# Direct Synthesis of Hazard-Free Asynchronous Circuits from STGs Based on Lock Relation and MG-Decomposition Approach

*Kuan-Jen Lin, Jih-Wen Kuo, Chen-Shang Lin*

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

**Abstract:** A new realization algorithm is proposed to synthesize asynchronous hazard-free circuits directly from STGs with underlying free-choice Petri nets. Based on signal lock relation and MG-decomposition approach, the synthesis method does not use state diagram and thereby maintains problem size polynomially proportional to the number of signal only. Moreover, the synthesized circuits are guaranteed to be hazard-free under unbounded gate-delay mode without any post-realization analysis and modification. A sufficient condition for hazard-free realization is also derived based on the signal lock relation in STGs. The high-levelness of this condition allows easier manipulation of specification for realizability. The proposed direct-synthesis algorithm has been shown successful on over thirty academic and industrial examples.

## 1 Introduction

The asynchronous circuits feature the advantages of clock-skew-freeness as well as low peak power over the synchronous ones. These advantages are expected to become more significant as the progress of semiconductor technology induces even denser and more complex circuits. However, asynchronous design has more inherent difficulties than synchronous design, mainly on specification size and hazard problems. Many techniques have been proposed to automate the design process. Early techniques are well surveyed in [1, 2]. A good introduction to some modern techniques is given in [3].

The asynchronous design studied in this paper starts from a graphical specification, Signal Transition Graphs (STGs) [4, 6]. The general STG synthesis procedure is comprised of the following two phases: (1) ensure that the STG is feasible for realization; and (2) synthesize for each non-input signal explicitly [7] or implicitly [8] from the corresponding state diagram of the STG using Boolean minimization. To ensure hazard-free realization, methods have also developed under different delay models. In [8], bounded wire-delay model is assumed. The circuit delays and pre-assumed environment delays are used to verify and modify for hazard-freeness by a post-realization procedure. The resultant circuit naturally is subjected to reverification and remodification for different environment and new technology to implement circuits. On the other hand, for unbounded gate-delay model as in [5, 9, 10], the hazard-freeness of circuit remains independent of the implementation technology. However, their sufficient conditions to ensure hazard-free realization are all based on state diagram. These conditions are difficult to apply to the original higher-level description on STG.

In addition to the above hazard-freeness problems, the state diagram synthesis of existent methods also presents a potential difficulty. Since the size of state diagram is exponential with respect to signal number, the time complexity of the synthesis process is also potentially exponential with respect to the number of signals. In [11], a realization algorithm was proposed to synthesize hazard-free asynchronous circuits directly from STGs with underlying marked-graphs and each signal having exactly one rising and falling transitions. The entire synthesis process is wholly on STG domain, i.e. without using state-diagram, and the realized circuits are ensured of hazard-freeness without post-realization modification. The work in this paper is a generalization of [11] to STGs with underlying free-choice Petri nets and multiple rising and falling signal transitions. The synthesis approach based on STG domain also can be found in [12]. However, their works did not concern physically hazard-free implementation and were restricted to STGs without free-choice behaviors.

In next section, background for STG approach and transitive lock relation will be briefly reviewed. Then the overview of our synthesis approach will be given. The kernel procedure in our method, *permissible cube extraction*, will be described in Section 4. The sufficient condition, *strong transitive lock relation*, is also proposed to ensure the hazard-free implementation. Then an algorithm based on MG-decomposition approach will be proposed to sovle STG with conditional (free-choice) behavior. Finally, we will give our evaluation results on over thirty examples and then the conclusion.

## 2 STG and Lock Relation

A signal transition graph, STG, can be viewed as an interpreted Petri net in which each event(transition) is a signal transition of asynchronous behavior[4]. In an STG, the places of single fanin and single fanout of the underlying net are removed and only two types of nodes are retained: the signal transitions and the multiple fanin or multiple fanout places. The transitions of a signal $a$, denoted by $a+$ and $a-$, are the rising and falling transitions respectively. And $t_a$ can be rising or falling transition of $a$. A signal in an STG may contain two or more transitions. If a signal contains only two transitions, then it is single-cycle; otherwise it is multi-cycle. For such a multi-cycle signal, /number(e.g. a+/1, a+/2) is used to distinguish its individual transitions of the same direction. In an STG, if a transition $t_1$ directly enables another transition $t_2$, we denote it by $t_1 \rightarrow t_2$. And $t_1 \Rightarrow t_2 ... \Rightarrow t_k$ implies $t_1$, $t_2$, ...$t_k$ are in a simple directed cycle with these transitions occurring in the order of their indexes.

