Journal of VLSI Signal Processing, 7, 101–115 (1994)

A Generalized State Assignment Theory for Transformations on Signal Transition Graphs*

PETER VANBEKBERGEN
IMEC Laboratory, Kapeldreef 75, B-3001 Leuven, Belgium

BILL LIN
IMEC Laboratory, Kapeldreef 75, B-3001 Leuven, Belgium

GERT GOOSSENS
IMEC Laboratory, Kapeldreef 75, B-3001 Leuven, Belgium

HUGO DE MAN
ESAT Laboratory, Katholieke Universiteit, K. Mercierlaan 94, B-3001, Leuven, Belgium

Received April 12, 1992; Revised November 23, 1992.

Abstract. In this article, we propose a global assignment theory for *encoding* state graph transformations. A constraint satisfaction framework is proposed that can guarantee *necessary and sufficient* conditions for a state graph assignment to result in a transformed state graph that is free of critical races. Performing transformations at the state graph level has the advantage that the requirements imposed on the initial STG are very weak. Unlike previous methods, the initial STG need not be a live, safe, nor a free choice net. The only requirement is that the corresponding initial state graph is finite, connected, and has a consistent state assignment. Hence, a very broad range of signal transition graphs can be synthesized. The transformations achievable using the proposed framework correspond to very complex transformations on signal transition graphs. Even transformations that convert a free choice net into a correct non-free choice net and a 1-safe net into a correct 2-safe net are feasible. Addition of transitions that do not follow the Petri net firing rule is also possible. Even though our method can search a large solution space, we will show that it is possible to solve the problem in an exact way in acceptable CPU times in many practical cases.

1. Introduction

Automatic synthesis of asynchronous control circuitry is rapidly gaining recognition as a pivotal problem. This is partly driven by system-level design issues, power consumption issues, and recent technological developments. The goal in this research is to address the asynchronous synthesis problem from a general Signal Transition Graph (STG) formalism that can model multiple transitions, concurrency, conflict (or choices), and sequencing. The STG formalism is a nice specification methodology for those types of asynchronous controllers for which it is natural to reason on signal transitions instead of signal levels. Examples are a.o. FIFO-controllers [1], controllers in signal processing chips and handshake circuits [2].

*Research supported by the ESPRIT 2260 (SPRITE) program of the EC.

Current synthesis techniques can be broadly taxonomized into two categories: those that work at the signal transition graph level and those that work at the state graph level. These techniques are aimed at satisfying the so-called complete state coding (CSC) requirement [1] so that hazard-free logic may be derived [3], [4]. At the signal transition graph level techniques have been previously developed for transforming an STG to satisfy the state coding requirements [5], [6]. This technique is only valid for a limited class of Petri nets (live-safe marked graphs [7]). The method proposed in this article can handle a much broader class of Petri nets. Transformations at STG level currently can only guarantee sufficient conditions with respect to the state coding. Moreover, it is difficult to determine at the signal transition graph level how new signals should be added to the STG. Consequently it is difficult to come up with a solution with a minimal number of newly added signals. On the other hand, [5] is able to perform

transformations that reduce the concurrency of the initial STG. This is currently not possible in our method.

Lavagno et al. [8] have recently proposed to solve the state coding probem at the state graph level by mapping an initial state graph into a flow table synthesis problem. The state coding problem is then solved by using flow table minimization and state assignment methods [3], [9]. This method only handles a restricted class of Petri nets namely live-safe free choice nets [7]. They have shown that the solutions achievable in their framework correspond only to a restricted class of signal transition graph transformations. Even within these restrictions they only guarantee a sufficient condition for CSC-satisfaction. [10] also presents a methodology to satisfy CSC. This method only handles a restricted class of Petri nets (marked graphs) [7]. There are also severe limitations on the transformations possible in this framework. And finally no exact solution to the problem is proposed. They eliminate CSC-violations through iteration of the encoding procedure.

In this article, we propose a global assignment theory for *encoding* state graph transformations. A transformation on the state graph will be represented by a generalized assignment on the state graph. We present a constraint satisfaction framework that can guarantee necessary and sufficient conditions for a state graph assignment to result in a transformed state graph that satisfies the complete state coding requirement. Necessary means that within certain restrictions (cfr. Section 3.2) any transformation can be performed in this framework. Performing transformations at the state graph level has the advantage that the requirements imposed on the initial STG are very weak. Unlike previous methods, the initial STG need not be a live, safe, nor a free choice net. The only requirement is that the corresponding initial state graph is finite, connected, and has a consistent state assignment (cfr. Section 3.1). Hence, a very broad range of signal transition graphs can be synthesized. The transformations achievable using the proposed framework correspond to very complex transformations on signal transition graphs. Even transformations that convert a free choice net into a correct non-free choice net and a 1-safe net into a correct 2-safe net are feasible. It is also possible to add transitions that do not follow the Petri net firing rule [11]. Even though our method can search a large solution space, we will show that it is possible to solve the problem in an exact way in acceptable CPU times in many practical cases.

The STG formalism is not the only specification methodology for asynchronous circuits. CSP (communicating sequential processes) provides a model for asynchronous circuits illustrated with algebraic laws. The interpretation of different statements is based on trace theory. Concurrent programming languages such as CSP generally impose a series/parallel graph structure on the description which is sometimes too restrictive for speed-independent circuits. Although the synthesis methods are theoretically well-founded, completeness of either the requirements to satisfy nor of the transformations that satisfy the requirements have been shown. The synthesis methods proposed in this area are all rule-based.

The FSM-model is a well-known model to design asynchronous controllers. In our opinion, this model has two basic deficiencies. The model dictates that at each instant the system must be in only one state. It cannot therefore describe concurrent operations in a direct and succinct way. The basic items the model works with are signal levels. This is not appropriate for many applications, especially when time is involved. Numerous techniques have been proposed to come up with state assignment techniques that guarantee a race-free implementation. Each technique has its own limitations so that only a small part of the solution space is covered.

The remainder of this article is organized as follows. In the next section, working notations and terminology are explained. In this section, our definition of *equivalent state graphs* is explained and a notion of *consistent state assignment* is introduced. In Section 3, our formulation of the global problem is presented, which is based on a generalized state assignment on the state graph. A procedure is given for deriving a new *encoded* state graph from a given generalized state assignment. In this section, necessary and sufficient conditions are given for a generalized state graph assignment to result in a new state graph that satisfies the consistent state assignment property and the semi-modularity condition. These conditions guarantee the existence of a new state graph. Additional conditions are required on the state assignment to ensure CSC satisfaction. The necessary and sufficient conditions to ensure the CSC property are given in Section 4. In Section 5, several examples are shown to illustrate the power of the framework. In Section 6, we show how the necessary and sufficient conditions given in Sections 3 and 4 can be solved efficiently using a Boolean constraints satisfaction framework. Specifically, the conditions can be reduced to Boolean constraints where known Boolean satisfiability algorithms can be employed. This Boolean constraints satisfaction framework is very general in the

sense that new constraints can be easily incorporated. Experimental results are given in Section 7. Finally, concluding remarks are given in Section 8.

2. Notations and Terminology

Signal Transition Graphs (STG) [1] are used to specify the behavior of asynchronous digital control circuits. The vertices of such a graph represent the rising and falling transitions of the signals of a control circuit. s_i^+ denotes an up-transition, s_i^- denotes a down-transition and s_i^* denotes any transition of s_i.

An STG is a Petri net Σ [7] represented by the 4-tuple $\Sigma = \langle P, T, F, m_0 \rangle$. T is the set of transitions (as described above), P is the set of places and F is called the flow relation $F \subseteq (P \times T) \cup (T \times P)$. m_0 is the set of tokens which represents the initial state of the system. S denotes the set of all signals in the STG. $S_{NI} \subseteq S$ denotes the set of all non-input signals for which logic is to be generated.

