Cover Approximations for the Synthesis of Speed-Independent Circuits

Enric Pastor* Jordi Cortadella* Alex Kondratyev† Oriol Roig*

∗ Dept. of Computer Architecture
Universitat Politècnica de Catalunya
08071 Barcelona, Spain

† Computer Architecture Laboratory
The university of Aizu
Aizu-Wakamatsu, 965 Japan

Abstract

This paper presents a novel methodology for the synthesis of speed-independent circuits from Signal Transition Graphs (STGs). The pursued objective is to avoid the derivation of the reachability graph to calculate the logic equations for the outputs of the circuit, hence overcoming the state explosion problem. The circuit synthesis is performed by only analyzing the structure of the STG with efficient polynomial algorithms. The proposed techniques are applied to any live and safe *free-choice* STG providing significant improvement over previous state-based and structural methods. The same methodology can be directly extended to more general STGs. The developed structural synthesis tool has synthesized STGs (some of them being non-free-choice) with over 10^{12} markings.

1 Introduction

Signal Transition Graphs (STG) [13, 2] are an elegant event-based formalism for the specification of asynchronous circuits. STGs model the causality, concurrency and conflict between events, implicitly describing a potentially large state space by using a small set of events.

STGs are composed of an underlying Petri net [3] with transitions interpreted as rising or falling signal transitions of the circuit. Each reachable marking of the STG is assigned a binary code with the value of the signals in that marking. The synthesis process requires deriving logic equations for the output signals, and hence most synthesis tools [14, 15, 1] exhaustively generate the binary codes for all markings.

Speed-independent circuits (SI-circuits) are a class of asynchronous circuits that assume an unbounded delay on the gates of the network and negligible delay on the wires. STGs are specially well-suited to specify the behavior of SI-circuits. However, even for small STGs (in the number of transitions), the set of reachable markings of a highly concurrent system is very large, and exponential in the worst case.

Some efforts have been devoted to propose structural methods for synthesis [8, 16], but they have been usually devised for restricted classes of STGs that compromise the potential expressiveness of this formalism. In this framework, "structural" refers to algorithms working at the STG level without requiring the explicit generation of the reachable markings.

We present new structural techniques to perform the synthesis of SI-circuits from STGs. The algorithms have a polynomial complexity when the underlying Petri net is *free-choice* [4, 3] although we aim at extending its application to more general classes of STGs. We intend to provide an alternate synthesis methodology to complement the existing state-based tools. Finally, the proposed techniques have been experimentally implemented, reporting the preliminary results obtained for a large set of benchmarks.

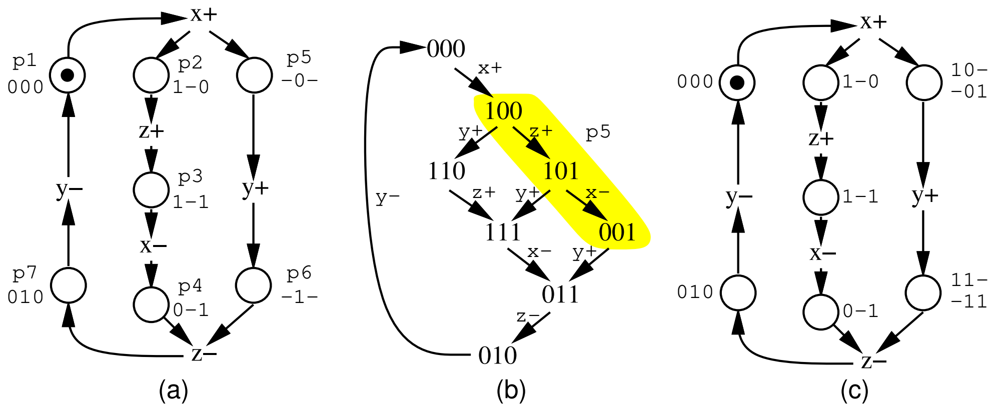

Figure 1: *xyz* (a)STG and covering cubes for places, (b)reachability graph, (c)refined covers.

2 Methodology Overview

This section presents an intuitive overview of the proposed methodology by using the well-known *xyz* example in Figure 1(a).

Assume that we want to derive a logic function for signal y and, in particular, for the markings in which transition y+ is enabled (*excitation region* of y+). This region corresponds to all markings in which place p_5 is marked. The *excitation region* of y+ is shadowed in the reachability graph obtained from the STG depicted in Figure 1(b).

Our objective is to calculate a set of cubes that safely cover the binary codes of the markings in the excitation region of y+, hence only overestimations with the *don't care* set will be allowed. A single cube approximation of the cover for a place p can be calculated as follows: if a signal transition can fire retaining place p marked, then the value of the signal is unknown while p is marked. Since transitions z+ and x− can fire while p_5 is marked, then the value of x and z is unknown in p_5. On the contrary, the value of y is exactly determined by analyzing the ordering relation of signal y with p_5. Thus, a single cube (-0-) can be derived for p_5. (Note that for places p_1 and p_7 the cubes can be exactly calculated *i.e.* (000) and (010) respectively.)

We can easily detect that cube (-0-) is an *overestimation* of the excitation region for y+. The marking (000) outside the excitation region for y+ is also covered by (-0-). Using these covers in a synthesis algorithm produce an erroneous circuit in which y+ may switch in marking (000). To overcome this situation we propose two strategies that can be combined to obtain a correct set of covers.

