On Hazard-Free Implementation of Speed-Independent Circuits

Alex Kondratyev
The University of Aizu
Aizu-Wakamatsu, 965-80, Japan
Tel: +81-242-37-2557
Fax: +81-242-37-2744
e-mail: kondraty@u-aizu.ac.jp

Michael Kishinevsky
The University of Aizu
Aizu-Wakamatsu, 965-80, Japan
Tel: +81-242-37-2617
Fax: +81-242-37-2744
e-mail: kishinev@u-aizu.ac.jp

Alex Yakovlev*
University of Newcastle upon Tyne
NE1 7RU England
Tel: +44-191-222-8184
Tel: +44-191-222-8232
e-mail: Alex.Yakovlev@newcastle.ac.uk

Abstract — We investigate the problem of hazard-free gate-level implementation of speed-independent circuits specified by event-based models, such as Signal Transition Graph (for processes with AND causality and input choice) or its extension, called Change Diagram (which allows OR-causality). The main result of the paper is twofold: (1) the proof that *any* speed-independent behavior can be implemented at the gate-level without hazards, and (2) an efficient method for such an implementation. This method is based on transformations of the specification to the form satisfying the Monotonous Cover requirement. Since this method is based on standard gate cells it can be used both in the full-custom and semi-custom VLSI design. Experimental results demonstrate area and performance efficiency of our method.

I. Introduction

Speed-independent circuits rely on the *unbounded delay model*, which provides pessimistic assumptions on gate delays (no bounds are known) and realistic assumptions on wire delays (the skew of wire delays at multiple fanouts is less than one gate delay). In such circuits every input change is *acknowledged* by the circuit to indicate that the environment can apply next input pattern. Therefore speed-independent circuits enjoy the property of being *self-checking to stuck-at faults* [12, 15] and are *easier to verify* than bounded delay circuits [5, 6].

Significant progress has recently been achieved in automating the synthesis of speed-independent circuits [2, 11] from Signal-Transition Graph (STG) specifications. From a STG specification it is possible to generate a state-transition description, called here Transition Diagram (TD). The existing techniques for synthesis of speed-independent circuits [2, 1, 15, 11, 4, 17] despite covering a rather large subclass of circuits, still fall short of providing a universal implementation methodology for the largest possible class of speed-independent behaviors.

*This work has been partly supported by EPSRC, grant GR/J5232.

The monotonous cover [8] and the corect cover [1] conditions were introduced as sufficient (and actually close to necessary) conditions for the hazard-free implementation under the standard RS- and C-implementation architectures. The synthesis method was formulated in [8] as a set of Boolean constraints under the rules of introducing additional signals. The solution can be found using the Boolean satisfiability solvers. This approach allows handling only relatively small specifications due to the exponential growth of the size of satisfiability task.

The summary of the results of the paper is as follows. For the speed-independent implementation of correct STG specifications (the basic concepts are defined in Section II) we choose a *standard RS-implementation* and a *standard C-implementation* structures, based on a two-level combinational logic and a latch (either an *RS* flip-flop or a C-element) for each output signal (Section III). Section IV presents a constructive procedure which allows to equivalently transform the initial specification to the form that meets the monotonous cover requirement [8]. From the latter, a speed-independent standard implementation can be directly derived. Only such transformations that preserve the language generated by the specification are allowed here. These results are further extended in several ways. Section V describes how the use of negative feedbacks and complex gates for implementation of up- and down-excitation functions can help in ensuring the monotonous cover requirement. In Section VI the monotonous cover theory is generalized to allow hazard-free implementation for a wider class of specifications, STGs without Unique Entry Condition and specifications with OR causality, called Change Diagrams. We also present experimental results (Section VII) to compare our method with SIS [9] and method from [1].

II. Modelling Circuit Behavior

A. Signal Transition Graphs

A *Signal Transition Graph (STG)* is a free-choice Petri net (PN) [13] whose transitions are interpreted as the signal transitions on the circuit inputs (input transitions) or gate outputs (output transitions). A signal transition can be represented by $+a_j$ for the j-th transition of signal a from 0 to 1 or $-a_j$ for its j-th transition from 1 to 0. $*a_j$ is used to denote either a "$+a_j$" transition or a "$-a_j$" transition. The functioning of STG is similar to that of Petri nets [13]. A transition is enabled if all its predecessor places are marked. When an enabled transition fires, the marking of each predecessor place is decremented, and the marking of each successor place is incremented. STG is graphically represented by a directed graph with transitions denoted by their names and places by circles, with places that have only one predecessor and successor transition are usually omitted. Transitions of input signals are underlined.

Two binary relations, called *precedence* and *concurrency*, are defined for signal transitions. Let $*a_j$ and $*b_k$ be two arbitrary signal transitions. If in every feasible sequence in the STG every i-th occurrence of $*a_j$ enters before the i-th occurrence of $*b_k$, then $*a_j$ *precedes* $*b_k$, denoted by $*a_j \Rightarrow *b_k$. If there is a reachable marking M such that two sequences of signal transitions: $*a_j, *b_k$ and $*b_k, *a_j$, can fire from M, then $*a_j$ and $*b_k$ are said to be *concurrent*, denoted $*a_j \parallel *b_k$.