A transition t is said to be enabled in a state (or marking) m in an STG if all the input places of t carry a token in marking m. This is denoted by $m[t\rangle$. The transformation of m into m' by firing a transition t is denoted by $m[t\rangle m'$ or by $m' = \delta(m, t)$ with δ a partial function: $M_S \times T \rightarrow M_S$. M_S denotes the set of all states (markings). This is done by removing all tokens from the input places of t and placing them in the output places of t. A state graph Φ can be derived from Σ by using the firing rule above. Φ is denoted by the triple $\langle M_S, T, \delta \rangle$. σ represents a sequence of transitions. The transformation of m into m' by firing the sequence of transitions σ is denoted by $m[\sigma\rangle m'$ or by $m' = \delta(m, \sigma)$.

A transition t is called *semi-modular* if and only if

$$\forall m, m' \in M_S : m[t\rangle \wedge \exists t' \neq t : m[t'\rangle m' \Rightarrow m'[t\rangle$$

Two transitions are said to be *concurrent* ($t_1 \parallel t_2$) if and only if there exists a state, reachable from the initial state, in which both transitions are enabled and both transitions are semi-modular.

A binary vector $\langle m(1), \ldots, m(n) \rangle$ of signal values is assigned to every state in the state graph according to the signals $\{s_1, s_2, \ldots, s_n\}$.

Definition 2.1. (State assignment) $m' = \delta(m, t) \Rightarrow$

- if $t = s_i^+$ then $m(i) = 0$ and $m'(i) = 1$.
- if $t = s_i^-$ then $m(i) = 1$ and $m'(i) = 0$.
- else $m'(i) = m(i)$.

These vectors that are in fact the code assigned to the states are used to derive the logic from the STG and state graph.

Definition 2.2. (Consistent state assignment) If the states can be encoded according to the rules given in Definition 2.1, the state graph is said to have a consistent state assignment.

This intuitively means that up- and down-transitions have to alternate in the state graph.

Chu has proven the following fundamental theorem [1]:

THEOREM 2.1. One can derive logic equations from Φ iff

$$\forall m, m' \in M_S : \exists k : m(k) \neq m'(k) \vee$$

$$\forall s_k \in S_{NI} : m[s_k^*\rangle \Leftrightarrow m'[s_k^*\rangle$$

The state graph is said to satisfy the Complete State Coding (CSC) requirement if and only if the state graph satisfies Theorem 2.1. According to Theorem 2.1 the CSC requirement is the necessary and sufficient requirement that should be satisfied before hazard-free logic equations may be derived from the state graph. Most of the state graphs specifying the behavior of asynchronous circuits will not satisfy the CSC-property. Therefore a transformation should be performed on the initial STG in order to satisfy the CSC-requirement. So it may be necessary to add new signals to the STG. These new signals will be called *state signals*. The transitions of the state signals are not observable for the environment. The set of transitions of state signals is denoted by T_{Nobs} (the set of non-observable signals).

2.1. Equivalent State Graphs

In this section the condition will be defined under which two state graphs are equivalent. Two state graphs are equivalent if the environment that observes the state graphs cannot distinguish them. So from the viewpoint of the environment the two state graphs have exactly the same behavior.

The set of transitions T of a state graph Φ can be partitioned into three subsets. The set of transitions of input signals T_I, the set of transitions of output signals T_O and the set of transitions of non-observable signals T_{nobs}. For the set $\{T_I \cup T_O\}$ (the set of observable signals) we will also use the notation T_{obs}.

Non-observable signals are signals that are not observed by the environment. They are internal signals

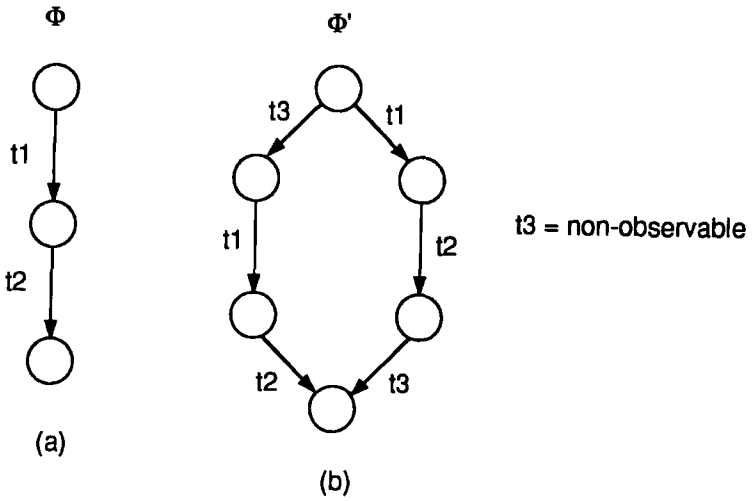

Fig. 1. Two equivalent state graphs.

of the circuit which are only there to satisfy certain requirements like the CSC requirement. So transitions of non-observable signals are to be ignored when checking the equivalency of two state graphs. This will be taken into account in the definition of equivalency.

First the restriction of a firing sequence is defined in [12] to be able to filter out non-observable transitions. Next the language that is accepted by a state graph is defined [13]. Finally equivalency of state graphs is defined [13].

Definition 2.3. (The restriction of a firing sequence) $\sigma \in T^*$, $T_1 \subset T$:

$$(\sigma t) \lceil T_1 = \sigma \lceil T_1 t \text{ if } t \notin T_1$$
$$= (\sigma \lceil T_1) t \text{ if } t \in T_1$$

$\sigma \lceil T_1$ is the restriction of σ onto the set T_1.

For example let $T_1 = \{t_1, t_2, t_3\}$. Then $(t_3 t_2 t_4 t_1 t_4) \lceil T_1 = (t_3 t_2 t_1)$.

Definition 2.4. (The language accepted by Φ)

$$\mathcal{L}(\Phi) = \{\sigma \in T^* : m_0[\sigma\rangle\}$$
$$\mathcal{L}(\Phi) \lceil T_1 = \{\sigma \lceil T_1 \in T^* : m_0[\sigma\rangle\}$$

The language accepted by Φ is the set of all firing sequences (also called strings [13] or traces [12]) that starting from the initial state lead to a valid state. The restriction operator is also defined for a set of firing sequences in the same manner.

The following definition states that two state graphs are equivalent iff they accept the same language when it is restricted to the observable transitions.

Definition 2.5. (Equivalent state graphs) Φ and Φ' are said to be equivalent iff

$$\mathcal{L}(\Phi) \lceil T_{obs} = \mathcal{L}(\Phi') \lceil T_{obs}$$

An example of two equivalent state graphs is shown in figure 1. For the first state graph we have: $\mathcal{L}(\Phi) = \{t_1 t_2\}$. The set of firing sequences accepted by Φ' is $\{t_1 t_2 t_3, t_3 t_1 t_2\}$. Because t_3 is a non-observable transition it has to be filtered out. $(t_1 t_2 t_3) \lceil \{t_1 t_2\} = t_1 t_2$ and $(t_3 t_1 t_2) \lceil \{t_1 t_2\} = t_1 t_2$. So $\mathcal{L}(\Phi') = \{t_1 t_2\}$. Note that the two state graphs are structurally different, so equivalence of state graphs is not a trivial property. It does not reduce to graph-isomorphism. This is known as extensional equality.

3. A Global Assignment Technique for State Graphs

3.1. The Problem Formulation

Several synthesis-methods have been proposed [5], [6], [8], [10]. The method proposed in [5] directly transforms the initial STG Σ into a new STG Σ' without investigating the state graph Φ. This has the advantage that the state graph does not have to be generated. It is known that the state graph may explode for even moderate sized STGs.

The method proposed by Lavagno et al. [8] and Kondratyev [10] derive information from the initial state graph Φ to transform Σ into Σ'. In this article we propose a method where the initial state graph Φ is transformed directly into Φ' without generating the new STG for Φ'. Obviously the new state graph Φ' should

satisfy a number of requirements. These are stated in the following problem formulation:

Problem 3.1. Given a finite connected state graph $\Phi = \langle M_S, T, \delta \rangle$ with a consistent state assignment (possibly derived from an STG Σ), derive a new $\Phi' = \langle M_S', T', \delta' \rangle$ that satisfies the following requirements:

- Φ' is equivalent to Φ.
- Φ' has a consistent state assignment.
- Φ' satisfies the CSC requirement.
- Every transition that is semi-modular in Φ should be semi-modular in Φ' and every newly added transition should be semi-modular.

