

A Technique to Automate STG Analysis and Refinement for CSC and Normalcy.

Abstract. Signal Transition Graph (STG) is a well-known form of specifying the system behavior along with the behavior of its environment. Normalcy has recently gained the circuits designers' attention as an STG property, which guarantees that the behavior given by STG is implementable by a circuit composed of negative or positive gates without input inverters. This contributes to the circuit robustness while the use of negative gates in CMOS technology is prospective for power consumption, area and/or latency minimization. The paper presents an automated technique for analyzing and refining the original STG for CSC and normalcy. The technique operates on the STG level facilitating the designer to effectively follow the procedures and deeply customize the refinement on the level of STG behavior specification.

1 Introduction

Signal Transition Graph (STG) is widely used for specifying the circuits behavior. It allows compact representation of causal relations, concurrency and choice. There exist a number of tools and frameworks for the synthesis of speed-independent (SI) circuits from STG specification including Petrify [3].

Complete State Coding (CSC) is an STG property crucial for for implementability of the STG behavior by SI circuit composed of logic gates. Number of works including [4] are devoted to refine STG to satisfy this property. Normalcy is another STG property presented in [8, 9] capable of guaranteeing on the STG level that certain signal (or all) can be implemented by a monotonic gate (AND, NAND, ANDOR e.t.c without input inverters) hence relaxing the input inverter zero delay assumption. Therefore a circuit mapped to a library containing only monotonic gates (this is usually the case) is more robust. Maximising the use of negative gates in turn allows to gain area and power consumption and/or speed due to the inherent simplicity of negative gates in CMOS technology relative to their positive counterparts. The results obtained by manual refinement for normalcy on the set of asynchronous benchmarks in [9] exhibit the gain of 23% in the power consumption and 28% in the area with 6% of loss in latency. These results are obtained by sequential signal transition insertion. Another strategy allows for another tradeoff.

We restrict the specification to satisfy among others the USF property (defined later) to make possible structural STG based analysis and refinement, the use of polynomial complexity synthesis algorithm for deriving SI circuit equations (presented in [6]) and provide a clear feedback to the designer during the entire analysis and refinement procedure. The latter means that all the properties' violations as well as available refinement solutions and any information available on the STG can be clearly presented to a designer on the initial canonical STG, while with other techniques operating internally on the objects derived from STG like state graph, unfoldings [7] it is hard to provide this kind of feedback and let the designer deeply customize the refinement.

2 Behavior specification model.

In this section we define the *canonical* STG, give some definitions used through the paper (2.1) and overview the generality of the allowed STG subclass (2.2).

2.1 Specification model overview.

Definition 21 *Petri net (PN) \mathcal{N} is $\mathcal{N} = (P, T, F, m_0)$ where the P, T ($P \cup T = V$) are disjoint sets of vertices (v_i) called places (p_i) and transitions (t_i), F - set of arcs (f_i) ($F \subseteq P \times T \cup T \times P$) and m_0 is an initial marking ($m_0 \subseteq P$).*

Definition 22 *Signal Transition Graph is $\mathcal{G} = (\mathcal{N}, Y, \Delta)$ where \mathcal{N} is a PN, $Y = X \cup Z$ a set of signals (environment and circuit respectively) and $\Delta : T' \rightarrow (Y) \times \{+, -\}$ is a mapping of the PN transitions on signal transitions. $T \setminus T'$ is a set of unmapped (dummy) transitions that can be used for synchronization.*

Further on by saying net we assume a Petri net underlying STG and deal with properties that do not take into account the transitions' interpretation.

Let E be a set of feasible STG events (transition firings) for m_0 , and Trc be a feasible sequence of STG events (called *trace*). The complete set of feasible traces of the STG is a *language accepted by \mathcal{G}* (denoted as $L(\mathcal{G})$). Suppose the projection of Trc on the subset of STG feasible events $E' \subseteq E$ ($Trc \downarrow E'$) produces another sequence (Trc') obtained from Trc by removing from it all events belonging to the set $E \setminus E'$. If the language $L(\mathcal{G})$ is accepted by \mathcal{G} then its projection $L(\mathcal{G}) \downarrow E'$ produces another set of traces $\{Trc \downarrow E' : Trc \in L(\mathcal{G})\}$. Let $\mathcal{G}, \mathcal{G}'$ be two STGs with the corresponding sets of events E, E' ($E' \subset E$). Then STG \mathcal{G} and \mathcal{G}' are *trace equivalent* if $L(\mathcal{G}) \downarrow E' = L(\mathcal{G}')$.

