A Halting Algorithm to Determine the Existence of Decoder

ShengYu Shen, Ying Qin, JianMin Zhang, and SiKun Li
School of Computer Science, National University of Defense Technology
410073,ChangSha,China
Email: {syshen,qy123,jmzhang,skli}@nudt.edu.cn

***Abstract*—Complementary synthesis is first proposed by us [1] to automatically synthesizes the decoder E^{-1} of an encoder E. It determines the existence of E^{-1} by checking the parameterized complementary condition (PC), i.e., whether E's input can be uniquely determined by its output. However, this algorithm will not halt if E^{-1} does not exist. To overcome this problem, we propose a novel halting algorithm to check PC in two steps.**

First, we over-approximate PC with the linear path unique condition (LP), i.e., every linear path of E longer than a particular parameter p always reach the unique state set S^U, in which the input letter can be uniquely determined by the output letter, current state and next state. LP can be falsified by searching for a loop-like path that does not reach S^U within E's recurrence diameter rd. If we find such a loop, then E^{-1} does not exist. Otherwise, we can eventually find a p to prove LP.

Second, if LP is proved in the first step, we construct a list of approximations that forms an onion-ring between PC and its over-approximation LP. If E is found to be in a certain ring but not in the next inner ring, then PC is falsified and E^{-1} does not exist. Otherwise, the existence of E^{-1} is proved.

To illustrate its usefulness, we ran our algorithm on several complex encoders from industrial projects, and their slightly modified variants without corresponding decoders. Experimental results show that our new algorithm always distinguishes correct Es from their incorrect variants and halts properly.

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

I. Introduction

We [1] proposed complementary synthesis to automatically synthesize an encoder E's decoder E^{-1} in two steps. **First**, it determines the existence of E^{-1} by checking the parameterized complementary condition(***PC***), i.e., whether E's input can be uniquely determined by its output on a bounded unfolding of E's transition function. **Second**, it builds E^{-1} by characterizing its Boolean function with an all-solution SAT solver.

However, the bounded nature of the first step makes it an incomplete algorithm that will not halt if E^{-1} does not exist.

To overcome this problem, as shown in Figure 1, we propose a novel halting algorithm to check PC in two steps:

1) First, we over-approximate PC with the linear path unique condition (***LP***), i.e., every linear path of E longer than a particular parameter p always reach the unique state set S^U, in which the input letter can be uniquely determined by the output letter, current state and next state. We then define the negative condition of LP, the loop-like non-unique condition(***LL***). We can falsify LP and prove LL by searching for a loop-like path that does not reach S^U within E's recurrence diameter rd. If we find such a loop-like path, then LL is proved and E^{-1} does not exist. Otherwise, a parameter p can eventually be found to prove LP. In this case, we need the second step below to further check PC.
2) Second, with p found in the first step that prove LP, we construct a list of approximations that forms an onion-ring between PC and its over-approximation LP. If E is found to be in a certain ring but not in the next inner ring, then PC is falsified and E^{-1} does not exist. Otherwise, the existence of E^{-1} is proved.

We have implemented our algorithm with the OCaml language, and solved the generated SAT instances with Zchaff SAT solver [2]. The benchmark set includes several complex encoders from industrial projects, e.g., PCIE and Ethernet, and also their slightly modified variants without corresponding decoders. Experimental results show that our new algorithm always distinguishes correct encoders from their incorrect variants and halts properly.

All these experimental results and related programs can be downloaded from http://www.ssypub.org.

The contribution of this paper is: We propose the first halting algorithm that determines the existence of an encoder's decoder.

The remainder of this paper is organized as follows. Section II presents background materials. Section III introduces how to over-approximate PC with LP, and how to falsify LP by searching for loop-like paths. Section IV introduces how to construct the onion-ring, and how to determine whether E belongs to a certain ring. Section V introduces how to remove redundant output letters to minimize circuit area, while Section VI and Section VII presents experimental results and related works. Section VIII concludes with a note on future work.

II. Preliminaries

A. Basic Notation of Propositional Satisfiability Problem

For a Boolean formula F over variable set V, the **Propositional Satisfiability Problem**(abbreviated as **SAT**) is to find a satisfying assignment $A : V \rightarrow \{0, 1\}$, so that F can be

Fig. 1. Relationship between PC,LP and LL

Fig. 2. Mealy finite state machine

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 [2], Grasp [3], Berkmin [4] and MiniSAT [5]. A formula to be solved by a SAT solver is also called a **SAT instance**.

