A Halting Algorithm to Determine the Existence of the Decoder

ShengYu Shen, *Member, IEEE,* Ying Qin, LiQuan Xiao, KeFei Wang, JianMin Zhang, and SiKun Li

***Abstract*—Complementary synthesis automatically synthesizes the decoder circuit of an encoder. It determines the existence of the decoder by checking whether the encoder's input can be uniquely determined by its output. However, this algorithm will not halt if the decoder does not exist.**

To solve this problem, a novel halting algorithm is proposed. For every state sequence of the encoder, this algorithm first checks whether the encoder's input can be uniquely determined by its output. If yes, the decoder exists; Otherwise, this algorithm checks that if this state sequence contains loops, which can be further expanded to prove the non-existence of the decoder for all those longer state sequences.

To illustrate its usefulness and efficiency, this algorithm has been run on several complex encoders, including PCI-E and Ethernet. Experimental results indicate that this algorithm always halts properly by distinguishing correct encoders from the incorrect ones, and it can be more than three times faster than the previous work.

***Index Terms*—Halting Algorithm, Complementary Synthesis**

I. Introduction

One of the most difficult jobs in designing communication and multimedia chips is to design and verify the complex complementary circuit pair (E, E^{-1}), in which the encoder E transforms information into a format suitable for transmission and storage, while its complementary circuit(or decoder) E^{-1} recovers this information. In order to facilitate this job, the complementary synthesis algorithm is proposed in [1], [2] to automatically synthesize the decoder circuit of an encoder, by checking the Parameterized Complementary Condition(PC), that is, whether the encoder's input letter can be uniquely determined by its output sequence.

However, that algorithm will not halt if E^{-1} does not exist. Another algorithm has been proposed in [3] to solve this problem by first constructing a list of over-approximations of PC that is similar to onion rings, and then checking whether E is in all these rings. This algorithm is very slow and complicated. Therefore yet another halting algorithm is proposed in this paper, which is both faster and more straightforward. **For every state sequence of the encoder, this new algorithm checks two cases:**

1) **Just like the non-halting algorithm [1], [2], checking whether the encoder's input can be uniquely determined by its output. If yes, the decoder exists;**
2) **Otherwise, checking whether there are loops in the state sequence mentioned above. As shown in Figure 1, the state sequence in Figure 1a) contains three sub-sequences a, b and c. The sub-sequence b is a loop that can be further expanded to obtain a longer state sequence shown in Figure 1b). In this way, the non-existence of the decoder can be proved for all those longer state sequences.**

This new algorithm has been implemented with the OCaml language. All generated SAT instances are solved with Zchaff SAT solver [4]. The benchmark set includes several complex encoders from industrial projects (e.g., PCI-E [5] and Ethernet [6]), and their slightly modified variants without corresponding decoders. Experimental results indicate that this paper's algorithm always halts properly by distinguishing correct encoders from incorrect ones, and can be three times faster than the other halting algorithm [3] proposed by us in FMCAD'10. All these experimental results and programs can be downloaded from http://www.ssypub.org.

There is one more issue that needs to be clarified. There are two classes of complementary circuit pairs with different characters and design methodologies:

1) **Standard datapath-intensive circuits: These circuits often work as digital signal processing components, such as Fast Fourier Transform (FFT) and Discrete Cosine Transform(DCT). These circuits usually have standard and highly optimized implementations from various foundries and IP vendors, such as Xilinx core generator [7] and Synopsys DesignWare library [8]. Therefore this paper's algorithm isn't design for them.**
2) **Non-standard and control-intensive circuits: These circuits, such as PCI-E [5] and Ethernet [6], are often used to handle communication protocols, and typically don't have standard implementations. This**

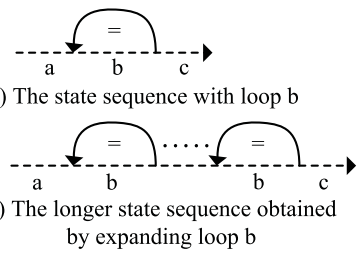

Fig. 1. Expanding the loop to prove the non-existence of the decoder for longer state sequence

The authors are with the School of Computer, National University of Defense Technology, ChangSha, HuNan, 410073 CHINA. e-mail: {syshen, qy123, lqxiao, kfwang, jmzhang, skli}@nudt.edu.cn, (see http://www.ssypub.org/).

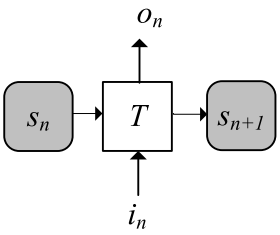

Fig. 2. Mealy finite state machine

paper's algorithm is designed for them.

The remainder of this paper is organized as follows. Section II presents background materials. Section III introduces this paper's algorithm, while Section IV describes how to remove redundant output letters to minimize the circuit area. Section V and VI present the experimental results and related works respectively. Finally, Section VII concludes this paper.

II. Preliminaries

A. Propositional Satisfiability

The Boolean value set is denoted as $B = \{0, 1\}$. For a Boolean formula F over a variable set V, the propositional satisfiability problem(abbreviated as SAT) is to find a satisfying assignment $A : V \to B$, so that F can be evaluated to 1. If such a satisfying assignment exists, then F is satisfiable; otherwise, it is unsatisfiable.

A computer program that decides the existence of such a satisfying assignment is called a SAT solver, such as Zchaff [4], Grasp [9], Berkmin [10], and MiniSAT [11]. A formula to be solved by a SAT solver is also called a SAT instance.

B. Recurrence Diameter

The encoder E can be modeled by a Mealy finite state machine [12].

Definition 1: **Mealy finite state machine** is a 5-tuple $M = (S, s_0, I, O, T)$, consisting of a finite state set S, an initial state $s_0 \in S$, a finite set of input letters I, a finite set of output letters O, a transition function $T : S \times I \to S \times O$ that computes the next state and the output letter from the current state and the input letter.

As shown in Figure 2, as well as in the remainder of this paper, the state is represented as a gray round corner box, and the transition function T is represented by a white rectangle. The state, the input letter and the output letter at the n-th cycle are denoted as s_n, i_n and o_n respectively. The sequence of state, input letter and output letter from the n-th to the m-th cycle are denoted as s_n^m, i_n^m and o_n^m respectively. **A loop is a state sequence s_n^m with $s_n \equiv s_m$. A loop-like path is a state sequence s_n^m with $s_i \equiv s_j$, where $n \leq i < j \leq m$.**

Kroening et al. [13] defined the **state variables recurrence diameter** of M, denoted by $rrd(M)$, as the longest state sequence that starts from an initial state and does not contain a loop.