We assume that Φ only contains observable signals and that the signals added to generate Φ' are all non-observable. Every time M_S', T' or δ' is used in this article we refer to the new state graph that is to be generated.

Note that the only requirement for the initial state graph is to be finite and connected and to have a consistent state assignment. So the STG from which the state graph is derived needs not be a live net, nor a safe net nor a free-choice net.

The procedure to derive a transformed state graph Φ' is the following. First the initial state graph Φ will be derived from the STG Σ. Then an assignment will be done on the state graph Φ. This technique will assign for each state signal s_k to be added and for each state in the original state graph one item out of the set {0, 1, up, down} to $m(k)$. Based on that assignment the new state graph Φ' can be derived from Φ. For that purpose a translation procedure will be defined in Section 3.3. With the following example we want to give a flavor of the method that is explained in a more theoretical way in the next sections. The initial STG is shown in figure 2(a) and the state graph derived from the initial STG is shown in figure 2(b). The states m_3 and m_5 violate the CSC-property. They are both assigned the same code, but different transitions of output signals are enabled in these states. So a new state graph should be derived that satisfies the CSC-requirement and the other requirements stated in Problem 3.1. So an assignment has been done on the state graph. Based on that assignment a new state graph shown in figure 2(c) is derived. For the exact translation procedure we refer to Section 3.3. The assignment of down to state m_1 intuitively means that there will be two states in the new state graph (n_1 and n_1') related to m_1. This relation, the *cover-relation* is defined in Section 3.3. All these relations are indicated in the figure by the thin lines with arcs on both sides. Between the two states n_1 and n_1' there will be a down-transition of the state signal that will be added to the graph. An assignment of a 0 to

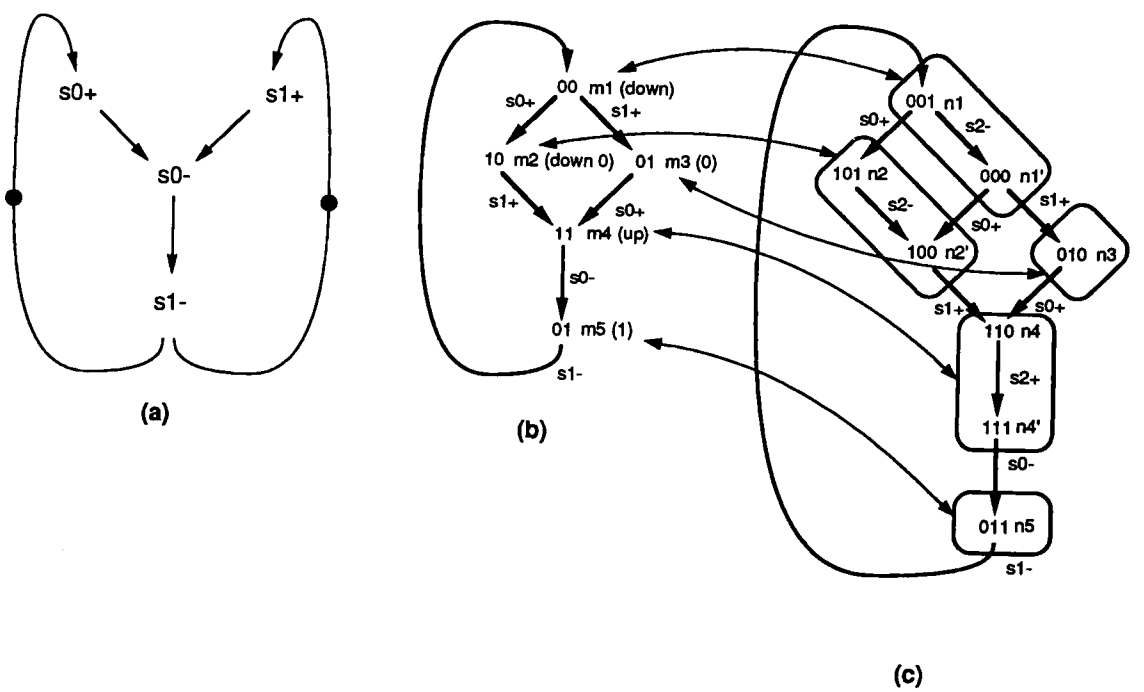

Fig. 2. The initial STG (a) the state graph with assignment (b) and the new state graph (c).

the state m_3 means that all states in the new state graph that are related to m_3 (n_3) will be encoded 0 by the state signal.

Because m_3 and m_5 violate the CSC-property the assignment of a 0 to m_3 and of a 1 to m_5 makes sure that this problem is circumvented for the new state graph. For the necessary and sufficient relations for the assignment to produce a new state graph that satisfies CSC, we refer to Section 4.

It is obvious that not any assignment on the state graph will produce a new state graph that satisfies all the requirements stated in Problem 3.1. They will be presented in Section 3.4.

3.2. The Restrictions

The method proposed in this article has two restrictions. The first restriction is due to our definition of equivalency of state graphs. When deriving Φ' from Φ no concurrency may be reduced. Transitions that are concurrent in Φ should also be concurrent in Φ', else Φ and Φ' are not equivalent. Reducing concurrency intuitively means that certain states, that are present in the original state graph, will not have any state in the new state graph related to them. Currently we cannot express such a situation in our framework. It is the subject of current research.

The second restriction is caused by the method itself. If two transitions of state signals (newly added signals) fire directly after each other in the state graph, they have to be concurrent. The reason why will be explained at the end of Section 3.3.

Restriction 3.1.

$$\forall t, t' \in T : t \| t' \text{ in } \Phi \Rightarrow t \| t' \text{ in } \Phi' \quad (1)$$

$$m_0'[\sigma t t' \sigma'\rangle \text{ with } t, t' \in T_{Nobs} \Rightarrow t \| t' \quad (2)$$

Note that the methods presented in [8] and [10] also have these restrictions (among other restrictions).

3.3. The Procedure to Derive Φ' from Φ

In this section we formally define the procedure to translate the original state graph with assignment into a new state graph Φ'.

The number of signals that are present in the original state graph Φ will be denoted by n. The number of signals that are present in the Φ' will be denoted by q. q equals n augmented with the number of state signals added to Φ to generate Φ'. Every signal will be represented by s_k. If $0 \leq k < n$ it is a signal present in the original state graph. If $n \leq k < q$ it is a state signal. In our formulation we assume that the number of state signals to be added is fixed in advance. In Section 7 it is explained how this formulation then can be transformed into an optimization problem (minimizing the number of state signals).

$m(k)$ corresponds to the code assigned by signal s_k to state m in Φ with $0 \leq k < n$ (see Definition 2.1). $m(k)$ with $n \leq k < q$ corresponds to the assignment of an item out of the set {0, 1, up, down} to state m for state signal s_k.

State m_1 in figure 3 is assigned ⟨00downdown⟩. 00 corresponds to the assignment of signals present in the initial state graph (s_0, s_1). downdown is the assignment for the state signals s_2 and s_3. The state graph will be expanded for each state signal.

First we will present the procedure *Expand*, shown in figure 4, that constructs Φ' from Φ with an assignment, if there is only one state signal present. The first part of *Expand* defines the splitting of states when the state is assigned an up or a down. The second part defines how all the new states are related when firing transitions present in the old state graph.

The procedure Derive-new-state-graph, which makes use of procedure *Expand*, constructs Φ' when there is more than one state signal. This is shown in figure 5.

In the first part of the procedure *Expand* three relations will be defined that relate each state in the new state graph Φ' to each state in the old state graph Φ. These three relations are $\succ$ (the cover relation), *after* and *before*. The relations *before* and *after* are only used in the second part of *Expand*. The cover relation is used throughout the article. In procedure Derive-new-state-graph the cover-relation is defined when there is more than one state signal.

Definition 3.1. (Cover-relation)

$$\forall m \in \Phi, \forall n \in \Phi' : n \succ m \text{ iff } \{\sigma \in T^* | n[\sigma\rangle\} \lceil T_{obs} = \{\sigma \in T^* | m[\sigma\rangle\}$$

The *before* and *after* relations are only valid when only one new state signal is added to the state graph. This is in fact the cause of the second restriction as will be explained at the end of this section.

