Timing Verification of Sequential Dynamic Circuits

David Van Campenhout, *Student Member, IEEE*, Trevor Mudge, *Fellow, IEEE*, and Karem A. Sakallah, *Fellow, IEEE*

***Abstract*— This paper addresses static timing verification for sequential circuits implemented in a mix of static and dynamic logic. We restrict our focus to regular domino logic and footless domino logic, a variant of domino logic. First we derive constraints for proper operation of dynamic gates. An important observation is that for dynamic gates, input signals may start changing near the end of the evaluate phase without compromising correct operation. This gives the circuit designer extra flexibility. We present two verification methods. Both are based on the Sakallah–Mudge–Olukotun (SMO) model for static timing analysis of sequential circuits. The first method models dynamic gates explicitly. The signals at the terminals of the dynamic gates are modeled by five events: the earliest/latest, rising/falling transitions, and a fifth event that models the occurrence of a spurious rising transition. The second method applies the original SMO model after a preprocessing step that computes the combinational delays. A postprocessing step checks the constraints specific to dynamic gates. The relationship between both methods is studied. We show that the second method may result in a more conservative analysis than the first method, but at a lower computational cost. We also examine a less aggressive set of constraints, which disallows spurious transitions. A detailed example illustrating the important features of the model is presented, and an electrical simulation of that circuit is performed. The results demonstrate the practical relevance of the methods.**

***Index Terms*—Modeling, timing verification.**

I. Introduction

HIGH-performance microprocessors employ advanced circuit techniques to help meet their performance objectives. Critical sections of the design are often implemented in dynamic logic. Domino logic [4], [16], a popular style of dynamic logic, has the advantage of small area and fast operation over complementary static logic. However the use of domino logic has been restricted mainly to full custom designs, in part because of the difficulty of verification. Not only do electrical effects such as charge sharing need to be verified, but also the timing of the circuits is critical. In the absence of good timing models, designers often have to depend solely on electrical simulators such as Spice [5] to verify their designs. This paper addresses static timing verification for sequential circuits implemented in a mix of static and dynamic logic. We consider two popular styles of dynamic logic: regular domino logic, and a variant of domino logic called footless domino.

The characteristic timing constraint for domino gates was stated in [4]: “All nodes can make at most only a single (rising) transition and then must stay there until the next precharge.” Most work in static timing analysis of sequential circuits has not considered dynamic logic. Also, most work on timing verification of dynamic logic has concentrated on the timing constraints for this logic, and has disregarded the issue of incorporating these constraints in a general static timing analysis framework. One explanation for this lacuna is that dynamic logic has been applied only to critical sections of designs. Synthesis of these sections has been a manual task and structure has been imposed to make timing analysis easier. Venkat *et al.* [13], [14] described a timing verification methodology for domino circuits. The focal points of their work are the identification of dynamic nodes, constraint generation for verifying the operation of the dynamic logic gates, and handling gated clocks. However, how domino gates can be handled during the actual static timing analysis, was not addressed. Narayanan *et al.* [6] investigated self-resetting CMOS (SRCMOS) [3], [15], a form of dynamic logic closely related to domino logic. They described how to augment an existing static timing analysis system to handle SRCMOS. In [12], we described methods for the static timing verification of circuits implemented from a mix of static and domino logic. Our methods were built on a standard static timing analysis framework [8]. In this work the analysis is refined and extended to handle footless domino logic.

We present a set of timing constraints for domino and footless domino circuits. Our work is distinguished from related work in the following: We present a systematic analysis of the requirements governing proper operation of domino and footless domino circuits. Our formulation is more general and applies to circuits that freely mix static with dynamic logic. Furthermore our formulation fits in a framework for static timing analysis of latch-based circuits and can therefore be used for timing verification of circuits that use cycle borrowing. As a byproduct of our analysis we have identified a behavior that is considered invalid under textbook domino timing constraints [16], but actually does not exclude correct operation. The next section summarizes the Sakallah–Mudge–Olukotun (SMO) model for static timing analysis of sequential circuits. The operation of dynamic logic is described in Section III. In Section IV, the timing behavior of dynamic gates is analyzed in detail and constraints for proper operation are presented.

Manuscript received August 21, 1997; revised October 19, 1998. This work was supported in part by the Semiconductor Research Corporation (SRC) under Contract 95-DJ-338, by the Advanced Research Projects Agency (ARPA) under Contract DAAH04-94-G0327, and by the National Science Foundation (NSF) under Grant MIP-9404632. This paper was recommended by Associate Editor T. Szymanski.

D. Van Campenhout is with Advanced Computer Architecture Laboratory, The University of Michigan, Ann Arbor, MI 48109 USA (e-mail: davidvc@eecs.umich.edu).

T. Mudge is with Advanced Computer Architecture Laboratory, The University of Michigan, Ann Arbor, MI 48109 USA (e-mail: tnm@eecs.umich.edu).

K. A. Sakallah is with Advanced Computer Architecture Laboratory, The University of Michigan, Ann Arbor, MI 48109 USA (e-mail: karem@eecs.umich.edu).

Publisher Item Identifier S 0278-0070(99)02969-3.

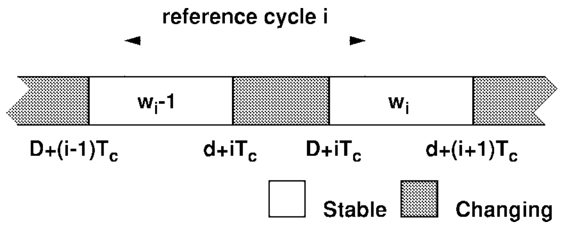


Fig. 1. Signal in the SMO model.

The actual verification methods are described in Section V. A detailed example illustrating the model's important features is provided in the Section VI, followed by concluding remarks in Section VII.

II. Static Timing Analysis

A comprehensive model for analyzing the temporal behavior of synchronous sequential circuits, the SMO model, is described in [8]–[10]. These sequential circuits are composed of an interconnection of static combinational logic and clocked storage elements, referred to as synchronizers. The synchronizers can either be level-sensitive (latches), or edge-triggered (flip-flops). The SMO model assumes a multiphase clocking system with common clock period T_c. The combinational logic between each pair of synchronizers is characterized by its minimum and maximum propagation delays. Each synchronizer is characterized by its setup time and hold time, the phase and the minimum and maximum delay (skew) of its clock signal, and the minimum and maximum delay between the data input and the output. The worst-case time complexity for verifying the timing of a circuit with $|L|$ latches and $|e|$ combinational edges connecting the latches is $O(|L||e|)$.

A key feature of this model is that the analysis is performed modulo T_c. A reference clock cycle is associated with each synchronizer. Referring to Fig. 1, the output of the synchronizer undergoes the following sequence during the reference cycle. First, it holds w_{i-1} the stable value that was latched at the end of the previous cycle. As soon as the clock enables the synchronizer, the output may start changing. The event time of the earliest change is denoted by d (earliest departure time). That event can be triggered either by the synchronizer's clock or by the synchronizer's data input, depending on their temporal relationship. Next, the output signal settles to its stable value, w_i. This is the value that is latched at the end of the cycle. The latest time this happens is denoted by D (latest departure time). For the latching to happen reliably, the data input must respect a setup and a hold constraint with respect to the latching edge of the clock. Hence, in the SMO model signal waveforms are modeled with two events in the reference clock cycle.

III. Dynamic Logic

Combinational circuits in static CMOS logic have the property that at any time that the circuits' signals have settled, the logic value of the outputs depend solely on the logic value of the inputs. This property does not hold for dynamic logic. In dynamic logic, circuit operation is divided in two phases.

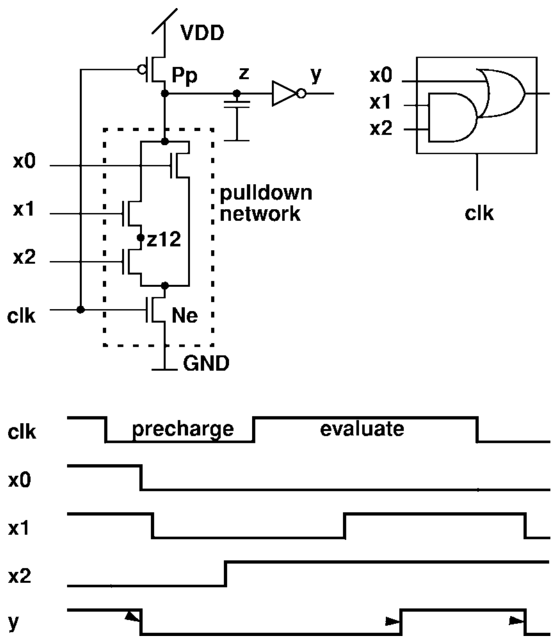


Fig. 2. ANDOR21 Domino gate with sample waveforms.

During the precharge phase, the circuit's output is brought to one logic value, regardless the logic values of the inputs. During the evaluate phase, the output conditionally makes a transition to the other logic value, depending on the logic values of the inputs. The phases are defined by a clock signal that is provided to every dynamic gate.

Domino logic is one of the most popular styles of dynamic logic and is the basis of many derivatives. Fig. 2 shows an ANDOR21 domino gate and some sample waveforms. The gate consists of a pulldown network (PDN), an evaluate transistor N_e, which is part of the PDN, a precharge transistor P_p, and a static CMOS inverting buffer. The operation of the circuit is as follows. When the clock *clk* is low, the internal node z is precharged, and the output node y is set to zero. The period in which *clk* is low is called the *precharge phase*. A rising transition on the clock conditionally discharges the internal node z through the PDN. The PDN consists of the n-transistors gated by the inputs. The values of the inputs determine whether the discharge actually takes place. This phase is called the *evaluate phase*. Once z is discharged, it will stay low for the rest of the evaluate phase even if the inputs change subsequently. Therefore, either the inputs have to settle to their stable values before the start of the evaluate phase, or they can settle to their stable high value by making single rising transitions during the evaluate phase.

