Synthesis of Feedback Decoders for Initialized Encoders

Kuan-Hua Tu and Jie-Hong R. Jiang

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

ABSTRACT

Encoding and decoding are common practice in data processing. Designing encoder and decoder circuitry manually can be error prone and time consuming. Although great progress has been made on automating decoder synthesis from its encoder specification, prior specification was limited to an uninitialized encoder only, whose decoder in turn cannot depend on the entire execution history of the encoder. Prior decoder existence condition is unnecessarily stringent as encoders are often initialized to some specific starting states. This paper shows how decoders of initialized encoders can be practically synthesized. Experimental results demonstrate effective decoder synthesis of initialized encoders, beyond existing methods' capabilities.

Categories and Subject Descriptors

B.6.3 [**Logic Design**]: Design Aids—*automatic synthesis*

General Terms

Algorithms, logic synthesis, verification

Keywords

Craig interpolation, decoder, encoder, finite-state transition system, satisfiability solving

1. INTRODUCTION

Data processing is pervasive in computation and communication, and relies on an encoding and decoding scheme for effective and robust data manipulation. An encoder transforms some input data to encoded data, whereas a decoder recovers the original input data from the encoded data (possibly being modified by the underlying communication channel). Given an encoder specification (and possibly channel characteristics), its decoder design may be non-trivial and hard to formally verify. Automating decoder synthesis eases the design and verification tasks.

There have been prior efforts on decoder synthesis. Shen et al. [11] proposed a bounded decoder synthesis method with no termination guarantee. Complete synthesis methods were later established independently by Shen et al. [12] and Liu et al. [7, 8]. Particularly, Liu et al. [7, 8] exploited incremental satisfiability (SAT) solving [5] and Craig interpolation [3, 9, 6] techniques for efficient computation.

All prior methods, however, can only synthesize a special class of decoders, whose decoding process must depend on a *bounded* execution history of their encoders. Therefore the considered decoder must be in a pipelined form without sequential feedbacks. Not all decoders, however, can be in such simplicity. Many encoders in real applications inevitably require their decoding to depend on the entire histories of output sequences starting from initial states. In essence, sequential feedbacks are a necessity for such decoders. Figure 1, modified from [7, 8], shows one such encoder example, whose decoder exists with respect to any initial state but does not exist if no initial state is specified. The decoding must depend on the entire input history, which is unbounded and accounts for the necessity of sequential feedbacks. Moreover, decoders with sequential feedbacks are potentially more compact than those without feedbacks especially when the number of pipeline stages is large for feedback-free decoders. On the other hand, prior work required encoders in their normal operation to be in non-dangling states (a state is called *dangling* if it cannot be reached by any states, including itself, or can be reached only by dangling states), and permitted not all the input data being recovered. However in certain applications such data loss is disadvantageous, if not unacceptable.

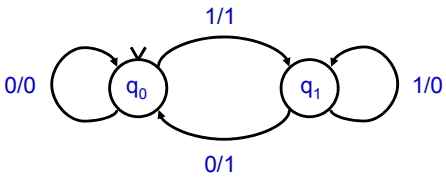

Figure 1: A 0-1 alternation detector with a designated initial state q_0.

This paper investigates the fundamental question how to synthesize decoders without restricting to a bounded execution history of their encoders and without admitting any data loss. The main results include 1) a sound and complete decoder existence checking algorithm, 2) practical decoder existence checking techniques using incremental SAT solving and property directed reachability analysis [2, 4], 3) a decoder synthesis method based on Craig interpolation. Experimental results demonstrate the effectiveness of decoder existence checking and synthesis, while the obtained decoders, unlike those derived previously, can depend on an unbounded encoder execution history and recover data without any loss.

This paper is organized as follows. Section 2 provides some preliminaries. After the decoding problem is stated in Section 3, our new results on decoder existence checking and decoder synthesis are presented in Sections 4 and 5, respectively. Experimental results are shown in Section 6, and finally concluding remarks are given in Section 7.

2. PRELIMINARIES

In the sequel the cardinality of a set S is denoted as $|S|$. The set of truth valuations of a vector $\vec{x} = (x_1, \ldots, x_k)$ of Boolean variables is denoted $[\![\vec{x}]\!]$, for instance, $[\![(x_1, x_2)]\!] =$

DAC 2013, May 29 - June 07, 2013, Austin, Texas, USA.

$\{(0,0),(0,1),(1,0),(1,1)\}$. Symbols $\neg$, $\wedge$, $\vee$, and $\Rightarrow$ stand for logical connectives negation, conjunction, disjunction, and implication, respectively.

2.1 SAT Solving and Craig Interpolation

We assume the reader's familiarity with Boolean satisfiability (SAT) solving [10, 5] and the circuit to conjunctive normal form (CNF) formula conversion [13]. A more detailed exposition can be found in [6].

For decoder synthesis, the following theorem is useful.

THEOREM 1 (CRAIG INTERPOLATION THEOREM). *[3] Given two Boolean formulas ϕ_A and ϕ_B, if $\phi_A \wedge \phi_B$ is unsatisfiable, then there exists a Boolean formula ψ_A, called the* interpolant *of ϕ_A with respect to ϕ_B, referring only to the common variables of ϕ_A and ϕ_B such that $\phi_A \Rightarrow \psi_A$ and $\psi_A \Rightarrow \neg\phi_B$.*

The interpolant ψ_A can be constructed in linear time from a refutation proof of $\phi_A \wedge \phi_B$ produced by a SAT solver [9].