Definition 3.2. (Before-relation)

$$\forall m \in \Phi, \forall n \in \Phi' : n = \text{before}(m) \text{ iff } n \succ m \wedge \neg([s_k^*\rangle n)$$

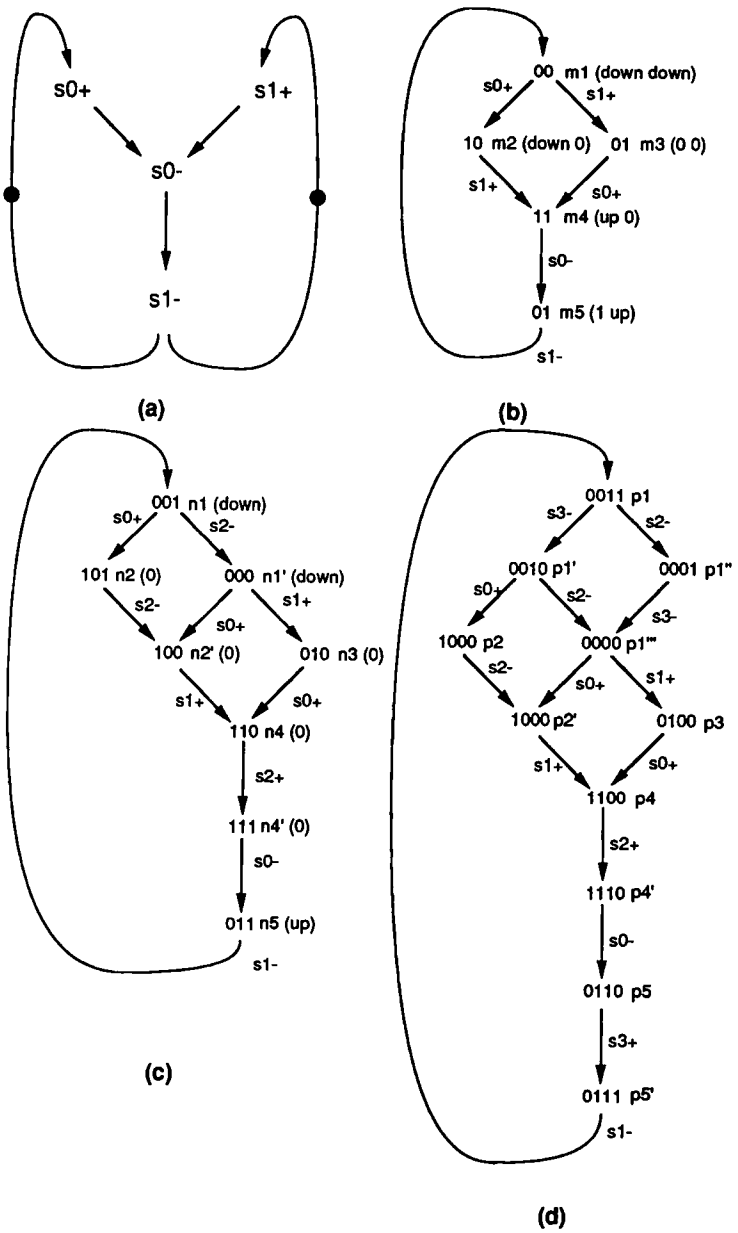

Fig. 3. The STG (a) the state graph with assignment and the expansion steps (c) and (d).

Definition 3.3. (After-relation)

$\forall m \in \Phi, \forall n \in \Phi' : n = \text{after}(m)$ iff $n \succ m \wedge \neg(n[s_k^*\rangle)$

An example is presented in figure 3. The initial STG is presented together with the initial state graph derived from the STG. On this state graph an assignment has been done for two state signals (s_2 and s_3). In figure 3(c) the state graph is shown after applying the procedure *Expand* for the signal s_2. This is what is defined in the first part of *Expand*.

$m_1(2) = \text{down} \Rightarrow n_1 \succ m_1$ and $n_1' \succ m_1$ and $n_1' = \delta'(n_1, s_2^-)$.

$m_2(2) = \text{down} \Rightarrow n_2 \succ m_2$ and $n_2' \succ m_2$ and $n_2' = \delta'(n_2, s_2^-)$.

```
Expand (Φ, s_i) {
  FOREACH (m ∈ M_S) {
    if(m(i) = up) {
      add new states m', m'' to Φ';
      m' = before(m); m'' = after(m);
      m' ≻ m;  m'' ≻ m;
      m'' = δ'(m', s_i^+);  m'(k) = 0;  m''(k) = 1;}
    if (m(i) = down) {
      add new states m', m'' to Φ';
      m' = before(m); m'' = after(m);
      m' ≻ m;  m'' ≻ m;
      m'' = δ'(m', s_i^-);  m'(k) = 1;  m''(k) = 0;}
    if (m(i) = 0 ∨ m(i) = 1) {
      add new state m' to Φ';
      m' = before(m) = after(m);
      m'(k) = m(i);  m' ≻ m;} }
  FOREACH (m_1 ∈ M_S) {
    FOREACH (m_2 = δ(m_1, t)) {
      if(m_1(i) = up ∧ m_2(i) = up∨
        m_1(i) = down ∧ m_2(i) = down) {
        before(m_2) = δ'(before(m_1), t);
        after(m_2) = δ'(after(m_1), t); }
      else
        before(m_2) = δ'(after(m_1), t);} } }
  FOREACH (m ∈ M_S)
    FOREACH (m' ≻ m)
      for (i = 0; i < n; i++)
        m'(i) = m(i);
  return (Φ'); }
```

Fig. 4. The *Expand* algorithm.

```
Derive-new-state-graph (Φ, q){
  oldΦ = Φ;
  for (i = n; i < q; i++) {
    Φ' = Expand(oldΦ, s_i);
    FOREACH (m in oldΦ){
      FOREACH (m' in Φ' : m' ≻ m in oldΦ){
        for (j = i+1; i < q; j++)
          m'(j) = m(j);
        FOREACH (m'' in Φ : m ≻ m'' in Φ)
          m' ≻ m'' in Φ;}}
    oldΦ = Φ';}
  return (Φ');}
```

Fig. 5. The algorithm for deriving the new state graph.

This is what is defined in the second part of *Expand*.

$m_1(2) = \text{down} \wedge m_2(2) = \text{down} \Rightarrow n_2 = \delta'(n_1, s_0^+)$ and $n_2' = \delta'(n_1', s_0^+)$.

$m_2(2) = \text{down} \wedge m_4(2) = \text{up} \Rightarrow n_4 = \delta'(n_2', s_1^+)$.

During this first expansion of Φ the assignment for the signal s_3 has been transferred to Φ'. Every state in Φ' has the same assignment as the state it covers in Φ. Now the second expansion for the signal s_3 can be done resulting in figure 3(d). This is the final state graph. Derive-new-state-graph also states for instance that $p_1 \succ m_1$, $p_1' \succ m_1$, $p_1'' \succ m_1$ and $p_1''' \succ m_1$.

Now it also becomes clear why there is the second restriction on the method (Section 3.2). Assume that s_0 and s_1 are state signals. We want to express the following: $m_1[s_0^+\rangle m_2[s_1^+\rangle$ in the new state graph. m_1 and m_2 should cover the same state in the old state graph (there is no transition of a signal already present in the old state graph in between). Let this state covered by m_1 and m_2 be m. The only way to express the above situation would be $m(0) = \text{up}$ and $m(1) = \text{up}$. If we follow the above procedure this means that beside the wanted situation $m_1[s_0^+\rangle m_2[s_1^+\rangle$ we also have the situation $m_1[s_1^+\rangle m_3[s_0^+\rangle$. So s_0^+ and s_1^+ are concurrent under these circumstances.

3.4. Necessary and Sufficient Requirements for Correctness

The state graph Φ' derived from Φ should satisfy the requirements stated in Problem 3.1. The following theorems provide us with the necessary and sufficient requirements the assignment on the state graph should satisfy in order that Φ' satisfies the requirements stated in Problem 3.1. In all these theorems we assume that some assignment is performed on the graph and that Φ' is derived according to the procedures above. For the proofs of the theorems, we refer to [14].