The inverter at the output of the gate is included for several reasons. First, it is required for proper operation of a chain of domino gates. Signals produced by domino gates that lack the inverter cannot be fed into another domino gate because they violate the requirement stated above. This can be seen as follows. The output of a domino gate that lacks the inverter, is reset to one during precharge. This high value will still be present at the beginning of the evaluate phase of the next gate in the chain, and thus that next gate will be inadvertently discharged. Consequently, that next gate becomes insensitive

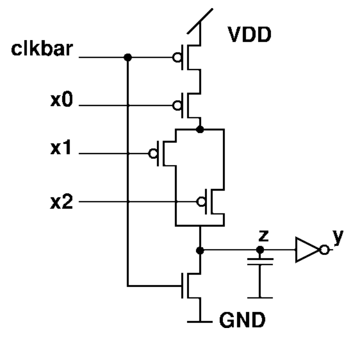


Fig. 3. Domino p-MOS gate.

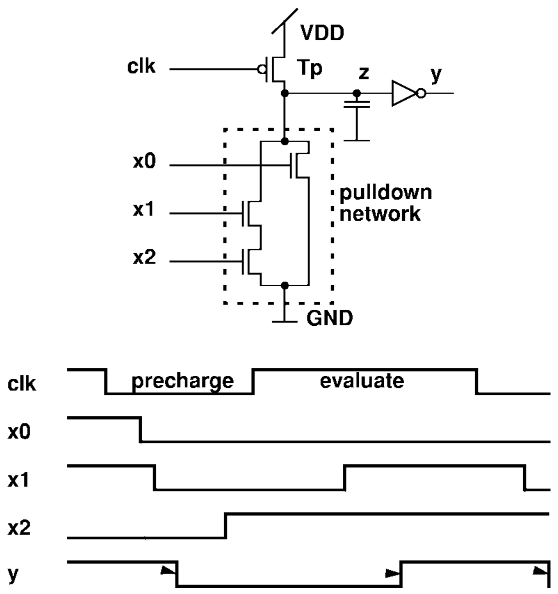


Fig. 4. Footless domino gate with sample waveforms.

to transitions on its inputs during evaluate. Second, the internal node z is a weak node. This is indicated in the figure by the capacitance at node z. When the clock is high, the high value on that node is not driven. The inverting buffer separates that dynamic node from the rest of the circuit, thus alleviating charge-sharing problems and minimizing capacitive coupling. A consequence of the inverting buffer is that a domino gate can only implement a noninverting function of its inputs.

Fig. 3 shows a domino p-MOS gate, which is the dual of the circuit from Fig. 2 and which implements the same function. However, in many CMOS technologies, the performance of such a dual gate is far inferior to that of the original gate due to the poor characteristics of the p-transistors. The following discussion will only be concerned with the type of gate shown in Fig. 2 (domino n-MOS).

A common variant of domino logic, termed *footless domino*, is shown in Fig. 4. This type of gate differs from the standard domino gate only in the absence of the evaluate transistor. The operation is similar to that of regular domino logic, except for the precharge phase. Precharge will only take place if all paths in the PDN are broken while *clk* is low. Due to the absence of the evaluate transistor, this requirement is no longer trivially satisfied. Another consequence is that during precharge, a falling transition can propagate through a chain of footless domino gates, which is generally undesirable. This can be seen in the figure, as the falling transition of y is not triggered by *clk*, but by $x0$ and $x1$. Also note that during the beginning of the precharge phase a conducting path from positive supply (VDD) to ground (GND) may exist, leading to increased power consumption over the regular domino version. Both these two problems are usually alleviated by clocking each stage in a chain of footless domino gates with a clock that is slightly delayed with respect to the clock of the preceding stage. The advantages of footless domino circuits are that they require less area and are faster.

IV. Modeling Dynamic Gates

A. Waveform Model

In this section we introduce a five-event periodic waveform model that captures the relevant temporal behaviors of dynamic gates. This waveform model will be used to represent the data input signals and output signal of dynamic gates. Four of these events are the earliest rising and falling transitions, and the latest rising and falling transitions. The corresponding event times are denoted by d^R, d^F, D^R, D^F, respectively, and are ordered as follows: $d^F \leq D^F < d^R \leq D^R$. For input signals the d's are replaced by a's (arrival times). This model is depicted in Fig. 5. A reference cycle with respect to a dynamic gate starts with the clock signal triggering the output signal to make a falling transition some time in the interval $[d^F, D^F]$. If the gate had not made a rising transition in the previous reference cycle, the falling transition does not take place. If the gate is to evaluate to “1” during the current reference cycle, its output will make a single rising transition in $[d^R, D^R]$. The high value is valid at least in $[D^R, d^F + T_c]$. Alternatively, the output signal remains low throughout the rest of the reference cycle: at least during $[D^F, d^R + T_c]$. However, as the analysis presented in Section IV-B will show, correct operation is possible even when the gate produces a glitch. That is, although the gate is to evaluate to “0”, it does make a single rising transition late in the reference cycle. By considering this phenomenon, which we term a *glitching zero*, constraints for correct operation can be derived that are looser than those previously published. Although the glitching zero behavior may occur only rarely in practical designs, we will present an example together with a detailed analysis that demonstrates this behavior (see Section VI). For a glitching zero not to affect the logic operation of the circuit, it should not propagate to the synchronizers. To analyze the timing in the presence of glitching zeros, a fifth parameter d^{R0}, which denotes the earliest time at which such a spurious rising transition occurs, is required. Note that since falling transitions are only triggered by falling transitions of the clock signal, the event times of glitches corresponding to the falling transition are the same as those during normal operation. The rising transition of the glitch must take place strictly later than the rising transition during normal operation (otherwise the glitch cannot be distinguished from a valid “1”), that is, $D^R < d^{R0} + T_c$. The effect of a glitching zero is that the period in which the low value of the signal is valid is shortened to

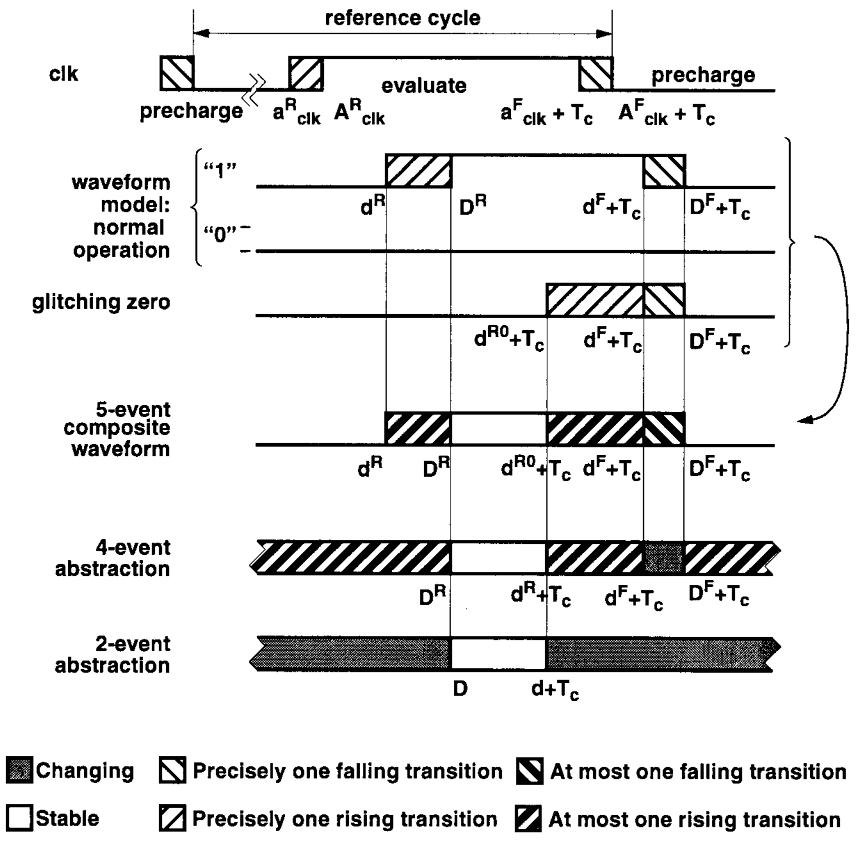

reference cycle
clk
precharge
evaluate
precharge
waveform model: normal operation
"1"
"0"
glitching zero
5-event composite waveform
4-event abstraction
2-event abstraction
Changing
Stable
Precisely one falling transition
Precisely one rising transition
At most one falling transition
At most one rising transition

Fig. 5. Enhanced waveform model for dynamic gates: output signal.

$[D^F, d^{R0} + T_c]$. Hence, considering both the high and the low values, the signal is valid at least during the interval $[D^R, \min\{d^{R0} + T_c, d^F + T_c\}]$. The value of the signal in this "valid" interval, will be referred to as the signal's *stable value*. Not every dynamic signal exhibits glitching. The absence of glitching is modeled by $d^{R0} = +\infty$. In case $d^{R0} \neq +\infty$, it satisfies the following ordering: $D^R < d^{R0} + T_c \leq d^F + T_c$. Fig. 5 illustrates the model. The figure shows the waveform of the clock to a domino gate, and the waveforms that can be exhibited by the gate's output. The waveform of the clock is at the top of the figure. The next two waveforms correspond to the output evaluating to "1" and "0," respectively. The fourth waveform shows the case where the output exhibits a glitching zero. Composing these three waveforms leads to the five-event waveform. Note that not all events with respect to a single reference cycle are shown. Instead we show the rising events in the current reference cycle and the falling and glitching events from the next reference cycle (as indicated by the term $+T_c$ on those event times). This simplifies the figure, as it reduces the number of cases to be considered.