B. Recurrence Diameter

A circuit can be modeled by **Kripke structure** $M = (S, I, T, A, L)$, with a finite state set S, the initial state set $I \subseteq S$, the transition relation $T \subseteq S \times S$, and the labeling of the states $L : S \to 2^A$ with atomic proposition set A.

Kroening et al. [6] defined the **state variables recurrence diameter** with respect to M, denoted by $rrd(M)$, as the longest loop-free path in M starting from an initial state.

(1)

In this paper, we need a similar concept: the **uninitialized state variables recurrence diameter** with respect to M, denoted by $uirrd(M)$, is the longest loop-free path in M.

(2)

The only difference between these two definitions is that our $uirrd$ does not consider the initial state.

We present these definitions here only to facilitate our proof. Our algorithm does not need to compute these diameters.

C. The Original Algorithm to Determine the Existence of Decoder

The complementary synthesis algorithm [1] includes two steps: determining the existence of decoder and characterizing its Boolean function. We will only introduce the first step here.

The encoder E can be modeled by a Mealy finite state machine [7].

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 output letter from the current state and input letter.

As shown in Figure 2, in the remainder of this paper, we draw the state as a grey round corner box, and draw the transition function T as a white rectangle.

We denote the state, input letter and output letter at the n-th cycle respectively as s_n, i_n and o_n. We denote the sequence of state, input letter and output letter from the n-th to the m-th cycle respectively as s_n^m, i_n^m and o_n^m.

A sufficient condition for the existence of E^{-1} is the **unique condition**, i.e., there exists two parameters d and l, so that i_n of E can be uniquely determined by the output sequence o_{n+d-l}^{n+d-1}. As shown in Figure 3a), 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}.

However, the unique condition is unnecessarily restrictive, because it may not hold when s_n is not reachable, even if E is a correct encoder whose input can be uniquely determined by its output in its reachable state set. So we need to rule out unreachable states before checking the unique condition.

The continuous running character of communication circuits provides us an opportunity to rule out unreachable states easily without paying the expensive cost of computing its reachable state set, i.e., we only need to check the unique condition on the state set RS^∞ that can be reached infinitely often from S.

(3)

(4)

Here, RS^q is the set of states that can be reached from S with exact q steps.

According to Equation (4) and Figure 3a), RS^∞ can be easily over-approximated by prepending a state transition sequence of length p to s_n, which forces s_n to be in the state set $RS^{>p} = \bigcup_{q>p} RS^q$. Obviously, RS^∞ and all $RS^{>p}$ form a total order shown below, which means a tighter over-approximation of RS^∞ can be obtained by increasing the length p of prepended state transition sequence.

$$RS^\infty \subseteq \cdots \subseteq RS^{>p2} \subseteq \cdots \subseteq RS^{>p1} \subseteq \ldots \text{ where } p2 > p1$$

Thus, as shown in Figure 3a), the parameterized complementary condition(PC) [1] can be defined as:

Definition 2: **Parameterized Complementary Condition (PC) :** For encoder E, there exists three parameters p, d and l, which make i_n to be uniquely determined by o_{n+d-l}^{n+d-1} on s_{n-p}^{n+d-1}, i.e., make $F_{PC}(p, d, l)$ in Equation (5) unsatisfiable. We denote it as $E \vDash PC(p, d, l)$, and define $E \vDash PC$ as $\exists p, d, l : E \vDash PC(p, d, l)$.

(5)

The 2nd and 3rd line of Equation (5) correspond to two unfolded instances of E's transition function. 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 unfolded instances to be the same. The 5th line forces their input letters to be different.

III. Over-approximating PC with LP and Falsifying LP by Searching for Loop-like Path

A. Definition of Over-approximation

We first present some related definitions before defining the over-approximation.

Definition 3: **Unique State Set S^U and Non-unique State Set S^N:** For a circuit E, its unique state set S^U is the set of states s_n that makes i_n to be uniquely determined by s_n,o_n and s_{n+1}, i.e., makes F^U in Equation (6) unsatisfiable. The non-unique state set S^N is the complementary set of S^U, i.e., $S^N \stackrel{def}{=} S - S^U$.

Fig. 3. The three unique conditions

(6)

To obtain a halting algorithm, we need to develop a negative condition for PC, which can recognize all those $E \vDash \neg PC$.