The complexity of an STG is classified according to its underlying Petri net. The STG under our consideration has an underlying free-choice Petri net, denoted as STG/FC, which allows conditional behavior to be mod-

eled. The conditional signals are restricted to be input signals. A live and safe free-choice Petri-net can be covered by a set of live and safe MG-components and SM-components[4]. A Marked Graph(MG) is a net in which each place has at most one fanin and at most one fanout. In other words, such STG, denoted by STG/MG, has all its places removed. By duality, a State Machine(SM) is a net in which each transition has at most one fanin and at most one fanout. The MG can model concurrent behavior and SM model conditional behavior. The STG/MG plays a major role in our MG-decomposition approach for realization theory. In an STG, two transition $t_a$ and $t_b$ covered by the same MG-component are said to be ordered if $t_a \Rightarrow t_b$; otherwise, they are concurrent. And if they cannot be covered by any MG-component, they are in conflict.

In order to be successfully synthesized, an STG should possess certain properties which are summarized in the following definition [4]. Although this *liveness* property has been shown to be not necessary by [13] in general realization, the relaxed conditions are given on state diagram and ensure the hazard-free realizations only when delay is concentrated on the outputs of complex gates.

**Definition** (Liveness): An STG is live iff

(1) the underlying Petri net is live and safe,

(2) every SM-component contains exactly one token,

(3) for each signal $a$, transitions $t_a$s in the same MG-component are ordered, and $a+$ and $a-$ occur alternately.

The interpretation of these properties can be found in [4]. The second requirement is particularly useful in the analysis of STGs. Reports from [14, 15] showed that transition relations such as $t_i \Rightarrow t_j \Rightarrow t_k$ in such live STG can be evaluated by a table lookup mechanism in $O(1)$ time after a preprocessing phase which needs $O(T^6)$ complexity, where $T$ is the number of transitions. Throughout our discussions, the complexity of other procedures are established based on this $O(1)$ operation. In addition to liveness, there are other conditions to be satisfied, which will be described along with the realization algorithm.

In an STG/MG, special relations between signals will be shown to have significant impact on the realizability.

**Definition** (Lock): In a live STG/MG, there is a lock relation between single-cycle signals $a$ and $b$, denoted as $a\ L\ b$, iff

$(a+ \Rightarrow b+ \Rightarrow a- \Rightarrow b-) \vee (a+ \Rightarrow b- \Rightarrow a- \Rightarrow b+)$

**Definition** (Transitive Lock): In a live STG/MG, there is a transitive lock relation between single-cycle signals $a$ and $b$, denoted as $a\ L^t\ b$, iff

$a\ L\ b \vee (a\ L^t\ c \wedge c\ L^t\ b)$, where $c$ is another single-cycle signal in STG.

In [6, 16, 17], it was shown that the transitive lock is a sufficient condition for complete state coding. In the present work, the transitive lock relation will be further used to derive circuit implementation.

# 3 Overview of Our Synthesis Approach

Our hazard-free realization of asynchronous circuits is based on collecting appropriate sets of signals for a realization circuit model. We will firstly describe the circuit model and then the main steps of our synthesis procedure.

## 3.1 Realization Circuit Model

Our realization circuit model for each non-input signal is shown in Fig. 1, which consists of a memory element and

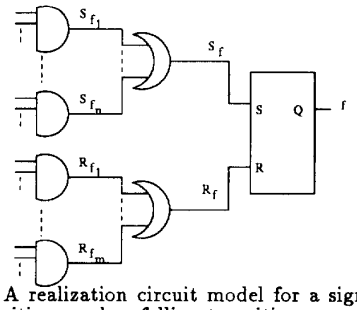


Figure 1: A realization circuit model for a signal with $n$ rising transitions and $m$ falling transitions.

two AND-OR subcircuits to set and reset the memory element. For simplicity, the memory element is assumed to be an SR-latch, although an C-element is preferred if delay hazard is to be completely avoided. Note also that when special situation arises, the realized circuit can be reduced into simpler circuit, such as combinational one or AND-only subcircuits.

In the circuit model, we let each AND gate responsible for the activation of one and only one transition of the synthesized signal, $f$. To ensure hazard-free implementation, each AND gate for its associated $t_f$, $AND_{t_f}$, will satisfy three requirements:

*(1) $AND_{t_f}$ covers all states enabling the associated $t_f$;*

*(2) $AND_{t_f}$ must be set off before the next transition of $f$ is enabled;*

*(3) Once $AND_{t_f}$ is set off, $AND_{t_f}$ remains off until the associated $t_f$ is enabled again.*

An OR gate then collects all the outputs of ANDs for rising transitions to set the latch and the other for falling transitions to reset the latch.

The function of a signal $f$ implemented with this model can be expressed as $f = S_f + f\overline{R_f}$, where $S_f = Cube_{f+/1} + ... + Cube_{f+/n}$ and $R_f = Cube_{f-/1} + ... + Cube_{f-/m}$ and each $Cube_{t_f}$ is the associated cube (AND) for transition $t_f$. From this expression, the functional correctness of the circuit model can be easily justified. To avoid delay hazards, SR-latch is replaced with a C-element attached with a inverter to the reset terminal. The formal proof for functional correctness and hazard-freeness for the circuits of signals implemented with our realization model can be found in [18].

