Min-Max Timing Analysis and An Application to Asynchronous Circuits

(in Proceedings of the IEEE, Vol. 87, No. 2, Feb. 1999, pp. 332-346)

Supratik Chakraborty, Member IEEE David L. Dill, Member IEEE Kenneth Y. Yun, Member IEEE

Abstract—

Modern high-performance asynchronous circuits depend on timing constraints for correct operation, so timing analyzers are essential asynchronous design tools. In this paper, we present a 13-valued abstract waveform algebra and a polynomial-time min-max timing simulation algorithm for use in efficient, approximate timing analysis of asynchronous circuits with bounded component delays. Unlike several previous approaches, our algorithm computes separate propagation delay bounds from each circuit input to each internal gate. This is useful for analyzing asynchronous circuits, where the relative transition times of the inputs may not be known *a priori*, unlike synchronous circuits. We also describe an efficient reconvergent fanout analysis technique that helps in increasing the accuracy of simulation. We have applied our algorithm to build an efficient timing analysis tool for extended burst-mode circuits (a class of timing-dependent asynchronous circuits) implemented in the 3D design style [1]. Our tool analyzes gate-level 3D circuits assuming bounded component delays and determines safe timing constraints for correct operation. Although our results represent conservative approximations of the true timing requirements in the worst case, experiments indicate that our technique is efficient and fairly accurate in practice.

Keywords— **Approximate timing analysis, asynchronous circuits, extended burst-mode circuits, min-max timing simulation, polynomial-time analysis reconvergent fanout analysis, 3D circuits,**

I. Introduction

As designers strive to improve the performance of hardware systems, there has been a revival of interest in asynchronous design techniques. Traditional asynchronous design styles, such as delay-insensitive (DI) and speed-independent (SI) designs [2], [3], [4], [5], [6], [7], [8], [9], generate robust circuits that function correctly regardless of component delays (SI circuits require wire delays to be negligible compared to gate delays). However, these circuits often have significant hardware overhead and suffer from degraded performance, rendering them unattractive for many applications. Practical asynchronous circuit designs geared towards high-performance applications, therefore, make judicious timing assumptions and exploit available knowledge of signal transition timing [10], [11], [12], [13], [1]. This results in fast and low-overhead circuits that depend on certain timing constraints for correct operation. Precise values of these constraints depend on the actual gate and wire delays. Since statistical variations in IC fabrication process, operating conditions, etc. result in uncertainties in gate and wire delays, it is important to analyze asynchronous circuits with uncertain component delays to check for violations of internal timing constraints, and to discover constraints on input timing that are sufficient to ensure correct operation. Timing simulation that determines upper and lower bounds on signal propagation delays, given unknown but bounded component delays, is called min-max timing simulation.

Timing analysis techniques for synchronous circuits are not directly applicable to asynchronous circuits. Synchronous timing analyzers work under the assumption that the relative transition times of all circuit inputs are known; for example, it is commonly assumed that all inputs to a synchronous circuit transition simultaneously. In contrast, the relative transition times of the inputs to asynchronous circuits are often not known *a priori*. In fact, it is often necessary to determine constraints on the input transition times that ensure correct operation of the circuit. In this paper, we describe an efficient algorithm for approximate min-max timing simulation of combinational circuits, for use in timing analysis of asynchronous circuits. As an application, we describe an efficient timing analysis tool for a class of practical asynchronous circuits, called extended burst-mode circuits implemented in the *3D* design style [1].

The work described in this paper analyzes asynchronous circuits at the gate level. Since the delays of gates and wires can be bounded by considering extreme "corner" cases of processing and operating conditions, we use the bounded wire delay model: each gate and wire is assumed to have a delay that lies within specified bounds. The delays considered in this paper are *pure* delays — they simply translate the input waveform in time and do not suppress narrow pulses, unlike *inertial* delay elements. We also assume that all gate and wire delays are uncorrelated. Although a certain degree of correlation can be modeled by using the *tracking delay* model of Lam and Brayton [14], it is beyond the scope of this paper.

We view an asynchronous circuit as a combinational network with some (or all) of its outputs fed back as inputs to the network. This is the classical Huffman model of asynchronous circuits [15]. Our approach to timing analysis consists of the following steps applied to the combinational network obtained by cutting the feedback loops:

1) First, we identify gates with potentially spurious transitions or *hazards* [16] on their outputs, generated as a result of timing constraint violations. We call these gates *problem gates*. In order to identify problem gates efficiently, we use a 13-valued abstract waveform algebra.

2) Second, we determine bounds on the signal propagation delays from each circuit input to each problem gate, using min-max timing simulation. Since computing exact bounds on signal propagation delays is computationally intractable [17], we use a polynomial-time algorithm that quickly computes approximate

This work was supported by Semiconductor Research Corporation (ref. 95-DJ-389) and by a gift from Sun Microsystems.

S. Chakraborty is with Fujitsu Laboratories of America, Inc., Sunnyvale, CA, 94086-3922, USA. Email: supratik@celery.stanford.edu

D. L. Dill is with the Computer Science Department, Stanford University, Stanford, CA 94305, USA. Email: dill@cs.stanford.edu

K. Y. Yun is with the Electrical and Computer Engineering Dept., University of California, San Diego, La Jolla, CA 92093-0407, USA. Email: kyy@paradise.ucsd.edu

bounds on the signal propagation delays. Existing polynomial-time algorithms for min-max timing simulation [18], [19] produce overly pessimistic results in the presence of reconvergent fanouts in the circuit, so we have incorporated an efficient reconvergent fanout analysis technique in our algorithm to increase the accuracy of the results.

3) Last, we use the signal propagation delay bounds computed above to derive timing constraints on the circuit inputs that ensure correct operation of the circuit.

To illustrate the practical utility of our min-max timing simulation algorithm, we have developed an efficient timing analysis tool for a class of practical asynchronous circuits, called extended burst-mode circuits implemented in the 3D design style [1]. Given a functionally correct 3D circuit that operates correctly if all timing constraints are satisfied, the tool determines bounds on global timing constraints required for correct circuit operation. Experiments on a suite of 3D benchmarks indicate that our technique is efficient and fairly accurate in practice.