Unfortunately, it is very difficult, if not impossible, to develop such a condition. So we choose to first over-approximate PC with the linear path unique condition(LP), and then develop a negative condition for LP – the loop-like non-unique condition(LL). The definitions of LP and LL are given below, and presented intuitively in Figure 3b) and 3c).

***Definition** 4:* **Linear Path Unique Condition (LP) :** For encoder E, every linear path of length p always reaches the unique state set S^U, i.e., makes $F_{LP}(p)$ in Equation (7) unsatisfiable. We denote it as $E \vDash LP(p)$, and define $E \vDash LP$ as $\exists p : E \vDash LP(p)$.

(7)

***Definition** 5:* **Loop-like Non-unique Condition (LL) :** For encoder E, there exists a loop-like path of length p that reaches the non-unique state set S^N, i.e., makes $F_{LL}(p)$ in Equation (8) satisfiable. We denote it as $E \vDash LL(p)$, and define $E \vDash LL$ as $\exists p : E \vDash LL(p)$.

(8)

Equation (8) is very similar to Equation (7), except that $\bigvee_{m=n-p+1}^{n}\{s_m \equiv s_{n-p}\}$ is inserted to find a loop-like path.

The intuition behind LP and LL is to check whether i_n can be uniquely determined by s_n,s_{n+1} and o_n with prepended s_{n-p}^{n-1}. Here, parameters d and l are removed, which makes it easier to find the value of p .

B. Relationships between PC, LP and LL

The relationships between PC, LP and LL are :

1. LP over-approximates PC,i.e.,$E \vDash PC \rightarrow E \vDash LP$.

2. Of LP and LL, there is always one and only one that holds,i.e.,$E \vDash LP \leftrightarrow E \vDash \neg LL$.

These relationships are presented intuitively in Figure 1, and their proofs are presented below. Those impatient readers can skip the remainder of this subsection.

***Theorem** 1:* $E \vDash PC \rightarrow E \vDash LP$

Proof: Let's prove it by contradiction. Assume that $A : V \rightarrow \{0,1\}$ is a satisfying assignment of $F_{LP}(p)$.

We can append a sub-formula $\bigwedge_{m=n+1}^{n+d-1}\{(s_{m+1}, o_m) \equiv T(s_m, i_m)\}$ to $F_{LP}(p)$, and obtain:

(9)

In Equation (9), the newly appended sub-formula represents a state transition sequence starting from s_{n+1}, so $F'_{LP}(p)$ is still satisfiable.

Similarly, we can append another another sub-formula $\bigwedge_{m=n+1}^{n+d-1}\{(s'_{m+1}, o'_m) \equiv T(s'_m, i'_m)\}$ to the 3rd line of Equation (9), and obtain another satisfiable formula:

(10)

We can prepend a third new sub-formula $\bigwedge_{m=n-p}^{n-1}\{(s'_{m+1}, o'_m) \equiv T(s'_m, i'_m)\}$ to the 3rd line of Equation (10), and get a new formula :

(11)

Because of $s_n \equiv s'_n$ in line 4 of Equation (11), this newly prepended sub-formula represents a state transition sequence s'^{n}_{n-p} that reaches s'_n, which means it can be satisfied by the same satisfying assignment of s^n_{n-p}. So $F'''_{LP}(p)$ is still satisfiable.

Assume that A' is a satisfying assignment of the satisfiable formula $F'_{LP}(p)$ in Equation (9). We define a new satisfying assignment A''' as:

(12)

Thus, A''' is a satisfying assignment of $F'''_{LP}(p)$.

By comparing Equation (5) with (11), it is obvious that A''' is a satisfying assignment of the unsatisfiable formula $F_{PC}(p, d, l)$.

This contradiction concludes the proof. ■

***Theorem** 2:* $E \vDash LP \leftrightarrow E \vDash \neg LL$

Proof: **For the $\rightarrow$ direction**, let's prove it by contradiction. Assume that $E \vDash LL$. This means that there exists a loop-like path that reaches state $s_n \in S^N$.

Assume the length of this loop is q, and the parameter of $E \vDash LP$ is p. Then we can unfold this loop $[p/q]+1$ times, to get a path that is longer than p, and reaches a state $s_n \in S^N$. This will lead to $E \vDash \neg LP(p)$.

