Avoiding Hazards in Self-timed Digital Circuits Derived from Signal Transition Graphs[†]

Edwin C.Y. CHUNG and Lindsay KLEEMAN

Department of Electrical and Computer Systems Engineering
Monash University
Clayton, Victoria 3168, Australia
Fax: +61 3 90 53453

Abstract—Since the introduction of Signal Transition Graphs (STGs) in the mid 1980s [1, 2], a number of techniques for the synthesis of self-timed circuits using STGs have been proposed. To achieve a hazard-free implementation, restrictions on the structure of the STG have been employed. Also, hazard-free design techniques have been incorporated into the synthesis procedure. Despite these, implementations derived using these techniques are not always hazard-free. Hazards are shown in this paper to be intrinsic to the function being implemented and cannot be eliminated. To avoid these hazards, certain timing conditions must be preserved.

Previous attempts [3, 4, 5] to eliminate hazards are shown to have important limitations. A new procedure is proposed in this paper for the detection of hazards and timing constraints to avoid these hazards. The procedure is compared with previous attempts at hazard detection, and examples presented to show the limitations of other approaches.

Keywords—Asynchronous, Self-timed digital circuits, Signal Transition Graphs, Hazard-free.

[†] The work described in this paper is supported in part by the Australian Telecommunication and Electronics Research Board.

GLOSSARY OF SYMBOLS AND NOTATION

$C \cap \psi$	The intersection of a cube C with a cover ψ ($C \cap \psi = \bigcup_i \{C \cap C_i \mid C_i \in \psi\}$).
$\lvert\sigma\rvert$	The length of sequence σ (i.e., the number of transitions in σ).
$\sigma \restriction T'$	The *projection* of transition sequence $\sigma \in T^*$ onto the set of transitions T'. In other words, the removal of all transitions $\tau \notin T'$ from sequence σ.
$s[\sigma\rangle$	Sequence $\sigma \in T^*$ is enabled in state $s \in S$.
$s[\sigma\rangle s'$	Transition sequence $\sigma \in T^*$ is enabled in state $s \in S$ and the firing of sequence σ from state s will lead to state $s' \in S$. In this paper, it is also taken to denote the set of states along the path $s[\sigma\rangle s'$ from s to s' including states s and s'.
$s[\tau\rangle$	Transition $\tau \in T$ is enabled in state $s \in S$.
$s[\tau\rangle s'$	Transition $\tau \in T$ is enabled in state $s \in S$ and the firing of transition τ from state s will lead to state $s' \in S$.
δ	*Transition function* $\delta\colon S \times T \rightarrow S$ such that $\langle s, t\rangle \in \mathrm{dom}(\delta)$ if and only if transition t is enabled in state s. This definition is usually extended to the mapping $\delta\colon S \times T^* \rightarrow S$. Likewise, $\langle s, \sigma\rangle \in \mathrm{dom}(\delta)$ if and only if sequence σ is enabled in state s.
$\Phi_J = \langle S, T, \delta, s_0\rangle$	A state graph description.
$\Sigma_J = \langle P, T, F, M_0\rangle$	An STG description.
$\mathrm{f}(s, x)$	The *implied* or *next state value* [2, 3, 10] of signal x in state s and is defined as: $\mathrm{f}(s,x) = \begin{cases} \mathsf{X}\ \text{(don' t care)} & \text{if } s \notin S \\ s'(x) & \text{if } \exists \text{ a state } s' \text{ such that } \delta(s, x\pm) = s' \\ s(x) & \text{otherwise} \end{cases}$ where $x\pm$ denotes either an $x+$ or $x-$ and $s(x)$ denotes the logical state of signal x in state s.
F	*Flow relation* ($F \subseteq (P \times T) \cup (T \times P)$ such that $\mathrm{dom}(F) = \mathrm{range}(F) = P \cup T$).
J	The set of *network signals* described in Φ and Σ ($J = J_\mathrm{I} \cup J_\mathrm{NI}$).
J_I	The set of *input signals*.
J_N	The set of *internal signals* (not input nor output).
J_NI	The set of *non-input* (output and internal) *signals* ($J_\mathrm{NI} = J_\mathrm{O} \cup J_\mathrm{N}$).
J_O	The set of *output signals*.
M_0	The *initial marking* on Σ_J.
P	The set of *places* in Σ_J.
s_0	The *initial state* in Φ_J corresponding to token marking M_0 in Σ_J.
S	The set of *states* in the Φ_J.
T	The set of *signal transitions* in Σ_J ($T = T_\mathrm{I} \cup T_\mathrm{NI} = J \times \{+, -\}$).
T_I	The set of *input signal transitions* ($T_\mathrm{I} = J_\mathrm{I} \times \{+, -\}$).
T_N	The set of *internal signal transitions* ($T_\mathrm{N} = J_\mathrm{N} \times \{+, -\}$).
T_NI	The set of *non-input signal transitions* ($T_\mathrm{NI} = J_\mathrm{NI} \times \{+, -\}$).
T_O	The set of *output signal transitions* ($T_\mathrm{O} = J_\mathrm{O} \times \{+, -\}$).
T^*	The set of all finite length sequences of symbols in T ($T^* = \{\langle \tau_i : i \in \mathbb{N}\rangle \mid \tau_i \in T\}$).

1. Introduction

Signal Transition Graphs (STGs) and their application to the synthesis of self-timed circuits were first introduced by Chu in the mid 1980s [1, 2, 3, 4]. Since then, a number of techniques for the synthesis of self-timed circuits using STGs have been proposed [5, 6, 7, 8]. Though diverse in many aspects, the synthesis procedures utilised in these techniques conform to the following structure, first introduced by Molnar *et al.* [9] for the synthesis of *Petri Nets*.

Step 1:	State graph derivation.
Step 2:	Karnaugh map generation for non-input signals.
Step 3:	Derivation of logic expressions from the Karnaugh maps.

In an attempt to achieve a hazard-free implementation, restrictions on the structure of STGs have been employed [3, 4, 6]. Hazard-free design techniques also have been incorporated into the synthesis procedure [7, 10]. Despite these measures however, implementations derived using this procedure can still contain hazards. These hazards are shown in this paper to be intrinsic to the function implemented and cannot be eliminated. Hence, to avoid these hazards, critical timing conditions necessary to ensure a hazard-free operation must be preserved.

These hazards are first described by Lavagno *et al.* in [5] where they are attributed to the switching of active prime implicants or implicates. A procedure that detects some, but not all hazards and their related timing conditions is proposed in [5].

In this paper, hazard conditions in implemented circuits are analysed. Other approaches to achieve hazard-free implementations are also examined. As we will show in this paper, implementations derived using these approaches can still contain hazards. A new algorithm is proposed that detects these hazards and determines critical timing which must be preserved to avoid these hazards.

The paper is organised as follows. In section 2, the technique for synthesis of self-timed circuits from STGs is reviewed and important terminology introduced. The properties of speed independence, semi-modularity and persistency are examined in Section 3 with regard to circuit hazards and examples are presented to clarify the concepts. These examples motivate the algorithms for detecting and avoiding hazards that are presented in Section 4. Our approach is compared with other techniques for hazard detection and avoidance in Section 5, and conclusions are given in Section 6.

2. Synthesis of Circuits from STGs

In this section, an overview of the synthesis of self-timed circuits from STGs is presented.

2.1. Signal Transition Graphs

A Signal Transition Graph (STG) is a form of *Petri Net* with properties referred to in the Petri Net literature as *live*, *safe* and *free choice* [3, 11, 12]. *Transitions* in these graphs are interpreted as *signal transitions* and are denoted by the name of a signal suffixed with a '+' or a '−' to express a rising or a falling transition of the signal respectively. To facilitate implementation, transitions of input signals are underlined to differentiate them from transitions of non-input (internal and output) signals. Similar to Petri Nets, *tokens* and *places* in STGs are denoted as dots and circles respectively. Places with only one input and one output transition are not represented in STGs for reasons of conciseness.

As an example, consider the STG description of a Muller-C element in Fig. 1b. Note that unlike its Petri Net equivalent in Fig. 1c, places in the STG are only represented implicitly by the directed arcs in the description. Bars denoting transitions in the STG are also omitted leaving behind a compact description of the Muller C-element. Using *firing rules* similar to those in Petri Net, sequences of signal transitions described by the STG can be determined by continually firing the enabled transitions (i.e., transitions with all input places filled with tokens). Alternatively, the STG may also be interpreted using the causal relations expressed. In other words, interpreting each "$\tau_1 \rightarrow \tau_2$" in the STG as the *causal relation* "transition τ_1 causes or is followed by transition τ_2".