Cover refinement. The covers for a place can be *refined* by analyzing the concurrent relations with other places of the STG. In our case, we use the fact that p_5 can only be simultaneously marked with p_2, p_3 or p_4, to obtain a multi-cube cover. The cover of p_5 is intersected with the conjunction of the covers of p_2, p_3 and p_4. In this example this refinement is sufficient to guarantee a correct synthesis (Figure 1(c)). However, in general several refinements may be needed.

Signal insertion. To insert state signals —similarly as it is done to solve encoding conflicts[7, 9]— to disambiguate covers whose intersection produces contradictions for synthesis. This is illustrated in Figure 2, in which a new signal v distinguishes the covers of p_1 and p_5. Then, assuming signal order x y z v the cube (-0-1) correctly covers the excitation region of y+.

In this paper we will concentrate on the efficient application of the cover refinement techniques. For more information about signal insertion techniques refer to [6].

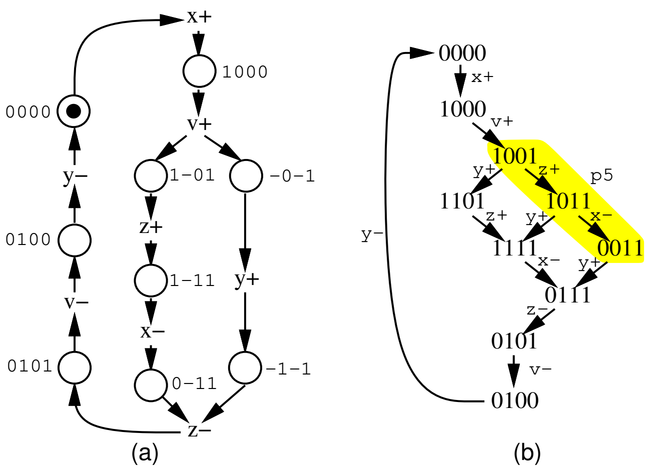

Figure 2: Encoded *xyz* (a)STG and covering cubes for places, (b)reachability graph.

3 Signal Transition Graphs

A *Petri net* (PN) is a 4-tuple $\Sigma = \langle \mathcal{P}, \mathcal{T}, \mathcal{F}, M_o \rangle$, where $\mathcal{P}$ is the set of places, $\mathcal{T}$ is the set of transitions, $\mathcal{F} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is the flow relation, and M_o is the initial marking. Given a node $x \in \mathcal{P} \cup \mathcal{T}$, its post-set and pre-set are denoted by $x^\bullet$ and ${}^\bullet x$ respectively. A *path* is a sequence $x_1 \ldots x_r$ of nodes such that $\forall i, 1 \leq i < r : (x_i, x_{i+1}) \in \mathcal{F}$. A path is called *simple* if no node appears on it more than once. A *Free-choice net* (FC net) is a PN such that every arc from a place is either a unique outgoing arc or a unique incoming arc to a transition.

A transition t is *enabled* in a marking M, denoted by $M[t\rangle$, when all places in ${}^\bullet t$ are marked. An enabled transition in M *fires*, removing a token from places in ${}^\bullet t$ and adding a token to places in $t^\bullet$, reaching a marking M' ($M[t\rangle M'$). A marking M is *reachable* from M_o if there is a sequence of *firings* $t_1 t_2 \ldots t_n$ that transforms M_o into M ($M_o[t_1 t_2 \ldots t_n\rangle M$), hence the sequence $t_1 t_2 \ldots t_n$ is *feasible*. The set of reachable markings from M_o is denoted by $[M_o\rangle$. A PN is *live* if every transition can be infinitely enabled through some *feasible* sequence of *firings* from any marking in $[M_o\rangle$. A PN is *safe* if no marking in $[M_o\rangle$ can assign more than one token to any place.

An STG is a triple $\langle \Sigma, S, \Lambda \rangle$, where Σ is a PN, S is a set of signals $\{\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots\}$, and Λ is a labeling function $\Lambda : \mathcal{T} \rightarrow S_{\mathrm{A}} \times \{+, -\}$, in which the transitions are interpreted as value changes on signals. Rising and falling transitions are denoted by $\mathsf{a}+$ and $\mathsf{a}-$ respectively, while $\mathsf{a}*$ denotes a generic transition. Multiple transitions for the same signal are distinguished by means of indices a_1+, a_2+ (in figures, instead of a_1+, $\mathsf{a}+/1$ is used). A transition $\mathsf{a}_i* \in next(\mathsf{a}_j*)$ if there exists a feasible sequence which contains no other transitions of signal a between a_j* and a_i* (the pair $(\mathsf{a}_j*, \mathsf{a}_i*)$ is **adjacent**).

Each marking is encoded with a *binary vector* of signal values by means of a labeling function $\lambda : [M_o\rangle \rightarrow \mathrm{B}^{|S_{\mathrm{A}}|}$. Function λ must *consistently encode* all markings, that is, no marking M can have an enabled rising (falling) transition $\mathsf{a}+$ ($\mathsf{a}-$) if $\lambda(M)_\mathsf{a} = 1$ ($\lambda(M)_\mathsf{a} = 0$).