The five-event waveform can be abstracted to the four-event waveform model proposed in [12]. Again, this is illustrated in the figure. In the four-event model, the event times are ordered $d^F \leq D^F$ and $d^R \leq D^R$, but $D^F < d^R$ does not necessarily hold. Also, in this model, the signal is stable in $[\max(D^R, D^F), T_c + \min(d^R, d^F)]$, but not necessarily zero. It will be shown later that it is important to be able to capture the period in which dynamic signals are guaranteed to be low. The four-event model does not handle glitches explicitly, but they can be taken into account by setting $d^R = d^{R0}$, as shown in the figure. The waveform can be abstracted further to the two-event model, which is also shown in the figure.

1) Static Signals: Input signals to dynamic gates that are produced by static logic are modeled by the four-event abstraction. The major difference with dynamic signals is that there is no longer an interval in which the signal is guaranteed low. Also, the event order is looser: the only requirements on the event order are $d^R \leq D^R$ and $d^F \leq D^F$.

B. Proper Operation

Prior to deriving the constraints that govern the operation of dynamic gates within a sequential circuit, we define the notion of proper operation.

Definition 1—Proper Operation of Dynamic Gates: A dynamic gate operates properly in a synchronous sequential circuit if the stable value of its output is determined solely by the stable values of its inputs.

We assume that input signals to a dynamic gate are independent and represented by the five-event waveform model introduced in the previous section. Static inputs (inputs produced by static logic) are represented by the four-event waveform model. Under this assumption necessary and sufficient constraints for proper operation of dynamic gates can be derived. These constraints dictate temporal relationships among the data input signals and between the data input signals and the clock signal.

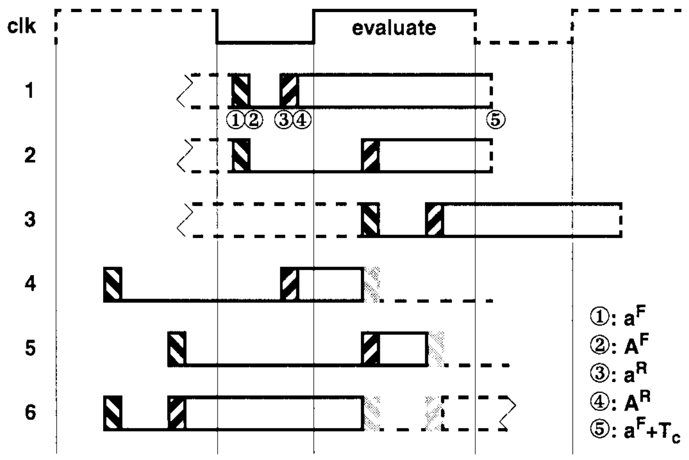


Fig. 6. Temporal relationships between a data input and the evaluate phase.

While going over the constraints it is useful to refer to Fig. 6. The figure enumerates all the different relationships between A^R and A^F of a dynamic data input and clock edges delineating the evaluate phase of the dynamic gate, that can lead to proper operation. Segments in dashed lines or gray pertain to the next or the previous reference cycle. The first case shown is that when both A^R and A^F occur during the precharge phase. In total there are six cases, and they have in common that the interval $[A^R, a^F + T_c]$, in which the input attains its high stable value, overlaps with the evaluate phase of the current reference cycle. This still leaves quite some latitude and both high stable values from the previous reference cycle (case 3), and high stable values from the next reference cycle (case 6), may overlap with the evaluate phase of the current reference cycle. Notice that if the waveforms were offset even more, such that values from two cycles apart would overlap with the current evaluate phase, correct operation would no longer be possible (there would no longer be overlap between the high stable value of the current reference and the current evaluate phase.) Each of the six cases can be differentiated further by considering a^R, a^F, and the fifth signal parameter, a^{R0}. These were omitted in order not to clutter the figure.

Consider a regular or footless domino gate whose inputs (including the clock input) are referred to by i, its clock input by *clk*, and a path in the PDN by p. To include the *clk* input in the set of inputs along a path p in the PDN of a footless domino gate, the notation $p' = p \cup clk$ is used. To differentiate dynamic signals from static signals we use the prefix "*dyn*;" the clock signal is also considered dynamic: the key property is that dynamic signals are guaranteed low during $[A^F, a^R]$.

1) Integrity of the High Output Value (IHV): Proper operation necessitates that the gate's output will rise in the current reference cycle, if stable values of the current reference cycle of the inputs indicate so. Therefore, the high stable interval of each input must overlap with the current evaluate phase. However, this alone is not sufficient. The stable high values of inputs along the same path p in the PDN have to overlap as well. For electrical reasons, the period of overlap must be at least T^{PWH}. This requirement can be formulated as a hold constraint with respect to the latest rising input along the path.

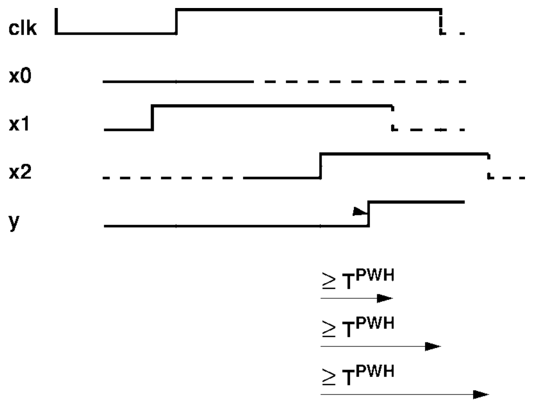


Fig. 7. Integrity of the high output value: IHV for each input along the path $x_1 - x_2 - clk$.

IHV:

$$a_i^F + T_c \geq \max_{j \in p'} \left\{ A_j^R \right\} + T^{\text{PWH}}.$$

Note that the constraint is to be satisfied for all inputs, including the clock input ($i = clk$). Since the max-operator concerns p', the clock input is always considered in the right hand side, for both types of gates. Hence the constraint also ensures overlap with the evaluate phase.

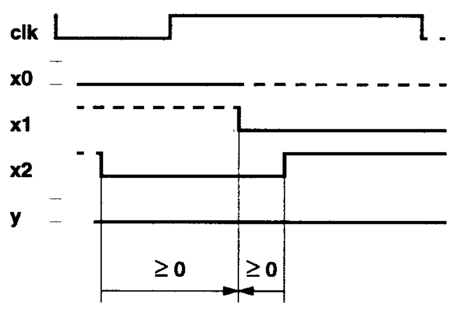
The constraint is illustrated in Fig. 7 for the domino gate from Fig. 2. Waveform segments associated with the current reference cycle are indicated with solid lines; segments associated with the next or the previous reference cycle are in dashed lines. The latest rising input along the path $x1 - x2 - clk$ is $x2$. The IHV constraint for each of the inputs along this path are shown in the figure.

2) Integrity of the Low Output Value (ILVP and ILVN): Proper operation necessitates that a low output value will be observed at the gate's output when the stable values of the current reference cycle of the inputs indicate so. During the precharge phase the gate's output will be brought to zero, but this low output value can be corrupted in two ways. First, an old high value on an input may still persist during the current evaluate phase and trigger the gate to inadvertently switch to one. Such a transition is always illegal because the output will remain high throughout the evaluate phase *and* this transition occurs *earlier* than a valid rising transition of the output triggered by the same input. Hence, if an old high value on an input i still persists during the current evaluate phase, at least one other input on each path through i in the PDN is to be guaranteed low. Only dynamic inputs can be guaranteed low in the interval $[A^F, a^R]$. The constraint that precisely tests this property is as follows.

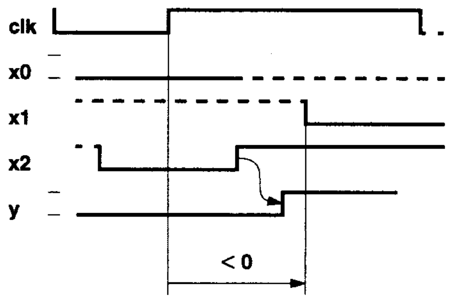
ILVP:

$$A_i^F < \max\left\{ a_{\text{clk}}^R, \max_{\text{dyn}\, j \in p'} \left\{ a_j^R \,\middle|\, \left(A_j^F < A_i^F \right) \right\} \right\}.$$

The constraint must hold for every data input, not for the clock input. This constraint also applies to footless domino gates and will be referred to as ILVP, as it protects the low value from corruption by signals from a previous reference cycle.

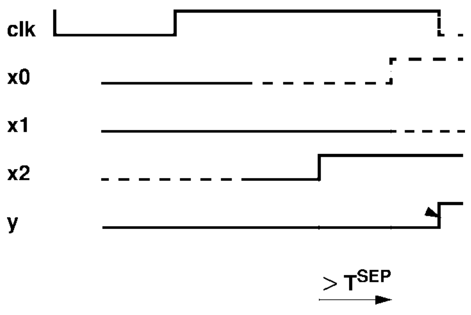


(a)

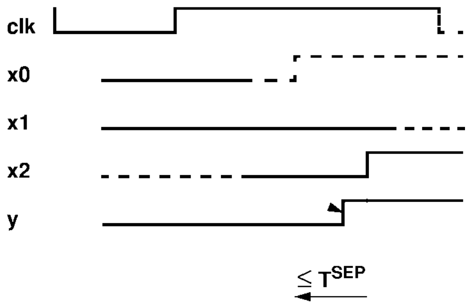


(b)

Fig. 8. ILVP: Integrity of the low output value: (a) x_1 satisfies constraint and (b) x_1 violates constraint.