This contradiction concludes the proof of the $\rightarrow$ direction.

For the $\leftarrow$ direction, assume that $E \vDash \neg LP$ and $E \vDash \neg LL$, then for all p, $F_{LP}(p)$ is satisfiable.

Assume the uninitialized state variables recurrence diameter of E is $uirrd$, and let $p = uirrd + 1$. Then $F_{LP}(p)$ is satisfiable, which means that there is a path of length p that reaches a state $s_n \in S^N$. Because p is larger than $uirrd$, this path must contain a loop in it, which also makes F_{LL} satisfiable.

So $E \vDash LL$ holds, which contradicts with $E \vDash \neg LL$.

This contradiction concludes the proof of the $\leftarrow$ direction. ■

C. Algorithm to Check $E \vDash LP$ and $E \vDash LL$

According to the relationships discussed in Subsection III-B, we develop the algorithm 1 shown below to check $E \vDash LP$ and $E \vDash LL$. This algorithm also discovers the value of parameter p if $E \vDash LP$ holds.

According to Theorem 2, algorithm 1 will eventually halt at line 3 or 6 before p reaches E's uninitialized state variables recurrence diameter $uirrd$.

With algorithm 1, we can determine whether E is an improperly designed encoder that leads to $E \vDash LL$. **But if**

Algorithm 1 checkLPLL

```
for p = 0 → ∞ do
  if F_LP(p) is unsatisfiable then
    print "E ⊨ LP(p)"
    halt;
  else if F_LL(p) is satisfiable then
    print "no E^{-1} due to E ⊨ LL(p)"
    halt;
  end if
end for
```

$E \vDash LP$**, how can we determine whether** E **is an correct encoder that leads to** $E \vDash PC$? We will discuss this problem in the next section.

D. A New Formula that Defines LP

To further discuss the relationship between PC and LP, we need to define a new formula $\overline{F}_{LP}(p,d,l)$, which is equal to $F_{LP}(p)$ and similar to $F_{PC}(p,d,l)$.

(13)

***Theorem** 3:* $\overline{F}_{LP}(p,d,l) \leftrightarrow F_{LP}(p)$

Proof: First, for the $\rightarrow$ direction. It is obvious that the clause set of $\overline{F}_{LP}(p,d,l)$ is a supper set of $F_{LP}(p)$, so the $\rightarrow$ direction is proved.

Second, to prove the $\leftarrow$ direction, we list below all additional formulas in Equation (13) compared to (7), and also our methods to satisfy them.

1. $\bigwedge_{m=n-p}^{n}\{(s'_{m+1}, o'_m) \equiv T(s'_m, i'_m)\}$: This formula can be satisfied by assigning the value of s_m, i_m and o_m to s'_m, i'_m and o'_m respectively.
2. $\bigwedge_{m=n+1}^{n+d-1}\{(s_{m+1}, o_m) \equiv T(s_m, i_m)\}$: this formula represents a state transition sequence starting from s_{n+1}, which are satisfiable.
3. $\bigwedge_{m=n+1}^{n+d-1}\{(s'_{m+1}\ , o'_m) \equiv T(s'_m, i'_m)\}$: this formula represents a state transition sequence starting from s'_{n+1}, which are satisfiable.

Thus, this theorem is proved. ∎

So, in the remainder of this paper, we will replace $F_{LP}(p)$ with $\overline{F}_{LP}(p,d,l)$ to define LP.

IV. Checking $E \vDash PC$ by Constructing Onion-Ring

A. Intuitive Description

To make it easier to follow our presentation, we present our idea intuitively here with an example.

To facilitate our presentation, as shown in Figure 4, we add two new parameters b and f to replace d and l. The backward parameter b refers to the distance between state s_{n-b} and s_n. The forward parameter f refers to the distance between state s_{n+1} and s_{n+1+f}. The relations between $< b, f >$ and $< d, l >$ are:

(14)

Because Algorithm 1 already rule out all Es that lead to $E \vDash LL$, we can assume here that $E \vDash LP(p)$. This means:

Fig. 4. Forward and backward constraints

***Proposition** 1:* i_n is uniquely determined by s_n, o_n and s_{n+1}.

To facilitate our presentation, as shown in Figure 4, we generalize Proposition 1 by:

1) Replacing o_n with o_{n-b}^{n+f},
2) Replacing s_n with s_{n-b},
3) Replacing s_{n+1} with s_{n+f+1},

and thus obtain:

***Proposition** 2:* i_n is uniquely determined by s_{n-b}, o_{n-b}^{n+f} and s_{n+f+1}.

It is obvious that Proposition 1 is a special case of Proposition 2, with $b \equiv 0$ and $f \equiv 0$.

