Externally Hazard-Free Implementations of Asynchronous Control Circuits

Milton H. Sawasaki, Chantal Ykman-Couvreur, and Bill Lin

***Abstract*—We present a new sum-of-product-based asynchronous architecture, called the N-SHOT architecture, that operates correctly under internal hazardous responses and guarantees hazard-freeness at the observable noninput signals. We formally prove that within this architecture, a very wide class of semimodular state graphs with input choices (either distributive or nondistributive) that satisfy the complete state coding property always admit a correct implementation. As with synchronous circuits, we permit internal hazards in the combinational logic core, which means that we can make use of conventional combinational logic minimization methods to produce the sum-of-product implementation. This represents a significant departure from most existing methods that require the combinational logic to be hazard-free and are mainly valid for distributive behaviors. We also present optimizations for this architecture which are related to state graph properties.**

I. Introduction

THE design of asynchronous circuits is a difficult task since circuit malfunction can occur during execution if the circuit is not hazard-free. In this paper, we consider the problem of producing gate-level asynchronous circuits for both *distributive* and *nondistributive* behaviors from state graph specifications that operate correctly under internal hazardous responses and guarantee hazard-freeness at the externally observable noninput signals (i.e., output and state signals).

We present a new sum-of-product-based architecture, called the N-SHOT architecture, for tackling this problem. We formally prove that within this architecture, a very wide class of semimodular state graphs with input choices that satisfy the complete state coding property *always admit a hazard-free implementation.* Our approach can uniformly handle both distributive and nondistributive specifications. The architecture that we propose for producing a hazard-free implementation for any noninput signal consists of sum-of-product logic implementations for its set and reset functions, an acknowledgment scheme with a local delay compensation to ensure correct restoring of transitions of this noninput signal, and a new flip-flop element for robust electrical operation. *In our approach, the logic implementation for the set and reset functions need not be hazard-free, which means any existing two-level logic minimization method can be used to synthesize logic without restrictions. We believe that this represents a significant contrast to virtually all existing gate-level synthesis methods that require the combinational logic to be hazard-free.*

Our framework is formulated at the state graph level, which means it is widely applicable to a broad class of high-level formalisms that can be semantically defined at the state graph level: the widely used signal transition graph model [3], [22], the change diagram model [23], the CSP model [11], and the burst-mode finite-state machine model [27]. Our method only requires the state graph to satisfy the complete state coding property, which is the minimal requirement necessary to derive unambiguously consistent logic and to satisfy the trigger requirement, which is practically always satisfied.

Our paper is organized as follows. Section II presents an overview of related work. Section III defines the state graph model and the basic concepts that are needed to characterize our method. Section IV presents the new architecture, called the N-SHOT architecture, and its properties. We characterize the class of state graphs that can be implemented with this new architecture and present the synthesis procedure. A key point to our approach is that the logic minimization step is reduced to a conventional logic minimization problem (as in the synchronous case). Section V presents optimizations of the N-SHOT architecture which are related to state graph properties. Our method has been automated in the ASSASSIN compiler [26]. We have carefully simulated a number of synthesized designs at the gate and transistor level (using SPICE) to ensure that the circuits produced by our method are both functionally and electrically correct. In Section VI, we present experimental results and compare them with several available methods. Finally, conclusions are drawn in Section VII.

II. Related Work

Several methods exist for producing hazard-free circuits.

Some approaches, like those described in [27] and [12], rely on a *fundamental mode* of operation to simplify the hazard-free implementation problem, but these methods cannot handle the generality of models like signal transition graphs that permit concurrent input–output changes.

One existing approach is based on *bounded gate/wire delay assumptions*, and relies on delay information to eliminate hazards [7]. Delay lines must be inserted, implying overhead in both area and delay. The current state of this method is restricted to distributive behaviors.

Manuscript received October 21, 1994; revised October 27, 1995 and July 30, 1997. This work was supported in part by the European Commission under the ESPRIT (6143) Project "EXACT." This paper was recommended by Associate Editor R. Camposano.

M. H. Sawasaki is with the Department of Electronics Engineering, University of São Paulo, São Paulo, Brazil.

C. Ykman-Couvreur is with the VLSI Systems and Design Methods Division, IMEC, Leuven, Belgium.

B. Lin is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407 USA.

Publisher Item Identifier S 0278-0070(97)08468-6.

Other existing approaches aim at producing *speed-independent circuits*, which are circuits that will work correctly regardless of the individual gate and wire delays, but the skew of all wire delays at a multiple fan-out point is assumed to be negligible. This is known as the unbounded gate delay model. Several methods [3], [11], [23] generate speed-independent circuits on the assumption that each noninput signal can be implemented with one single complex gate. This complex gate is assumed to contain no internal hazards. However, it is often the case that the required combinational logic functions are too complex to have single complex gate implementations from a standard library. Thus, researchers have investigated the problem of synthesizing speed-independent circuits that assumes only the availability of basic gates such as AND gates and OR gates.

One practically useful approach is to use an architecture that consists of sum-of-product logic implementations for the up- and the down-excitation functions and asynchronous latches (C element or RS) for restoring the primary noninput signals of the specification. Beerel and Meng [1] have developed an approach using this architecture, based on a basic method originally developed by Varshakvsky *et al.* [23], to produce speed-independent circuits. Their approach is based on the use of heuristics and rules to eliminate hazards by restricting logic transformations or inserting additional circuitry. However, their method is restricted to distributive specifications, and even for distributive specifications, it cannot guarantee a correct solution in all cases. Recently, another approach was proposed in [6], also based on the same architecture, that formalizes the requirement that a state graph has to satisfy in order to produce a speed-independent implementation. This requirement, called the *monotonous cover property*, requires additional state signals to ensure that each up and down transition can be separately implemented by a single monotonous AND gate. Thus far, this has proven to be a difficult state assignment problem where a solution may not always be found in practice even though it is theoretically possible. This method is also restricted to distributive specifications.

In [16], another method is presented that makes use of *synchronizers* and a *locally generated clock* to solve the hazard problem. In this method, all external inputs and feedback state signals must be bounded by a synchronizer called a Q flop. This can be very expensive in area since the number inputs is typically much more than the number of feedback state signals, thus requiring many more memory elements than our solution or other existing approaches based on the use of C elements or RS latches [1], [23], [6]. Moreover, the internal clocking scheme requires a tree of N C elements to implement an N-way rendezvous, where N is again the number of external inputs *plus* the number of feedback state signals. The locally generated clock is produced by inserting a delay line that is at least as long as the longest path through the combinational circuit, which means the circuit has to operate in steps that are at least as slow as the worst case delay through the combinational logic. The performance is further hampered by the delay through the N-way rendezvous scheme, which can be significant if N is large. Also, the implementation area of the delay lines is significant. Thus, compared to our approach and other existing approaches like [1], [23], [6] and [7], this approach can be significantly more expensive in terms of both area and performance.

III. State Graph Model

A. State Graphs

A state graph (SG) is a finite automaton given by $G = \langle X, S, T, \delta, s_o \rangle$, where the components are defined as follows.

$X = X_I \cup X_O$ is the set of *signals*, X_I is the set of *input* signals, X_O is the set of *noninput* signals, and $X_I \cap X_O = \emptyset$. Noninput signals refer to both *external output signals* as well as *internal state signals*.

$T = T_I \cup T_O$ is the set of *signal transitions*, T_I is the set of *input* signal transitions, and T_O is the set of *noninput* signal transitions. Each transition can be represented as $+x_j$ or $-x_j$ for the jth up or down transition of signal x. $*x_j$ denotes either a "$+x_j$" transition or a "$-x_j$" transition. Sometimes index j can be omitted.

S is the set of states, and $s_0 \in S$ is the *initial state*. $\delta : S \times T \mapsto S$ is a partial function representing the *transition function* such that if $\delta(s,t) = s'$ is defined, then t is said to be *enabled* in state s and the *firing* of t takes the system from s to s'. This is denoted as $s \xrightarrow{t} s'$ or simply as $s[t\rangle$. The *firing* of a *sequence* of transitions $\sigma = t_1 t_2 \cdots t_m$ is denoted as $s \xrightarrow{\sigma} s'$ or simply as $s[\sigma\rangle$.

Each state $s \in S$ is coded with a *binary vector* $\langle s(1), s(2), \cdots, s(n) \rangle$ according to the signals of $X = \{x1, x2, \cdots, xn\}$ of the system. For a given state $s \in S$, $s(i)$ denotes the value of signal xi in state s. For $s, s' \in S$ and $t \in T$ such that $s \xrightarrow{t} s'$, the state coding is defined as follows: 1) if $s(i) = 0 \wedge t = +xi$, then $s'(i) = 1$; 2) else, if $s(i) = 1 \wedge t = -xi$, then $s'(i) = 0$; 3) otherwise, $s'(i) = s(i)$. In a state $s \in S$, if a transition $t = *xi$ is enabled, the corresponding signal xi is said to be *excited* (which can be denoted as $s(i)*$ in the binary code of s), and if it is not excited, it is said to be *stable*.

A state in the SG captures the state of all signals in a circuit, while a transition between states is a transition of exactly one signal. There may be many signals enabled in a state, but exactly one signal transition is fired at a time. This corresponds to the *interleaved concurrency* model.

An example of an SG is shown in Fig. 1. Here, signals a and b are the input signals and signal c is the output signal.

B. Properties and Objects of SG's

We now formally define some basic properties and objects of SG's which are needed in the characterization of our method. We first recall the *complete state coding* (CSC) property [3] which is the necessary and sufficient condition to the existence of a race-free implementation for any noninput signal.