An STG specification is composed of several signal transitions and causality relations between them. Hence, to derive the correspondence between transitions and markings different **signal regions** are defined:

- The **excitation region** $\mathrm{ER}(\mathsf{a}_i*)$ is the maximal connected set of markings in which $\mathsf{a}*$ is *enabled*.
- The **quiescent region** $\mathrm{QR}(\mathsf{a}_i*)$ is the maximal connected set of markings that can be reached from $\mathrm{ER}(\mathsf{a}_i*)$, and the **backward quiescent region** $\mathrm{BR}(\mathsf{a}_i*)$ is the maximal connected set of markings that can reach $\mathrm{ER}(\mathsf{a}_i*)$, in both cases without enabling any other transition a_j*.

A transition b_j* (and signal b) is **trigger** for a_i* if ER(a_i*) is reached by firing the transition b_j*. A signal b is **concurrent** with a_i* if b_j* is enabled in ER(a_i*) and the firing of a_i* or b_j* does not disable the other. A *trigger* transition b_j* is **non-persistent** to a_i* if some transition b_l* is *concurrent* with a_i*, otherwise it is *persistent*.

Figure 3 depicts the STG used as example along this work. The STG has two output r1 and r2, and two input signals c1 and c2 (with its transitions underlined). The underlying PN contains 12 places, 8 transitions, and of 16 reachable markings. Note that signal c2 is *non-persistent* for r1+ and r1−. Similarly, c1 is *non-persistent* for r2+ and r2−.

4 Synthesis Conditions

The restrictive operating discipline to guarantee the correctness of a SI-circuit requires a formal set of synthesis conditions [1, 5]. The synthesis conditions apply to the synthesis of every output signal, and follow a two-level *Sum-of-Functions* (SOF) architecture with a memory element on its output.

The **signal network** for each output signal a is constructed in three successive steps:

1. The switching of each transition a_i* is implemented by a **region cover** $R(\mathsf{a}_i*)$. The first level of the signal network consists of the set of rising and falling region covers.

2. The rising region covers are combined with an OR-gate to create the **set region network** (S_a), while the falling region covers are similarly combined creating the **reset region network** (R_a).

3. The set and reset region networks are connected to a C-element to create the output signal. S_a forces the memory element to switch from 0 to 1, while R_a produces the switching from 1 to 0.

The two-input C-element implements the output signal a with a next state function $\mathsf{a} = S_\mathsf{a}\overline{R}_\mathsf{a} + \mathsf{a}(S_\mathsf{a} + \overline{R}_\mathsf{a})$. The objective of the synthesis process is to simplify the complexity of the *region covers* while preserving the correct circuit operation.

The SOF architecture evolves under a **one-hot encoding** discipline on its rising (falling) region covers. Only one rising (falling) region cover can be responsible for a rising (falling) transition of the output signal. When one rising and one falling region cover are competing to produce a transition of the output signal, the C-element contains enough information to decide the correct ordering of the transitions [5].

The region cover $R(\mathsf{a}_i*)$ may cover markings in QR(a_i*) or markings in BR(a_i*) under certain restrictions [12]. Markings shared by several QR and BR regions of the same signal cannot be covered by any region cover. Otherwise, the *one-hot encoding* discipline of the SOF signal network is violated *i.e.* more than one rising (falling) region cover of the same signal is excited simultaneously. The subsets of the quiescent and backward quiescent regions that do not contain shared markings are called **restricted regions**, and denoted by $\text{QR}^r(\mathsf{a}_i*)$ and $\text{BR}^r(\mathsf{a}_i*)$ respectively. Restricted signal regions coincide with the signal regions when only exists two transitions for a.

Covering markings in the backward quiescent regions (allowing rising and falling region covers to be excited at the same time) is only possible because of the C-element [6]. Any marking in $\text{BR}^r(\mathsf{a}_i*)$ covered by $R(\mathsf{a}_i*)$ must be also covered by the region cover of some of its predecessor transitions a_j*.

A set of region covers $R(\mathsf{a}_i*)$ for the output signal a is **correct** if

1. Any marking in ER(a_i*) is covered by $R(\mathsf{a}_i*)$.

2. $R(\mathsf{a}_i*)$ does not cover any reachable marking outside $\text{ER}(\mathsf{a}_i*) \cup \text{QR}^r(\mathsf{a}_i*) \cup \text{BR}^r(\mathsf{a}_i*)$.

3. Markings in $\text{BR}^r(\mathsf{a}_i*)$ covered by $R(\mathsf{a}_i*)$ are covered by some $R(\mathsf{a}_j*)$ such that $\mathsf{a}_i* \in next(\mathsf{a}_j*)$.