With this generalization, the intuitive description of our algorithm includes the following five steps:

1) First, we ignore both s_{n-b} and s_{n+f+1}, and test whether i_n can be uniquely determined by o_{n-b}^{n+f}. **If yes, our algorithm halts with** $E \vDash PC$.
2) Otherwise, we ignore s_{n-b}, and test whether i_n can be uniquely determined by o_{n-b}^{n+f} and s_{n+f+1}. If yes, then i_n definitely does **NOT** depend on any o_k with $k < n-b$, but it may still depend on some o_k with $k > n+f$. So we need to increase f by 1 and goto step 1.
3) Otherwise, we ignore s_{n+f+1}, and test whether i_n can be uniquely determined by s_{n-b} and o_{n-b}^{n+f}. If yes, then i_n definitely does **NOT** depend on any o_k with $k > n+f$, but it may still depend on some o_k with $k < n-b$. So we need to increase b by 1 and goto step 1.
4) Otherwise, we test whether i_n can be uniquely determined by s_{n-b}, o_{n-b}^{n+f} and s_{n+f+1}. If yes, then i_n may depend on some o_k with both $k < n-b$ and $k > n+f$, we need to increase b and f by 1, and goto step 1.
5) If the algorithm reaches here, then i_n can be uniquely determined by $s_{n-b'}$, $o_{n-b'}^{n+f'}$ and $s_{n+f'+1}$, but **NOT** by s_{n-b}, o_{n-b}^{n+f} and s_{n+f+1}, where $b' \leq b$ and $f' \leq f$. This means that adding even more o_k into o_{n-b}^{n+f} by increasing b and f further, will never make PC holds. **Our algorithm halts with** $E \vDash \neg PC$.

In this algorithm, every step with a pair of b and f corresponds to an onion-ring defined in subsection IV-B. If it halts at step 5, then E belongs to the ring corresponds to b' and f', but does not belong to the next inner ring corresponds to b and f, which means E^{-1} does not exist.

B. Constructing Onion-Ring between PC and LP

According to Figure 4, we define the following formulas:

F_{unfold} defines two unfolded instances of transition function, and constrains that their output sequence are equivalent, and also constrains that their input letters are not equivalent:

(15)

$F_{backward}$ constrains that s_{n-b} equals to s'_{n-b}:

(16)

$F_{forward}$ constrains that s_{n+1+f} equals to s'_{n+1+f}:

(17)

With these formulas, we define another 4 new unique conditions between PC and LP.

$\boldsymbol{LP\_nobf(p,b,f)}$: i_n can be uniquely determined by o_{n-b}^{n+f}, i.e., make F_{LP_nobf} in Equation (18) unsatisfiable.

(18)

$\boldsymbol{LP\_f(p,b,f)}$: i_n can be uniquely determined by o_{n-b}^{n+f} and s_{n+1+f}, i.e., make F_{LP_f} in Equation (19) unsatisfiable.

(19)

$\boldsymbol{LP\_b(p,b,f)}$: i_n can be uniquely determined by s_{n-b} and o_{n-b}^{n+f}, i.e., make F_{LP_b} in Equation (20) unsatisfiable.

(20)

$\boldsymbol{LP\_bf(p,b,f)}$: i_n can be uniquely determined by s_{n-b}, o_{n-b}^{n+f} and s_{n+1+f}, i.e., make F_{LP_bf} in Equation (21) unsatisfiable.

(21)

These new unique conditions, when coupled with parameters b and f, will act as onion-rings between PC and its over-approximation LP. It is obvious that, LP_bf is very similar to LP, while LP_nobf is very similar to PC.

Some lemmas used to prove the correctness of the onion-ring approach are given below:

***Lemma** 1:* $E \vDash LP\_bf(p,b,f) \leftarrow E \vDash LP\_bf(p,b,f+1)$

Proof: Let's prove it by contradiction. Assume that $E \vDash \neg LP\_bf(p,b,f)$ and $E \vDash LP\_bf(p,b,f+1)$, which means that $F_{LP\_bf}(p,b,f)$ is satisfiable while $F_{LP\_bf}(p,b,f+1)$ is unsatisfiable.