2.2 State Transition Systems

A *state transition system* consists of a state transition relation $T(\vec{x},\vec{s},\vec{y},\vec{s}')$ and a set $I(\vec{s})$ of initial states, where $\vec{s}$, $\vec{s}'$, $\vec{x}$, and $\vec{y}$ are referred to as the current-state variables, next-state variables, input variables, and output variables, respectively. (In the sequel, state sets are represented with characteristic functions. We shall not distinguish between a characteristic function and the set that it represents.) For a deterministic system as we shall assume for an encoder, the transition relation $T(\vec{x},\vec{s},\vec{y},\vec{s}')$ can be alternatively treated as the transition function $T : [\![\vec{x}]\!] \times [\![\vec{s}]\!] \rightarrow [\![\vec{y}]\!] \times [\![\vec{s}']\!]$.

A time-frame expansion of the state transition system $T(\vec{x},\vec{s},\vec{y},\vec{s}')$ is the time unrolling of T into multiple time-indexed copies, denoted $T^t = T(\vec{x}^{\,t},\vec{s}^{\,t},\vec{y}^{\,t},\vec{s}^{\,t+1})$, the transition relation at time t. In contrast, in the sequel $T^* = T(\vec{x^*},\vec{s^*},\vec{y^*},\vec{s^*}')$ denotes a renamed copy of T with variables $\vec{x}$, $\vec{s}$, $\vec{y}$, and $\vec{s}'$ of T substituted with fresh new variables $\vec{x^*}$, $\vec{s^*}$, $\vec{y^*}$, and $\vec{s^*}'$, respectively.

3. PROBLEM STATEMENT

Given a state transition system with transition relation $T(\vec{x},\vec{s},\vec{y},\vec{s}')$ and initial states $I(\vec{s})$, as an encoder it transforms an input sequence to an output sequence. The decoder synthesis problem asks whether a decoder exists that recovers the original input sequence from the observed encoded sequence, and furthermore how to synthesize it if it exists.

A realistic decoder should satisfy the following two properties. First, the decoder must have a finite amount of memory elements for practical implementation. Second, the decoding must be an online process as the input and output sequences can be indefinitely extended without a pre-specified length upper bound. That is, the decoder must causally recover a prefix of the original input sequence on-the-fly based on a so-far observed encoded sequence. Unlike prior work [11, 12, 7, 8], the encoder under our consideration can have normal operation under non-dangling states, and the synthesized decoder can recover the original input sequence starting from the very first input (though with some time delay).

4. DECODER EXISTENCE CHECKING

4.1 Bounded Decoder Existence Checking

Given an encoder starting in some known state at time $t = 0$, the following proposition states the necessary and sufficient condition that the original input to the encoder at time $t = 0$ can be uniquely determined by the encoded sequence of length p generated by the encoder.

PROPOSITION 1. *Given an encoder with transition relation $T(\vec{x},\vec{s},\vec{y},\vec{s}')$ and initial states $I(\vec{s})$, let formula $\varphi_{M(p)}$ be*

$$\bigwedge_{t=0}^{p-1} \Big(T^t \wedge T^{*t} \wedge (\vec{y}^{\,t} = \vec{y^*}^{\,t})\Big) \wedge (\vec{x}^{\,0} \neq \vec{x^*}^{\,0}) \wedge (\vec{s}^{\,0} = \vec{s^*}^{\,0}) \quad (1)$$

where predicate "$=$" asserts the bit-wise equivalence of its two argument variable vectors and "$\neq$" asserts the corresponding negation. Then the original input $\vec{i}^{\,0} \in [\![\vec{x}^{\,0}]\!]$ can be uniquely determined by observing the encoded outputs $\vec{o}^{\,0},\ldots,\vec{o}^{\,p-1} \in [\![\vec{y}^{\,0}]\!] \times \cdots \times [\![\vec{y}^{\,p-1}]\!]$ of length $p \geq 1$ if and only if the formula

$$\varphi_{M(p)} \wedge I(\vec{s}^{\,0}) \quad (2)$$

is unsatisfiable.

In the sequel, we shall call Formula (1), namely $\varphi_{M(p)}$, the miter formula.

To ensure that the original input $\vec{x}^{\,t}$ at any time $t \geq 0$ can be uniquely determined by observing an encoded output sequence of length p starting from a known state at time t, Proposition 1 has to be strengthened by relaxing the state variable constraint of $\vec{s}^{\,0}$ from the initial states to any states reachable from the initial states.

THEOREM 2. *For a given encoder with (deterministic) transition relation $T(\vec{x},\vec{s},\vec{y},\vec{s}')$ and initial states $I(\vec{s})$, let $R(\vec{s})$ be the set of reachable states. Then the original input $\vec{i}^{\,t} \in [\![\vec{x}^{\,t}]\!]$ at any time $t \geq 0$ can be uniquely determined by observing the encoded outputs $\vec{o}^{\,t},\ldots,\vec{o}^{\,t+p-1} \in [\![\vec{y}^{\,t}]\!] \times \cdots \times [\![\vec{y}^{\,t+p-1}]\!]$ of length $p \geq 1$ if and only if the formula*

$$\varphi_{M(p)} \wedge R(\vec{s}^{\,0}) \quad (3)$$

is unsatisfiable.

PROOF. ($\Longrightarrow$) For the sake of contradiction, assume $\varphi_{M(p)} \wedge R(\vec{s}^{\,0})$ is satisfiable. Then there exists some reachable state $\vec{q} \in [\![\vec{s}]\!]$ such that $\varphi_{M(p)} \wedge (\vec{s}^{\,0} = \vec{q})$ is satisfiable. By Proposition 1, we know the original input $\vec{i}^{\,0}$ with current state $\vec{q}$ cannot be uniquely determined by the encoded outputs $\vec{o}^{\,0},\ldots,\vec{o}^{\,p-1}$. (Note that $t = 0$ is a relative reference time point.) Since $\vec{q}$ is a reachable state, it follows that not every original input at any time can be uniquely determined from an encoded output sequence of length p.