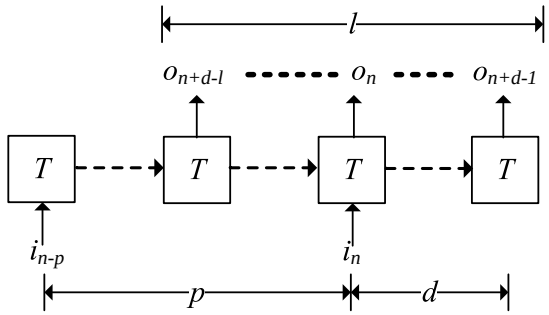

Fig. 3. The parameterized complementary condition

$$rrd(M) \stackrel{def}{=} \max\{i | \exists s_0 \dots s_i i_0 \dots i_i o_0 \dots o_i : I(s_0) \wedge \bigwedge_{j=0}^{i-1}(s_{j+1}, o_j) \equiv T(s_j, i_j) \wedge \bigwedge_{j=0}^{i-1}\bigwedge_{k=j+1}^{i} s_j \neq s_k\} \quad (1)$$

In this paper, a similar concept: the **uninitialized state variables recurrence diameter** of M, denoted by $uirrd(M)$, is defined as the longest state sequence without loop.

$$uirrd(M) \stackrel{def}{=} \max\{i | \exists s_0 \dots s_i i_0 \dots i_i o_0 \dots o_i : \bigwedge_{j=0}^{i-1}(s_{j+1}, o_j) \equiv T(s_j, i_j) \wedge \bigwedge_{j=0}^{i-1}\bigwedge_{k=j+1}^{i} s_j \neq s_k\} \quad (2)$$

The only difference between these two definitions is that $uirrd$ does not consider the initial state. **These definitions are only used in proving the theorems below. This paper's algorithm does not need to compute these diameters.**

C. The Original Non-halting Algorithm to Determine the Existence of the Decoder

The complementary synthesis algorithm [1] includes two steps: determining the existence of the decoder and characterizing its Boolean function. Only the first step is introduced here.

According to our previous research [1]–[3], and our engineering experience in designing communication chips for the TH-1A supercomputer [14], many communication protocols are lossless, that is, every input letter can be recovered from its output sequence.

More formally, as shown in Figure 3, a sufficient condition for the existence of the decoder E^{-1} is, there exist three parameter values p, d and l, so that i_n of the encoder E can be uniquely determined by E's output sequence o_{n+d-l}^{n+d-1}. d is the relative delay between o_{n+d-l}^{n+d-1} and the input letter i_n, while l is the length of o_{n+d-l}^{n+d-1}, and p is the length of the prefix state sequence used to rule out some unreachable states of the encoder. Thus, the parameterized complementary condition(PC) [1] can be formally defined as:

Definition 2: **Parameterized Complementary Condition (PC) :** For encoder E, $E \vDash PC(p, d, l)$ holds if i_n can be uniquely determined by o_{n+d-l}^{n+d-1} in the state sequence s_{n-p}^{n+d-1}.

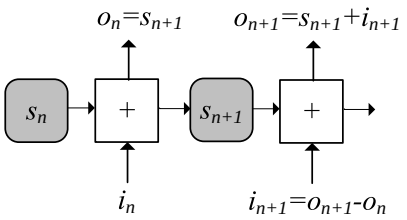

Fig. 4. The circuit that breaks the causal relation

This equals the unsatisfiability of $F_{PC}(p,d,l)$ in Equation (3). $E \vDash PC$ is further defined as $\exists p,d,l : E \vDash PC(p,d,l)$.

$$F_{PC}(p,d,l) \stackrel{def}{=} \left\{ \begin{array}{cc} & \bigwedge_{m=n-p}^{n+d-1}\{(s_{m+1},o_m) \equiv T(s_m,i_m)\} \\ \wedge & \bigwedge_{m=n-p}^{n+d-1}\{(s'_{m+1},o'_m) \equiv T(s'_m,i'_m)\} \\ \wedge & \bigwedge_{m=n+d-l}^{n+d-1} o_m \equiv o'_m \\ \wedge & i_n \neq i'_n \end{array} \right\} \quad (3)$$

The 2nd and 3rd lines of Equation (3) correspond respectively to two state sequences of the encoder. The only difference between them is that a prime is appended to every variable in the 3rd line. The 4th line forces the output sequences of these two state sequences to be the same, while the 5th line forces their input letters to be different.

This non-halting algorithm [1] just iterate through all valuations of p,d and l, from small to large, until one valuation of p,d and l that makes Equation (3) unsatisfiable is found. Then the existence of the decoder is proved. However, if the decoder does not exist, this algorithm will never halt. This problem will be solved in the next section.

There is one more problem to be explained. According to Figure 3, if $l > d$, then, to compute the value of input i_n, one may need to know the output o_m where $m < n$. It looks like breaking the causal relation. This problem will be explained with the circuit in Figure 4. Assume its transition function T is:

$$\begin{array}{c} s_{n+1} = i_n + s_n \\ o_n = i_n + s_n \end{array} \quad (4)$$

Intuitively, this circuit adds its input to its current state, and puts the result to its output and next state. So, to recover the input letter i_{n+1}, the output letter o_n must be subtracted from o_{n+1}. In this case, l is 1, and d is 0. This explains why l can be larger than d.

III. A Halting Algorithm to Determine the Existence of the Decoder

A. Determining the Non-existence of the Decoder

PC in Definition 2 only defines how to determine the existence of the decoder E^{-1}. But how to determine the non-existence of E^{-1} is left undefined. So the key to a halting algorithm is to find out a necessary and sufficient condition for the non-existence of E^{-1}.

According to Definition 2 and Figure 3, E^{-1} exists if there is a parameter value tuple $< p,d,l >$, such that $E \vDash PC(p,d,l)$ holds. So, intuitively, E^{-1} does not exist if for every parameter value tuple $< p,d,l >$, another tuple $< p',d',l' >$ with $p' > p$,$l' > l$ and $d' > d$ can always be found, such that $E \vDash PC(p',d',l')$ does not hold.