An example of STG is shown in Figure 1,a. This STG has only one free-choice place $p1$ that is initially marked. This place is a predecessor one for transitions $+a_1$ and $+b_2$ and both of them are enabled initially. The firing of any transition, say $+a_1$, disables $+b_2$ and enables the events $+c_1$ and $+d_1$ that are successors for $+a_1$). One of the firing sequences that is feasible in this STG is $+a_1 + c_1 - a_1 + d_1 + b_1 - c - b - d \ldots$. It is clear that in this example transition $+a_1$ precedes $+b_1$ and transition $+d_1$ is concurrent to $-a_1$.

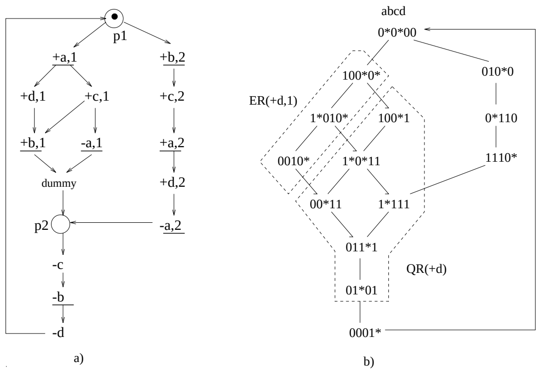

Fig. 1. Example of STG (a) and corresponding TD (b)

Definition II.1 *An STG is* **correct** *iff it satisfies the following four conditions: (1) Each feasible sequence of signal transitions is switchover correct. (2) The set of reachable events contains no autoconcurrent transitions. (3) STG is bounded (the underlying Petri net has to be bounded) (4) Any free-choice place precedes only input transitions.*

The STG correctness is necessary and sufficient for implementability of the STG by a speed-independent circuit[6]. Derivation of logic functions for output signals is usually solved by a state based model, called Transition Diagram (TD) that can be easily generated from the STG.

B. Transition Diagrams

TD is a directed graph where each vertex is in one-to-one correspondence with the markings reachable from initial marking M_0 of a given STG. A TD vertex (state) is labeled with a boolean vector $s = < s^1, \ldots, s^n >$, representing the value of STG signals (n is the number of signals in STG). Two states s_1 and s_2 corresponding to markings M_1 and M_2 are connected with an edge in TD ($s_1 \to s_2$) if M_2 is reachable from M_1 by the firing of some event $*a$ of the STG (edge of TD can be labeled with the $*a$ name: $s_1 \overset{*a}{\to} s_2$). Transition $*a$ is called *enabled* in state s_1. Figure 1,b shows the TD that corresponds to the STG from Figure 1,a.

The notion of speed-independence can be formalized in terms of states as follows[12, 15]. A state w of TD is a *conflict state* if for a pair of signals, a and b, excited in w the firing of b disables a. If a is an output signal, then w is an *output conflict state*; otherwise it is an *input conflict state*. A TD is *semimodular* with respect to state u if no output conflict state is reachable from u. A state w is said to be *detonant* with respect to output signal a if and only if there exists a pair of states u and v that directly follow w (i.e., $w \overset{*b}{\to} u, w \overset{*c}{\to} v$) such that transitions $*b$ and $*c$ are concurrent and a is stable in state w and is excited in states u and v. A TD is *distributive* with respect to state u if it is semimodular and no detonant states are reachable from u.

Output conflict states localize the states in TD where hazards corresponding to output signals can occur. Input conflicts determine the states where the circuit's environment chooses the control flow. In its initial state 0*0*00 (Figure 1,b), both a and b signals are excited but the firing of any one of them disables the excitation of the other (see states 100*0* and 010*0). Thus 0*0*00 is a conflict state and the presented TD is not semimodular. As a and b signals are the input signals, state 0*0*00 is an input conflict state. There are no other conflict states in our TD, so it is semimodular and can be implemented by a speed-independent circuit. Distributivity is an important case of semimodularity. There are no detonant states in the TD of Figure 1,b and this TD is distributive.

Important structural properties of TDs are given by the following definitions.

An *excitation region*[15] of signal a in TD is a maximal connected set of states in which a has the same value and is excited. A *quiescent region*[1] of signal a in TD is the maximal connected set of states in which a has the same value and is not excited. A state u is a *minimal* state for the excitation region $ER(*a_i)$ if it has no predecessors within the region. It is denoted as $u_{min}(*a_i)$. An excitation region satisfies the *unique entry condition* (UEC) if it has exactly one minimal state. A transition $*b_j$ (as well as its underlying signal b) is called a *trigger* one for transition $*a_i$ if by firing $*b_j$ we can enter the excitation region $ER(*a_i)$. A trigger transition $*b_j$ is *non-persistent*[3] to $*a_i$ if b is concurrent to $*a_i$; otherwise, it is said to be *persistent*. A TD is *persistent* if for each excitation region $ER(*a_i)$ the corresponding transition $*a_i$ is persistent to its trigger signals.