Two transitions t_i, t_j are *concurrent* $t_i \parallel t_j$ if they can occur concurrently from some marking m reachable from m_0 ($m \in [m_0)$). Places p_i, p_j are *concurrent* $p_i \parallel p_j$ iff $\exists m \in [m_0)$ such that $p_i, p_j \in m$ i.e. some reachable marking marks both p_i, p_j . Arcs f_i, f_j are *concurrent* $f_i \parallel f_j$ iff $\bullet f_i = \bullet f_j \in T \vee f_i \bullet = f_j \bullet \in T \vee \bullet f_i \parallel \bullet f_j$, where $\bullet f$ ($f \bullet$) denotes the source (destination) vertex for the arc f . Likewise $\bullet v$ ($v \bullet$) denotes the set of input (output) arcs for the vertex v .

Let us color in all feasible traces of \mathcal{N} the regions where some place $p_k \in P$ is marked. Then transitions t_i, t_m are said to be in *choice* relation ($t_i \succ t_m$) if in neither feasible trace t_i, t_m are encountered in the same uncolored region. Arcs f_i, f_j are in *choice* relation ($f_i \succ f_j$) iff $\bullet f_i = \bullet f_j \in P \vee f_i \bullet = f_j \bullet \in P \vee \bullet f_i \succ \bullet f_j$.

Concurrency and choice relations can be computed between whatever STG objects. Therefore to ease the notation we use along the paper abstract concurrency $CcS(a)$ and choice $CfS(a)$ relations, being the sets of whatever objects being concurrent (in choice relation) with a . Unless otherwise defined assume a relation of an object being a set of other objects to be the union of the corresponding relations of all objects of the set.

Cut C^{max} is a set of places corresponding to a marking $m_i \in [m_0)$. *Subcut C* is $C \subset C^{max}$. As follows from the definition concurrency $\forall p_i, p_j \in C : p_i \parallel p_j$.

Definition 23 (STG (partial) state.) *Let STG (partial) state be a pair $s_i = (C_j, c_k)$, where C_j is a (sub)cut and c_k - is a cube - a binary vector, consisting of the values of signals defined in the (partial) state s_i .*

Definition 24 (USF STG.) *STG \mathcal{G} is said to satisfy the Unique State Factorization (USF) property iff $\forall p_i \in P$ holds : $\forall f_j, f_k \in \bullet p_i : CcS(f_j) \equiv CcS(f_k)$ i.e. in other words $CcS(p_i)$ is unique regardless of the prehistory.*

For a net $\mathcal{N} = (P, T, F)$ the *dual* net is the net $\mathcal{N}^d = (T, P, F^{-1})$ i.e. $\forall v_i \in \mathcal{N} \exists v_i^d \in \mathcal{N}^d : v_i \in P \Rightarrow v_i^d \in T$ and vice versa: $v_i \in P \Rightarrow v^d \in T$ and the corresponding arcs are reversed.

We restrict STG model to the form we call *canonical* implying that STG is (1) consistent (transitions of the same signal alternate in every feasible trace),

(2) output persistent (non-input signals' behavior is deterministic), (3) live (for every reachable marking $m \in [m_0]$ and $t \in T$ there exists $m' \in [m]$ enabling t), (4) safe (or 1-bounded i.e. the number of markers at every place in any reachable marking $m \in [m_0]$ does not exceed 1) (5) connected and (6) USF. By saying that a net (instead of STG) is canonical we it to satisfy the later four properties.

Lemma 21 (Canonical net properties.) (1) if $v_i \parallel v_j \Rightarrow v_i \not\prec v_j$ and vice versa - $v_i \prec v_j \Rightarrow v_i \not\parallel v_j$ i.e. the concurrency and conflict relations are never interleaved; (2) $\forall f_i, f_j \in \bullet t_k (t_k \in T) : CFS(f_i) \equiv CFS(f_j)$; (3) in canonical STG (sub)cut uniquely identifies the system (partial) state; (4) $\forall v_i, v_j \in P \cup T : v_i \parallel v_j \Leftrightarrow v_i^d \prec v_j^d$ i.e. if any two vertices of the canonical net N are concurrent the corresponding vertices of the net N^d are in conflict and vice versa; (5) if a net N is canonical its dual net N^d is also canonical; (6) $(N^d)^d = N$ i.e. duality is bidirectional;

2.2 Expressiveness of canonical STG model.

The general goal is implementing the specified behavior by a bounded speed-independent circuit with single-output gates. The most general STG implementable in this framework is k-bounded, output persistent, live, consistent and satisfy the Complete State Coding (CSC) property. CSC can be enforced by STG transformation. Refining STG to enforce CSC (also called encoding) is considered in previous works including [4] and for this framework in sections 3 and 4. Hence the only unnecessary in general, but required in our framework restrictions are 1-boundedness and USF.

Safeness (1-boundedness). It is well-known that for any k-bounded PN there exists a 1-bounded PN specifying the same behavior. In [2] it is proven to preserve the original (k-bounded) net interleaving behavior. Thus, translation of any k-bounded PN to a safe one is possible and limiting the class of STGs to be safe does not lead to the loss of generality.