($\Longleftarrow$) If Formula (3) is unsatisfiable, no state $\vec{q} \in [\![\vec{s}]\!]$ reachable from I satisfies $\varphi_{M(p)} \wedge (\vec{s}^{\,0} = \vec{q})$. For $\vec{q}$ being an initial state, the unsatisfiability of $\varphi_{M(p)} \wedge (\vec{s}^{\,0} = \vec{q})$ implies the original input $\vec{i}^{\,0} \in [\![\vec{x}^{\,0}]\!]$ can be uniquely determined from the encoded output sequence $\vec{o}^{\,0},\ldots,\vec{o}^{\,p-1}$ by Proposition 1. Thereby the next state $\vec{q}'$ of $\vec{q}$ can be uniquely determined from the deterministic transition relation T under current state $\vec{q}$ and current input $\vec{i}^{\,0}$. Again since $\vec{q}'$ is a reachable state, $\varphi_{M(p)} \wedge (\vec{s}^{\,0} = \vec{q}')$ must be unsatisfiable, and the original input under current state $\vec{q}'$ can be uniquely determined from the encoded output sequence $\vec{o}^{\,1},\ldots,\vec{o}^{\,p}$ and so is its next state. Repeating this argument, we know that the original input $\vec{i}^{\,t}$ at any time t can be determined from the encoded output sequence $\vec{o}^{\,t},\ldots,\vec{o}^{\,t+p-1}$. ■

Notice that the above decoder existence condition, Formula (3), is with respect to some pre-specified length bound p. The decoder non-existence at $p = n$ does not exclude the decoder existence at a larger $p = n+k$ for $k \geq 1$ however. To determine if there exists no decoder for arbitrary $p \geq 1$, additional constraints need to be imposed to make the checking finitary and complete as we show below.

4.2 Unbounded Decoder Existence Checking

The following proposition provides a necessary and sufficient condition for determining decoder existence without referring to a pre-specified length bound.

PROPOSITION 2. *Given an encoder with transition relation $T(\vec{s},\vec{x},\vec{s}',\vec{y})$ and initial states $I(\vec{s})$, its decoder does not exist if and only if the encoder starting from some reachable state $\vec{q} \in [\![\vec{s}^{\,t}]\!]$ at time t produces the same infinite encoded output sequence $\vec{o}^{\,t}, \vec{o}^{\,t+1}, ... \in [\![\vec{y}^{\,t}]\!] \times [\![\vec{y}^{\,t+1}]\!] \times \cdots$ under two input sequences $\vec{i_1}^{\,t}, \vec{i_1}^{\,t+1}, \ldots$ and $\vec{i_2}^{\,t}, \vec{i_2}^{\,t+1}, \ldots \in [\![\vec{x}^{\,t}]\!] \times [\![\vec{x}^{\,t+1}]\!] \times \cdots$ with $\vec{i_1}^{\,t} \neq \vec{i_2}^{\,t}$ at time t.*

The following theorem provides a computational means to demonstrate the non-existence of decoders, that is, Formula (1) is satisfiable for any arbitrary $p \geq 1$.

THEOREM 3. *The decoder of a transition system $T(\vec{x},\vec{s},\vec{y},\vec{s}')$ with initial states $I(\vec{s})$ does not exist if and only if the formula*

$$\varphi_{M(p)} \wedge R(\vec{s}^{\,0}) \wedge \bigvee_{i=0}^{p-1} \bigvee_{j=i+1}^{p} \left((\vec{s}^{\,i} = \vec{s}^{\,j}) \wedge (\vec{s^*}^{\,i} = \vec{s^*}^{\,j}) \right), \quad (4)$$

is satisfiable for some p, where R characterizes the set of states reachable from I under T.

PROOF. ($\Longrightarrow$) If there exists no decoder, then there exist two distinct inputs $\vec{i}^{\,0} \in [\![\vec{x}^{\,0}]\!]$ and $\vec{i^*}^{\,0} \in [\![\vec{x^*}^{\,0}]\!]$ at time $t = 0$ that are consistent (in terms of input-output traces confined by T) yielding the same infinite encoded output sequence. Let the corresponding two state traces be $\vec{q}^{\,0}, \vec{q}^{\,1}, \ldots$ and $\vec{q^*}^{\,0}, \vec{q^*}^{\,1}, \ldots$, respectively, with $\vec{q}^{\,0} = \vec{q^*}^{\,0}$. Because of the finite state space of the encoder, there must be a state pair $(\vec{q}^{\,k}, \vec{q^*}^{\,k})$ at time k in the two traces repeating itself at some other time $k + l$, that is, $(\vec{q}^{\,k+l}, \vec{q^*}^{\,k+l}) = (\vec{q}^{\,k}, \vec{q^*}^{\,k})$. Hence Formula (4) is satisfiable.