## 3.2 Synthesis method

The main task of our synthesis is to extract appropriate cubes to satisfy the three requirements for our realization model. *Permissible cube*, defined later, can be used to satisfy those requirements. By signal transitive lock relation well defined on STG/MG, we can extract permissible cubes directly from STG domain. For STGs with free-choice behaviors, based on that a live STG/FC can be covered by a set of live STG/MG, an MG-decomposition approach will be used to decompose the original problem of STG/FC into some subproblems on STG/MGs. Then subcircuits derived from STG/MGs are integrated to fit our realization model. The main steps for synthesizing a signal $f$ can be summaried below.

*1. For each transition $t_f$,*

*1.1. Decompose cube extraction into some subproblems on STG/MGs;*

*1.2. Derive permissible cubes from STG/MGs by transitive lock relation;*

*1.3. Integrate subcircuits from STG/MGs to meet cube requirements.*

*2. Fit each cube into the realization circuit model.*

# 4 Permissible Cube Extraction on STG/MG

The main task of our synthesis is to extract appropriate cubes to satisfy the three requirements for the realization circuit model. The extraction will be performed on STG domain rather than state diagram. We will firstly derive the characteristics of *permissible cube*, which promise us to have requirements transformed from stat-diagram to STG domain for constructing our realization circuit model. Then the theoretical results for the identifying of permissible cubes based on transitive lock are described.

## 4.1 Permissible Cubes

We will first give the definition of permissible cubes and then discuss the means of identifying these cubes directly from STG/MGs.

**Definition** (Permissible Cube): A cube is permissible if its covered states in the corresponding state diagram of STG/MG all are connected.

The main property of the permissible cube is shown below whose proof can be found in [18].

**Lemma 1:** A cube $x_1x_2...x_n$ is permissible for a live STG/MG iff in the corresponding state diagram, any path (firing sequence) starting from a state in the cube and then causing the cube value to change twice must fire each transition of signals $x_1, x_2, ..., x_n$ equal times.

Take the example in Fig. 2 to show the above, where Fig 2(b) is the corresponding state diagram of the STG in Fig. 2(a) encoded with total signals [4] and Fig 2(c) is the corresponding lock graph in which arc represents lock relation. $\overline{a}\ \overline{c}d$ is a permissible cube because its covered states, 0001 and 0101, are connected. Then it is seen that any paths from the state 0001 (0101) to 0001 or 0101 will contain each of $a-/1$, $a+/1$, $a-/2$, $a+/2$, $c+$, $c-$, $d+$, $d-$ equal times. Furthermore, the firing of these transitions will enable $b-$ again. Another cube, $ad$, is not permissible because its covered states are separated into two disconnected sets, {1101} and {1001, 1011}.

Note that the permissible cube is not defined for specific transitions. But for a cube which has satisfied requirement (1) and (2) for a transition, if it is permissible, then it can meet our realization model. It is nontrivial to derive cubes for requirement (1) and (2). We are to derive theoretical results for extracting permissible cubes directly from STG domain. In our realization algorithm, permissible cubes are derived from simpler cubes based on transitive lock relation as stated in the following lemma.

**Lemma 2 [11]:**

(1) For a live STG/MG, the nonempty intersection of two permissible cubes which have at least one common literal is also a permissible cube.

(2) Let $x_1$ and $x_2$ be two single-cycle signals in a live STG/MG. If $x_1\ L\ x_2$, then the four cubes: $(x_1,\ \overline{x_1}) \times (x_2,\ \overline{x_2})$ all are permissible.

(3) Let $x_1$, $x_2$ and $y$ all be single-cycle signals in a live STG/MG. If $x_1\ L\ y$ and $y\ L\ x_2$ and let the two transitions of $x_1$ and $x_2$ between $y+$ and $y-$ be $x_1+$ and $x_2+$ ($\overline{x_1}+$, $\overline{x_2}-$), then the cubes $x_1x_2$ and $\overline{x_1}\ \overline{x_2}$ ($\overline{x_1}x_2$ and $x_1\overline{x_2}$) are permissible.

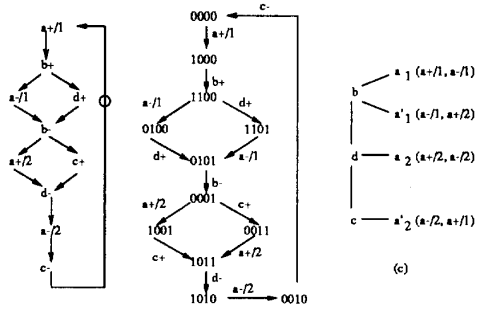


Figure 2: (a) An STG/MG example and (b) its state diagram and (c) its lock graph.

Let us take the STG example in Fig. 2 to show these applications. Because of $b\ L\ d$, by Lemma 2(2), $bd$, $b\overline{d}$, $\overline{b}d$ and $\overline{b}\ \overline{d}$ are all permissible. And because of $b\ L\ d\ L\ c$, by Lemma 2(3), $b\overline{c}$ and $\overline{b}c$ are both permissible. Then by Lemma 2(1), $b\overline{c}d$, $b\overline{c}\ \overline{d}$, $\overline{b}cd$ and $\overline{b}c\overline{d}$ all are permissible.