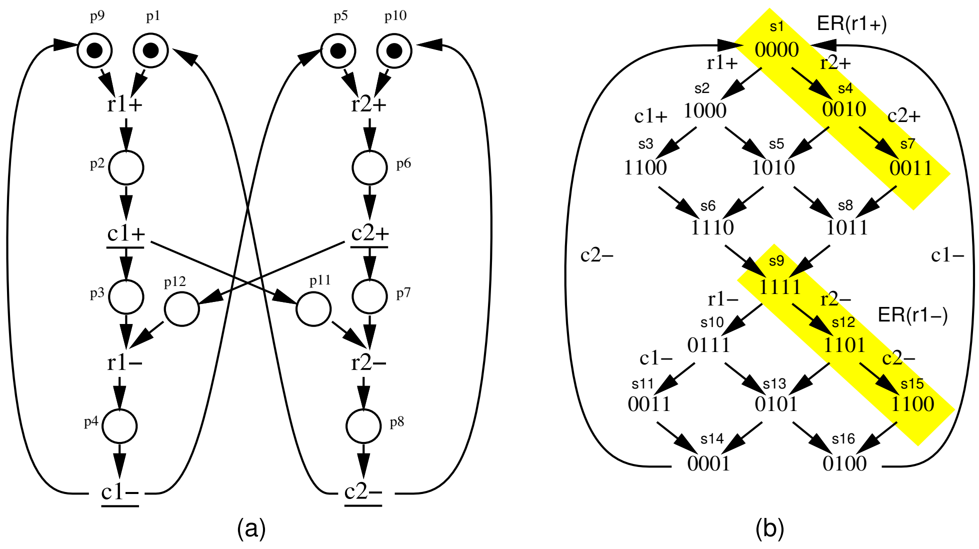

Figure 3: (a)STG specification, (b)reachability graph.

In Figure 3, output signal $\mathsf{r1}$ has two transitions $\mathsf{r1}+$ and $\mathsf{r1}-$, and its region covers $R(\mathsf{r1}+)$ and $R(\mathsf{r1}-)$ must cover $\text{ER}(\mathsf{r1}+) = \{\mathsf{s}_1, \mathsf{s}_4, \mathsf{s}_7\}$ and $\text{ER}(\mathsf{r1}-) = \{\mathsf{s}_9, \mathsf{s}_{12}, \mathsf{s}_{15}\}$. However, they may also cover markings in its quiescent and backward quiescent regions:

$$\begin{array}{lcllcl} \text{QR}(\mathsf{r1}+) & = & \{\mathsf{s}_2, \mathsf{s}_5, \mathsf{s}_8, \mathsf{s}_3, \mathsf{s}_6\}, & \text{BR}(\mathsf{r1}+) & = & \{\mathsf{s}_{10}, \mathsf{s}_{13}, \mathsf{s}_{16}, \mathsf{s}_{11}, \mathsf{s}_{14}\}, \\ \text{QR}(\mathsf{r1}-) & = & \{\mathsf{s}_{10}, \mathsf{s}_{13}, \mathsf{s}_{16}, \mathsf{s}_{11}, \mathsf{s}_{14}\}, & \text{BR}(\mathsf{r1}-) & = & \{\mathsf{s}_2, \mathsf{s}_5, \mathsf{s}_8, \mathsf{s}_3, \mathsf{s}_6\}. \end{array}$$

Every transition of the output signal must be **monotonic**, that is guarantee that $R(\mathsf{a}_{i*})$ changes exactly twice to implement the specified transition, where the rising change is at a marking in $\text{BR}^r(\mathsf{a}_{i*}) \cup \text{ER}(\mathsf{a}_{i*})$ and the falling change is produced before reaching the next excitation region of signal a. A region cover $R(\mathsf{a}_{i*})$ is *monotonic* iff

1. $\forall M \in \text{QR}^r(\mathsf{a}_{i*})$ covered by $R(\mathsf{a}_{i*})$, $\forall M' \in \text{QR}^r(\mathsf{a}_{i*}) \,:\, M'[t\rangle M, \Rightarrow M'$ is covered by $R(\mathsf{a}_{i*})$,

2. $\forall M \in \text{BR}^r(\mathsf{a}_{i*})$ covered by $R(\mathsf{a}_{i*})$, $\forall M' \in \text{BR}^r(\mathsf{a}_{i*}) \,:\, M[t\rangle M', \Rightarrow M'$ is covered by $R(\mathsf{a}_{i*})$.

Again in Figure 3, $R(\mathsf{r1}+)$ is non-*monotonic* if in addition to $\text{ER}(\mathsf{r1}+)$ it covers (for example) markings s_3 and s_6. Then $R(\mathsf{r1}+)$ will switch $0 \to 1$ entering $\text{ER}(\mathsf{r1}+)$, $0 \to 1$ leaving $\text{ER}(\mathsf{r1}+)$, and may switch again $0 \to 1 \to 0$ going through markings s_3 and s_6. However, covering markings s_2, s_5 and s_8 maintains the monotony of the region cover.

It can be guaranteed that a SI-circuit implementation is obtained from *correct* and *monotonic* region covers [1, 5]. Based on this result, we tackle the efficient analysis of the signal regions in order to show structurally verify the monotonous cover conditions, hence without generating the markings of the STG.

5 STG Structural Analysis

This section defines the relations between the structure of a *free choice* STG and its reachable markings. We intend to implicitly manipulate the STG without explicitly compute any of its markings. The dynamic behavior of the STG is determined by analyzing the pairwise concurrency between transitions, and the reachable markings approximated by sets of cubes.