($\Longleftarrow$) Assume Formula (4) is satisfiable under some p. Then there must exist two infinite input sequences with two distinct current inputs and the same current reachable state that result in the same infinite encoded output sequence. By Proposition 2, the decoder does not exist. ■

PROPOSITION 3. *If Formula (4) is satisfiable under $p = 1, 2, \ldots, |R|^2$, where $|R|$ denotes the cardinality of the reachable state set R of the encoder, then Formula (4) remains satisfiable for arbitrary $p > |R|^2$.*

Suppose the decoder existence condition of Formula (4) is checked by incrementing p starting from 1 until decoder existence or non-existence is concluded. Then the looping sub-formula of Formula (4), i.e.,

$$\bigvee_{i=0}^{p-1} \bigvee_{j=i+1}^{p} \left((\vec{s}^{\,i} = \vec{s}^{\,j}) \wedge (\vec{s^*}^{\,i} = \vec{s^*}^{\,j}) \right). \quad (5)$$

can be simplified to

$$\varphi_{L(p)} \;=\; \bigvee_{i=0}^{p-1} \left((\vec{s}^{\,i} = \vec{s}^{\,p}) \wedge (\vec{s^*}^{\,i} = \vec{s^*}^{\,p}) \right). \quad (6)$$

This simplification is possible due to the fact that the looping sub-constraints of Formula (5) corresponding to $j < p$ have been checked and shown unsatisfiable in conjuncting with $\varphi_{M(p-1)} \wedge R(\vec{s}^{\,0})$. Since $\varphi_{M(p)} \Rightarrow \varphi_{M(p-1)}$, these looping sub-constraints of $j < p$ play no role contributing to the satisfiability of Formula (4) and can be removed. That is, Formula (4) can be simplified to

$$\varphi_{M(p)} \wedge R(\vec{s}^{\,0}) \wedge \varphi_{L(p)} \quad (7)$$

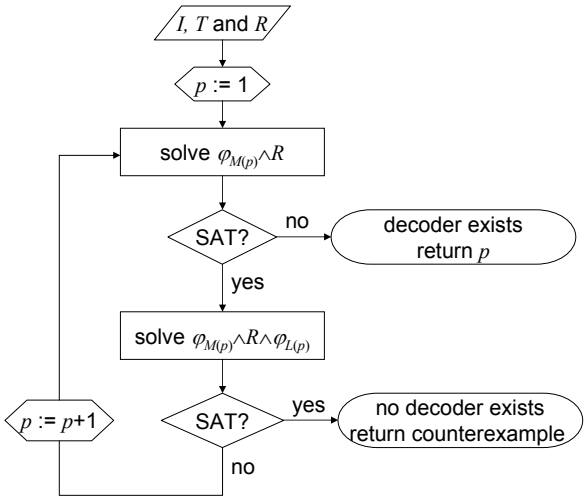

Figure 2: Flow of decoder existence checking with *a priori* reachability knowledge.

in incremental satisfiability solving.

The flow of decoder existence checking, with the reachable state set R given, is shown in Figure 2. In the procedure, p starts from 1 and is increased by 1 until either $\varphi_{M(p)} \wedge R$ is unsatisfiable (a decoder exists) or $\varphi_{M(p)} \wedge R \wedge \varphi_{L(p)}$ is satisfiable (no decoder exists). By Proposition 3, the procedure is guaranteed to terminate at some $p \leq |R|^2$.

4.3 Decoder Existence Checking without A Priori Reachability Knowledge

The above discussion assumes the reachable state set R is given. Exact state reachability analysis, however, is often too expensive to be practically computed. Fortunately, recent advances in SAT-based unbounded model checking (UMC), in particular the interpolation method [9] and the property directed reachability method [2, 4], allow efficient computation of over-approximated reachable state sets. Given a state transition system T, initial state set I, and final state set F as input, an UMC algorithm, denoted UMC(I, T, F), returns either an input trace as an evidence in the case of F reachable from I, or an over-approximated state set $R^\dagger$ satisfying

$$\forall \vec{s}.\ I(\vec{s}) \Rightarrow R^\dagger(\vec{s}), \quad (8)$$

$$\forall \vec{x},\vec{s},\vec{y},\vec{s}'.\ R^\dagger(\vec{s}) \wedge T(\vec{x},\vec{s},\vec{y},\vec{s}') \Rightarrow R^\dagger(\vec{s}'), \text{ and} \quad (9)$$

$$\forall \vec{s}.\ R^\dagger(\vec{s}) \Rightarrow \neg F(\vec{s}) \quad (10)$$

in the case of F not reachable from I. It is immediate that $R \Rightarrow R^\dagger$ and thus $R^\dagger$ over-approximates R.

An UMC algorithm can be exploited as a black-box tool in decoder existence checking as follows. Instead of checking directly whether Formula (7) is satisfiable, we check if any state $\vec{q} \in [\![\vec{s}]\!]$ satisfying $\varphi_{F(p)}(\vec{s}) =$

$$\exists \vec{x}^{\,0}, \ldots, \vec{x}^{\,p-1}, \vec{y}^{\,0}, \ldots, \vec{y}^{\,p-1}, \vec{s}^{\,1}, \ldots, \vec{s}^{\,p}.\varphi_{M(p)} \wedge \varphi_{L(p)} \quad (11)$$

can be reached from the initial states I. (Notice that in Formula (11) the original free variables $\vec{s}^{\,0}$ are renamed to $\vec{s}$ to avoid confusion as the time index 0 of $\vec{s}^{\,0}$ is merely a relative reference time point rather than the absolute initial time point.) That is, Formula (11) is treated as the final state set F. Notice that no quantifier elimination needs to be performed on these existentially quantified variables as they are simply free variables during SAT-based UMC computation.

