STG-Level Decomposition and Resynthesis of Speed-Independent Circuits

Ren-Der Chen and Jer-Min Jou

***Abstract*—This paper presents a time-efficient method for the decomposition and resynthesis of *speed-independent* (SI) circuits. Given the specification of an SI circuit, our method first generates its standard *C* implementation. Then, the combinational decomposition is performed to decompose each high-fanin gate that does not exist in the gate library into some available low-fanin gates. The time efficiency of our method is achieved in two ways. First, the *signal transition graph* (STG), whose complexity is polynomial in the worst case, is adopted as our input specification. Second, to reduce the resynthesis cycles, which constitute a major part of the run time, our method first investigates the hazard-free decomposition of each high-fanin gate without adding any signals. Then, for those gates that cannot be decomposed hazard free, two signal-adding methods constructed at the STG level are developed for resynthesis. This decomposition and resynthesis process is iterated until all high-fanin gates are successfully decomposed or no solution can be found. Several experiments have been done on the asynchronous benchmarks and it can be seen from the results that our method largely reduces the run time only at a little more area expense when compared with previous work.**

***Index Terms*—Hazard-free decomposition, resynthesis, signal transition graph (STG), speed-independent (SI) circuit.**

I. Introduction

Speed-independent (SI) circuits are hazard free under the unbounded gate-delay assumption that gate delays are unbounded but finite and wires have zero or negligible delay. For a high-fanin gate that does not exist in the gate library, the only way to make it implementable is to decompose it into some available low-fanin gates. This decomposition must be done with care so that speed-independence is still preserved on the new gates. Some work on the decomposition of SI circuits has been done. The method in [19] deals with a hazard-free decomposition of some gates only; further solution space searching and gate sharing are not considered. In [3], the sequential decomposition is performed on a sequential circuit with high-fanin gates after the correctness conditions are analyzed. However, it lacks an efficient method for using the correctness check and sharing the decomposed gates. An algebraic factorization technique for the combinational and sequential decomposition of complex gates is proposed in [11] and later improved in [7] by using Boolean algebra for logic decomposition and extending the use of *C* elements to different types of storage elements. The last two methods, however, are time-consuming when dealing with large circuits since they are both constructed at the *state graph* (SG) level. Moreover, they add new signals to the initial *signal transition graph* (STG) for each gate to be decomposed, which means more resynthesis cycles are required. Although some area saving can be achieved, much more CPU time has to be spent. Our method overcomes this time-consuming problem by adopting STG as the input specification and reducing the resynthesis cycles during decomposition. Although STG has its limitation in circuit specification, it has the advantage that the problem complexity is only proportional to the number of signals and can thus save a lot of run time. Moreover, due to the high-level property of an STG, it is also easier for designers to describe the circuit behavior. Thanks to these properties of an STG, it has been previously applied in solving the state conflict problem and synthesizing the SI circuits [2], [4], [10], [15], [16], [20]. To our knowledge, however, no work has been constructed at the STG level for the decomposition and resynthesis of SI circuits.

In this paper, given the STG specification of an SI circuit, a standard *C* implementation is first generated. Then, for those high-fanin gates, the combinational decomposition is performed to decompose them into some low-fanin ones. To avoid an explicit generation of all the markings in an STG, some properties peculiar to cubes are analyzed with the help of the *concurrency* [4] and *interleave* [16] relations between signals. Since the resynthesis cycles constitute a major part of the run time, they are reduced in our method by first finding a hazard-free decomposition of each high-fanin gate without adding any signals to the STG. By observing the change of cubes and the firing relation between transitions, the requirements for a hazard-free decomposition are proposed. If no such decomposition exists, the initial STG is modified to incorporate as few new signals as possible for resynthesis. Two signal-adding methods, *intermediate gate acknowledging* (IGA) and *signal replacing* (SR), constructed at the STG level are developed. This decomposition and resynthesis process terminates when all high-fanin gates are successfully decomposed or no further progress can be made. When compared with previous work, our method largely reduces the run time only at a little more area expense.

This paper is organized as follows. Section II gives the background needed in our discussion. To make the whole work done at the STG level, Section III discusses some of the cube properties. The hazard-free decomposition is discussed in Section IV, and for some gates that cannot be successfully decomposed, the resynthesis procedure is given in Section V. Our experimental results are shown in Section VI. Section VII concludes this paper and gives the directions for future development.

Manuscript received September 28, 2000; revised February 1, 2002. This work was supported in part by the National Science Council, Taiwan, R.O.C. under Grant NSC-90-2215-E-006-007. This paper was recommended by Associate Editor R. Sridhar.

The authors are with the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C. (e-mail: crd@j92a21.ee.ncku.edu.tw; jou@j92a21.ee.ncku.edu.tw).

Digital Object Identifier 10.1109/TCSI.2002.805704

II. Background

In this section, we recall some of the basic definitions on *Petri nets* (PNs) [14] and STGs [4]. Then, we discuss the implementation structure adopted in our method and the requirements for the hazard-free implementation of an SI circuit.

A. PNs and STGs

An STG is an interpreted PN $N = (P, T, F, M_0)$, where P is a set of places, T is a set of signal transitions, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation and M_0 is the initial marking. For a node $x \in (P \cup T)$, the fanin and fanout nodes of x are denoted by $\bullet x$ and $x\bullet$, respectively. A *marking* of a PN is an assignment of *tokens* to each place. A transition t is said to be *enabled* in a marking M if each fanin place of t is marked in M. When this enabled transition *fires*, it reaches a new marking M' by removing a token from each place in $\bullet t$ and adding a token to each place in $t\bullet$. A marking M is said to be *reachable* from the initial marking M_0 if there exists a sequence of firing $t_1 t_2 \ldots t_n$, called a *feasible sequence*, that transforms M_0 into M. A *path* in a PN is a sequence of nodes $x_1 x_2 \ldots x_n$ such that any two adjacent nodes x_i and $x_{i+1} (1 \leq i \leq n-1)$ are directly connected, i.e., $(x_i, x_{i+1}) \in F$ and it is called a *cycle* if x_i and x_n are identical. A path or cycle is said to be *simple* if no node appears in it more than once.

A *free-choice* (FC) net is a PN where if two or more transitions share one fanin place p, then p is the only fanin place for all of them. A place is said to be a *multifanin* or *multifanout* place if it has more than one fanin or fanout transition. In an FC net, a multifanout place p is an FC place; when p is marked, only one of its fanout transitions can fire. A PN is *live* if for each reachable marking M and each transition t, another marking where t is enabled can be reached from M. A PN is *safe* if for each reachable marking M and each place p, p is assigned at most one token in M. A *marked graph* (MG) is a PN where each place has exactly one fanin and one fanout transition and a *state machine* (SM) is a PN where each transition has exactly one fanin and one fanout place. It has been proved in [9] that a live and safe FC net can be decomposed into a potentially exponential set of strongly connected MG or SM components that covers the net and each SM contains exactly one token. We assume that only the live and safe FC net is considered in this paper.

For a signal a, the rising and falling transitions of a are denoted by $a+$ and $a-$, respectively, and a^* represents a generic transition. If a changes several times in one cycle of the circuit operation, $a^*_{/i}$ is used to denote the i-th instance of a^*. Transition $a^*_{/i}$ and its next transition $a^*_{/i+1}$ are *adjacent*; there exists no other transitions of a between them. An STG is said to be *output-semimodular* if no output signal transition enabled in any marking can be disabled by other enabled transitions. The implementation of an STG that is not output-semimodular may produce unspecified changes on gate outputs, i.e., hazards. The fanout transitions of an FC place are hence restricted to be transitions of input signals. Each marking M of an STG is encoded by the binary value of each signal and the *characteristic function* for the binary code of M is denoted by $\widehat{M}$. The encoding of a marking M is said to be *consistent* if there exists no rising or falling transition of a signal a in M when the corresponding value of a is 1 or 0 in $\widehat{M}$. An STG is said to satisfy the *complete state coding* (CSC) property, if, for any two different markings with the same binary code, the output signal transitions enabled in them are identical. A CSC violation means that the circuit being in the same state has to produce different transitions on gate outputs. These conflict states can be distinguished in the circuit by an additional memory [5]. A more restrictive condition, called the *unique state coding* (USC) property, requires that each reachable marking be assigned a unique binary code. An STG satisfying the output-semimodularity, consistency and CSC properties is called an *implementable* STG, from which a correct SI circuit can be derived [12].

Two transitions a_1^* and a_2^* are said to be *concurrent* if they can fire in the same marking without disabling each other and they are *autoconcurrent* [16] if they are of the same signal. Signal a_1 is said to be concurrent to a transition a_2^* if there exists some transition of a_1 concurrent to a_2^*. For a marking where a transition a^* is enabled and a place p is marked, if a^* can fire without removing the token in p, a^* is concurrent to p. For two transitions x_1 and x_2 (or a transition x_1 and a place x_2), a node $x \in P \cup T$ is said to be *interleaved* with (x_1, x_2) if x_1 and x_2 are nonconcurrent to each other and x exists in at least one of the simple paths from x_1 to x_2. Similarly, a set of nodes X is said to be interleaved with (x_1, x_2) if each node in X is interleaved (x_1, x_2). To be more general, for two node sets X_1 and X_2, X is said to be interleaved with (X_1, X_2) if for each $x_i \in X_1$ and $x_j \in X_2$, X is interleaved with (x_i, x_j). A transition a_1^* is called a *trigger* transition (and its underlying signal a trigger signal) of a transition a_2^* if there exists a place p belonging to $(a_1^*)\bullet$ and $\bullet(a_2^*)$ and a_1^* is said to be *persistent* to a_2^* if the underlying signal a_1 is nonconcurrent to a_2^*. A transition a^* is said to satisfy the persistency property if each trigger transition of a^* is persistent to a^*. This persistency property is a necessary condition for an STG to be realized by the standard *C* implementation [12]. In our work, the input specification is hence assumed to be an implementable STG satisfying persistency, useless in some cases where the standard *C* implementation can be reduced to a combinational implementation. Fig. 1(a) shows the STG specification of a benchmark example *master-read.g*; it is an MG containing the master read operation of Intel Multibus (IEEE Standard 796). Fig. 1(b) is the STG specification of another example *sbuf-send-pkt2.g*, obtained from the Post Office chip [8]; it is a free-choice net composed of four MGs.