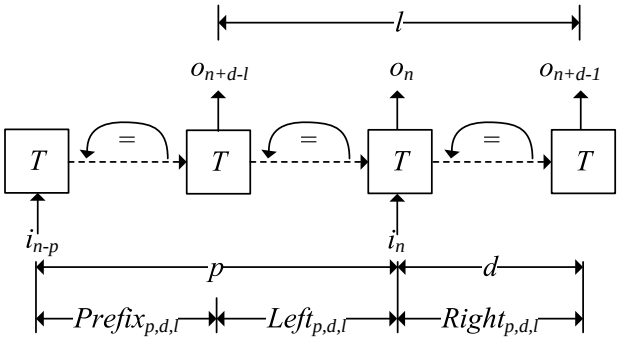

Fig. 5. The loop-like non-complementary condition

This case can be detected by the SAT instance in Figure 5, which is similar to Figure 3, except that three new constraints are inserted to detect loops in state sequences s_{n-p}^{n+d-l},$s_{n+d-l+1}^{n}$ and s_{n+1}^{n+d}. **If this SAT instance is satisfiable for the parameter value $< p,d,l >$, these three loops can be expanded in the following way: Assume that the length of loops in s_{n-p}^{n+d-l},$s_{n+d-l+1}^{n}$ and s_{n+1}^{n+d} are l_1, l_2 and l_3 respectively, and that these loops are expanded for q times. Then, the SAT instance generated from this expanding is shown in Figure 6. This expanded SAT instance corresponds to $F_{LN}(p",d",l")$, where:**

$$\begin{array}{rcc} p" & = & l_1 * q + (d - l + p - l_1) + l_2 * q + (l - d - l_2) \\ d" & = & l_3 * q + (d - l_3) \\ l" & = & l_2 * q + (l - d - l_2) + l_3 * q + (d - l_3) \end{array} \quad (5)$$

It is obvious that for every particular $< p,d,l >$ and $< p',d',l' >$, there always exists a q, such that $s_{n-p"}^{n+d"-l"}$,$s_{n+d"-l"+1}^{n}$ and $s_{n+1}^{n+d"}$ resulted from this expanding are not shorter than $s_{n-p'}^{n+d'-l'}$,$s_{n+d'-l'+1}^{n}$ and $s_{n+1}^{n+d'}$ respectively. The satisfiability of this expanded instance will be proved in Lemma 1 in next subsection. This means for every particular $< p',d',l' >$, we can always find another $< p",d",l" >$, such that $E \vDash PC(p",d",l")$ does not hold. So the decoder does not exist.

According to the 2nd and 3rd lines of Equation (3), there are actually two state sequences, so these loops must be detected in both of them, i.e., on the productive machine M^2 defined below:

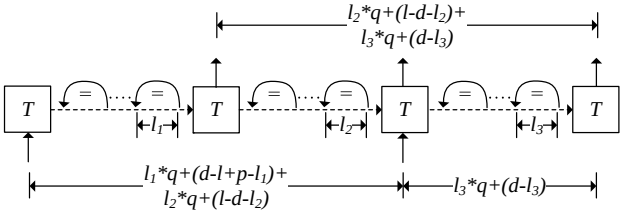

Fig. 6. The loop-like non-complementary condition expanded for q times

***Definition** 3:* **Productive machine :** For Mealy machine $M = (S, s_0, I, O, T)$, its Productive machine is $M^2 = (S^2, s_0^2, I^2, O^2, T^2)$, where

1) $S^2 = S \times S$
2) $s_0^2 = s_0 \times s_0$
3) $I^2 = I \times I$
4) $O^2 = O \times O$
5) T^2 is defined as $(< s_{m+1}, s'_{m+1} >, < o_m, o'_m >) = T^2(< s_m, s'_m >, < i_m, i'_m >)$ with $(s_{m+1}, o_m) = T(s_m, i_m)$ and $(s'_{m+1}, o'_m) = T(s'_m, i'_m)$.

Thus, the loop-like non-complementary condition is formally defined below to determine the non-existence of E^{-1}:

***Definition** 4:* **Loop-like Non-complementary Condition (LN) :** For encoder E and its Mealy machine $M = (S, s_0, I, O, T)$, assume its productive machine is $M^2 = (S^2, s_0^2, I^2, O^2, T^2)$, then $E \vDash LN(p, d, l)$ holds if i_n can not be uniquely determined by o_{n+d-l}^{n+d-1} in the state sequence s_{n-p}^{n+d-1}, and there are loops in $(s^2)_{n-p}^{n+d-l}$,$(s^2)_{n+d-l+1}^{n}$ and $(s^2)_{n+1}^{n+d}$. This equals the satisfiability of $F_{LN}(p, d, l)$ in Equation (6). $E \vDash LN$ is further defined as $\exists p, d, l : E \vDash LN(p, d, l)$.

$$F_{LN}(p, d, l) \stackrel{def}{=}$$
$$\left\{ \begin{array}{ll} & \bigwedge_{m=n-p}^{n+d-1}\{(s_{m+1}, o_m) \equiv T(s_m, i_m)\} \\ \wedge & \bigwedge_{m=n-p}^{n+d-1}\{(s'_{m+1}, o'_m) \equiv T(s'_m, i'_m)\} \\ \wedge & \bigwedge_{m=n+d-l}^{n+d-1} o_m \equiv o'_m \\ \wedge & i_n \neq i'_n \\ \wedge & \bigvee_{x=n-p}^{n+d-l-1} \bigvee_{y=x+1}^{n+d-l}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \\ \wedge & \bigvee_{x=n+d-l+1}^{n-1} \bigvee_{y=x+1}^{n}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \\ \wedge & \bigvee_{x=n+1}^{n+d-1} \bigvee_{y=x+1}^{n+d}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \end{array} \right\} \quad (6)$$

By comparing Equations (3) and (6), it is obvious that their only difference is the last three newly inserted lines in (6), which will be used to detect loops in the following three state sequences:

$$\begin{array}{rcl} Prefix_{p,d,l} & = & (s^2)_{n-p}^{n+d-l} \\ Left_{p,d,l} & = & (s^2)_{n+d-l+1}^{n} \\ Right_{p,d,l} & = & (s^2)_{n+1}^{n+d} \end{array} \quad (7)$$

The correctness of this approach will be proved in the next subsection.

B. Proving Correctness

Before proving correctness of this approach, some lemmas are needed.

