on runtime. In the following subsections, we will describe how to find remodeling candidates, detail MUX remodeling and patch minimization, and extend MSPM in an incremental manner.

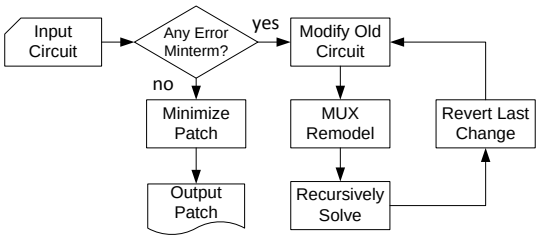


Figure 2. Flowchart of MSPM solution

A. *Finding a Remodeling Candidate*

Given an error vector $\vec{v}$, we say a signal *s* in the old circuit is *rectifiable under* $\vec{v}$ if and only if it satisfies the following property:

$$[Old_i \oplus G_i](\vec{v}) \equiv \frac{d(Old_i)}{ds}(\vec{v}), \forall i, \quad (1)$$

where Old_i is the function of a primary output in the old circuit, and G_i is the corresponding one in the golden circuit. In other words, if we change the value of *s*, we rectify the difference between the two circuits under the vector $\vec{v}$.

To find a remodeling candidate to rectify the error vector, clearly, we should identify the rectification signals first. To locate such signals, the two circuits are first simulated with the error vector $\vec{v}$. Then starting from the outputs with different simulation values, we traverse backward on the old circuit to see if there is a controlling signal that, when its value is flipped, can change the values on the outputs simultaneously. If we fail to find such signals, we will continue for the next error vector.

Our error-model library includes various frequently-encountered error patterns and can be categorized as follows:

- Missing inverter: There is an inverter missing or undesirably present on a wire.
- Rewiring: An input of a gate is connected from an incorrect signal.
- Wrong gate type: A gate is of wrong type. Let a and b be the gate inputs, we categorize gates into five types, including $a \cdot b$ (AND), $a + b$ (OR), $a \oplus b$ (XOR), $a \cdot \bar{b}$, and $\bar{a} \cdot b$. Note that the last two types are included to cover the missing inverter modifications on wire branches.

For a rectification signal, we modify it by different types of error models in the library that can flip the simulation values. If the resulting values at primary outputs are rectified, then we have a remodeling candidate. However, there may be too many such candidates and most of them may be spurious. To quickly screen out the spurious ones, we simulate the circuits together with a great amount of correct vectors. In other words, a valid remodeling candidate should fix the error vector, yet at the same time keep the correct vectors intact. This can greatly reduce the probability of choosing a wrong candidate.

B. *MUX remodeling*

Once we find a remodeling candidate, we do not commit it directly. Instead, we use a MUX to remodel the modification. As illustrated by the example in Fig. 3, suppose the OR gate is a remodeling candidate for a rectification signal (the AND gate) in the old circuit. We then build the MUX structure, which has the original signal as one of its input, and the modified signal as the other. In a later recursion step, we can set the value of *D* to 1 so that the signal *C* is equivalent to the modified signal. This allows us to identify the next remodeling candidate for other error vectors.

This MUX model makes it easy to recover from the modification if we need to backtrack in the searching process and at the same time enables the patch minimization as we will describe in the following subsection.

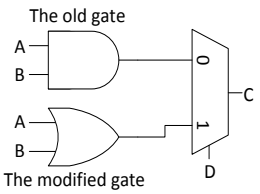


Figure 3. An example of MUX modeling

C. Patch Minimization by SAT Proof

For the MUX-remodeled circuit, if there exists an assignment on the selecting signals of the MUXes that make the old and golden circuits equivalent, a possible patch is found. This equivalence checking problem is solved by the SAT engine when it concludes that the output difference of the merged circuits is unsatisfiable.

To facilitate the generation of the patch, we arrange the MUX selecting signals as the earlier decision variables in the SAT process. In other words, any assignment on the selecting signals can be viewed as an assumption for the subsequent equivalence checking proof. If the SAT returns satisfiable, we can learn a (partial) assignment on the selection signals that is impossible to be a patch (i.e. make the proof unsatisfiable). We then add this learned constraint to the SAT problem and continue. If there is a conflict occurring at or before the decision levels of these selecting signals, an assignment that leads to the output equivalence is found. We can then compute the SAT proof core based on this assumption and identify a minimal set of modification signals as the ECO patch.