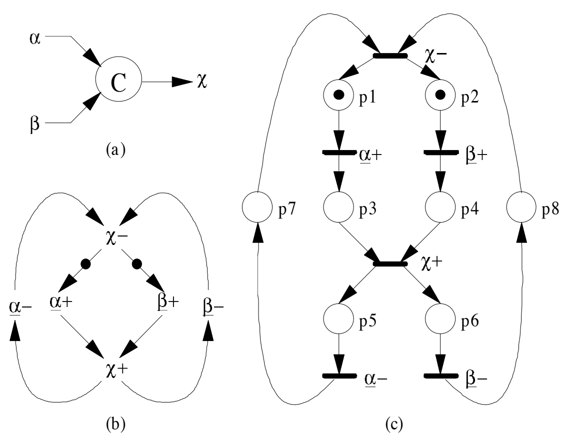


Figure 1: *(a) The block description of a Muller-C element, (b) its STG description and (c) its Petri Net equivalent.*

In addition to the graphical representation, an STG is denoted formally as a 4-tuple, $\Sigma_J = \langle P, T, F, M_0 \rangle$, where

> J is the set of *network signals* described in Σ,
> P, the set of *places* in the STG,
> T, the set of all *signal transitions* ($T = J \times \{+,-\}$),
> F, the *flow relation* ($F \subseteq (P \times T) \cup (T \times P)$ such that $\mathrm{dom}(F) = \mathrm{range}(F) = P \cup T$) and
> M_0 is the *initial token marking* on the STG.

For more details on the syntax and semantics of Signal Transition Graphs, interested readers should consult [2, 3, 4].

2.2. State Graphs

Another graphical description of circuit behaviour used during the synthesis procedure is the *state graph* description. In brief, a *state graph* is the *reachability graph* of an STG. However, unlike its Petri Net equivalent, *states* in state graphs are denoted by binary vectors obtained by putting the logical states of all signals in J in a predefined sequence. Directed arcs between states in the state graph are labelled with the corresponding signal transitions causing the state transitions.

Fig. 2 shows the state graph description of the STG in Fig. 1b. States are represented by binary vectors $\langle \alpha, \beta, \chi \rangle$ and the initial state corresponding to the initial marking of the STG is shown with symbol '•'.

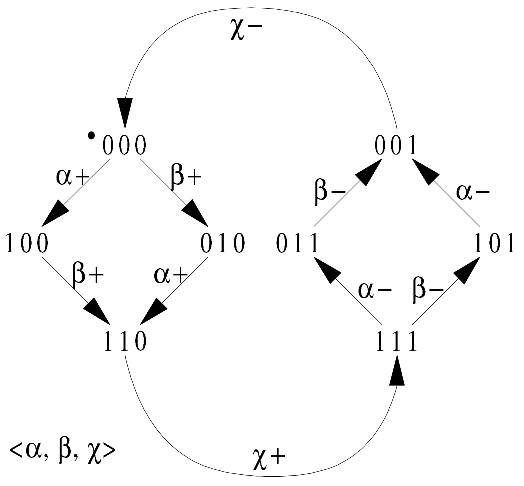


Figure 2: *The state graph description of the Muller-C element depicted in Fig. 1.*

Similar to STGs, a state graph description is denoted formally as a 4-tuple, $\Phi_J = \langle S, T, \delta, s_0 \rangle$, where

J is the set of *network signals* specified in Φ and Σ,
S, the set of all *states* in the state graph,
T, the set of all *signal transitions* ($T = J \times \{+, -\}$),
s_0, the *initial state* on the state graph corresponding to M_0 on the STG and
δ, the *transition function* δ: $S \times T \rightarrow S$ such that $\langle s, t \rangle \in \mathrm{dom}(\delta)$ if and only if transition t is enabled in state s.

2.3. Representation of Subspaces or Cubes in Karnaugh Maps

Having derived the state graph description, Karnaugh maps for non-input signals are next generated. For a non-input signal, $x \in J$, its Karnaugh map is obtained by filling each cell in the Karnaugh map with the implied value, $\mathrm{f}(s, x)$, of signal x in the corresponding state, s, in the state graph [2, 4, 10]. The logic expression for signal x is then derived from the Karnaugh map [10].

In this paper, an n literal *Karnaugh map* is denoted by an *n-space*. *Prime implicants* and *implicates* derived from the Karnaugh map are denoted by cubes or *p-subspaces* within the n-space. *n-Space* and *p-subspaces* are formally defined as follows:

Definition 1: *n-Space*
A multi-dimensional Boolean space of n dimensions with discrete coordinate values of 0 and 1 on each of its dimensions is called an n-space.

□

Definition 2: *p-Subspace*

A multi-dimensional Boolean space of p dimensions within an n-space ($p \le n$) is called a p-subspace.

□

Using a ternary notation of '0', '1' and '–' for *logic values* and *undefined* respectively, cubes within a Karnaugh map are represented in the same way as prime implicants and implicates in the Quine-McCluskey tabular Karnaugh mapping [13, 14]. As an example, consider the 4 dimensional Karnaugh map in Fig. 3. Note that a cube in the Karnaugh map is denoted using this notation as ––1– where the value of literal c remains at logic 1 and values of literals a, b and d are variable.

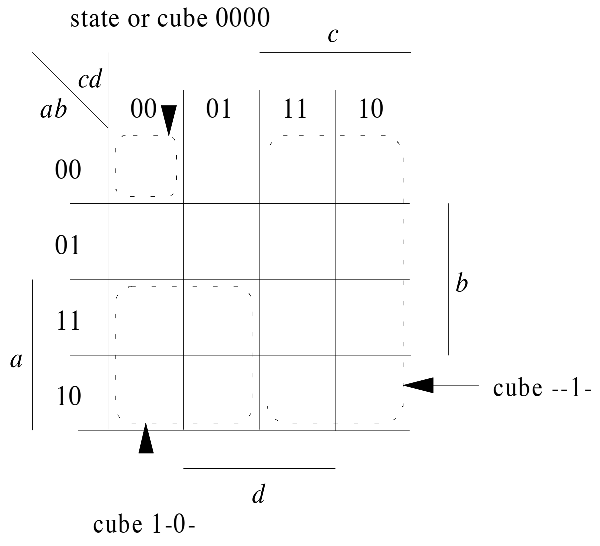


Figure 3: *Ternary representation of cubes within a Karnaugh map.*

In this paper, a set of cubes within a Karnaugh map is called a *cover*. The intersection of a cube C and a cover ψ is defined to be the cover $C \cap \psi = \bigcup_i \{C \cap C_i \mid C_i \in \psi\}$. Refer to the Appendix for more details.

3. Circuit Hazards

Unlike synchronous circuits where signals are only sampled and updated at periodic intervals, a self-timed circuit constantly monitors the state of its signals and reacts to changes almost instantaneously. Consequently, spurious transient changes or *hazards* in a self-timed circuit can result in malfunctions. The success of a self-timed implementation technique relies on its ability to eliminate and/or avoid circuit hazards.

In this section, circuit properties and restrictions commonly used to achieve hazard-free implementations are examined. Hazard conditions that cannot be eliminated using these properties and restrictions are analysed. An algorithm is presented that detects these conditions and determines the critical timing constraints necessary to avoid these hazards.

3.1. Speed-independence

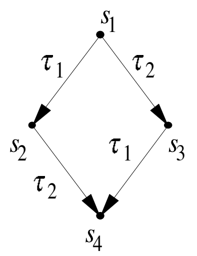
An important circuit property in the implementation of self-timed circuits is the property of *speed-independence* [15, 16]. Informally, *speed-independence* refers to circuits that operate independently of the delays in logic elements of the circuit. Frequently utilised as a means to ensure hazard-free implementations, speed-independence is derived through the property of *semi-modularity* defined as follows:

Definition 3: *Semi-modularity* [15, 16]
A circuit is said to be *semi-modular* if its state graph description, $\Phi_J = \langle S, T, \delta, s_0 \rangle$, satisfies

$$\forall\, s_1, s_2, s_3 \in S;\ \forall\, \tau_1, \tau_2 \in T\ \ (s_1[\tau_1\rangle s_2 \wedge s_1[\tau_2\rangle s_3) \Rightarrow (\exists\, s_4 \in S : s_2[\tau_2\rangle s_4 \wedge s_3[\tau_1\rangle s_4).$$

The notation, $s_i[\tau_j\rangle s_k$, denotes that transition τ_j is enabled in state s_i and that the firing of τ_j from s_i will lead to state s_k. In other words, if transitions τ_1 and τ_2 are enabled in state s_1, the firing of τ_1 should not disable the firing of τ_2 and vice versa forming a "semi-modular structure" in the state graph as depicted in Fig. 4. In [15, 16], Muller proved that a *semi-modular* circuit is also *speed-independent*.

□



$$\begin{tikzcd}[column sep=small] & s_1 \arrow[dl, "\tau_1"'] \arrow[dr, "\tau_2"] & \\ s_2 \arrow[dr, "\tau_2"'] & & s_3 \arrow[dl, "\tau_1"] \\ & s_4 & \end{tikzcd}$$

Figure 4: *The semi-modular structure corresponding to the concurrent firing of transitions* τ_1 *and* τ_2 *from state* s_1.