For multi-cycle signals, single-cycle transformation[17] can be used to derive permissible cubes with multi-cycle signals from cubes of its transformed single-cycle signals. For the example in Fig. 2, signal $a$ can be transformed in odd-order to be $(a_1+,\ a_1-) = (a+/1,\ a-/1)$ and $(a_2+,\ a_2-) = (a+/2,\ a-/2)$, and in even-order to be $(a_1'-,\ a_1'+) = (a-/1,\ a+/2)$ and $(a_2'-,\ a_2'+) = (a-/2,\ a+/1)$. Note that the above orders are reversed when the opposite polarity of signal is taken, e.g., even-order for signal $a$ is odd-order for $\overline{a}$. A transformed single-cycle signal of even(odd) order will remain high(low) when transitions occur for other transformed sibling signals. Consequently, the ANDing(Oring) of all even-order(odd-order) transformed signals is behaviorally equivalent to the original multi-cycle signal. For instance in Fig. 2, $a = a_1'a_2'$ and $a = a_1 + a_2$. Then permissible cubes with multi-cycle signals can be derived from cubes of its even-order transformed signals. For example in Fig. 2, since $a_1'\overline{d}$ and $a_2'\overline{d}c$ are permissible, $a_1'a_2'\overline{d}c$ is permissible, and therefore $a\overline{d}c$ is permissible. Note also that $a_1'\overline{d}$ and $a_2'\overline{d}c$ both can cover $a\overline{d}c$.

## 4.2 Procedure for Permissible Cube Extraction

We now present the procedure to extract permissible cubes for certain transitions. To ensure the success of the permissible cube extraction, the given live STG/MG should satisfy certain conditions. To define this, we first introduce *semi-lock*, $L^s$. A single-cycle signal $x$ transformed from a multi-cycle signal $mx$, is said to has $L^s$ relation with a transition $f+$, if $x+ \Rightarrow f+ \Rightarrow x-$, and in addition, all transitions of $mx$ are ordered with $f+$ and neither transition of $mx$ occurs between $x-$ and $x+$. For example in Fig. 2, $\overline{a_1}$ has $L^s$ relation to $b-$, but not to $c+$ because $a+/2$ is concurrent with $c+$. Another example, $a_2$ does not have $L^s$ relation to $d-$ because $a+/1$ and $a-/1$ occur between $a_2-$ and $a_2+$. In the following definition, single-cycle may be original or transformed from some multi-cycle signal.

**Definition (Strong $L^t$)**: In a live STG/MG, a single-cycle signal $x_1$ has strong $L^t$ relation to another single-cycle signal $f$ if (1) $x_1\ L^s\ f+$ and (2) $x_1\ L^t\ f$ with $x_1\ L\ x_2$, $x_2\ L\ x_3$,..., $x_n\ L\ f$ such that $\forall i,\ i = 2, ..., n-1$, $x_i\ L^s\ f+$ or a $t_{x_i}$ is concurrent with $f+$ while $x_{i-1}\ L^s\ f+$ and $x_{i+1}\ L^s\ f+$.

With strong $L^t$, we can now describe our procedure shown in Fig. 3. In step 2, the *permissible path* is the path which satisfies the second condition of strong $L^t$ definition. In step 4, the literals of transformed signals are directly replaced with multi-cycle signal literals. This replacement may cause multiple pulse during the positive period defined by the cube before replacement. This is checked and removed in step 5. If required, the intersection with $\overline{f}(f)$ is used to reduce the covering to just cover all the enabling states of $t_f$ such that multiple pulse is impossible. The selection of signal phase, ($a$ or $\overline{a}$), can be easily determined according to which phase holds during the enabling of $t_f$. For example, if $a+ \Rightarrow t_f \Rightarrow a-$, $a$ should be used in the realization.

The correctness of this procedure is established in the following theorem, whose proof can be found in [18]. In the theorem, the *transformed* single-cycle signal of an enabling transition is composed of this enabling transition and its preceding inverse transition and the *transformed* single-cycle signal of the enabled transition is composed of the enabled transition and its succeeding inverse transition.

**Theorem 1:** Given a transition $t_f$ in a live STG/MG, if all the *transformed* single-cycle signals of its enabling transitions have strong $L^t$ relation to the *transformed* single-cycle signal for $t_f$, then $Cube_{t_f}$ derived from Procedure *Extract_Cube($t_f$, STG/MG)* satisfies the requirements for our realization model.