We can append a state transition to $F_{LP\_bf}(p,b,f)$ after s_{n+f+1}, to get a new formula $F'_{LP\_bf}(p,b,f)$.

(22)

This newly appended state transition is satisfiable, which make F'_{LP_bf} a satisfiable formula.

Assume A is a satisfying assignment of $F'_{LP\_bf}(p,b,f)$. We define another satisfying assignment A' as

(23)

Obviously, A' is a satisfying assignment of unsatisfiable formula $F_{LP\_bf}(p,b,f+1)$.

This contradiction concludes the proof. ■

***Lemma** 2:* $E \vDash LP\_b(p,b,f) \leftarrow E \vDash LP\_b(p+1,b+1,f)$

Its proof is similar to that of Lemma 1.

***Lemma** 3:* If $E \vDash PC(p,d,l)$, then the following Equation holds.

(24)

Proof: **For the $\leftarrow$ direction**, according to Equation (20) and (21), it is obvious that the clause set of F_{LP_bf} is a super set of F_{LP_b}, which means the unsatisfiability of the latter one implies the unsatisfiability of the former one. So the $\leftarrow$ direction is proved.

For the $\rightarrow$ direction, let's prove it by contradiction. Assume that $F_{LP\_bf}(p+d-l,0,d)$ is unsatisfiable, and A is a satisfying assignment of $F_{LP\_b}(p+d-l,0,d)$.

We define another assignment A' as :

(25)

Obviously, A is a satisfying assignment of Equation (5), which means $E \nvDash PC(p,d,l)$. This contradiction concludes the proof of the $\rightarrow$ direction.

■

***Lemma** 4:* If $E \vDash PC(p,d,l)$, then the following Equation holds.

(26)

Its proof is similar to that of Lemma 3.

An important theorem that defines the onion-ring is presented and proved below:

***Theorem** 4:* If $E \vDash PC(p,d,l)$, then there exists a list of unique conditions with their relationship shown below:

(27)

Proof: According to Equation (13) and (21), the $\leftrightarrow$ relation between the 1st and 2nd line of Equation (27) holds.

According to Lemma 1, the $\leftarrow$ relations between the 2nd and 6th line of Equation (27) holds.

According to Lemma 3, the $\leftrightarrow$ relation between the 6th and 7th line of Equation (27) holds.

According to Lemma 2, the $\leftarrow$ relations between the 7th and 11th line of Equation (27) holds.

According to Lemma 4, the $\leftrightarrow$ relation between the 11th and 12th line of Equation (27) holds.

According to Equation (5) and (18), the $\leftrightarrow$ relations between the last two lines of Equation (27) holds. ■

In Equation (27), all $\leftarrow$ symbols form a total order, which makes all unique conditions on right-hand side of $\vDash$s to form an onion-ring shown in Figure 1.

C. Algorithm Implementation

With those theorems presented in Subsection IV-B, we use the following Algorithm 2 to check $E \vDash PC$.

Algorithm 2 is invoked with the form $checkPCLP(p,0,0)$, with p computed by algorithm 1.

This algorithm just follows the onion-ring defined by Equation (27), from the first line to the last line. If $E \vDash PC$ holds, it will eventually reach line 2, and the existence of E^{-1} is proved. Otherwise, it will eventually reach line 11, which means that E does not belongs to the current ring, and PC is falsified. So E^{-1} does not exist.

V. Removing Redundant Output Letters

Although Algorithm 1 and 2 together are suffice to determine the existence of E^{-1}, the parameter found by line 2 of Algorithm 2 contains some redundancy, which will cause unnecessary large overhead of circuit area.

Algorithm 2 $checkPCLP(p, b, f)$

1: **if** $F_{LP\_nobf}(p, b, f)$ is unsatisfiable **then**
2: **print** "$E \vDash PC(p, f, b + f + 1)$"
3: **halt**;
4: **else if** $F_{LP\_f}(p, b, f)$ is unsatisfiable **then**
5: $checkPCLP(p, b, f + 1)$
6: **else if** $F_{LP\_b}(p, b, f)$ is unsatisfiable **then**
7: $checkPCLP(p + 1, b + 1, f)$
8: **else if** $F_{LP\_bf}(p, b, f)$ is unsatisfiable **then**
9: $checkPCLP(p + 1, b + 1, f + 1)$
10: **else**
11: **print** "no E^{-1} due to $E \vDash \neg PC$"
12: **halt**;
13: **end if**

Fig. 5. Redundant Output Letters

For example, as shown in Figure 5, 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 $k > n$.