USF. This property requires that the choice and concurrency relations do not interleave. The form of such an STG mimics the form of corresponding state graph still having the power of compact representation of concurrency. Apart from the formal definition 24 the Figure 1 illustrates simple non-USF and a trace equivalent USF STGs.

Theorem 21 (Existence of equivalent canonical net.) *For every bounded, live, consistent STG there exists a trace equivalent canonical STG specifying the same interleaving behavior.*

Indeed as follows from STG boundedness the state graph is finite and the most naive USF STG can be obtained by translating the state graph to an STG by substituting states by places. Such a STG is USF because it contains no concurrency. The theorem can be proven more precisely on the net level by exploiting the net language, but the idea behind is generally the same.

Although most of controlled choice is not allowed in USF, the latter does not imply free-choice. Figure 1 (right) demonstrates a canonical non-FC STG.

We have shown that every STG implementable in the considered framework can be transformed to the canonical form. This sometimes comes at the cost of increase in the STG size, but the specification clarity is also improved.

3 STG analysis.

In this section we present the algorithm to discover CSC and normalcy violations, and show how the USF is checked dropping the rest of canonicity analysis as being well-known.

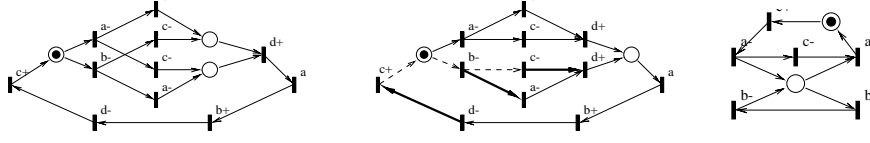


Fig. 1. FC non-USF STG (left) and a trace equivalent STG satisfying USF (middle). Dashed arcs show a D-path between the bold cuts. Non-free-choice USF STG (right).

To make the STG analysis and refinement procedures more efficient both concurrency and choice relations are used. The relations are dual for connected USF nets hence the algorithm for deriving concurrency relations used for a net N can be applied to the net N^d and the relations computed are being the choice relations for the net N . The complexity of computing the relations is shown in [5] to be $O(n^4)$. As the concurrency relations are the same for all linear parts of a net the relations are computed for the net with linear parts collapsed and therefore with lower n .

To check the USF property it is enough to compute the concurrency relations for the arcs of every place $p_i \in P$ such that $|\bullet p_i| > 1$. p_i satisfies the USF iff $\forall f_j, f_k \in \bullet p_i : CcS(f_j) \equiv CcS(f_k)$.

3.1 Checking STG for CSC and normalcy.

Two STG states s_i, s_j are *comparable* $s_i \cong s_j$ iff their cubes are (bitwise) comparable i.e. denote $c_i[k]$ to be a k -th bit of c_i then $c_i \leq c_j$ iff $\forall k : c_i[k] \leq c_j[k]$.

Two STG states $s_i \neq s_j$ are in (*Unique State Coding*) *USC conflict* iff $c(s_i) = c(s_j)$ their cubes are equal. Note that s_i, s_j being in USC conflict implies $s_i \cong s_j$.

Definition 31 Two STG states s_i, s_j are mutually normal if they are either incomparable $s_i \not\cong s_j$ or for every circuit signal z_k enabled in either s_i or s_j holds: either positive: $c(s_i) \geq c(s_j) \Rightarrow f_{z_k}(s_i) \geq f_{z_k}(s_j)$ or negative: $c(s_i) \geq c(s_j) \Rightarrow f_{z_k}(s_i) \leq f_{z_k}(s_j)$ condition, where $f_{z_k}(s_i), f_{z_k}(s_j)$ - values of a circuit signal z_k in the next states of s_i, s_j and $c(s_i), c(s_j)$ are the cubes of s_i, s_j ; otherwise s_i, s_j are in normalcy conflict.

Definition 32 Two STG states s_i, s_j are in CSC conflict iff $c(s_i) = c(s_j)$ and $\mathcal{E}(s_i) \downarrow T^Z \neq \mathcal{E}(s_j) \downarrow T^Z$, where $\mathcal{E}(s_i) \downarrow T^Z$ is a projection of the set of transitions $\mathcal{E}(s_i)$ enabled in s_i on the set of circuit signals' transitions.

STG \mathcal{G} is *normal* if for every circuit signal z_k either positive or negative normality condition holds for every two distinct STG states s_i, s_j ($s_i \neq s_j$). STG \mathcal{G} satisfies CSC property if neither two STG states s_i, s_j are in CSC conflict.

Keeping in mind the property 3 of Lemma 21 further on we work with (sub)cuts.