Though semi-modularity is sufficient to ensure speed-independence as defined by Muller [15, 16], it cannot ensure hazard-free self-timed implementations. To clarify this, consider the semi-modular implementation $c = a + b{\cdot}d$ for the STG in Fig. 5a. In particular, consider the firing of transitions $b+$ followed by $a-$ while the circuit is in the initial state where

$a = c = d = 1$ and $b = 0$ (cell marked '•' in Fig. 5b). Notice that even when c is implemented from a semi-modular state graph description (superimposed on the Karnaugh map in Fig. 5b), a hazard can occur at the output c depending upon the delay between the firing of $b+$ and $a-$, and the propagation delay through the AND and the OR gate implementing signal c as follows:

> Firstly, let the delay between the firing of $b+$ and $a-$ be denoted as Δ and the propagation delay from b to c and from a to c be denoted as $\Delta 1$ and $\Delta 2$ respectively. Starting at the initial state, note that signal c is being held high by signal a being high. The firing of $b+$ from this state will cause prime implicant $b \cdot d$ (i.e., the output of the AND gate) to become high, subsequently holding signal c high. Note however that the firing of $b+$ will also lead to the firing of $a-$ terminating the effect of "a high holding signal c high". Hence, on a normal operation, the effect of "$b \cdot d$ high holding signal c high" is expected to occur before the firing of $a-$ relinquishes the effect of signal a on signal c. In other words, $\Delta + \Delta 2$ is expected to be greater than $\Delta 1$. As illustrated in Fig. 5d a hazard will appear at output c if $\Delta + \Delta 2 < \Delta 1$. Therefore, the implementation is **not** hazard-free, nor is it speed-independent as characterised earlier.

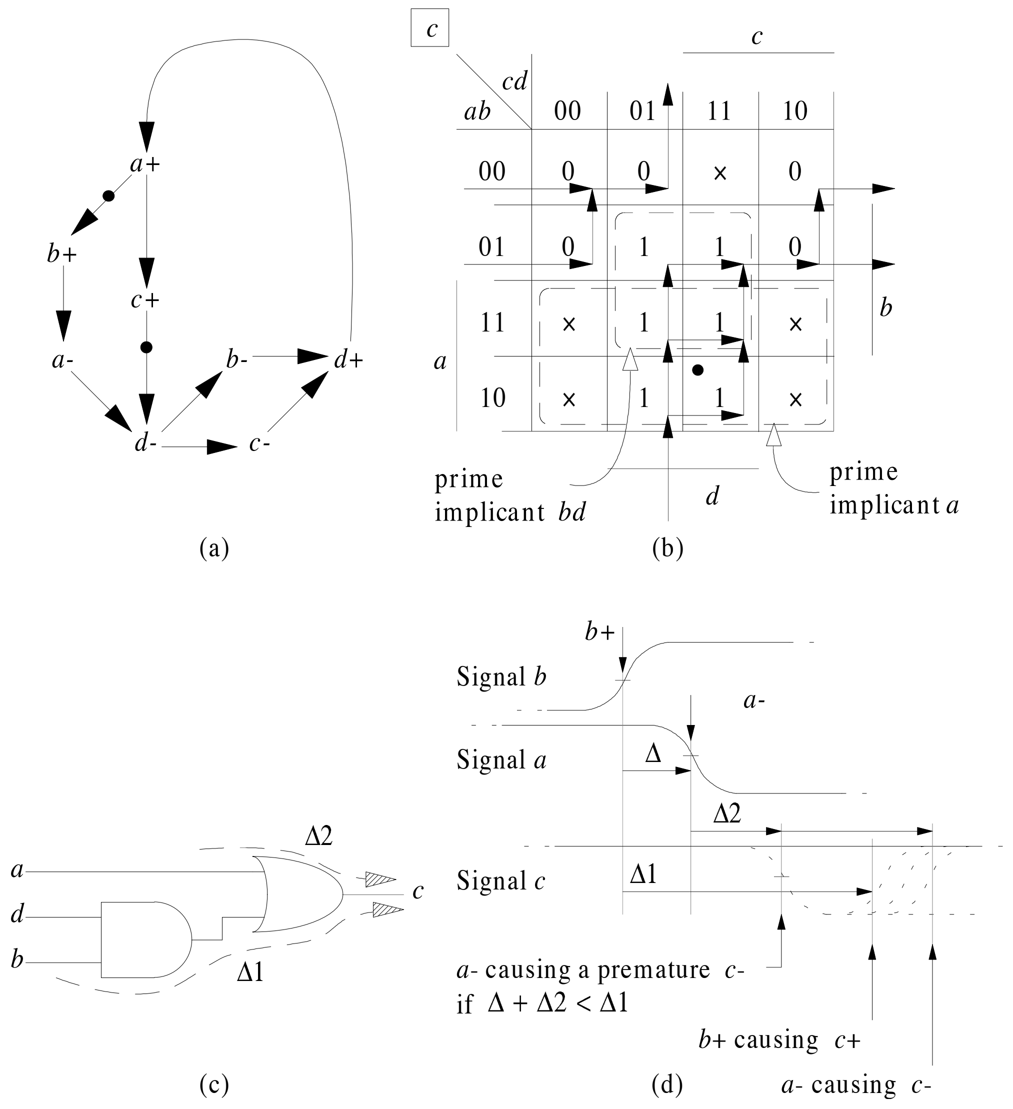


Figure 5: *(a) An STG with (b) its state graph superimposed on the Karnaugh map for signal c. (c) The implementation of signal c in the SOP form and (d) the timing diagram analysing the firing of* b+ *followed by a– and its effect on signal c.*

The cause of this hazard is attributed to the switching of active prime implicants in [5]. In our example, from *a* to *b·d* during the firing of *b*+ followed by *a*– as follows:

> Denoting the state of the circuit as a binary vector $\langle a\,b\,c\,d\rangle$, the firing of *b*+ followed by *a*– from state 1011 can be interpreted as a transition from state 1011 to state 1111 followed by a transition to state 0111. As illustrated in the Karnaugh map in Fig. 5b, notice that prime implicant *a* is active when the circuit is in states 1011 and 1111, while prime implicant *b·d* is active when the circuit is in states 1111 and 0111. In other words, the active prime implicants holding signal *c* high while the circuit traverses from state 1011 through to state 0111 switches from *a* to *b·d* and a hazard will appear at

output c if prime implicant $b{\cdot}d$ is not active and holding signal c high before prime implicant a is deactivated and relinquishes its effect on output c.

Though the cause of this hazard can be attributed to the switching of active prime implicant (implicate), one should not be tempted into thinking that an implementation in the SOP (POS) form which does not involve any switching of prime implicants (implicates) will be hazard-free. The reason for this is that an implementation in the SOP (POS) form not involving any switching of prime implicant (implicate) can also be equivalent to an implementation in the POS (SOP) form which may entail the switching of active prime implicates (implicants). As such, these implementations may still contain hazards. To illustrate this, consider the implementation $c = \overline{d}{\cdot}a$ for the STG in Fig. 6a. Note that $c = \overline{d}{\cdot}a$ can be taken as a single prime implicant implementation, and hence will not involve any switching of prime implicants, or an implementation in the POS form consisting of prime implicates $\overline{d}$ and a as illustrated in Fig. 6b. Similar to the example in Fig. 5, a hazard can occur at output c when the circuit traverses from state 0000 to state 1001 depending upon the delay between the occurrence of $d+$ and $a+$ and the propagation delay through the logic elements implementing signal c as illustrated in Fig. 6d.