The excitation region corresponding to transition $*a_i$ will be denoted as $ER(*a_i)$. We will call an excitation region that corresponds to a "$+a$"("$-a$") transition an *up-excitation region* (*down-excitation region*). The quiescent region of transition $*a_i$, $QR(*a_i)$, is the set of states between $ER(*a_i)$ and $ER(*a_{i+1})$, where $*a_{i+1}$ denotes the next transition of signal a. A constant function region, $CFR(*a_i)$, corresponding to transition $*a_i$ is $ER(*a_i) \cup QR(*a_i)$. On Figure 1,b the up-excitation region $ER(+d_1)$ and the following quiescent region $QR(+d_1)$ are shown by dashed lines. The notion of "unique entry condition" is important because it is a necesssary condition for the existence of a *single cube* to cover an excitation region.

If a trigger transition $*b_j$ is non-persistent to $*a_i$, there has to be a state v in $ER(*a_i)$ in which $*b_{j+1}$ is excited. This is easily seen in our TD example. We can reach the minimal state of $ER(+d_1)$ (state 100*0* see Figure 1,b) only by firing trigger transition $+a_1$. However, inside $ER(+d_1)$, transition $-a_1$ is excited, which leads to the non-persistency of transition $+a_1$ with respect to $+d_1$.

III. Basic Implementation Structures

We consider two implementation structures: one with Muller C-elements and the other with RS-latches used as asynchronous memory elements for restoring output signals. We call them a standard C-implemenetation and a standard RS-implemenetation, respectively. Both structures are essentially the same except that the latter is *dual-rail* encoded. A structure implementing a function for one signal is called a *signal network*.

Our synthesis strategy for deriving a signal network for each output signal consists of the following steps (for the case of a standard C-implementation):

(1) for each excitation region $ER(*a_i)$ derive a **region function** $S_a(i)$ or $R_a(i)$ as a single cube implemented by AND gate;

(2) combine the region functions corresponding to the up-excitation regions (down-excitation regions) with an OR gate to create an **up-excitation function** S_a (**down-excitation function** R_a) for a;

(3) combine the functions S_a and R_a with the C-element to create the signal network, as depicted in Figure 2,a.

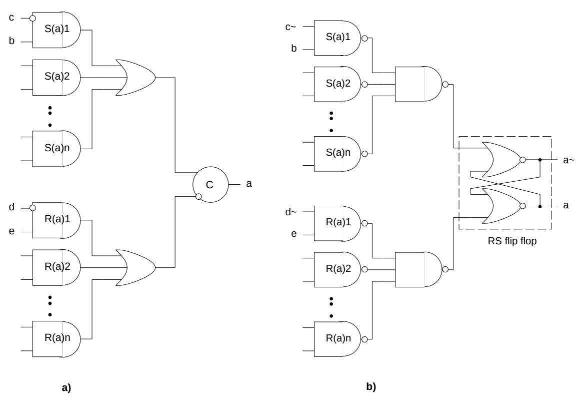

Fig. 2. Signal network for standard C- (a) and RS- (b) implementation structures

For implementing a cube of an excitation function with inverse literals in C-implementation AND-gates with input inversions are needed. When RS flip-flops are used, all the internal signals of a circuit are already implemented in dual-rail form that allows to replace all the inverse occurrences of each internal signal in the region functions by the signal from the inverse output of a corresponding RS flip-flop. If all input signals are also presented in dual-rail form, all the region functions can be implemented simply by AND-gates (or better $NAND$-gates) without input inversions. From the suggested implementation architecture follows that the main synthesis task is to ensure the correspondence *one AND gate → one excitation region*. This is possible for TDs satisfying the Monotonous Cover requirement.

Definition III.1 (Cover cube) *A cube $c = c_1, \ldots, c_k$ is said to be a* **cover cube** *for the excitation region $ER(*a_i)$, denoted as $c(*a_i)$, if each literal c_j corresponds to some signal b ordered with a_i and $c_j = b$ if b has the value 1 in $ER(*a_i)$, otherwise $c_j = \overline{b}$.*

Definition III.2 (Monotonous cover (MC)) *A cover cube $c(*a_i)$ is said to be a* **Monotonous Cover** *(MC) for $ER(*a_i)$ if (1) $c(*a_i)$ covers all states $ER(*a_i)$, (2) $c(*a_i)$ changes at most once in any trace of states inside $QR(*a_i)$ and (3) $c(*a_i)$ can cover only states from $CFR(*a_i)$ and does not cover any reachable state from other $CFR(*a_j)$.*

If for every excitation region of output signals in TD there exist a corresponding monotonous cover cube, then we will say that TD satisfies the Monotonous Cover (MC) requirement. To optimize the logic of region function we can allow the cover cubes for different excitation functions to be implemented on one AND-gate. To deal with such kind of optimization, the definitions of monotonous cover can be generalized to cater for a set of transitions [8].