(a)



(b)

Fig. 9. ILVN1: Integrity of the low output value: (a) x_0 satisfies constraint and (b) x_0 violates constraint.

The constraint is illustrated in Fig. 8, again for the domino gate from Fig. 2. The stable values of the inputs are $x_0 = 0$, $x_1 = 0$, $x_2 = 1$; hence, the stable value of y should be zero. In Fig. 8(a) an old logic one on input x_1 is still present during the current evaluate phase, but does not affect proper operation since input x_2 along the same path is guaranteed to be low. If the old high value persisted longer, as shown in Fig. 8(b), it may enable input x_2 to trigger a spurious rising transition on y. There is no input along the same path which is guaranteed to be low while x_1 makes its latest falling transition, hence that transition must take place before the beginning of the evaluate phase; this is not the case and, hence, ILVP is violated.

A second way in which the low output value may get corrupted is that a high value associated with the next reference cycle triggers an inadvertent output transition. Such a spurious transition (glitching zero) is permissible as long as it can be distinguished from a valid rising transition. This means that if a spurious transition takes place, it must happen *later* than the latest time a valid rising transition is triggered. The next-cycle events that need to be checked for are $a^R + T_c$ and $a^{R0} + T_c$. Either, these events have to take place after the latest valid rising transition is triggered, or at least one other input along each path has to be guaranteed low. This property is precisely expressed by the following:

ILVN1:

$$a_i^R + T_c > T^{\mathrm{SEP}} + \max\left\{\max_j\left\{A_j^R\right\},\right.$$
$$\left.\min_{\mathrm{dyn}\, j\in p}\left\{A_j^F + T_c \,\middle|\, \left(a_i^R < a_j^R\right)\right\}\right\}.$$

ILVN2:

$$a_i^{R0} + T_c > T^{\mathrm{SEP}} + \max\left\{\max_j\left\{A_j^R\right\},\right.$$
$$\left.\min_{\mathrm{dyn}\, j\in p}\left\{A_j^F + T_c \,\middle|\, \left(a_i^{R0} < a_j^R\right)\right\}\right\}$$

where T^{SEP} is a timing constant. These constraints also apply to footless domino gates and will be referred to as ILVN1 and ILVN2, as they protect the low value from corruption by signals from a next reference cycle.

ILVN1 is illustrated in Fig. 9, again for the domino gate from Fig. 2. The stable values of the inputs are $x_0 = 0$, $x_1 = 0$, $x_2 = 1$; hence the stable value of y is zero. In the next reference cycle x_0 goes high, but this occurs so early that the event takes place during the evaluate phase of the current reference cycle of y. Consequently, that early rising transition of x_0 triggers a spurious rising transition of y (glitching zero). Assuming that x_2 is the overall latest rising input it can be seen that in Fig. 9(a) the low output value is not corrupted as the spurious transition takes place at least a separation time constant after the latest valid rising transition. In Fig. 9(b) the ILVN1 is violated: the spurious rising transition takes place before the latest rising transition and, hence, cannot be distinguished from a valid rising transition.

3) Precharge: Constraints ILVP, ILVN1, and ILVN2 ensure that the low output value will not be corrupted by signals from other reference cycles, but it remains to be seen if the gate will correctly settle to that low value during the precharge phase. A regular domino gate precharges correctly if its clock respects a minimum width, T^{PWL}, of the low phase (precharge phase). For footless domino gates this is not sufficient. For

such a gate every path in the PDN must remain broken till the end of the precharge phase, starting from T^{PWL} before the end of the precharge phase. This property can be precisely expressed by two constraints. The first constraints, IPCH1, stipulates that the latest time at which a path becomes broken must be at least T^{PWL} before the end of the precharge phase.
IPCH1:

$$\max\left\{A^F_{\text{clk}}, \max_p\left\{A^F_i \,\middle|\, \forall(\text{dyn}\, j \in p) : \left(\left(A^F_i < A^F_j\right) \vee \left(a^R_j < A^R_i < a^R_{\text{clk}}\right)\right)\right\}\right\} \le a^R_{\text{clk}} - T^{\text{PWL}}.$$

The second constraint, IPCH2, stipulates that no path shall be turned on before the start of the evaluate phase.
IPCH2:

$$\min_p\left\{\max\left\{-\infty, \max_{\text{dyn}\, j\in p}\left\{a^R_j\right\}\right\}\right\} \ge a^R_{\text{clk}}.$$

The term $-\infty$ reflects that if a path exists for which all inputs are static, precharge may fail. Constraint PCH2 only applies to footless domino gates. Constraint PCH1 reduces to $A^F_{\text{clk}} \le a^R_{\text{clk}} - T^{\text{PWL}}$ for regular domino gates.

These findings are summarized below. An important problem, to be addressed by electrical analysis [2], is the determination of the minimum pulse widths and minimum separation times T^{PWH}, T^{PWL}, and T^{SEP}. Furthermore some these parameters actually depend on the discharge path involved.

Constraints governing proper operation of a domino gate: A domino gate operates properly in a synchronous sequential circuit iff the following constraints are satisfied:

a) Integrity of high-output volume (IHV):

$$a^F_i + T_c \ge \max_{j\in p'}\left\{A^R_j\right\} + T^{\text{PWH}}. \tag{1}$$

b) Integrity of low-output volume (ILVP):

$$A^F_i < \max\left\{a^R_{\text{clk}}, \max_{\text{dyn}\, j\in p'}\left\{a^R_j \,\middle|\, \left(A^F_j < A^F_i\right)\right\}\right\}. \tag{2}$$

ILVN1:

$$a^R_i + T_c > T^{\text{SEP}} + \max\left\{\max_j\left\{A^R_j\right\}, \min_{\text{dyn}\, j\in p}\left\{A^F_j + T_c \,\middle|\, \left(a^R_i < a^R_j\right)\right\}\right\}. \tag{3}$$

ILVN2:

$$a^{R0}_i + T_c > T^{\text{SEP}} + \max\left\{\max_j\left\{A^R_j\right\}, \min_{\text{dyn}\, j\in p}\left\{A^F_j + T_c \,\middle|\, \left(a^{R0}_i < a^R_j\right)\right\}\right\}. \tag{4}$$

c) Integrity of Precharge
IPCH1:

$$\max\left\{A^F_{\text{clk}}, \max_p\left\{A^F_i \,\middle|\, \forall(\text{dyn}\, j \in p) : \left(\left(A^F_i < A^F_j\right) \vee \left(a^R_j < A^R_i < a^R_{\text{clk}}\right)\right)\right\}\right\} \le a^R_{\text{clk}} - T^{\text{PWL}}. \tag{5}$$

IPCH2 (only for footless):

$$\min_p\left\{\max\left\{-\infty, \max_{\text{dyn}\, j\in p}\left\{a^R_j\right\}\right\}\right\} \ge a^R_{\text{clk}}. \tag{6}$$

It is easy to see that proper operation of a domino gate consists of exactly the following five requirements:

i) If the stable values of the inputs from the current reference cycle dictate a high output value, the gate's output must rise during the current reference cycle.
ii) If the stable values of the inputs from the current reference cycle dictate a low output value, the low output value set during precharge of the current reference cycle is not to be corrupted by high input values belonging to a previous reference cycle.
iii) If the stable values of the inputs from the current reference cycle dictate a low output value, the low output value set during precharge of the current reference cycle is not to be corrupted by high input values belonging to the next reference cycle.
iv) Every path in the PDN must become open at least some time before the end of the precharge phase.
v) No path in the PDN is to turn on from the time specified in IV, until the start of the next evaluate phase.

These five requirements correspond precisely to our constraints IHV, ILVP, (ILVN1 and ILVN2), IPCH1 and IPCH2, respectively. The concrete expressions for the constraints were explained before.

4) Discussion: We have derived the constraints for proper operation under the assumption that there are no correlations between the input signals. If there are such correlations, the constraints are still sufficient for proper operation, but they may not be necessary. Unfortunately, it is not practical to include input correlations in the analysis in general.

Narayanan, Chappell and Fleischer [6] formulated three types of constraints for SRCMOS: pulse overlap, pulse width, and collision-avoidance constraints. The pulse overlap correspond to IHV. The pulse width constraints with respect to the high pulse are already implied by IHV; those with respect to the low pulse correspond to IPCH1 and IPCH2. The collision-avoidance constraints correspond to ILVP, ILVN1, ILVN2. The main difference between [6] and our analysis, that goes beyond the differences between SRCMOS and domino logic, are that [6] does not allow high stable values that originated from a previous or next reference cycle to overlap the evaluate phase of the current reference cycle. Consequently no spurious transitions have to be considered in [6]. As already shown in [12], by allowing these spurious pulses extra flexibility is gained. The constraints presented in [12] are stricter. This is because we do consider the topology of the PDN in this work.

Puri and Shepard [7] studied the interaction of static and dynamic logic in the context of logic synthesis. A two-phase clocking methodology is assumed. Their synthesis system requires that the latest falling transition on static inputs to a domino gate must take place before the evaluate phase. They also present a relaxed constraint, to be used during resynthesis, which is identical to our ILVP. In their work, spurious transitions are not allowed.