Definition 3.1 (CSC): An SG satisfies CSC if and only if $\forall s, s' \in S$, either they have different binary codes, or the sets of excited noninput signals in s and s' are identical.

Informally, the CSC property requires that all states of the SG be distinguishable by the environment. In general, the initial SG generated from a high-level specification must be

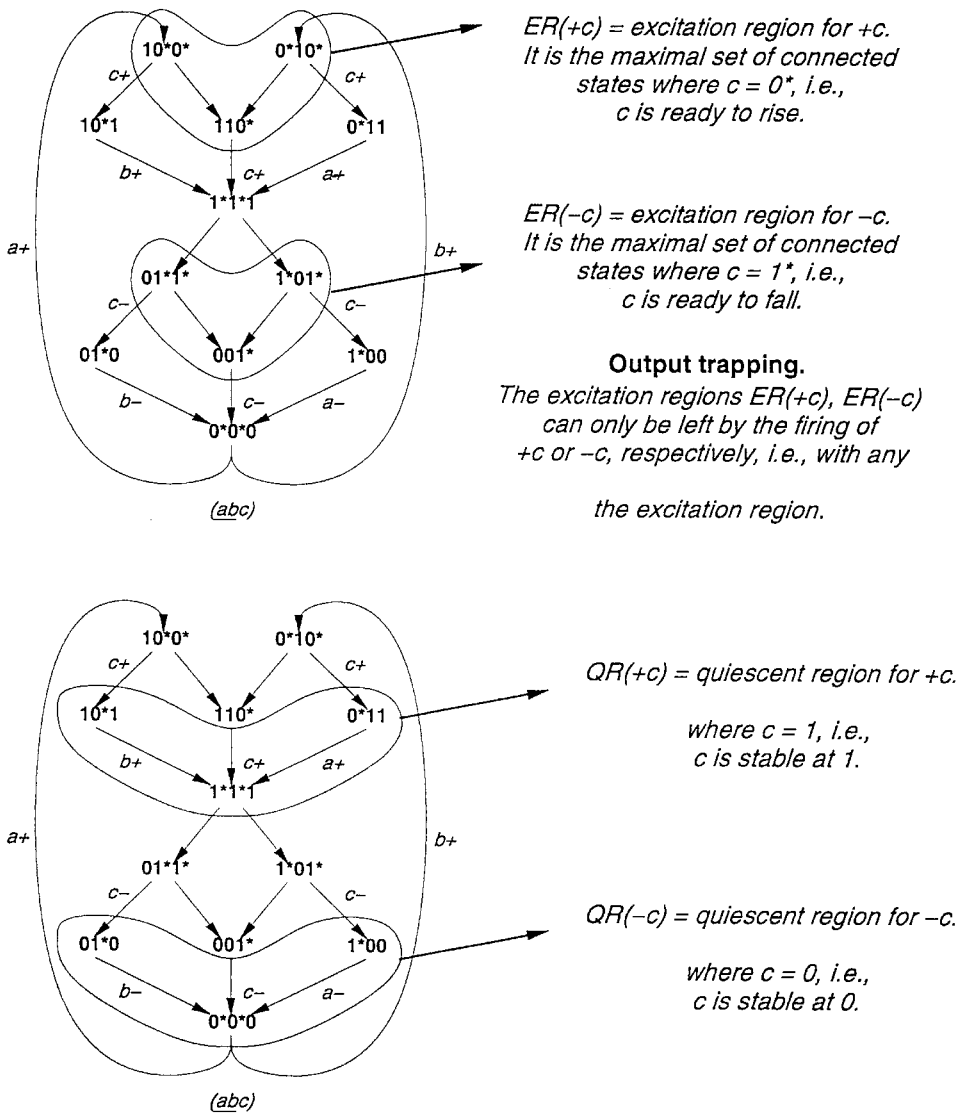

Fig. 1. Example SG.

transformed to satisfy this CSC property. This is referred to as the state assignment problem. Several methods exist for solving this CSC problem at the SG level (see [21], [15], and [25]).

In the example SG of Fig. 1, both states $0*10*$ and $01*0$ violate the CSC property.

We now introduce semimodular SG's with input choices which are SG's used in asynchronous circuit design.

Definition 3.2 (Semimodularity with Input Choices): An SG is semimodular with input choices if and only if $(\forall t_1 \in T_O)$ and $(\forall t_2 \in T)$ and $(\forall s \in S)$, we have

$$s[t_1\rangle \text{ and } s[t_2\rangle \Rightarrow \exists s' \in S : s \stackrel{t_1 t_2}{\rightarrow} s' \text{ and } s \stackrel{t_2 t_1}{\rightarrow} s'.$$

It means that transitions of noninput signals can never be disabled. Only input transitions can be disabled by other input transitions. This decision is assumed to be correctly managed by the environment, and hence creates no problem since only a circuit for noninput signals has to be synthesized. The SG of Fig. 1 is one.

Semimodular SG's with input choices can be classified into either *distributive SG's* or *nondistributive SG's*. This classification is based on the notion of detonant states.

Definition 3.3 (Detonant) [23]: A state w is detonant with respect to a noninput signal a if and only if there exists a pair of states u and v that directly follow w (i.e., $w \rightarrow u, w \rightarrow v$) such that a is stable in state w and is excited in states u and v.

Definition 3.4 (Distributivity): A semimodular SG with input choices is *distributive* with respect to a noninput signal a if and only if there are no detonant states with respect to a.

The SG of Fig. 1 is not distributive because both states $0*0*0$ and $1*1*1$ are detonant with respect to c. In order to derive a correspondence between signal transitions and states in the SG, different regions are defined as follows.

Definition 3.5 (Excitation Region) [23]: An excitation region of signal a in SG is a maximal connected set of states in which a has the same value and is excited.

The excitation region corresponding to transition $*a_i$ is denoted $ER(*a_i)$. Note that there can be several excitation regions for a corresponding to multiple transitions of a.

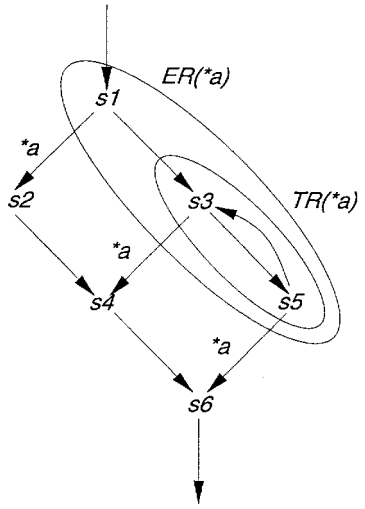

Fig. 2. Trigger region.

An excitation region corresponding to a "$+a$" transition is called an *up-excitation region*. Similarly, an excitation region corresponding to a "$-a$" transition is called a *down-excitation region*.

Definition 3.6 (Quiescent Region) [1]: A quiescent region of signal a in SG is the maximal connected set of states reachable from some $ER(*a)$ in which a has the same value and is stable.

The quiescent region corresponding to transition $*a_i$ is denoted as $QR(*a_i)$, and 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 after $*a_i$.

Fig. 1 illustrates excitation regions and quiescent regions of signal c in the example SG.

Property 3.1: In a semimodular SG with input choices, once we enter an excitation region $ER(*a)$ of a noninput signal a, we can only leave it by firing $*a$.

Proof: We have to prove that $\forall s \in ER(*a)$, $\forall s' \in S$, and $\forall t \in T$

$$s \xrightarrow{t} s' \Rightarrow \text{ either } s' \in ER(*a) \text{ or } t = *a.$$

Any $s \in ER(*a)$ satisfies $s[*a\rangle$. Since the SG is semimodular with input choices, $*a$ cannot be disabled in any such s. Hence, if $s \xrightarrow{t} s'$ and $t \neq *a$, then $s'[*a\rangle$, and hence $s' \in ER(*a)$. □

Definition 3.7 (Trigger Region): A trigger region of signal a in SG is a minimal connected set of states in $ER(*a)$ such that once we enter it, we can only leave it by firing $*a$.

A trigger region corresponding to transition $*a_i$ is denoted as $TR(*a)$. A trigger region can be a single state or a cyclic region. An excitation region can contain several trigger regions. Fig. 2 illustrates a cyclic trigger region. Fig. 8(a) illustrates an excitation region with three trigger regions, each of them composed of one state.

Property 3.2: In a semimodular SG with input choices, a trigger region is always reachable from any state of the corresponding excitation region $ER(*a)$ of a noninput signal a.

Proof: The proof follows from Property 3.1. □

IV. N-SHOT Architecture

A. Overview

In this section, we intuitively describe the N-SHOT architecture, the gate-level implementation of an SG using this architecture, the delay assumptions, and the way the complete circuit behaves within this architecture. In the next sections, we detail the underlying theory of our approach and give the needed proofs.

We assume a *pure delay* model where gates have pure delays. That is, a pulse of any length that occurs on a gate input can propagate to the gate output.

In Fig. 3, the N-SHOT architecture is presented. It can be used to implement in a hazard-free way almost any semimodular SG with input choices, including nondistributive ones, that satisfy the CSC property. Our strategy is to use this architecture to implement each noninput signal in the SG specification. The architecture consists of the following.

1) A *set* sum-of-product (SOP) logic network and a *reset* SOP network for implementing the *up transitions* and the *down transitions* of a noninput signal, respectively. These SOP networks may be minimized using any conventional multioutput two-level minimizer, without special concern for hazards, including the sharing of product terms (AND gates) between different functions.
2) A new asynchronous storage element, called MHS *flip-flop*, for restoring the primary noninput signals of the specification.[1]
3) A *built-in acknowledgment* scheme using two AND gates and a negligible local delay compensation.