Concurrency relation $CcS(C_j) = \bigcap CcS(p_i) \mid p_i \in C_j$. Then, two subcuts C_i, C_j are *equally precise* ($C_i \rightleftharpoons C_j$) iff $CcS(C_i) \equiv CcS(C_j)$, C_i is *less precise* than C_j ($C_i \rightarrow C_j$) iff $CcS(C_i) \supset CcS(C_j)$ (C_j defines the state of more signals than C_i); *more precise* ($C_i \leftarrow C_j$) iff $CcS(C_i) \subset CcS(C_j)$; C_i, C_j are *compatible* ($C_i \sim C_j$) if they are equally precise or one of them is more precise than the other.

As follows from 31, 32 normalcy and CSC conflicts can be identified by first identifying the comparable and equal subcuts and analyzing the sets of transitions enabled in these states afterwards. A conflict identified between subcuts stands for at least one pair (usually a set of pairs) of cuts and therefore states.

3.2 D-path.

A *path* from some PN vertex v_i to another vertex v_j ($v_i \rightarrow v_j$) is a nonempty sequence v_i, \dots, v_j of nodes (vertices) satisfying $(v_i, v_{i+1}), \dots, (v_{j-1}, v_j) \in F$, where the set $\{(v_i, v_{i+1}), \dots, (v_{j-1}, v_j)\}$ is a set of *arcs generated on the path*.

In order to identify the comparable (as well as equal) states without traversing we use a distance path or D-path \mathcal{P}^D . D-path is illustrated on the Figure 1 in the middle. The reason for the name is that the path is used to calculate the Hamming distance between the cubes corresponding to STG subcuts. D-path is defined between (sub)cuts hence it is a multipath. We restrict it to contain neither circuits (loops) nor choice inside the path. Formally we have:

Definition 33 D-path between STG subcuts C_i (called start) and C_j (called end) denoted as $\mathcal{P}^D_{C_i, C_j}$ is a multipath between subcuts $C_i \rightarrow C_j$ restricted as follows: (1) $\forall v \in \mathcal{P}^D_{C_i, C_j} : v \notin CcS(C_i) \cup CcS(C_j)$ i.e. the path contains no vertices concurrent to either of the start and stop subcuts; (2) $\forall v_i, v_j \in \mathcal{P}^D_{C_i, C_j} : v_i \neq v_j$ i.e. neither two vertices of the path are in choice; (3) $\forall p_k \in C_i$ there is a path $p_k \rightarrow p_l$ such that $p_l \in C_j$ and vice versa ($\forall p_l \in C_j$ there is a path $p_k \rightarrow p_l$ from at least one $p_k \in C_i$) and $p_k \rightarrow p_l \subseteq \mathcal{P}^D_{C_i, C_j}$;

It can be also shown that in canonical STG \mathcal{G} for any two compatible subcuts $C_i \sim C_j \mid C_i, C_j \in \mathcal{G}$ there exist at least two D-paths $\mathcal{P}^D_{C_i, C_j}$ and $\mathcal{P}^D_{C_j, C_i}$. In an STG with choice there may exist more than one distinct D-paths $\mathcal{P}^D_{C_i, C_j}$ (in the same direction).

3.3 Metrics.

D-path contains all and only the transitions that definitely fire to move the system from the state corresponding to the start (sub)cut to the state corresponding to the end (sub)cut. Therefore every transition $t \in T, t \in \mathcal{P}^D_{C_i, C_j}$ changes the corresponding bit (the corresponding signal state) in c_j relative to c_i .

Let $Diff^+, Diff^-$ be the metrics that reflect the numbers of signals changed its value from 0 to 1 and from 1 to 0 respectively after firing all transitions $t_i \in \mathcal{P}^D$.

To obtain these metrics on the set of transitions of the D-Path we build $T^{diff} \subseteq \mathcal{P}^D \downarrow T$ such that $\forall t_i \in T^{diff} : \Delta(t_i) \parallel C_i, \Delta(t_i) \parallel C_j$ where $\Delta(t_i) \parallel C_i$ iff $\exists t_j \parallel C_i$ such that $\Delta(t_i) = \Delta(t_j)$ i.e. a set containing only the transitions of signals concurrent to neither start nor end (sub)cut of the D-path. On the d-path shown on Figure 1 (middle) $T^{diff} = \{c+, b-, c+\}$.

We also remove from T^{diff} all pairs of transitions that "compensate" each other - pairs of transitions of the same signal and opposite direction (t_i-, t_j+) where $\Delta(t_i+) = \Delta(t_j-)$ since they do not contribute to the difference between the cubes. The sought metrics $Diff^+$ and $Diff^-$ are the numbers of positive and negative transitions of the resulting set T^{diff} . After removing compensated signal transitions on the Figure 1 $T^{diff} = \{b-\}$ hence $Diff^+ = 0, Diff^- = 1$.

From the definition of the D-path and the STG canonicity it follows that equal metrics are obtained for any D-path for two given compatible subcuts C_i, C_j hence any of the D-paths for C_i, C_j can be used for calculation.