**Extract_Cube($t_f$, STG/MG):**
*(Given a live STG/MG and a transition $t_f$ whose enabling transitions satisfy the condition of Theorem 1. LG is the corresponding lock-graph of STG. Let $f_j$ be the single-cycle signal composed of $t_f$ and its succeeding inverse transition and all enabling transitions of $t_f$ be $s_1+, ..., s_n+$. )*

*1. Let $Cube_{t_f}$ be a universal cube.*
*2. Foreach $s_k$, k=1, 2,.., n. {*
  *Select a shortest permissible path from $s_k$ to $f_j$ on LG.*
  *Let this path be $q_0,\ q_1, ...,\ q_{l+1} (q_0 = s_k,\ q_{l+1} = f_j)$.*
  *$F_{s_k} = q_0 q_1 ... q_l$.*
  *Forloop i = 0 to l, {*
    *Remove literals $q_i$ from $F_{s_k}$ unless $q_i\ L^s\ t_f$;*
    *i = i+1.*
  *}endforloop*
  *$Cube_{t_f} = Cube_{t_f} \cap F_{s_k}$.*
*}endforeach*
*3. $Cube'_{t_f} = Cube_{t_f}$.*
*4. Update $Cube_{t_f}$ with multi-cycle signals.*
*5. For each multi-cycle signal m in $Cube_{t_f}$,*
  *if m+ occurs during the positive period of $Cube'_{t_f}$,*
    *if $t_f = f+$, $Cube_{t_f} = Cube_{t_f} \cap \overline{f}$;*
    *otherwise $Cube_{t_f} = Cube_{t_f} \cap f$*
*6. Return($Cube_{t_f}$).*

Figure 3: Procedure for extracting permissible cubes on STG/MG.

The time complexity of above procedure is mainly on the selection of the shortest path in step 2. Since there is at most $T$ nodes in the lock graph, this step takes $O(T^2)$ complexity assuming that the number of enabling transitions is constant, where $T$ is the number of transitions in STG. This time complexity does not include the the derivation of Lock-graphs preprocessed for all signals, which is $O(T^3)$ in the worst case [15, 14].

# 5 Realization Algorithm for STG/FC Based on MG-decomposition

With the kernel procedure stated in last section, we now propose the MG-decomposition approach to synthesize circuits for STGs with free-choice behaviors. The cube extraction is decomposed into some subproblems on a set of MG-components of STG/FC. Each subproblem will be treated by the techniques in last section. Then those subcircuits will be easily integrated to meet realization model. In our subsequent discussions, we mainly consider the class of STG/FCs in which only one MG-component is running at one time and thus the running MG-component determines the states of STG. Moreover, there are no concurrent free-choice places in STGs. All examples from Berkely belong to such class. Under the considered STG class, all states of an STG can be determined by a set of MG-components which covers the original STG/FC, Covering MG-component Set (CMGS). An CMGS thus will suffice in our realization and we need not to consider all possible MG-components.

**Realization Algorithm for STG/FC:**
*(Given a live STG/FC in which each non-input transition and all of its enabling transitions satisfy the conditions of Theorem 1 in considered STG/MGs, let f be the synthesized signal, and $f+/1, ..., f+/n$ and $f-/1, ..., f-/m$ be all the rising transitions and falling transitions of f, respectively.)*

*1. Let both $S_f$ and $R_f$ be empty covers.*
*2. Forloop j = 1 to n,*
*2.1. Let $Cube_{f+/j}$ be a universal cube.*
*2.2. Foreach $MG_i$, $MG_i \in CMGS$ and $f+/j \in MG_i$*
  *$Cube_{f+/j} = Cube_{f+/j} \cap Extract\_Cube(f+/j,\ MG_i)$.*
  *If $Cube_{f+/j}$ is not permissible,*
    *$Cube_{f+/j} = Cube_{f+/j} \cap \overline{f}$.*
*2.3. Foreach $MG_j$, $MG_j \in CMGS$ and $f+/j \notin MG_j$*
  *Foreach $MG^*$, $MG^*$ be the DMGs of $MG_j$.*
    *$Cube_{f+/j} = Cube_{f+/j} \cap Extract\_Cube(f+/j,\ MG^*)$.*
  *If $Cube_{f+/j}-$ occurs outside the branch,*
    *$Cube_{f+/j} = Cube_{f+/j} \cap \overline{f}$.*
*2.4. $S_f = S_f \cup Cube_{f+/j}$.*
*}endforloop*
*3. Repeat step 2 with m, f, $f-/j$ and $R_f$ replacing n, $\overline{f}$, $f+/j$ and $S_f$ respectively.*
*4. Return $f = S_f + f(\overline{R_f})$.*

Figure 4: Realization Algorithm for STG/FC.

The realization algorithm is shown in Fig. 4. The main steps are 2.2 and 2.3. In step 2.2, a set of MG-components each of which contains the considered transition $f+/j$ and belong to the given CMGS will be first considered to derive subcircuits. Permissible cubes are derived separately from these considered MGs (by *Extract_Cube($t_f$, STG/MG)* and then those cubes are intersected together. Since all subcubes can satisfy the requirements (1) and (2) for our realization model, their intersection cube also can meet both. The question is whether the intersection cube can satisfy requirement (3). Since the intersection reduces state spaces, the states originally not covered by subcubes

now are still not covered by the intersection cube. However, the states originally covered by some subcube now may be split into several disconnected state groups and each is covered by the intersection cube. Hence, multiple pulses from $Cube_{f+/j}$ may occur in a cycle of $f+/j$ such that violates requirement (3). We can use $\overline{f}$ to intersect the cube to reduce the covering to just cover all the enabling states of $f+/j$ such that multiple pulses cannot occur. Of course, $Cube_{f+/j}$ now is permissible in all considered MGs.