With this patch minimization technique, redundant modifications in the patch are easily found and discarded, and the patch size is reduced.

D. Incremental MSPM

When MSPM fails to fix the entire old circuit, we can still attempt to fix some of the primary outputs. At each time, we extract one erroneous primary output along with all correct primary outputs at a time. If MSPM is able to rectify this subcircuit, we accept this modification. This process is repeated until no erroneous primary output can be fixed.

To be more specific, we attempt to fix one erroneous primary output at a time until we can no longer fix any primary output by MSPM. This can speed up the ECO process by resolving the simple fixes first before we turn to a more sophisticated procedure by the interpolation technique.

IV. Interpolation-Based ECO

Fig. 4 outlines the flow of our interpolation-based ECO approach. We first search for a rectification signal which is able to fix some primary outputs and select a set of signals as the inputs of the patch function (Subsections A and B). Using these patch input signals, old and golden networks, we compose two networks that characterize the on-set and off-set of the patch functions (Subsection A). Then by proving the unsatisfiability on the conjunction of the on-set and off-set networks, we can derive an interpolant that can serve as a patch of this ECO problem (Subsection C). A synthesis process is followed to optimize the patch, and the above flow is repeated if any of the primary outputs is not yet fixed.

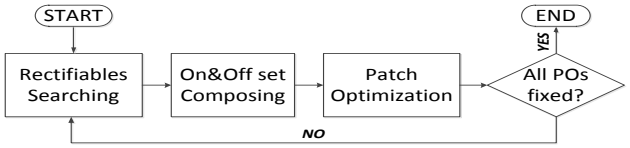


Figure 4. Flow of our interpolation-based ECO approach.

A. Theorems of the Interpolation-Based ECO Technique

In this subsection, we present the theorems that constitute our interpolation-based ECO techniques. We first focus on the single-output ECO problem in order to simplify the explanation. Table I lists the notations used in this subsection.

Theorem 1 Let c be stuck-at-0 and stuck-at-1 in two copies of the old network, respectively. If the following formula is unsatisfiable, then c can be a rectification signal for the difference between the old and golden circuits.

$$\begin{aligned}&\left[Old_1(c=0,I_1)\neq G_1(I_1)\right]\wedge\left(V=\vec{f}_v(I_1)\right)\wedge\\&\left[Old_2(c=1,I_2)\neq G_2(I_2)\right]\wedge\left(V=\vec{f}_v(I_2)\right)\end{aligned}\quad(2)$$

Proof:

To help describing Formula (2), we construct the corresponding networks as Fig. 5.

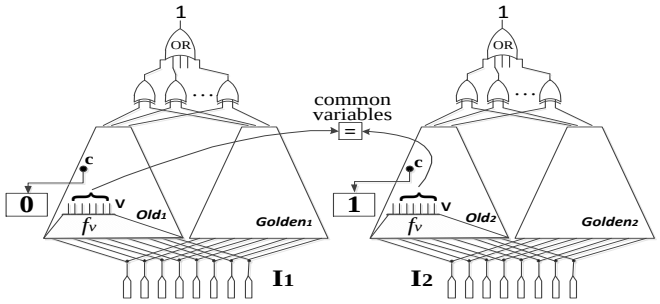


Figure 5. The corresponding network for Formula (2).

By definition, there must be at least one function such that modifying the rectification signal to this function will result in equivalence of the old and golden networks. Since V is equivalent to $\vec{f}_v$ by definition, if Formula (2) is unsatisfiable, then either Old_1 is equivalent to G_1 or Old_2 is equivalent to G_2. That is, for all the combinations of values on I_1, I_2, and V, one of conditions, $c=0$ or $c=1$, will always make the old and golden networks equivalent. In other words, we can find a maximal set of value combinations on V, say $\vec{v}_0$, that can make the old and golden networks equivalent under the $c=0$ condition. Clearly, the rest of the value combinations on V, say $\vec{v}_1$, will make the networks equivalent under $c=1$. Therefore, there must be a function of V with $\vec{v}_0$ and $\vec{v}_1$ as its off- and on-sets, respectively, that can replace the signal c and fix the old network. Therefore, the signal c is a rectification signal. □