B. Implementation Structure

Each gate in our work is assumed to be *atomic* and satisfies the pure-delay assumption [13]. An atomic gate can be modeled as an instantaneous Boolean function of its inputs with a single pure delay on the output. Given any input change of an atomic gate, it responds with a corresponding output change after some (potentially unbounded) delays. The relative ordering of signal transitions on the inputs is hence preserved irrespective of the actual gate delay. Our implementation structure, shown in Fig. 1(c), is based on a restricted class of asynchronous circuits called the *standard C implementation* [12]. Each noninput signal is implemented by a *signal network* that consists of the *C* element and the combinational *set* and *reset networks*. A gate is *enabled* or *disabled* if its output is ready for a rising or falling

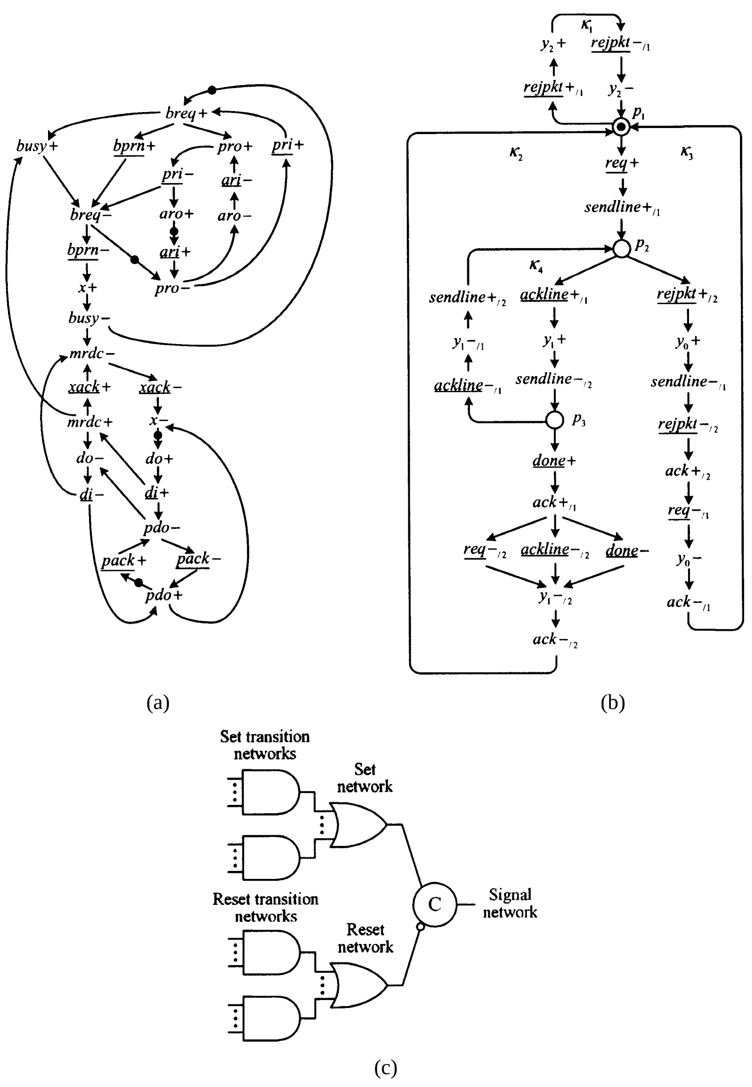

Fig. 1. (a) STG example *master-read.g*. (b) STG example *sbuf-send-pkt2.g*. (c) Standard *C* implementation.

change. The set or reset network, composed of *set* or *reset transition networks*, is responsible for enabling or disabling the *C* element. For a signal a, if the persistency property is satisfied for each rising transition of a and the set and reset functions of a are complements of each other, a can just be realized by the combinational set network. It is hence not necessary for each falling transition of a to satisfy persistency. Similarly, it is also possible for a to be realized by the combinational reset network. For the reason of circuit implementability or area saving, a set or reset transition network may sometimes be shared by multiple instances of the rising or falling transition. The discussion of transition network sharing, however, is omitted here due to space limitations.

C. Hazard-Free Implementability

A *hazard* is an unexpected output transition, either rising or falling, in response to a change in some input(s). An output transition is said to be *expected* if it occurs in accordance with the STG specification. In an SI circuit, due to the lack of a global clock and the pure delay assumption, no internal transition is allowed to occur unexpectedly. An unexpected internal transition will be propagated to the output, causing an unexpected output transition, i.e., hazard. Since an internal transition cannot be perceived by the outside environment, it must be *acknowledged* by some expected output transition(s). A transition a_1^* is said to be *acknowledged* by a transition a_2^* if there exists a gate with a_1 as one of its inputs and a_2 as its output, satisfying that only after a_1^* is *completed* will a_2^* be completed. The completion of a signal transition means that the signal has changed to the new value from its original value. This acknowledgment property can also be further extended to be applied to signals not belonging to the same gate.

For the standard *C* implementation, the enabling or disabling of a transition network is modeled by the *set-on* or *set-off* operation of a cube. A *cube* is a set of literals representing the logic value of each signal in the original or complementary form. A cube c is said to be *set on* in a marking M or c *covers* M, i.e., $c \supseteq \widehat{M}$, if each literal a or $\bar{a}$ has a corresponding binary value 1 or 0 in $\widehat{M}$; otherwise, c is *set off* in M, i.e., $c \cap \widehat{M} = \phi$. To obtain the hazard-free conditions, several regions must first be defined. For a transition $a^*_{/i}$, the *excitation region* for $a^*_{/i}$, denoted by $\mathrm{ER}(a^*_{/i})$, is the maximal connected set of markings where $a^*_{/i}$ is enabled. All the markings in $\mathrm{ER}(a^*_{/i})$ are covered by a *cover cube* $c(a^*_{/i})$, which is then implemented by a single AND gate.

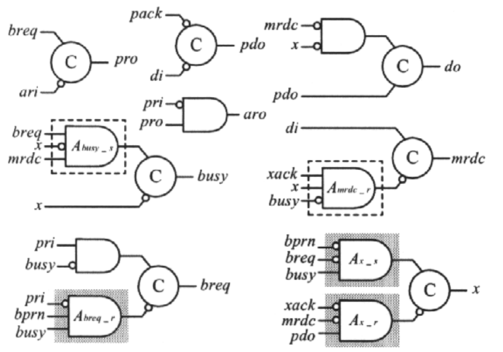

Fig. 2. A hazard-free implementation of *master-read.g*.

The *quiescent region* (QR) for $a^*_{/i}$, denoted by QR$(a^*_{/i})$, is the maximal connected set of markings reached from ER$(a^*_{/i})$ before $a^*_{/i+1}$ is enabled. It has been shown in [1] and [12] that the resulting circuit operates with SI without any hazards if each cover cube $c(a^*_{/i})$ is a *monotonous cover* (MC) satisfying the following conditions.

1) *Cover condition*: $c(a^*_{/i})$ covers all the markings in ER$(a^*_{/i})$, i.e., $c(a^*_{/i}) \supseteq \widehat{\mathrm{ER}}(a^*_{\subset})$.
2) *Monotonicity condition*: In every sequence of markings inside ER$(a^*_{/i}) \cup$ QR$(a^*_{\subset})$, $c(a^*_{/i})$ changes at most once.
3) *One-hot condition*: $c(a^*_{/i})$ does not cover any reachable markings outside ER$(a^*_{/i}) \cup$ QR$(a^*_{/i})$.

For an MC $c(a^*_{/i})$, each signal in $c(a^*_{/i})$ is either a trigger or a context signal of $a^*_{/i}$. A transition a_1^* is a *context* transition (and its underling signal a context signal) of a transition a_2^* if signal a_1 is nonconcurrent to a_2^* and any transition that is interleaved with (a_1^*, a_2^*) is not of the same signal as a_1 or a_2. Owing to the characteristic of a *C* element that the two input values are allowed to be identical at the same time, the markings covered by $c(a^*_{/i})$, under some constraints, may be extended to include the markings in QR$(a^*_{/i-1})$ [16] for circuit implementability or area saving. Since the outputs of the first-level AND gates in the standard *C* implementation are one-hot encoded, the SI property is still preserved if any valid Boolean decomposition is applied to the second-level OR gates [19]. Fig. 2 shows a hazard-free implementation of *master-read.g*. There are five three-input AND gates having to be decomposed if only two-input gates are available in the gate library.

III. Cube Properties for Processing at STG Level

To perform our decomposition and resynthesis work at the STG level, some cube properties are analyzed in this section. The formulation is borrowed from [16] with some adjustments to make it more suitable for our discussion.

A. Finding Markings From Cubes

We first discuss how to find markings with the same property from a cube. For a place p and a transition $a^*_{/i}$, there are the following three types of markings that we are concerned about:

1) the markings in which p is marked;
2) the markings in which $a^*_{/i}$ is enabled;
3) the markings directly reached by the firing of $a^*_{/i}$.

To avoid generating all the markings of an STG, whose complexity is usually exponential in the worst case, markings of the same type are captured by a single cube. A cube is only a conservative approximation for the binary codes of the markings to be captured and may cover markings that do not exist. To make the cube estimate more accurate, it should be the smallest among those possible, i.e., with the largest number of literals [12]. To find the cube, we need the relation between any two nodes in an STG. The deriving of this relation is based on the utilization of the *cycle*, *MG*, and SM decomposition of the free-choice STG [4, 16]. For any two nodes x_1 and x_2 in an FC net N, they are *ordered* if there exists a simple cycle in N to which both of them belong. They are *concurrent* if they are not ordered and there exists an MG component in N to which both of them belong. Lastly, they are conflicting if they are not ordered and there exists an SM component in N to which both of them belong. If N is an MG each SM component in N will be a simple cycle. This net decomposition, in some cases, cannot provide enough information to determine the relation. If x_1 and x_2 are not ordered and there exists no MG component or SM component to which both of them belong, their relation (either concurrent or conflicting) can be obtained by the relation of their fanin and fanout nodes. A signal a is conflicting with a place p if each transition of a is conflicting with p; is concurrent to p if there exists some transition of a concurrent to p; ordered with p if a is neither conflicting with nor concurrent to p.

For a place p, the set of markings in which p is marked is called a *marked region* (MR) [16] of, denoted by MR(p). A cube is then defined to cover all the markings in MR(p).

Definition 3.1: The *place cube* of a place p, denoted by $c_p(p)$, is the smallest cube that covers all the markings in MR(p), i.e., $c_p(p) \supseteq \widehat{\mathrm{MR}}(p)$.