The remainder of this paper is organized as follows. Section II describes related work on bounded delay timing analysis of asynchronous circuits and min-max timing simulation. In Section III, we describe a 13-valued abstract waveform algebra for succinctly representing waveforms and efficiently identifying problem gates. Section IV describes a polynomial-time algorithm for min-max timing simulation of combinational circuits. This forms the core simulation engine of our timing analysis technique. Section V describes an application of our technique to timing analysis of extended burst-mode circuits implemented in the 3D style. We briefly review extended burst-mode specifications and 3D circuits, describe a procedure for efficiently identifying problem gates in 3D circuits, and show how min-max timing simulation can be used to obtain timing constraints for correct circuit operation. A timing analysis tool for 3D circuits is also described. Section VI presents results of applying our timing analysis tool to a suite of 3D benchmarks. Finally, we conclude the paper in Section VII.

II. Related Work

A large number of min-max timing simulators [18], [20], [21], [17], [22], [19], [23], [24], [25] are described in the literature. However, most of these are not suitable for our purpose because: (a) they require the user to specify the relative transition times of the primary inputs, which is contrary to our requirements: we would like to determine constraints on primary input transitions that guarantee correct operation of the circuit, or (b) they have exponential time-complexity in the worst case, which makes them unattractive for analyzing large systems, or (c) they produce overly pessimistic results in the presence of reconvergent fanouts. We briefly discuss a few representative works below.

Ishiura has shown that the problem of computing exact bounds on signal propagation delays is computationally intractable in general [17]. The problem stems from an exponentially large number of combinations of component delays that must be considered in the worst case. Devadas et al. [26] have shown that an exponential number of events may be generated in certain circuits in response to a single input transition. Therefore, event-driven simulators that keep track of all events are inherently inefficient.

One of the earliest polynomial-time algorithms for approximate min-max timing simulation of combinational circuits is that of Breuer and Friedman [18]. Unfortunately, their analysis is extremely pessimistic in the presence of reconvergent fanouts. Another interesting work is that of Ulrich, Lentz, Demba and Razdan [19]. Their algorithm has a complexity that is polynomial in the circuit size and computes separate propagation delays from each primary input and fanout point to each gate output. However, their technique of eliminating *false glitches* [19] produces overly pessimistic results in the presence of nested reconvergent fanouts.

Time-symbolic simulation (TSS) [27] and coded time-symbolic simulation (CTSS) [28] are two well-known techniques for hazard detection and timing verification of asynchronous circuits with bounded component delays. Devadas et al. [25] have also addressed the problem of verifying a gate-level implementation of an asynchronous circuit with bounded wire delays. Although these methods yield exact solutions, their worst-case complexity is exponential in the circuit size for each set of input transitions simulated. Pruning techniques [26] can be used to reduce simulation time to a certain extent. While pruning works well for some examples, the simulation complexity is still high, and can be exponential in the circuit size in the worst case. Additionally, the effectiveness of pruning strongly depends on the interval during which the output is observed; widening the interval can result in a significant increase in running time.

In his dissertation [24], Lindermann described a sophisticated timing simulator, called MTV, and compared it to an earlier simulator, called SCALD [20]. While the results of MTV are more accurate than those of SCALD [24], MTV does not guarantee exact bounds on signal propagation delays in all cases, despite its worst case exponential complexity.

Lavagno et al. [11] have described a procedure for enforcing ordering of signal transitions in asynchronous circuits synthesized from *Signal Transition Graphs (STG)* by inserting delay buffers. Their method uses delay bounds from the circuit inputs to internal gates to estimate the required buffer delays. However, they do not address the problem of efficiently computing the signal propagation delays. As discussed above, this is a non-trivial problem. The work described in this paper could potentially be used to obtain the delay bounds required in Lavagno et al.'s method. However, a discussion of this is beyond the scope of the paper.

III. 13-Valued Waveform Algebra

We now describe a 13-valued abstract waveform algebra for succinctly representing signal transitions and for efficiently identifying gates with spurious transitions or hazards on their outputs. Although the details are somewhat different from previous work, we are essentially following a long tradition of hazard analysis using multi-valued logic [29], [18], [30], [31]. We assume that all feedback loops of the asynchronous circuit have been cut. 13-valued simulation is performed on the underlying combinational network.

We represent the waveform on each wire or gate output us-

ing a triple $\langle b, m, e\rangle$. b and e represent the initial and final states of the signal and are assigned values from 0, 1 and X (X represents 0, 1 or changing repeatedly). The m component represents the intermediate behavior of the signal and is assigned a value: 1 (stable high), 0 (stable low), $\uparrow$, $\downarrow$ (single rising/falling transition), or X (potentially multiple transitions). Using this representation, there are two constant values: $\langle 1, 1, 1\rangle$ and $\langle 0, 0, 0\rangle$; two clean transitions: $\langle 1, \downarrow, 0\rangle$ and $\langle 0, \uparrow, 1\rangle$; four hazards $\langle 0, X, 0\rangle$, $\langle 1, X, 1\rangle$, $\langle 0, X, 1\rangle$ and $\langle 1, X, 0\rangle$; one undefined signal, $\langle X, X, X\rangle$; two signals that are undefined to begin with but eventually settle to binary values, $\langle X, X, 1\rangle$ and $\langle X, X, 0\rangle$; and two signals that have binary values to begin with but eventually become undefined, $\langle 1, X, X\rangle$ and $\langle 0, X, X\rangle$.

Every Boolean function can be extended to operate on values from the 13-valued algebra. Consider an assignment of values $\langle b_i, m_i, e_i\rangle$ to n input variables of a Boolean function f. We define a *trajectory* to be a sequence of assignments of Boolean values to the input variables, where exactly one variable changes value from one element of the sequence to the next. The assignment of 13-valued waveforms to the inputs represents a set of trajectories in the points of an n-dimensional Boolean hypercube. Each trajectory starts in a subcube defined by the vector of start values (b_i for each input i) and ends in a subcube defined by the vector of end values (e_i).

When an input variable has a constant binary value for m_i, the Boolean value of the variable must have the value m_i at every point in the trajectory. If $m_i = \uparrow$ or $\downarrow$, the Boolean variable must change exactly once in the proper direction along every trajectory. If $m_i = X$, the Boolean value of the variable can change arbitrarily in the trajectory. Let $\langle b_f, m_f, e_f\rangle$ represent the value of the Boolean function f on the 13-valued input values. If f is constant throughout the start subcube, b_f is the constant value. Otherwise, $b_f = X$. The definition of e_f is similar. If the value of f throughout every trajectory is a constant 1 or 0, m_f has that value. If f changes from 0 to 1 (1 to 0) exactly once on every trajectory, m_f has the value $\uparrow$ ($\downarrow$). Otherwise, m_f has the value X.