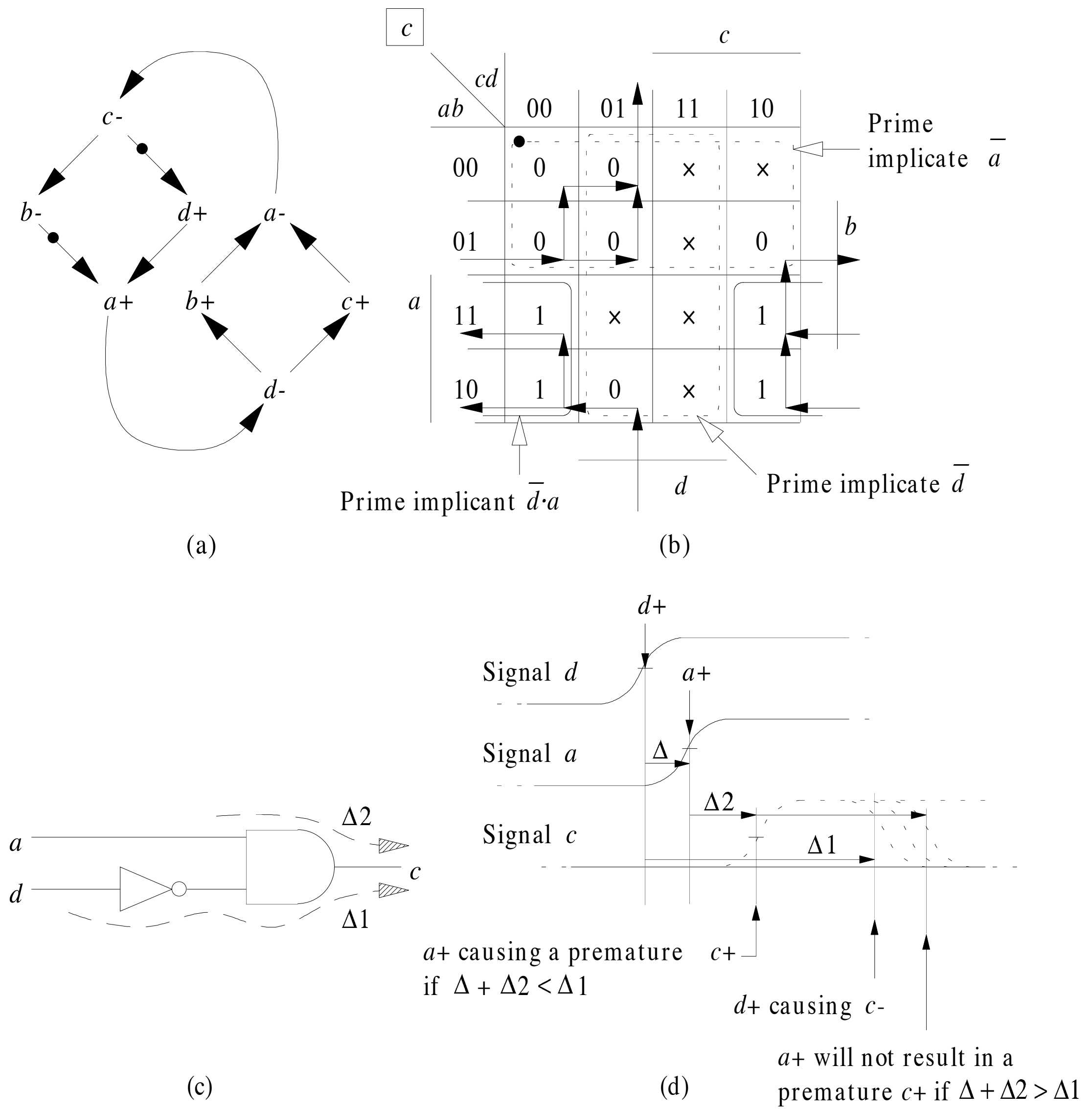

c-
b-
d+
a-
a+
b+
c+
d-
(a)
c
cd
ab
00
01
11
10
c
Prime implicate $\overline{a}$
b
a
Prime implicant $\overline{d}$·a
d
Prime implicate $\overline{d}$
(b)
a
d
Δ2
c
Δ1
(c)
d+
Signal d
a+
Signal a
Δ
Δ2
Signal c
Δ1
a+ causing a premature if Δ + Δ2 < Δ1
c+
d+ causing c-
a+ will not result in a premature c+ if Δ + Δ2 > Δ1
(d)

Figure 6: *(a) An STG and (b) its semi-modular state graph description superimposed on the Karnaugh map of signal c. (c) The implementation c = $\overline{d}$·a can be viewed as a single prime implicant POS form or a double prime implicates SOP form and (d) the timing diagram analysis the condition which can results in a hazard.*

Additionally, notice that wire or stray delays deferring the arrival of transition *b*+ at the input of the AND gate in the example in Fig. 5 will have an equivalent effect of decreasing the effective value of Δ or increasing the effective value of Δ1, worsening the situation. Likewise, wire or stray delays deferring the arrival of *a*+ at the input of the AND gate in the example in Fig. 6. As such, the effect of wire delays can aggravate or even be the cause of circuit hazards and cannot be ignored.

From the above examples, it is concluded that semi-modularity is insufficient to ensure hazard-free speed-independence in self-timed circuits implemented from well-formed STGs with unique state coding. However, semi-modular circuits do not rely upon the ordering of concurrent signal changes, and thus are free from *critical races* and *essential hazards*. These hazard conditions are defined as follows:

Definition 4: *Races* [17, 18]
A transition from one state to another involving the change of more than one state variable (feedback signals) is known as a *race*. Races where the ordering in the changes in these state variables can lead to a different final state is known as *critical race*.

□

Definition 5: *Essential Hazard* [17, 18, 19]
An *essential hazard* is a critical race between an input and a state variable.

□

3.2. Persistency

Another property that is used to achieve hazard-free speed-independence is the property of *persistency* [3, 4]. Introduced by Chu, *persistency* means that an enabled transition cannot be disabled by the occurrence of some other transition and is defined using state graph terminology as follows:

Definition 6: *Persistency* (summarised from Definition 5.3 pp.86 [3])
A state graph description, $\Phi_J = \langle S, T, \delta, s_0 \rangle$, is said to be *persistent* if and only if

$$\forall\, \tau \in T_{\mathrm{NI}} \qquad (\exists\, s, s' \in S, \tau' \in T : s[\tau\rangle \wedge s[\tau'\rangle s') \Rightarrow s'[\tau\rangle$$

□

Similarly, note that it can be stated as $\forall\, \tau \in T_{\mathrm{NI}}\ (\exists\, s, s'' \in S, \tau' \in T : s[\tau\rangle s'' \wedge s[\tau'\rangle) \Rightarrow s''[\tau'\rangle$. In other words, $\forall\, \tau \in T_{\mathrm{NI}}\ (\exists\, s, s', s'' \in S, \tau' \in T : s[\tau\rangle s'' \wedge s[\tau'\rangle s') \Rightarrow (s''[\tau'\rangle \wedge s'[\tau\rangle)$ which is equivalent to the definition of semi-modularity defined above [15 p.208, 16 p.213-214]. This equivalence between the two definitions is lost however when Chu characterised persistency in STGs in the following manner [2, 3]:

A transition $\mu \in T_{\mathrm{NI}}$ is non-persistent if there exists a transition $\tau \in T$ such that τ enables μ and the firing of $\overline{\tau}$ (the reverse transition of τ) and μ are concurrent

To illustrate this, consider the STG in Fig. 7a. Using Chu's characterisation of persistency in STGs, output c in the STG would be considered as non-persistent since $a+$ □ $c+$ (centre top Fig. 7a) and the firing of $c+$ and $a-$ are concurrent (lower left portion of the STG in Fig. 7a).

However, since the state graph description of the STG (depicted in Fig. 7b) is semi-modular, it is also persistent as defined in Definition 5 above! More importantly however, as illustrated in the example in Fig. 6, persistency as defined or characterised by Chu cannot ensure a hazard-free implementation since the STG in Fig. 6 is persistent and yet its implementation has a hazard. Despite this, as we will show later, non-persistent STGs as characterised by Chu do lead to hazard conditions in the implemented circuits.

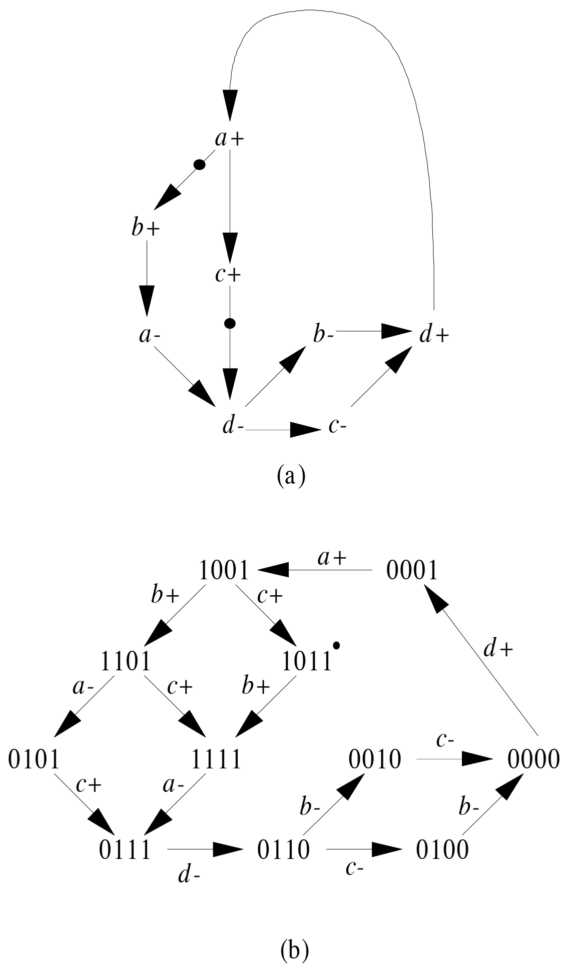


Figure 7: *(a) An STG description with a non-persistent signal c and (b) its equivalent semi-modular state graph*

4. Detecting and Avoiding Hazards in Self-timed Circuits

The circuit hazards in examples of Fig. 5 and 6 have been explained by the switching of prime implicants and implicates. An alternative explanation is now proposed that leads to an efficient approach to detect these hazards. These hazards are explained in terms of *function hazards* as defined below.

Definition 7: *Function hazard* [13, 14, 19]

A combinational circuit implementing a Boolean function, f, is said to contain a *function hazard* for a p-variable input change from $A = (a_1,\ldots a_p, a_{p+1},\ldots a_n)$ to $B = (\overline{a}_1,\ldots \overline{a}_p, a_{p+1},\ldots a_n)$ if

(1) $f(A) = f(B)$ and

(2) both 1s and 0s are specified for function f within the Karnaugh map sub-cube $(a_{p+1},\ldots a_n)$

In other words, the output before and after the input change are identical and there exists a path from A to B corresponding to the order of these input changes such that f changes.