5.1 Concurrency Relation

The behavior the STG is indirectly determined by analyzing which pairs of transitions are concurrently enabled —the *Concurrency Relation* ($\mathcal{CR}$). Two transitions are *concurrent* if there is a reachable marking where both transitions can fire without disabling each other. $\mathcal{CR}$ can be extended to all the nodes of an STG [9], *e.g.* two places p_1 and p_2 are *concurrent* if there is a reachable marking M in which both of them are marked. Two nodes (u_1, u_2) are *concurrent* if $(u_1, u_2) \in \mathcal{CR}$. Additionally, the *Signal Concurrency Relation* ($\mathcal{SCR}$) between signals and nodes of an STG can be analyzed by using the $\mathcal{CR}$ of the STG. A signal a and node u_j are said to be *concurrent* ($(\mathsf{a}, u_j) \in \mathcal{SCR}$) if exists a transition a_i* concurrent to u_j ($(\mathsf{a}_i*, u_j) \in \mathcal{CR}$).

The overall proposed synthesis cycle has a polynomial complexity except for the $\mathcal{CR}$ relation computation [11]. Polynomial-time algorithms to compute the $\mathcal{CR}$ relation for live and safe FC PNs are presented in [9], while other techniques can be used to extend the synthesis process to more general classes of STG.

Table 1 depicts the $\mathcal{SCR}$ relation for Figure 3. Concurrent pairs are denoted **co**. (Specially note that all signals behave concurrently for places p_1, p_5, p_{11}, and p_{12}.)

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
r1	**co**				**co**	**co**	**co**	**co**		**co**	**co**	**co**
c1	**co**				**co**	**co**	**co**	**co**		**co**	**co**	**co**
r2	**co**	**co**	**co**	**co**	**co**				**co**		**co**	**co**
c2	**co**	**co**	**co**	**co**	**co**				**co**		**co**	**co**

Table 1: $\mathcal{SCR}$ between signals and places for the STG in Figure 3.

5.2 Marking Approximation

We will approximate the STG markings by means of the indirect analysis of its binary codes. Markings are partitioned into sets of regions with respect to the STG nodes, using a single cube —the **covering cube**— to approximate the binary codes in each region.

The **marked region** MR(p) is the set of markings in which place p is marked. The **enabled region** ER(t) is the set of markings in which transition t is enabled. The enabled region of a transition is equivalent to the intersection of the marked regions of its predecessor places, *i.e.* ER$(t) = \bigcap_{p \in {}^\bullet t}$ MR(p). The marked regions for places p_1 and p_9 in Figure 3 are MR$(p_1) = \{\mathsf{s}_1, \mathsf{s}_4, \mathsf{s}_7, \mathsf{s}_{16}, \mathsf{s}_{15}\}$ and MR$(p_9) = \{\mathsf{s}_1, \mathsf{s}_4, \mathsf{s}_7, \mathsf{s}_{11}, \mathsf{s}_{14}\}$ hence, the enabled region for transition r1+ is ER(r1+) $=$ MR$(p_1) \cap$ MR$(p_9) = \{\mathsf{s}_1, \mathsf{s}_4, \mathsf{s}_7\}$.

The $\mathcal{SCR}$ relation provides the information to decide if the value of a signal a is constant in a marked region. However, the relative position of the place with respect to the transitions a_i* is needed to determine the signal value.

We can characterize the position of a place or a transition with respect to *adjacent* transitions of a signal. To determine if $\mathsf{a}_i* \in next(\mathsf{a}_j*)$ it must be found a simple path between a_j* and a_i* exclusively composed of nodes non-concurrent to a [10]. The *Interleave Relation* ($\mathcal{IR}$) defines the interleaving between a node and pairs of *adjacent* transitions. A node u_j is *interleaved* with a_j* and a_i* ($u_j \in \mathcal{IR}(\mathsf{a}_j*, \mathsf{a}_i*)$), if there exists a simple path from a_j* to a_i* containing u_j and without nodes concurrent to a. For example, in Figure 1(a), p_2 is interleaved with (x+, x−), whereas p_5 is not.

The **cover cubes** $\mathcal{C}_p$ and $\mathcal{C}_t$ are the cubes with the greatest number of literals that cover MR(p) and ER(t) respectively. The value of signal a in $\mathcal{C}_p$ depends on the signal concurrency between a and p, and the position of p with respect to the transitions of a. The value of each literal $\mathcal{C}_p^{\mathsf{a}}$ in the cover cube of place p is computed:

$$\mathcal{C}_p^{\mathsf{a}} = \begin{cases} \mathsf{a} & \text{if } \exists \text{ adjacent } (\mathsf{a}_i+, \mathsf{a}_i-) : p \in \mathcal{IR}(\mathsf{a}_i+, \mathsf{a}_i-) \ , \\ \overline{\mathsf{a}} & \text{if } \exists \text{ adjacent } (\mathsf{a}_i-, \mathsf{a}_i+) : p \in \mathcal{IR}(\mathsf{a}_i-, \mathsf{a}_i+) \ , \\ - & \text{if } (\mathsf{a}, p) \in \mathcal{SCR} \ . \end{cases}$$

The cover cube $\mathcal{C}_t$ is computed in terms of its predecessor places: $\mathcal{C}_t = \bigcap_{p \in \bullet t} \mathcal{C}_p$.

Table 2 depicts the cover cubes for the nodes in Figure 3. Due to the concurrency relations the cover cubes for places p_1, p_5, p_{11}, and p_{12} do not contain any literal. It can be also determined that the cover cubes for $\mathsf{r1}+$ and $\mathsf{r1}-$ are an imprecise approximation of markings in ER($\mathsf{r1}+$) and ER($\mathsf{r1}-$), *i.e.* $\mathcal{C}_{\mathrm{r1}+}$ is covering s_{14} in MR(p_8), and $\mathcal{C}_{\mathrm{r1}+}$ is covering s_6 in MR(p_6).