As an example, consider a 2-input AND function, with input waveforms $\langle 0, \uparrow, 1\rangle$ and $\langle 1, \downarrow, 0\rangle$. Fig. 1 shows a 2-dimensional Boolean hypercube with the start and end subcubes circled (the binary values of the inputs are represented as ordered pairs). The AND function evaluates to 0 in both these subcubes, so $b_f = e_f = 0$. The trajectories corresponding to the input waveforms are: $(0,1) \rightarrow (0,0) \rightarrow (1,0)$ and $(0,1) \rightarrow (1,1) \rightarrow (1,0)$, as shown in Fig. 1. The AND function evaluates to 0 at all points in the first trajectory. However, it changes from 0 to 1 and then to 0 along the second trajectory. Therefore, we assign $m_f = X$, and the 13-valued output of the AND function is $\langle 0, X, 0\rangle$. For efficiency, the behavior of standard gates on all 13-valued inputs can be pre-computed and stored in a table. 13-valued simulation of a combinational circuit then proceeds in the obvious way: gates are processed in topological order and the 13-valued waveform on each gate output is determined from the waveforms on its inputs by looking up the table.

IV. Polynomial-Time Min-Max Timing Simulation

We now describe a polynomial-time algorithm for computing approximate bounds on signal propagation delays from each primary input to each gate in a combinational circuit. This forms the core simulation engine of our asynchronous timing analysis technique. We consider circuits composed of basic gates (AND, NAND, OR, NOR, NOT and delay buffers). We model wire delays by inserting a delay buffer along each wire. For notational convenience, we refer to the output of a gate G by the same label (G) and view each primary input as a gate with zero inputs. We also assume that the delay from an input p of gate G to its output lies in the interval $[d_{p,G}, D_{p,G}]$. It should be understood that this is actually a function of the signal values (rising/falling), output capacitance, etc. The gate delay model is as shown in Fig. 2. Each input waveform is delayed by the corresponding input-to-output delay. A zero-delay functional block is then used to generate the output waveform. In the remainder of this paper, we will refer to the inputs of the zero-delay functional block as the inputs of the gate.

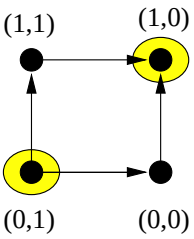


Fig. 1. Illustrating trajectories in 13-valued simulation.

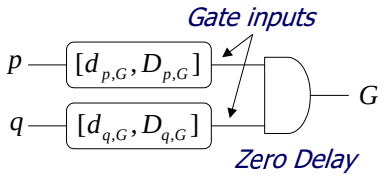


Fig. 2. Gate delay model.

A. Basic Min-Max Timing Simulation

There are two inputs to our timing simulator: (i) a combinational circuit with delay bounds annotated on each gate, and (ii) an input stimulus consisting of 13-valued waveforms associated with the primary inputs. The output of our simulator is an assignment of 13-valued waveforms to each gate in the circuit. In addition, each gate is annotated with a set of intervals, one for *each primary input* i, giving bounds on the signal propagation delay from i to the output of the gate *for the given input stimulus*.

The following data structures are associated with each gate G, and are updated as the algorithm executes.

- A $value$ field which stores a 13-valued waveform.
- An array $del[1 \ldots \text{max\_primary\_inputs}]$ of tuples (t, T). $del[i].t$ stores a lower bound on the signal propagation delay from primary input i to the output of G, and $del[i].T$ stores an upper bound on the same delay. If the transition on i does not cause a transition on G, we assign $G.del[i] = (+\infty, -\infty)$.

The top-level algorithm is shown in Fig. 3. For each gate, a 13-valued output is first determined by looking up a precomputed table. If the result is a constant value, 13-valued

evaluation guarantees that for all possible orderings of the gate input transitions, the constant value is produced at the output of the gate, so no further analysis is required. Otherwise, a more detailed and costly analysis is performed by function MinMaxAnalyze.

Top Level Algorithm

1. Sort gates topologically.
2. (a) **for each** gate G (includes primary inputs)
   for each primary input i
   $G.del[i] = (+\infty, -\infty)$;
   /* G not affected by transition on i */
   (b) **for each** transitioning primary input i
   $i.del[i] = [0, 0]$;
3. **for each** gate G in topological order
   (a) LookUpThirteenValuedTable(G);
   (b) **if** ($G.value \neq \langle 0,0,0\rangle$ or $\langle 1,1,1\rangle$)
   MinMaxAnalyze(G);

Fig. 3. Top-level algorithm.

To understand how function MinMaxAnalyze works, we define potential sensitization of a gate as follows. Gate G is said to be *potentially sensitized* to a transition on gate input p if the transition on p potentially causes a transition on the output of G. The action of MinMaxAnalyze can then be summarized as follows: For each input p of G, we determine whether G is potentially sensitized to the transition on p. If it is, the signal propagation delays from the primary inputs to p are used to update the corresponding delays from the primary inputs to G.

To determine whether G is potentially sensitized to p, we need to know all possible combinations of values on the other gate inputs at the time p transitions. In general, this requires examining all possible orderings between all gate input transitions, which is not efficiently computable for circuits with reconvergent fanouts. We, therefore, use a conservative approximation: For each gate input q that is different from p, we try to determine if the transition on q is ordered relative to that on p. If we cannot detect an ordering, we conservatively assume that the transitions on p and q overlap. However, if an ordering is detected, the ordering information is used to determine if G is potentially de-sensitized to p or if hazards on the output of G are masked.

The simplest way to detect an ordering between p and q is to check if both p and q are affected by transitions propagating from only one primary input i. If so, the delay bounds from i to p and q can be used to determine the ordering of transitions, if any. However, this can produce overly pessimistic results since the correlation of signal propagation delays arising from reconvergent fanouts is ignored. In addition, if multiple primary inputs affect p or q, the above strategy cannot predict an ordering of transitions on p and q because the relative transition times of the primary inputs are unknown. Therefore, the naive strategy for detecting ordering of transitions outlined above may not be very useful in practice. Nevertheless, for simplicity of explanation, we will first describe the min-max timing simulation algorithm assuming the naive strategy. We will return to the topic of determining more accurate ordering information by reconvergent fanout analysis in Section IV-B.