Although the constraints stated above are necessary and sufficient for proper operation of both regular and footless domino gates, they do not limit the time in which a conducting path between VDD and GND may exist in footless domino gates. IPCH1 and IPCH2 ensure that near the end of the precharge phase no path in the PDN will be on for at least a time specified by T^{PWL}. However, in the first part of the precharge phase several paths in the PDN may be on, leading to excessive power consumption. To limit that short circuit period we can impose the following constraint (compare to IPCH1).
SCC:

$$\max\Big\{A^F_{\mathrm{clk}}, \max_p\Big\{A^F_i \,\Big|\, \forall(\mathrm{dyn}\, j \in p): \Big(\Big(A^F_i < A^F_j\Big) \vee \Big(a^R_j < A^R_i < a^R_{clk}\Big)\Big)\Big\} \le a^F_{\mathrm{clk}} + T^{\mathrm{SC}}. \tag{7}$$

SCC ensures that each reference cycle, VDD and GND cannot be shorted through the PDN for a period longer than T^{SC}.

C. Gate Delay Model

We assume that for each gate a set of min/max, input to output delays are available for each input/output pair. The minimum delay from a transition of type T_i (either rising or falling) on input i that triggers a transition of type T_y on output y is denoted by $\delta^{T_iT_y}_{i,y}$, and similarly for the maximum delay $\Delta^{T_iT_y}_{i,y}$. In the case of a single output gate, we simplify the notation to $\delta^{T_iT_y}_i$, $\Delta^{T_iT_y}_i$. Depending on the gate type, not all combinations apply. For example, a two-input regular domino AND-gate is characterized by the set $\{\delta^{RR}_a, \delta^{RR}_b, \delta^{RR}_{\mathrm{clk}}, \delta^{FF}_{\mathrm{clk}}, \Delta^{RR}_a, \Delta^{RR}_b, \Delta^{RR}_{\mathrm{clk}}, \Delta^{FF}_{\mathrm{clk}}\}$. Note that only the clock input can trigger falling transitions on the output. For a static XOR-gate, any transition on the output can be triggered by any transition on an input. Hence all transition-pairs (T_i, T_j) apply. Worst case conditions are assumed for the side inputs. Consider the domino ANDOR21-gate in Fig. 2. The minimum delay δ^{RR}_i is most likely to be exhibited when both discharge paths turn on simultaneously. The maximum delay Δ^{RR}_i is most likely to be exhibited when only a single discharge path turns on while the other path remains off. Note that the differentiation between minimum and maximum delays reflects the differences in delay due to different conditions of the side inputs. This is illustrated with the domino ANDOR21-gate in Fig. 2. Consider the propagation of a rising event on input x_0 to the output. In order for the event to propagate, the clock *clk* must be high and at least one of the other data inputs needs to be low. In case both x_1 and x_2 are low, the only capacitance that needs to be discharged is that at internal node z. The same is true in case x_1 is low and x_2 high. In this case the capacitance at node z_{12} was already discharged before x_0 rises. However when x_1 is high and x_2 low, the capacitance at node z_{12} needs to be discharged through the x_0 transistor as well. This results in a larger delay. For complex gates this effect can be significant. The example also suggests that in general, the delay from a transition on input x_i to output y can be dependent on the logic value of the side inputs. However, taking into account this dependency greatly complicates the timing analysis. The delay characterization of gates is an important and difficult problem to be addressed by electrical analysis, but is beyond the scope of this paper.

D. Propagation Equations

The output signal of a dynamic gate can now be expressed in terms of input signals. Again the assumption is that the input signals are independent and are represented by five-event waveforms. The propagation equations are only valid for inputs that satisfy the constraints for proper operation.

1) Rising Transition: The output of a dynamic gate switches to "1" as soon as one path in the PDN is turned on (see Fig. 2). Any paths that are turned on later, do not affect the gate's output. Along the path that is turned on first, it is the latest rising input that triggers the output transition. Hence

$$d^R = \min_p\Big\{\max_{i\in p}\Big\{a^R_i + \delta^{RR}_i\Big\}\Big\}. \tag{8}$$

The latest time for a rising transition on the output occurs in case only a single path is turned on. There are as many such cases as there are paths. Again input transitions arriving before the clock goes high do not affect the output

$$D^R = \max_i\Big\{A^R_i + \Delta^{RR}_i\Big\}. \tag{9}$$

Notice the latest departure of a rising transition is independent of the gate's functionality whereas the earliest rising event does depend on the functionality.

2) Falling Transition: A falling transition can only be triggered during the precharge phase. For the earliest falling transition, consider the case where only a single path was on. The first input along that path to fall will trigger the output transition. Hence

$$d^F = \max\Big\{a^F_{\mathrm{clk}} + \delta^{FF}_{\mathrm{clk}}, \min_i\Big\{a^F_i + \delta^{FF}_i\Big\}\Big\}. \tag{10}$$

The analysis for determining the latest falling transition for a footless domino gate is the same as for determining IPCH1. Consider the case where a single path in the PDN is on. Such a path will break to the arrival of the falling transition on any of its inputs. For such a transition to be the first input to break the path, it is not to overlap with the interval $[A^F_j, a^R_j]$ in which any dynamic side inputs is guaranteed low. This can be expressed as follows:

$$D^F = \max\Big\{A^F_{\text{clk}} + \Delta^{FF}_{\text{clk}},\ \max_p\Big\{A^F_i + \Delta^{FF}_i \,\Big|\, \forall(\text{dyn}\, j \in p) : \Big(\Big(A^F_i < A^F_j\Big) \vee \Big(a^R_j < A^R_i < a^R_{\text{clk}}\Big)\Big)\Big\}\Big\}. \tag{11}$$

For the case of a regular domino gate (9) and (10) reduce to: $d^F = a^F_{\text{clk}} + \delta^{FF}_{\text{clk}}$, $D^F = A^F_{\text{clk}} + \Delta^{FF}_{\text{clk}}$.

3) Generation of Glitching Zero: The earliest spurious rising transition is triggered by either an early arriving rising transition from the next reference cycle $(a^R_i + T_c)$, or from a spurious rising transition on an input $(a^{R0}_i + T_c)$. In order for such an input event to make the output rise, the side inputs along a path have to be nonzero, i.e., $(a^R_i + T_c)$, or $(a^{R0}_i + T_c)$ must not overlap with the interval $[A^F_j + T_c, a^R_j + T_c]$ in which any dynamic side input is guaranteed low. This can be expressed as follows:

$$d^{R0} + T_c = \min\Big\{\Big\{a^{R0}_i + T_c + \delta^{RR}_i \,\Big|\, \forall(\text{dyn}\, j \in p') : \Big(a^{R0}_i < a^F_j\Big)\Big\},\ \Big\{a^R_i + T_c + \delta^{RR}_i \,\Big|\, \forall(\text{dyn}\, j \in p') : \Big(\Big(a^R_i < a^F_j\Big) \vee \Big(a^F_{\text{clk}} > a^R_i > a^F_j\Big)\Big)\Big\},\ +\infty\Big\}. \tag{12}$$

If the set considered in the right hand side of the equation is empty, $d^{R0} = +\infty$.

V. Timing Verification

For timing verification we propose two approaches. The first approach extends the SMO model by explicitly modeling the dynamic gates. The alternative approach makes use of the basic SMO model. Dynamic gates are taken into account during combinational delay computation and in a postprocessing step which checks constraints specific to dynamic gates.

A. Method 1: Explicit Modeling of Dynamic Gates

In this approach, dynamic gates are modeled explicitly. Data input signals of synchronizers and dynamic gates are modeled by four event times: (a^R, a^F, A^R, A^F). Similarly, the output signal of synchronizers and dynamic gates are modeled by (d^R, d^F, D^R, D^F). Signals produced by dynamic gates have an additional parameter d^{R0}. The combinational delay from element (synchronizer/dynamic gate) j to element i is given by $\delta^{T_jT_i}_{j,i}$, $\Delta^{T_jT_i}_{j,i}$.

B. Method 2: Implicit Modeling of Dynamic Gates

This method consists of three steps:

1) preprocessing: computation of combinational delays;
2) standard SMO timing analysis;
3) postprocessing: verification of all timing constraints associated with dynamic gates.

During a preprocessing phase the combinational delay between each connected pair of synchronizers, $\delta^{T_jT_i}_{j,i}$ and $\Delta^{T_jT_i}_{j,i}$, is computed. In contrast to method 1, these paths may cross dynamic gates (only rising transitions propagate through domino gates). The presence of dynamic gates necessitates the consideration of an additional type of path, namely those paths starting at a transition (rising or falling) of a primary clock phase and entering a dynamic gate through its clock input. Once the combinational delays have been computed, the timing verification reduces to the basic problem addressed by the SMO model. Unlike in method 1, the system of equations solved in Step 2 contains only events associated with the synchronizers as variables. Consequently, Step 2, which has the highest worst-case complexity ($O(|L||e|)$ for a circuit with $|L|$ latches and $|e|$ combinational edges connecting the latches) among the three steps, has the same computational cost as that of analyzing the timing of an equivalent circuit containing only static logic. This makes method 2 attractive for large dynamic circuits. After Step 2, the departure times of the output signals of all synchronizers are known. In the last step, the arrival times of the input signals of all dynamic gates are computed, and the constraints associated with dynamic gates are checked. This can be done with a single traversal of the circuit. Note that the absence of dynamic gates during the SMO analysis no longer necessitates distinguishing between rising and falling transitions as in method 1. For the analysis below, the SMO analysis is assumed to differentiate between transitions. This is also preferred in most practical cases as it can significantly increase accuracy at relatively low cost.