If p is a fanin place of a transition a $a^*_{/i}$, $c_p(p)$ is called a *fanin place cube* of $a^*_{/i}$; if p is a fanout place of $a^*_{/i}$, $c_p(p)$ is a *fanout place cube* of $a^*_{/i}$. Each literal in $c_p(p)$ can be determined by the binary value of the corresponding signal when p is marked. Any signal that is concurrent to p does not have a literal in $c_p(p)$; i.e., it is a *don't care*. For a signal a ordered with p, the value of a can be determined by the interleave relation between p and a pair of adjacent transitions of a. That is, if p is interleaved with $(a+_{/i}, a-_{/i+1})$, the value of a is 1 in $\widehat{\mathrm{MR}}(p)$ and the literal is a in $c_p(p)$; if p is interleaved with $(a -_{-/i}, a+_{/i+1})$, the value of a is 0 in $\widehat{\mathrm{MR}}(p)$ and the literal is $\bar{a}$ in $c_p(p)$. As shown in [16], this interleave relation gives a polynomial-time algorithm to determine each signal literal of a place cube. Take transition $\mathrm{mrdc}-$ in *master-read.g* as an example, let p_1, p_2 and p_3 be its three fanin places corresponding to trigger transitions $\mathrm{xack}+$, $di-$ and $\mathrm{busy}-$, respectively. Since there are only two signals xack and mrdc ordered with p_1 and p_2, (interleaved with $\mathrm{xack}+$, $\mathrm{xack}-$ and $(\mathrm{mrdc}+, \mathrm{mrdc}-)$, $c_p(p_1)$ can be determined as $\mathrm{xack}\ \mathrm{mrdc}$. Similarly, $c_p(p_2)$ and $c_p(p_3)$ can be determined as $\overline{di}\ \mathrm{mrdc}\ \overline{\mathrm{do}}$ and $\overline{\mathrm{busy}}\ \mathrm{mrdc}\ \mathrm{x}$, respectively.

For a signal a conflicting with a place p, there is no transition of a in the MG κ that contains p. The literal of a in $c_p(p)$ can be derived from $c_p(p')$ of another multifanout place p' in κ. If p' is still conflicting with a, this literal-finding process has to be recursively applied to p' until a multifanout place p'' that is not conflicting with a is found. The literal of a in $c_p(p)$ can hence be obtained from $c_p(p'')$. In *sbuf-send-pkt2.g*, for example, let p be the place with fanin transition $ackline{-}_{/1}$ and fanout transition $y_1{-}_{/1}$. Since p (in MG κ_4) is conflicting with signal y_2 (in MG κ_1), the literal of y_2 in $c_p(p)$ can be obtained from $c_p(p_2)$. However, since p_2 is also conflicting with y_2, the literal of y_2 in $c_p(p_2)$ has to be further derived from $c_p(p_1)$. Since p_1 is ordered with y_2 and interleaved with (y_2-, y_2+) in κ_1, the literal of y_2 in $c_p(p)$ can hence be determined as $\overline{y}_2$.

For a transition $a^*_{/i}$, its excitation region ER$(a^*_{/i})$ is the set of markings where each fanin place of $a^*_{/i}$ is marked and can hence be defined as the intersection of MRs of all its fanin places, i.e., ER$(a^*_{/i}) = \cap_{p \in \bullet(a^*_{/i})}MR(p)$. After a $a^*_{/i}$ fires, a new marking region is reached.

Definition 3.2: The *entry quiescent region* EQR[1] of a transition $a^*_{/i}$, denoted by EQR$(a^*_{/i})$, is the maximal connected set of markings directly reached by the firing of $a^*_{/i}$.

This is a set composed of markings where each fanout place of $a^*_{/i}$ is marked and can also be defined as the intersection of MRs of all the fanout places of $a^*_{/i}$, i.e., ER$(a^*_{/i}) = \cap_{p \in (a^*_{/i})\bullet}MR(p)$. Two cubes are then defined to cover all the markings in ER$(a^*_{/i})$ and EQR$(a^*_{/i})$, respectively.

Definition 3.3: The *fanin cube* of a transition $a^*_{/i}$, denoted by $c_{\text{fin}}(a^*_{/i})$, is the smallest cube that covers all the markings in ER$(a^*_{/i})$, i.e., $c_{\text{fin}}(a^*_{/i}) \supseteq \widehat{\text{ER}}(a^*_{/i})$; the *fanout cube* of $a^*_{/i}$, denoted by $c_{\text{fout}}(a^*_{/i})$, is the smallest cube that covers all the markings in EQR$(a^*_{/i})$, i.e., $c_{\text{fout}}(a^*_{/i}) \supseteq \widehat{\text{EQR}}(a^*_{/i})$.

The fanin and fanout cubes of $a^*_{/i}$ can be obtained by intersecting all the fanin and fanout place cubes of $a^*_{/i}$, respectively, i.e., $c_{\text{fin}}(a^*_{/i}) = \cap_{p \in \bullet(a^*_{/i})} c_p(p)$ and $c_{\text{fout}}(a^*_{/i}) = \cap_{p \in (a^*_{/i})\bullet} c_p(p)$. In addition, $c_{\text{fout}}(a^*_{/i})$ differs from $c_{\text{fin}}(a^*_{/i})$ only in the literal of a, thus can also be derived from $c_{\text{fin}}(a^*_{/i})$ by complementing this literal. Take transition $\text{mrdc}-$ of *master-read.g* as an example, its fanin cube can be obtained by intersecting its three fanin place cubes xack mrdc, $\overline{di}\ \text{mrdc}\ \overline{\text{do}}$ and $\overline{\text{busy}}\ \text{mrdc x}$. i.e., $c_{\text{fin}}(\text{mrdc}-) = \text{xack}\ \overline{\text{di}}\ \overline{\text{busy}}\ \text{mrdc}\ \overline{\text{do}}\ \text{x}$. Then, its fanout cube can be obtained by complementing the literal of mrdc in $c_{\text{fin}}(\text{mrdc}-)$, i.e., $c_{\text{fout}}(\text{mrdc}-) = \text{xack}\ \overline{\text{di}}\ \overline{\text{busy}}\ \overline{\text{mrdc}}\ \overline{\text{do}}\ \text{x}$.

B. Changes of Cubes

How a cube is changed by the firing of transitions is now discussed. For two markings M_1 and M_2, a marking M is said to be interleaved with (M_1, M_2) if M will always be reached before M_2 when the marking goes from M_1 to M_2. For a marking M_1 where a cube c is set off and a marking M_2 where c is set on, c will be *turned on* if the marking goes from M_1 to M_2 and c is set off in any marking interleaved with (M_1, M_2). Conversely, if the marking goes from M_2 to M_1 and c is set on in any marking interleaved with (M_2, M_1), c will be *turned off*. For a cube c to be set on in a marking M, *each* signal a with a binary value 1 in $\widehat{M}$ must have a corresponding literal a or a *don't care* in c, whereas only *one* signal a with a literal $\overline{a}$ in c is sufficient to make c set off. It can be similarly stated if the binary value of a is 0 in $\widehat{M}$. The condition for turning on a cube, which is constructed on the literal correspondence of all signals, is hence much stricter than that for turning off a cube, which requires only one literal correspondence.

Definition 3.4: A transition set T is a *turn-on set* of a cube c if c will always be turned on by firing all the transitions in T; T is a *turn-off set* of c if c will always be turned off by firing any one of the transitions in T.

Let T be composed of m transitions $t_1, t_2, \ldots, t_m$. If T is a turn-on set of a cube c, then for a marking M where c is set off and each transition in T is enabled, another marking where c is set on will be reached from M by firing all the transitions in T. However, c is still set off in any marking reached from M by firing only some (not all) of the transitions in T, i.e., $c \cap (\cup_{i=1}^{i=m} \widehat{\text{ER}}(t_i)) = \phi \wedge c \supseteq (\cap_{i=1}^{i=m} (\widehat{\text{EQR}}(t_i))$. If T is a turn-off set of c, then for a marking M where c is set on and each transition in T is enabled, another marking where c is set off will be directly reached from M by firing any one of the transitions in T, i.e., $c \supseteq (\cap_{i=1}^{i=m} (\widehat{\text{ER}}(t_i)) \wedge c \cap (\cup_{i=1}^{i=m} (\widehat{\text{EQR}}(t_i)) = \phi$. How the firing of a transition t contributes to the turn-on or turn-off change of a cube c is analyzed here by indicating the covering relations between c and the fanin/fanout cubes of t. For two cubes c_1 and c_2, there are three covering relations involved.

Relation 1: c_1 covers c_2, i.e., $c_1 \supseteq c_2$.

Relation 2: the intersection of c_1 and c_2 is empty, i.e., $c_1 \cap c_2 = \phi$.

Relation 3: the intersection of c_1 and c_2 is not empty, but c_1 does not cover c_2, i.e., $c_1 \cap c_2 = \phi \wedge c_1 \supsetneq c_2$.

If c_1 covers c_2, c_1 covers all the markings covered by c_2; if their intersection is empty, there is no common marking covered by them; if their intersection is not empty, but c_1 does not cover c_2, there exists some common marking covered by them and some marking covered by c_2 but not by c_1. Another relation "c_2 covers c_1" is excluded in our discussion since it is synonymous to the first relation "c_1 covers c_2" when the roles of c_1 and c_2 are interchanged. For a transition t, since its fanin cube $c_{\text{fin}}(t)$ covers all the markings in ER(t), if a cube c covers $c_{\text{fin}}(t)$, c also covers all the markings in ER(t). If the intersection of c and $c_{\text{fin}}(t)$ is empty, c does not cover any marking in ER(t). If their intersection is not empty, but c does not cover $c_{\text{fin}}(t)$, there exists in ER(t) some marking covered by c and some marking not covered by c. These covering relations between c and the fanin cube of t can also be applied to c and the fanout cube of t.

By the above three covering relations, for a cube c, considering its covering relations with $c_{\text{fin}}(t)$ and $c_{\text{fout}}(t)$ of one of its transitions t, there are nine cases as to what influence the firing of t has on the turn-on or turn-off change of c, as shown in Table I. The transition t whose underlying signal literal is a *don't care* in c is excluded in our discussion since the firing of t does not have any influence on c. In case 1, since the firing of t does not make any change on c, t is called a *static* transition of c. In case 2, since c is turned on by the firing of t, t is called a *turn-on transition* of c and constitutes a turn-on set of c. In case 3, since c is turned off by the firing of t, t is called a *turn-off transition* of c and also constitutes a turn-off set of c. In case 4, since c can be turned on or kept at off by the firing of t,

[1]This is also called switching region in [21].