Intuitively $\varphi_{F(p)}$ characterizes the set of bad states (starting from them, state pair recurrence happens within p steps

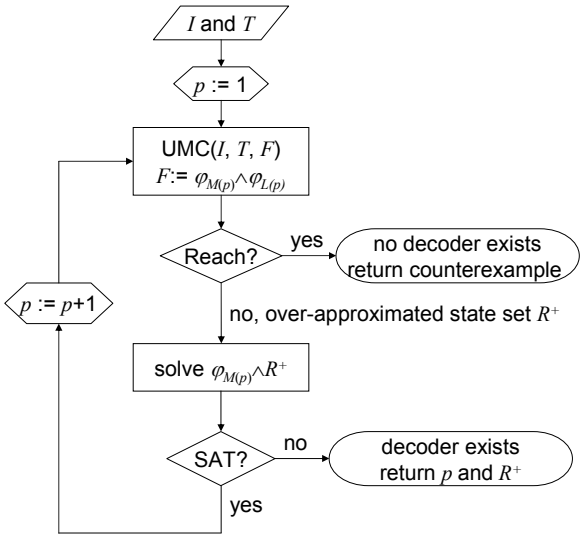

Figure 3: Flow of decoder existence checking without *a priori* reachability knowledge.

in the state trace, i.e., $\varphi_{L(p)}$ is satisfied), whose inputs cannot be uniquely determined by observing output sequences of finite lengths. So if any state of $\varphi_{F(p)}$ is reachable from I, then no decoder exists and an UMC algorithm returns an input trace that accounts for the non-existence. Otherwise, an over-approximated reachable state set $R^\dagger$ is returned by an UMC algorithm. It allows the replacement of Formula (3) with

$$\varphi_{M(p)} \wedge R^\dagger(\vec{s}^0). \tag{12}$$

If Formula (12) is unsatisfiable, then decoder exists and can be further synthesized (as to be detailed in Section 5). Otherwise the length parameter p is incremented for a new iteration of computation. The overall flow of the computation is summarized in Figure 3.

The correctness of the above computation flow is based on the following theorems.

THEOREM 4. *For a given encoder with transition relation $T(\vec{x}, \vec{s}, \vec{y}, \vec{s}')$ and initial states I, its decoder does not exist if and only if there is some state $\vec{q} \in [\![\vec{s}^0]\!]$ that satisfies*

$$\varphi_{M(p)} \wedge \varphi_{L(p)} \tag{13}$$

for some $p \geq 1$ and is reachable from I under transition relation T.

PROOF. ($\Longrightarrow$) If there exists no decoder, formula $\varphi_{M(p)} \wedge (\vec{s}^0 = \vec{q})$ is satisfiable for any reachable state $\vec{q}$ and any p. Moreover, because the cardinality of reachable state set is finite, a state pair $(\vec{q_1}, \vec{q_2}) \in [\![\vec{s}]\!] \times [\![\vec{s^*}]\!]$ in the state trace satisfying $\varphi_{M(p)}$ must recur at some p. Hence $\varphi_{L(p)}$ will eventually be satisfiable, and so will Formula (13).

($\Longleftarrow$) If Formula (13) is satisfiable, then there exists a reachable state whose input cannot be uniquely determined by observing output sequences of length up to p. Since a state pair recurs within the state trace as asserted by $\varphi_{L(p)}$, appending the time-frames corresponding to the recurrent state trace of length n to the miter formula $\varphi_{M(p)}$ makes $\varphi_{M(p+n)} \wedge \varphi_{L(p+n)}$ remain satisfiable. By continuing appending time-frames, formula $\varphi_{M(p+kn)} \wedge \varphi_{L(p+kn)}$ remains satisfiable for arbitrary $k \geq 0$. Therefore no decoder exists. ■

THEOREM 5. *For a given encoder with transition relation $T(\vec{x}, \vec{s}, \vec{y}, \vec{s}')$ and initial states I, let $R^\dagger$ be an over-approximated set of reachable states. Then the original input $\vec{i}^t \in [\![\vec{x}^t]\!]$ at any time $t \geq 0$ can be uniquely determined by observing the encoded outputs $\vec{o}^t, \ldots, \vec{o}^{t+p-1} \in [\![\vec{y}^t]\!] \times \cdots \times [\![\vec{y}^{t+p-1}]\!]$ of length $p \geq 1$ if Formula (12) is unsatisfiable.*

PROOF. Since $R \Rightarrow R^\dagger$, the unsatisfiability of Formula (12) implies that of Formula (3). By Theorem 2, the statement holds. However the converse is not true since Formula (12) can be satisfiable under the unreachable states $R^\dagger \wedge \neg R$ while $\vec{i}^t$ can still be uniquely determined. ■

Due to the reachability over-approximation, it is possible that the computation flow of Figure 3 terminates at some $p = n$ that is much larger than that of Figure 2. Nevertheless it is possible to overcome this deficiency with a modified computation flow of Figure 3 by replacing the SAT solving $\varphi_{M(p)} \wedge R^\dagger$ with the computation UMC$(I, T, \varphi_{M(p)})$. If the final states $F = \varphi_{M(p)}$ are unreachable from I, then a decoder exists. Otherwise, p is incremented for the next computation iteration. (Surely the price to pay is to perform the more expensive UMC computation rather than SAT solving.)

THEOREM 6. *The decoder existence checking procedure of Figure 3 with the SAT solving $\varphi_{M(p)} \wedge R^\dagger$ replaced by the computation UMC$(I, T, \varphi_{M(p)})$ (such that a decoder exists if final states $F = \varphi_{M(p)}$ are reachable from I, and p is incremented for next iteration otherwise) terminates at $p = n$ for some $n \geq 1$ same as the procedure of Figure 2.*

PROOF. Assume the procedure of Figure 2 terminates at iteration $p = n$ under decoder existence. Then $\varphi_{M(n)} \wedge R(\vec{s}^0)$ is unsatisfiable, for R the exact reachable state set. That is, any state $\vec{q} \in [\![\vec{s}]\!]$ cannot satisfy both $R(\vec{q})$ and $\varphi_{M(n)} \wedge (\vec{s}^0 = \vec{q})$. Hence any state in the final states $F = \varphi_{M(n)}$ is not in the reachable state set R. Because the final states $F = \varphi_{M(n)}$ are unreachable from I, UMC$(I, T, \varphi_{M(p)})$ should return unreachability when $p = n$ and thus the modified procedure terminates at the same iteration n.