3.4 Identifying comparable subcuts.

Equally precise subcuts enjoy the following important property: $C_i \rightleftharpoons C_j$ implies that $\exists C^{max}_k, C^{max}_l$, such that $C_i \subset C^{max}_k, C_j \subset C^{max}_l$ and $(C^{max}_k \setminus C_i) \equiv$

$(C_i^{max_l} \setminus C_j) \subseteq CcS(C_i)$ and as follows from metrics and D-path definitions $Diff^+_{C_i, C_j} = Diff^+_{C_i^{max_k}, C_i^{max_l}}$ and $Diff^-_{C_i, C_j} = Diff^-_{C_i^{max_k}, C_i^{max_l}}$ i.e. both equally precise subcuts can always be extended with the same sets of places to form a nonempty set of pairs of cuts with the same metrics between them.

Lemma 31 (USC and comparability in terms of distance.) *Consider two equally precise subcuts $C_i \rightleftharpoons C_j$ and the corresponding distances $Diff^+$ and $Diff^-$.*

1. C_i, C_j are comparable iff $Diff^+ \equiv 0 \vee Diff^- \equiv 0$;
2. C_i, C_j are in USC conflict iff $Diff^+ \equiv 0 \equiv Diff^-$.

A subcut C_i is *potentially* comparable or in USC conflict with another subcut C_j if they are compatible and the corresponding condition holds for the metrics.

The idea behind the two subcuts being *potentially* comparable or in USC conflict is that the less precise of them can possibly be extended to form a real conflict. The subcuts are extended iteratively in the search of the conflict.

The algorithm for identifying the subcuts in USC conflict is drafted in 31. It starts from initializing PC with one-place subcuts composed of every STG place except for the places p where all transitions from $p \bullet$ are dummy i.e. $p_i \bullet \cap T^Y \equiv \emptyset$. On the step 4 for every subcut C_i from PC the set of places with zero metric relative to C_i is identified. Then all (sub)cuts are built on the set PP . If the obtained subcut is equally precise with C_i the conflict is in place. The more precise subcut is potentially in conflict - added to the set PC for further analysis. Otherwise the obtained subcuts are discarded. It can be shown that this algorithm is capable of identifying all comparable subcuts in the STG.

Algorithm 31 (Identifying (sub)cuts in USC conflict.)

- 1: $PC := \{C_i\}$ such, that (1) $|\{C_i\}| \leq |P|$, (2) $\forall C_i : C_i = \{p\} \in P$;
- 2: **for** any subcut $C_i \in PC$
- 3: $PC := PC \setminus C_i$;
- 4: $PP0 := \{p_j \mid Diff^+_{C_i, p_j} = 0 = Diff^-_{C_i, p_j}\}$;
- 5: $PC0 := \{C_k\}$ such that $\forall C_k \in PC0 \text{ iff } \forall p_l \in C_k : p_l \in PP0$;
- 6: **for** any subcut $C_k \in PC0$;
- 7: **if** $C_k \leftarrow C_i$ **then** $PC := PC \cup C_k$;
- 8: **else if** $C_k \rightleftharpoons C_i$ **then** C_k, C_i are in USC conflict;

The same algorithm is used for identifying comparable subcuts by composing also the sets $PP1, PC1$ (at the same step as $PP0, PC0$) with appropriate metric $PP1 := \{p_j \mid (Diff^+_{C_i, p_j} = 0) \vee (Diff^-_{C_i, p_j} = 0)\}$. The metrics obtained are important for later identifying if comparable subcuts are mutually normal.

It is possible that two subcuts C_i, C_j have the same set of defined signals, but $C_i \not\rightleftharpoons C_j$. The reason is that the set of concurrent transitions is taken into account for the subcuts to decide if they are equally precise, while the sets of concurrent signals are meaningful for cubes. Because of that, two partial states with comparable cubes may not be identified by the algorithm, but the more precise (sub)cuts containing those comparable are always identified.

The algorithm complexity depends on the number of USC conflicts and/or comparable subcuts in the analyzed STG. In comparison to the state based techniques the number of conflicts is usually reduced for STGs with concurrency by representing sets of conflicts between states by conflicts between subcuts.

3.5 Identifying CSC and normalcy conflicts.

For identifying CSC conflicts among the USC conflicts the sets $\mathcal{E}(C_i)$ and $\mathcal{E}(C_j)$ are projected on the set of circuit signal transitions and the obtained sets are

compared. Their difference points out a CSC conflict between C_i and C_j . Transition t_j is enabled by a subcut C_i ($t_j \in \mathcal{E}(C_i)$) iff $\forall f_k \in \bullet t_j : \bullet f_k \in C_i \vee f_k \in CcS(C_i)$ i.e. the places that form the t_j enabling condition must either belong to C_i or be concurrent to it and therefore could be included in the cut $C^{max}_l \supset C_i$.