We assume again that line 2 of Algorithm 2 find out $E \vDash PC(p, d, l')$. It is obvious that $l' > d$, which make i_n to 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 should be removed to prevent them from being instantiated as latches in circuit E^{-1}.

Algorithm 3 that removes these redundant output letters is presented below:

Algorithm 3 $RemoveRedundancy(p, d, l')$

1: **for** $l = 0 \rightarrow l'$ **do**
2: **if** $F_{PC}(p, d, l)$ is unsatisfiable **then**
3: **print** "$E \vDash PC(p, d, l)$"
4: **halt**;
5: **end if**
6: **end for**

VI. Experimental Results

We have implemented our algorithm in Zchaff [2], and ran it on a PC with a 2.4GHz Intel Core 2 Q6600 processor, 8GB memory and CentOS 5.2 linux. All experimental results and programs can be downloaded from http://www.ssypub.org.

A. Benchmarks

Table I shows information of the following benchmarks.

1. A XGXS encoder compliant to clause 48 of IEEE-802.3ae 2002 standard [8].

2. A XFI encoder compliant to clause 49 of the same IEEE standard.

3. A 66-bit scrambler used to ensure that a data sequence has sufficiently many 0-1 transitions , so that it can run through high-speed noisy serial transmission channel.

4. A PCIE physical coding module.

5. The Ethernet module of Sun's OpenSparc T2 processor.

TABLE I
Information of Benchmarks

B. Experimental Results on Properly Designed Encoders

The 2nd and 6th row of Table II compares the run time of checking $E \vDash PC$ between [1] and our approach. The run time of our approach are much larger than [1]. This is caused by checking the unique and non-unique conditions defined in Section III and IV.

The 3rd and 7th row compare the discovered parameter values: There are some minor differences on the parameter p. This is caused by the different orders in checking various parameter combinations.

According to [9], p is used to constrained the reachable states, only d and l will affect the run time of building E^{-1} and its circuit area. To prove this, we compare the run time of building E^{-1} with all-solution SAT solver in the 4th and 8th row of Table II, and also compare the area of E^{-1} in the 5th and 9th row. These E^{-1} are synthesized with DesignCompiler and LSI10 target library

It is obvious that the differences in parameter p does not cause significant overhead in the run time of all-solution SAT solver and circuit area.

C. Experimental Results on Improperly Designed Encoders

To show the usefulness of our algorithm, we need some improperly designed encoders without corresponding decoders.

We obtained these improperly designed encoders by modifying each benchmark's output statements, such 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 III shows the result of Algorithm 1, while the 3rd row shows the result of algorithm 2. The total run time is shown in the 4th row.

For XFI and scrambler, the result of Algorithm 1 is LL, which falsifies PC directly. So result of Algorithm 2 is **NA**.

It is obvious that our algorithm always terminated, and recognized these modified incorrect encoders.

VII. Related Works

A. Temporal Logic Synthesis

The temporal logic synthesis was first addressed by Clarke et.al [10] and Manna et.al [11]. But Pnueli et.al [12] pointed out that the complexity of LTL synthesis is double exponent.

One line of research [13]–[15] focuses on the so-called generalized reactive formulas of the form: $(\Box\Diamond p_1 \wedge \cdots \Box\Diamond p_m) \rightarrow (\Box\Diamond q_1 \wedge \cdots \Box\Diamond q_n)$. Complexity of solving synthesis problem for such formula is $O(N^3)$.

The other line of research focuses on finding efficient algorithm [16] for expensive safra determination algorithm [17] on an useful formula subset, or just avoiding it [18].

TABLE II
Experimental Results on Properly Designed Encoders

TABLE III
EXPERIMENTAL RESULTS ON IMPROPERLY DESIGNED ENCODERS

Based on these research works, some tools [19] that can handle small temporal formulas have been developed.

All these works assume a hostile environment, which seems too restrictive for many applications. So Fisman et.al [20], Chatterjee et.al [21] and Ummels et.al [22] proposed rational synthesis algorithm, which assumes that each agents act to achieve their own goals instead of failing each other.