The following theorem states that Φ and Φ' are equivalent.

THEOREM 3.1. $\mathcal{L}(\Phi') \lceil T_{obs} = \mathcal{L}(\Phi)$

Definition 3.4. $m_i, m_j \in M_S$, $n \leq k < q$

$$
\begin{aligned}
arc\ (m_i, m_j, k) &= 0 \text{ if } (m_i(k) = 0 \wedge m_j(k) = 1) \\
&= 0 \text{ if } (m_i(k) = 0 \wedge m_j(k) = \text{down}) \\
&= 0 \text{ if } (m_i(k) = 1 \wedge m_j(k) = 0) \\
&= 0 \text{ if } (m_i(k) = 1 \wedge m_j(k) = \text{up}) \\
&= 0 \text{ if } (m_i(k) = \text{up} \wedge m_j(k) = 0) \\
&= 0 \text{ if } (m_i(k) = \text{down} \wedge m_j(k) = 1) \\
&= 1 \text{ in all other cases} \qquad (3)
\end{aligned}
$$

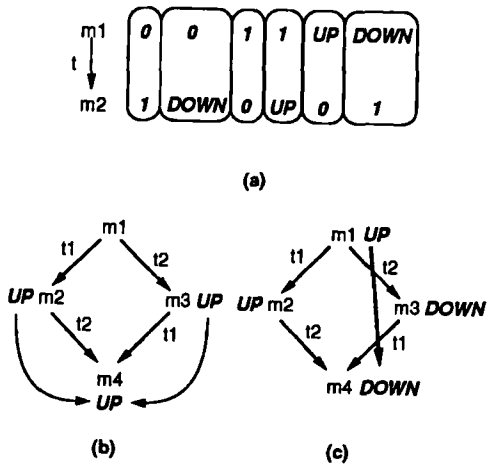

Fig. 6. The rules for a valid assignment.

This definition is introduced because it indicates what assignments are not allowed on two states when there is an arc between these two states. The ones that are not allowed are shown in figure 6(a). Intuitively $m_1(k) = 0$ and $m_2(k) = 1$ is not allowed because there has to be an up-transition of the signal s_k in one of the states covering m_1 or m_2. The assignment of 0 to m_1 and down to m_2 is impossible. Intuitively this would mean that the state signal is at level 0 in one state and should go down in a next state. This corresponds to an inconsistent state assignment. The other cases are analogous.

THEOREM 3.2. Φ' has a consistent state assignment iff Φ has a consistent state assignment and $\forall k, n \leq k < q$:

$$\forall m, m' \in M_S : m' = \delta(m, t) \Rightarrow arc(m, m', k) \quad (4)$$

The transitions of the newly added signals (the state signals) should be semi-modular. The following theorem explains that this is the case if the conditions of the previous theorem are satisfied.

THEOREM 3.3. If the conditions of the previous theorem are satisfied then every transition of every state signal is semi-modular.

The following theorem provides us with the necessary and sufficient requirements such that all transitions that are semi-modular in the old state graph remain semi-modular in the new state graph. These rules are shown in figure 6(b) and (c). If m_2 and m_3 are assigned up then m_4 should be assigned up. If m_2 and m_3 are assigned up and down, then m_1 and m_4 should be assigned different items.

THEOREM 3.4. If t is semi-modular in m in Φ and Φ' has a consistent state assignment then t is semi-modular in each state covering m iff $\forall k, n \leq k < q : \forall m_1, m_2, m_3, m_4 \in M_S$:

$$((m_1[t_1\rangle m_2) \wedge (m_1[t_2\rangle m_3) \wedge (m_2[t_2\rangle m_4) \wedge (m_3[t_1\rangle m_4)):$$

$$(m_2(k) = \text{up} \wedge m_3(k) = \text{down}) \Rightarrow m_4(k) \neq m_1(k) \quad (5)$$

$$(m_2(k) = \text{up} \wedge m_3(k) = \text{up}) \Rightarrow m_4(k) = \text{up} \quad (6)$$

$$(m_2(k) = \text{down} \wedge m_3(k) = \text{down}) \Rightarrow m_4(k) = \text{down} \quad (7)$$

4. Necessary and Sufficient Relations for CSC-Satisfaction

In this section the necessary and sufficient requirements for an assignment to result in a state graph Φ' with the CSC property will be given. We first need two definitions to lighten the notational burden.

Definition 4.1. $m, m' \in M_S, n \leq k < q$

$$\begin{aligned} \text{diff}(m, m', k) &= 1 \text{ if } (m(k) = 0 \wedge m'(k) = 1) \\ &= 1 \text{ if } (m(k) = 1 \wedge m'(k) = 0) \\ &= 0 \text{ in all other cases} \end{aligned} \quad (8)$$

diff(m, m', k) expresses that the state signal s_k gives a code 0 to all the new states that cover m and a code 1 to all the new states that cover m' (or vice versa).

Definition 4.2. $m, m' \in M_S, n \leq k < q$

$$\begin{aligned} \text{cond}(m, m', k) &= 1 \text{ if } (m(k) = 0 \wedge m'(k) = \text{up}) \\ &= 1 \text{ if } (m(k) = 1 \wedge m'(k) = \text{down}) \\ &= 1 \text{ if } (m(k) = \text{up} \wedge m'(k) = \text{down}) \\ &= 1 \text{ if } (m(k) = \text{up} \wedge m'(k) = 0) \\ &= 1 \text{ if } (m(k) = \text{down} \wedge m'(k) = 1) \\ &= 1 \text{ if } (m(k) = \text{down} \wedge m'(k) = \text{up}) \\ &= 0 \text{ in all other cases} \end{aligned} \quad (9)$$

cond(m, m', k) expresses that there exists a state m'' that covers m and a state m''' that covers m' that satisfy the following condition: if m'' and m''' are assigned the same code, then the CSC condition will be violated for the state signal s_k (it will have a transition enabled in one state and not in the other state).

Definition 4.3. (USC-violation-relation)

$$(m, m') \in U : M_S \times M_S \text{ iff}$$
$$\forall k, 0 \leq k < n : m(k) = m'(k)$$

Definition 4.4. (CSC-violation-relation)

$(m, m') \in E : M_S \times M_S$ iff

$$(\forall k, 0 \leq k < n : m(k) = m'(k)) \wedge$$
$$(\exists s_k \in S_{NI} : m[s_k^*\rangle \wedge \neg(m'[s_k^*\rangle))$$

THEOREM 4.1. (CSC-satisfaction) Φ' will satisfy the CSC condition iff

$$\forall (m, m') \in E : \exists k, n \leq k < q : \text{diff}(m, m', k) \text{ and} \quad (10)$$

$$\forall (m, m') \in U : \exists k, n \leq k < q : \text{cond}(m, m', k) \Rightarrow$$
$$\exists l \neq k, n \leq l < q : \text{diff}(m, m', l) \quad (11)$$

Let us look at the example of figure 7. s_0 is an output signal and s_1 is an input signal. In the state graph there are two states that should be encoded differently according to the CSC-condition, namely m_0 and m_4. m_4 enables a transition of the output signal s_0 and m_0 does not. So a state signal should be added that encodes these two states differently. This is expressed in relation 10. States m_0 and m_2 should not be encoded differently