TABLE I. Notations for the Illustration of the Interpolation-Based ECO Algorithm

Variable	*Description*
c	*A selected signal from old network.*
V	*Output variables of functions f and, effectively, patch inputs.*
$\vec{f}_v$	*Input functions of variables V in terms of primary inputs*
I_1, I_2	*Two copies of the primary inputs.*
G_1, G_2	*Two copies of the golden network*
Old_1, Old_2	*Two copies of the old network*

Note that our formula is similar to the one proposed in [9], which can be rewritten in our notations as:

$$\begin{aligned}&\left[Old_1(c=0,I_1)\neq G_1(I_1)\right]\wedge\left[Old_1(c=1,I_1)=G_1(I_1)\right]\wedge\left(V=\vec{f}_v(I_1)\right)\wedge\\&\left[Old_2(c=1,I_2)\neq G_2(I_2)\right]\wedge\left[Old_2(c=0,I_2)=G_2(I_2)\right]\wedge\left(V=\vec{f}_v(I_2)\right)\end{aligned}$$

In comparison, it is clear that our formulation is simpler. In our experience, this simplification can always lead to more than twice speedup with less memory consumption.

The signals V in Theorem 1 are introduced to serve as the common variables between the two duplicated networks. They will be used as the input variables of the patch function later.

Theorem 2 If Formula (2) is unsatisfiable, we perform Craig interpolation on V with $[Old_1(c=0,I_1)\neq G_1(I_1)]\wedge(V=\vec{f}_v(I_1))$ as the

on-set and $[Old_2(c=1,I_2)\neq G_2(I_2)] \wedge (V= \vec{f}_v(I_2))$ as the off-set. In addition, the interpolant must be able to serve as a patch function for signal *c* to fix the difference between the old and golden circuits.

Proof:

We will prove this theorem by contradiction.

Let E_0 be $[Old_1(c=0,I_1)\neq G_1(I_1)]\wedge(V= \vec{f}_v(I_1))$, and E_1 be $[Old_2(c=1,I_2)\neq G_2(I_2)]\wedge(V= \vec{f}_v(I_2))$. Formula (2) becomes $E_0 \wedge E_1$. Since Formula (2) is unsatiafiable and *V* are the only common variables between E_0 and E_1, there must be a Craig interpolation defined on *V* [13-14]. Let the interpolant function be *P*(*V*). Assume that *P*(*V*) is NOT a patch to fix the difference between old and golden networks.

Let the cube I_d make these two networks different with the replacement of the signal *c* by *P*(*V*) and the signals on *V* have values $\vec{v}_d$. Let's consider the following two cases of $P(\vec{v}_d)$:

(a) $P(\vec{v}_d)=0$

Since *c* is replaced by *P*(*V*) now, we have $c = P(\vec{v}_d)= 0$. Plugging I_d and $\vec{v}_d$ to E_0, the expression $[Old_1(c=0,I_d)\neq G_1(I_d)] \wedge (\vec{v}_d = \vec{f}_v(I_d))$ should be true because *P*(*V*) is NOT a patch.

On the other hand, by the definition of interpolation, $P(\vec{v}_d) \supseteq [Old_1(c=0,I_d)\neq G_1(I_d)] \wedge (\vec{v}_d = \vec{f}_v(I_d))$ must hold. Then we will obtain a conflict of $0 \supseteq 1$. Therefore, the interpolant must be a feasible patch function in this case.

(b) $P(\vec{v}_d)=1$

Similarly, $[Old_2(c=1,I_d)\neq G_2(I_d)] \wedge (\vec{v}_d = \vec{f}_v(I_d))$ is always true due to the assumption. By the definition of interpolation, $P(\vec{v}_d) \wedge [Old_2(c=1,I_d)\neq G_2(I_d)] \wedge (\vec{v}_d = \vec{f}_v(I_d)) = \perp$ must hold. Then we will obtain a conflict of $1\wedge 1=\perp$. Therefore, the interpolant is a feasible patch function in this case.

From the above two cases, we know that the interpolant must be a valid patch. □

Based on the two theorems, we can construct the circuits of Formula (2) to search for a possible rectification signal and its corresponding *V*. After a rectification signal and *V* are determined, we will apply the rectification signal to generate an interpolant as the desired patch by Theorem 2.

B. Searching for Rectification Signals

Given a primary output (PO) of the old circuit inequivalent to its golden counterpart, by Theorems 1 and 2, it can always be rectified and thus is a definite rectification signal. In our experience, however, the corresponding patch size is often very large. Therefore we iteratively search the rectification signal and patch inputs *V* until a patch of reasonable size is found.