MinMaxAnalyze(G : **gate**)

for each transitioning gate input p of G
1. **for each** other gate input q of G
   (a) Determine if transitions on p and q are ordered.
   (b) Using ordering information:
   for each primary input i that affects p
   qep_i = most-sensitizing value of q at the earliest time a transition on i reaches p;
   qlp_i = most-sensitizing value of q at the latest time a transition on i reaches p;
2. **if** (G has a potential hazard)
   Use ordering information between gate input transitions to determine if hazard is masked.
3. **if** (hazard is masked)
   (a) Update G.value to hazard-free waveform.
   (b) **if** (new G.value == $\langle 000\rangle$ or $\langle 111\rangle$)
   $G.del[i] = (+\infty, -\infty)$ for all primary inputs i.
4. **else**
   for each primary input i that affects p
   /* Let p be connected to the output of gate G' */
   (a) Using qep_i values:
   if G is potentially sensitized to transition on p
   $p.del[i].t = G'.del[i].t + d_{p,G}$;
   $G.del[i].t = \min(G.del[i].t,\ p.del[i].t)$;
   (b) Using qlp_i values:
   if G is potentially sensitized to transition on p
   $p.del[i].T = G'.del[i].T + D_{p,G}$;
   $G.del[i].T = \max(G.del[i].T,\ p.del[i].T)$;

Fig. 4. Basic min-max timing analysis.

Let q be an input of G different from p and let the 13-valued waveform on q be $\langle b, m, e\rangle$. If we know that q transitions before p, then the final value of q, namely $\langle e, e, e\rangle$, is used to determine the sensitizability of G to p. Similarly, if we know that q transitions after p, the initial value of q, namely $\langle b, b, b\rangle$, is used. However, if we are unable to determine any ordering between the transitions on p and q, the value $\langle b, m, e\rangle$ is used, representing that q may be at its b or e values or transitioning in between when p transitions. Hence, the value of q used is the "most-sensitizing value" for determining the potential sensitizability of G to p. Specifically, we never de-sensitize a gate to a transition on an inputs unless we detect an ordering between input transitions that causes the gate to be de-sensitized.

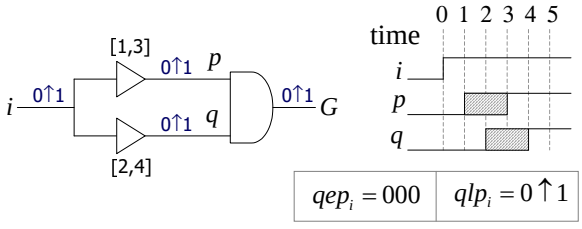
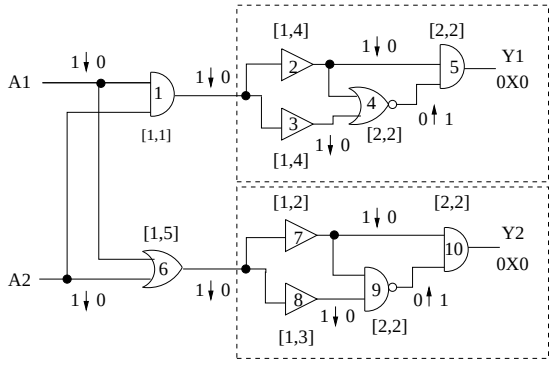


Fig. 5. Computation of qep_i and qlp_i.

Fig. 4 shows the steps in function MinMaxAnalyze. To de-

termine whether p is affected by primary input i [in steps 1(b) and 4], we check to see if $p.del[i]$ is different from $(+\infty, -\infty)$. The qep_i and qlp_i terms used in step 1(b) of MinMaxAnalyze are illustrated in Fig. 5. The test for potential hazards in step 2 essentially checks whether the output of G is one of $\langle 0, X, 0\rangle$, $\langle 1, X, 1\rangle$, $\langle 0, X, 1\rangle$ and $\langle 1, X, 0\rangle$. To determine whether a potential hazard is masked in step 3, we take into account the ordering of gate input transitions determined in step 1(a) and recompute the 13-valued waveform at the output of G. If the m-component of the resulting waveform is not an X, the potential hazard is masked. For example, if one input input of an AND function falls before the other input rises, then the output of the function is $\langle 0, 0, 0\rangle$, and not $\langle 0, X, 0\rangle$. Fig. 6 shows the results of applying the min-max timing simulation algorithm to an example circuit constructed from subcircuits adapted from [17]. It is assumed that the naive strategy for detecting transition orderings is used. Since the circuit is symmetric with respect to $A1$ and $A2$ for the given input stimulus, the delay bounds from $A2$ to the internal gates are identical to those from $A1$. The results are more pessimistic than the reader might expect because transitions propagating from multiple primary inputs affect the inputs of each gate and the relative transition times of the primary inputs are unknown. A careful manual analysis shows that the bounds computed for gates 5, 9 and 10 are conservative. Similarly, the 13-valued waveforms on the outputs of gates 5 and 10 are pessimistic: these gates actually have stable 0 at their outputs.



GATE:	A1	1	2	3	4	5	6	7	8	9	10
DEL[A1]:	[0,0]	[1,1]	[2,5]	[2,5]	[4,7]	[4,9]	[1,5]	[2,7]	[2,8]	[4,10]	[4,12]

Fig. 6. Basic min-max algorithm applied to example circuit. Gate delays shown are [min, max].

Theorem 1 *If a transition on primary input i propagates to the output of gate G, the minimum propagation delay from i to G is bounded below by $G.del[i].t$ and the maximum propagation delay from i to G is bounded above by $G.del[i].T$.*

Proof: We first introduce notation and terminology used in the proof. A controlling value of a basic gate is an input value that determines the output value of the gate regardless of the other input values, e.g., 0 input to an AND or NAND gate. A non-controlling value is a value that is not a controlling value for the gate. As shown in Fig. 2, a gate input is an input to the zero-delay functional block in the gate delay model. All times are assumed to be relative to the time of transition of i. The earliest time that a transition on i reaches p is denoted t_p, and the latest time is denoted T_p. All gates that feed the inputs of gate G are assumed to have topological indices less than that of G. The theorem is proved by complete induction on the indices.

Basis: Step 2(b) of the top-level algorithm (Fig. 3) ensures that the claim is true for all primary inputs.

Hypothesis: Let the claim be true for all gates with indices less than or equal to n.