Theorem III.1 *[8] If the excitation function R_a and S_a for each output signal a from the TD is represented as sum of cubes $c(-a_1), \ldots, c(-a_k)$, and $c(+a_1), \ldots, c(+a_k)$, respectively, where 1) $c(*a_i)$ corresponds to the monotonous cover of some excitation regions $ER(*a_{i1})$, $ER(*b_{i2})$, $\ldots$, $ER(*d_{ik})$ and 2) each region $ER(*a_i)$ is covered by exactly one cube, then both standard* RS- *and C-implementations are semimodular.*

Theorem III.1 provides both necessary and sufficient conditions for guaranteeing implementation correctness if two special cases are taken into account:

Degenerative Sum-of-Product. A SOP implementation becomes degenerative, for example, if a cube c consists of one literal (b or $\overline{b}$) and/or the corresponding excitation function (e.g., S_a) consists of cube c only. Then we can remove the *AND* and *OR*-gate from its implementation and connect the output b directly to the corresponding input of C-element or *RS* flip-flop. In this case it is not necessary to demand from c to change only once in corresponding quiescent region.

Extending don't cares for the standard C-implementation. To operate correctly, an *RS*-flip-flop based on *NOR*-gates requires the R and S functions to be disjoint, i.e., the following condition should be met: $R \cdot S \equiv 0$. For the standard C-implementation this condition is not necessary, which allows one to expand the *dc-set* for the R and S functions. Indeed, if the output a of a C-element is in the state 1 and its S_a input has also the value 1, the value on the other input $\overline{R_a}$ is of no importance for the output behavior. Therefore function R_a can be set to 1 *before* $ER(-a)$, i.e. in the states of $QR(+a)$ where $S_a = 1$, which formally violates the monotonous cover condition.

IV. Reduction to the MC form

If the initial specification does not satisfy the MC requirement, it has to be altered without changing the input-output behavior of the specification. These transformations preserve the language generated by the specification. The task thus stated is, to reduce the specification to the MC-form by adding extra signals in such a way that these signals will be internal for the implementation. For the outer observer, the implementation will show the same behavior as the initial specification, except, probably, for timing/performance characteristics.

Two questions however arise here:

1. Is such a reduction always possible?
2. What are the methods of doing it?

For a correct STG the answer to the first question is positive. This fact will be proved in a constructive way, i.e. by presenting an efficient technique that allows the reducing of any STG to its MC form, so the solution for the second problem will be shown altogether.

In [8] it was shown that the violation of Complete State Coding requirement [3] is a particular case of Monotonous Cover violation. However, the problem of ensuring the Complete State Coding is well investigated and has efficient methods for its solving [6, 14, 10]. Therefore, further for simplicity, we will work with the specification that already satisfies the CSC property assuming that the necessary transformation is done in advance. Similar reasons apply to the Unique Entry Condition. We shall thus refer to STGs that are safe[1] and satisfy both the CSC and UEC conditions as *strongly correct* STGs.

We will reduce STGs to MC form in two steps:

1. STG reduction to the persistent form.
2. Reduction of the persistent STG to the MC form.

A. Eliminating non-persistency

The idea of transformation ensuring the persistency is illustrated in Figure 3 (where indirect precedence relations are shown by dotted lines). It is quite straightforward: if the trigger transition $*b_j$ is non-persistent to $*a_i$, to eliminate this non-persistency we can introduce the transition of an additional signal x (e.g., $+x$), between $*b_j$ and $*a_i$, in such a way that the opposite transition of signal x ($-x$) will be ordered with $*a_i$ and thus $+x$ will become persistent to $*a_i$. Of course, such a transformation can be done in different ways – transition $+x$ must be inserted between $*b_j$ and $*a_i$ but for the opposite transition we have more freedom. Finding the right place to insert a new signal transition can strongly affect the size of logic and create opportunities for logic optimization.

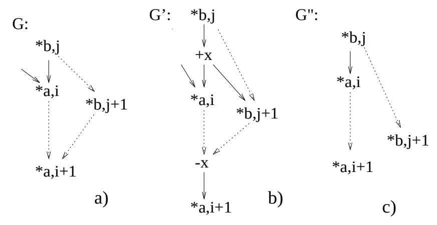

Fig. 3. Reduction of STG to a persistent form

Although, in general method of Figure 3 not always possible the following theorem gives the upper bound on the *number* of signals that have to be inserted to remove non-persistency for a given pair of transitions in STGs with choices. Moreover, it shows *where* these signals should be inserted. In many practical cases, the required transformations can be simplified in the way that was shown earlier.

Theorem IV.1 *Assume that there is non-persistency between a pair of transitions $*b_j$ and $*a_i$ in a safe correct STG. If $*b_j$ does not violate Unique State Coding requirement, then non-persistency between $*b_j$ and $*a_i$ can be eliminated by introducing two aditional signals and no new non-persistencies arise.*

[1]A PN is called *safe* if for every reachable marking each place can have at most one token.