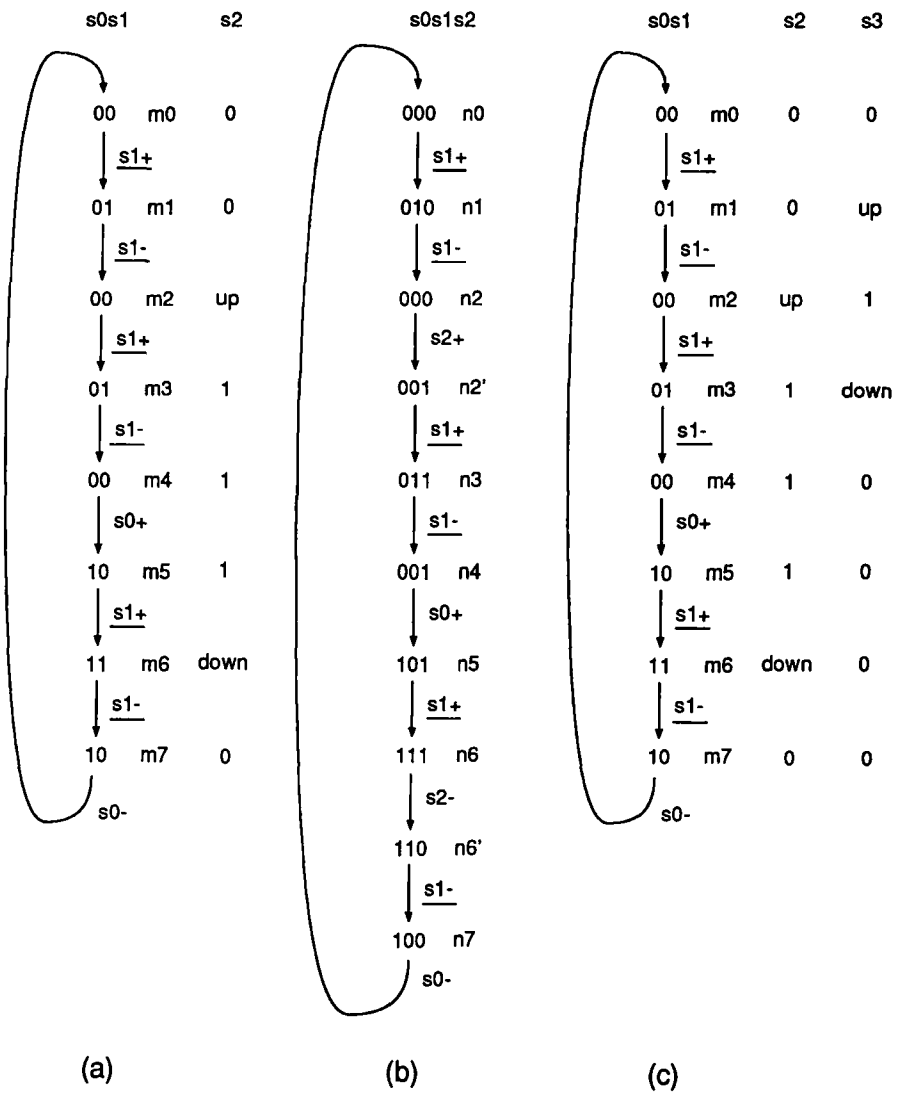

Fig. 7. The original state graph with assignment (a), the new state graph (b) and an assignment that satisfies the CSC-requirement.

according to the CSC-condition. The assignment on the state graph will make sure that the state signal s_2 will encode all the states covering m_4 and m_0 differently (namely n_0 and n_4 in figure 7(b)). But in figure 7(b), a new CSC-violation has been introduced between the states n_0 and n_2. This violation can also be checked on the original state graph with the assignment (figure 7(a)). A 0 and an up were assigned to two states (m_0 and m_2) that were assigned the same code by signals in the original STG. If this is the case another state signal should be added to solve this conflict (not shown in the figure). This is expressed in Equation 11. Note that this problem is formulated in a global way and that the synthesis process will not try to resolve conflicts in a step by step manner. This was only done in this paragraph for demonstrative purposes.

The same example, but now fully worked out, is shown in figure 7(c). The reader can check that now the CSC-requirement is satisfied. It is of course possible to add state signals with multiple transitions. An example is presented in [14].

5. Examples

The purpose of this section is to show how powerful this state assignment technique can be. The examples demonstrate that transformations can be performed that are conceptually difficult to do at STG-level. Free choice nets [7] can be transformed into non-free choice nets. 1-safe marked graphs [7] can be transformed into 2-safe marked graphs. Transitions may be added that do not follow the Petri net firing rule [11]. Still the transformations are correct, according to the requirements stated in Section 3.1. An example is shown in figure 8. The state graph with a valid assignment is shown in figure 8(b). The new STG that corresponds to this assignment is shown in figure 8(c). The transition s^+ can now be fired at the same time when the choice between t_3 and t_6 is made. So this transformation technique is not limited to putting a new transition between two transitions that belong to the same conditional branch. The resulting net is not a free choice net any more.

An initial 1-safe MG is shown in figure 9(a). The state graph with assignment is shown in figure 9(b). The STG that corresponds to this assignment is shown in figure 9(c). The arc pointing from s^+ to t_2 may contain two tokens (fire t_3, s^-, t_5, s^+). But still the transformation is correct. For other examples the reader is referred to [14].

6. Boolean Satisfiability Framework

In this section, we present a Boolean satisfiability approach to the generalized state graph assignment

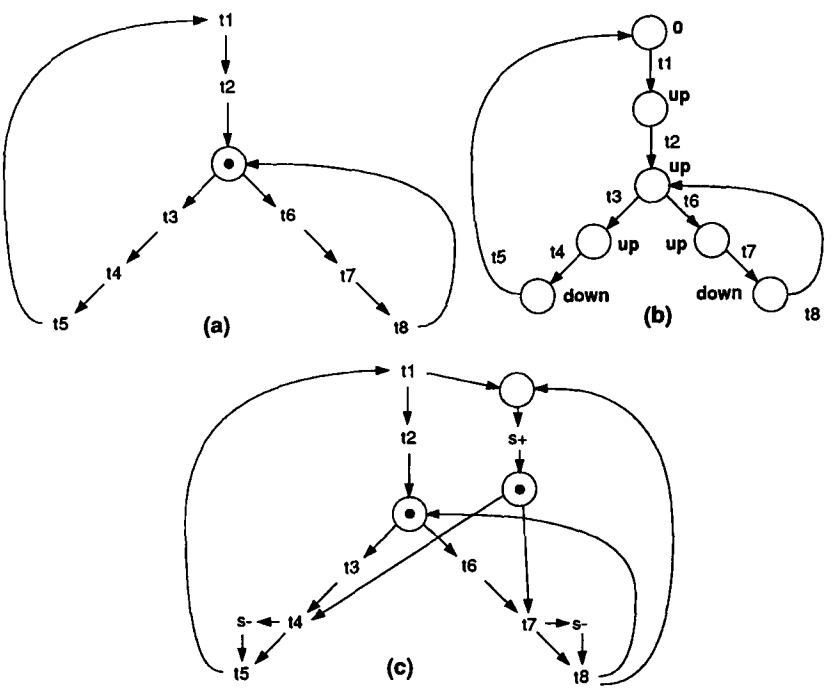

Fig. 8. Transforming a free choice net.

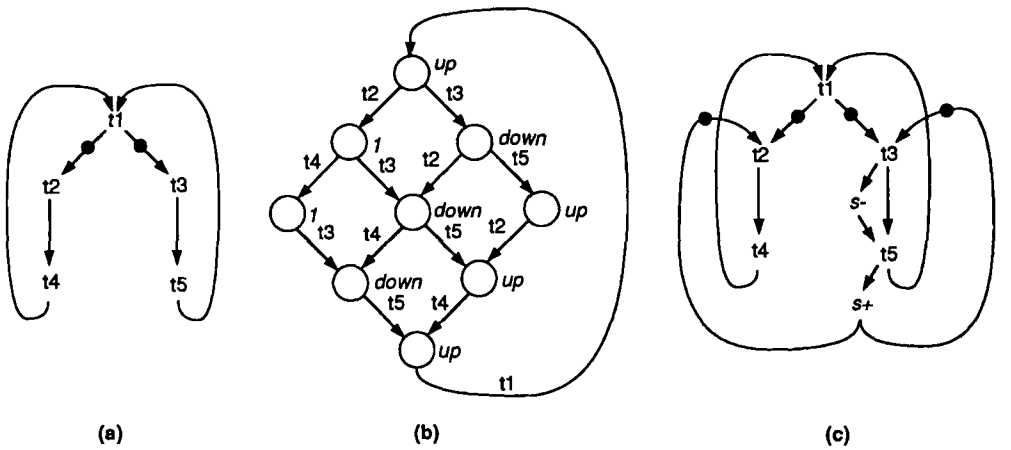

Fig. 9. Transforming a 1-safe MG into a 2-safe MG.

problem. Specifically, all the necessary and sufficient conditions expressed in Sections 3 and 4 can be transformed into Boolean constraints. The problem of checking whether CSC can be satisfied with l state signals now reduces to a Boolean satisfiability problem. The state graph assignment is obtained as a by-product from the satisfying Boolean assignment.