The MHS flip-flop behaves functionally in the same way as a C element. However, there are two differences. First, the MHS flip-flop is dual-rail encoded. Second, it is designed to be electrically robust to small pulses. The latter difference is important because we permit the SOP logic networks to be hazardous, which means small pulses due to hazards may be produced (this will be addressed in detail in the next sections).

The *set* logic network for a noninput signal a is derived as follows.

1) Identify all excitation and quiescent regions for a.
2) Take the union of all *up-excitation regions* $\bigcup ER(+a_i)$, and regard it as the *on set* F of the set function.
3) Take the union of all *up-quiescent regions* $\bigcup QR(+a_i)$, and regard it as the *don't-care set* D of the set function. Also, add all *unreachable states* to D.
4) Take the union of all *down-excitation* and *down-quiescent regions* $\cup ER(-a_i) \cup \bigcup QR(-a_i)$, and regard it as the *off set}* R of the set function.
5) Use any multioutput conventional two-level minimizer (e.g., [17]) to produce an optimal SOP implementation from (F, D, R). The don't-care set D may be used freely.

The *reset* logic network for a can be derived analogously.

With the above procedure for deriving the SOP networks, the set and reset logic functions of a noninput signal a are,

[1] M for "M"aster RS latch, H for "H"azard filter, and S for "S"lave RS latch.

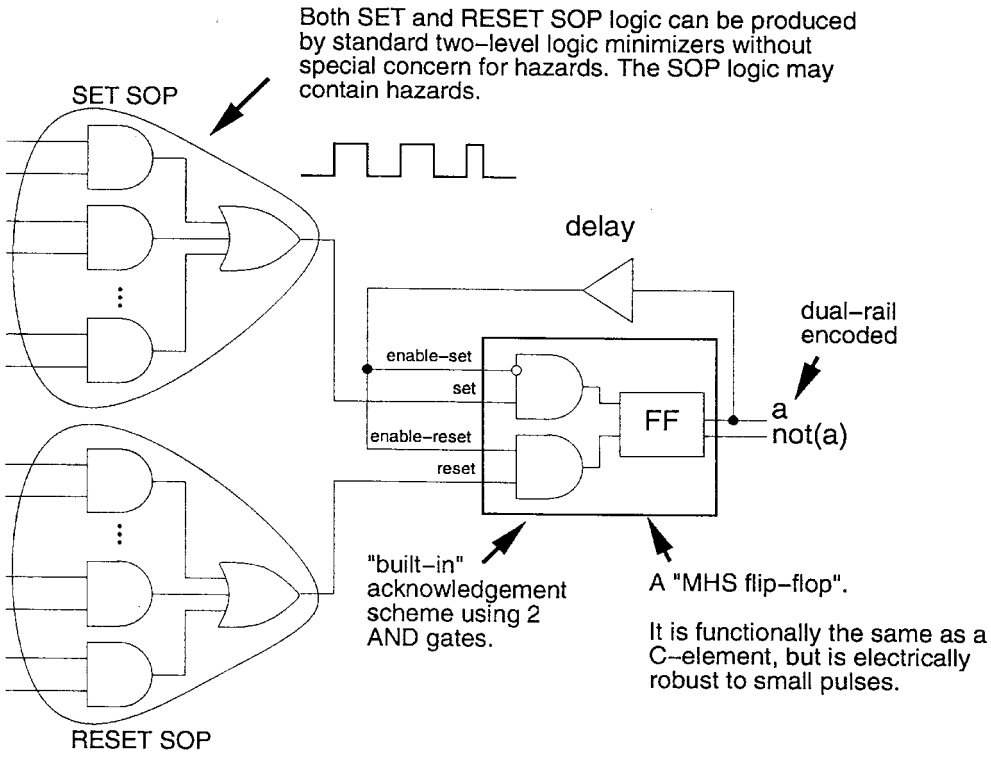

Fig. 3. N-SHOT architecture.

TABLE I
CORRESPONDENCE BETWEEN SG REGIONS AND OPERATION MODES OF MHS FLIP-FLOP

$s \in S$	SET	RESET	mode
$s \in ER(+a)$	1	0	$+a$
$s \in QR(+a)$	$*$	0	$a = 1$
$s \in ER(-a)$	0	1	$-a$
$s \in QR(-a)$	0	$*$	$a = 0$
unreachable s	$*$	$*$	memory

in fact, characterized by its excitation and quiescent regions in the SG. This correspondence is summarized in Table I, where $*$ denotes the don't-care value. In the set mode, where $+a$ must be fired, $ER(+a)$ is traversed. Some set-SOP cubes are excited, and may produce a hazardous response. Then $+a$ is fired and $QR(+a)$ is reached. In the quiescent mode, where $a = 1$, $QR(+a)$ is traversed. When $QR(+a)$ is left, $ER(-a)$ is reached. In the reset mode, where $-a$ must be fired, $ER(-a)$ is traversed. Some reset-SOP cubes are excited, and may produce a hazardous response. Then $-a$ is fired and $QR(-a)$ is reached. In the quiescent mode, where $a = 0$, $QR(-a)$ is traversed. When $QR(-a)$ is left, a next $ER(+a)$ is reached. Thus, we close a cycle on the operation of the MHS flip-flop relative to the SG traversal.

In our logic derivation procedure, the produced SOP logic only requires AND gates and OR gates. Since the MHS flip-flop is dual-rail encoded, inverters are not needed for inverting noninput signals. If all input signals are also presented in dual-rail form, then no inverter at all is needed. Otherwise, we assume that either AND gates with input inversions are available as basic gates or input inversions are implemented with separate inverters whose delays obey some constraints given in [6]. These constraints are also needed in the other basic gate implementations such as [1] and [6]. Under these constraints, the SOP logic never produces 0–1–0 static combinational logic hazards.

In terms of delay assumption, we assume that all gates and wires inside the architecture, including the SOP networks, can have arbitrary delays. Internal wire forks inside the *set* and the *reset* SOP logic networks are not required to be isochronic. External I/O wires can also have arbitrary delays. However, I/O signals that are distributed to multiple destinations must have negligible skews. Also, the feedback wire fork from the output of the MHS flip-flop to the *enable-set* and *enable-reset* inputs must have negligible skews. These delay assumptions are weaker than those assumed by speed-independent implementation methods that require all wire forks, both internal and external wires, to have negligible skews at multiple-fan-out points.

In terms of environment assumptions, we assume that the environment can react immediately, or when it likes, as long as it is enabled to do so in accordance with the SG specification. We do not impose any delay constraints on the environment such as a fundamental mode operation [13], [27].

Since we use conventional logic minimization methods to produce the SOP logic networks, and the gates and wires can have arbitrary delays, the SOP networks may produce hazards that are manifested as pulse streams. This is depicted in Fig. 3. As pulse streams are used to excite the flip-flop, two problems must be solved.

• Any pulse stream, while traversing one $ER(*a) \cup QR(*a)$, must always be converted into one single $*a$ transition at the output of the flip-flop. This is proven in Section IV-B.

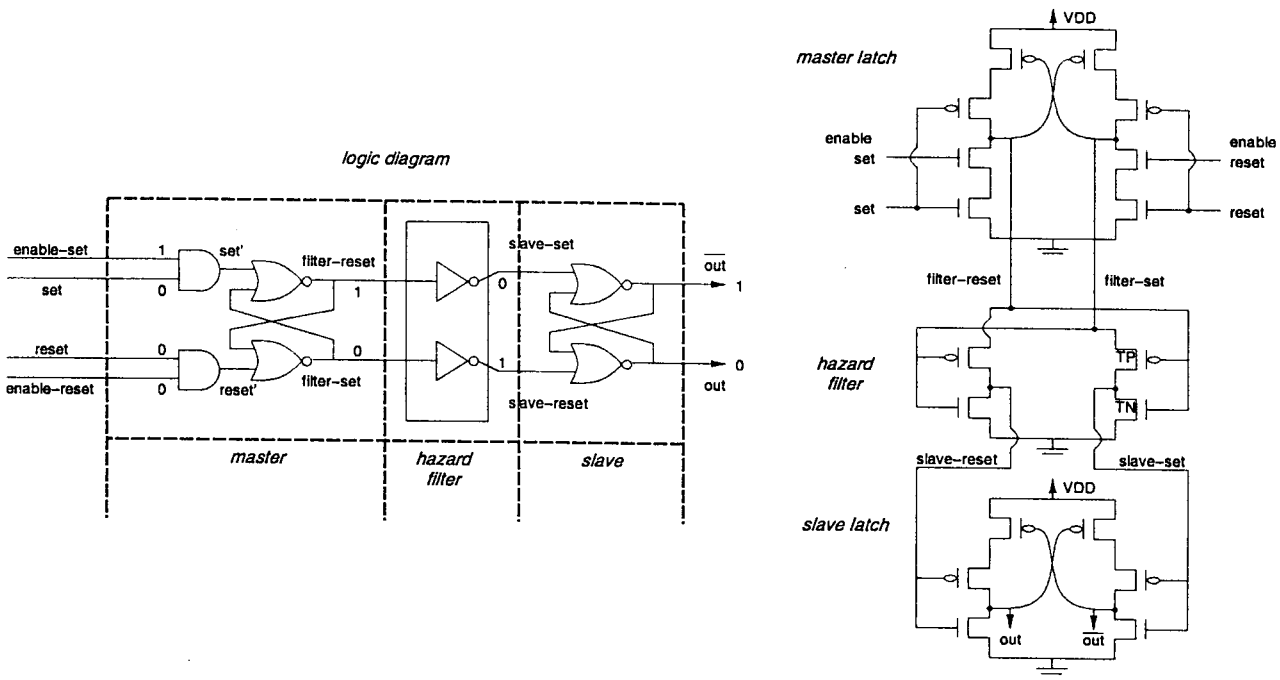

Fig. 4. MHS flip-flop.