On the other hand, assume the procedure of Figure 2 terminates at iteration $p = n$ under decoder non-existence. Then $\varphi_{M(n)} \wedge R \wedge \varphi_{L(n)}$ is satisfiable. Since UMC is complete in proving reachability, UMC$(I, T, \varphi_{M(n)} \wedge \varphi_{L(n)})$ must establish the reachability of $\varphi_{M(n)} \wedge \varphi_{L(n)}$ from I, and thus the modified procedure terminates at iteration n. ■

5. DECODER SYNTHESIS

By the unsatisfiability of $\varphi_{M(p)} \wedge R^\dagger(\vec{s}^0)$ (which includes $\varphi_{M(p)} \wedge R(\vec{s}^0)$ as a special case), a decoder can be synthesized from the corresponding resolution refutation by Craig interpolation. The decoding function f_i corresponding to every output bit $x_i^0 \in \vec{x}^0$ of the decoder is synthesized one at a time. The actual length p_i $(1 \leq p_i \leq p)$ of the observation window for f_i is usually smaller than p. Let $\varphi_{M_i(p_i)}$ be

$$\bigwedge_{t=0}^{p-1} \left(T^t \wedge T^{*t}\right) \wedge \bigwedge_{t=0}^{p_i-1} \left(\vec{y}^t = \vec{y^*}^t\right) \wedge (x_i^0 \neq x_i^{*0}) \wedge (\vec{s}^0 = \vec{s^*}^0). \tag{14}$$

Then $\varphi_{M_i(p_i)} \wedge R^\dagger(\vec{s}^0)$ must remain unsatisfiable and f_i can be obtained as follows, similar to the synthesis approach proposed in [6].

THEOREM 7. *Given a transition system $T(\vec{x}, \vec{s}, \vec{y}, \vec{s}')$ and its (over-approximated) reachable state set $R^\dagger$, if $\varphi_{M_i(p_i)} \wedge R^\dagger(\vec{s}^0)$ is unsatisfiable, then the interpolant ψ_{i_A} of ϕ_{i_A} with*

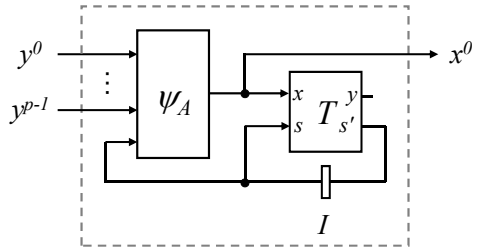

Figure 4: Block diagram of synthesized decoder.

respect to ϕ_{i_B} is a valid decoding function for $x_i^0 \in \vec{x}^0$, where

$$\phi_{i_A} : \bigwedge_{t=0}^{p-1} T^t \wedge x_i^0, \text{ and} \tag{15}$$

$$\phi_{i_B} : \bigwedge_{t=0}^{p-1} T^{*t} \wedge \bigwedge_{t=0}^{p_i-1} \left(\vec{y}^t = \vec{y^*}^t\right) \wedge \neg x_i^{*0} \wedge (\vec{s}^0 = \vec{s^*}^0) \wedge R^\dagger(\vec{s}^0). \tag{16}$$

PROOF. The common variables of ϕ_{i_A} and ϕ_{i_B} are $\vec{s}^0$ and $\vec{y}^0, \ldots, \vec{y}^{p_i-1}$. By Theorem 1, the interpolant ψ_{i_A} refers only to these common variables. Furthermore its onset (respectively offset) contains the assignments to the common variables that satisfy ϕ_{i_A} (respectively ϕ_{i_B}). Because x_i^0 is asserted to be true in ϕ_{i_A} and x_i^{*0} is asserted to be false in ϕ_{i_B}, the onset (respectively offset) of the interpolant contains the assignments to the common variables that determine x_i^0 to true (x_i^{*0} to false). So the interpolant validly implements the decoding function of variable x_i^0. ■

Let $\vec{\psi_A}$ be the collection of obtained decoding functions. Then the decoder can be constructed by composing $\vec{\psi_A}$ and the encoder transition function T as the circuit shown in Figure 4. Note that the decoder should start its operation after its first p inputs $\vec{o}^0, \ldots, \vec{o}^{p-1} \in [\![\vec{y}^0]\!] \times \cdots \times [\![\vec{y}^{p-1}]\!]$ are ready. So it produces its first output at the p^{th} clock cycle.

THEOREM 8. *For a given encoder with transition relation $T(\vec{x}, \vec{s}, \vec{y}, \vec{s}')$ and initial states I, its decoder has transition relation $T_d(\vec{x_d}, \vec{s_d}, \vec{y_d}, \vec{s_d}')$ equal to*

$$\left(\vec{y_d} = \vec{\psi_A}(\vec{x}_d, \vec{s_d})\right) \wedge \left(\exists \vec{y}.T(\vec{y}_d, \vec{s_d}, \vec{y}, \vec{s_d}')\right), \tag{17}$$

with $\vec{x_d} = (\vec{y}^0, \ldots, \vec{y}^{p-1})$, and has initial states $I_d(\vec{s_d})$ equal to $I(\vec{s_d})$.

PROOF. By Theorem 7, $\vec{\psi_A}(\vec{x}_d, \vec{s_d})$ for $\vec{x}_d = (\vec{y}^0, \ldots, \vec{y}^{p-1})$ and $\vec{s_d} = \vec{s}^0$ form the decoding functions of variables $\vec{x}^0 = \vec{y_d}$. The output function $[\![\vec{x}_d]\!] \times [\![\vec{s_d}]\!] \to [\![\vec{y_d}]\!]$ of the decoder can be represented as relation $\vec{y_d} = \vec{\psi_A}(\vec{x}_d, \vec{s_d})$.