TABLE I
COVERING RELATIONS BETWEEN A CUBE c AND THE FANIN/FANOUT CUBES OF ONE ITS TRANSITIONS t

Case	Covering relations		Possible change of cube c when t fires
	$c_{fin}(t)$	$c_{fout}(t)$	
1	*Rel. 2*	*Rel. 2*	*Kept at off*
2	*Rel. 2*	*Rel. 1*	*Turned on*
3	*Rel. 1*	*Rel. 2*	*Turned off*
4	*Rel. 2*	*Rel. 3*	*Turned on or Kept at off*
5	*Rel. 3*	*Rel. 2*	*Turned off or Kept at off*
6	*Rel. 1*	*Rel. 1*	*Null*
7	*Rel. 1*	*Rel. 3*	*Null*
8	*Rel. 3*	*Rel.1*	*Null*
9	*Rel. 3*	*Rel.3*	*Null*

t is called a *quasiturn-on (Q-on) transition* of c. Similarly in case 5, since c can be turned off or kept at off by the firing of t, t is called a *quasiturn-off (Q-off) transition* of c. In the above two cases, there exists other Q-on or Q-off transitions concurrent to t and these concurrent transitions (including t) can be properly combined to make a turn-on or turn-off set of c. In case 6, c is set on in both $\mathrm{ER}(t)$ and $\mathrm{EQR}(t)$. This is a case that never occurs since if the underlying signal of t has a literal a in c, the binary value of a is 1 in $\widehat{\mathrm{ER}}(t)$. After t fires, due to the consistency property of the STG, the value of a is changed to 0 in $\widehat{\mathrm{EQR}}(t)$, where c is obviously set off. It can be similarly explained if the literal of the underlying signal of t is $\overline{a}$ in c. Besides, the remaining three cases 7–9 will not occur, either; there exists a marking M_1, where c is set on and another marking M_2 reached from M_1 by the firing of t, where c is also set on.

C. Concurrent Set

Here, we show how to derive the turn-on and turn-off sets of a cube c by the concurrency relation between transitions. Let $\psi(c)$ be a set composed of all the turn-on, turn-off, Q-on and Q-off transitions of c.

Definition 3.5: A *concurrent set* of a cube c is a subset of $\psi(c)$ where any two transitions are concurrent to each other.

A concurrent set is said to be *maximal* if it is not a subset of any other concurrent set. A concurrent set composed of only one turn-on or turn-off transition is a maximal concurrent set with the smallest number of transitions. If a concurrent set is composed only of Q-on transitions, it is a *Q-on set*; if it is composed only of Q-off transitions, it is a *Q-off set*. Then we have the following conclusions. Due to space limitations, all the theorems in this paper are presented without proof.

Theorem 3.1: Let T be a Q-on set of a cube c, then T is also a turn-on set of c if and only if T is maximal.

Theorem 3.2: Let T be a Q-off set of a cube c, then T is also a turn-off set of c if and only if T is maximal.

Since a Q-on transition may be concurrent to a Q-off transition, a maximal concurrent set may contain both Q-on and Q-off transitions.

Theorem 3.3: Let T be a maximal concurrent set of a cube c composed of Q-on and Q-off transitions at the same time, then T is neither a turn-on nor a turn-off set of c and there exists unexpectable turn-on and turn-off changes on c during the firing process.

TABLE II
PROPERTIES FOR THE TRANSITIONS OF CUBE $\mathrm{xack}\ \bar{\mathrm{d}}\mathrm{i}\ \mathrm{b\bar{u}sy}$

Tran	Fanin cube	Fanout cube	Role	Concurrent transitions
busy$-$	*busy*	$\overline{busy}$	*Q-on*	*xack*$+$, *di*$-$
di$-$	*di*	$\overline{di}$	*Q-on*	*xack*$+$, *busy*$+$, *busy*$-$
xack$+$	$\overline{xack}$	*xack*	*Q-on*	*di*$-$, *busy*$+$, *busy*$-$
busy$+$	$\overline{busy}$	*busy*	*Q-off*	*xack*$+$, *di*$-$
xack$-$	$xack\ \overline{di}\ \overline{busy}$	$\overline{xack}\ \overline{di}\ \overline{busy}$	*Turn-off*	ϕ
di$+$	$\overline{xack}\ \overline{di}\ \overline{busy}$	$\overline{xack}\ di\ \overline{busy}$	*Static*	ϕ

Definition 3.6: A maximal concurrent set is said to be *well behaved* if it does not contain Q-on and Q-off transitions at the same time.

For an SI circuit, if there exists a gate whose underlying cube contains a maximal concurrent set that is not well behaved, there may exist hazards on this gate. A cube is said to be well behaved if all its maximal concurrent sets are well behaved. Take a cube $c = \mathrm{xack}\ \overline{\mathrm{di}}\ \overline{\mathrm{busy}}$ of *master-read.g* as an example, Table II gives some properties for the transitions of c and the roles they play. The fanin and fanout cubes of c are represented only by literals related to c. Using the concurrent relation between transitions, three maximal concurrent sets can be obtained, i.e., a turn-on set $T_1 = \{\mathrm{xack}+, \mathrm{di}-, \mathrm{busy}-\}$, a turn-off set $T_2 = \{\mathrm{xack}-\}$ and a set $\{\mathrm{xack}-, \mathrm{di}-, \mathrm{busy}-\}$ that is not well behaved. After c is turned on by firing all the transitions in T_1, it will be turned off by the firing of $\mathrm{xack}-$ in T_2. Then, if $\mathrm{xack}+$ and $\mathrm{di}-$ fire before $\mathrm{busy}+$, c will be turned on again and then turned off by $\mathrm{busy}+$; otherwise, there is no change on c until all the transitions in T_1 are enabled again in the next operation cycle.

After classifying all the transitions in $\psi(c)$ into turn-on or turn-off set for a well-behaved cube c, the disjoint relation between maximal concurrent sets has to be discussed.

Theorem 3.4: Let $\mathbf{\Psi}(c)$ be a collection of all the maximal concurrent sets of a well-behaved cube c, then any two sets in $\mathbf{\Psi}(c)$ are disjoint.

Since any two maximal concurrent sets of a well-behaved cube c are disjoint, each transition in $\psi(c)$ belongs to exactly one turn-on or turn-off set in $\mathbf{\Psi}(c)$ and contributes to at most one turn-on or turn-off change of c in one cycle of the circuit operation. Each turn-on or turn-off set of c, therefore, has a one-to-one correspondence with a turn-on or turn-off change of c. These cube properties can then be applied in checking whether a cover cube $c(a^*_{/i})$ is a MC for a transition $a^*_{/i}$; i.e., the following three requirements must be met:[2]

1) $c(a^*_{/i})$ is well behaved;
2) there exists exactly one turn-on set of $c(a^*_{/i})$ composed of all the trigger transitions of $a^*_{/i}$;
3) there exists exactly one turn-off set of $c(a^*_{/i})$ interleaved with $(a^*_{/i}, a^*_{/i+1})$.

If $c(a^*_{/i})$ is well behaved, there is no unexpectable change of $c(a^*_{/i})$. If there exists exactly one turn-on set of $c(a^*_{/i})$ composed of all the trigger transitions of $a^*_{/i}$, $c(a^*_{/i})$ will be turned on by firing all these trigger transitions; i.e., $c(a^*_{/i})$ will be set on in $\mathrm{ER}(a^*_{/i})$. This meets the cover condition that $c(a^*_{/i})$ covers all the markings in $\mathrm{ER}(a^*_{/i})$. If there exists exactly one turn-off set

[2]The gate sharing for multiple instances of transitions is omitted here due to space limitations.

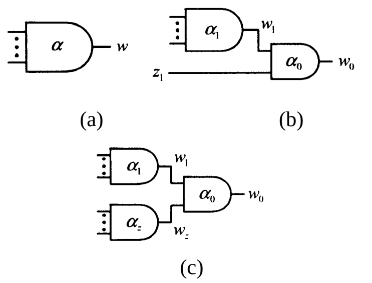

Fig. 3. (a) A high-fanin gate α to be decomposed. (b) A two-level decomposition of α. (b) Another two-level decomposition of α.

of $c(a^*_{/i})$ interleaved with $(a^*_{/i}, a^*_{/i+1})$, $c(a^*_{/i})$ will be turned off before the firing of $a^*_{/i+1}$. After $c(a^*_{/i})$ is turned off, it will be kept unchanged until all the trigger transitions of $a^*_{/i}$ are enabled again in the next operation cycle. This meets the remaining monotonous and one-hot conditions.

IV. Hazard-Free Decomposition

For a high-fanin gate that does not exist in the gate library, the only way to make it implementable is to decompose it into some available low-fanin gates. This implicitly introduces new signals to the circuit and each of these signals must be properly acknowledged or hazards may result. As mentioned in Section II-C, no hazard occurs when an OR gate is decomposed, whereas for an AND gate, this must be done with care. In the following discussion, gate decomposition hence refers to the decomposition of AND gates and we assume only two-input gates are available in the gate library. Our method basically consists of the following two steps:

1) decompose each high-fanin AND gate hazard-freely into some 2-input gates and if necessary;
2) generate a new STG and perform resynthesis, then return to the first step.

This process is iterated until all high-fanin gates are decomposed or no solution can be found. Our search method is basically greedy and heuristic; an exhaustive enumeration with backtracking is sometimes necessary to find the solution.

A. Decomposition Hazards

Let α be the high-fanin gate in the transition network of a transition $a^*_{/i}$ to be decomposed [see Fig. 3(a)]. When it is decomposed into two-level gates, two cases exist for this. One is a lower-fanin *intermediate gate* α_1 cascaded with a two-input *final gate* α_0 and the other is two lower fanin intermediate gates α_1 and α_z cascaded with a two-input final gate α_0, as shown in Fig. 3(b)–(c), respectively. In Fig. 3(b), hazards may occur if there exists any unacknowledged transition of w_1 or w_0. The unacknowledged transition of w_0 is caused by that of w_1, which cannot be acknowledged by an expected transition of w_0. An expected transition of w_0 is a transition having the same behavior as that of w in Fig. 3(a). No hazard occurs on α_1 if each transition of w_1 can be acknowledged by an expected transition of w_0. Similarly, to make the decomposition in Fig. 3(c) free from hazards, each transition of w_1 and w_z must also be acknowledged by an expected transition of w_0.

Here, we show by an example of how improper decomposition can lead to hazards and why the knowledge of the environment is necessary for a hazard-free decomposition. For gate $A_{\mathrm{busy\_s}}$ in Fig. 2, there are three ways to decompose it into two-level gates, as shown in Fig. 4(a)–(c). The corresponding timing diagrams are also depicted in Fig. 4(d)–(f), respectively. There are unacknowledged transitions in the first two decompositions. Only the third decomposition is hazard-free; each transition of w_1 can be acknowledged by an expected transition of w_0. We can see from this example that the decomposition of an AND gate cannot be done at will; the environment behavior must be taken into consideration during decomposition to make each new transition acknowledged.