Induction: Consider gate G (not a primary input) with index $n+1$. We assume that G is affected by primary input i and show that $G.del[i].t$ is a lower bound of t_G; the proof that $G.del[i].T$ is an upper bound of T_G is similar.

Suppose the transition on i propagates through gate input p in order to reach the output of G at time t_G. By definition, G is potentially sensitized to p at time t_G and $t_p \leq t_G$. Since p is an input of G, its topological index is no greater than n. Therefore, by hypothesis, $p.del[i].t \leq t_p$ and hence, $p.del[i].t \leq t_p \leq t_G$.

There are two cases that need to be considered:

(A) If the algorithm finds that G is potentially sensitized to p at time $p.del[i].t$, by definition, the qep_i values indicate that G is sensitizable to p. Therefore, by step 4(a) of MinMaxAnalyze, $G.del[i].t \leq p.del[i].t \leq t_G$.

(B) If the algorithm finds that G is not potentially sensitized to p at time $p.del[i].t$, then primary input i affects one or more other inputs of G that control G at that time. Let r be the controlling input (or one of the controlling inputs in case of a tie) with the largest $r.del[i].t$ value (not $+\infty$, which would indicate r is not affected by i). Since G has no controlling inputs at time t_G, $t_r \leq t_G$. By the induction hypothesis, $r.del[i].t \leq t_r$, so $r.del[i].t \leq t_G$. The proof is completed by showing that $G.del[i].t \leq r.del[i].t$.

By our choice of r, when we consider r in the outermost loop of function MinMaxAnalyze, we find that all other controlling inputs of G have potentially transitioned to non-controlling values by time $r.del[i].t$. In addition, since $r.del[i].t \leq t_G$, no gate input q can settle to a final controlling value before $r.del[i].t$. Therefore, by the induction hypothesis, all gate inputs q that transition to a final controlling value have $q.del[i].T \geq T_q \geq r.del[i].t$. The most-sensitizing values for r computed in step 1(b) of MinMaxAnalyze, therefore, indicate that G is potentially sensitized to r at time $r.del[i].t$. By step 4(a) of MinMaxAnalyze, $G.del[i].t \leq r.del[i].t \leq t_G$.

■

B. Reconvergent Fanout Analysis

Reconvergent fanouts are signals that branch out from the output of one gate and converge back at the inputs of another gate. It is well-known that they are the cause of pessimism in conventional min-max timing simulation [18], [19]. In this section, we describe an efficient technique to conservatively detect ordering of signal transitions in the presence of reconvergent fanouts. This helps in improving the accuracy of our simulation.

Given a combinational circuit and a set of primary input transitions, we define the following two relations on the gates:

- Gate G_1 *enables* gate G_2, if G_2 cannot transition until the transition on G_1 has propagated to one of its inputs. Fig. 7a shows an example. The enables relation can be represented by

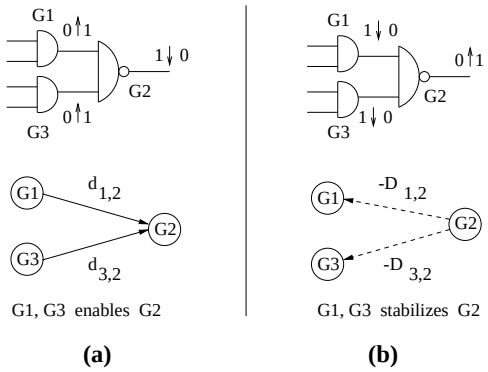


Fig. 7. *Enables* and *stabilizes* relations.

an enabling-graph, where G_1 and G_2 are represented by vertices, and a solid directed edge is drawn from G_1 to G_2 (see Fig. 7a). If the minimum propagation delay from G_1 to G_2 is $d_{1,2}$, the edge is labeled with the weight $d_{1,2}$. Mathematically, if t_i denotes the time when G_i *starts* transitioning, $t_2 \geq t_1 + d_{1,2}$.

- G_1 *stabilizes* G_2, if G_2 can no longer transition after the last transition on G_1 has propagated to G_2. Fig. 7b shows an example of this relation. Here, the output of $G2$ cannot transition after the $\langle 1, \downarrow, 0\rangle$ transition on $G1$ reaches $G2$ and causes it to stabilize to 1, so $G1$ stabilizes $G2$. This is graphically represented by a stabilizing-graph, where a dotted directed edge is drawn from G_2 to G_1 (see Fig. 7b). If $D_{1,2}$ represents the maximum propagation delay from G_1 to G_2, this edge is assigned the weight $-D_{1,2}$. Mathematically, if T_i denotes the time when G_i *ends* transitioning, then $T_2 \leq T_1 + D_{1,2}$.

The above relations depend on the particular input stimulus being simulated, which is implicit.[1]

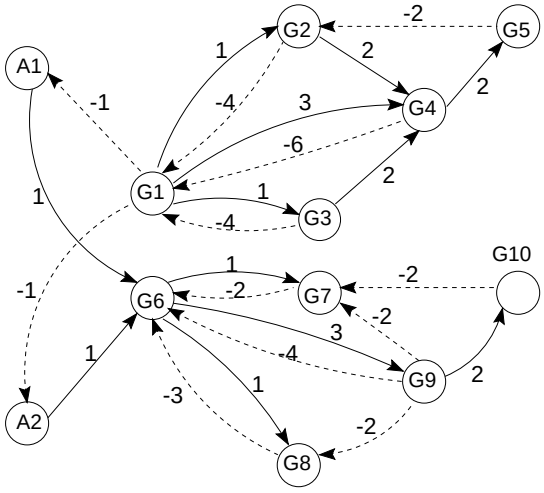


Fig. 8. *enabling* and *stabilizing* DAGs for Fig. 6.

Each of the above relations defines a partial order on the gates. The corresponding graphs are, therefore, directed acyclic graphs (DAGs). Fig. 8 shows these DAGs for the circuit and primary input stimulus of Fig. 6, superimposed on each other. The vertex corresponding to gate i has been labeled G_i. For clarity, we have not shown some edges that are implied by transitivity.