In order to identify the subcuts that are not mutually normal among comparable subcuts in addition to $\mathcal{E}(C_i)$ and $\mathcal{E}(C_j)$ we use the set T^{diff} used for metrics' calculation. $Diff^- > 0 \Rightarrow c_i > c_j$; $Diff^+ > 0 \Rightarrow c_i < c_j$. Therefore as follows from 31, 33 and STG consistency the following conditions stand for the normalcy conflict between C_i and C_j .

Let inequality condition for a signal y_k (if it holds $f_{z_k}(C_i) \neq f_{z_k}(C_j)$) be $\mathfrak{S}(y_k) = (t_{y_k} \notin T^{diff} \wedge t_{y_k} \notin \mathcal{E}(C_j)) \vee (t_{y_k} \in T^{diff} \wedge t_{y_k} \in \mathcal{E}(C_j))$. Particularly if in addition to $\mathfrak{S}(y_k)$ takes place $t_{y_k}^- \in \mathcal{E}(C_i)$ then $f_{z_k}(C_i) < f_{z_k}(C_j)$ and $t_{y_k}^+ \in \mathcal{E}(C_i) \Rightarrow f_{z_k}(C_i) > f_{z_k}(C_j)$. Given that the appropriate condition is chosen for negative and positive signals according to the definition 31.

4 STG refinement for CSC and normalcy.

In this section we assume that STG is canonical and the conflicts to be solved between pairs of equally precise subcuts are identified. The goal is to transform the STG to the form satisfying both properties at the same time preserving the STG canonicity. In addition it may be required that the circuit interface with its environment is preserved.

To perform the transformations we define the set of transformation *rules* ϕ called *kit* $\Phi = \{\phi_i\}$.

4.1 Net kit.

The net kit is depicted on the Figure 2. For every rule ϕ we define its input $In(\phi^*)$ being the set of original net elements affected by the transformation and conditions on the input necessary to preserve the net canonicity properties during transformation. Construction rules are clear enough from the Figure 2.

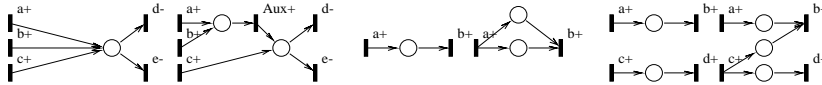


Fig. 2. Net kit construction rules. Original-transformed pairs from left to right: transition insertion ϕ^{TI} , concurrent dependency insertion ϕ^{CDI} , concurrency reduction ϕ^{CR} .

In general the named rules can be applied in both directions with some restrictions, but for the needs of refinement we only need *synthesis rules* (the rules introducing new elements in the net) so we keep this assumption.

Transition insertion rule ϕ^{TI} : Input of the rule is $In(\phi^{TI}) = F^{tr}$, where $F^{tr} \subset F$. Conditions: (1) $F^{tr} \subseteq \bullet p_k$, where $p_k \in P$. Thus, F^{tr} is loosely speaking a set of arcs to be moved to the place input for the new transition. It is important to move only the arcs input to some place for preserving the net liveness keeping in mind that canonical net is not necessarily free-choice. By construction $CcS(t^{new}) = CcS(In(\phi^{TI})) = CcS(f_i)$, where $f_i \in F^{tr}$ and $CfS(t^{new}) = CfS(In(\phi^{TI})) = \bigcap CfS(f_i)$ for all arcs $f_i \in F^{tr}$. (We'll use it later.)

Concurrent dependency insertion ϕ^{CDI} : $In(\phi^{CDI}) = (t_{src}^{orig}, t_{dst}^{orig})$ being source and destination transitions for the dependency. Conditions: (1) $t_{src}^{orig} \prec t_{dst}^{orig}$ i.e.

transitions must be ordered (to distinguish this rule from ϕ^{CR}); (2) $CfS(t_{src}^{orig}) \equiv CfS(t_{dst}^{orig})$ i.e. their conflict relations must be identical to satisfy the property 4 of the canonical nets.

Concurrency reduction ϕ^{CR} : $In(\phi^{CR}) = (t_{src}^{orig}, t_{dst}^{orig})$ being source and destination transitions for the dependency. Conditions: (1) $t_{src}^{orig} \parallel t_{dst}^{orig}$ i.e. transitions are concurrent; (2) $CfS(t_{src}^{orig}) \equiv CfS(t_{dst}^{orig})$ (as in ϕ^{CDI}).

4.2 STG kit.

In this section the net kit introduced in 4.1 is used to build the rules of the STG canonicity preserving kit. The STG specific properties of the canonical nets are consistency and output persistency.

Consistency is a trace based property requiring alternation of the transitions of the same signal in every feasible trace. As a consequence the rules that do not alter the net language cannot introduce consistency violations. Output persistency implies indeterministic choice for circuit signal transitions.