***Lemma** 1 ():* For $F_{LN}(p", d", l")$ in Figure 6, $E \vDash LN(p, d, l)$ implies $E \vDash LN(p", d", l")$.

Proof: The formula $F_{LN}(p", d", l")$ is:

$$F_{LN}(p", d", l") \stackrel{def}{=}$$
$$\left\{ \begin{array}{ll} & \bigwedge_{m=n-p"}^{n+d"-1}\{(s_{m+1}, o_m) \equiv T(s_m, i_m)\} \\ \wedge & \bigwedge_{m=n-p"}^{n+d"-1}\{(s'_{m+1}, o'_m) \equiv T(s'_m, i'_m)\} \\ \wedge & \bigwedge_{m=n+d"-l"}^{n+d"-1} o_m \equiv o'_m \\ \wedge & i_n \neq i'_n \\ \wedge & \bigvee_{x=n-p"}^{n+d"-l"-1} \bigvee_{y=x+1}^{n+d"-l"}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \\ \wedge & \bigvee_{x=n+d"-l"+1}^{n-1} \bigvee_{y=x+1}^{n}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \\ \wedge & \bigvee_{x=n+1}^{n+d"-1} \bigvee_{y=x+1}^{n+d"}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \end{array} \right\} \quad (8)$$

$E \vDash LN(p, d, l)$ means that $F_{LN}(p, d, l)$ is satisfied. Assume its satisfying assignment is A. The directed arcs numbered from 1 to 12 in Figure 7 show the correspondence between $F_{LN}(p, d, l)$ and $F_{LN}(p", d", l")$.

The arcs 2 and 3 mean applying the satisfying assignment of the loop in $Right_{p,d,l}$ to the expanded loops in $Right_{p",d",l"}$. The arcs 1 and 4 mean applying the satisfying assignments of the two state sequences that is not in the loop, to $Right_{p",d",l"}$. With the arcs 1,2,3 and 4, the state sequence $Right_{p",d",l"}$ is satisfied.

Similarly, the state sequences $Prefix_{p",d",l"}$ and $Left_{p",d",l"}$ can also be satisfied with A. Thus the 2nd line of Equation (8) is satisfied with the assignment A.

Similarly, the 3rd to 5th lines of Equation (8) are also satisfied with the assignment A.

At the same time, there are q loops in $Prefix_{p",d",l"}$, $Left_{p",d",l"}$ and $Right_{p",d",l"}$, which will make the last three lines in Equation (8) satisfied.

Thus,the satisfying assignment A of $F_{LN}(p, d, l)$ can also make $F_{LN}(p", d", l")$ satisfied. This concludes the proof. ■

***Lemma** 2 ():* For two tuples $< p, d, l >$ and $< p', d', l' >$, if $Prefix_{p',d',l'}$,$Left_{p',d',l'}$ and $Right_{p',d',l'}$ are **not shorter** than $Prefix_{p,d,l}$,$Left_{p,d,l}$ and $Right_{p,d,l}$ respectively, then $E \vDash PC(p, d, l) \rightarrow E \vDash PC(p', d', l')$.

Proof: It is obvious that $F_{PC}(p, d, l)$ is a sub-formula of $F_{PC}(p', d', l')$, so the unsatisfiability of the former one implies the unsatisfiability of the latter one. Thus, $E \vDash PC(p, d, l) \rightarrow E \vDash PC(p', d', l')$ holds. ■

The following two theorems will prove that $E \vDash LN \leftrightarrow \neg\{E \vDash PC\}$.

***Theorem** 1 ():* $E \vDash LN \rightarrow \neg\{E \vDash PC\}$

Proof: This can be proved by contradiction. Assume that $E \vDash LN$ and $E \vDash PC$ both hold. This means there exist

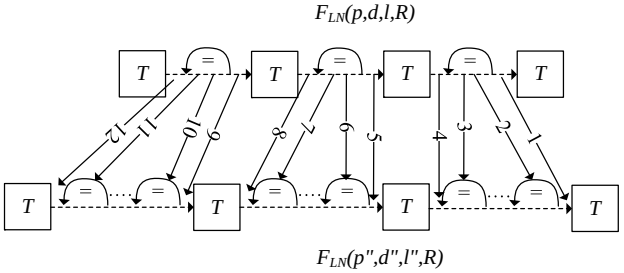

Fig. 7. Correspondence between $F_{LN}(p, d, l)$ and $F_{LN}(p", d", l")$

$< p, d, l >$ and $< p', d', l' >$, such that $E \vDash PC(p, d, l)$ and $E \vDash LN(p', d', l')$.

On the one hand, $E \vDash LN(p', d', l')$ implies that there are loops in the state sequences $Prefix_{p',d',l'}$,$Left_{p',d',l'}$ and $Right_{p',d',l'}$. By expanding these loops, another tuple $< p", d", l" >$ can be found, so that :

1) $Prefix_{p",d",l"}$,$Left_{p",d",l"}$ and $Right_{p",d",l"}$ are not shorter than $Prefix_{p,d,l}$,$Left_{p,d,l}$ and $Right_{p,d,l}$ respectively;
2) According to Lemma 1, $F_{LN}(p", d", l")$ is satisfiable.

$F_{PC}(p", d", l")$ is a sub-formula of $F_{LN}(p", d", l")$, so $F_{PC}(p", d", l")$ is also satisfiable, which means that $E \vDash PC(p", d", l")$ does not hold.

On the other hand, according to Lemma 2, $E \vDash PC(p", d", l")$ holds.

This contradiction concludes the proof. ■

***Theorem** 2 ():* $E \vDash LN \leftarrow \neg\{E \vDash PC\}$

Proof: Let's prove it by contradiction. Assume that neither $E \vDash LN$ nor $E \vDash PC$ holds. Then for every $< p, d, l >$ and $< p', d', l' >$, $F_{PC}(p, d, l)$ is satisfiable, while $F_{LN}(p', d', l')$ is unsatisfiable.

Thus, assume the $uirrd(M^2)$ is the uninitialized state variables recurrence diameter of E's productive machine. Let's define $< p, d, l >$ as:

$$\begin{aligned} p &= uirrd(M^2) * 2 + 2 \\ d &= uirrd(M^2) + 1 \\ l &= uirrd(M^2) * 2 + 2 \end{aligned} \tag{9}$$

With this definition, it is obvious that $Prefix_{p,d,l}$,$Left_{p,d,l}$ and $Right_{p,d,l}$ are both longer than $uirrd(M^2)$. This means that there are loops in all these three state sequences, which will make $F_{LN}(p, d, l)$ satisfiable. This contradicts with the fact that $F_{LN}(p', d', l')$ is unsatisfiable for every $< p', d', l' >$.