Let us now consider the following sequence of edges in Figure 8: $G7 \stackrel{-2}{\rightarrow} G6 \stackrel{3}{\rightarrow} G9$. The information represented by these edges is: $T_6 \geq T_7 - 2$; $t_9 \geq t_6 + 3$. This implies that $t_9 - T_7 \geq 1 + (t_6 - T_6)$. However, we know from the 13-valued annotations in Fig. 6 that each of $G6$, $G7$ and $G9$ has a clean transition on it. Consequently, both t_6 and T_6 represent the transition time of $G6$, and similarly for $G9$ and $G7$. It then follows that $t_9 - T_7 \geq 1$, so $G9$ transitions at least 1 time unit after $G7$. This implies that the hazard on $Y2$ in Fig. 6 actually does not occur.

The method for detecting transition orderings outlined above can be generalized. We first superimpose the enabling and stabilizing DAGs for a given circuit and primary input stimulus. A directed path in the resulting graph is a sequence of vertices $(G_1, G_2, \ldots G_n)$ such that an enabling or stabilizing edge exists from G_i to G_{i+1} for all i in 1 through $n-1$.

Theorem 2 *Let there be a directed path from G_1 to G_n satisfying the following constraints.*
1. Each gate G_r (other than G_1 and G_n) in the path has a clean transition on it.
2. The summation of all edge weights along the path is positive.
3. If G_1 has a potential hazard on it, the first edge in the path, $\langle G_1, G_2\rangle$, is a stabilizing edge.
4. If G_n has a potential hazard on it, the last edge in the path, $\langle G_{n-1}, G_n\rangle$, is an enabling edge.
Then G_n starts transitioning after G_1 has stabilized to its final value.

Proof: Let $(G_1, G_2, \ldots G_n)$ be a directed path satisfying the constraints. It is assumed that the i^{th} edge, $\langle G_i, G_{i+1}\rangle$, has weight w_i. First, consider the case where $\langle G_1, G_2\rangle$ is a stabilizing edge and all other edges are enabling edges, as shown in Fig. 9. The information represented by this path can be expressed by the following set of inequalities:

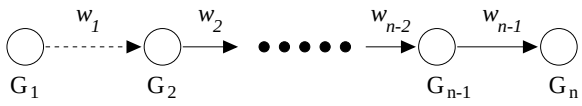


Fig. 9. A path in the superimposed DAGs.

$$\begin{aligned} t_2 = T_2 \geq T_1 + w_1; \quad & t_3 \geq t_2 + w_2; \\ & \vdots \\ t_{n-1} \geq t_{n-2} + w_{n-2}; \quad & t_n \geq t_{n-1} + w_{n-1}. \end{aligned} \tag{1}$$

Note that since all intermediate gates have clean transitions (required by constraint 1), t_i is the same as T_i for all gates G_i other than G_1 and G_n. Therefore, the inequalities in (1) hold regardless of the type (enabling/stabilizing) of the intermediate edges. It follows from inequalities (1) that

$$t_n - T_1 \quad \geq \quad \sum_{i=i}^{n-1} w_i = \sigma. \tag{2}$$

[1] The *enables* and *stabilizes* relations were called *waits_for* and *yields_to* in a preliminary work [32].

Therefore, if σ is positive, we have $t_n > T_1$. In other words, G_n starts transitioning only after G_1 has stabilized to its final value.

Now, suppose $\langle G_1, G_2 \rangle$ is an enabling edge. Constraint 3 requires that G_1 must have a clean transition. This implies t_1 is the same as T_1, so the set of inequalities (1) still apply. Similarly, if the final edge $\langle G_{n-1}, G_n \rangle$ is a stabilizing edge, constraint 4 requires that G_n must have a clean transition. Therefore, t_n is the same as T_n and inequalities (1) apply. ■

The above theorem guarantees that the technique described above never reports a false ordering of transitions. Note, however, that the technique is conservative and may fail to report some orderings that would have been detected by a more expensive and detailed analysis.

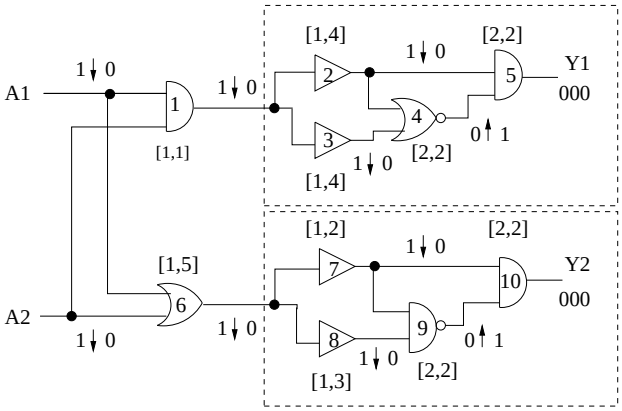
When constructing a path in the superimposed DAGs in accordance with constraints 1, 3 and 4, if a cycle is formed, the length of this cycle must be non-positive. To see why this is so, note that constraint 1 implies that at most one gate in the cycle can have a hazard on its output. Moreover, if there is one such gate G, it must be the first as well as the final gate in the directed path, i.e., $G_1 = G_n = G$, since all intermediate gates must have clean transitions. In this case, by constraint 3, the edge out of G must be a stabilizing edge and by constraint 4, the edge into G must be an enabling edge. Given these edges, it can be easily shown (using the same technique used to derive inequality (2)) that $t_G - T_G \geq \sigma$, where σ denotes the length of cycle. Since $t_G \leq T_G$, we have $\sigma \leq 0$. However, if all gates in the cycle have clean transitions, then for any gate G' in the cycle, $t_{G'}$ is the same as $T_{G'}$. Depending on whether the edges coming into and going out of G' are enabling or stabilizing edges, we have $t_{G'} - T_{G'} \geq \sigma$ or $T_{G'} - t_{G'} \geq \sigma$ or $t_{G'} - t_{G'} \geq \sigma$ or $T_{G'} - T_{G'} \geq \sigma$. Since $t_{G'}$ equals $T_{G'}$, the inequality reduces to $\sigma \leq 0$ in each case.

Since cycles are of non-positive length, the problem of finding a path of positive length can be solved using the Bellman-Ford longest path algorithm [33] with the additional constraints 1, 3 and 4. In fact, since the lengths of paths between multiple pairs of vertices may be needed when applying this technique to determine multiple transition orderings, a longest-path variant of Floyd-Warshall's all-pairs shortest paths algorithm [33] may be used to compute the longest path lengths between all pairs of vertices. If there are n_g gates, this takes $O(n_g^3)$ time. The results may be stored and subsequently used by the min-max timing simulation algorithm to determine if pairs of transitions are ordered. Note, however, that an algorithm that computes an underestimate of the true longest path would also suffice for our purpose, as long as the underestimate is > 0. Such an underestimate can often be efficiently computed by considering only a subset of all the paths.