The mapping to the Boolean satisfiability problem is done as follows. For each state $m \in M_S$, l *four-value* variables are introduced. Each four-value variable can assume one of four possible assignments, namely {0, 1, up, down}. Each four-value variable is then encoded using two Boolean variables as follows:

four-value	b_1	b_2
0	0	0
1	0	1
up	1	0
down	1	1

Thus, in this formulation there are $2 \times l \times \#M_S$ Boolean variables, with l the number of state signals and $\#M_S$ the number of states in the original state graph Φ. The overall problem can be succinctly represented using only three sets of constraints, namely

1. the consistent state assignment constraints,
2. the semi-modularity preservation constraints, and
3. CSC-satisfaction constraints.

Each is explained below. To lighten the equations, we will use $f \Rightarrow g$ to mean $\bar{f} + g$. It reads f implies g.

6.1. Formulating the Constraints

Consistent state assignment constraints: Let $\Delta = \{(i, j) | m_j = \delta(m_i, t)\}$. Δ is simply the set of adjacent state pairs in the state graph Φ. Let $b_{nk,1}$ and $b_{nk,2}$ be the two Boolean variables encoding the k-th state signal of the n-th state in M_S. Then the constraints to guarantee consistent state assignment, corresponding to Theorem 3.2, can be expressed as follows:

$$\prod_{(i,j)\in\Delta} \prod_{k=1}^{l} (b_{ik,1} + b_{ik,2} + b_{jk,1} + \bar{b}_{jk,2}).$$

$$\prod_{(i,j)\in\Delta} \prod_{k=1}^{l} (b_{ik,1} + b_{ik,2} + \bar{b}_{jk,1} + \bar{b}_{jk,2}).$$

$$\prod_{(i,j)\in\Delta} \prod_{k=1}^{l} (b_{ik,1} + \bar{b}_{ik,2} + b_{jk,1} + b_{jk,2}).$$

$$\prod_{(i,j)\in\Delta} \prod_{k=1}^{l} (b_{ik,1} + \bar{b}_{ik,2} + \bar{b}_{jk,1} + b_{jk,2}).$$

$$\prod_{(i,j)\in\Delta} \prod_{k=1}^{l} (\bar{b}_{ik,1} + b_{ik,2} + b_{jk,1} + b_{jk,2}).$$

$$\prod_{(i,j)\in\Delta} \prod_{k=1}^{l} (\bar{b}_{ik,1} + \bar{b}_{ik,2} + b_{jk,1} + \bar{b}_{jk,2}).$$

This can be more efficiently rewritten as follows:

$$\prod_{(i,j)\in\Delta} \prod_{k=1}^{l} (b_{ik,1} + b_{ik,2} + \bar{b}_{jk,2})(b_{ik,1} + \bar{b}_{ik,2} + \bar{b}_{jk,2}).$$

$$\prod_{(i,j)\in\Delta}\prod_{k=1}^{l}(\bar{b}_{ik,1}+b_{ik,2}+b_{jk,1}+b_{jk,2}).$$

$$\prod_{(i,j)\in\Delta}\prod_{k=1}^{l}(\bar{b}_{ik,1}+\bar{b}_{ik,2}+b_{jk,1}+\bar{b}_{jk,2}).$$

Semi-modularity preservation constraints: Let (p, q, r, s) be the set of states with the following relationship: $((m_p[t_p\rangle m_q) \wedge (m_p[t_q\rangle m_r) \wedge (m_q[t_q\rangle m_s) \wedge (m_r[t_p\rangle m_s))$. Let I be the set of states satisfying this relationship. Then the following constraints must be satisfied to ensure the preservation of semi-modularity (note that only signal transitions that were semi-modular in the original Φ need to be preserved, but semi-modularity per se is not required).

$$\prod_{(p,q,r,s)\in I}\prod_{k=1}^{l}(b_{qk,1}\bar{b}_{qk,2}b_{rk,1}b_{rk,2} \Rightarrow (b_{sk,1}\oplus b_{pk,1}) + (b_{sk,2}\oplus b_{pk,2})).$$

$$\prod_{(p,q,r,s)\in I}\prod_{k=1}^{l}(b_{qk,1}\bar{b}_{qk,2}b_{rk,1}\bar{b}_{rk,2} \Rightarrow b_{sk,1}\bar{b}_{sk,2}).$$

$$\prod_{(p,q,r,s)\in I}\prod_{k=1}^{l}(b_{qk,1}b_{qk,2}b_{rk,1}b_{rk,2} \Rightarrow b_{sk,1}b_{sk,2}).$$

This essentially corresponds to Theorem 3.4.

CSC-satisfaction constraints: Let $\hat{U} = \{(i, j)|(m_i, m_j) \in U\}$ where U is the USC-violation relation as defined in Definition 4.3. Let $\hat{E} = \{(i, j)|(m_i, m_j) \in E\}$ where E is the CSC-violation relation as defined in Definition 4.4.

The first set of constraints is intended to resolve state pairs that originally violated the CSC property. The constraints can be written as follows.

$$\prod_{(i,j)\in\hat{E}}\sum_{k=1}^{l}(\bar{b}_{ik,1}\bar{b}_{ik,2}\bar{b}_{jk,1}b_{jk,2}+\bar{b}_{ik,1}b_{ik,2}\bar{b}_{jk,1}\bar{b}_{jk,2}).$$

The second set of constraints is intended to resolve state pairs that originally violated the USC property.

$$\prod_{(i,j)\in\hat{U}}\prod_{k=1}^{l}$$

$$((\bar{b}_{ik,1}\bar{b}_{ik,2}b_{jk,1}b_{jk,2}+\bar{b}_{ik,1}b_{ik,2}b_{jk,1}b_{jk,2}+b_{ik,1}\bar{b}_{ik,2}b_{jk,2}b_{jk,1}+$$

$$b_{ik,1}\bar{b}_{ik,2}\bar{b}_{jk,1}\bar{b}_{jk,2}+b_{ik,1}b_{ik,2}\bar{b}_{jk,1}b_{jk,2}+b_{ik,1}b_{ik,2}b_{jk,2}\bar{b}_{jk,2})$$

$$\Rightarrow\sum_{p=1,p\neq k}^{l}(\bar{b}_{ip,1}\bar{b}_{ip,2}\bar{b}_{jp,1}b_{jp,2}+\bar{b}_{ip,1}b_{ip,2}\bar{b}_{jp,1}\bar{b}_{jp,2}))$$

These constraints essentially correspond to Theorem 4.1.

6.2. *Solving the Boolean Satisfiability Problem*

Given the constraints described above, we have to find a Boolean assignment to the variables $b_{ik,j}$ that satisfies all the constraints. This problem is known as the Boolean satisfiability problem. In [15], a branch-and-bound algorithm was proposed to solve the Boolean satisfiability problem in an exact way. Though the algorithm was originally intended for testing, it can also be used in this context. A variant of this approach was proposed in [16]. We have used this implementation and have been able to come up with a solution with over 200 variables. This is sufficient to find an exact solution in many practical cases. For larger examples, heuristics are currently being developed to come up with optimized solutions. It is known that the satisfiability problem is NP-hard, but at this moment it has not been proven that our problem is also NP-hard. Once a satisfying assignment is found, the Boolean assignments can be mapped back to the four-value set {0, 1, up, down}. Then the procedure *Derive-new-state-graph* described in figure 5 can be employed to derive the new state graph.

7. Experimental Results

Table 1 contains the experimental results. The first six examples are from the HP-benchmark [8] the following five examples are in-house examples. Not all these examples are self-timed. Some of them contain timing constraints. The columns labeled n, T and M_S contain the number of initial signals, transitions and states. The

Table 1.