This contradiction concludes the proof. ■

Theorems 1 and 2 illustrate that, a halting algorithm can be implemented by enumerating all combinations of $< p, d, l >$ from small to large, and checking $E \vDash PC(p, d, l)$ and $E \vDash LN(p, d, l)$ in every iteration. This process will eventually terminate with one and only one answer between $E \vDash PC$ and $E \vDash LN$. The implementation of this algorithm will be presented in the next subsection.

C. Algorithm Implementation

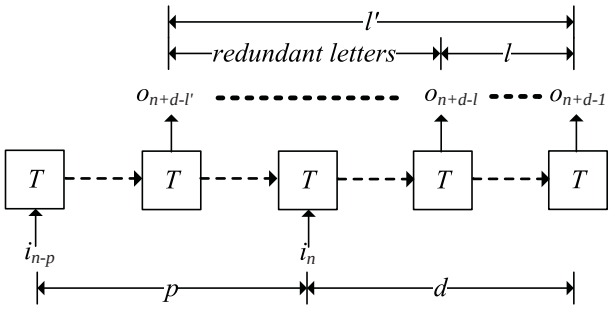

Fig. 8. The Redundant Output Letters

Algorithm 1 $check\_PCLN$

```
1: for x = 1 → ∞ do
2:    p = 2x
3:    d = x
4:    l = 2x
5:    if F_PC(p, d, l) is unsatisfiable then
6:       print "E^-1 exists with < p, d, l >"
7:       halt;
8:    else if F_LN(p, d, l) is satisfiable then
9:       print "E^-1 does not exist"
10:      halt;
11:   end if
12: end for
```

Instead of enumerating all combinations of parameter value p, d and l like [1], [2], Line 2,3 and 4 of Algorithm 1 ensure that the lengths of $Prefix_{p,d,l}$,$Left_{p,d,l}$ and $Right_{p,d,l}$ are all set to x, which is enumerated at Line 1. In this way, the run time overhead is further reduced because many redundant combinations do not need to be tested any more.

According to Theorems 1 and 2, Algorithm 1 will eventually terminate at line 6 or 9.

IV. Removing Redundant Output Letters

Algorithm 2 $RemoveRedundancy(p, d, l)$

```
1: for p' = p → 0 do
2:    if F_PC(p' − 1, d, l) is satisfiable then
3:       break
4:    end if
5: end for
6: for d' = d → 0 do
7:    if F_PC(p', d' − 1, l) is satisfiable then
8:       break
9:    end if
10: end for
11: for l' = 1 → l − (d − d') do
12:    if F_PC(p', d', l') is unsatisfiable then
13:       break
14:    end if
15: end for
16: print "final result is < p', d', l' >"
```

Although Algorithm 1 is sufficient to determine the existence of E^{-1}, the parameter values found by line 6 of Algorithm 1 contain some redundancy, which will cause unnecessarily large overheads on the circuit area and the run time of characterizing.

For example, as shown in Figure 8, assume that l is the smallest parameter value that leads to $E \vDash PC(p, d, l)$, and $l < d$, which means that i_n is uniquely determined by some output letters o_k with $k > n$. Further assume that line 6 of Algorithm 1 prove $E \vDash PC(p, d, l')$. It is obvious that $l' > d$, which makes i_n depend on some redundant o_k with $k \leq n$. So $o_{n+d-l'}^{n+d-l-1}$ is the sequence of redundant output letters, which

TABLE I
INFORMATION OF BENCHMARKS

	XGXS	XFI	scrambler	PCI-E	T2 ethernet
Line number of verilog source code	214	466	24	1139	1073
#regs	15	135	58	22	48
Data path width	8	64	66	10	10

should be removed to prevent them from being instantiated as latches in circuit E^{-1}. At the same time, also according to Figure 8, larger p and d lead to larger SAT instances for the characterization algorithm in the 2nd step of complementary synthesis.

So, Algorithm 2 is used to minimize $< p, d, l >$ before passing it to the characterization algorithm:

V. EXPERIMENTAL RESULTS

This algorithm has been implemented with the OCaml language. The generated SAT instances are solved with Zchaff SAT solver [4]. All experiments are run on a PC with a 2.4GHz Intel Core 2 Q6600 processor, 8GB memory and CentOS 5.2 linux. All these experimental results and programs can be downloaded from http://www.ssypub.org.

A. Benchmarks

Table I shows the information of the following benchmarks.

1) A XGXS encoder that is compliant to clause 48 of IEEE-802.3ae 2002 standard [6].
2) A XFI encoder that is compliant to clause 49 of the same IEEE standard.
3) A 66-bit scrambler that is used to ensure that a data sequence has sufficiently many 0-1 transitions , so that it can run through a high-speed noisy serial transmission channel.
4) A PCI-E physical coding module [5].
5) The Ethernet module of Sun's OpenSparc T2 processor.

B. Determining the existence of the decoder for Properly Designed Encoders

The 2nd and 4th rows of Table II compare the run time of checking $E \models PC$ between [3] and this paper. It is obvious that this paper's approach is much faster than that of [3].

The 3rd and 5th rows compare the discovered parameter values, and some minor differences are found on parameter value p. This is caused by the different orders in checking various parameter combinations.

TABLE II
EXPERIMENTAL RESULTS ON PROPERLY DESIGNED ENCODERS

		XGXS	XFI	scrambler	PCI-E	T2 ethernet
[3]	Time to check PC(sec)	1.06	70.52	5.74	2.40	66.37
	d, p, l	1,1,1	0,3,2	0,2,2	2,1,1	4,1,1
This paper	Time to check PC(sec)	0.29	17.86	2.67	0.47	29.64
	d, p, l	1,2,1	0,3,2	0,2,2	2,2,1	4,4,1

TABLE III
COMPARING DECODER AREA

	XGXS	XFI	scrambler	PCI-E	T2 ethernet
The decoders built manually	921	6002	1629	852	1446
The decoders built by this paper's algorithm	700	12754	1455	455	552

C. Comparing Decoder Area

Table III compares the circuit area of the decoders built manually, and the decoders built by this paper's algorithm. These decoders are synthesized with LSI10K technology library from Synopsys DesignCompiler.

From Table III, it is obvious that, except for the most complex XFI, synthesis results of this paper's algorithm are more compact than those decoders built manually. However, this comparison is unfair because those decoders built manually include additional functionality, such as testing logic.