In addition, because the decoding functions $\vec{\psi_A}$ need the output and state information of the encoder to reconstruct the original input sequence while the decoder only receives the encoded output sequence from the encoder, the decoder needs to compute the state information by itself. Embedding the transition relation of the encoder inside allows the decoder to keep track of the state transition in synchronization with the encoder. Notice that the output of the embedded encoder transition relation is of no use in the decoder and can be removed through logic minimization. The overall transition relation of the decoder is generated as the conjunction of relations $\vec{y_d} = \vec{\psi_A}(\vec{x}_d, \vec{s_d})$ and $\exists \vec{y}.T(\vec{y_d}, \vec{s_d}, \vec{y}, \vec{s_d}')$. ■

Notice that the above discussion imposes no constraint on the cardinality of the initial state set, i.e., $|I| \geq 1$. Although there are multiple initialization choices when $|I| > 1$, the start state of the encoder and that of the decoder should match. Otherwise the decoder may not correctly recover the expected original input sequence of the encoder.

Table 1: Benchmark Statistics.

circuit	#gate/#level	#reg	#input
PCIE	265/24	22	14
XGXS	177/15	15	9
Scrambler	535/9	58	65
T2Ethernet	1094/18	48	8
CC13	63/8	13	1
CC14	66/8	14	1
CC3	12/4	3	1
CC4	24/4	4	1
AD	5/2	1	1
Huff-alphabet	297/18	9	5
Huff-jpeg	1528/27	12	8
Huff-ran8	2553/32	13	8
Huff-ran9	5155/32	14	9
Huff-skew5	538/21	10	5
Huff-skew6	1092/23	12	6
LFSR-12-6-4	12/6	12	1
LFSR-26-6-2	12/6	26	1
LFSR-32-7-6	12/6	32	1
SBC-Add(4,4)	171/23	16	4
C2670	717/21	0	233
s38417	9219/31	1636	28
s444	155/13	21	3
s5378	1343/17	164	35
s6669	2263/80	239	83

6. EXPERIMENTAL RESULTS

The proposed method, named DECOSY-I, was programmed in the C language and implemented within the ABC system [1], where command `pdr` [4] was used for the UMC reachability computation. The experiments were conducted on a Linux machine with Xeon 2.53GHz CPU and 48GB RAM.

The considered benchmark circuits are listed in Table 1, where the numbers of gates, logic levels, registers, and inputs are shown. Circuits `XGXS`, `Scrambler`, `PCIE`, and `T2Ethernet` are from [11]; the `CC` series circuits are convolutional code encoders from [7, 8]; circuit `AD` is the encoder of Figure 1; the `Huff` series circuits are Huffman code encoders; the `LFSR` series circuits correspond to linear feedback shift registers; the `SBC` circuit is a sliding block code encoder; others are from the MCNC and ISCAS benchmark suites. In the following experiments, the obtained decoder circuits were optimized in ABC under the script "`strash`; `scleanup`; `dsd`; `strash`; `dc2`; `dc2`; `dch`; `map`", where technology mapping was performed using the *mcnc.genlib* library.

Tables 2, 3, and 4 show the statistics of DECOSY [7, 8] assuming uninitialized encoder operation and our new DECOSY-I assuming initialized encoder operation. It should be emphasized that the two approaches cannot be directly compared as their decoding problems are fundamentally different. The circuits shown in Table 2 have decoders under both initialization assumptions; those in Table 3 have decoders only under the initialized assumption; those in Table 4 have no decoders at all.

Table 2 lists the window size of observing output sequences in Columns 2 and 6, the numbers of decoder inputs/registers in Columns 3 and 7, decoder area/delay in Columns 4 and 8, and CPU time (including decoder generation time plus script optimization time in parentheses) in Columns 5 and 9. As shown, the runtimes are comparable except for circuits `CC-13` and `CC-14`, where DECOSY timed out at 3600 seconds (with a window size of 13 upon timeout). On the other hand, the decoders generated by our new DECOSY-I are usually slightly larger than those by DECOSY since we require the transition

Table 2: Comparison on Decoders Synthesized under Different Encoder Assumptions.

circuit	DECOSY (for uninitialized encoder)				DECOSY-I (for initialized encoder)			
	window size	#in/#reg	area/delay	time (s)	window size	#in/#reg	area/delay	time (s)
PCIE	4	11/0	147/5.2	0.13 (+0.10)	3	10/0	168/5.6	0.31 (+0.10)
XGXS	3	11/0	294/7.7	0.04 (+0.05)	2	10/1	468/10.2	0.18 (+0.09)
Scrambler	3	65/64	640/3.8	0.54 (+0.07)	1	64/58	727/3.8	1.26 (+0.09)
T2Ethernet	6	11/0	388/9.9	4.96 (+0.08)	5	10/0	423/10.2	0.94 (+0.04)
CC3	3	4/0	10/3.8	0.00 (+0.09)	2	1/2	12/1.9	0.01 (+0.00)
CC4	3	6/0	15/6.9	0.00 (+0.00)	2	1/3	18/3.8	0.01 (+0.00)
CC13	13	–	–	> 3600	2	1/12	52/5.7	0.09 (+0.02)
CC14	13	–	–	> 3600	2	1/13	86/5.5	0.02 (+0.02)

Table 3: Comparison on Decoder (Non-)Existence Checking under Different Encoder Assumptions.