A node closer to primary inputs has a higher priority to be selected as a candidate rectification signal because its fanin cone is usually smaller and similarly its patch, if exist. After a node is selected as a candidate of rectification signal, we select a cut from its fanin cone as the corresponding patch inputs *V*. A cut closer to the node has a higher priority to be selected since the patch is usually smaller. Our experience suggests that the cut selection strategy performs very well in most cases. However, to rectify rewiring-type ECO problems, this strategy may miss some rectification opportunities when patch inputs do not exist in the fanin cone of the given node. In such cases, it may be beneficial to explore signals outside of the fanin cone. To identify good candidates not restricted to the fanin cone, Formula (2) can be useful by letting *V* be the primary inputs. By the interpolant resulted from the unsatisfiability of Formula (2), we know which of the primary inputs can rectify the circuit and search only those signals that depend on these primary inputs.

Given a candidate rectification signal and its corresponding candidate patch inputs *V*, we are ready to construct the circuit of Formula (2). To test its satisfiability, we divide the computation into two phases, the simulation phase and the SAT-based proof phase. In the simulation phase, we adaptively simulate about 100~500 sets of 64-bit random vectors to test the satisfiability of Formula (2) in order to quickly prune impossible rectification signals. If all the simulation efforts fail to show the satisfiability of Formula (2), we enter the second phase to test if it is indeed unsatisfiable.

In the SAT-based proof phase, we apply SAT solving on Formula (2). If it is satisfiable, then we have picked a wrong combination of rectification signal and patch inputs. So we have to choose either another node or another set of patch inputs. On the other hand, if it is unsatisfiable, the rectification signal and patch inputs are found and we will modify the rectification signal using the obtained patch if its size is reasonably small.

By the above two computation phases, we can reduce the usage of a SAT solver, and thus find a good rectification signal very efficiently. In addition, we ensure that there always exists at least one rectification signal — the primary output. Because Formula (2) is always unsatisfiable if signal *c* is a primary output and signal *V* contains all primary inputs, the searching process will always succeed to find a rectification signal.

C. Patch Generation by Interpolation

After a rectification signal is determined, we can construct an interpolant from the refutation proof of Formula (2). Since the interpolant can be highly redundant, it often can be substantially simplified using logic optimization [15]. The resulting circuit, if reasonably small, is the desired patch.

By the above approach, the functional ECO problem for single-output circuits can be robustly resolved. Below we extend the approach to multi-output circuits.

D. Extension to Multi-Output Circuits

We extend and rewrite Formula (2) for multi-output circuits as follows:

$$\begin{aligned}&\left[Old_1(c=0,I_1)\not\equiv_p G_1(I_1)\right]\wedge\left(V=\vec{f}_v(I_1)\right)\wedge\\&\left[Old_2(c=1,I_2)\not\equiv_p G_2(I_2)\right]\wedge\left(V=\vec{f}_v(I_2)\right)\end{aligned} \quad (3)$$

where "$\not\equiv_p$" stands for the condition that at least one of a selected set of to-be-fixed primary outputs of the old circuit is different from its counterpart of the golden circuit (other primary outputs not constrained by the formula will remain unchanged). Formula (3) is unsatisfaible if and only if *c* is a rectification signal. The corresponding interpolant is a function on *V* and is a valid patch function to simultaneously fix all the selected primary outputs.

Fig. 6 shows the algorithm of the interpolation-based technique. *choose_node_POs* will select a node and some primary outputs. Also, *choose_V* will select a set of signals *V* within a specified circuit level with respect to the chosen node. After the above process, we build the Formula (3) circuit by *circuit_setup*, and *check_rectifiable* checks whether the chosen node can be a rectification signal or not. If yes, the generated optimized interpolant from *get_interpolant_and_optimize* can fix the selected primary outputs *po_to_be_fixed* by the function *replace* and keep the originally correct primary outputs still correct. If not, we continue to select other candidate patch inputs *V* or other rectification signals. Note that one of the conditions to leave the *while-loop* in the algorithm is that all POs are functionally equivalent to the golden circuit.

To avoid creating large patches, we set a size upper bound for accepting a patch. The function *size_check* will determine to acceptance or rejection the simplified interpolant according to its size.