• No pulse stream used to fire $+a$ (respectively, $-a$) can continue rippling at the input of the MHS flip-flop after firing of $-a$ (respectively, $+a$). This is ensured by the *enable-set* (respectively, *enable-reset*) signal that is set to *high* only when $-a$ (respectively, $+a$) is fired and when the set (respectively, reset) SOP has completely settled to 0. This is the purpose of both AND gates in the acknowledgment scheme and of the local delay compensation.

Concerning this delay line, the delay compensation requirement is always satisfiable. Moreover, the delay value is negligible because it is related to the *difference* of delays between the set and reset SOP's. If the difference is small compared to the delay of the MHS flip-flop itself, which is almost always the case when SOP implementations are used, then *no* delay compensation is required. When delay compensation is required, the delay is on the order of a gate delay or less. The delay line is placed in parallel with both set SOP and reset SOP. Hence, it is out of their critical paths, and its overhead effect, *if any*, is indirect.

This is proven in Section IV-C.

Although the gates and wires used in our architecture can have arbitrary delays, *bounds* on the delays must be known in order to determine what local delay compensation, if any, is required. As such, our designs in general are neither speed-independent nor delay-insensitive. However, optimizations related to SG properties are possible, and can lead to speed-independent designs as explained in Section V. Again, it is important to remark that usually no delay compensation is required. For all the examples tested in Section VI, delay compensation was never required.

B. Set and Reset Modes

In this section, we explain how any pulse stream, while traversing one $ER(*a) \cup QR(*a)$, is converted into one single $*a$ transition at the output of the flip-flop.

To that end, we first describe the MHS flip-flop. The logic diagram of the MHS flip-flop with the acknowledgment scheme is shown in Fig. 4(a), and a custom designed transistor level implementation (without initializations[2]) is shown in Fig. 4(b). The MHS flip-flop behaves as a C element. However, a C element is not immune to short pulse misbehavior. Also, the MHS flip-flop is dual-rail encoded.

The MHS flip-flop is composed of three distinct parts that provide hazard filtering in two stages. Its parts and functions are as follows.

1) The master RS latch converts a pulse into an analog voltage.
2) The hazard filter produces hazard-free up transitions on its output signals. This is the first hazard filtering stage.
3) The slave RS latch filters the hazardous down transitions from the filter. This is the second filtering stage.

As both set and reset modes of the MHS flip-flop are symmetric, we only explain the functions of its three parts when it is in a set mode. In the implementation of a noninput signal a, using the N-SHOT architecture, the MHS flip-flop is in the set mode when up-excitation regions of a are traversed in the SG. The set SOP logic responds with a pulse stream that excites the MHS flip-flop to fire $+a$. The initial state of the MHS flip-flop in the set mode is shown in Fig. 4(a). The response time is illustrated in Fig. 5, where τ is the delay

[2]Initialization of the MHS flip-flop may be required. This is addressed in Section IV-E.

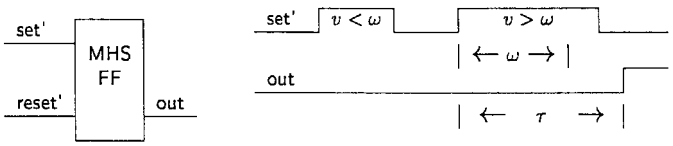

Fig. 5. MHS flip-flop response.

response of the MHS flip-flop and $\omega < \tau$ is the delay of the used RS latches. ω is about twice the delay of a NOR gate. It is also the threshold time value of the MHS flip-flop. The MHS flip-flop does not transmit a pulse shorter than ω and for pulses larger than ω, the output transition is simply translated forward in time by τ.

Behavior of the Master RS Latch: First, we analyze the behavior of the master RS latch. Assume that a pulse stream is applied to its input *set'*. Each pulse has some unknown width v, and for each of them, the master RS latch can respond in one of the following regions.

- The *stable region* $(v \geq \omega)$: As illustrated in Fig. 6(a), the output *filter_reset* makes a single monotonic down transition, and the output *filter_set* makes a single monotonic up transition.
- The *inversion region* $(v < \omega)$: The behavior of the outputs *filter_reset* and *filter_set* is illustrated in Fig. 6(b).
- The *metastable region* $(v < \omega)$: It is a special case of the inversion region, where both outputs *filter_set* and *filter_reset* remain for some time at a constant value. This is illustrated in Fig. 6(c). However, even if a pulse can cause the master latch to get metastable (the probability of such an event is almost zero [2], [4]), any next pulse of width $v \geq \omega$ takes it out of this region. The existence of such a pulse is guaranteed by the trigger requirement, characterized below.

Requirement 4.1 (Trigger Requirement): For every noninput signal a implemented with the N-SHOT architecture, and for any transition $*a$ specified in the SG, a pulse of width $v \geq \omega$, on the output of the corresponding SOP network, must exist that can cause the MHS flip-flop to fire $*a$.

Behavior of the Hazard Filter: Then we analyze the behavior of the hazard filter. The hazard filter is two degenerated inverters. In fact, the master latch and this filter can also be used as a basis for a mutual exclusion element [18], [11]. The latter is used in the context of arbitration to solve output nonsemimodularities. It is designed to avoid the transmission of metastable states. An analysis on metastable states can be found in [4]. Although we have a different context (semimodular SG's with input choices do not capture output nonsemimodularities), we profit from some characteristics of the mutual exclusion element to build the first hazard filtering stage of the MHS flip-flop.

Assume that the trigger requirement is satisfied. If a pulse stream is applied to *set'*, a hazardous down transition of *filter_reset* and a hazardous up transition of *filter_set* are produced. The hazard filter produces a hazard-free up transition on its output *slave_set* and a hazardous down transition on its output *slave_reset*. This is the first hazard filtering stage. This is illustrated in Fig. 7, where some simulation of the complete response of the MHS flip-flop to hazardous inputs *set* and *reset* is given.

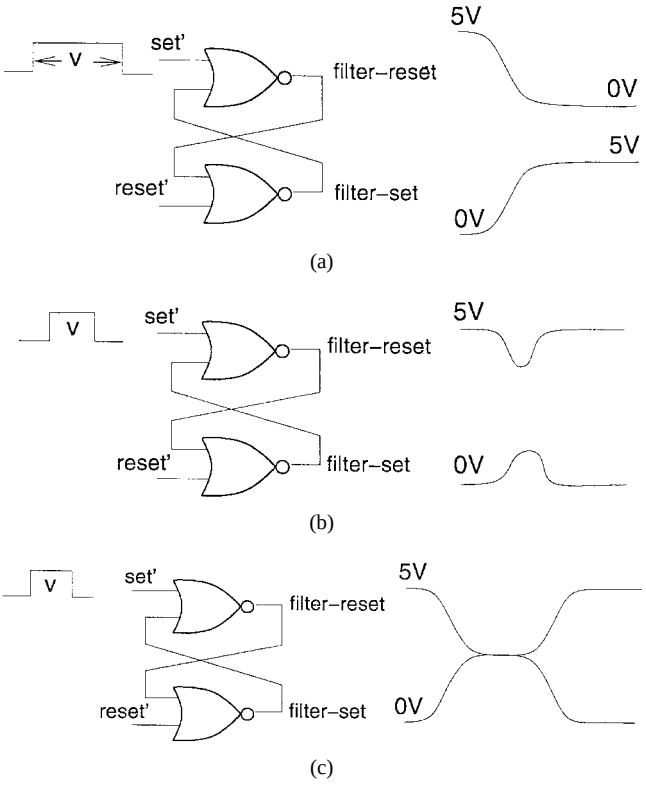

Fig. 6. Responses of the master latch.

Property 4.1: If the trigger requirement is satisfied, any output up transition of the hazard filter is always hazard-free.

Proof: Again, as the hazard filter is symmetric, we only show that any up transition on *slave_set* is hazard-free. As indicated in Fig. 4(a), the initial state of the hazard filter is *filter_reset* $= 1$, *filter_set* $= 0$, *slave_set* $= 0$, and *slave_reset* $= 1$.

filter_set serves as virtual VDD for the transistor TP [see Fig. 4(b)]. Initially, *filter_set* $= 0$ so that TP is *OFF* too. The difference $V_{\text{diff}} = V_{\textit{filter\_set}} - V_{\textit{filter\_reset}}$ determines the switching of TP. If V_{TP} is the threshold voltage to turn TP *ON*, we observe the following.

- In the inversion and metastable regions, before reaching the stable region, $V_{\text{diff}} < V_{TP}$. Hence, TP is kept *OFF*.
- Since the trigger requirement is satisfied, the stable region is reached.
- When the stable region is reached, we move from $V_{\text{diff}} < V_{TP}$ to $V_{\text{diff}} > V_{TP}$. Hence, TP is turned *ON*, and a single monotonic up transition is produced on *filter_set*.□

Behavior of the Slave RS Latch: Finally, we analyze the behavior of the slave RS latch. A hazard-free up transition on its input *slave_set* is produced, whereas a hazardous down transition on its input *slave_reset* is produced. Then the slave RS latch behaves as follows.

1) As long as *slave_set* is *OFF*, *slave_reset* can fluctuate. This fluctuation has no effect on the output *out* because signal $\overline{out}$ is *ON* and holds *out OFF*.
2) Simultaneously, a single monotonic up transition is produced on *slave_set* and *slave_reset* is definitively turned *OFF*.