B. Reduction of persistent STG to MC-form

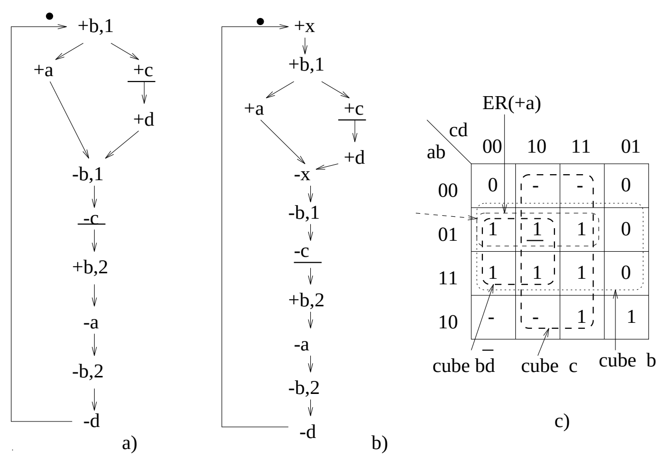

Fig. 4. Reduction of STG to MC-form

We can now consider the reduction of STGs to the MC form as applied only for persistent strongly correct STGs. First, we illustrate the method by example of the specification shown in Figure 4,a, with input signal c and output signals a, b, d. In this STG, all output signals are persistent. The Karnaugh map corresponding to the logic function of signal a is shown in Figure 4,c. Excitation region $ER(+a)$ consists of three states. The only signal that is ordered with $+a$ is b, and thus cube b is chosen to cover $ER(+a)$. However, this cover is neither monotonous nor even correct because cube b also covers two states from the *off-set* of the function for a. To make this cover correct, we need to split cube b into two cubes $b\overline{d}$ and c (term takeover phenomenon [15]) but this violates the MC requirement because *two* cubes will now be turned on in state 0110 (underlined in Figure 4) inside *one* excitation region. Therefore, function S_a does not meet the MC requirement and is thus hazardous when implemented by simple gates. For all other excitation functions the MC requirement is satisfied and the logic for their implementation is hazard-free. The set of excitation functions is:

$R_b = acd + \overline{a}d\overline{c}$, $S_b = \overline{d} + a\overline{c}$, $R_d = a\overline{b}$, $S_d = c$, $R_a = b\overline{c}d$, $S_a = b\overline{d} + c$ (1)

To reduce the cover for $ER(+a)$ to the MC-form, we need to distinguish the two situations: when signal b is equal to 1 after firing $+b_1$ (a has to be equal to 1); and b is still in 1 (after $+b_2$) but a has to become 0 after $-a$. This can be done, e.g., by adding signal x that will be high in $ER(+a)$ and will become low before $-a$ (see Figure 4,b). Such a transformation will result in the following logic:

$b = a\overline{c}d + x$, $R_d = \overline{a}\overline{b}$, $S_d = c$, $R_a = \overline{c}b\overline{x}$, $S_a = bx$, $R_x = ad$, $S_x = \overline{d}$ (2)

The area estimate of the logic corresponding to standard C-implementation for (1) and (2) in the SIS library shows that the hazard-free implementation is even smaller than the original hazardous one: 464 area units for (1) and 374 for (2). The area of hazard-free implemenetation can be further reduced down to 344 units by extending don't cares for signal x. This allows to implement signal x simply by a C-element with an inverted output. The result strongly depends on the place where the transitions of signal x are inserted. In our case, the logic is reduced because after adding x signal b can be implemented by a hazard-free combinational circuit without a latch. This is an issue of the logic optimization strategy.

The following statement, similar to Theorem IV.1, holds for the reduction to MC form.

Theorem IV.2 *Let, in a persistent strongly correct STG G, the MC-requirement be violated for some excitation region $ER(*a_i)$ that corresponds to transition $*a_i$. Then, by inserting new signals, a persistent strongly correct STG G' equivalent to G can be derived, where this violation is eliminated and no new violations of the MC-requirement arise.*

V. Extending implementation structures

In this section, we show that even minor refinements of implementation structure may significantly improve the efficiency of the implementation.

A. Inverse feedback

All states of the excitation region $ER(+a_i)$ (or $ER(-a_i)$) belong to the *on-set* of the corresponding excitation function S_a (or R_a). The states of the quiescent region $QR(+a_i)$ (or $QR(-a_i)$) belong to the *dc-set* of the corresponding excitation function. According to MC-requirement, additional constraints exist for the use of the states from *dc-set* as the cover cube can change only once inside a quiescent region. One way to obtain a monotonous cover is to set the excitation function inside the excitation region and to reset it *immediately* after it. Note that, in all states of the excitation region $ER(*a_i)$, the value of signal a is inverse to its value in all states of the following quiescent region. For example, in the states of $ER(+a_i)$ signal a has value $a = 0$ and in all states of the quiescent region $QR(+a_i)$, $a = 1$. Signal a has a constant value in $ER(*a)$ and so can be added to the cover cube. This will restrict the cover only by the states from the excitation region. In such a case, the requirement for the cover cube to change only once inside the quiescent region is fulfilled automatically.