Algorithm: Multiple POs ECO

```
24: Procedure MultiplePO_ECO ()
25: Let p_set denote the set of all generated patches
26:     unfix_set denote the set of all un-fixed POs
27: level_vector ← [1,3,200]
28: repeat
29:     n ← choose_a_node( )
30:     po_to_be_fixed ← choose_POs( n, unfix_set )
31:     for_each level in level_vector do
32:         V = choose_V( level )
33:         circuit_setup( n, po_to_be_fixed, V )
34:         isRect ← check_rectifiable( n )
35:         if isRect is True do
36:             patch ←get_interpolant()
37:             Collapse( patch )
38:             if size_check( patch ) is passed do
39:                 p_set ← p_set ∪ patch
40:                 replace( patch )
41:                 goto EquivalenceCheck
42:             end if
43:         end if
44:     end for_each
45: EquivalenceCheck:
46:     if unfix_set is empty or all POs are selected do
47:         return p_set
48:     end if
49: end repeat
50: return p_set
```

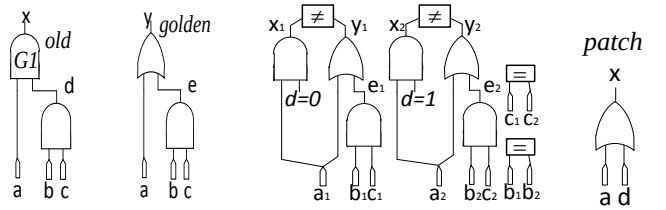
Figure 6. Pseudo codes for multi-output ECO problems

If all the patches of a to-be-fixed primary output are rejected, we will skip fixing it and continue to fix other primary outputs. There may be some unfixed primary outputs at the end. In this case, we directly use the corresponding correct outputs of the golden circuit as patches. This strategy may prevent runtime overhead due to creating large useless patches, and guarantee that patch sizes cannot be too large.

E. Example

In this subsection, we present an example in order to illustrate our idea and algorithm more clearly.

Fig. 7 (a) and (b) show the old and the golden circuits, respectively. To solve the simple example, we need to modify the old circuit such that the outputs *x* and *y* have the same functionality. First, we search for a rectification signal. Assume that we take signal *d* first. We then choose signals *b* and *c* as the signal set *V*. The corresponding network of Formula (2) is constructed as in Fig. 7(c) to test whether signal *d* is a rectification signal or not. We obtain that $\{a_1=0, b_1=b_2=1, c_1=c_2=1, a_2=0\}$ satisfies Formula (2) so that signal *d* is not a rectification signal. We then choose another signal *x*. Construct the network of Formula (2) in the same way and choose signals *a* and signal *d* as the signal set *V*. We can prove it unsatisfiable by SAT engine. Therefore, signal *x* is a rectification signal and we can obtain the interpolant by the UNSAT proof. Fig. 7(d) is the interpolant and is a feasible patch. After replacing gate *G1* in the old circuit by this patch, *x* and *y* are now functionally equivalent and the ECO process is done.



(a) Old (b) Golden (c) Network of formula (2) (d) Patch

Figure 7. ECO Example of interpolation-based method.

V. EXPERIMENTAL RESULTS

We integrate the above approaches: FRAIG, MSPM, incremental MSPM and the interpolation techniques together as our ECO engine. We create several testcases from two industry designs, and ISCAS89 and ITC99 benchmark circuits by randomly changing several cubes and wires in the gate-level netlists or some lines in the RTL codes and then performing logic optimization. The modified circuits are treated as golden circuits, and the original ones are the circuits to be rectified. The experiments are conducted on an AMD Opteron(TM) 280, 2.40GHz machine. We apply MiniSAT [16] for FRAIG operations and unsatisfiability proof. Table II shows how we measure the cost of the patch circuit.

TABLE II. COSTS OF PATCHES FOR DIFFERENT GATE TYPES

Gate Type	*Size*	*Gate Type*	*Size*	*Gate Type*	*Size*
AND2	*1*	*OR2*	*1*	*NOT*	*1*
NAND2	*1*	*NOR2*	*1*	*XOR2*	*3*

A. Comparison of Hybrid and Interpolation-Only Methods

Table III lists the results of 10 of the largest benchmark circuits. We compare the results of the hybrid (i.e. MSPM + interpolation) and the interpolation-only techniques. Column 2 records the types of modifications on the old circuits. The numbers of nodes in the old and golden circuits, denoted as OLD and GOLD, are shown in Columns 3 and 4, respectively. From Column 5 to 10 are the results of the hybrid method, which include the patch sizes, total runtime, runtime for the individual steps (FRAG, MSPM and interpolation), and the memory usage. The results for the interpolation-only approach are presented in Columns 11 to 15.