Now, we can state that the derived cube after step 2.2 can satisfy the requirement for realization model if STG runs only MG-components containing $f+/j$. However in a cycle of $f+/j$, in addition to running some MG-component containing $f+/j$ one time, it may run some MG-components which do not contain $f+/j$. Therefore, we render the derived cube always off in MG-components not containing $f+/j$. This is the main concern in step 2.3. We will recombine these MG-components, not containing $f+/j$, with some parts of MG-components considered in step 2.2 to make some new derived MGs. Then we can again use the technique in last section to force the cube on only inside the MG parts considered in step 2.2. In other words, the cube remains off in MG parts which do not contain $f+/j$. The recombination takes the techniques, *branch reversing and self-loop unfolding*, which were originally introduced to decompose an STG/FC into STG/MGs for state-coding problem[17]. The details for these two operations will be not discussed here, but the following illustrative example will show their applications. Furthermore, it is shown that each MG-component considered in step 2.3 needs at most two new Derived MGs (DMGs) to cover itself[18]. We extract permissible cubes from these DMGs as step 2.2 and render cube always off outside the period from $f+/j$ to next $t_f$. This can make the derived cube off in most parts of the MGs not containing $f+/j$. However, it is still possible for states in these MGs and inside the period from $f+/j$ to next $t_f$ to set on the cube. To prevent this, we can again use $\overline{f}$ to intersect the cube to reduce the covering to just cover all the enabling states of $f+/j$ such that the cube always goes off before entering the states in MG-components not containing $f+/j$.

Take the example in Fig. 5 to illustrate this procedure. We are to synthesize signal $d$. The CMGS of this STG is $\{MG_1 = (a+,\ b+,\ a-,\ b-),\ MG_2 = (c+,\ d+,\ e-,\ d-/1,\ c-, e+), MG_3 = (c+,\ d+,\ c-,\ d-/2,\ e-, e+)\}$. We first consider the cube for $d+$. Since $MG_2$ and $MG_3$ cover it, we derive two permissible cubes, $ce$ and $c$, from them. The intersection of these two cubes, $ce$, does not cause multiple pulses problem, so the derived cube after step 2.2 is $ce$. In step 2.3, $MG_1$ is considered since it does not cover $d+$. Only one new derived MG shown in Fig. 5 (b), which is made from $MG_2$ plus the unfolded $MG_1$, is used to consider the $ce$ behavior on $MG_1$. A cube is derived from Fig. 5(b) and intersected with original $ce$ cube. We can see that the final cube is still $ce$ and it is set off before entering the $MG_1$. Consequently, $Cube_{d+} = ce$. Next to see $d-/1$, there is only $MG_2$ covering it. We can find $Cube_{d-/1} = \overline{e}$ after step 2.2. In step 2.3, there are tow MGs, $MG_1$ and $MG_3$, to be considered. For $MG_1$, the STG/MG in Fig. 5 (b) is used to consider the $\overline{e}$ behavior on $MG_1$. No additional literal is added to $Cube_{d-/1}$ and $Cube_{d-/1}$ is off in $MG_1$. For $MG_3$, we consider the new derived MG shown in Fig 5(c), which is made from branch $(e- \rightarrow d-/1 \rightarrow c-)$ plus the branch $(c+ \leftarrow d+ \leftarrow e+)$ reversed from $(c- \rightarrow d-/2 \rightarrow e-)$. The cube $\overline{e}$ is again derived for $d-/1$. The cube now is off outside the period $(d-/1 \rightarrow c- \rightarrow e+ \rightarrow d+)$, but the cube is not set off before leaving branch $(e- \rightarrow d-/1 \rightarrow c-)$. It can be set on in states $abcde = 00000$ such that $Cube_{d-/1}$ can be set on again without enabling $d-/1$again from off-set states, unexpected in hazard-free implementation. To prevent this, we need to intersect $\overline{e}$ with $d$. Then the derived cube $\overline{e}d$ always remains off in $(c- \rightarrow d-/2 \rightarrow e-)$. Since the remaining part of $MG_3$ is covered by $MG_2$, no further consideration is needed. After the cube for $d-/2$ is derived, $\overline{e}d$, we can fit $Cube_{d+}$, $Cube_{d-/1}$ and $Cube_{d-/2}$ into the realization model. The $d$ in reset subcircuits here will not cause hazards since the feedback line inside memory-element is assumned to be faster than any outside feedback line.

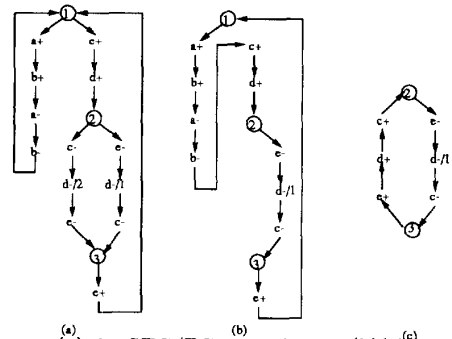

Figure 5: (a) An STG/FC example and (b)(c) its derived STG/MGs.