□

To clarify this, consider again the firing of $b+$ □ $a-$ in Fig. 5. Note that a long propagation delay through the AND gate deactivating prime implicant a before activating prime implicant $b{\cdot}d$ is equivalent to the firing of $a-$ □ $b+$ (i.e., the firing of $b+$ □ $a-$ perceived in the reverse order). This corresponds to a traversal from state 1011 to state 0111 via state 0011 instead of state 1111. Since signal c has different implied values in states 0011 and 1111, a hazard is produced. Likewise, due to wire delays, the reversal of transitions $d+$ and $a+$ as perceived at the input of the AND gate during the firing of $d+$ □ $a+$ in Fig. 6 will also result in a hazard. Similarly, this is equivalent to a traversal from state 0000 to 1001 via state 1000 instead of 0001. In short, due either to propagation and/or interconnection delays, a combination circuit may perceive itself to be in a different state from the actual internal state. This will result in a hazard if the combinational circuit produces a different output at the altered state than it otherwise would.

In formal terms, these hazard conditions can be summarised in the following theorems.

Theorem 1: *The smallest cube, C_p, which contains the transition $s_1[\sigma\rangle s_2$ from state s_1 to s_2 and all permutations of $\sigma \in T^*$ is one such that*

$$C_p(i) = \begin{cases} s_1(i) & \textit{if transition } i\pm \notin \sigma \\ -\ (\textit{undefined}) & \textit{otherwise} \end{cases}$$

where $s_1[\sigma\rangle s_2$ denotes the path from s_1 to s_2 associated with the firing of transition sequence σ from state s_1, while $C_p(i)$ and $s_1(i)$ denote the logical state of signal i in C_p and s_1.

Proof:

For some $s_1, s_2 \in S$ and $\sigma \in T^*$, the path $s_1[\sigma\rangle s_2$ and its permutations will traverse from s_1 to s_2 over a region where all signal $j \in J : j\pm \in \sigma$ changes during the transition. Hence, the smallest cube containing this region is as defined above.

□

Theorem 2: *The combinational logic implementing function f derived from a well-formed STG with unique state coding using hazard-free design techniques [13 p.247-249, 14] is hazard-free if and only if*

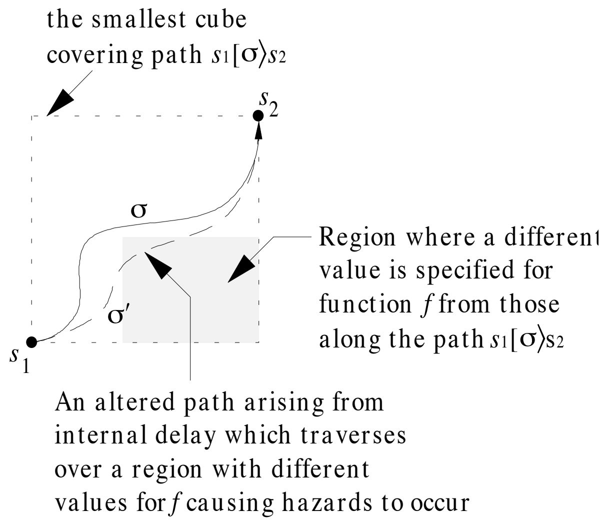
$\forall\, \sigma \in T^*;\ s_1, s_2 \in S : s_1[\sigma\rangle s_2$

if f remains constant during the transition $s_1[\sigma\rangle s_2$ from s_1 to $s_2 \Rightarrow$ only 1s or 0s are specified for function f within the smallest cube containing path $s_1[\sigma\rangle s_2$

Proof:

Firstly, the use of a hazard-free design techniques will eliminate all except function hazards of the implementation [13, 14, 19]. As we have shown earlier, propagation and interconnection delays can alter the effective path perceived by the implementation. However, if the above condition is satisfied, these altered paths will not result in function hazards since function f produces the same output throughout all the possible transitions from s_1 to s_2. If this condition is not satisfied however, then there exists an altered path $s_1[\sigma'\rangle s_2$ which results in a hazard as illustrated in Fig. 8.

□



the smallest cube covering path $s_1[\sigma\rangle s_2$
s_2
σ
σ'
s_1
Region where a different value is specified for function f from those along the path $s_1[\sigma\rangle s_2$
An altered path arising from internal delay which traverses over a region with different values for f causing hazards to occur

Figure 8: *An altered path, $s_1[\sigma'\rangle s_2$, resulting in a hazard as it traverses over a region where function f changes value.*

Using these theorems, an algorithm for the detection of hazard susceptibility is developed.

Algorithm for detecting hazard susceptibility conditions

Given a well-formed STG with unique state coding, $\Sigma_J = \langle P, T, F, M_0 \rangle$, *for each combinational logic,* η_f, *implementing logic function f:*

Let ψ_s *be a set of cubes denoting the cover in the Karnaugh map containing all states, s, such that f(s) = 1 and*

ψ_r, *a cover containing states, s, such that f(s) = 0 (*$\psi_r = \overline{\psi_s}$, *the complement of* ψ_s*).*

For each path $s_1[\sigma\rangle s_2$ *for some* $s_1, s_2 \in S$ *and* $\sigma \in T^*$ *such that f(s) remains constant during the transition from* s_1 *to* s_2

Let C_p *be the smallest cube covering path* $s_1[\sigma\rangle s_2$ *and*

$$\psi_c = C_p \cap \psi_{s/r} \text{ where } \psi_{s/r} = \begin{cases} \psi_r & \text{if } f(s) = 1 \\ \psi_s & \text{if } f(s) = 0 \end{cases}$$

If $\psi_c = \varnothing$ *then implementation* η_f *is free from hazard during the transition* $s_1[\sigma\rangle s_2$ *from* s_1 *to* s_2.

□

Notes: (i) To improve efficiency, the path $s_1[\sigma\rangle s_2$ can be selected to be the longest path that *f* is constant in each case.

(ii) Covers (sets of cubes) could be replaced by sets of states, however with a penalty of efficiency in implementation.

Theorem 3: *Given a transition* $s_1[\sigma\rangle s_2$ *for some* $s_1, s_2 \in S$ *and* $\sigma \in T^*$, *if there exists a cube C such that*

(1) $s_i \notin C \quad \forall$ state $s_i \in s_1[\sigma\rangle s_2$

(2) $C_c = C_p \cap C \neq \varnothing$. *In other words, the cube* C_p *covering transition* $s_1[\sigma\rangle s_2$ *and all its permutations overlaps cube C*

then

$$\forall\, \sigma' \in T^* : \delta(s_1, \sigma') = s_2 \wedge |\sigma'| = |\sigma| \wedge \tau \in \sigma' \Leftrightarrow \tau \in \sigma$$
$$\sigma' \upharpoonright T_C = \sigma \upharpoonright T_C \Rightarrow s'_j \notin C \quad \forall\, s'_j \in s_1[\sigma'\rangle s_2$$

where $T_C = J_C \times \{+, -\}$,

$J_C = \{j \in J \mid C_p(j) = \text{'–'} \wedge C_C(j) \neq \text{'–'}\}$ *(i.e.,* $J_C = \{j \in J \mid C_c(j) \neq \text{'–'}\}$*)*

and

$\sigma \upharpoonright T_C$ *denotes the projection of sequence* σ *onto the set of transitions* T_C.

In other words, given a transition $s_1[\sigma\rangle s_2$ and cube C such that C_p covering transition $s_1[\sigma\rangle s_2$ shares a common region with cube C; if transition $s_1[\sigma\rangle s_2$ only traverses through a region which does not overlap cube C then transition $s_1[\sigma'\rangle s_2$, a permutation of transition $s_1[\sigma\rangle s_2$, will also be restricted to a region which does not overlap cube C if $\sigma' \lceil T_C = \sigma \lceil T_C$.

Proof:
Firstly, notice that:

(1) given transition $s'[\sigma_1\rangle s''$ for some $s', s'' \in S$ and $\sigma_1 \in T^*$, the value of $s''(j)$ for some signal $j \in J$ is determined by the value of $s'(j)$ and the sequence $\sigma_1 \lceil \{j+, j-\}$. In other words, the value of signal j in the starting state s' and the transitions of signal j in the sequence σ_1.

(2) $\forall\, s \in S : s \in C_p$
$s \notin C \Rightarrow \exists\, j \in J : s(j) \neq C_c(j) \wedge C_c(j) \neq$ '–'
$\Rightarrow \exists\, j \in J_C : s(j) \neq C_c(j)$ since $C_c(j) \neq$ '–' $\forall\, j \in J_C$

From (1) and (2), the mutual exclusion of transition $s_1[\sigma\rangle s_2$ and cube C can be stated as

$$\forall\, s_i \in s_1[\sigma\rangle s_2$$
$$\exists\, j \in J_C : s_i(j) \neq C_c(j)$$

In other words, the ordering of transitions $\tau \in T_C$ in σ is one such that mutual exclusion of transition $s_1[\sigma\rangle s_2$ and cube C is assured. As such, $\sigma' \lceil T_C = \sigma \lceil T_C$ would also imply that transition $s_1[\sigma'\rangle s_2$ also will traverse only through a region which is mutually exclusive of C.