Among all of these 10 testcases, the hybrid method can consistently produce equal-sized or smaller patches than the interpolation-only ECO. Please note that the runtime for the interpolation-proof in 5 of the 10 testcases is zero. It means that these cases are totally solved by the MSPM method. In these cases, the patch sizes are ensured to be very small (patch sizes = 1 or 2) with less memory usage. This is due to the fact that we limit the number of remodeling candidates in the MSPM approach. For the case *s38417*, incremental MSPM partially fixes some POs and then the interpolation technique takes care of the rest. We can see that the patch size by hybrid method is substantially smaller than the interpolation-only approach (122 vs. 180) under comparable runtime. For the other four cases, the patches are completely generated by the interpolation technique and the sizes are also reasonably small.

In general, the performance of the ECO engine is greatly impacted by several factors, including the utilization of simulation, optimization of patches, number of POs to be fixed in each iteration, and the limitation of the patch size accepted by our engine, etc. We perform a great amount of regression to tune the best trade-off parameters for them. We will discuss one of these experiments in the following subsections.

B. Effects of Limitaions on Individual Patch Size

In this sub-section, we will utilize several different upper bounds for individual patch size and discuss how this parameter affects the patch size. For each upper bound, 15 cases are tested for the results. In Fig. 8, the Y-axis represents the final patch size and the X-axis represents the case number. We limit the individual patch size to be 1%, 5% and 10% of the FRAIGed circuits. From the results, the 10% limitation can generate smaller patches for most cases, and the 1% limitation seems too tight so that almost all fixes are choosing POs as the patches and thus become very large. Therefore, we choose 10% limitation for our ECO engine.

TABLE III. EXPERIMENTAL RESULTS

Circuit Name	Change Type	# Nodes		MSPM+Interpolation						Only Interpolation				
		OLD	*GOLD*	*Patch size*	*Total Time (s)*	*FRAIG (s)*	*MSPM (s)*	*Itp Proof(s)*	*Mem (M)*	*Patch size*	*Total Time (s)*	*FRAIG (s)*	*Itp Proof(s)*	*Mem (M)*
Industry01	RTL	794	668	2	0.8	0.07	0.7	0	5.285	6	0.22	0.06	0.01	42.55
Industry02	RTL	1668	3501	2	1.84	0.29	1.5	0	7.344	8	0.57	0.3	0.02	43.86
s9234.1	Cube	4476	3424	1	1.97	1.2	0.53	0	10.72	1	1.57	1.19	0.01	25.73
s13207	Cube	6936	5857	26	77.24	4.05	27.8	7.99	95.44	26	49.99	3.81	8.05	81.58
s13207	Rewire	6936	5822	7	42.36	3.84	30.09	1.94	86.81	7	13.36	3.82	1.93	73.12
s15850	Cube	8697	7118	1	8	6.14	1.86	0	18.8	3	8.55	6.1	0.31	40.49
s15850	Rewire	8697	7121	5	41.55	6.13	33.13	0.32	56.29	5	8.32	6.11	0.32	48.25
s38417	Cube	23915	20724	122	214.08	87.08	76.85	47.2	168.8	180	214.8	86.98	81.91	157.1
s35932	Cube	30938	23067	41	397.9	173.9	81.66	86.89	202.6	41	358.3	174	86.77	186.6
s35932	Rewire	30938	23045	1	188	173.2	15.1	0	60.6	1	193.4	173	11.27	96.84

VI. CONCLUSION

Most existing algorithms for engineering change orders can only achieve high solution quality for certain types of cases. In this paper, we propose a robust approach that can work for both structurally similar and different circuits. Our MSPM algorithm uses error models with SAT proof core minimization and is able to find a solution quickly if a simple fix exists. Circuits that only differ in simple rewiring or gate-type changes can be fixed with MSPM very efficiently. On the other hand, if the difference is too complex for MSPM, we then turn to incremental MSPM and the interpolation technique. This hybrid solution chooses candidate nodes and re-synthesizes their signals by interpolation generation. With FRAIG and a more sophisticated interpolation generation technique, our approach is quite robust and efficient.