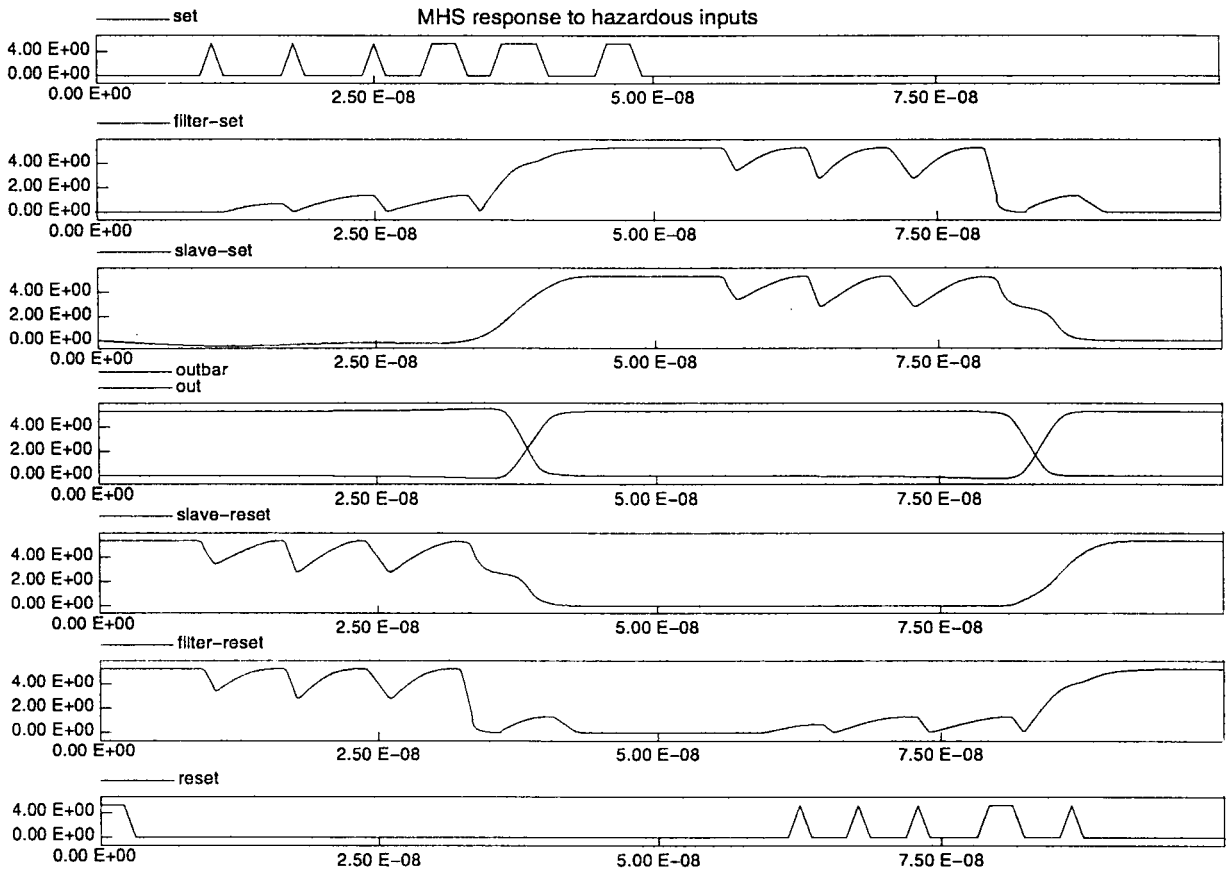

Fig. 7. Response of the MHS flip-flop to hazardous inputs.

3) A single monotonic down transition is produced on $\overline{out}$ and a single monotonic up transition is produced on *out*.

This is the second hazard filtering stage. It is obtained through the protocol used in the communication between the filter and the slave latch.

Conclusion: If the trigger requirement is satisfied, the MHS flip-flop transforms any SOP pulse stream on either *set*′ or *reset*′ into a single transition on its output *out* and $\overline{out}$ during the traversal of the corresponding excitation region.

We now formulate a necessary and sufficient condition on the SG to guarantee the firing of any specified noninput transition $*a$ in the N-SHOT architecture.

Definition 4.1 (Trigger Cube): A trigger cube is a cube from either the set or the reset SOP that completely covers a trigger region.

Theorem 4.1: A semimodular SG with input choices satisfies the trigger requirement if, and only if in the implementation of any noninput signal a, using the N-SHOT architecture, there exists a trigger cube associated with each trigger region of a.

Proof: Due to Properties 3.1 and 3.2, once we enter an excitation region $ER(*a)$, we can only leave it by traversing one of its trigger regions.

We first prove the necessary condition. This is done by contradiction. Suppose that, in the implementation of a, two cubes are needed to entirely cover some trigger region. Then once we enter this trigger region, we can move from one cube to another one, and a pulse stream can be generated. Since we cannot predict the speed at which those cubes are traversed, those pulses can all be shorter than ω. If $*a$ corresponding with $ER(*a)$ is not yet produced at the output of the MHS flip-flop, none of them can cause the MHS flip-flop to fire any more. Then we violate the trigger requirement and enter a deadlock situation.

We then prove the sufficient condition. Once a trigger region is reached, the corresponding trigger cube in the implementation of a, switches *ON*, and can only switch *OFF* after $*a$ has fired. Indeed, all reachable states inside the trigger region are covered by the same cube. Thus, this trigger cube guarantees that the trigger requirement is satisfied. □

C. Quiescent Mode

In this section, we show that any pulse stream produced during the traversal of $ER(+a_1) \cup QR(+a_1)$ does not go across the traversal of $ER(-a_2)$. Indeed, if it goes across $ER(-a_2)$ and reaches $QR(-a_2)$ where $-a$ has already fired, one of those pulses can cause a to misfire during the traversal of $QR(-a_2)$. This trespassing pulse problem is solved using the acknowledgment scheme and the local delay compensation.

In the N-SHOT architecture of Fig. 3, we make the following timing assumptions.

- t_{mhs+} (respectively, t_{mhs-}) is the response time of the MHS flip-flop to produce a $+a$ (respectively, $-a$).
- t_{set1} (respectively, t_{set0}) is the propagation time through the set SOP (that is, through two gates) to produce

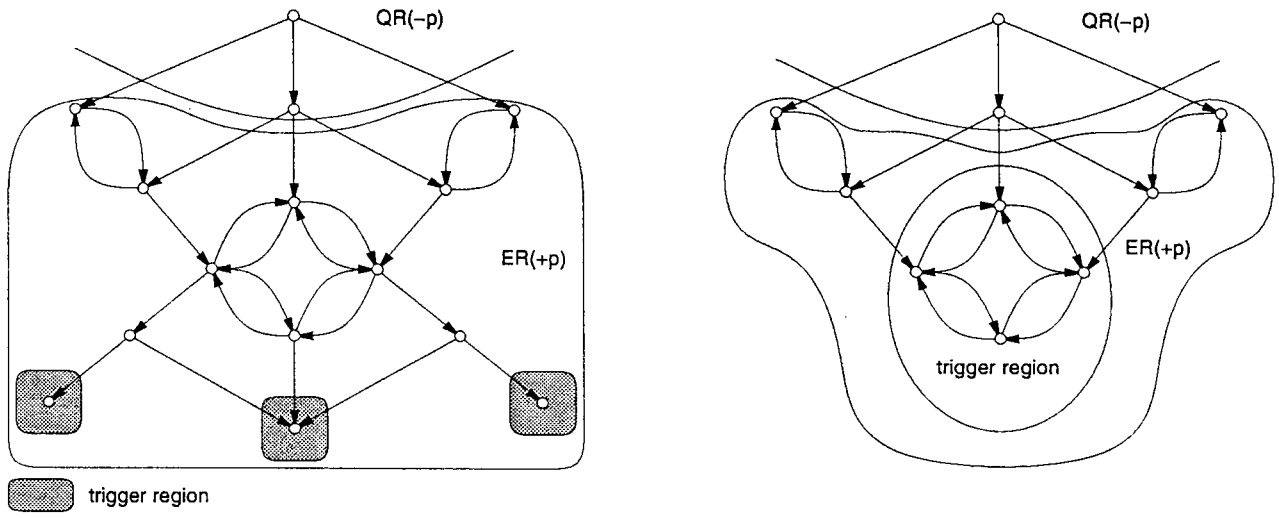

Fig. 8. (a) Single traversal SG. (b) Nonsingle traversal SG.

an up (respectively, down) transition. Dependent on the currently traversed AND gate, $lt_{set1} \leq t_{set1} \leq ut_{set1}$ and $lt_{set0} \leq t_{set0} \leq ut_{set0}$.

- t_{reset1} and t_{reset0} are defined analogously.

To derive the delay requirement, let us examine the mechanism behind this acknowledgment scheme of Fig. 3.