B. Hazard-Free Conditions

To make the decomposition in Fig. 3(b) free from hazards, each transition of w_1 must be acknowledged by a corresponding transition of w_0. Here the enabling or disabling of a gate is modeled as the turn-on or turn-off change of the corresponding cube. The cubes of gates α, α_1, α_0 and α_z in Fig. 3 are represented by, c_α, c_1, c_0 and c_z, respectively. The condition for acknowledging the rising transition of w_1 in Fig. 3(b) is first presented.

Theorem 4.1: The rising transition of w_1 can be acknowledged if and only if the intermediate cube c_1 is well behaved and the number of turn-on sets of c_1 is less than or equal to that of c_α in any MG.

Since the turn-on and turn-off changes of a cube alternate in an STG during the firing process, for an MG containing exactly one turn-on set of c_1 and c_α, it also contains exactly one turn-off set of them. This, however, does not mean that the falling transition of w_1 can also be acknowledged; another requirement must be met. For a decomposition where the rising transition of w_1 is acknowledged, the condition on which the falling transition of w_1 can also be acknowledged is given in the following theorem. Since there is no acknowledgment problem of w_1 in an MG containing no turn-off set of c_1, only the MG containing exactly one turn-off set of c_1 is discussed.

Theorem 4.2: The falling transition of w_1 can also be acknowledged if and only if for any MG κ that contains exactly one turn-off set of the intermediate cube c_1, the primary input z_1 is kept unchanged between the firing of $a^*_{/i}$ and its next transition $a^*_{/i+1}$ in κ.

For an MG κ containing $a^*_{/i}$ and $a^*_{/i+1}$, there exists two cases where z_1 can be kept unchanged between the firing of $a^*_{/i}$ and $a^*_{/i+1}$: (1) $z^*_{1/i}$ and $z^*_{1/i+1}$ do not exist in κ; (2) $a^*_{/i+1}$ is interleaved with $(a^*_{/i}, z^*_{1/i+1})$. Now we illustrate the above two theorems by the example in Fig. 4. The turn-on and turn-off sets of each intermediate cube c_1 in Fig. 4(a)–(c) are given in Table III. For $c_1 = \mathrm{breq\ mrdc}$ in Fig. 4(a), since $\mathrm{busy}-$ is not interleaved with $(\mathrm{busy}+, \mathrm{x}+)$, the falling transition of w_1 cannot be acknowledged. And since c_1 contains a maximal concurrent set that is not well behaved, there exists unexpectable changes on α_1. For $c_1 = \mathrm{mrdc}\ \bar{\mathrm{x}}$ in Fig. 4(b) and $c_1 = \mathrm{breq}\ \bar{\mathrm{x}}$ in Fig. 4(c), since they are well behaved and contain exactly one turn-on and one turn-off set, the rising transition of w_1 can be acknowledged by that of w_0. However, since $\mathrm{busy}-$ is not interleaved with $(\mathrm{busy}+, \mathrm{breq}-)$, the falling transition of w_1 in Fig. 4(b) still cannot be acknowledged. But in Fig. 4(c),

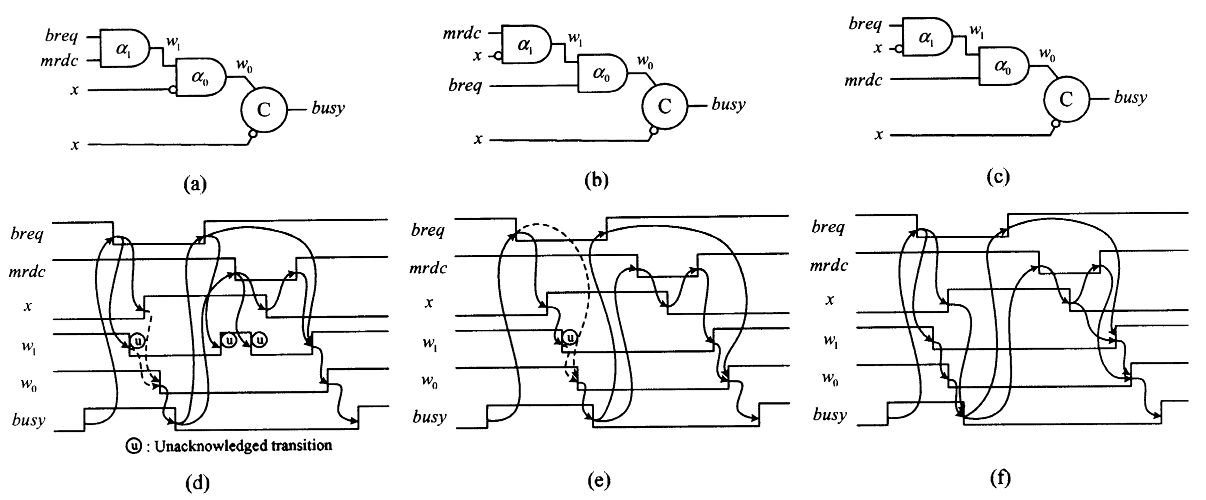

Fig. 4. Three possible decompositions of gate in $A_{\text{busy\_s}}$ Fig. 2. (a) First bad decomposition. (b) Second bad decomposition. (c) Good decomposition. (d)–(f) The corresponding timing diagrams.

TABLE III
TURN-ON AND TURN-OFF SETS OF EACH INTERMEDIATE CUBE IN FIG. 4(a)–(c)

Cube	Turn-on transitions	Turn-off transitions	Q-on transitions	Q-off transitions	Turn-on sets	Turn-off sets	Not well-behaved maximal concurrent sets
breq mrdc	ϕ	*breq−*	*mrdc+*, *breq+*	*mrdc−*	{ *mrdc+*, *breq+* }	{ *breq−* }	{ *mrdc−*, *breq+* }
mrdc $\overline{x}$	*mrdc+*	*x+*	ϕ	ϕ	{ *mrdc+* }	{ *x+* }	ϕ
breq $\overline{x}$	ϕ	*breq−*	*breq+*, *x−*	ϕ	{ *breq+*, *x−* }	{ *breq−* }	ϕ

since $\text{busy}-$ is interleaved with $(\text{busy}+, \text{mrdc}-)$, the falling transition of w_1 can be acknowledged by that of w_0.

For the decomposition in Fig. 3(c), to make the rising transitions of w_1 and w_z acknowledged, constraints imposed on the intermediate cube c_1 in Fig. 3(b) must also be satisfied for the two intermediate cubes c_1 and c_z in Fig. 3(c); i.e., c_1 and c_z must be well behaved and the number of turn-on sets of c_1 and c_z must be less than or equal to that of c_α in any MG. Moreover, to make the falling transitions of w_1 and w_z also acknowledged, the constraints imposed on the primary input z_1 in Fig. 3(b) must be satisfied for both w_1 and w_z. To satisfy the constraint for w_z, for an MG κ that contains a turn-on set $T_{\text{on}}(c_1)_{/i}$ and the next turn-off set $T_{\text{off}}(c_1)_{/i+1}$ of c_1, c_z must be kept unchanged between the firing of $a^*_{/i}$ and $a^*_{/i+1}$ in κ. If κ does not contain any turn-on or turn-off set of c_z, c_z is sure to be kept unchanged; the falling transition of w_1 can hence be acknowledged by that of w_0. Since the roles of gates α_1 and α_z are interchangeable, now the falling transition of w_z can also be acknowledged. On the other hand, if κ also contains a turn-on set $T_{\text{on}}(c_z)_{/i}$ and the next turn-off set $T_{\text{off}}(c_z)_{/i+1}$ of c_z, $a^*_{/i+1}$ must be interleaved with $(a^*_{/i}, T_{\text{off}}(c_z)_{/i+1})$ to make the falling transition of w_1 acknowledged. Similarly, to satisfy the constraint for w_1, $a^*_{/i+1}$ must also be interleaved with $(a^*_{/i}, T_{\text{off}}(c_1)_{/i+1})$ to make the falling transition of w_z acknowledged. This does not work since satisfying these two constraints will result in the fact that neither $T_{\text{off}}(c_z)_{/i+1}$ nor $T_{\text{off}}(c_1)_{/i+1}$ is interleaved with $(a^*_{/i}, a^*_{/i+1})$, making it impossible for the final cube c_0 to be turned off before the firing of $a^*_{/i+1}$. Therefore, to make the falling transitions of w_1 and w_z acknowledged, the next turn-off sets of c_1 and c_z, i.e., $T_{\text{off}}(c_1)_{/i+1}$ and $T_{\text{off}}(c_z)_{/i+1}$, cannot exist in the same MG.

During the decomposition of α in Fig. 3(a), each possible decomposition of α must be checked to be hazard-free by the theorems presented above until a successful decomposition is found. The intermediate gate α_1 in Fig. 3(b) has to be decomposed again if it is still not available in the gate library. This decomposition process is repeatedly performed on each new intermediate gate until all gates are implementable. However, the hazard-free decomposition of an AND gate does not always exist. For the five three-input AND gates in Fig. 2, only $A_{\text{busy\_s}}$ and $A_{\text{mrdc\_r}}$ can be successfully decomposed.

V. RESYNTHESIS

For a gate α that cannot be hazard-freely decomposed, new signals are added to the STG and resynthesis is performed. Assume α has been decomposed into $(q+1)$-level gates as Fig. 5(a) shows, each intermediate gate α_i $(1 \leq i \leq q-1)$ and α_z are hazard-free, but there exists unacknowledged transitions of signal w_q. The intermediate cube c_q of gate α_q hence belongs to one of the following three unacknowledged cases:

1) c_q is not well behaved;
2) c_q is well behaved, but there exists some MG in which the number of turn-on sets of c_q is greater than that of c_α;
3) c_q is well behaved and the number of turn-on sets of c_q is less than or equal to that of c_α in any MG, but there exists some MG containing a turn-off set of c_q where the primary input z_q is not kept unchanged between the firing of $a^*_{/i}$ and $a^*_{/i+1}$.