On the other hand, for XFI, the circuit area of this paper's algorithm is about 2 times larger. This means the circuit area must be improved in the future work.

D. Comparing Decoder Timing

Table IV compares the critical path latencies of the decoders built manually and the decoders built by this paper's algorithm. Their synthesis settings are the same as Subsection V-C. For all those circuits, the critical path latencies of the decoders built by this paper's algorithm are all better.

E. Determining the non-existence of the decoder for Improperly Designed Encoders

To further show the usefulness of this paper's algorithm, some improperly designed encoders without corresponding decoders are needed. These improperly designed encoders are obtained by modifying each benchmark's output statements, so that they can explicitly output the same letter for two different input letters. In this way, input letter i_n can never be uniquely determined by E's output sequence.

The 2nd row of Table V shows the run time of [3] on checking these improperly designed encoders, while the 3rd row shows the run time of this paper's algorithm. These results indicate that this paper's algorithm always terminate correctly, and is much faster than [3].

TABLE IV
COMPARING CRITICAL PATH LATENCIES IN NANOSECOND

	XGXS	XFI	scrambler	PCI-E	T2 ethernet
The decoders built manually	12.33	46.65	6.54	19.03	23.36
The decoders built by this paper's algorithm	11.96	28.13	6.54	9.09	12.69

TABLE V
COMPARING RUN TIME OF IMPROPERLY DESIGNED ENCODERS

	XGXS	XFI	scrambler	PCI-E	T2 ethernet
The algorithm of [3](sec)	0.98	35.08	2.54	1.36	17.39
This paper's algorithm(sec)	0.16	7.59	1.17	0.33	2.19

VI. RELATED WORKS

A. Complementary Synthesis

The concept of complementary synthesis was first proposed by us [1] in ICCAD 2009. Its major shortcomings are that it is not halting, and its run-time overhead of building complementary circuit is too large.

The halting problem was handled by building a set of over-approximations that are similar to onion rings [3], while the run-time overhead problem was addressed by simplifying the SAT instance with unsatisfiable core extraction [2].

B. Program Inverse

According to Gulwani [17], program inverse is the problem that derives a program P^{-1} that negates the computation of a given program P. So the definition of program inverse is very similar to complementary synthesis.

The initial work on deriving program inverse used proof-based approaches [18], but it could only handle very small programs and very simple syntax structures.

Glück et.al [19] inversed the first-order functional programs by eliminating nondeterminism with LR-based parsing methods. The requirement that the program to be inversed should be expressed in a functional language make it impossible to be applied to complementary synthesis.

Srivastava et.al [20] assumed that an inverse program was typically related to the original program, so the space of possible inverses can be inferred by automatically mining the original program for expressions, predicates, and control flow. This algorithm inductively rules out invalid paths that can't fulfill the requirement of inversion, to narrow down the space of candidate programs until only the valid ones remain. So it can only guarantee the existence of a solution, but not the correctness of this solution if its assumptions do not hold.

C. The Completeness of Bounded Model Checking

Bounded model checking(BMC) [15] is a model checking technology that considers only the paths of limited length. So it is an incomplete algorithm. Many researchers try to find out complete approaches for BMC.

One line of research [13], [15] tried to find out a bound b, which can guarantee the correctness of a specification, if the specification is correct on all paths that are shorter than b. **Line 8 of Algorithm 1 finds out the value of p,d and l that can prove the non-existence of the decoder. This is similar to these related researches.**

The other line of research [16] tried to find out a pattern for induction, such that the correctness of a specification within any bound b implies the correctness on bound $b+1$. **Our algorithm proves the non-existence of the decoder by expanding loops. This is similar to finding induction pattern in this related research.**

D. Protocol Converter Synthesis

The protocol converter synthesis is the problem that automatically generates a translator between two different communication protocols. **This is related to our works because both of us focus on synthesizing communication circuits.**

Avnit et.al [21], [22] first defined a general model for describing the different protocols. Then it provided an algorithm to decide whether there are some functionality of a protocol that can't be translated into another. Finally, it synthesized the translator by computing a greatest fixed point for the update function of the buffer's control states. Avnit et.al [23] improved the algorithm mentioned above with a more efficient design space exploration algorithm.

VII. CONCLUSIONS

This paper proposes a faster and more straightforward halting algorithm that checks whether a particular encoder has its corresponding decoder. The theoretical analysis shows that this paper's approach always distinguishes correct encoders from their incorrect variants and halts properly. Experimental results show that this paper's approach is much faster than that of [3].

ACKNOWLEDGMENT

The authors would like to thank the editors and anonymous reviewers for their hard work.

This work was funded by projects 60603088 and 61070132 supported by National Natural Science Foundation of China.

REFERENCES

[1] S. Shen, J. Zhang, Y. Qin, and S. Li, "Synthesizing complementary circuits automatically," in Proc. IEEE ICCAD, Nov. 2009, pp. 381-388.
[2] S. Shen, Y. Qin, K. Wang, L. Xiao, J. Zhang, and S. Li, "Synthesizing Complementary Circuits Automatically," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 29, no. 8, pp. 1191-1202, Aug. 2010.
[3] S. Shen, Y. Qin, J. Zhang, and S. Li, "A Halting Algorithm to Determine the Existence of Decoder," in Proc. IEEE FMCAD, Oct. 2010, pp. 91-100.
[4] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in Proc. IEEE Int. DAC, Jun. 2001, pp. 530-535.
[5] "PCI Express Base Specification Revision 1.0". [Online]. Available: http://www.pcisig.com
[6] "IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gb/s Operation", IEEE Std. 802.3, 2002.
[7] Xilinx. (2008) Xilinx core generator system. [Online]. Available: http://www.xilinx.com/tools/coregen.htm
[8] Synopsys. (2008) Designware library. [Online]. Available: http://www.synopsys.com/IP/DESIGNWARE/Pages/default.aspx