Obviously neither ϕ^{CDI} nor ϕ^{CR} introduce neither new choice nor new signal transitions hence persistency is always preserved. ϕ^{CR} eliminates some feasible traces by reducing the STG concurrency, but does not change the order of events in the remaining traces. ϕ^{CDI} does not alter the language at all since the involved transitions are already ordered, what is being one of conditions for this rule. So far both ϕ^{CDI}, ϕ^{CR} always preserve the original STG canonicity and can form part of the STG kit with no additional restrictions.

ϕ^{TI} introduces a new transition - a new event. Inserting the new transition before the original place ensures preserving output persistency. However preserving consistency needs special care if the new event represents a signal transition (not just a dummy) because the STG language is affected in the case.

To formalize these further restrictions we build a new rule *signal insertion* ϕ^{SI} as a part of STG kit on the top of ϕ^{TI} . $In(\phi^{SI}) = (T^+, T^-)$ Denote T^* to refer to either T^+ or T^- . Then $T^* = \{In(\phi^{TI})_i\}$ i.e. the sets of inputs for the transition insertion rule of the net kit. Conditions: (1) let $p(\phi^{TI}) = f \bullet | f \in \phi^{TI}$ (by definition $p(\phi^{TI})$ is unique) then the condition is $\forall \phi_i^{TI} \in T^+ : \exists p_j = p(\phi_j^{TI})$, such that $\phi_j^{TI} \in T^-$ and $\exists \mathcal{P}^D_{\{p_i\}, \{p_j\}}$ and vice versa i.e. we generally require that $In(\phi^{TI})$ of T^+ and T^- alternate to make the resulting signal transitions alternate as well; (2) $\nexists \phi_i^{TI}, \phi_j^{TI} \in \phi_k^{SI}$, such that $\phi_i^{TI} \parallel \phi_j^{TI}$ i.e. avoid autoconcurrency.

Finally the canonicity preserving STG kit Φ^{STG} consists of the rules $\Phi^{STG} = \{\phi^{CDI}, \phi^{CR}, \phi^{SI}\}$.

Preserving the interface with environment is fairly simple. Denote *direct dependency* $Dep(t^{src}, t^{dst})$ to be a path $t^{src} \rightarrow t^{dst}$ where t^{src}, t^{dst} are signal transitions and that contains no signal transitions except for t^{src}, t^{dst} . Let \mathcal{I} be the set of all STG direct dependencies $Dep(t^{src}, t^{dst})$, such that $\Delta(t^{dst}) \in X$. Then $In(\phi^{SI})$ is interface preserving iff $\nexists In(\phi^{TI}) \in In(\phi^{SI})$, such that $In(\phi^{TI}) \cap \mathcal{I}$ i.e. the signal transition rule is interface preserving iff it does not introduce any transition in such a way that an environment signal transition comes to directly depend on it.

4.3 Solving CSC and Normalcy conflicts.

We consider applying the STG kit Φ^{STG} 4.2 to solve CSC and normalcy conflicts. We avoid the use of ϕ^{CR} whenever possible because it alters the initially specified behavior by reducing concurrency.

Normalcy and CSC conflicts are similar in the nature. In the case of USC it is necessary to distinguish two cubes c_i, c_j , while for normalcy - to make them incomparable. It can be done by extending them with one more bit with different values in c_i and c_j . Clearly to make c_i, c_j incomparable the value matters.

This is usually done by inserting new signal transitions in the D-paths corresponding to the conflict subcuts. Thus, to solve a conflict $C_i < C_j$ it is necessary to apply $In(\phi^{SI}) = (T^+, T^-)$ such that loosely speaking $T^+ \cap \mathcal{P}^D_{C_j, C_i} \neq \emptyset, T^- \cap \mathcal{P}^D_{C_i, C_j} \neq \emptyset$. If $CfS(C_i) \neq CfS(C_j)$ to fulfill the ϕ^{SI} condition some more transitions must be introduced to ensure the STG consistency. Transition insertion may be done as sequentially - by forming the $In(\phi^{SI})$ out of existing arcs so concurrently by first applying ϕ^{CDI} so that $In(\phi^{CDI}) \subset \mathcal{P}^D_{C_i, C_j}$ or $In(\phi^{CDI}) \subset \mathcal{P}^D_{C_j, C_i}$ i.e. with source and destination being inside the D-path (such a dependency will be a part of the D-path by definition).

It can be shown that the D-path contains only a single transition t_0 (no space for transition insertion) only in case t_0 is involved in *signal persistency* (SP) conflict. Although SP is not crucial for implementability of the behavior specified by the STG in such cases the problem must be solved to solve the CSC or normalcy conflict involved. The algorithm of enforcing STG signal persistency is out of scope of this paper, so we only mention that ϕ^{CR} is used for this purpose as well as possibly the other two rules of the STG kit as shown on the Figure 3.