1) During the traversal of $ER(+a_1) \cup QR(+a_1)$, the set SOP produces a pulse stream.
2) When $ER(-a_2)$ is entered, two things happen in parallel.
   - The set SOP either has already settled to 0 or starts to settle to 0. The time it takes is at most ut_{set0}.
   - The reset SOP begins to produce a pulse stream. Then $-a_2$ is fired at least after $lt_{res1} + t_{mhs-}$.
3) After the firing of $-a_2$, signal *enable-set* switches *ON* again after t_{del}, the value of the delay line. We have to guarantee that the set SOP is completely stabilized to 0 when this occurs. It implies that
$$t_{del} \geq ut_{set0} - (lt_{res1} + t_{mhs-}).$$
4) When $ER(+a_3)$ is entered, since the MHS flip-flop is symmetric, we also have to guarantee
$$t_{del} \geq ut_{res0} - (lt_{set1} + t_{mhs+}).$$

Hence, the delay requirement is

$$t_{del} \geq \mathrm{MAX}\{ut_{set0} - lt_{res1} - t_{mhs-}, \; ut_{res0} - lt_{set1} - t_{mhs+}\}. \quad (1)$$

It can always be satisfied, and the delay line becomes unnecessary when $MAX \leq 0$. In general, the delay value is negligible because it is related to the *difference* of delays between the set and reset SOP's. Moreover, if the difference is small compared to the delay of the MHS flip-flop itself, which is almost always the case when SOP implementations are used, then again *no* delay compensation is required. When delay compensation is required, the delay is on the order of a gate delay or less. Since the delay line is placed in parallel with both set SOP and reset SOP, it is out of their critical paths. Hence, its overhead effect, *if any*, is indirect.

As we are dealing with transmission of signals through delay lines, it is important to make sure that the signal is really transmitted. Suppose that $+a$ is produced at the output of the MHS flip-flop. Then a is kept stable to 1 because the enabling of the MHS flip-flop to fire $-a$ can only occur after the delay line changes its value from 0 to 1. Hence, when $-a$ is produced, it means that $a = 1$ has already been safely translated through the delay line. That is, we have a causal chain of events in the acknowledgment scheme of the N-SHOT architecture that automatically provides the required stability of the signal exciting the delay line.

D. Hazard-Freeness and Synthesis Procedure

Theorem 4.2: A semimodular SG with input choices *always admits a hazard-free implementation* using the N-SHOT architecture, if and only if it satisfies the CSC property and the trigger requirement.

Proof: First, the CSC property is a necessary and sufficient condition to the existence of a race-free implementation for any noninput signal a of the SG.

From the previous analysis on the set and reset mode of the MHS flip-flop, any set and reset SOP networks, implementing the set and reset functions of a, may be used to produce at the output of the MHS flip-flop the up and down transitions of a specified in the SG, if and only if each trigger region of a is entirely covered by a trigger cube, that is, if and only if trigger requirement is satisfied (see Theorem 4.1).

From the previous analysis on the quiescent mode of the MHS flip-flop, the delay requirement can always be satisfied. Hence, the MHS flip-flop, together with the acknowledgment scheme and the local delay compensation, may be used as set–reset element for a. □

Definition 4.2 (Single Traversal SG): A single traversal SG is an SG for which any trigger region contains only one state.

In Fig. 8, both a single traversal SG and a nonsingle traversal SG are shown. A nonsingle traversal SG can occur, for example, when free-running signals like clocks are used

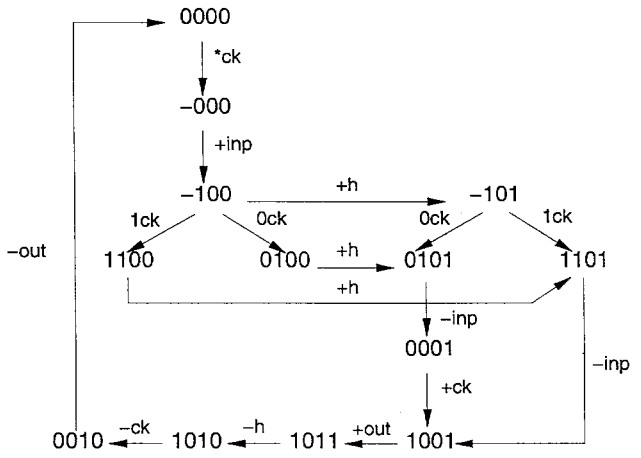

Fig. 9. SG with free-running signals.

in the specifications. Observe that this nonsingle traversal SG, however, can satisfy the trigger requirement.

Theorem 4.3: A semimodular SG with input choices *always admits an optimal hazard-free implementation* using the N-SHOT architecture, if and only if it satisfies the CSC property and is single traversal.

Proof: A semimodular SG with input choices that satisfies the CSC property and is single traversal always satisfies the trigger requirement. Indeed, any trigger region consists of one state, and any cube in the implementation covering this state is always a trigger cube. Hence, such an SG always admit a hazard-free implementation (see Theorem 4.2).

Then since no constraint must be put on the SOP networks implementing the set and reset functions of a, any minimization technique, such as ESPRESSO-EXACT, can be freely used to generate them. □

Synthesis Procedure: Hence, the synthesis procedure for any semimodular SG with input choices satisfying the CSC property and the trigger requirement is straightforward and is described as follows.

1) Derive an optimal set SOP and an optimal reset SOP using any logic minimizer.
2) If the SG is not single traversal, ensure that a trigger cube corresponds with each trigger region.
3) Map these SOP logic into the N-SHOT architecture.
4) Determine the delay value.

Consider the SG of Fig. 9. This is an extended SG as defined in [22], which is single traversal, and for which this N-SHOT architecture can also be used. ck is a free-running clock resulting in don't-care values in some state codes. *inp* is another input signal, and *out*, h are the output signals. ESPRESSO yields the following logic for both set and reset of signals *out* and h:

$$\begin{aligned} \mathit{set\_out} &= ck\ \overline{\mathit{inp}}\ h \\ \mathit{reset\_out} &= \overline{ck} \\ \mathit{set\_}h &= \mathit{inp} \\ \mathit{reset\_}h &= \mathit{out}. \end{aligned} \tag{2}$$

Consider again the SG of Fig. 1, and solve the CSC violations by inserting a state signal h, as indicated in Fig. 10.

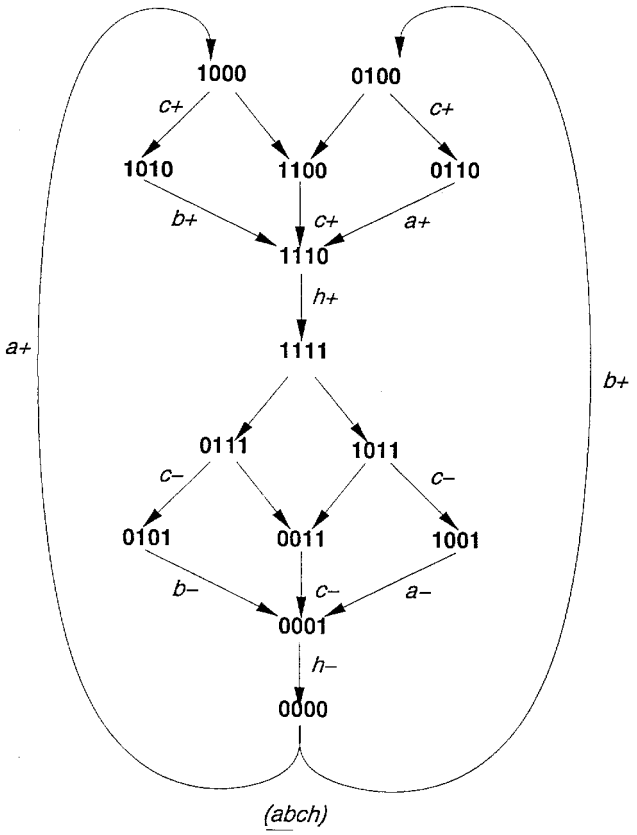

Fig. 10. Example SG of Fig. 1 after state assignment.

This is a nondistributive SG, as already mentioned in Section III-B. ESPRESSO yields the following logic for both set and reset of signals c and h:

$$\begin{aligned} \mathit{set\_}c &= a\,\overline{h} + b\,\overline{h} \\ \mathit{reset\_}c &= \overline{a}\,h + \overline{b}\,h \\ \mathit{set\_}h &= a\,b\,c \\ \mathit{reset\_}h &= \overline{a}\,\overline{b}\,\overline{c}. \end{aligned} \tag{3}$$

In none of the above examples are delay lines needed.

E. Initialization of the MHS Flip-Flop

Initialization of the MHS flip-flop can be required. One trivial solution is to provide a "reset" product term at one output of the master RS latch in the MHS flip-flop. A short analysis of the final set and reset SOP's can detect when this reset is unnecessary.

For any noninput signal a implemented with the N-SHOT architecture, consider the initial state $s_0 \in S$. The reset is only needed in both situations:

1) $s_0 \in QR(+a)$ and $\mathit{set\_}a(s_0) = 0$, in which case the MHS flip-flop must be reset to 1;
2) $s_0 \in QR(-a)$ and $\mathit{reset\_}a(s_0) = 0$, in which case the MHS flip-flop must be reset to 0.

In the other situations, the MHS flip-flop is always automatically initialized to either 1 if $s_0 \in ER(+a) \cup QR(+a)$ or 0 if $s_0 \in ER(-a) \cup QR(-a)$.

V. Optimizations

In this section, we consider the possible optimizations in the N-SHOT architecture implementing a noninput signal a. When the set and reset SOP networks satisfy some properties related to the SG, the N-SHOT architecture can be simplified, permitting either to eliminate the MHS flip-flop and yield an SOP implementation as final circuit (see Theorem 5.1), or to replace the MHS flip-flop by a C element (see Theorem 5.2).

In this section, we use the notation

$$\text{set SOP} = \sum_i S_i \quad \text{and} \quad \text{reset SOP} = \sum_j R_j$$

where S_i denotes any set-SOP cube in the implementation of the set function of a, and R_j denotes any reset-SOP cube in the implementation of the reset function of a.

Theorem 5.1: Assume that set SOP $= \Sigma_i\ S_i$ satisfies the following.