[9] J. P. M. Silva and K. A. Sakallah, "GRASP - a new search algorithm for satisfiability," in Proc. IEEE ICCAD, Nov. 1996, pp. 220-227.
[10] E. Goldberg and Y. Novikov, "BerkMin: A fast and robust Sat-solver," Discrete Applied Mathematics, vol. 155, no. 12, pp. 1549-1561, Dec. 2007.
[11] N. Eén and N. Sörensson., "Extensible SAT-solver," in Proc. Int. Conf. SAT, May 2003, pp. 502-518.
[12] G. H. Mealy , "A method for synthesizing sequential circuits," Bell Syst. Tech. J., vol. 34, pp. 1045-1079, Jan. 1955.
[13] D. Kroening and O. Strichman, "Efficient Computation of Recurrence Diameters," in Proc. Int. Conf. VMCAI, Jan. 2003, pp. 298-309.
[14] "China Grabs Supercomputing Leadership Spot in Latest Ranking of World's Top 500 Supercomputers". [Online]. Available: http://www.top500.org/lists/2010/11/press-release.
[15] E. M. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded Model Checking Using Satisfiability Solving," Formal Methods in System Design, vol. 19, no. 1, pp. 7-34, Jan. 2001.
[16] M. Sheeran, S. Singh, and G. Stalmarck, "Checking Safety Properties Using Induction and a SAT-Solver," in Proc. IEEE FMCAD, Oct. 2000, pp. 108-125.
[17] S. Gulwani, "Dimensions in program synthesis," in Proc. ACM PPDP, Jul. 2010, pp. 13-24.
[18] E. W. Dijkstra, "Program Inversion," in Program Construction, 1978, pp. 54 - 57.
[19] R. Glück and M. Kawabe, "A method for automatic program inversion based on LR(0) parsing," Fundam. Inf., vol. 66, no. 4, pp. 367-395, Dec. 2005.
[20] S. Srivastava, S. Gulwani, S. Chaudhuri, and J. Foster, "Program inversion revisited," Technical Report MSR-TR-2010-34, Microsoft Research, 2010.
[21] K. Avnit, V. D'Silva, A. Sowmya, S. Ramesh, and S. Parameswaran, "A Formal Approach To The Protocol Converter Problem," in Proc. IEEE DATE Conf. Exposit., Mar. 2008, pp. 294-299.
[22] K. Avnit, V. D'Silva, A. Sowmya, S. Ramesh, and S. Parameswaran, "Provably correct on-chip communication: A formal approach to automatic protocol converter synthesis," ACM Trans. Design Autom. Electr. Syst., vol. 14, no. 2, pp. 1-41, Apr. 2009.
[23] K. Avnit, and A. Sowmya, "A formal approach to design space exploration of protocol converters," in Proc. IEEE DATE Conf. Exposit., Mar. 2009, pp. 129-134.
[24] K. Avnit, A. Sowmya, and J. Peddersen, "ACS: Automatic Converter Synthesis for SoC Bus Protocols," in Proc. Int. Conf. TACAS, Mar. 2010, pp. 343-348.

VIII. Response to reviewers

I will respond to every review one by one. For every review, my response and corresponding modification will be in **bold font**.

A. Review Number 1

1) This is a journal paper. I think providing some background information would help understand the motivation better.

Yes, you are right.

I add the 1st paragraph in Section I to provide background and motivation.

Could you elaborate a bit on why the protocols you consider have to be lossless? (An informal explanation will do.) I am not a specialist in the area of data transmission but as far as I understand loss of information is quite common in data transmission.

Yes, you are right.

I add the 2nd paragraph in subsection II-C to explain why these protocols are lossless.

2) In a previous paper, you mentioned that your approach does not work for data-intensive protocols. It makes sense to repeat this statement here (unless you have reconsidered it).

Yes, you are right. I add the 6th, 7th and 8th paragraphs in Section I to mention this point.

3) Why can't one build a decoder manually?

Yes, you are right. The decoder can be built manually. And as mentioned in the 1st paragraph in Section I, this is a tedious and error-prone job. That is why we propose complementary synthesis.

If it is possible, could you say a few words about how the quality of decoders built manually compares with that of decoders generated automatically?

I add Subsections V-C and V-D to compare the quality of decoders built manually and generated automatically.

4) What is a loop-like path? Is it a reconvergent path?

I add a sentence at the end of the 3rd paragraph in Subsection II-B to explain the concept of loop-like path.

A more general remark is that the paper lacks informal explanations. Intuitively, a Mealy machine M is lossless if for every pair of inputs i and i' M either a) produces different outputs o and o' or it b) switches into different states s and s' such that any two sequences starting in s and s' and converging to a state $s*$ have to produce different outputs in state $s*$.

I would like to see more informal explanations like that.

To explain my idea informally, I rewrite the 2nd paragraph of abstraction, and add the the 3rd and 4th paragraphs and Figure 1 in Section I.

For more informal explanations, please refer to the 3rd and 4th paragraphs of subsection III-A. Please also refer to Lemma 1.

5) Here an example of unnecessary complicating things. In expression (1) describing the recurrent diameter, instead of just saying that the state are different pairwise i.e. that ($s_j \neq s_k$) for all $j, k = 1, .., i, j \neq k$ you give a more complex expression. In this expression, the indexes are chosen in such a way that the check ($s_j \neq s_k$) is performed only once. It would make sense if you were describing an algorithm for finding the recurrence diameter. But you are giving a definition.

Actually, according to the last paragraph of Subsection II-B, we don't need to compute these diameters in our algorithms. We only need them in proving our theorems.

6) In Figure 2, if $l > d$, then, to compute the value of input i_n, one may need to know the output o_m where $m < n$. It looks like breaking the causal relation. Could you give an informal explanation of why it may be necessary?

Yes, you are right. I add the last two paragraphs in Subsection II-C to explain this.

7) The miter of your definition 3 is not what people usually call a miter. In your miter, the number of input variables is

doubled, while in a regular miter the input variables of two circuits are identified. It is worth mentioning this fact. An alternative is to use a different name, like ”pseudo-miter” or ”generalized miter”.

Yes, you are right. I change it to ”productive machine”.

8) The experimental results are not terribly impressive. Your best result is reducing the runtime from 70 sec. to 21 sec. on XFI. If you found an example where you reduced time from 10 hours to 3 hours it would make more sense. Waiting for 70 sec. instead of 21 sec. is tolerable, unless you can argue that in some scenarios your algorithm may be applied many times.

Yes, you are right. I handle this problem in the following ways:

First,I further improve the performance of Algorithm 1. Please refer to the first paragraph of subsection III-C, and please also refer to experimental results in subsection V-B and V-E.

Second, in using the complementary synthesis algorithm, the user need to manually specify an assertion on some configuration pins, to prevent the encoder from reaching some non-working states, such as testing and sleep states. These assertions are implementation specific, and can not be found in standard documentations. So the user often need to try many combinations before successfully finding a correct one. Thus, our algorithm are indeed invoked many times. So boosting its performance can significantly reduce the trouble of the users.