The complexity of this procedure depends on the number of MG-components considered in step 2.2 and 2.3. It is polynomially proportional to the size of the given CMGS. As for the complexity of synthesizing one signal, it is $O(T^2 \times MG)$ where $T$ is the number of transitions and $MG$ is the number of marked graphs in CMGS. Since the covering marked graphs are smaller in number than transitions, $MG$ is bounded by $O(T)$. Consequently, the complexity of of synthesizing one signal with our procedure is bounded by $O(T^6)$. Furthermore, since the transition number is linearly proportional to signal number, the complexity is also bounded by $O(Sg^3)$, where $Sg$ is the signal number in STG.

The circuit realization from the proposed algorithm may not always provide the simplest circuits. Some simple simplification techniques allow the original realization to be optimized, e.g., memory elements, gates, or literals can be removed as reported in [10, 18]. Moreover, the transitions of the same signal which are in the apparent equivalent structures between conflict branches are allowed to share the same enabling cube.

# 6 Experimental Results

The proposed realization procedure in this paper has been successfully evaluated with the set of STG benchmarks from Berkely. Table 1 shows the evaluated results of over thirty examples, the circuit implementations of which are measured in terms of C-element number and literal count in the combinational parts. The *Hazard* column is obtained by running general logic synthesis which optimizes the logic implementations from state diagrams without considering hazards. The result obtained from our procedures with simplification techniques is shown in the next column. The last column shows the result reported in [10] which synthesizes hazard-free circuits with unbounded gate-delay model from state diagrams.

The comparison between our result and "Hazard" indicates an overhead of about +22% in combinational parts and +10% for memories for guaranteeing hazard-freeness. By comparison with hazard-free implementation reported in [10], our result needs the same number of memory elements as [10] and 4% more literals. The difference mainly results from two cases: nowick.g and pe-send-ifc.g, in which the same cubes can be used to cover the enabling states of two distinct transitions of some signal without violating function. These common cubes are not shared in our present realization. Aside from these two cases, there is essential no difference in the results. This demonstrates the effectiveness of our realization algorithm. It is also worth noting that the considered $MG$ in all cases is equal to the covering MG-component number of the CMGS, that is, only one DMG is required for each MG-component in the step 2.3 of our procedure for STG/FCs.

# 7 Conclusion

In this paper, we have proposed a new realization algorithm to synthesize asynchronous hazard-free circuits under unbounded gate-delay model. Our approach does not use state diagram and thereby maintains problem size polynomially proportional to the number of signal only. A sufficient condition for hazard-free realization is derived based on the signal lock relation in STGs. The high-levelness of this condition allows easier manipulation of specification for realizability. The proposed direct-synthesis algorithm has been shown successful on over thirty academic and industrial examples. Our result has shown to need no more overhead than those with state diagram approaches. This demonstrates the effectiveness of our realization algorithm. Furthermore, all but one of the evaluated cases satisfy the proposed sufficient condition for hazard-freeness without modification. This shows the applicability of the sufficient condition as well as the algorithm.

# References


[1] S. H. Unger, *Asynchronous Sequential Switching Circuits*, Wiley Interscience, 1969.

[2] R. E. Miller, "Switching Theory," volume 2, Jhon Wiley and Sons, 1965.

[3] J. A. Brzozowski and C-J. H Seger, "Advances in Asynchronous Circuit Theory(Part II)," *EATCS Bulletin*, No. 43, pp. 199-263, 1991.

[4] T. A. Chu, "Synthesis of Self-Timed Control Circuits from Graphical Specifications", PhD thesis, MIT, June, 1987.

[5] V. I. Varshavky, *Self-Timed Control of Concurrent Process*, Kluwer Academic Publisher 1990.

[6] A. Y. Kondratyev, L. Y. Rosenblum, A. V. Yakovlev, "Signal Graphs: A Model for Designing Concurrent Logic," *Proceeding of the International Conference on Parallel Processing*, pp.51-54, 1988.

[7] T. H. Meng, *Synchronization Design for Digital Systems*, Kluwer Academic Publisher 1990.

[8] L. Lavagno, K. Keutzer, A. Sangiovanni Vincentelli, "Algorithms for Synthesis of Hazard-free Asynchronous Circuits," *Proceeding of 28th Design Automatic Conference*, pp. 302-308, 1991.

[9] Cho W. Moon, Paul R. Stephan and Robert K. Brayton, "Synthesis of Hazard-free Asynchronous Circuits from Graphical Specifications," *Proceeding of the International Conference on Computer-Aided Design*, pp. 322-325, 1991.

[10] Peter A. Beerel and Teresa H.-Y Meng, "Automatic Gate-Level Synthesis of Speed-Independent Circuits," *ICCAD* pp. 581-586, 1992.