- $\Sigma_i\ S_i$ is independent of a.
- $\forall S, S' \in \Sigma_i\ S_i$: $S \cap S'$ covers no reachable state.
- Each $ER(+a) \cup QR(+a)$ is completely covered by some $S \in \Sigma_i\ S_i$.

Then $a = \Sigma_i\ S_i$ is a speed-independent SOP implementation of a.

Proof: Since any $ER(+a) \cup QR(+a)$ is completely covered by a cube S of the implementation of a, once we enter $ER(+a)$, the cube S switches *ON* only once, and can only switch *OFF* when $ER(-a)$ is reached. All cubes remain *OFF* until a next $ER(+a)$ is reached. □

$\overline{a} = \Sigma_j\ R_j$ is also a speed-independent SOP implementation of a when the symmetric conditions, relative to the reset-SOP cubes R_j, are satisfied.

Theorem 5.2: Assume the following.

- $\Sigma_i\ S_i$ and $\Sigma_j\ R_j$ are both independent of a.
- $\forall S, S' \in \Sigma_i\ S_i$: $S \cap S'$ covers no reachable state, and $\forall R, R' \in \Sigma_j\ R_j$: $R \cap R'$ covers no reachable state.
- Each $ER(+a)$ is completely covered by some $S \in \Sigma_i\ S_i$, and each $ER(-a)$ is completely covered by some $R \in \Sigma_j\ R_j$.
- No delay line is needed.

Then the MHS flip-flop together with the acknowledgment scheme can be replaced by a C element in the N-SHOT architecture.

Proof: Since any excitation region $ER(*a)$ is completely covered by some cube of the implementation of a, once we enter $ER(*a)$, the corresponding cube switches *ON* and produces a first pulse that causes the C element to fire $*a$. Then we enter $QR(*a)$, and other pulses can follow that produce no effect on the output of the C element.

Since no delay line is needed, it implies that the difference of delays between the set and reset SOP's is negligble. Hence, when we enter the next excitation region, the current cube has already settled to 0. □

Consider again the SG of Fig. 9 and the logic (2) for the set and reset SOP's. Both *out* and h satisfy the conditions of this theorem and can be implemented with only a C element.

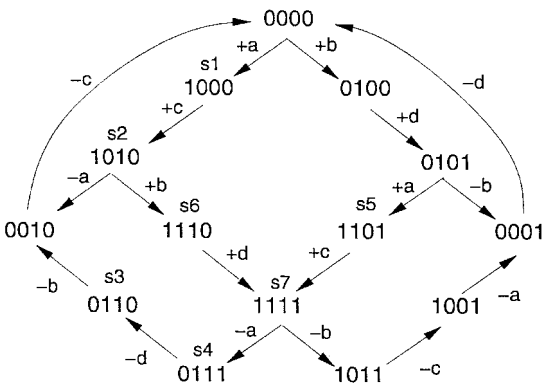

Fig. 11. Example SG.

Finally, if the SG satisfies the *monotonous cover* condition, as defined in [6], then the MHS flip-flop together with the acknowledgment scheme and the local delay compensation can be replaced by a C element in the N-SHOT architecture. This implementation is speed-independent, as proved in [6]. The monotonous cover condition essentially requires that all excitation regions be uniquely covered by a single cube.

Consider the SG of Fig. 11 where a, b are input signals and c, d are output signals. Consider the output signal c. One possible set SOP is $a\overline{bd}+abd$ covering states s_1, s_2, s_5, s_7. One $QR(+c)$ consists of the states s_2, s_3, s_4, s_6, s_7, and those that are covered by the set SOP are s_2, s_7 which are not connected in the SG. Hence, the implementation of c using this set SOP is not speed-independent.

For the SG of Fig. 9 and the corresponding logic in (2), the implementation of h satisfies the conditions of this theorem. It implies that it is speed-independent. However, the implementation of *out* is not speed-independent: the states of $QR(-out)$ covered by the cube $\overline{ck}$ in the reset implementation are not connected in the SG.

For the SG of Fig. 10 and the corresponding logic in (3), the implementation of h also satisfies the conditions of this theorem, and hence is speed-independent. However, the implementation of c must use the general N-SHOT architecture: in the set SOP, cubes cover common states 1100, 1110, and 1111; in the reset SOP, cubes cover common states 0011, 0001, and 0000.

VI. Experimental Results

In our view, the most important contribution of this work is a new method that is guaranteed to produce an optimal hazard-free gate-level circuit for a very broad class of SG's, including nondistributive specifications. This method can make use of conventional highly tuned two-level logic minimization methods (as in the synchronous case) without the usual complexities and difficulties of ensuring hazard-freeness during logic minimization. Our method only requires the SG to satisfy the CSC property, which is the minimal requirement necessary to derive unambiguously consistent logic and to satisfy the trigger requirement, which is practically always satisfied.

The synthesis procedure and optimization techniques described in this paper are implemented in the ASSASSIN com-

TABLE II
EXPERIMENTAL RESULTS

name	states	SIS (Lavagno)	SYN (Beerel)	ASSASSIN	
		area/del	area/del	area/del	largest cube
chu133	24	352/5.2	232/4.8	256/4.8	3
chu150	26	232/7.0	240/4.8	240/4.8	3
chu172	12	104/1.6	152/3.6	120/2.4	2
converta	18	432/6.8	496/6.0	488/4.8	3
ebergen	18	280/5.6	344/4.8	312/4.8	2
full	16	224/5.2	240/4.8	240/4.8	2
hazard	12	296/6.6	256/4.8	232/4.8	2
hybridf	80	274/6.6	352/4.8	336/4.8	2
pe-send-ifc	117	1232/12.2	1832/6	1408/6	5
qr42	18	280/5.6	344/4.8	312/4.8	2
vbe10b	256	1008/10.0	800/4.8	744/4.8	7
vbe5b	24	272/4.2	240/3.6	240/3.6	3
wrdatab	216	824/4.8	840/4.8	760/4.8	4
sbuf-send-ctl	27	408/5.2	696/4.8	320/3.6	4
pr-rcv-ifc	65	1176/9.8	1640/6.0	1144/4.8	3
master-read	2108	1016/6.4	880/4.8	824/4.8	4
read-write	315	740/7.6	(2)	608/6	4
tsbmsi	1023	(4)	960/4.8	928/4.8	3
tsbmsiBRK	4729	(4)	(3)	1648/4.8	2
pmcm1	26	(1)	(1)	304/4.8	3
pmcm2	13	(1)	(1)	160/3.6	3
combuf1	32	(1)	(1)	480/4.8	3
combuf2	24	(1)	(1)	456/4.8	4
sing2dual-inp	65	(1)	(1)	386/4.8	4
sing2dual-out	204	(1)	(1)	648/3.6	3

(1) : Extended SG
(2) : Must add state signals (not handled in version 2.3)
(3) : can be handled with the latest version
(4) : Input file in SG format

piler [26]. In our implementation, we use the ESPRESSO minimizer from SIS [19] to produce the SOP logic implementation.[3]

We have applied our implementation to generate hazard-free implementations for two sets of benchmarks. A first set of benchmarks is taken from either [7], [1] or our own industrial designs. These benchmarks are given as SG's that are already transformed to satisfy the CSC property. For this set of SG's, two tools developed by Lavagno [7] and Beerel [1] are available to us for comparisons. Results concerning this first set of benchmarks are reported in the first part of Table II. A second set of benchmarks is generalized nonfree-choice STG's [22]. The circuits *pmcm1, pmcm2, combuf1,* and *combuf2* are interface circuits from a mobile terminal design [14]. The circuits *sing2dual-inp* and *sing2dual-out* are interface circuits for performing switchable single-rail to dual-rail conversion. These circuits are needed for the design of an asynchronous DCC decoder [20], [24]. Some of them are not distributive or not single traversal. But all of them satisfy the trigger requirement. We have carefully simulated these industrial designs at the gate-level using VERILOG and at the transistor level using SPICE. For these designs, no comparison is currently possible. Results concerning this second set of benchmarks are reported in the second part of Table II.

The obtained circuits for all three methods are realized using the SIS library. The area and performance estimates are derived from this library too, using the same strategy chosen in [1]. Since the SIS library contains AND gates up to four inputs with some inputs inverted, we extrapolated the area of AND gates with more than four inputs from smaller gates, and added an area and delay penalty equivalent to breaking up the large AND gate into two smaller ones. For the tool developed by Beerel [1], these two small AND gates must be considered to form one atomic gate to avoid the introduction of hazards. For the tool developed by Lavagno [7], either the two small AND gates are considered as one atomic gate or delays must be readjusted to avoid hazards. For our method, as we primarily allow hazards to appear in a single gate, then hazards may also appear in a network of gates that substitute a single high-fan-in gate. To guarantee the equivalence between this network of gates and a single gate, only certain classes of hazards are allowed. This problem must be tackled in our future work. The last column of Table II indicates the maximum fan-in of the AND gates used in our implementation for the corresponding benchmark.

The delay of the circuit is taken to be the maximum path delay through a signal network. Whenever an inverter

[3] We use the ESPRESSO command from inside SIS that performs two-level minimization heuristically. Improved results can still be obtained by using the ESPRESSO-EXACT minimizer [17] instead.

is needed at some input of an AND gate, it is modeled as a separate INV gate. It is important to remark that, in all of the tested examples, delay compensation in the N-SHOT architecture is *never* required.

Concerning the implementations obtained from the SYN tool (version 2.3) of Beerel, two-level implementations for sets and resets were almost always generated. This explains why results are often similar to those produced by ASSASSIN. Differences are explained because some optimizations present in SYN are not yet implemented in ASSASSIN. For *read-write* and *tsbmiBRK*, SYN version 2.3 either requires the insertion of new state signals (which is not handled in SYN version 2.3) or generates a memory space problem.