Computing the enabling and stabilizing DAGs for a given circuit and input stimulus is straightforward. We process gates in topological order. Let G be a gate with a potential transition on its output and let p be a transitioning input of G. To determine whether p enables G, we hold p at its initial state ($\langle b, b, b \rangle$) and allow all other transitioning inputs of G to transition. If 13-valued evaluation now indicates that G *does not* have a potential transition on its output, p is said to enable G. We also consider another situation that arises in practice. Suppose the set of transitioning inputs of G is $\{p_i, \ldots p_n\}$. If there exists a gate G' that enables each p_k in this set, then G' also enables G. This is because a stable gate must be enabled by at least one of its inputs for it to transition, and each transitioning input is enabled by G'. Once we have determined these simple relations, the transitive property can be used to add new edges in the enabling DAG. When computing edge weights, if there exists a sequence of edges $G'' \xrightarrow{a} G' \xrightarrow{b} G$ and if the current weight of edge $G'' \rightarrow G$ is c, its new weight is easily seen to be $\max(c, a + b)$. The procedure for computing the stabilizing DAG is very similar. It is easy to see that for a circuit with n_g gates, each DAG can be constructed in $O(n_g^3)$ time by first constructing the edges as indicated above, and then invoking the transitive closure algorithm [33].

We have used our reconvergent fanout analysis technique in the basic min-max timing simulation algorithm to determine more accurately whether a gate is de-sensitized to one of its inputs. Since our analysis never reports a false ordering of transitions, gates are never falsely de-sensitized. Consequently, delay bounds computed by the min-max algorithm with reconvergent fanout analysis are still conservative, though often more accurate than before.

If the enabling and stabilizing DAGs are constructed and analyzed once using the all-pairs longest paths algorithm and the results stored and subsequently utilized, the complexity of the min-max timing simulation algorithm with reconvergent fanout analysis is $O(n_g^3 + n_g.n_{pi}.n_{fanin}^2)$, where n_g is the number of gates, n_{pi} is the number of primary inputs and n_{fanin} is the maximum fanin of a gate. For efficiency, the current implementation does not compute the transitive closure of each DAG, and uses underestimates of longest path lengths. This results in a more conservative and efficient analysis which has complexity $O(n_g.n_{pi}.n_{fanin}^2.n_g^*)$, where n_g^* is the number of incoming enabling edges or outgoing stabilizing edges at a gate. The worst case value of $n_g^* = n_g$; in practice, however, it is a small number, typically 10 or less. For technological reasons, n_{fanin} is also typically 8 or less. Consequently, the observed complexity is $O(n_g.n_{pi})$.



GATE:	A1	1	2	3	4	6	7	8	9
DEL[A1]:	[0,0]	[1,1]	[2,5]	[2,5]	[4,7]	[1,5]	[2,7]	[2,8]	[4,9]

Fig. 10. More accurate simulation with reconvergent fanout analysis. Gate delays shown are [min, max].

Fig. 10 shows the results of applying our min-max timing simulation algorithm with reconvergent fanout analysis to the circuit of Fig. 6. It may be verified that all the computed delay bounds and 13-valued waveforms are exact.

V. Application: Timing Analysis of Extended Burst-Mode Circuits

We now describe an application of our technique to timing analysis of a class of practical asynchronous circuits. Specifically, extended burst-mode circuits implemented in the 3D design style [1] are analyzed. Although this discussion focuses on 3D circuits, it is likely that circuits designed by other burst-mode implementation methods, such as locally clocked circuits [34], [13] or circuits synthesized by the ACK tool [35], [36], can be analyzed by similar techniques. However, we will not discuss them in this paper.

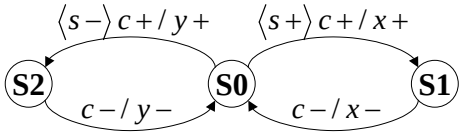


Fig. 11. Example of extended burst-mode specification.

A. XBM Specifications: A Review

Fig. 11 shows an example of an extended burst-mode (XBM) specification [1] of a circuit that works as follows: If the mode bit s is 1 when it is sampled by the rising edge of signal c, then output x follows c for one cycle while output y remains 0. If the sampled value of s is 0, y follows c for one cycle and x remains 0.

Signals enclosed within angle brackets, such as $\langle s+\rangle$, represent *conditional* signals. $\langle s+\rangle$ and $\langle s-\rangle$ denote "if s is 1" and "if s is 0" respectively. Signals which are not enclosed within angle brackets are called *edge* signals. An edge signal with a $+$ or a $-$ (e.g., $c+$ in Fig. 11) constitutes a *terminating* edge, whereas one with a $*$ (not shown in Fig. 11) is a *directed don't care* [1]. A terminating edge $c+$ denotes a $0 \rightarrow 1$ transition of c if c was initially 0, and no transition if c was initially 1. The interpretation of $c-$ is similar. A directed don't care is a signal that can change exactly once in a multi-state path of the state transition diagram (it is a "don't care" because the specification does not care where along the path the change occurs). A terminating edge that is not immediately preceded by a directed don't care (e.g., $c-$ during $S_2 \rightarrow S_0$ preceded by $c+$ during $S_0 \rightarrow S_2$ in Fig. 11) is called a *compulsory* edge. If a conditional signal is not specified in a state transition, it may transition freely during the state transition. Edge signals must, however, remain stable if they do not participate in a state transition. Details of XBM specifications are described in Yun's thesis [1].

An *input burst* consists of a non-empty set of input edges, at least one of which must be a compulsory edge. The edges may appear in any order. An *output burst* consists of a possibly empty set of output edges. When the circuit is in a given state, and all conditional signals are stable at their desired values, the arrivals of all terminating edges in an input burst cause the corresponding output burst to be generated and the machine to transition to the specified next state.

B. 3D Implementation of Burst-Mode

The 3D design style for extended burst-mode circuits was developed by Yun, Nowick and Dill [37], [38], [39], [1]. The hardware implementation of a 3D circuit consists of a combinational network implementing the next state function, with some of its outputs fed back as inputs to the network. The 3D synthesis method guarantees that if the feedback paths are opened, the combinational logic implementing the next state function is free of function and logic hazards [16], regardless of gate and wire delays. A 3D implementation of the extended burst-mode specification in Fig. 11 is shown in Fig. 12.