Name	n	T	M_S	$q - n$	CPU
alloc-outbound	7	18	17	2	0.2
mp-forward-pkt	7	14	20	1	0.1
nak-pa	9	18	56	1	0.6
ram-read-sbuf	10	20	36	1	0.2
sbuf-ram-write	10	20	58	2	1.6
sbuf-read-ctl	6	12	15	1	0.1
adfast	6	12	44	2	0.9
I2C	2	68	90	2	0.5
count	6	16	29	1	0.2
postoffice	8	39	62	1	2.2

column labeled state $q - n$ contains the minimum number of state signals needed to find a solution that satisfies CSC. The CPU-times for finding a satisfying assignment are indicated in the last column (in seconds on a DEC5000).

8. Conclusion

In this article, we proposed a global assignment theory for encoding state graph transformations. We presented a constraint satisfaction framework that can guarantee necessary and sufficient conditions for a state graph assignment to result in a transformed state graph that satisfies the complete state coding requirement. Necessary means that within certain restrictions (Section 3.2) any transformation can be performed in this framework. The initial STG need not be a live net, nor a safe net, nor a free choice net. The only requirement is that the state graph is finite, connected, and has a consistent state assignment (Section 3.1). Transformations that convert a free choice net into a non-free choice net and a 1-safe net into a 2-safe net are feasible. It is even possible to add transitions that do not follow the Petri net firing rule [11]. Even though our method can search a large solution space, we have shown that it is possible to solve the problem in an exact way in acceptable CPU times in many practical cases.

The theory can be extended to avoid the current restrictions. The assignment set can be extended for that purpose. Finally it is known that even for moderate sized STGs the state graph may explode. When the state graphs become too large heuristics are needed to find optimized solutions.

The technique proposed in this article minimizes the number of state signals to be added. Currently techniques are being developed to do an assignment in such a way that the logic derived from the state graph is minimized.

The constraint framework is also being extended in such a way that certain timing constraints can be taken into account during synthesis.

Finally the specification methodology is being extended such that a broader class of circuits can be synthesized.

Acknowledgment

The authors wish to thank Chantal Ykman for the many stimulating discussions. We also wish to thank her for implementing the Boolean satisfiability framework in a very efficient way. This explains the big improvement over the results reported in [17].

References

1. T.A. Chu, "Synthesis of self-timed VLSI circuits from graph-theoretic specifications," Ph.D. thesis, MIT, June 1987.
2. Teresa H. Meng, *Synchronization Design for Digital Systems*. Boston, MA: Kluwer Academic Publishers, 1991.
3. S.H. Unger, *Asynchronous Sequential Switching Circuits*. Wiley Interscience, 1969.
4. L. Lavagno, K. Keutzer, and A. Sangiovanni-Vincentelli, "Algorithms for synthesis of hazard-free asynchronous circuits," *Proceedings of the Design Automation Conference*, June 1991.
5. P. Vanbekbergen, "Optimized synthesis of asynchronous control circuits from graph-theoretic specifications," *Proceedings of the International Conference on Computer-Aided Design*, pp. 184–187, November 1990.
6. A.V. Yakovlev and A. Petrov, "Petri nets and parallel bus controller design," *International Conference on Application and Theory of Petri Nets*, Paris, France, June 1990, pp. 245–263.
7. T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, April 1989, pp. 541–580.
8. L. Lavagno, C.W. Moon, R.K. Brayton, and A. Sangiovanni-Vincentelli, "Solving the state assignment problem for signal transition graphs," *Proceedings of the Design Automation Conference*, June 1992.
9. J.H. Tracey, "Internal state assignments for asynchronous sequential machines," *IEEE Transactions on Electronic Computers*, vol. EC-15, 1966, pp. 551–560.
10. M.A. Kishinevsky, A.Y. Kondratyev, and A.R. Taubin, "Formal method for self-timed design," *Proceedings of the European Design Automation Conference*, 1991.
11. V.I. Varshavsky, M.A. Kishinevsky, A.Y. Kondratyev, L.Y. Rosenblum, and A.R. Taubin, "Models for specification and analysis of processes in asynchronous circuits," *Soviet Journal of Computer and Systems Sciences*, 1989.
12. C.A.R. Hoare, "Communicating sequential processes," *Communications of the ACM*, 1978, pp. 666–677.
13. J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison-Wesley, 1979.
14. P. Vanbekbergen, "Synthesis of asynchronous control circuits from graph-theoretic specifications," Ph.D. thesis, Catholic University of Leuven, ESAT, 1992. To appear.

15. T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Transactions on CAD*, vol. 11, 1992.
16. Paul R. Stephan, Robert K. Brayton, and Alberto L. Sangiovanni-Vincentelli, "Combinational test generation using satisfiability," Technical Report UCB/ERL M92/112, ucb, October 1992.
17. P. Vanbekbergen et al., "A generalized state assignment theory for transformations on signal transition graphs," *Proceedings of the International Conference on Computer-Aided Design*, 1992.

Peter Vanbekbergen received the degree in Electrical Engineering from the Katholieke Universiteit Leuven, Belgium, in 1987.

Since September 1987 he has been with the VLSI Systems Design Methodologies Group of the Inter-University Microelectronics Center (IMEC), Heverlee, Belgium, where he is working on synthesis techniques for asynchronous controllers and interface circuits.

Bill Lin received the B.Sc. degree, the M.S. degree, and the Ph.D. degree in Electrical Engineering and Computer Sciences from the University of California, Berkeley, in 1985, 1988, and 1991, respectively. Currently, he is leading a group in the VLSI Systems Design Methodologies division of IMEC (Leuven, Belgium) working on various aspects of system-level design technology. His current research interests include hardware/software co-design, system-level integration of mixed signal processing and control-intensive systems, design and synthesis of control-intensive applications and interface modules, and logic synthesis. He has previously worked at the Hewlett Packard Corp., the Hughes Aircraft Co., and the Western Digital Corp. In 1987, he received the Best Paper Award at the 24th Design Automation Conference in Miami. In 1989 and 1990, respectively, he received a best paper nomination at the IFIP VLSI conference in Munich and a distinguished paper citation at the ICCAD conference in Santa Clara.

Gert Goossens received the degree in Electrical Engineering and the Ph.D. degree in Applied Sciences from the Katholieke Universiteit Leuven, Belgium, in 1984 and 1989, respectively.

Since 1984 he has been with the VLSI Design Methodologies Division of the Interuniversity Micro-Electronics Centre (IMEC), Leuven, Belgium. Initially he worked as a research assistant, involved in the development of the CATHEDRAL CAD environment for architectural synthesis of ICs for real-time signal processing. From 1989 to 1992 he was heading the medium throughout and interface synthesis group. Since 1992, he has been heading a research group in embedded microcode systems design.

Dr. Goossens's research interests include architectural synthesis, communication synthesis and system design, for digital signal processing systems. He received a best paper award at the 26th ACM/IEEE Design Automation Conference in 1989.

Hugo J. De Man received the electrical engineering degree and the Ph.D. degree in Applied Sciences from the Katholieke Universiteit Leuven, Heverlee, Belgium, in 1964 and 1968, respectively.

In 1968 he became a member of the staff of the Laboratory for Physics and Electronics of Semiconductors at the University of Leuven, working on device physics and integrated circuit technology. From 1969 to 1971 he was at the Electronic Research Laboratory, University of California, Berkeley, as an ESRO-NASA Postdoctoral Research Fellow, working on Computer-Aided Device and Circuit Design. In 1971 he returned to the University of Leuven as a Research Associate of the NFWO (Belgian National Science Foundation). In 1974 he became a Professor at the University of Leuven. During the winter quarter of 1974–1975 he was a Visiting Associate Professor at the University of California, Berkeley. He was an Associate Editor for the IEEE Journal of Solid-State Circuits from 1975–1980 and was European Associate Editor for the IEEE Transactions on CAD from 1982 to 1985. He received a Best Paper Award at the ISSCC of 1973 on Bipolar Device Simulation and at the 1981 ESSCIRC conference for work on an integrated CAD system. In 1986 he became fellow of the IEEE. His actual field of research is the design of Integrated circuits and Computer-Aided Design.

Since 1984 he has been Vice-President of the VLSI systems design group of IMEC (Leuven, Belgium).