[11] K. J. Lin and C. S. Lin, "A Realization Algorithm of of Asynchronous Circuits from STGs," *Proceeding of 1992 European Conference on Design Automation*, pp. 322-326.

[12] Chantal Ykman-Couvreur, Bill Lin, Gret Goossens and Hugo de Man, "Synthesis and Optimization of Asynchronous Controllers Based on Extended Lock Graph Theory," *Proceeding of 1993 European Conference on Design Automation*, pp. 512-517.

[13] A. Yakovlev, "A unified Signal Transition Graph Model for Asynchronous Control Circuits Synthesis," *ICCAD*, pp. 104-111, 1992.

[14] K. J. Lin and C. S. Lin, "Automatic Synthesis of Asynchronous Circuits," Ms Thesis, EE Dept. of National Taiwan University.

[15] Peter Vanbekbergen, "The Analysis of Temporal Relation in STGs," Report ESPRIT 2260 project.

[16] Peter Vanbekbergen, Francky Catthoor, Jef Van Meerbergen, Hugo de Man, "Optimized Synthesis of Asynchronous Control Circuits from Graph-theoretic Specifications," *ICCAD*, pp. 184-187, 1990.

[17] K. J. Lin and C. S. Lin, "On the Verification of State Coding in STGs," *ICCAD*, pp. 118-122, 1992.

[18] K. J. Lin and C. S. Lin, "Direct Synthesis of Hazard-Free Asynchronous Circuits from STGs under Unbounded Gate-Delay Model," submitted for publication, 1993.


| Circuits (a) | St | Sg | Hazard Me/Li | Ours Me/Li | [10](b) Me/Li |
|---|---|---|---|---|---|
| chu133.g | 24 | 6 | 13/2 | 13/2 | 13/2 |
| full.g | 15 | 4 | 4/2 | 4/2 | 4/2 |
| hybridf.g | 80 | 8 | 8/3 | 8/3 | 8/3 |
| vbe10b.g | 256 | 11 | 32/7 | 32/7 | 32/7 |
| vbe5b.g | 24 | 6 | 10/2 | 12/2 | 11/2 |
| vbe5c.g | 24 | 6 | 9/2 | 9/3 | 10/3 |
| chu150.g | 26 | 6 | 10/1 | 11/1 | 11/1 |
| chu172.g | 12 | 6 | 6/1 | 7/1 | 7/1 |
| converta.g | 18 | 5 | 18/3 | 20/3 | 20/3 |
| ebergen.g | 18 | 3 | 10/2 | 14/3 | 14/3 |
| hazard.g | 12 | 4 | 6/2 | 10/2 | 10/2 |
| qr42.g | 18 | 5 | 10/2 | 14/3 | 14/3 |
| nowick.g | 20 | 6 | 14/1 | 21/3 | 18/3 |
| wrdatab.g | 216 | 10 | 31/5 | 37/5 | 34/5 |
| pe-send-ifc.g | 108 | 10 | 52/5 | 84/5 | 80/5 |
| chu-stage.g | 40 | 8 | 9/2 | 9/2 | - |
| chu-fifo.g | 64 | 8 | 12/2 | 12/2 | - - |
| half.g | 14 | 5 | 7/2 | 7/2 | - |
| hund.g | 14 | 5 | 4/0 | 4/0 | - |
| mp-forward-pkt.g | 22 | 8 | 14/1 | 14/1 | - |
| nak-pa.g | 58 | 10 | 17/1 | 17/1 | - |
| ram-read-sbug.g | 39 | 11 | 21/3 | 22/3 | - |
| sbuf-ram-write.g | 64 | 12 | 19/2 | 21/2 | - |
| scsi.g | 20 | 5 | 11/3 | 11/3 | - |
| sendr-done.g | 9 | 4 | 5/1 | 5/1 | - |
| trimos-send.g | 336 | 9 | 18/6 | 24/6 | - |
| Lin.g | 2108 | 13 | 28/7 | 28/7 | - |
| alloc-outbound-chu1.g | 25 | 11 | 26/4 | 31/4 | - |
| alloc-outbound.g | 20 | 9 | 14/2 | 15/2 | - |
| rcv-setup.g | 11 | 5 | 7/1 | 8/1 | - |
| sbuf-read-ctl.g | 20 | 8 | 13/3 | 19/4 | - |
| sbuf-send-ctl.g | 27 | 8 | 16/3 | 27/4 | - |
| sbuf-send-pkt2.g | 26 | 9 | 16/3 | 25/4 | - |
| wrdata.g | 13 | 5 | 14/3 | 16/3 | - |
| Lavagno.g | 16 | 4 | 6/1 | 14/1 | - |
| pe-rcv-ifc.g | 44 | 11 | 33/4 | 68/5 | - |
| Total | | | 566/94 | 693/103<br>288/45 | 278/45 |

St: State number. Sg: Signal number.
Li: Literals. Me: Memories.
(a) Some STG/FCs are rewritten by merging apparent equivalent structures between conflict branches.
(b) "- " in entries denotes this result is not reported in [10].

Table 1: Experimental Results.