There are unacknowledged rising and falling transitions of w_q in the first two cases, whereas in the third case, only the falling

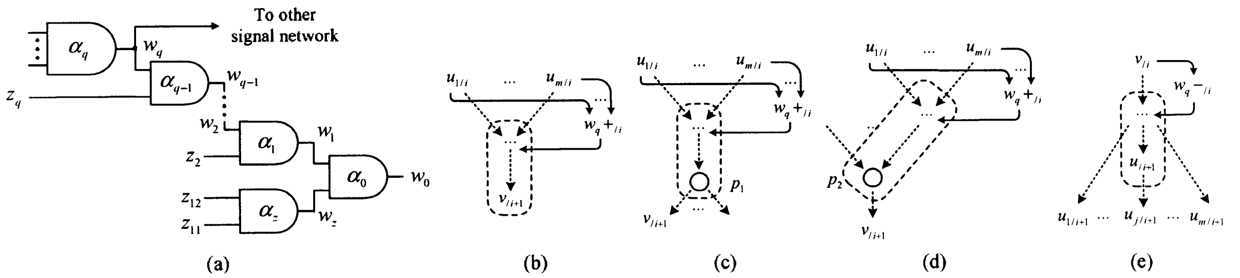

Fig. 5. (a) The IGA signal-adding method. The adding of a rising IGA transition to (b) an MG; (c) an STG with multifanout place; (d) an STG with multifanin place. (e) The adding of a falling IGA transition to an MG.

transition of w_q cannot be acknowledged. If c_q is not well behaved, there exists unexpectable transitions of w_q. Since it is impossible for an unexpectable transition to be acknowledged, the only way to solve this is to find another possible decomposition where c_q is well behaved. For a well-behaved c_q, two signal-adding methods are developed to generate a new STG for resynthesis.

A. IGA

As shown in Fig. 5(a), for an unacknowledged gate α_q whose underlying cube c_q is well behaved, each unacknowledged transition of w_q can be made acknowledged if w_q becomes explicit in the STG, called IGA. This idea of global acknowledgment was suggested in [1] and later formalized in [7], [11] constructed at the SG level. The IGA signal w_q is added to the initial STG G so that each transition of w_q can be acknowledged by a corresponding transition specified in the new STG G'. In the circuit level, the output of α_q is connected, with an inverter attached if necessary, to some signal network that acknowledges w_q. Since the adding of a new signal to G may change the circuit of some signal(s) to be implemented, G' has to be resynthesized. To add w_q to G, each transition of w_q, corresponding to a turn-on or turn-off change of the IGA cube c_q, must be carefully inserted so that G' still preserves the behavior of G. Properties such as CSC, consistency and persistency, which are originally satisfied in G, should also be satisfied in G'. To add a new transition $w^*_{q/i}$ to G, we have to find the transitions that are connected to $w^*_{q/i}$, i.e., the trigger transitions of $w^*_{q/i}$ and the transition to which $w^*_{q/i}$ is connected, i.e., the acknowledge transition of $w^*_{q/i}$. For each added rising transition $w_q+_{/i}$ and falling transition $w_q-_{/i}$, these trigger and acknowledge transitions can be obtained with the help of the corresponding turn-on set $T_{\text{on}}(c_q)_{/i}$ and turn-off set $T_{\text{off}}(c_q)_{/i}$ of c_q. Since the behavior of the outside environment cannot be changed, the transition that can be chosen for acknowledging $w^*_{q/i}$ must be a noninput signal transition.

1) Trigger Transitions of $w_q+_{/i}$: Since c_q will always be turned on by firing all the transitions in $T_{\text{on}}(c_q)_{/i}$, $w_q + _{/i}$ must be enabled after these transitions fire. The transitions in $T_{\text{on}}(c_q)_{/i}$ hence constitute the trigger transitions of $w_q + _{/i}$.

2) Trigger Transitions of $w_q - _{/i}$: Since c_q will always be turned off by firing any one of the transitions $v_{j/_i}$ in $T_{\text{off}}(c_q)_{/i}$, $w_q - _{/i}$ must be enabled after $v_{j/_i}$ fires. This OR-causality behavior, however, is not allowed in the STG for signal adding. Each turn-off set is hence restricted to consist of only one turn-off transition and this transition uniquely constitutes the trigger transition of $w_q - _{/i}$.

3) Acknowledge Transition of $w_q + _{/i}$: Let the turn-on set $T_{\text{on}}(c_q)_{/i}$ be composed of m Q-on transitions $u_{1/i}, u_{2/i}, \ldots, u_{m/i}$ and $T_{\text{off}}(c_q)_{/i+1}$ be its next turn-off set composed of only one turn-off transition $v_{/i+1}$. To avoid generating any autoconcurrent transitions of $w_q + _{/i}$, the acknowledge transition t_{ack} must be chosen such that $w_q + _{/i}$ fires before its next transition $w_q - _{/i+1}$. And to make each trigger transition $u_{j/i} \in T_{\text{on}}(c_q)_{/i}$ persistent to $w_q + _{/i}$, t_{ack} must also be chosen such that $w_q + _{/i}$ fires before the next transition of each $u_{j/i}$. Any transition that is interleaved with $(T_{\text{on}}(c_q)_{/i}, v_{/i+1})$, can be chosen to attain this purpose, as Fig. 5(b) shows. This restriction ensures that $w_q + _{/i}$ fires before $v_{/i+1}$ and since $w_q - _{/i+1}$ is enabled after the firing of $v_{/i+1}$, $w_q + _{/i}$ and t_{ack} hence fire before $w_q - _{/i+1}$. No autoconcurrent transition of $w_q+_{/i}$ is generated. For transition $v_{/i+1}$, if it is the next transition to some transition in $T_{\text{on}}(c_q)_{/i}$, any transition that is also the next transition to some other transition in $T_{\text{on}}(c_q)_{/i}$ fires after $v_{/i+1}$. If not, $v_{/i+1}$ will fire before the next transition of each transition in $T_{\text{on}}(c_q)_{/i}$. Since $w_q+_{/i}$ fires before $v_{/i+1}$, it also fires before the next transition of each transition in $T_{\text{on}}(c_q)_{/i}$. All the trigger transitions of $w_q+_{/i}$, i.e., all the transitions in $T_{\text{on}}(c_q)_{/i}$, are hence persistent to $w_q+_{/i}$. Since $w_q+_{/i}$ is a new trigger transition of t_{ack} and t_{ack} fires before $w_q-_{/i+1}$, $w_q-_{/i}$ is persistent to t_{ack}.

If there exist multiple MGs in an STG, some additional restrictions must be satisfied when choosing the acknowledge transition t_{ack}. For an MG κ that contains $T_{\text{on}}(c_q)_{/i}$, it also contains $w_q+_{/i}$; there must be an acknowledge transition in κ. As Fig. 5(c) shows, if there exists a multifanout place p_1 interleaved with ($T_{\text{on}}(c_q)_{/i}$, $v_{/i+1}$), t_{ack} must be interleaved with ($T_{\text{on}}(c_q)_{/i}$, p_1). Otherwise, if t_{ack} is interleaved with (p_1, $v_{/i+1}$), there exists some MG that contains $T_{\text{on}}(c_q)_{/i}$ but not t_{ack}. Similarly, for an MG κ that does not contain $T_{\text{on}}(c_q)_{/i}$, it does not contain $w_q+_{/i}$; there must not be any acknowledge transition in κ. As Fig. 5(d) shows, if there exists a multifanin place p_2 interleaved with ($T_{\text{on}}(c_q)_{/i}$, $v_{/i+1}$), t_{ack} must be interleaved with ($T_{\text{on}}(c_q)_{/i}$, p_2). Otherwise, if t_{ack} is interleaved with (p_2, $v_{/i+1}$), there exists some MG that contains t_{ack}, but not $T_{\text{on}}(c_q)_{/i}$.

4) Acknowledge Transition of $w_q-_{/i}$: Let the turn-off set $T_{\text{off}}(c_q)_{/i}$ be composed of only one turn-off transition $v_{/i}$ and $T_{\text{on}}(c_q)_{/i}$ be its next turn-on set composed of m Q-on transitions $u_{1/i+1}, u_{1/i+2}, \ldots, u_{m/i+1}$. To avoid generating any autoconcurrent transitions of $w_q-_{/i}$, the acknowledge transition t_{ack} must be chosen such that $w_q-_{/i}$ fires before its next transition $w_q+_{/i+1}$. And to make the trigger transition $v_{/i}$ persistent to

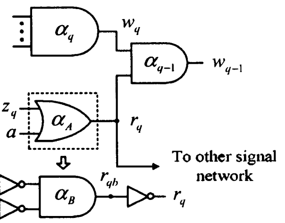

Fig. 6. SR signal-adding method.

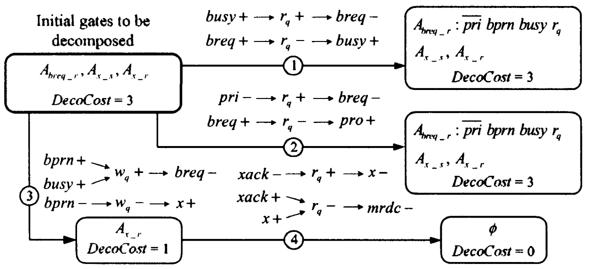
Initial gates to be decomposed
$A_{breq\_r}, A_{x\_s}, A_{x\_r}$
$DecoCost = 3$
$busy+ \rightarrow r_q+ \rightarrow breq-$
$breq+ \rightarrow r_q- \rightarrow busy+$
1
$A_{breq\_r}: \overline{pri}\ bprn\ busy\ r_q$
A_{x_s}, A_{x_r}
$DecoCost = 3$
$pri- \rightarrow r_q+ \rightarrow breq-$
$breq+ \rightarrow r_q- \rightarrow pro+$
2
$A_{breq\_r}: \overline{pri}\ bprn\ busy\ r_q$
A_{x_s}, A_{x_r}
$DecoCost = 3$
$bprn+, busy+ \rightarrow w_q+ \rightarrow breq-$
$bprn- \rightarrow w_q- \rightarrow x+$
3
$xack- \rightarrow r_q+ \rightarrow x-$
$xack+, x+ \rightarrow r_q- \rightarrow mrdc-$
A_{x_r}
$DecoCost = 1$
4
ϕ
$DecoCost = 0$

Fig. 7. Resynthesis procedure of *master-read.g*.

$w_q-_{/i}$, t_{ack} must also be chosen such that $w_q-_{/i}$ fires before the next transition of $v_{/i}$, let it be $u_{/i+1}$. Any transition that is interleaved with ($v_{/i}$, $u_{/i+1}$) can be chosen to attain this purpose, as Fig. 5(e) shows. This restriction ensures that $w_q-_{/i}$ fires before $u_{/i+1}$; the trigger transition $v_{/i}$ is hence persistent to $w_q-_{/i}$. Since only after $u_{/i+1}$ fires will it be possible for $w_q - _{/i+1}$ to be enabled, $u_{/i+1}$ must be a transition in $T_{\text{on}}(c_q)_{/i+1}$ or fire before at least one of the transitions in $T_{\text{on}}(c_q)_{/i+1}$. Therefore, $w_q-_{/i}$ fires before at least one of the transitions in $T_{\text{on}}(c_q)_{/i+1}$. Since w_q+_{i+1} is enabled after the firing of all the transitions in $T_{\text{on}}(c_q)_{/i+1}$, $w_q-_{/i}$ and t_{ack} hence fire before $w_q+_{/i+1}$. No autoconcurrent transition of $w_q+_{/i}$ is generated. Moreover, since $w_q+_{/i}$ is a new trigger transition of t_{ack} and t_{ack} fires before $w_q+_{/i+1}$, $w_q+_{/i}$ is persistent to t_{ack}. Similarly, if there exists a multifanin or multifanout place p interleaved with ($v_{/i}$, $u_{/i+1}$), t_{ack} must be interleaved with ($v_{/i}$, p).