Concerning the implementations obtained from the SIS tool of Lavagno, delay lines must be inserted to solve hazards, implying overhead in both area and delay. Both *tsbmsi* and *tsbmsiBRK* are given in SG format, and hence cannot currently be handled by the SIS tool.

On several examples, our method produced significantly better results. Consider, for example, the benchmarks *pe-send-ifc*, *wrdatab*, *sbuf-send-ctl*, and *pr-rcv-ifc*. On these circuits, our approach produced noticeably smaller circuits than SYN which required extra internal hardware to ensure proper acknowledgment. On the same circuits, and others, our method generally produced faster circuits than SIS which required delay insertions that lengthen the critical path to ensure that the circuit is hazard-free.

VII. Conclusions

We have presented a new architecture that permits us to generate optimal hazard-free implementations for a very large class of state graphs, namely, any semimodular state graphs with input choices, both distributive and nondistributive, that satisfy the complete state coding property and the trigger requirement, which is practically always satisfied. Using this N-SHOT architecture, a solution is guaranteed to exist without the addition of state signals. Both set and reset functions are implemented in a two-level form. No constraints are added for their implementation. The determination of a possible local delay compensation is done without any inspection of the state graph. It depends purely on the delays associated with the set and reset logic. This delay requirement can always be satisfied, and is nearly always negligible. When a delay line is necessary, it is placed in parallel to the set and reset logic, i.e., out of their critical paths. Thus, the overhead, if any, is indirect. We have also presented optimizations of this new architecture, related to state graph properties. These optimizations permit us either to eliminate the MHS flip-flop, yielding an SOP implementation as final circuit, or to replace the MHS flip-flop by a C element.

One remaining problem is the implementation of high-fan-in AND gates which are not available in the given library. We plan to solve this problem in two different ways: 1) extending the N-SHOT architecture to permit multilevel logic for implementing the set and reset functions, and 2) inserting new state signals at the SG level. We also plan to extend our method in order to reach the optimized N-SHOT architecture as much as possible in the generation of hazard-free implementations.

Acknowledgment

The authors would like to acknowledge the valuable suggestions from the reviewers.

References

[1] P. A. Beerel and T. H. Meng, "Automatic gate-level synthesis of speed-independent circuits," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1992.
[2] T. J. Chaney, "Measured flip-flop responses to marginal triggering," *IEEE Trans. Comput.*, vol. C-32, pp. 1207–1209, Dec. 1983.
[3] T. A. Chu, "Synthesis of self-timed VLSI circuits from graph-theoretic specifications," Ph.D. dissertation, M.I.T., June 1987.
[4] T. Jackson and A. Albicki, "Analysis of metastable operation in D latches," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1392–1404, Nov. 1989.
[5] T. Kolks, S. Vercauteren, and B. Lin, "Control resynthesis for control-dominated asynchronous designs," in *Int. Symp. Adv. Res. Asynchronous Circuits Syst.*, Mar. 1996.
[6] A. Kondratyev, M. Kishinevsky, B. Lin, P. Vanbekbergen, and A. Yakovlev, "Basic gate implementation of speed-independent circuits," in *Proc. Design Automation Conf.*, June 1994.
[7] L. Lavagno, K. Keutzer, and A. Sangiovanni-Vincentelli, "Algorithms for synthesis of hazard-free asynchronous circuits," in *Proc. Design Automation Conf.*, 1991, pp. 302–308.
[8] B. Lin, G. de Jong, and T. Kolks, "Hierarchical optimization of asynchronous circuits," in *ACM/IEEE Design Automation Conf.*, June 1995.
[9] B. Lin and S. Devadas, "Synthesis of hazard-free multi-level logic implementations under multiple-input changes from binary decision diagrams," *IEEE Trans. Computer-Aided Design*, Aug. 1995.
[10] B. Lin, C. Ykman-Couvreur, and P. Vanbekbergen, "A general state graph transformation framework for asynchronous synthesis," in *European Design Automation Conf.*, Sept. 1994.
[11] A. J. Martin, *Formal Program Transformations for VLSI Circuit Synthesis*. Reading, MA: Addison-Wesley, 1990.
[12] S. M. Nowick and B. Coates, "Uclock: Automated design of high-performance unclocked state machines," in *Proc. Int. Conf. Comput. Design*, Oct. 1994, pp. 434–441.
[13] S. M. Nowick and D. L. Dill, "Asynchronous state machine synthesis using a local clock," in *Proc. Int. Workshop Logic Synthesis*, May 1991.
[14] L. Philips, I. Bolsens, B. Vanhoof, J. Vanhoof, and H. De Man, "Silicon integration of digital user-end mobile communication systems," in *Proc. IEEE Int. Conf. Commun.*, May 1993.
[15] R. Puri and J. Gu, "Area efficient synthesis of asynchronous interface circuits," in *Proc. Int. Conf. Computer Design*, Oct. 1994, pp. 212–216.
[16] F. U. Rosenberger, C. E. Molnar, T. J. Chaney, and T. P. Fang, "Q-modules: Internally clocked delay-insensitive modules," *IEEE Trans. Comput.*, vol. 37, pp. 1005–1018, Sept. 1988.
[17] R. Rudell, "Logis synthesis for VLSI design," Tech. Rep. UCB/ERL M89/49, Berkeley, CA, 1989.
[18] C. L. Seitz, *System Timing*. Reading, MA: Addison-Wesley, 1980.
[19] E. M. Sentovich, L. Lavagno, C. W. Moon, P. R. Stephan *et al.*, "Sis: A system for sequential circuit synthesis," Report UCB/ERL/M92/41, 1992.
[20] K. van Berkel, private communications, Dec. 1993.
[21] P. Vanbekbergen, B. Lin, G. Goossens, and H. De Man, "A generalized state assignment theory for transformations on signal transition graphs," *J. VLSI Signal Processing*, vol. 7, pp. 101–116, Feb. 1994.
[22] P. Vanbekbergen, Ch. Ykman-Couvreur, B. Lin, and H. de Man, "A generalized signal transition graph model for specification of complex interfaces," in *Proc. European Design Automation Conf.*, Feb. 1994, pp. 378–384.
[23] V. Varshavsky, M. Kishinevsky, V. Marakhovsky, V. Peschansky, L. Rosenblum, A. Taubin, and B. Tzirlin, *Self-Timed Control of Concurrent Processes*. Dordrecht, The Netherlands: Kluwer Academic, 1990.
[24] S. Vercauteren, "Interface design for switchable single-rail to dual-rail conversion," EXACT Int. Rep., IMEC, May 1994.
[25] Ch. Ykman-Couvreur and B. Lin, "Optimized state assignment for asynchronous circuit synthesis," in *2nd Working Conf. Asynchronous Design Methodologies*, May 1995.
[26] Ch. Ykman-Couvreur, B. Lin, and H. D. Man, "Assassin: A synthesis system for asynchronous control circuits," Tech. Rep., IMEC User and Tutorial Manual, Sept. 1994.
[27] K. Y. Yun and D. L. Dill, "Automatic synthesis of 3D asynchronous state machines," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1992.

Milton H. Sawasaki received the Electronics Engineering degree from the University of São Paulo, Brazil, in 1985 and the Master's and Ph.D. degrees in electronics engineering from the Katholieke Universiteit of Leuven, Belgium, in 1990 and 1994, respectively.

He worked at IMEC while pursuing his graduate work at Leuven. Currently, he is a Research Fellow at the Division of VLSI System Design Methodologies, Laboratory of Integrated Systems, Department of Electronics Engineering, University of São Paulo, where he is continuing research in the area of asynchronous design.

Chantal Ykman-Couvreur is a Mathematician. She worked at Philips Research Laboratory of Belgium from November 1979 to June 1991. Her main activities were concentrated on information theory and coding, cryptography (public-key cryptosystems, authentication methods and digital signatures, factorization methods and discrete logarithms, fast generation of large prime numbers for cryptographic applications, scrambling–unscrambling methods for pay-TV systems), multilevel logic synthesis for VLSI circuits (research and development of a new multilevel logical optimization system, called PHIFACT, and based on factorization of Boolean functions). She joined IMEC in September 1991, to develop techniques and implement new tools in the ASSASSIN compiler for both specifications and synthesis of asynchronous control circuits. She is currently working on the hardware/software codesign of systems at the chip level.

Bill Lin received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 1985, 1988, and 1991, respectively.

From 1992 to 1997, he was at the Inter-University Micro-Electronics Center (IMEC), Leuven, Belgium, where he headed the System Control and Communications Group that worked on various aspects of system design technology. He has previously held summer positions at Hewlett Packard, Hughes Electronics, and Western Digital. Since 1997, he has been with the Department of Electrical and Computer Engineering, University of California, San Diego, where he is affiliated with the Center for Wireless Communications.

In the fields of design automation and VLSI systems, Dr. Lin has authored or coauthored more than 70 journal and conference papers. In 1987, he received a Best Paper Award at the DAC Conference. In 1989 and 1990, respectively, he received a Distinguished Paper Citation at the IFIP VLSI Conference and at the ICCAD Conference. In 1994, he received a Best Paper nomination at the DAC Conference. In 1996, he received the Best Paper Award in the IEEE TRANSACTIONS ON VLSI SYSTEMS. He has served on panels and given invited presentations at several major international conferences, including the ICCAD Conference, the EDTC Conference, and the International Workshop on Hardware–Software Codesign. He has also served on the Program Committees of several major international conferences, including the DAC and the EDTC Conferences.