$\mathcal{C}_{p_1}$	----	$\mathcal{C}_{p_2}$	10--	$\mathcal{C}_{\mathrm{r1}+}$	00--	$\mathcal{C}_{\mathrm{r1}-}$	11--
$\mathcal{C}_{p_3}$	11--	$\mathcal{C}_{p_4}$	01--	$\mathcal{C}_{\mathrm{c1}+}$	10--	$\mathcal{C}_{\mathrm{c1}-}$	01--
$\mathcal{C}_{p_5}$	----	$\mathcal{C}_{p_6}$	--10	$\mathcal{C}_{\mathrm{r2}+}$	--00	$\mathcal{C}_{\mathrm{r2}-}$	--11
$\mathcal{C}_{p_7}$	--11	$\mathcal{C}_{p_8}$	--01	$\mathcal{C}_{\mathrm{c2}+}$	--10	$\mathcal{C}_{\mathrm{c2}-}$	--01
$\mathcal{C}_{p_9}$	00--	$\mathcal{C}_{p_{10}}$	--00				
$\mathcal{C}_{p_{11}}$	----	$\mathcal{C}_{p_{12}}$	----				

Table 2: Cover cubes, with signal order (r1 c1 r2 c2), for the example in Figure 3.

6 Signal Region Approximation

A precise signal region approximation is needed to effectively check the synthesis conditions. We determine the signal regions in terms of sets of nodes that define the structure of the region, and a logic function for each node that defines the binary codes of the markings in the region.

6.1 Signal Region Structure

The *excitation region* ER(a_i*) is defined by the **excitation transition set** ETS(a_i*) containing transition a_i*. The *quiescent region* QR(a_i*) is defined by the **quiescent place set** QPS(a_i*). A place belongs to QPS(a_i*) if it is "between" a_i* and a_j*, $\mathsf{a}_j* \in next(\mathsf{a}_i*)$. This can be expressed by using the *Interleave Relation*, *i.e.*

$$p \in \mathrm{QPS}(\mathsf{a}_i*) \Leftrightarrow \exists \mathsf{a}_j* \in next(\mathsf{a}_i*) : p \in \mathcal{IR}(\mathsf{a}_i*, \mathsf{a}_j*) \ .$$

The *backward quiescent region* BR(a_i*) is defined by the **backward quiescent place set** BPS(a_i*). A place belongs to BPS(a_i*) if it is "between" a_j* and a_i*, $\mathsf{a}_i* \in next(\mathsf{a}_j*)$, *i.e.*

$$p \in \mathrm{BPS}(\mathsf{a}_i*) \Leftrightarrow \exists \mathsf{a}_j*,\ \mathsf{a}_i* \in next(\mathsf{a}_j*) : p \in \mathcal{IR}(\mathsf{a}_j*, \mathsf{a}_i*) \ .$$

The *restricted quiescent region* is approximated by the **restricted quiescent place set** QPS^r(a_i*), computed as the *quiescent place set* minus the places shared with the QPS sets of other transitions of the same signal, *i.e.* $\mathrm{QPS}^r(\mathsf{a}_i*) = \mathrm{QPS}(\mathsf{a}_i*) - \bigcup_{j \neq i} \mathrm{QPS}(\mathsf{a}_j*)$. Similarly, the **restricted backward quiescent place set** $\mathrm{BPS}^r(\mathsf{a}_i*) = \mathrm{BPS}(\mathsf{a}_i*) - \bigcup_{j \neq i} \mathrm{BPS}(\mathsf{a}_j*)$.

In Figure 3, the set QPS($\mathsf{r1}+$) contains the places in the path $\{\mathsf{r1}+, p_2, \mathsf{c1}+, p_3, \mathsf{r1}-\}$, while the set QPS($\mathsf{r1}-$) contains the places in the path $\{\mathsf{r1}-, p_4, \mathsf{c1}-, p_9, \mathsf{r1}+\}$. Similarly, the BPS sets can be computed.

6.2 Signal Region Coding

Each one of the nodes used to structurally define the signal regions is assigned a logic function, named **cover function** and denoted $\mathsf{cv}(u)$, that approximates the binary codes of the markings in the signal region of u. Our methodology starts from the *cover cubes* as initial *cover functions*.

In that case the QPS(a_{i*}) and BPS(a_{i*}) sets are imprecise approximations at their boundaries. By definition QPS(a_{i*}) e.g. contains all the places "between" a_{i*} and $\mathsf{a}_{j*} \in next(\mathsf{a}_{i*})$. However the cover function of places in $^\bullet(\mathsf{a}_{j*})$ covers also ER(a_{j*}). To avoid overlapping QPS(a_{i*}) and ETS(a_{j*}), places in $^\bullet(\mathsf{a}_{j*})$ used in QPS(a_{i*}) should be **refined** into $\mathsf{cv}^{\text{ref}}(p) = \mathsf{cv}(p) - \mathsf{cv}(\mathsf{a}_{j*})$. (Note that refining the places at the quiescent region also refines the backward quiescent region.)