B. SR

When the intermediate cube c_q belongs to the third unacknowledged case, only the falling transition of w_q cannot be acknowledged. If c_q is turned off before the firing of $a^*_{/i+1}$, another signal-adding method called SR can be applied. For the convenience of explanation, the gate to be decomposed is assumed to exist in the transition network of a rising transition $a+_{/i}$ and let $z_q+_{/i}$ be the corresponding trigger or context transition of $a+_{/i}$ on the primary input z_q. Gate in the transition network of a falling transition $a-_{/i}$ can also be dealt with the same way. The idea of SR is to replace z_q with a new SR signal r_q so that $z_q+_{/i}$ and its next transition $z_q-_{/i+1}$ are acknowledged by the rising and falling transitions of r_q, respectively, and r_q is kept unchanged between the firing of $a+_{/i}$ and its next transition $a-_{/i+1}$. The falling transition of w_q can hence be acknowledged by that of w_{q-1}. As Fig. 6 shows, r_q is realized in the circuit level as the output of a new OR gate α_A with input set $\{z_q, a\}$, satisfying that α_A is enabled by the firing of $z_q+_{/i}$ and disabled by the firing of $z_q-_{/i+1}$ and $a-_{/i+1}$. Except that, α_A will not be enabled or disabled by any other transitions. Adding a new gate to the implementation will change the circuit of some signal(s) to be implemented; the whole implementation has to be regenerated. That is, each transition of r_q must be added to the STG and resynthesis has to be performed.

By the duality principle, the OR gate α_A can be transformed into an AND gate α_B with all the inputs and the output inverted. The rising transition of r_q, corresponding to the falling transition of r_{qb}, can hence be perceived from the turn-off change of the SR cube $c_B = \overline{z_a a}$. Similarly, the falling transition of r_q, corresponding to the rising transition of r_{qb}, can be perceived from the turn-on change of c_B. To avoid generating any unexpectable change of c_B, c_B must be well behaved. Since α_A is enabled (or α_B is disabled) only by the firing of $z_q+_{/i}$, c_B must contain exactly one turn-off set $\{z_q+_{/i}\}$. Since α_A is disabled (or α_B is enabled) only by the firing of $z_q-_{/i}$ and $a-_{/i+1}$, c_B must also contain exactly one turn-on set $\{z_q-_{/i+1}, a-_{/i+1}\}$. The trigger and acknowledge transitions of each rising or falling transition of SR signal r_q can be obtained from those of the corresponding falling or rising transition of r_{qb}, which are found in the same way as that of an IGA transition. For an STG with multifanin or multifanout place, the SR signal can be similarly added as that in adding an IGA signal.

C. Resynthesis Procedure: An Example

After a new STG G' is created from the initial STG G by the IGA or SR signal-adding method, it has to be resynthesized and a new implementation A' is generated. The decomposition work is finished if there exists no undecomposable gate in A', or another new STG G'' has to be created for the next resynthesis. Since the adding of a new signal to G may result in more undecomposable gates in A', A' has to be checked if *progress* is made when compared with the initial implementation A. Here, progress being made means that the *decomposition cost* of A' is lower than that of A. The decomposition cost of an implementation is the total decomposition cost of all the undecomposable gates; the decomposition cost of an undecomposable gate α is defined as that of the smallest unacknowledged gate in one of the possible decompositions of α. For an unacknowledged gate α_q, the decomposition cost is defined as the number of two-input gates which is decomposed into. The more such gates, the more resynthesis cycles are required. Considering a four-input undecomposable gate α, if it can only be hazard-freely decomposed into a three-input gate cascaded with a two-input gate, the smallest unacknowledged gate of α is a two-input gate whose decomposition cost is one. If no such hazard-free decomposition exists for α, the smallest unacknowledged gate of α is a three-input gate with decomposition cost two. If progress is made after resynthesis, G'' is created from G'. Otherwise, it has to be created from G again by working on a different acknowledge transition, a different unacknowledged gate, a different signal-adding method, or a different undecomposable gate. In some cases, however, if a circuit cannot be hazard-freely decomposed after all the solution space is searched, the condition for deciding if progress is made has to be relaxed.

Now, we illustrate the resynthesis procedure by decomposing the three undecomposable gates $A_{\text{breq\_r}}$, A_{x_s} and A_{x_r} in Fig. 2. Fig. 7 gives the whole resynthesis procedure. Shown in the boxs are the undecomposable gates, along with the corresponding cubes, in each implementation and this cube is

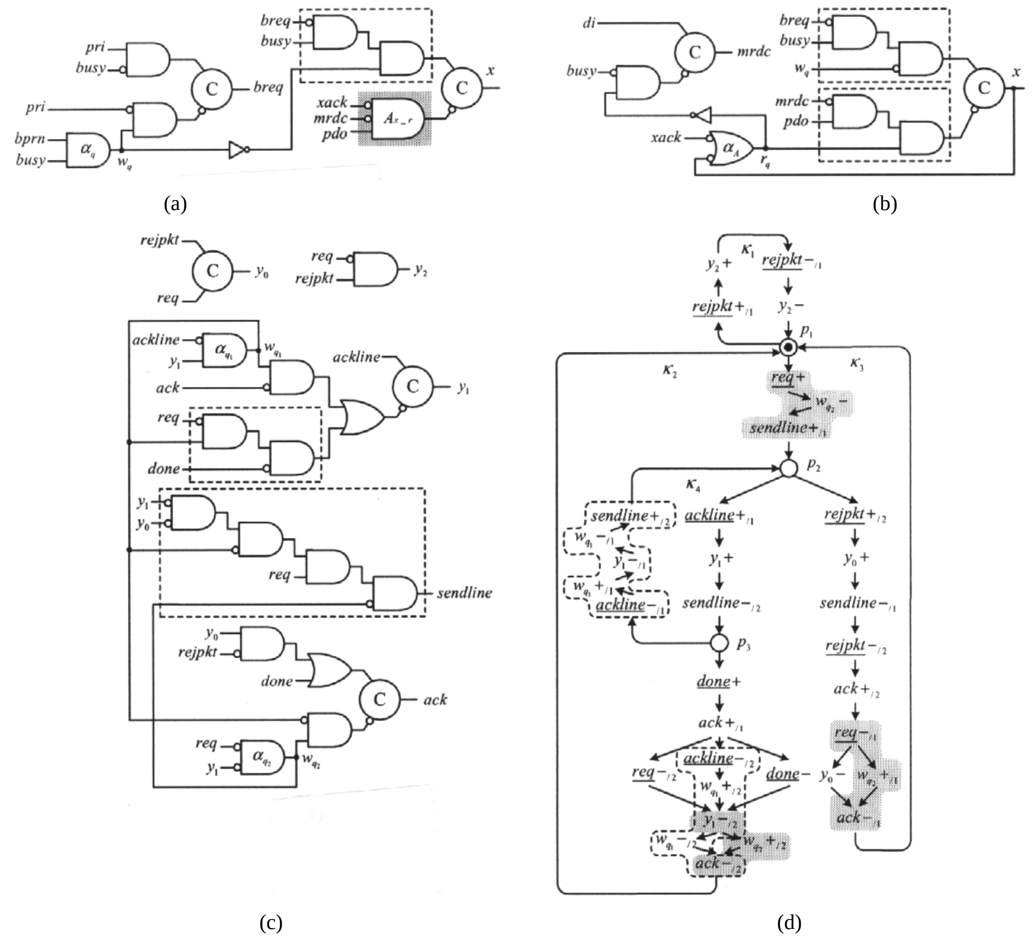

Fig. 8. Partial implementation of *master-read.g* after adding an (a) IGA signal and (b) SR signal. (c) Decomposed implementation of *sbuf-send-pkt2.g*. (d) Final STG.

omitted if it is the same as that in the previous implementation. It can be seen from Fig. 7 that starting from gate $A_{\mathrm{breq\_r}}$, four resynthesis cycles are required to make each gate decomposed. Progress is first made after the third resynthesis, where $A_{\mathrm{breq\_r}}$ has been hazard-freely decomposed by the IGA method. Fig. 8(a) shows how this is achieved in the circuit level. Due to the change of the circuit of signal x, A_{x_s} can also be decomposed hazard free. The decomposition work is finished after the fourth resynthesis. Fig. 8(b) shows how the new OR gate α_A is added to the circuit. For another example *sbuf-send-pkt2.g*, there is only one 3-input gate, corresponding to cube $\overline{\mathrm{ackline}}\ \overline{\mathrm{ack}}\ y_1$, in the transition network of $y_1-_{/1}$ that cannot be hazard-freely decomposed. By the IGA method, two signals w_{q_1} and w_{q_2} (both with multiple instances of transitions) have to be added to the STG for resynthesis. The decomposed implementation is given in Fig. 8(c) and 8(d) shows the final STG.

VI. Experimental Results

The method for the decomposition and resynthesis of SI circuits presented in the previous sections has been automated and incorporated into our earlier synthesis tool [10] in approximately 20 000 lines of *C* code. We have also demonstrated the time efficiency of our method by testing it on the asynchronous benchmarks [18], assuming a gate library where only two-input basic gates and *C* elements are available. The experimental results, shown in Table IV, were obtained on a Sparc Ultra-30 station with a clock speed of 248 MHz and 128 MB of physical memory. All the decomposition results have also been verified to be correct by the tool developed in [17]. In Table IV, column "Sigs" reports the number of signals to be added to the initial STG for resynthesis. The complexity of an AND/OR gate is measured as the number of literals, either complemented or not. Column "Lits" reports the number of literals of all the combinational gates and the number of required *C* elements is given in column "*C*-eles". In addition, the number in the parentheses "old" indicates the number of *C* elements in the initial implementation. The required CPU time (in seconds) is reported in column "Time".