C. Relationship Between Method 1 and Method 2

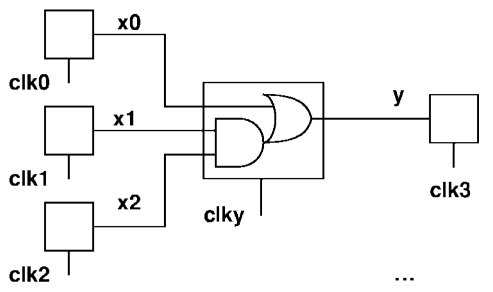
To analyze the relationship between the two methods, consider the circuit shown in Fig. 10. The circuit consists of a single regular domino gate and four level sensitive latches. For sake of generality each latch is clocked with a distinct clock phase. In method 1, each signal shown is modeled explicitly with the five-event model, and the events on signal y relate to those on x_0, x_1, x_2 and *clk*$_y$ via the event propagation equations (8)–(12) for the domino gate. In method 2, the domino gate is reduced to a set of combinational delays during the preprocessing step, as shown in the right-hand side of the figure. During this preprocessing step the temporal relationship among the inputs of a domino gate are not known, and hence conservative assumptions have to be made. For a regular domino gate, it is assumed that every rising input transition can trigger an output transition; only the clock input can trigger falling transitions of the output. The effect of the abstraction can be seen as a modification of the event propagation equations of a regular domino gate. The modified event propagation equations are

$$d^R = \min_i\Big\{a^R_i + \delta^{RR}_i\Big\} \tag{13}$$

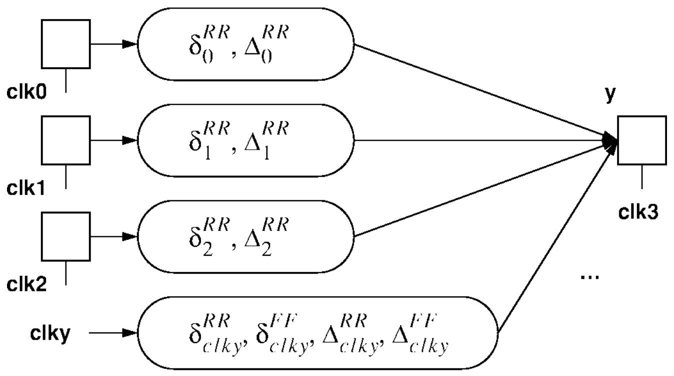
$$D^R = \max_i\Big\{A^R_i + \Delta^{RR}_i\Big\} \tag{14}$$

$$d^F = d^F_{\text{clk}} + \delta^{FF}_{\text{clk}} \tag{15}$$

$$D^F = D^F_{\text{clk}} + \Delta^{FF}_{\text{clk}}. \tag{16}$$

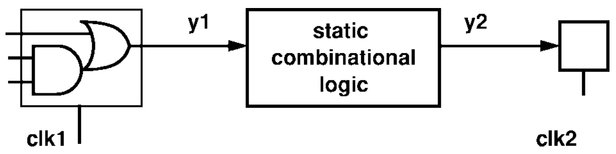


(a)

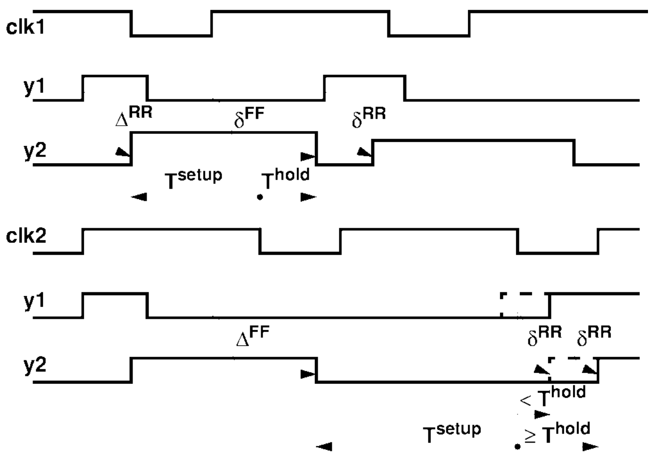

$\delta_0^{RR}, \Delta_0^{RR}$
clk0
$\delta_1^{RR}, \Delta_1^{RR}$
clk1
$\delta_2^{RR}, \Delta_2^{RR}$
clk2
clky
$\delta_{clky}^{RR}, \delta_{clky}^{FF}, \Delta_{clky}^{RR}, \Delta_{clky}^{FF}$
y
clk3
...

(b)

Fig. 10. Circuit view of (a) method 1 and (b) method 2.



(a)



(b)

Fig. 11. Example in which a violating short path involves a rising transition through a domino gate.

Comparing (13)–(16) to (8)–(12), we can see that these earliest event times computed by (13) and (15) are smaller than those computed by (8), (12) and (10). Likewise the latest event times computed by (14) and (16) are larger than those computed by (9) and (11). Hence, the modified propagation equations underestimate the interval in which the computed signal is valid, and thus they are conservative compared to the original equations. It can be seen that method 1 would yield identical results as method 2 if the original event propagation equation for domino signals were replaced by (13)–(16). We can conclude that method 2 provides a conservative approximation for method 1. The computational cost of method 2 is smaller than that of method 1, because the standard SMO analysis (Step 2) is performed on a system with many fewer variables.

For footless domino gates, the modified propagation for the rising transition are also given by (13) and (14), but those for the falling transition are similar to (13) and (14), with the R's replaced by F's. Again a conservative approximation is obtained.

D. Domino Verification Revisited

The choice of clocking methodology is an important design decision. Cycle time and power consumption, but also manufacturability and yield, reliability, scalability, testability and time to market are all affected by that decision. Therefore more aggressive clocking methodologies might be rejected. In this section we consider a set of timing constraints for domino logic that require that input signals to domino gates not change earlier than the beginning of the precharge phase. This implies that spurious rising transitions are not allowed. The fifth signal parameter d^{R0} is no longer needed. Referring to Fig. 6, cases 4, 5, and 6 can no longer lead to valid behavior. IHV reduces to

$$a_{\text{clk}}^F + T_c \geq \max_{j\in p'}\left\{A_j^R\right\} + T^{\text{PWH}} \tag{17}$$

$$a_i^F \geq A_{\text{clk}}^F. \tag{18}$$

ILVN1 and ILVN2 are replaced by

$$a_i^R \geq A_{\text{clk}}^F. \tag{19}$$

ILVP, IPCH1, and IPCH2 remain intact.

The verification problem with these constraints will be referred to as the *conservative verification problem*. Only slight modifications to the methods for solving the original problem are necessary. For the case of regular domino circuits the event propagation equations are given by

$$d^R = \min_p\left\{\max_{i\in p}\left\{a_i^R + \delta_i^{RR}\right\}\right\} \tag{20}$$

$$D^R = \max_i\left\{A_i^R + \Delta_i^{RR}\right\} \tag{21}$$

$$d^F = a_{\text{clk}}^F + \delta_{\text{clk}}^{FF} \tag{22}$$

$$D^F = A_{\text{clk}}^F + \Delta_{\text{clk}}^{FF}. \tag{23}$$

Revisiting our comparison between method 1 and method 2, only the modified equation for the propagation of the earliest rising transition (13) introduces a conservative approximation over the propagation equations used in method 1. In most cases, the earliest arrival of a rising transition does not determine the overall timing of circuit. This is because in

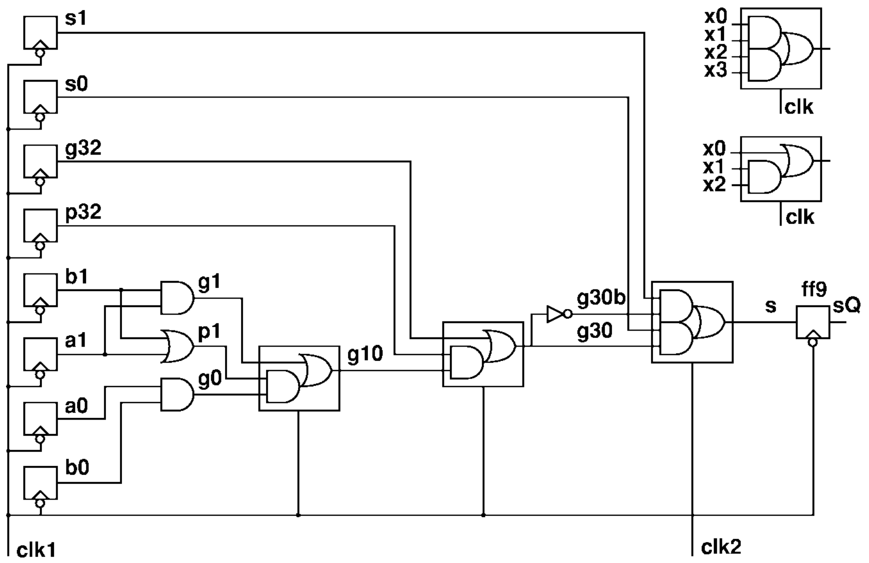


Fig. 12. Example circuit.

the conservative verification problem, a rising transition can never occur later than the earliest falling transition in the same reference cycle $(a_i^R > A_i^F \geq a_i^F)$. Hence, if a short path involves a domino gate, it will most likely involve a falling transition of that domino gate's output rather than a rising transition. Thus the conservatism involved in (13) will never be exposed, and for most practical cases method 2 and method 1 will yield identical results under the conservative domino constraints. A pathological case arises when the output of a domino gate feeds into a static combinational network for which a very large discrepancy between the short path delays triggered by a rising transition and those triggered by a falling transition exists. Fig. 11 shows such a pathological case. A domino gate feeds a block of static combinational logic, which is captured by a positive level-sensitive latch. In the example there are only noninverting paths through the static combinational logic between y_1 and y_2. The first set of waveforms show that when y_2 transitions to one, the setup and hold constraints on the latch are satisfied. Causality is denoted by the arrows and the combinational delays through the static logic are indicated. The bottom set of waveforms show that this is also the case when y_2 is set to zero. However, if the earliest rising event is computed with the conservative propagation equation (13), a hold violation is incorrectly reported. In the figure the conservative waveforms for y_1 and y_2 are shown in dashed line.