The use of the **self-dependent covers** of excitation regions allows softening the MC-requirements and reformulating their Definition III.2 in the following way: *A cover cube $a'c(*a_i)$, where $a' = a$ if $*a_i = -a_i$ and $a' = \overline{a}$ if $*a_i = +a_i$, is a monotonous cover for $ER(*a_i)$ if (1) $c(*a_i)$ covers all states of $ER(*a_i)$ and (2) $c(*a_i)$ can cover only states from $CFR(*a_i)$ and does not cover any reachable state from other $CFR(*a_j)$.*

At the implementation level, self-dependent cover involves an additional inverse feedback wire from the latch output to its corresponding signal network. Note that for a C-implementation this feedback does not require an additional inverter.

B. Poly-term covers

Let us extend the basis of standard gates for the implementation of SOP structure to complex gates (*AND-OR*). Since the basis is more powerful, the requirements for hazard-free implementation can be relaxed:

- *The persistency requirement is not necessary.* In the STG shown in Figure 1,a the excitation function of the non-persistent signal c can be implemented without hazards by a complex gate as $S_c = a + b\overline{d}$. No additional signals are needed.
- *The Unique Entry Condition is not necessary.* Several cover cubes corresponding to different minimal states of one excitation region can be combined together by one *AND-OR* gate.

However, the complex gate implementation has its own correctness criteria. It was shown in [7] that to ensure the hazard-free behavior of complex gate inside the excitation region it is necessary to satisfy the so-called *internal transition* condition, i.e. any transition within one excitation region is internal for at least one of the cover cubes of complex gate. The following definition summarizes the formal properties of the hazard-free implementation in the basis of complex gates.

Definition V.1 (Poly-term monotonous cover) *The union of cubes $C = \{c_1(*a_i) + \ldots + c_r(*a_i)\}$ is a* **poly-term monotonous cover** *for $ER(*a_i)$ if: (1) C covers all states in $ER(*a_i)$, (2) The logic function corresponding to C changes at most once in any trace of states inside $QR(*a_i)$, (3) C can cover only states from $CFR(*a_i)$ and does not cover any reachable state from other $CFR(*a_j)$, (4) Any signal transition $u \xrightarrow{*b} v$ inside $ER(*a_i)$ is an internal transition for at least one of the cubes from C, i.e., both u and v are covered by some $c_j \in C$.*

VI. Extending class of specifications

In this section, we expand the method of hazard-free synthesis to STGs without Unique Entry Condition and to specifications with OR-causality between signal transitions.

A. Revising the Unique Entry Condition

Up to now, we required the excitation regions of STG to satisfy the UEC requirement, i.e., to have only one minimal state. However this requirement is not satisfied for every correct STG. In Figure 5,a transition $+c$ is triggered either by $+a$ or by $+b$ and the corresponding excitation region $ER(+c)$ (Figure 5,b) contains two minimal states. For implementing such STG we need to soften the UEC requirement.

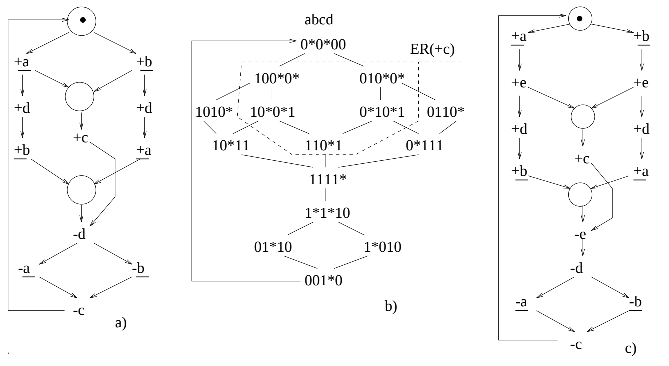

Fig. 5. STG (a) and TD (b) with violation of unique entry condition

Definition VI.1 (Generalized UEC) *An excitation region satisfies the* **generalized unique entry condition** *if each of its minimal states has the same set of trigger transitions.*

An excitation region satisfying the UEC condition with several minimal states, still can be covered with one cover cube. So the task of equivalent transformation is to reduce the original STG to such "good" cases. For our example of Figure 5, one can add a new signal, e.g. e, which triggers $+c$ in both alternative branches of the process (see Figure 5,c). This technique works generally, too [7]: *Any correct STG can be reduced to the MC form and thus can have a hazard-free implementation in the basis of simple gates.*

B. Change Diagrams for modeling processes with OR-causality

The STGs considered so far have only one type of causal relation between signal transitions. It is called AND-type as any transition can occur only if *all* of its direct predecessors have occurred. However, for the specification of processes in semimodular logic circuits, OR-causal relations between signal transitions are also necessary [6, 16]. Any circuit with an *OR*-gate (or, dually, *AND*-gate) has an OR-causal relation, if *concurrent* rising (dually, falling) transitions can occur at the inputs of the gate. In terms of TD the OR-causality corresponds to the notion of a detonant state. In [6] it was proved that *the TD derived from a correct STG is always distributive.* From this theorem follows that the semimodular but not distributive processes cannot be specified by STG. The latter is the reason why most of the synthesis methods impose the restriction of distributivity on the original processes [4, 1, 9]. To bridge this gap we will extend the STG model to the Change Diagram model with the two types of causal relations and will modify the methods of implementation to deal with the entire class of semimodular processes.