□

The restriction of $\sigma' \lceil T_C = \sigma \lceil T_C$ is sufficient but not necessary for the transition $s_1[\sigma'\rangle s_2$ to avoid the cube C. We can relax this ordering by considering the effect of these transitions on the state relative to C. For a signal $x \in J_C$, a transition τ of signal x has the effect of shifting the state value *away* from cube C if $x \neq C_c(x)$ after the firing of transition τ. It has the reverse effect of shifting *towards* and possibly into cube C if $x = C_c(x)$ after the firing of transition τ. A transition τ of signal x will not result in a shift into cube C if:

(1) transition τ is a transition which shifts away from cube C_c or
(2) there exits an earlier transition of another signal $y \in J_C$ such that $y \neq C_c(y)$ after the firing of transition τ.

From theorem 3 and (1) & (2) above, an algorithm to determine the critical timing which must be preserved to avoid function hazard is as follows:

Algorithm to determine the critical timing which must be preserved to avoid function hazard

For a combinational circuit, η_f, implementing logic function f; if there exists a transition $s_1[\sigma\rangle s_2$ and the corresponding cube C_p and cover $\psi_{s/r}$ such that $\psi_c = C_p \cap \psi_{s/r} \neq \emptyset$, then the critical timing which must be preserved to avoid the hazard condition during the transition $s_1[\sigma\rangle s_2$ from state s_1 to s_2 can be derived by determining the critical timing required to avoid each cube $C_c \in \psi_c$ as follows:

```
let $\sigma_c = \sigma \upharpoonright T_C$
for each $C_c \in \psi_c$
    Let $J_C = \{j \in J \mid C_c(j) \neq \text{'–'} \wedge C_P(j) = \text{'–'}\}$,
        $T_C = J_C \times \{+, -\}$,
        $T_C^{□} = \{\tau \in T_C \mid$ the firing of transition $\tau$ will shift away from cube $C\}$ and
        $T_C^{□} = \{\tau \in T_C \mid$ the firing of transition $\tau$ will shift towards from cube $C\}$.
    The ordering of all transitions $\tau \in T_C$ along the path from $s_1$ to $s_2$ should be
    preserved such that the firing of these transitions will not result in a shift into
    cube $C_c$ (i.e., $\forall \tau_2 \in T_C^{□}$, if $\exists \tau_1 \in T_C^{□}$ such that $\sigma_c \upharpoonright \{\tau_1 \bar{\tau}_1 \tau_2\} = \tau_1 \tau_2 \ldots$
    or $\bar{\tau}_1 \tau_1 \tau_2 \ldots$, then the ordering of transition $\tau_1$ must preceed $\tau_2$).
```

□

As an example, consider the implementation $c = b{\cdot}d$ in Fig. 9. Notice that $\psi_s = \{–1–1\}$ while $\psi_r = \{–0––, –––0\}$. During the firing of $c-$ □ $d-$ □ $b+$ □ $a-$ from state 1011 to 0100, we have $C_p = \{––––\}$ giving $\psi_c = C_p \cap \psi_s = \{–1–1\} \neq \emptyset$. Using Theorem 3, $J_C = \{b, d\}$ and hence the ordering of $(c-; d-; b+; a-) \upharpoonright T_C = (d-; b+)$ should be preserved in order to avoid function hazard from occurring during the transition from state 1011 to 0100. In other words, the relation between the delay from b to c and from d to c (denoted in Fig. 9b as $\Delta 1$ and $\Delta 2$ respectively) should be such that the ordering of $d-$ □ $b+$ will always be perceived by the circuit in the right order.

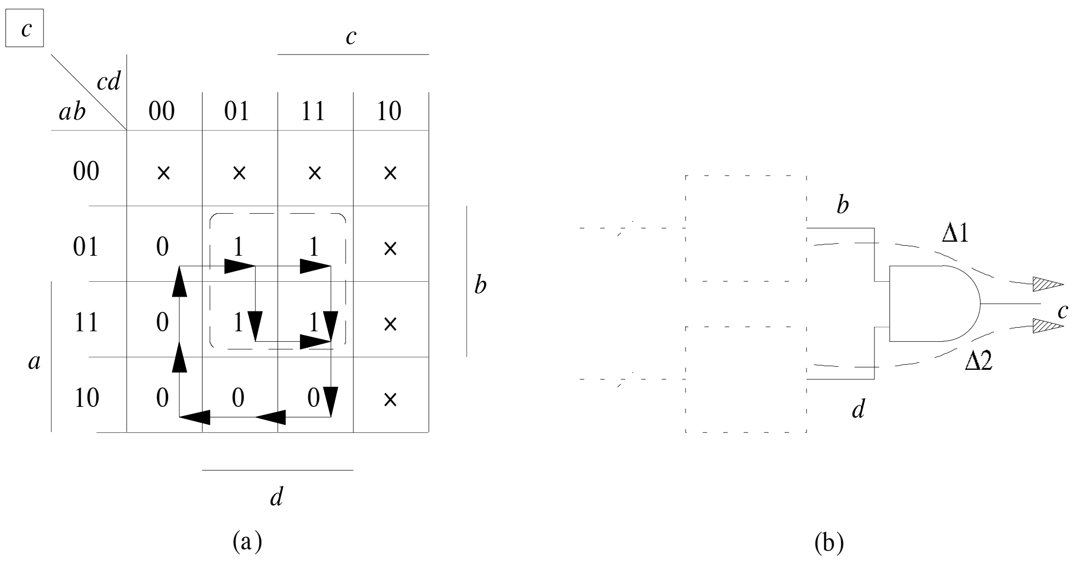


Figure 9: *(a) The K-map for a signal c with its state graph superimposed and (b) the implementation c = b·d.*

In much the same way, the firing of $c-$ □ $b-$ □ $d+$ from state 0110 to 0001 in the example in Fig. 5 will also be detected using the above algorithm. In this example, notice that $\psi_s = \{1---, -1-1\}$ while $C_p = \{0---\}$. Hence, $\psi_c = C_p \cap \psi_s = \{01-1\}$ and $J_C = \{b, d\}$ indicating that the ordering of transitions ($c-$ □ $b-$ □ $d+$) $\upharpoonright T_C = (b-; d+)$ must be preserved to avoid the hazard condition.

This algorithm for determining timing constraints to avoid hazards has no knowledge of pratical circuit delays, wire delays and input transition separations. Consequently, many of the timing constraints will be met by physical realisations. A CAD system could eliminate many constraints automatically. For example in Fig. 9, if Δ1 is known to be always less than Δ2, the ordering $b-$ □ $d+$ is guaranteed.

5. A Comparison with Existing Hazard Detection and Prevention Methods

In this section, the hazard detection algorithm described above is compared with the property of persistency utilised in Chu's [3] technique as well as the hazard removal procedure proposed by Lavagno et al. [5].

As depicted in Fig. 10a, an STG is non-persistent if $\exists\, \mu \in T_N$ and $\tau \in T$ such that transition τ enables transition μ while the firing of μ and $\bar{\tau}$ are concurrent. Using $\sigma \in T^*$ to denote the sequence between transition τ and $\bar{\tau}$, a non-persistent structure in an STG will transform to a structure in its state graph as depicted in Fig. 10b. In this structure, transition μ is not enabled in state s_1 while it remains enabled throughout the transition $s_2[\sigma\bar{\tau}\rangle s_4$. Taking τ and μ as transitions of signal x and y respectively, notice that the smallest cube C_p containing transition $s_2[\sigma\overline{\tau}\rangle s_4$ and its permutations will have $C_p(x) =$ '–'. As such, $C_p \cap s_2 \neq \varnothing$ and $s_1[\tau\rangle s_2$ will further imply that $C_p \cap s_1 \neq \varnothing$ (i.e., cube C_p also contains state s_1). However, since transition μ is not enabled in state s_1, both 1s and 0s will be specified in cube C_p in the Boolean functions implementing signal y or the set/reset signal to the RS flip-flop implementing signal y. In other words, non-persistency corresponds to a hazard condition as defined in the above theorems.

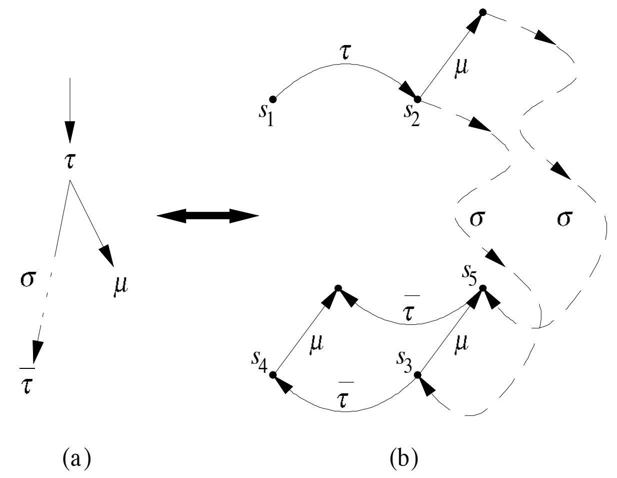


Figure 10: *(a) The characterisation of non-persistency in STG and (b) its corresponding structure in the state graph description.*

Additionally, as depicted in Fig. 11, the insertion of a persistent constraint to ensure persistency will lead to the elimination of path $s_3[\overline{\tau}\rangle s_4[\mu\rangle$. This will reduce the sequence of transitions where μ remains enabled from transition $s_2[\sigma\overline{\tau}\rangle s_4$ to transition $s_2[\sigma\rangle s_3$ or shorter, eliminating the hazard condition in this case. However, as illustrated in the example in Fig. 6, which shows a persistent STG with a hazard in its implementation, persistency alone is insufficient to eliminate all of these hazard conditions.