Note that in the aggressive verification problem, short paths involving domino gates are as likely to be due to rising spurious transitions as they are due to falling transitions.

TABLE I
DELAYS δ : Δ [ps]

Gate	Input	RR	FR	FF	RF
INV	x0	N/A	50:50	N/A	15:15
AND2	x0,x1	75:100	N/A	50:75	N/A
OR2	x0,x1	50:75	N/A	100:125	N/A
ANDOR21	x0	80:90	N/A	N/A	N/A
	x1,x2	90:100	N/A	N/A	N/A
	clk	75:100	N/A	125:150	N/A
ANDOR22	x0,x2	80:100	N/A	N/A	N/A
	x1,x3	90:100	N/A	N/A	N/A
	clk	80:115	N/A	100:125	N/A
DFFN	clk	N/A	150:150	125:125	N/A

TABLE II
TIMING PARAMETERS [ps]

Gate	Parameter	Value [ps]
DFFN	Setup time	150
	Hold time	0
ANDOR21, ANDOR22	T^{PWH}	200
	T^{PWL}	200
	T^{SEP}	200

VI. EXAMPLE

An example circuit is shown in Fig. 12. Gate delays are listed in Table I, setup and hold parameters in Table II. These parameters were obtained by means of Spice simulations. The waveforms exhibited on each circuit node are shown in Fig. 13. Causality is indicated by the arrows. The flip-flops at the boundaries of the circuit are negative-edge triggered, and share the same clock. One clock cycle is available to propagate the signals between the flip-flops. The combinational logic between the flip-flops consists of both static and regular domino logic. The domino ANDOR22 gate driving s is of special interest. This gate implements a multiplexer. Since domino gates can only implement noninverting functions, the select signal (g30) has to be made available to the domino gate in both polarities. Given that the uncomplemented select signal is generated by domino logic, there are two alternatives

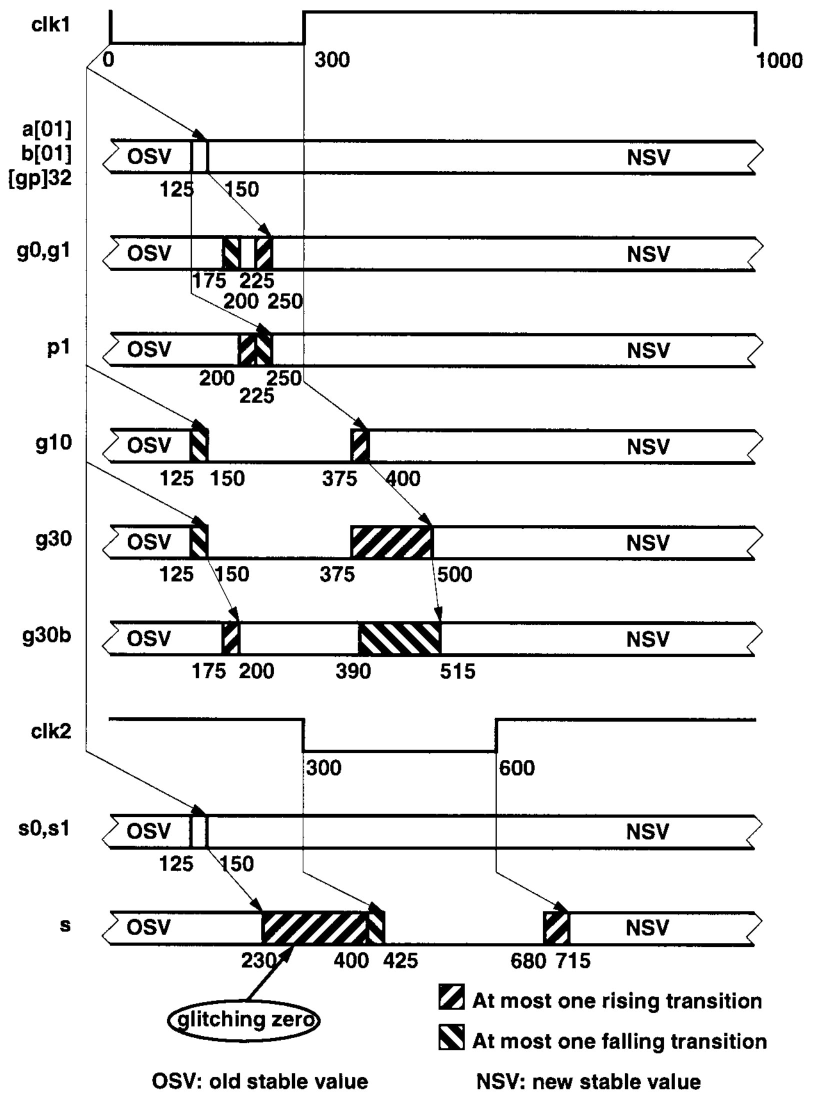


Fig. 13. Waveforms for example circuit; times in ps.

for generating the complemented version. The first one is to produce the complemented signal by a separate chain of domino logic. This effectively drives back the inversion to the flip-flops. In this case either the complemented or the uncomplemented select signal will stay low throughout a clock cycle, whereas the other signal makes a single rising transition during the evaluate phase. The advantage of this implementation is that the ANDOR22 gate can be clocked by the same phase as the other domino gates. The second alternative, adopted in the example, has a lower area compared to the first, but incurs a delay penalty. A static inverter produces the complemented select signal. If the ANDOR22-gate is clocked with clk1, incorrect operation may occur. If g30 evaluates to zero, it stays low throughout the evaluate phase and g30b, stays high throughout the evaluate phase. But if g30 evaluates to one, it makes a single rising transition during the evaluate phase and g30b makes a single falling transition during the evaluate phase. This last case leads to incorrect operation for $g30 = 1$, $s0 = 0$, and $s1 = 1$ (s switches to one when it is supposed to stay low).

To ensure correct operation the falling transition on g30b must be hidden in the precharge phase of the ANDOR22 gate. In the example, this is accomplished by clocking the ANDOR22-gate with clk2, which is a delayed version of clk1. Another alternative, not shown in the figure, would be to make sure that s0 stays low during [300, 515], which is the period in which g30b may still be at a high value from the previous clock cycle (see Fig. 13). This could be accomplished by replacing the static flip-flop producing s0 by a domino latch. In that case the ANDOR22 gate can still use clk1, but the domino latch would use another clock phase.

The example also illustrates that early signal changes originating from the next clock cycle, might arrive during the evaluate phase, producing a glitch. Such behavior violates the conservative domino rules, which stipulate that no input signal change from its stable value earlier than the end of the evaluate phase. Input signals s0 and s1 are in violation of the conservative domino rules: they may start changing as early as $t = 150$, which is before the end of the evaluate phase of the ANDOR22 at $t = 300$. Suppose again that

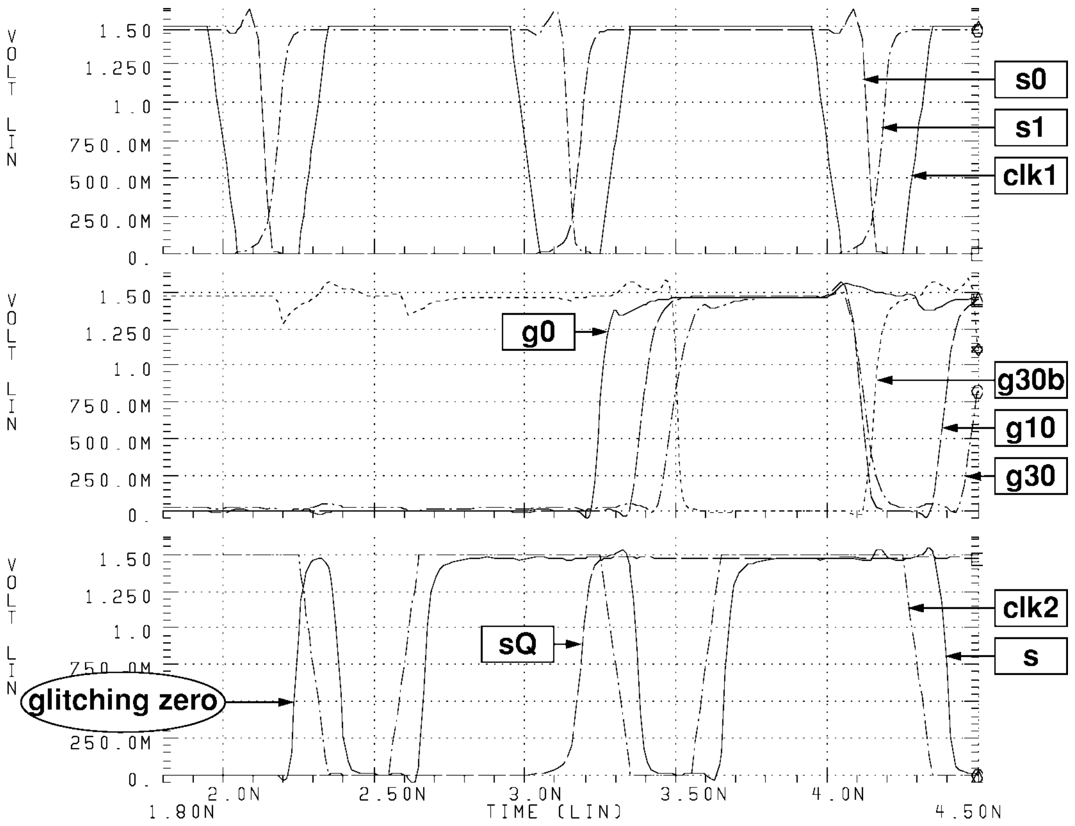


Fig. 14. Electrical simulation.