circuit	DECOSY (for uninitialized encoder)			DECOSY-I (for initialized encoder)				
	window size	exist?	time (s)	window size	exist?	#in/#reg	area/delay	time (s)
AD	1	no	0.00	1	yes	1/1	10/1.9	0.01 (+0.00)
Huff-alphabet	5	no	0.02	10	yes	10/9	261/9.6	0.52 (+0.04)
Huff-jpeg	5	no	0.44	16	yes	16/12	1058/16.5	19.16 (+0.20)
Huff-ran8	7	no	1.57	19	yes	19/13	1995/14.3	72.07 (+0.28)
Huff-ran9	7	no	2.37	19	yes	19/14	4110/16.5	467.96 (+0.76)
Huff-skew5	3	no	0.00	31	yes	31/10	395/25.9	7.17 (+0.05)
Huff-skew6	3	no	0.02	63	yes	63/12	826/51.5	289.21 (+0.12)
LFSR-12-6-4	31	no	11.63	1	yes	1/12	36/4.0	0.01 (+0.05)
LFSR-26-6-2	55	no	> 3600	1	yes	1/26	57/4.0	0.01 (+0.02)
LFSR-32-7-6	15	no	0.14	1	yes	1/32	66/3.8	0.01 (+0.01)
SBC-Add(4,4)	5	no	0.29	1	yes	7/16	48584§/22.4§	50.65 (+12.25§)

Table 4: Comparison on Decoder Non-Existence Checking under Different Encoder Assumptions.

circuit	DECOSY (for uninitialized encoder)		DECOSY-I (for initialized encoder)	
	window size	time (s)	window size	time (s)
C2670	1	0.00	1	0.03
s38417	3	0.15	2	346.88
s444	1	0.00	1	0.01
s5378	1	0.01	1	1.00
s6669	1	0.02	1	91.59

function of the encoder to be embedded as part of the decoder although our decoders require smaller observation windows. Note that, unlike our obtained decoder, the decoder generated by DECOSY assumes steady state operation and may not recover the entire original input sequence.

Table 3 reveals that, under different initialization assumptions, decoder existence checking may exhibit very different characteristics. For circuits LFSR-12 and LFSR-26, for example, DECOSY required large window sizes to conclude decoder non-existence whereas DECOSY-I efficiently synthesized their decoders with window sizes 1. (DECOSY timed out on LFSR-26 with a window size of 55.) Another special case is SBC-Add(4,4), where logic optimization failed in dsd of our ABC synthesis script. The reported decoder area/delay and script runtime exclude dsd transformation and are marked with "§" in superscript.

Table 4 suggests that the runtime of DECOSY-I is proportional to circuit and window sizes (particularly dominated by pdr computation) while DECOSY concludes decoder non-existence within 0.15 seconds for all the circuits.

7. CONCLUSIONS

A sound and complete approach has been proposed to synthesizing decoders for initialized encoders. An obtained decoder can depend on an unbounded history of encoder execution (which should be distinguished from a bounded observation window), and recover the original input sequence without any prefix loss. This approach exceeds the capabilities of prior methods, and, as justified by experimental results, achieves computational efficiency by SAT-based approximative reachability analysis and interpolation-based synthesis.

Acknowledgments

This work was supported in part by the National Science Council under grants NSC 99-2221-E-002-214-MY3, 99-2923-E-002-005-MY3, and 101-2923-E-002-015-MY2.

8. REFERENCES

[1] Berkeley Logic Synthesis and Verification Group. *ABC: A system for sequential synthesis and verification.* http://www.eecs.berkeley.edu/~alanmi/abc/

[2] A. R. Bradley. SAT-based model checking without unrolling. In *Proc. Int'l Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, pp. 70-87, 2011.

[3] W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *J. Symbolic Logic*, 22(3):269-285, 1957.

[4] N. Eén, A. Mishchenko, and R. Brayton. Efficient implementation of property-directed reachability. In *Proc. Int'l Conf. on Formal Methods in Computer Aided Design (FMCAD)*, pp. 125-134, 2011.

[5] N. Eén and N. Sörensson. An extensible SAT-solver. In *Proc. Int'l Conf. on Theory and Applications of Satisfiability Testing (SAT)*, pp. 502-518, 2003.

[6] J.-H. R. Jiang, C.-C. Lee, A. Mishchenko, and C.-Y. Huang. To SAT or not to SAT: Scalable exploration of functional dependency. *IEEE Trans. on Computers*, 59(4):457-467, April 2010.

[7] H.-Y. Liu, Y.-C. Chou, C.-H. Lin, and J.-H. R. Jiang. Towards completely automatic decoder synthesis. In *Proc. Int'l Conf. on Computer-Aided Design (ICCAD)*, pp. 389-395, 2011.

[8] H.-Y. Liu, Y.-C. Chou, C.-H. Lin, and J.-H. R. Jiang. Automatic decoder synthesis: methods and case studies. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 31(9): 1319-1331, September 2012.

[9] K. McMillan. Interpolation and SAT-based model checking. In *Proc. Int'l Conf. on Computer Aided Verification (CAV)*, pp. 1-13, 2003.

[10] M. Moskewicz, C. Madigan, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proc. Design Automation Conference (DAC)*, pp. 530-535, 2001.

[11] S. Shen, Y. Qin, K. Wang, L. Xiao, J. Zhang, and S. Li. Synthesizing complementary circuits automatically. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 29(8):1191-1202, August 2010.

[12] S. Shen, Y. Qin, J. Zhang, and S. Li. A halting algorithm to determine the existence of the decoder. In *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 30(10): 1556-1563, October 2011.

[13] G. Tseitin. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic*, pp. 466-483, 1970.