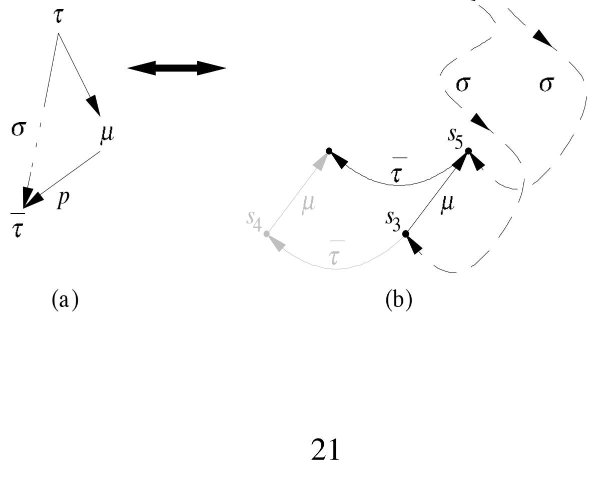


Figure 11: *(a) The insertion of a persistent constraint to ensure the linear ordering of transitions* μ *and* $\bar{\tau}$*. (b) Its corresponding structure in the state graph.*

In the hazard removal procedure proposed by Lavagno et al. [5], these hazard conditions are attributed to the switching of active prime implicants (implicates) as described earlier. Accordingly, the hazard detection technique employed in this procedure is based fundamentally on the detection of a transition sequence $t_2\pm$ □ $t_1\pm$ where transition $t_1\pm$ switches off a prime implicant (implicate) while transition $t_2\pm$ switches on another. Using the above notation, a summary of this procedure is as follows.

Hazard removal procedure (taken from procedure 3.2 [5 p.305-306])
Given an implementation η_f *implementing function* f,
  for each path $s_1[\sigma\rangle s_2$ *such that:*

- s_1 *and* s_2 *have a maximal Hamming distance and*
- $f(s_1) = f(s_2) = f(s_j)$ *for all* s_j *on a state graph path from* s_1 *to* s_2*:*

(a) *let* C *be the smallest cube covering states* s_1 *and* s_2.

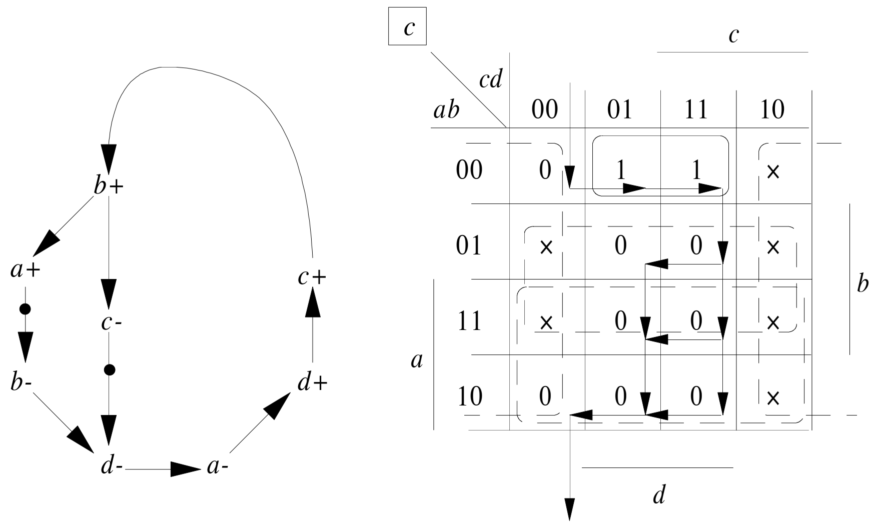
(b) *if* $f(s_1) = f(s_2) = 1$ *then:*
  for each cube $C_0 \in \bar{\psi}_s$ *(the complement of* ψ_s*, see Appendix) intersecting* C *and for each pair of distinct implicants* $C_1, C_2 \in C_s \cap C$ *such that the following Hamming distances are satisfied*
    $d(C_0, s_1) = d(C_1, s_1) + 1$, $d(C_1, C_0) = 1$, $d(C_0, s_2) = d(C_2, s_2) + 1$ *and* $d(C_2, C_0) = 1$
  A. *let* $t_1\pm$ *be the transition moving from* C_1 *to* C_0 *(i.e., '*t_1-*' if* $C_1(t_1) = 1$ *and* $C_0(t_1) = 0$, *'*t_1+*' if* $C_1(t_1) = 0$ *and* $C_0(t_1) = 1$*) and* $t_2\pm$ *the transition moving from* C_0 *to* C_2.
  B. *let* d_1 *be a lower bound on the delay along the path from input* t_1 *to the output of* η_f.
  C. *let* d_2 *be an upper bound is the delay along the path from input* t_2 *to the output of* η_f.
  D. *let* d_3 *be a lower bound on the delay between transition* $t_1\pm$ *and* $t_2\pm$.
  E. *if* $(d_2 - d_1) > d_3$ *then hazard condition exists.*

(c) *else* $(f(s_1) = f(s_2) = 1)$*:*
  similar to step (b) above, but replace ψ_s *with* ψ_r.

$\square$

Recall that in our algorithm, cover $\psi_c = C_p \cap \psi_{s/r}$ for each maximal path, $s_1[\sigma\rangle s_2$, is used to derive **all** the critical paths in the implementation associated with the transition from s_1 to s_2. Compared with the hazard removal procedure proposed in Lavagno *et al.* [5], the need to **search** for cubes C_0, C_1 and C_2 satisfying a set of criteria before the critical ordering of **only**

two signal transitions is determined makes the procedure less efficient than our algorithm. Additionally, the hazard removal procedure proposed in [5] does not detect all hazards. For example in Fig. 12, hazards in the implementation $c = \overline{a} \cdot \overline{b} \cdot d$ during the firing of $d-$ ▯ $a-$ and $a+$ ▯ $b-$ will not be detected by the procedure. This is due to the absence of distinct C_1 and C_2 satisfying the Hamming distance constraints.



$c = \overline{a} \cdot \overline{b} \cdot d$

$\psi_s = \{00\text{–}1\}$ and $\psi_r = \{1\text{–––}, \text{–}1\text{––}, \text{–––}0\}$

Hazard removal procedure proposed in [5]	Our algorithm
Consider the path "$1111[b-c-d-a-\rangle 0000$" with maximal Hamming distance.	Consider the maximal path from 0111 to 0000
$s_1 = 1111, s_2 = 0000, C_p = \text{––––}$ and $C_p \cap \psi_r = \{1\text{–––}, \text{–}1\text{––}, \text{–––}0\}$	$s_1 = 0111, s_2 = 0000, C_p = \text{––––}$ $C_p \cap \psi_s = \{00\text{–}1\} \Rightarrow J_C = \{a, b, d\}$
Note that $\exists\, C_0 \in \psi_s$ and $C_1, C_2 \in C_p \cap \psi_r$ such that $d(C_0, C_1) = 1, d(C_0, C_2) = 1, d(s_1, C_0) = d(s_1, C_1) + 1$ and $d(s_2, C_0) = d(s_2, C_2) + 1$ resulting in a failure to detect any hazard in the implementation $c = \overline{a} \cdot \overline{b} \cdot d$.	$\sigma \lceil T_C = a+\ b-\ d-\ a-$ $T_C^{▯} = \{a+, d-\}, T_C^{▯} = \{b-, a-\}$ detecting the critical ordering $d-$ ▯ $a-$ and $a+$ ▯ $b-$

Figure 12: *An example where the hazard removal procedure proposed in [5] fails to detect hazard conditions in the implementation* $c = \overline{a} \cdot \overline{b} \cdot d$.

6. Conclusion

This paper has shown that previous approaches to achieve hazard-free implementations are inadequate. A new efficient technique has been proposed which detects hazards in implementations and determines critical orderings of signal transitions that are required to avoid these hazards.

APPENDIX: Set operations for cubes in ternary representation

MUTUAL EXCLUSION

Given two cubes, C_1 and C_2, $C_1 \cap C_2 \neq \varnothing$ (i.e., C_1 and C_2 do not overlap) if and only if there does not exist a literal, x, such that $C_1(x) \neq C_2(x)$ and neither of which equals '–'.

Proof:

Given two cubes, C_1 and C_2, if there exists a literal x which has a value 0 in C_1 and a value 1 in C_2 (or vice versa), this implies that C_1 is within a region where x is always 0, while C_2 is within another region where x is always 1 (or vice versa). Since these regions are mutually exclusive, the two cubes, C_1 and C_2, will also be mutually exclusive.

$\square$

INTERSECTION

Given two cubes, C_1 and C_2, a third cube C_3 denoting the intersection of C_1 and C_2 can be obtained using the following algorithm.