TABLE III
STABLE VALUES

Cycle [*ns*]	a0	b0	a1	b1	p32	g32	s0	s1	g1	p1	g0	g10	g30	g30b	s
(1.00,2.00)	0	1	1	0	1	0	1	0	0	1	0	0	0	1	0
(2.00,3.00)	0	1	1	0	1	0	0	1	0	1	0	0	0	1	1
(3.00,4.00)	1	1	1	0	1	0	1	0	0	1	1	1	1	0	1

g30 $= 1$, s0 $= 0$, and s1 $= 1$, and that in the next cycle s0 switches to 1. As can be seen from the waveforms, the output s of the ANDOR22 is affected at $t = 230$: it switches to 1, although its stable value is zero. This is not a problem provided that the stable value of s propagates to the flip-flops. This is the case as at $t = 0$, the stable value of s, is captured by the output flip-flop. This is well before the early arrival of s0 corrupts s. According to the conservative domino rules we would be forced to make the falling edge of clk2 occur earlier so that the early arrival of s0 occurs during the precharge phase. This requires more complex circuitry. In our case clk2 can simply be generated by delaying clk1.

To demonstrate that these results are realistic, we also performed an electrical simulation of the circuit from Fig. 12. The actual circuit was implemented in a complementary gallium-arsenide technology [1]. Some representative waveforms are shown in Fig. 14. The stable values of the signals are listed in Table III. Note that at 2.3 ns the ANDOR22 gate (whose output is s) switches. This is due to the effect of the early arrival of the s1 signal as was described above.

VII. CONCLUSION

We addressed static timing verification for sequential circuits implemented in a mix of static and dynamic logic. We restricted our focus to regular domino logic and footless domino logic, a variant of domino logic. The operation of dynamic gates was systematically examined and a set of constraints for proper operation was derived. An important observation is that input signals to dynamic gates may start changing near the end of the evaluate phase. This gives the circuit designer extra flexibility. Two verification methods were presented. Both are based on the SMO model for static timing analysis of sequential circuits. The first method models dynamic gates explicitly. The signals at the terminals of the dynamic gates are modeled by five events: the earliest/latest rising/falling transition and a so-called glitching zero event. The second method applies the original SMO model after a preprocessing step that computes the combinational delays. A postprocessing step checks the dynamic-specific constraints. The relationship between both methods was studied. We show that the second method may result in a more conservative analysis than the first method, but at a smaller computational cost. A detailed example illustrating the important features of the model was presented, and an electrical simulation of the circuit under consideration was performed. The results demonstrate the practical relevance of the models. A possible topic for future work in this area is the study of gated clocks.

ACKNOWLEDGMENT

The authors would like to thank T. G. Szymanski and the anonymous reviewers for many useful comments.

REFERENCES

[1] B. Bernhardt, M. LaMacchia, J. Abrokwah, J. Hallmark, R. Lucero. B. Mathes, B. Crawforth, D. Foster, K. Clauss, S. Emmert, T. Lien, E. Lopez, V. Mazzotta, and B. Oh, "Complementary GaAs (CGaAs(TM)): A high-performance BiCMOS alternative," in *Tech. Dig. 1995 IEEE GaAs IC Symp.*, 1995, pp. 18–21.

[2] E. Friedman, "Latching characteristics of a CMOS bistable register," *IEEE Trans. Circuits Syst.Part 1*, vol. 40, pp. 902–908, Dec. 1993.

[3] R. A. Haring, M. S. Milshtein, T. I. Chappell, S. H. Dhong, and B. A. Chappell, "Self resetting logic register and incrementer," in *Proc. Symp. on VLSI Circuits*, 1996, pp. 18–19.

[4] R. H. Krambeck, C. M Lee, and H.-F. S. Law, "High-speed compact circuits with CMOS," *IEEE J. Solid-State Circuits*, vol. 17, no. 3, pp. 614–619, June 1982.

[5] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Univ. California, Berkeley, Tech. Rep. ERL-M520, 1978.

[6] V. Narayanan, B. A. Chappell, and B. M. Fleischer, "Static timing analysis for self resetting circuits," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1996, pp. 119–126.

[7] R. Puri and K. L. Shepard, "Timing issues in static-dynamic synthesis," in *Proc. Int. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU'97)*, 1997, pp. 89–94.

[8] K. A. Sakallah, T. Mudge, and O. A. Olukotun, "Analysis and design of latch-controlled synchronous digital circuits," in *Proc. ACM/IEEE Design Automation Conf.*, 1990, pp. 111–117.

[9] ——, "Checktc and mintc: Timing verification and optimal clocking of synchronous digital circuits," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1990, pp. 552–555.

[10] K. A. Sakallah, T. Mudge, and O. A. Olukotun, "Analysis and design of latch-controlled synchronous digital circuits," *IEEE Trans. Computer-Aided Design*, pp. 322–333, Mar. 1992.

[11] T. G. Szymaski and N. Shenoy, "Verifying clock schedules," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1992, pp. 124–131.

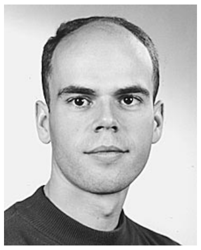
[12] D. Van Campenhout, T. Mudge, and K. A. Sakallah, "Timing verification of sequential domino circuits," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1996, pp. 127–132.

[13] K. Venkat, L. Chen, I. Lin, P. Mistry, and P. Madhani, "Timing verification of dynamic circuits," *IEEE J. Solid-State Circuits*, pp. 452–455, Mar. 1996.

[14] K. Venkat, L. Chen, I. Lin, P. Mistry, P. Madhani, and K. Sato, "Timing verification of dynamic circuits," in *Proc. 17th IEEE Annu. Custom Integrated Circuits Conf.*, 1995, pp. 271–274.

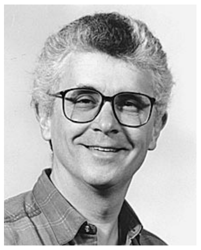
[15] D. Wendell, "Reset logic circuit and method," U.S. Patent 5 430 390.

[16] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*. Reading, MA: Addison-Wesley, 1985.



David Van Campenhout (S'90) received the Engineering degree from the Katholieke Universiteit Leuven, Belgium, in 1993, and the M.S. degree from the University of Michigan, Ann Arbor, in 1994, both in electrical engineering. He is currently pursuing the Ph.D. degree at the University of Michigan.

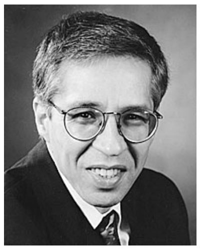
His research interests include hardware design verification, timing analysis, and VLSI design. He was a 1993–1994 graduate fellow of the Belgian American Educational Foundation. He is a member of the IEEE Circuits and Systems Society, and the Royal Flemish Engineering Society (KVIV).



Trevor Mudge (S'74–M'77–SM'84–F'95) received the B.Sc. degree in cybernetics from the University of Reading, U.K., in 1969 and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana, in 1973 and 1977, respectively.

Since 1977, he has been on the faculty of the University of Michigan, Ann Arbor. He is presently a Professor of Electrical Engineering and Computer Science and the Director of the Advanced Computer Architecture Laboratory—a group of eight faculty and 70 graduate research assistants. He is author of more than 150 papers on computer architecture, programming languages, VLSI design, and computer vision, and he holds a patent in computer-aided design of VLSI circuits. He has also chaired some 24 theses in these research areas. His research interests include computer architecture, computer-aided design, and compilers. In addition to his position as a faculty member, he is a consultant for several computer companies.

Dr. Mudge is an Associate Editor for IEEE TRANSACTIONS ON COMPUTERS and *ACM Computing Surveys*. He is a member of the ACM, the IEE, and the British Computer Society.



Karem A. Sakallah (S'76–M'81–SM'92–F'98) received the B.E. degree (with distinction) in electrical engineering from the American University of Beirut, Beirut, Lebanon, in 1975 and the M.S.E.E. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, in 1977 and 1981, respectively.

In 1981 he joined the Department of Electrical Engineering at CMU as a Visiting Assistant Professor. From 1982 to 1988 he was with the Semiconductor Engineering Computer-Aided Design Group at Digital Equipment Corporation, Hudson, MA, where he headed the Analysis and Simulation Advanced Development team. Since September 1988 he has been at the University of Michigan, Ann Arbor, as Professor of Electrical Engineering and Computer Science. From September 1994 to March 1995, he was on a six-month sabbatical leave at the Cadence Berkeley Laboratory, Berkeley, CA. He has published more than 90 papers and has presented seminars and tutorials at many professional meetings and various industrial sites. His research interests are primarily in the area of computer-aided design, with particular emphasis on simulation, timing verification and optimal clocking, modeling, synthesis, knowledge abstraction, and design environments.

Dr. Sakallah is a member of the ACM and Sigma Xi. From 1995–1997, he was Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. He has served on the program committees of ICCAD, DAC, ICCD, and numerous other workshops.