B. Protocol Converter Synthesis

The protocol converter synthesis was first proposed by Avnit et.al [23] to automatically generate a translator between two different communication protocols. Avnit et.al [24] improves it with a more efficient design space exploration algorithm.

VIII. CONCLUSIONS AND FUTURE WORKS

This paper proposes the first complete approach that checks whether a particular encoder E has corresponding decoder. Theoretical analysis and experimental results show that our approach always distinguishes correct encoders from their incorrect variants and halts properly.

One future work is to develop a debugging method to find out why the E^{-1} does not exist. For the failure caused by loop-like path, we plan to develop a debugging mechanism based on loop-like counterexample minimization [25].

REFERENCES

[1] ShengYu Shen, JianMin Zhang, Ying Qin, SiKun Li. Synthesizing Complementary Circuits Automatically. in ICCAD'09, pp 381-388, 2009.

[2] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, S. Malik. Chaff: Engineering an Efficient SAT Solver. In DAC'01, pp 530-535, 2001.

[3] João P. Marques Silva, Karem A. Sakallah. GRASP - a new search algorithm for satisfiability. in ICCAD, pp 220-227, 1996.

[4] E. Goldberg, Y Novikov. BerkMin: A Fast and Robust Sat-Solver. in DATE'02, pp 142-149, 2002.

[5] N. Eén, N. Sörensson. Extensible SAT-solver. in SAT'03, pp 502-518, 2003.

[6] D. Kroening and Ofer Strichman. Efficient Computation of Recurrence Diameters. in VMCAI'03, pp 298-309, 2003.

[7] Mealy, George H. A Method for Synthesizing Sequential Circuits. Bell Systems Technical Journal v 34, pp 1045-1079, 1955.

[8] *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.

[9] ShengYu Shen, Ying Qin, KeFei Wang, LiQuan Xiao, JianMin Zhang, SiKun Li. Synthesizing Complementary Circuits Automatically. Accepted by IEEE transaction on CAD of Integrated Circuits and Systems, 2010.

[10] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In IBM Workshop on Logics of Programs,LNCS 131, pp 52-71, 1981.

[11] Z. Manna and P. Wolper. Synthesis of communicating processes from temporal logic specifications. ACM Trans. Prog. Lang. Sys., 6:68-93, 1984.

[12] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In Proc. 16th ACM Symp. Princ. of Prog. Lang.,pages 179-190, 1989.

[13] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In IFAC Symposium on System Structure and Control, pages 469-474. Elsevier, 1998.

[14] R. Alur and S. La Torre. Deterministic generators and games for LTL fragments. ACM Trans. Comput. Log., 5(1):1-25,2004.

[15] N. Piterman, A. Pnueli and Y. Saar, Synthesis of Reactive(1) Designs, in VMCAI'06, pp 364-380, 2006.

[16] O. Maler, D. Nickovic and A. Pnueli. On Synthesizing Controllers from Bounded-Response Properties. In CAV'07, pp 95-107, 2007.

[17] S. Safra. Complexity of Automata on Infinite Objects. PhD thesis, The Weizmann Institute of Science, Rehovot, Israel, March 1989.

[18] A. Harding, M. Ryan and P. Schobbens. A New Algorithm for Strategy Synthesis in LTL Games. in TACAS'05, pp 477-492, 2005.

[19] B. Jobstmann, S. Galler, M. Weiglhofer and R. Bloem. Anzu: A Tool for Property Synthesis. in CAV'07, pp 258-262, 2007.

[20] D Fisman, O Kupferman, Yoad Lustig. Rational Synthesis.in TACAS'10, pp 190-204, 2010.

[21] Chatterjee, K., Henzinger, T.A. Assume-guarantee synthesis. In TACAS'07, pp 261-275, 2007.

[22] Ummels, M. Rational behaviour and strategy construction in in?nite multiplayer games. In FSTTCS'06,pp 212-223, 2006.

[23] K. Avnit, V. D'Silva, A. Sowmya, S. Ramesh, S. Parameswaran. A Formal Approach To The Protocol Converter Problem. in DATE'08, pp 294-299, 2008.

[24] K. Avnit, A. Sowmya. A formal approach to design space exploration of protocol converters. in DATE'09, pp 129-134, 2009.

[25] ShengYu Shen, Ying Qin, Sikun Li. Minimizing Counterexample of ACTL Property. in CHARME'05, pp 393-397, 2005.