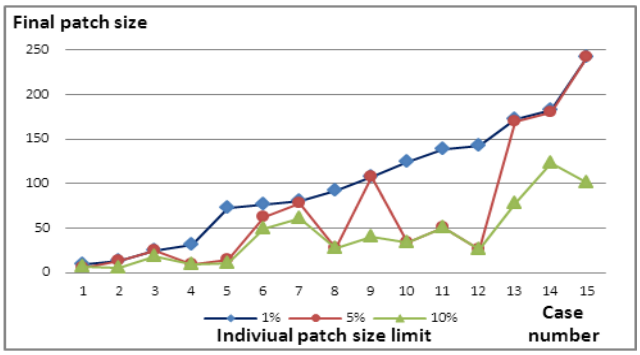


Figure 8. Final patch sizes with limitations on the different individual patch size.

ACKNOWLEDGEMENTS

This work is supported in part by National Science Council under grants NSC 99-2221-E-002 -211 -MY3 and 99-2221-E-002-214-MY3.

REFERENCES

[1] C.-C. Lin, K.-C. Chen, S.-C. Chang, M. Marek-Sadowska, and K.-T. Cheng, "Logic synthesis for engineering change," in Proc. Design Automation Conference, 1995, pp. 647-652.

[2] D. Hoffmann and T. Kropf, "Efficient design error correction of digital circuits," in Proc. International Conference on Computer Design, 2000, pp. 465-472.

[3] A. Veneris and I. Hajj, "A fast algorithm for locating and correcting simple design errors in VLSI digital circuits," in Proc. Great Lake Symposium on VLSI Design, 1997, pp. 45-50.

[4] S.-Y. Huang, K.-C. Chen and K.-T. Cheng, "Error correction based on verification techniques," in Proc. Design Automation Conference, 1996, pp. 258-261.

[5] A. Veneris and I. Hajj, "Design error diagnosis and correction via test vector simulation," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 18, pp. 1803-1816, 1999.

[6] K.-H. Chang, I.L. Markov and V. Bertacco, "Fixing design errors with counterexamples and resynthesis," in Proc. Asia and South Pacific Design Automation Conference, 2007, pp. 944-949.

[7] S. Krishnaswamy, H. Ren, N. Modi, and R. Puri, "DeltaSyn: an efficient logic difference optimizer for ECO synthesis." in Proc. International Conference on Computer-Aided Design, 2009, pp.789-796.

[8] A. Abdollahi and M. Pedram, "Symmetry Detection and Boolean Matching Utilizing a Signature-Based Canonical Form of Boolean Functions," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems vol. 27, no. 6, June, 2009

[9] A. Ling, J. Zhu, S. Brown and S. Safarpour, "Towards automated ECOs in FPGAs," in Proc. of International Symposium on Field Programmable Gate Arrays, 2009.

[10] C.-C. Lee, J.-H. R. Jiang, C.-Y. R. Huang, and A. Mishchenko, "Scalable exploration of functional dependency by interpolation and incremental SAT solving," in Proc. International Conference on Computer-Aided Design, 2007, pp. 227–233.

[11] V. Manohararajah, D. P. Singh, and S. D. Brown, "Post-placement BDD-based decomposition for FPGAs," in Proc. International Conference on Field Programmable Logic and Applications, 2005, pp. 31–38.

[12] A. Mishchenko, S. Chatterjee, R. Jiang and R. Brayton, "FRAIGs: A unifying representation for logic synthesis and verification," EECS Dept., UC Berkeley, Tech. Rep, 2005.

[13] C.-J. Hsu, S.-L. Huang, C.-A. Wu and C.-Y. R. Huang, "Interpolant Generation without Constructing Resolution Graph," in Proc. International Conference on Computer-Aided Design, 2009.

[14] K. L. McMillan, "Interpolation and SAT-based modelchecking," in Proc. International Conference on Computer Aided Verification, pp. 1–13, 2008.

[15] Berkeley Logic Synthesis and Verification Group. ABC: A System for Sequential Synthesis and Verification. http://www.eecs.berkeley.edu/~alanmi/abc/

[16] N. Sorensson and N. Een, "Minisat v1. 13-a sat solver with conflict-clause minimization," in SAT Competition, 2005.