Unlike STGs, Change Diagrams (CDs) have two types of transitions: (1) the AND-transition that occurs only after *all* its direct predecessors occur; (2) the OR-transition that occurs after *at least one* of its direct predecessors occur.

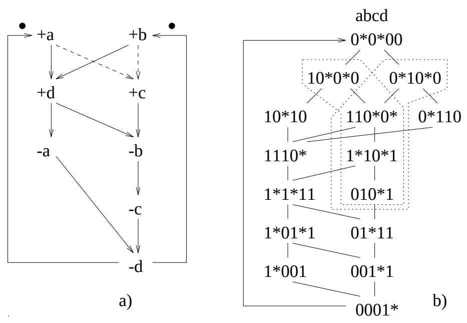

Fig. 6. Process with OR-causality

Definition VI.2 *CD G is a tuple* $\langle \mathcal{N}, A, \lambda, \mu \rangle$*, where* $\mathcal{N}$ *is a free-choice net, A is the set of signal changes,* $\lambda : T \rightarrow A$ *is the labeling function and* $\mu : T \rightarrow \{AND, OR\}$ *is the function that defines the type of transition.*

An example of CD is shown in Figure 6,a. The places that have only one predecessor and successor are omitted. The only one OR-transition in CD Figure 6,a is $+c$ (denoted by dashed input arcs). It has two OR-causes $+a$ and $+b$ and in the corresponding TD Figure 6,b the initial state 0*0*00 is detonant and $ER(+c)$ has two minimal states: 10*0*0 and 0*10*0. Thus, CDs have more descriptive power than free-choice STGs and as was shown in [6] *the TD derived from a correct CD is semimodular.*

The idea of a simple gate implementation for processes with OR-causality is straightforward. If each OR-transition c in CD has only two triggering transitions a and b and the detonant excitation region $ER(c)$ can be covered by the sum of two cubes a and b (*OR-monotonous condition*), then this excitation region can be implemented in the signal network by simple OR-gate. An efficient technique for ensuring the OR-monotonous condition in any CD by adding extra signals is presented in [7]. Therefore, *any CD allows the hazard-free implementation under the generalized C- or RS-architectures.*

VII. Experimental results

The proposed approach has been tested on a set of benchmarks known from [9]. The CAD tool "FORCAGE" [6] was used to derive the excitation functions and check the implementations with respect to their freedom from hazards.

We will give the comparison of our solutions to the speed-independent implementations obtained by [1] and to the hazard-free implementations under bounded gate delays[9] using SIS library of simple gates. The results are shown in Table I. The column labeled "Trans./states" shows the number of signal transitions in the initial STG specification and the number of states in the corresponding transition diagram. Columns labeled "Area" give the total area (excluding routing) of each circuit, using a "generic" standard cell library from SIS. Columns labeled "Del" give the maximum delay inside *one* signal network based on SIS conventions for delay estimate. Although this method does not allow one to observe the actual cycle time of the circuit, we had to choose it in order to be compatible with the delay estimate for Stanford's and SIS methods. Columns labeled "SIS" and "Stanford" are directly borrowed from [1]. The numbers for SIS were obtained in [1] under the assumption of the fixed delay model (when the lower and upper bounds for gate and wire delays coincide) with an optimization script optimizing for area. The last column shows the best (with respect to area) between standard C- and RS-implementations based on monotonous cover technique. We summarize the experimental results in Table II.

These results show that, in comparison with SIS implementation, our area optimized standard implementation on the average reduces the area by about 10% and increases the speed of the circuit by about 34%. SIS needs to pad extra delay lines to ensure hazard-freedom. These delays represent a significant fraction of the overall delay through the circuit. On the opposite, speed-independent standard implemenetations provide hazard-