Cover functions computed as cover cubes give an initial approximation of the corresponding signal regions, but usually are *overestimated* covering markings outside the correct region. The overestimations in the cover functions can be eliminated using two approaches:

1. Add state signals that eliminate the ambiguity of the cover function (see example in Figures 1 and 2).
2. Further **refine** the cover functions to increase the cubes in the cover and therefore its precision.

Overestimations are not acceptable in $\mathsf{cv}(\mathsf{a}_{j*})$ when recomputing $\mathsf{cv}^{\text{ref}}(p)$ with $p \in {}^\bullet(\mathsf{a}_{j*})$, because it may lead to erroneous **underestimations** of the regions. Non-persistent transitions and the pairwise intersection of the cover functions for several sets of places provide the necessary information to avoid the underestimation and guarantee the correctness of the methodology [10]. Both the *correctness* and the *monotony* synthesis conditions can be checked by using the signal region approximations and the refined cover functions.

In Figure 3, the cover functions for places p_1, p_5, p_{11}, and p_{12} are clearly overestimated. However, they are not used to approximate the signal regions. On the other hand, transitions $\mathsf{r1}+$, $\mathsf{r1}-$, $\mathsf{r2}+$, and $\mathsf{r2}-$ are overestimated due to the *non-persistent* trigger transitions and should be refined.

7 Cover Function Refinement

Exploiting the structure of casual relations between STG nodes we can **refine** the initial approximation by other cover functions in the STG. The set of places in the STG provide a complete and global view of the reachability graph that is lost when looking to a single place. Places that can be concurrently marked with respect a place p reflect the causal relations that are lost in the cover cubes and provide the main idea under refinement.

Formally, the refinement of $\mathsf{cv}(p)$ [1] by the concurrent places to p results in a cover that is obtained as the product of $\mathsf{cv}(p)$ with the sum of the cover functions of the places $p_i : (p_i, p) \in \mathcal{CR}$, *i.e.*

$$\mathsf{cv}^{\text{ref}}(p) = \mathsf{cv}(p) \cdot \sum_{(p_i,p)\,\in\,\mathcal{CR}} \mathsf{cv}(p_i) \ .$$

Such refinement procedure is safe for live FC STGs, that is no marking from the marked region is lost due to refinement. Place p is refined only by its concurrent places hence, the only markings that can be incorrectly removed due to refinement are markings M with a token in p that does not mark any place p_i such that $(p_i, p) \in \mathcal{CR}$. However, if such a marking exists it will contradict the liveness of the STG.

Refinement can be iteratively applied to improve the precision of the cover functions, but increasing the number of required cubes that in extreme cases may be comparable to the number of reachable markings. However, in general a few number of cubes is required to achieve good approximations. The idea of

[1] It is enough to consider the refinement for places because cover functions for a transition are obtained from them.

STG	states	cubes	area	CPU (sec.) SIS	ASSASSIN	our tool
tsbmsi	1024	143	744	43	256	1
tsbmSIBRK	4730	298	1136	1876	12219	8
master-read-1	2254	117	610	250	1252	2
master-read-2	18856	2786	1052	> 9 h.	> 24 h.	34
master-read-3	21848	2802	1102	> 9 h.	> 24 h.	35
par4	1303	95	500	80	271	1
par8	1.7×10^6	342	920	$*$	$*$	7
par16	2.8×10^{12}	1190	1768	$*$	$*$	45

Table 3: CPU time for synthesis: comparison with SIS and ASSASSIN ($*$ failure due to memory limits).

refinement was illustrated in Section 2 where one iteration was enough to get the exact cover for non-persistent transition y+.

In some particular cases the refinement methodology cannot be directly applied. Refine transition r1+ in Figure 3 is equivalent to refine place p_1. But the cover function for some of the concurrent places to p_1 does not contain any literal, hence preventing any practical refinement (places $\{p_3, p_4, p_5, p_6, p_7, p_9, p_{10}, p_{12}\}$ are concurrent to p_1).

Then, the refinement should be restricted to subsets of the concurrent places. The objective is to avoid those places that do not help eliminating the overestimations. A *refinement set* $\mathrm{RS}(p)$ for p is a maximal subset of its concurrent places, such that

1. no pair of places p_i and p_j in $\mathrm{RS}(p)$ is mutually concurrent $(p_i, p_j) \in \mathcal{CR}$,
2. all places p_i in $\mathrm{RS}(p)$ at least contain a literal in $\mathbf{cv}(p_i)$.

Again, this restricted refinement procedure is safe for free-choice live STGs, otherwise the liveness of the STG is contradicted.