It can be seen from Table IV that since only the combinational decomposition is considered in our method, there is no big change in the number of C elements before and after decomposition of every circuit except *vbe5c.g* (using one fewer C element). We also compared our results with [11] (by the experimental data from [7]) and Petrify [6] (by the command "petrify -lit2") assuming the same gate library. The number of added signals in our method is lower than that in [11] and Petrify, implying that fewer resynthesis cycles are required by our method. If the cost of a C element is considered to be three literals, the total cost of our implementation (Total 1) is 666

TABLE IV
EXPERIMENTAL RESULTS

Circuit	Ours				[11]				Petrify			
	Sigs	Lits	C-eles(old)	Time	Sigs	Lits	C-eles	Time	Sigs	Lits	C-eles	Time
alloc-outbound	0	14	3(3)	0.1	1	14	4	7.0	1	14	2	1.7
chu133	0	12	2(2)	0.1	2	12	1	2.0	1	12	1	1.6
chu150	0	14	1(1)	0.1	2	14	2	2.0	2	14	1	2.2
converta	1	16	3(3)	2.0	1	12	3	2.0	4	16	3	6.4
ebergen	1	14	2(2)	0.7	3	20	3	2.0	3	14	1	3.1
half	0	2	2(2)	0.1	1	2	2	1.0	0	2	2	0.5
hazard	2	8	2(2)	0.1	2	12	2	1.0	4	10	2	3.1
master-read	2	26	7(7)	2.8	8	30	9	274.0	n.a.	n.a.	n.a.	n.a.
mp-forward-pkt	0	12	3(3)	0.1	2	12	3	3.0	1	14	1	1.8
nak-pa	1	12	4(4)	0.5	1	20	4	4.0	3	18	1	5.0
nowick	0	20	2(2)	0.1	2	16	1	3.0	2	14	1	2.5
ram-read-sbuf	0	14	4(4)	0.1	2	20	4	8.0	4	18	3	9.7
rcv-setup	0	8	1(1)	0.1	2	10	1	2.0	1	8	1	0.7
rpdft	0	22	0(0)	0.1	5	22	1	10.0	7	22	0	5.0
sbuf-ram-write	1	20	2(2)	0.6	4	22	6	23.0	3	20	2	9.4
sbuf-send-ctl	1	16	3(3)	0.8	3	22	5	19.0	2	14	2	3.5
sbuf-send-pkt2	2	24	3(3)	1.6	6	32	5	130.0	4	16	3	6.9
seq_mix	2	14	3(3)	3.2	9	30	6	247.0	0	14	3	3.0
seq4	0	12	3(3)	0.2	4	18	7	12.0	1	12	3	2.3
trimos-send	6	40	6(6)	4.2	10	36	8	129.0	6	30	7	34.2
vbe5b	1	10	2(2)	0.1	1	10	2	1.0	0	8	2	1.6
vbe5c	1	6	2(3)	0.1	1	4	3	1.0	1	4	3	0.9
vbe6a	0	16	6(6)	0.1	8	16	7	31.0	4	16	6	7.8
vbe10b	2	32	7(7)	12.9	10	28	7	76.0	8	24	8	31.2
wrdatab	6	48	5(5)	5.7	7	48	7	93.0	15	46	9	394.5
Total 1	29	432	78(79)	36.5	97	482	103	1,083.0				
Total 2	27	406	71(72)	33.7					77	380	67	538.6

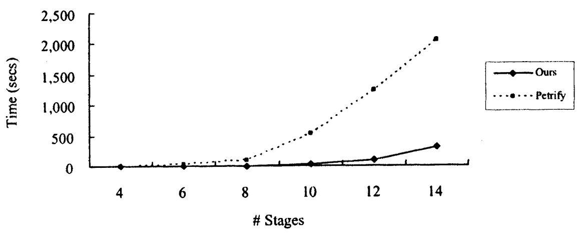

Fig. 9. Run time comparison of *vbe6a.g* with different stages.

literals. Compared with [11], where the total cost is calculated as 791 literals, our method can reach approximately a 18.8% area reduction. Moreover, our run time, which has been normalized due to different machines used, is only 16.7% of that in [11] obtained on a Sparc 20 machine. On the other hand, we found that the circuit *master-read.g* cannot be hazard-freely decomposed by Petrify. The total cost of our implementation excluding this circuit (Total 2) is 619 literals, whereas the total cost by Petrify is 581 literals. Though our method needs 6.1% more circuit area, this slightly worse but acceptable result can be further improved in the future by taking the sequential decomposition into consideration. The required run time by our method, however, is only 6.3% of that by Petrify. In addition, we also used *vbe6a.g* as another scalable example to further show the time efficiency of our method. It is a MG with four stages of request-acknowledge pair executing sequentially. Several of its variations with different number of stages were also tested. The run-time comparison with Petrify is depicted in Fig. 9.

VII. CONCLUSION

We have presented a time-efficient method for the decomposition and resynthesis of SI circuits. The method starts with the STG specification of an SI circuit, then various hazard-free decompositions are investigated for each high-fanin AND gate that does not exist in the gate library. For those gates that cannot be decomposed hazard free, new signals are added to the STG and resynthesis is performed. By adopting the STG as our input specification, reducing the resynthesis cycles, and applying the efficient signal-adding methods for resynthesis, the run time has been largely reduced with only a little more area expense. In the future, we plan on modifying our method to take the sequential decomposition and general storage elements into consideration and to investigate more latch sharing possibilities for area reduction.

ACKNOWLEDGMENT

The authors wish to thank all the anonymous reviewers for their valuable suggestions and careful review which helped to enhance the quality of the manuscript.

REFERENCES

[1] P. A. Beerel and T. H.-Y. Meng, “Automatic gate-level synthesis of speed-independent circuits,” in *Proc. Int. Conf. Computer-Aided Design*, 1992, pp. 581–586.
[2] T. H.-Y. Meng, R. W. Brodersen, and D. G. Messerschmitt, “Automatic synthesis of asynchronous circuits from high-level specifications,” *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 1185–1205, Nov. 1989.
[3] S. M. Burns, “General conditions for the decomposition of state holding elements,” in *Proc. Int. Symp. Advanced Research in Asynchronous Circuits and Systems*, 1996, pp. 48–57.

[4] T.-A. Chu, “Synthesis of Self-Timed VLSI Circuits From Graph-Theoretic Specifications,” Ph.D. dissertation, Mass. Inst. Technol., Cambridge, 1987.
[5] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, “A region-based theory for state assignment in speed-independent circuits,” *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 793–812, Aug. 1997.
[6] ——, “Petrify: A tool for manipulating concurrent specifications and synthesis of asynchronous controllers,” *IEICE Trans. Inform. Syst.*, vol. E80-D, no. 3, pp. 315–325, 1997.
[7] J. Cortadella, M. Kishinevsky, L. Lavagno, E. Pastor, and A. Yakovlev, “Decomposition and technology mapping of speed-independent circuits using Boolean relations,” in *Proc. Int.Conf. Computer-Aided Design*, 1997, pp. 220–229.
[8] A. Davis and K. Stevens, “The post office experience: Design a large asynchronous chip,” in *Proc. 26th Int. Conf. System Sciences*, 1993, pp. 409–418.
[9] M. Hack, “Analysis of Production Schemata by Petri Nets,” M.S. thesis, Mass. Inst. Technol., Cambridge, 1972.
[10] J.-M. Jou, R.-D. Chen, and K.-M. Lin, “An integrated synthesis system for speed-independent asynchronous circuits,” in *Proc. Int. Symp. Circuits and Systems*, 1997, pp. 1600–1603.
[11] A. Kondratyev, J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev, “Technology mapping for speed-independent circuits: Decomposition and resynthesis,” in *Proc. Int. Symp. Advanced Research in Asynchronous Circuits and Systems*, 1997, pp. 240–253.
[12] A. Kondratyev, M. Kishinevsky, and A. Yakovlev, “Hazard-free implementation of speed-independent circuits,” *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 749–771, Sept. 1998.
[13] L. Lavagno and A. Sangiovanni-Vincentelli, *Algorithms for Synthesis and Testing of Asynchronous Circuits*. Norwell, MA: Kluwer, 1993.
[14] T. Murata, “Petri nets: Properties, analysis and applications,” *Proc. IEEE*, vol. 77, pp. 541–580, Apr. 1989.
[15] S.-B. Park and T. Nanya, “Synthesis of asynchronous circuits from signal transition graph specifications,” *IEICE Trans. Inform. Syst.*, vol. E80-D, no. 3, pp. 326–335, 1997.
[16] E. Pastor, J. Cortadella, A. Kondratyev, and O. Roig, “Structural methods for the synthesis of speed-independent circuits,” *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 1108–1129, Nov. 1998.
[17] O. Roig, J. Cortadella, and E. Pastor, “Hierarchical gate-level verification of speed-independent circuits,” in *Asynchronous Design Methodologies*. Piscataway, NJ: IEEE Press, 1995, pp. 129–137.
[18] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, “SIS: A System for Sequential Circuit Synthesis,” Univ. California, Berkeley, UCB/ERL M92/41, 1992.
[19] P. Siegel and G. De Micheli, “Decomposition methods for library binding of speed-independent asynchronous designs,” in *Proc. Int. Conf. Computer-Aided Design*, 1994, pp. 558–565.
[20] P. Vanbekbergen, B. Lin, G. Goossens, and H. De Man, “A generalized state assignment theory for transformation on signal transitions graphs,” *J. VLSI Signal Process.*, vol. 7, no. 1–2, pp. 101–116, 1994.
[21] C. Ykman-Couvreur and B. Lin, “Optimal state assignment for asynchronous circuit synthesis,” in *Proc. 2nd Working Conf. Asynchronous Design Methodologies*, 1995, pp. 118–127.

Ren-Der Chen received the B.S. degree in electrical engineering in 1992 from the National Cheng Kung University, Tainan, Taiwan, R.O.C., where he is currently working toward the Ph.D. degree.

His research interests are in the area of design and synthesis of asynchronous circuits and VLSI chip design.

Jer-Min Jou received the Ph.D. degree in electrical engineering and computer science from National Cheng Kung University, Tainan, Taiwan, R.O.C., in 1987.

In 1989, he was an Associate Professor in the Department of Electrical Engineering, National Cheng Kung University, where he is currently a Professor. His research interests include SoC hardware-software codesign, System design, ASIC design/synthesis, VLSI CAD, and asynchronous circuit design.

Dr. Jou was the recipient of a Distinguished Paper Citation at the 1987 IEEE ICCAD Conference, Santa Clara, CA., the Longterm Best Paper Award from Acer Foundation in 1998 and 1999, and the First Level of 2001 IP Competition sponsored by Ministry of Education, R.O.C. He has been a reviewer of many journals including the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: FUNDAMENTAL THEORY AND APPLICATIONS.