TABLE I
Experimental results

Circuit	*trans./ states*	SIS Area/Del	Stanford Area/Del	MC-opt. Area/Del
chu133	14/24	352/5.2	240/4.8	216/4.8
chu150	14/26	232/7.0	240/4.8	232/4.8
chu172	13/12	104/1.6	152/3.6	112/2.4
converta	14/18	432/6.8	520/6.0	320/4.8
ebergen	14/18	280/5.6	344/4.8	280/4.8
full	8/16	224/5.2	112/2.4	112/2.4
hazard	10/12	296/6.6	256/4.8	224/3.6
hybridf	16/80	274/6.6	152/2.4	152/2.4
nowick	16/20	264/6.6	504/6.0	376/3.6
pe-send-ifc	53/117	1232/12.2	2072/7.2	1480/4.8
qr42	14/18	280/5.6	344/4.8	280/4.8
vbe10b	22/256	1008/10.0	800/6.0	640/4.8
vbe5b	12/24	272/4.2	264/4.8	192/3.6
vbe5c	12/24	224/5.2	264/4.8	200/3.6
wrdatab	24/216	824/7.0	840/4.8	744/4.8
var1	8/12	184/3.0	256/5.0	240/3.8
xyz	6/8	120/3.2	200/4.8	160/3.6
totals		6602/102	7560/82	5960/67.40

TABLE II
Comparison with SIS tool and Stanford's tool

Parameter	SIS	Stanford	MC-opt.
Area	1	1.15	0.90
Delay	1	0.80	0.66

freedom by construction. This is the reason for our speed advantage in comparison with SIS. In comparison with Stanford's method, our optimized implementation on the average reduces the area by about 21% and increases the speed of the circuit by about 18%.

Acknowledgements

We are grateful to Bill Lin and Peter Vanbekbergen for numerous discussions that have led to the creation of Monotonous Cover Conditions and for our common work under implementation of these conditions in the satisfiability framework [8]. We thank Luciano Lavagno for many useful comments and assistance with the SIS tool and Peter Beerel for providing us with detailed information on his benchmark.

References

[1] P. Beerel and T. Meng. Automatic gate-level synthesis of speed-independent circuits. In *Proceedings of ICCD-92*, pages 581–586. IEEE Computer Society Press, 1987.

[2] T.-A. Chu. Synthesis of self-timed VLSI circuits from graph-theoretic specifications. In *Proceedings of ICCD-87*, pages 220–223. IEEE Computer Society Press, 1987.

[3] T.-A. Chu. *Synthesis of Self-Timed VLSI Circuits from Graph-Theoretic Specifications.* PhD thesis, MIT Laboratory for Computer Science, June 1987.

[4] T.-A. Chu. Synthesis of hazard-free control circuits from asynchronous finite state machine specifications. In *Proceedings of the ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, Princeton, March 1992.

[5] D.L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits.* The MIT Press, Cambridge, Mass., 1988. An ACM Distinguished Dissertation 1988.

[6] M. Kishinevsky, A. Kondratyev, A. Taubin, and V. Varshavsky. *Concurrent Hardware: The Theory and Practice of Self-Timed Design.* John Wiley and Sons, London, 1993.

[7] A Kondratyev, Kishinevsky M., and Yakovlev A. Monotonous cover transformations for speed-independent implementation of asynchronous circuits. Technical report, TR-Aizu University, March 1994.

[8] A Kondratyev, Kishinevsky M., Lin B., Vanbekbergen B., and Yakovlev A. Basic gate implementation of speed-independent circuits. In *Proceedings of 31th Design Automation Conference*, San Diego, June 1994.

[9] Luciano Lavagno, Kurt Keutzer, and Alberto Sangiovanni-Vincentelli. Algorithms for synthesis of hazard-free asynchronous circuits. In *Proc. Design Automation Conf.*, pages 302–308, 1991.

[10] Luciano Lavagno, Cho Moon, Robert Brayton, and Alberto Sangiovanni-Vincentelli. Solving the state assignment problem for signal transition graphs. In *Proc. Design Automation Conf.*, pages 568–572, 1992.

[11] Cho W. Moon, Paul R. Stephan, and Robert K. Brayton. Synthesis of hazard-free asynchronous circuits from graphical specifications. In *Proceedings of ICCAD-91*, pages 322–325. IEEE Computer Society Press, November 1991.

[12] D. Muller and W. Bartky. A theory of asynchronous circuits. In *Annals of Computation Laboratory*, pages 204–243. Harvard University, 1959.

[13] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of IEEE*, 77(4):541–580, April 1989.

[14] P. Vanbekbergen, B. Lin, G. Goossens, and H. De Man. A generalized state assignment theory for transformations on signal transition graphs. In *Proceedings of ICCD'92*, Cambridge, MA, October 1992.

[15] V. Varshavsky, M. Kishinevsky, V. Marakhovsky, V. Peschansky, L. Rosenblum, A. Taubin, and B. Tzirlin. *Self-Timed Control of Concurrent Processes.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990. V.I. Varshavsky, Ed.

[16] A. Yakovlev, M. Kishinevsky, A. Kondratyev, and L. Lavagno. OR causality: modelling and hardware implementation. In *Proceedings of the 15th International Conference on Application and Theory of Petri Nets*, Zaragosa, Spain, June 1994. to appear.

[17] M. Yu and P. Subrahmanyam. Hazard-free asynchronous circuit synthesis. In *Proceedings of Working Conference on Asynchronous Design Methodologies*, Manchester, March 1993.