Finally, in our recent new research, we try to infer these assertions automatically. In that new algorithm, this paper’s algorithm is invoked for hundreds of times, and results in thousands of seconds of run time. So boosting performance of this paper’s algorithm can significantly reduce the overall run time overhead of inferring assertions.

B. Review Number 2

****

Comments to the Author ————————-

The paper presents incremental work on earlier conference papers, and in fact is written in conference style.

There are major lacunae in the presentation:

1. The introduction to the problem is inadequate. Why is synthesis of a decoder an important problem?

Yes, you are right. I add the first paragraph in Section I to solve this problem.

What are the existing techniques to tackle this problem?

Please refer to Subsection VI-B. It introduces a similar problem, Program Inverse. We also explain why their approaches can’t be applied to solve our problem.

How is your proposed solution an improvement?

According to Section I, our approach is faster and more straightforward when compared with our FMCAD’10 paper [3].

All comparisons, including in the experiments, are to your own previous work, is there no other work in this area?

Yes, complementary synthesis is proposed by us at ICCAD’09, and now we are the only group that work on this topic.

2. The approach should be better motivated,

Yes, you are right. I add the first paragraph in Section I to tackle this problem.

and the algorithms more intuitively explained. Spewing a lot of formulas does not make for clarity.

To explain my idea informally, I rewrite the 2nd paragraph of abstraction, and add the the 3rd and 4th paragraphs and Figure 1 in Section I.

For more informal explanations, please refer to the 3rd and 4th paragraphs of subsection III-A. Please also refer to Lemma 1.

3. Related work section contains short descriptions of just two sets of work: one on a related technique (bounded model checking), and another to a piece of work on protocol converter synthesis. What is the relevance of these works to your own work? Did any of them inspire your methods? What is similar and what is different, compared to your work? As it stands, this section is utterly useless to the reader.

Yes, you are right. I have rewrote them to explain their relation with our research.

Comments on Style: Similar to a conference presentation, the paper is full of ‘we’, ‘us’ etc, which should be removed.

Yes, you are right. I have already rewrote them.

Sentences start with ‘Section’, which should be rectified.

I am sorry that I don’t understand.Can you please give me more details?

C. Review Number 3

****

Comments to the Author ————————-

This work continues a line of research that was previously published in TCAD and proposes a better algorithm for synthesizing complementary circuits. A key advance is a reversibility check that can produce negative answers, rather than run forever as the previous algorithm would.

The paper is structured well, and the figures are very nicely done. However, the use of ”Halting” in the title seems unacceptable, as it mostly refers to the authors earlier work, where an algorithm could run forever in some cases. This can be confusing to readers not familiar with previous work. I suggest using ”effective” instead (”efficient” seems inappropriate, since this algorithm is not polynomial-time). ”Effective” usually means ”does the job”.

Thank you for your suggestion. But halting is the major innovation of this paper. For those readers not familiar

with previous work, the subsection II-C gives enough details for them.

Experiments are thorough, and results are convincing. The writing is generally clear, but there are several glitches.

One reoccurring problem with writing is that the authors do not distinguish between >parameters< and >parameter values<. Consider this sentence appearing after Definition 1 "A sufficient condition for the existence of E^{-1} is, there exist three parameters p, d and l,..." I think it refers to >parameter values< because the three parameters already exist.

Yes, you are right. I have already rewrote them.

There are mismatches between singular and plural forms, e.g., in "there are actually two unfolding of transition function T", "two unfoldings" would sound more natural, and "two ways to unfold" would sound better yet. "The 2nd and 4th rows of Table II compares" should use the plural form "compare" (as is done in the next paragraph).

Yes, you are right. I have already rewrote them.

The authors start a new paragraph after 2-3 sentences. This is, of course, convenient when writing and proofreading a paper, but typical journal papers have only several paragraphs per text column. Please merge some paragraphs.

Yes, you are right. I have already rewrote them.

References need some clear-up, as they use inconsistent styles, sometimes in the same reference. For example, in Ref. [8], the first name of the first author is abbreviated as "D.", but the first name of the second author "Ofer" is not. Please abbreviate first names of all authors everywhere.

Yes, you are right. I have already rewrote them.

In Ref. [9], the last name of the author "Mealy" is given before the first name "George". Refs [13] and [14] are inconsistent in how "in" is spelled. It would be helpful to double quote publication titles. Why is Ref. [10] italicized ?

Yes, you are right. I have already rewrote them.

When a journal version of a conference paper is available, reference the journal version.

I can only find the following two journal papers:

1 Eugene Goldberg, Yakov Novikov: BerkMin: A fast and robust Sat-solver. Discrete Applied Mathematics 155(12): 1549-1561 (2007)

2 K. Avnit, V. D'Silva, A. Sowmya, S. Ramesh, S. Parameswaran. Provably correct on-chip communication: A formal approach to automatic protocol converter synthesis. ACM Trans. Design Autom. Electr. Syst. 14(2), pp 1-41, 2009.

3 E. M. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded Model Checking Using Satisfiability Solving," Formal Methods in System Design, vol. 19, no. 1, pp. 7-34, Jan. 2001.

Are they complete?

Finally, I would like to comment on the practice of submitting a fundamentally deficient algorithm to TCAD and then submitting another paper with a simpler, better algorithm to TCAD soon after. This is borderline unethical. The first paper did look unnecessarily complicated, and TCAD normally does not accept algorithms that can run forever. An exception was made, given the novelty of the problem. However, if the authors can quickly develop a simpler, more elegant, faster algorithm that actually stops in all cases, then why should TCAD reviewers and readers waste their time struggling through the first paper ? TCAD publishes results of completed research projects, not preliminary results. Please keep this in mind when submitting papers to TCAD in the future.

First I must thank you for your hard effort in reviewing my paper.

But I think there are some little confusions between us.

First, my previous TCAD paper actually describes two steps of complementary synthesis, the first step checks the existence of decoder, while the second step builds the decoder. The first step is a very simple one, which only checks the existence but not the non-existence of decoder, this is the reason that makes it never stop. And most complexity of that paper is in its second step.

On the other hand, this new TCAD paper describes a totally new and halting algorithm for the first step, and it is SIMPLE and FASTER only when compared to FMCAD'10 conference version. Actually, when compared to my old TCAD paper's first step, it is still a much more complicated one.

And it is not my will to do such borderline unethical thing, all I do is just following the idea that pops up in my mind, and publishing the results.

Thank you again for your hard effort in reviewing my paper.