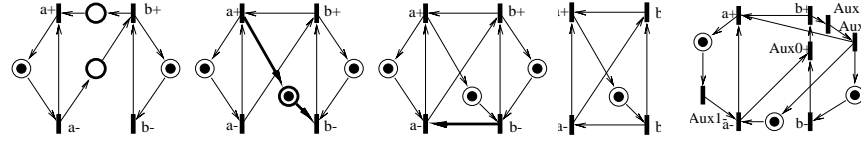


Fig. 3. In the initial STG (leftmost) CSC conflict is detected between the subcuts shown by explicit unmarked places (bold). Signal persistency conflict: $a+$ is enabled by $b+$ but disabled by $b-$. On the 2nd STG this conflict is solved by concurrency reduction (bold dependency), but another still exists - solved by another new dependency, redundant arcs are then removed. The rightmost STG presents an alternative solution.

The restriction of preserving interface with environment can make some conflict unresolvable. In that case a new signal transition can be inserted in a way that some environment event known to be slow come to depend on it. Then despite the fact that environment event does not actually depend on the newly inserted signal transition the order of events is not affected due to the delay of environment event.

Heuristics are applied to avoid introducing new conflicts while inserting new signal transitions. Some of them are: (1) avoid inserting two signal transitions so that one of them directly depends on the other (to avoid introducing a new USC conflict); (2) introduce new events in such a way that the direction of signal transitions alternate (to introduce no new pairs of comparable subcuts); (3) in the case a newly inserted transition t^{new} violates the interface with environment recheck the normalcy and CSC conditions for the conflicts involving the subcut(s) enabling t^{new} (to detect the normalcy or CSC conflict for t^{new} if any).

These heuristics are applied during the automatic search of the complete solution enforcing CSC and/or normalcy for the whole STG. In interactive refinement these criterias are used to estimate the quality of a particular transformation.

Interactiveness and feedback. The CSC and normalcy conflicts as well as inputs for the transformation rules in this framework are defined in terms of the

original STG objects (arcs, places, transitions). This allows to present a designer with all information on the conflicts discovered as well as on the solutions available on the initial STG. Thus, any degree of interactiveness is possible with use of this framework from automatically providing a complete solution to generating and displaying the complete set of conflicts and available solutions allowing to choose the most appropriate.

5 Conclusion

The framework is presented for analysis and interactive refinement of the STG for CSC and normalcy. Normalcy guarantees the behavior implementation by logic gates without input inverters contributing to its robustness and the use of negative gates can be maximized, but the circuit quality criteria remains unclear. The complete solution can be built out of first (high quality) simple solutions or the number of signals inserted can be minimized, but we believe that the best solution can only be provided by interactive refinement when the hardly formalizable designer's knowledge of the behavior to be implemented is utilized. This is the main reason for designing this technique in a way to enable any level of designer assistance.

The techniques presented in this paper are implemented in the TaxoSynthesis CAD tool [1]. A strategy to build the complete solution optimising the circuit area and power consumption and/or latency as well as other ways to customize the refinement remain as possible subjects for future research.

References

1. TaxoSynthesis CAD tool. <http://www.lsi.upc.es/~alexs/synth/>.
2. Eike Best and Harro Wimmel. Reducing k-safe petri nets to pomset-equivalent 1-safe petri nets. In *21st International Conference on Applications and Theory of Petri Nets, PN-2000*, June 2000.
3. Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, and Alexandre Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. In *XI Conference on Design of Integrated Circuits and Systems*, Barcelona, November 1996.
4. A. Kondratyev, J. Cortadella, M. Kishinevsky, L. Lavagno, A. Taubin, and A. Yakovlev. Identifying state coding conflicts in asynchronous system specifications using Petri net unfoldings. In *Int. Conf. on Application of Concurrency to System Design*, March 1998.
5. A. Kovalyov and J. Esparza. A polynomial algorithm to compute the concurrency relation of free-choice signal transition graphs. In *Proceedings of the International Workshop on Discrete Event Systems, WODES'96*, pages 1–6, August 1996.
6. Ilya V. Krotchkov, Alexander B. Smirnov, and Nikolai A. Starodoubtsev. Verification driven synthesis of asynchronous circuits from STG specification. In Anne-Marie Trullemans-Anckaert and Jens Sparsø, editors, *Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 377–386, October 1998.
7. Alex Semenov, Alexandre Yakovlev, Enric Pastor, Marco Pe na, and Jordi Cortadella. Synthesis of speed-independent circuits from STG-unfolding segment. In *Proc. ACM/IEEE Design Automation Conference*, pages 16–21, 1997.
8. N. Starodoubtsev, M. Goncharov, I. Klotchkov, and A. Smirnov. Synthesis of asynchronous interface circuits by STG refinement. In *Asynchronous Interfaces: Tools, Techniques, and Implementations*, pages 65–74.
9. N. Starodoubtsev, M. Goncharov, I. Klotchkov, and A. Smirnov. Towards synthesis of monotonic asynchronous circuits from Signal Transition Graphs. In *2nd International Conference on Application of Concurrency to System Design.*, June 2001.