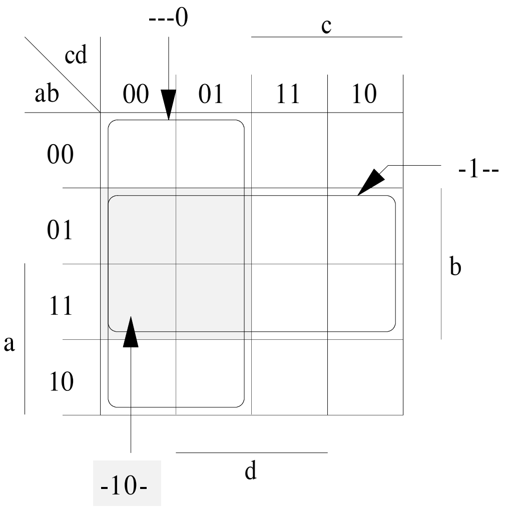
```
if there exists a literal x such that $C_1(x) \neq C_2(x)$ and neither of which equal '–' then
    $C_3 = \varnothing$
else
    for each literal compute
```

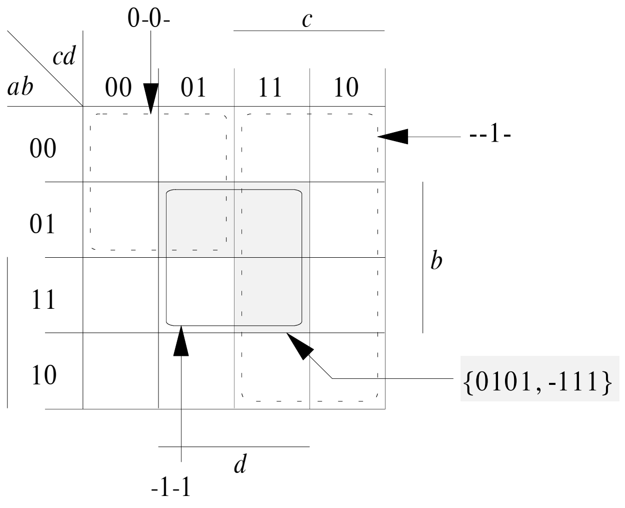
$$C_3(i) = \begin{cases} 0 & \text{if } C_1(i) = C_2(i) = 0 \text{ or if } C_1(i) = 0 \text{ and } C_2(i) = - \text{ or if } C_1(i) = - \text{ and } C_2(i) = 0 \\ 1 & \text{if } C_1(i) = C_2(i) = 1 \text{ or if } C_1(i) = 1 \text{ and } C_2(i) = - \text{ or if } C_1(i) = - \text{ and } C_2(i) = 1 \\ - & \text{if } C_1(i) = C_2(i) = - \end{cases}$$

In this paper, a set of cubes within a Karnaugh map is called a *cover*. The intersection of a cube C and a cover ψ is defined to be the cover

$$C \cap \psi = \bigcup_i \{ C \cap C_i \mid C_i \in \psi \}.$$

Examples:





---0 $\cap$ -1-- = -10-   -1-1 $\cap$ {0-0-, --1-} = {0101, -111}

COMPLEMENT

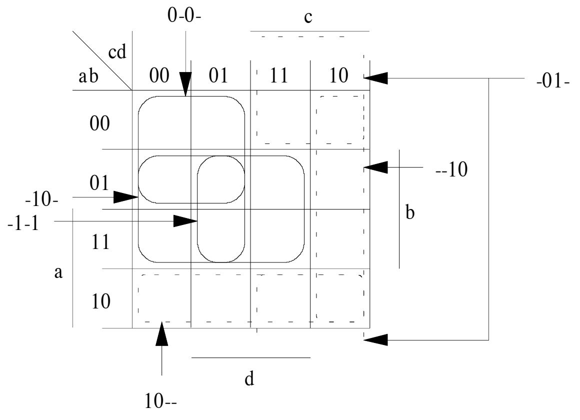
Given a cube C_1, $\overline{C}_1$ (the complement of C_1) is derived using the algorithm:

```
For each literal, x, such that $C_1(x)$ is either a 1 or 0
        invert the value of $C_1(x)$ and set the other literals to '–'
The union of the set of cubes obtained is the complement of $C_1$
```

To find the complement of a cover ψ

- *determine the complement for each of the cube $C_i \in \psi$*
- *find the intersection of all $\overline{C}_i$ (the complement of C_i) derived*
- *the union of all the cubes obtained is the complement of ψ*

Example:



$\psi = \{0\text{–}0\text{–}, \text{–}10\text{–}, \text{–}1\text{–}1\}$

Let $C_1 = (0\text{–}0\text{–})$, $C_2 = (\text{–}10\text{–})$ and $C_3 = (\text{–}1\text{–}1)$

$$
\begin{aligned}
&\overline{C}_1 = \{1{-}{-}{-}, {-}{-}1{-}\}, \overline{C}_2 = \{{-}0{-}{-}, {-}{-}1{-}\}, \overline{C}_3 = \{{-}0{-}{-}, {-}{-}{-}0\} \\
&\overline{C} = \overline{C}_1 \cap \overline{C}_2 \cap \overline{C}_3 \\
&\quad = \{10{-}{-},\ 10{-}0,\ 101{-},\ 1{-}10,\ {-}01{-},\ {-}010,\ {-}01{-},\ {-}{-}10\} \\
&\bigcup_i \{C_i : C_i \in \overline{C}\} = \{10{-}{-},\ {-}01{-},\ {-}{-}10\}
\end{aligned}
$$

References

[1] Chu, T.A., Leung, C.K.C. and Wanuga, T.S., "A Design Methodology for Concurrent VLSI Systems", *Proc. IEEE International Conference on Computer Design*, New York, USA, October , 1985, pp. 407-410.

[2] Chu, T.A. and Glasser, L.A., "Synthesis of Self-timed Control Circuits from Graphs: An Examples", *Proc. IEEE International Conference on Computer Design*, New York, USA, October 6-9, 1986, pp. 565-571.

[3] Chu, T.A., "Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications", Ph.D. dissertation Department of Electrical and Computer Science, Massachusetts Institute of Technology, June 19687.

[4] Chu, T.A., "Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications", *Proc. IEEE International Conference on Computer Design*, New York, USA, October 5-8, 1987, pp. 220-223.

[5] Lavagno, L., Keutzer, K. and Sangiovanni-Vincentelli, A., "Algorithms for Synthesis of Hazard-free Asynchronous Circuits", *Proc. 28th ACM/IEEE Design Automation Conference*, San Francisco, California, USA, June 17-21, 1991, pp. 302-308.

[6] Meng, T.H.Y., Brodersen, R.W. and Messerschmitt, D.G., "Automatic Synthesis of Asynchronous Circuits from High-Level Specifications", *IEEE Trans. on Computer-Aided Design*, Vol. 8, No. 11, November 1989, pp. 1185-1205.

[7] Moon, C.W., Stephan, P.R. and Brayton, R.K., "Synthesis of Hazard-free Asynchronous Circuits from Graphical Specifications", *Proc. IEEE International Conference on Computer Aided Design*, Santa Clara, California, USA, November 11-14, 1991, pp. 322-325.

[8] Vanbekbergen, P. *et al.*, "Optimized Synthesis of Asynchronous Control Circuits from Graph-theoretic Specifications", *Proc. IEEE International Conference on Computer-Aided Design*, Santa Clara, California, USA, November 11-15, 1990, pp. 184-187.

[9] Molnar, C.E., Fang, T-P. and Rosenberger, F.U., "Synthesis of Delay-Insensitive Modules", *Proc. 1985 Chapel Hill Conference on VLSI*, North Carolina, USA, May 15-17, 1985, pp. 67-86.

[10] Chung, E.C.Y. and Kleeman, L., "An Optimal Approach to Implementing Self-timed Logic Circuits from Signal Transition Graphs", *Australian Telecommunication Research*, Vol. 27, No. 2, 1993, pp. 41-56.

[11] Thiagarajan, P.S. and Voss, K., "A Fresh Look at Free Choice Nets", *Information and Control*, Vol. 61, No. 2, May 1984, pp. 85-113.

[12] Wong, C.Y., "On Petri Net Models with Time", Ph.D. dissertation Department of Electrical and Computer Systems Engineering, Monash University, 1987, Chapter 2.

[13] McCluskey, E.J., *Logic Design Principles: with emphasis on testable semicustom circuits*, Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[14] McCluskey, E.J., "Transients in Combinational Logic Circuits" in Wilcox, R.H. & Mann, W.C. (Eds), *Transients in Combinational Logic Circuits*, Spartan Book, Washington 12, D.C., 1962, pp. 9-46.

[15] Miller, R.E., *Switching Theory Volume II: Sequential Circuits and Machines*, John Wiley & Sons, New York, 1965, Chapter 10, pp. 192-244.

[16] Muller, D.E. and Bartky, W.S., "A Theory of Asynchronous Circuits", *Proc. of an International Symposium on the Theory of Switching Part I*, Harvard University, Cambridge, Massachusetts, USA, April 2-5, 1957, pp. 204-243.

[17] Lewin, D., *Design of Logic Systems*, Van Nostrand Reinhold, England, 1985.

[18] Unger, S.H., *Asynchronous Sequential Switching Circuits*, John Wiley, New York, 1969.

[19] Eichelberger, E.B., "Hazard Detection in Combinational and Sequential Switching Circuits", *IBM Journal of Research and Development*, Vol. 9, No. 2, March 1965, pp. 90-99.