Two RS sets can be computed for place p_1. $\mathrm{RS}_1(p_1) = \{p_{10}, p_6, p_7\}$ that will refine the sequences that evolve concurrently to r1+, and $\mathrm{RS}_2(p_1) = \{p_3, p_4, p_9\}$ that refines the sequences that evolve sequentially to r1+. Refining by $\mathrm{RS}_1(p_1)$ leads to:

$$\mathsf{cv}^{\mathrm{ref}}(p_1) = (\text{----}) \cdot \{\text{--00} + \text{--10} + \text{--11}\} = \{\text{---0} + \text{--1-}\} \ , \mathsf{cv}(\mathsf{r1+}) = \mathsf{cv}(p_9) \cdot \mathsf{cv}(p_1) = \{\text{10-0} + \text{101-}\} \ .$$

eliminating the overestimation of the excitation region. While refining by $\mathrm{RS}_2(p_1)$ leads to:

$$\mathsf{cv}^{\mathrm{ref}}(p_1) = (\text{----}) \cdot \{\text{11--} + \text{01--} + \text{00--}\} = \{\text{-1--} + \text{0---}\} \ , \mathsf{cv}(\mathsf{r1+}) = \mathsf{cv}(p_9) \cdot \mathsf{cv}(p_1) = \{\text{00--}\} \ .$$

8 Experimental Results

We have combined the synthesis conditions and the structural signal region analysis into a heuristic minimization algorithm. The objective is to reduce the number of cubes and literals required to implement the monotonous region covers. Any successfully applied minimization reduces the complexity of the implementation, but assuring that the transformed implementation remains correct.

To demonstrate the superiority of structural over state-based methods, we provide some experiments for large state space STGs. The compared CPU times (in a SUN SPARC20) with SIS[14] and ASSASSIN[15] are shown in Table 3. The number of markings and the used cubes demonstrates the efficiency of the refinement methodology. (Note that the *master-read* 1 and 2 contain many non-persistent transitions hence requiring multiple refinements.) The running time improvement is also evident.

9 Conclusions

Structural techniques for the analysis and synthesis of STGs can fill the gap when the state space becomes unmanageable by state graph level algorithms. We have presented new methods of polynomial complexity to synthesize free choice STGs that could be be extended to more general classes of STGs. The area/time results show that the proposed methodology obtains efficient implementations in short CPU times, while some of the existing tools have not been able to synthesize the largest circuits. Future work will be devoted to fully characterize the class of STGs that can be handled by the proposed techniques.

Acknowledgments

This work has been supported by the Ministry of Education of Spain (CICYT) under contract TIC 95-0419 and Departament d'Ensenyament de la Generalitat de Catalunya. We are also indebted to Michael Kishinevsky for many valuable discussions.

References

[1] Peter A. Beerel and Teresa H. Meng. Automatic gate-level synthesis of speed-independent circuits. In *Proc. ICCAD*, pages 581–586, November 1992.

[2] Tam-Anh Chu. *Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications*. PhD thesis, MIT, June 1987.

[3] J. Desel and J. Esparza. *Free Choice Petri Nets*. Cambridge University Press, Cambridge, Great Britain, 1995.

[4] M. Hack. *Analysis of production schemata by Petri nets*. M.s. thesis, MIT, February 1972.

[5] A. Kondratyev, M. Kishinevsky, B. Lin, P. Vanbekbergen, and A. Yakovlev. Basic gate implementation of speed-independent circuits. In *Proc. DAC*, pages 56–62, June 1994.

[6] A. Kondratyev, M. Kishinevsky, and A. Yakovlev. On hazard-free implementation of speed-independent circuits. In *Proc. of the ASP-DAC'95*, pages 241–248, August 1995.

[7] Luciano Lavagno, Cho W. Moon, Robert K. Brayton, and A. Sangiovanni-Vincentelli. Solving the state assignment problem for Signal Transition Graphs. In *Proc. DAC*, pages 568–572, June 1992.

[8] K.-J. Lin and C.-S. Lin. Automatic synthesis of asynchronous circuits. In *Proc. DAC*, pages 296–301, June 1991.

[9] Enric Pastor and Jordi Cortadella. Polynomial algorithms for the synthesis of hazard-free circuits from signal transition graphs. In *Proc. ICCAD*, pages 250–254, November 1993.

[10] Enric Pastor, Jordi Cortadella, Alex Kondratyev, and Oriol Roig. Structural algorithms for the synthesis of speed-independent circuits from signal transition graphs. Technical Report RR-95/35, UPC/DAC, October 1995.

[11] Enric Pastor, Jordi Cortadella, Alex Kondratyev, and Oriol Roig. Structural methods for the synthesis of speed-independent circuits. In *Proc. EDAC-ETC-EuroASIC*, March 1996. To appear.

[12] Enric Pastor, Jordi Cortadella, and Oriol Roig. A new look at the conditions for the synthesis of speed-independent circuits. In *Proc. Fifth Great Lakes Symposium on VLSI*, pages 230–235, May 1995.

[13] L. Ya. Rosenblum and A. V. Yakovlev. Signal graphs: From self-timed to timed ones. In *International Workshop on Timed Petri Nets*, pages 199–206, July 1985.

[14] Ellen M. Sentovich, Kanwar Jit Singh, Luciano Lavagno, Cho Moon, Rajeev Murgai, Alexander Saldanha, Hamid Savoj, Paul R. Stephan, Robert K. Brayton, and A. Sangiovanni-Vincentelli. SIS: A system for sequential circuits synthesis. Technical Report M92/41, UCB/ERL, May 1992.

[15] C. Ykman-Couvreur, B. Lin, and H. De Man. ASSASSIN: A synthesis system for asynchronous control circuits. Technical report, IMEC, September 1994. User and Tutorial manual.

[16] C. Ykman-Couvreur, Bill Lin, Gert Goossens, and Hugo De Man. Synthesis and optimization of asynchronous controllers based on extended lock graph theory. In *Proc. EDAC*, pages 512–517, February 1993.