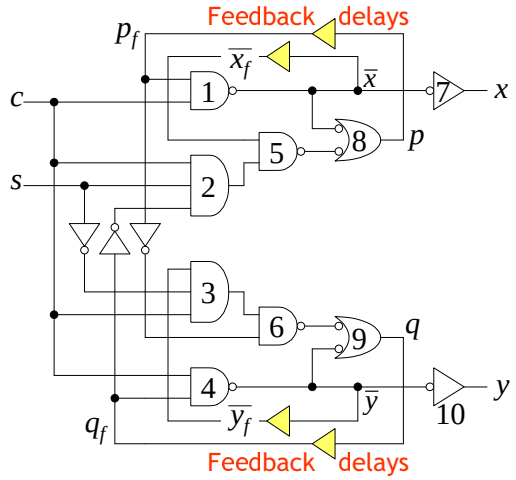


Fig. 12. 3D implementation.

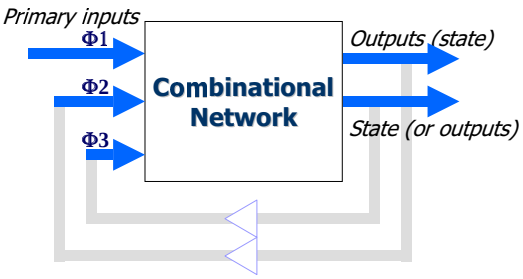


Fig. 13. Phases in the operation of a 3D circuit.

Given an input burst, the response of a 3D machine is comprised of two or three phases, depending on the circuit implementation (see Fig. 13). During the first phase ($\phi 1$), transitions applied to the primary inputs propagate forward through the combinational network and give rise to transitions on the primary outputs and/or state variables. These are fed back as inputs to the combinational network, and the second phase of operation ($\phi 2$) follows. During this phase, the fed back signal transitions propagate forward through the combinational network. If there are only two phases of operation, all signals in the circuit stabilize at the end of this phase. Otherwise, transitions on some primary outputs or state variables are produced at the end of this

phase, which are again fed back as inputs to the circuit. All signals in the circuit stabilize at the end of the third phase.

There are certain global timing constraints when the sequential behavior of a 3D circuit is considered.

1) *Fundamental-mode requirement*: A minimum time interval must elapse between the last primary output transition of the current burst and the first compulsory input transition of the next burst. This is the classical fundamental-mode environmental constraint of asynchronous circuits [40], [16].

2) *Minimum feedback delay requirement*: The feedback paths should have some minimum delay in order to avoid essential hazards [16] – a problem specific to asynchronous sequential circuits.

3) *Setup and hold-time requirement*: Each conditional signal that is sampled by an input burst must remain stable from a certain setup-time before the first compulsory transition to a certain hold-time after the last terminating transition in the input burst.

In the remainder of this paper, we assume that the 3D design is functionally correct. In other words, any malfunctioning of the circuit is attributable solely to violations of the above timing constraints.

C. Useful Properties of 3D circuits

Functionally correct 3D circuits have special properties which simplify their timing analysis. These are summarized below:

1) Any hazard appearing on the outputs of the combinational network during the first phase is due to setup-time violations.

2) Assuming there are no setup-time violations, any hazard appearing on the outputs during the second and third phases is either due to insufficient feedback delays or hold-time violations.

3) Assuming there are no setup-time, feedback delay or hold-time constraint violations, any hazard appearing on the outputs when two consecutive input bursts are applied is due to a fundamental-mode constraint violation.

Because of these properties, it is desirable to analyze the combinational part of a 3D circuit for each phase of each extended burst-mode state transition separately. This effectively isolates the effects of different types of timing constraint violations from each other, simplifying the analysis of each potential violation.

Another important property of 3D circuits is the absence of cyclic dependencies between timing constraints. The values of hold-time and fundamental-mode constraints depend on (i) differences in signal propagation delays along different paths in the combinational circuit, and on (ii) feedback delays. However, the minimum feedback delays as well as setup-time constraints are determined solely by the differences in signal propagation delays along different paths in the combinational circuit. Consequently, feedback delays and setup-time constraints may be determined before the values of the other constraints are known. Once the feedback delays are determined, they can be used to compute the hold-time and fundamental-mode constraints.

D. Identifying problem gates

As noted earlier, a problem gate is a gate that has a hazard or spurious transition on its output, generated as a result of a timing constraint violation. To identify problem gates efficiently, we analyze each phase of operation of each extended burst-mode state transition separately. The combinational network is simulated under the following two conditions.

1) First, we assume there are no timing constraint violations and perform a 13-valued simulation. All gates with potential hazards or spurious transitions on their outputs are marked.

2) Second, we assume that a particular timing constraint is violated, and re-simulate the circuit.

Gates that were not marked in step 1, but have potential hazards on their outputs in step 2 are identified as the problem gates for the timing constraint violation assumed in step 2. To see why this is so, notice that gates that were marked in step 1 had potentially spurious transitions despite all timing constraints being satisfied. 3D design guarantees that these transitions will eventually get masked before reaching the outputs of the circuit. Therefore, the gates that were free of spurious transitions in the first pass but have potential hazards in the second pass are the ones where the timing constraint violation potentially manifests itself.

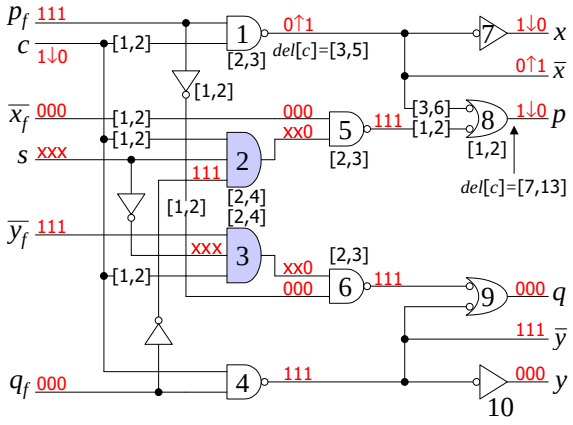


Fig. 14. 13-valued simulation and min-max timing analysis. Gate and wire delays shown are [min, max].

As an example, consider the 13-valued annotations in the circuit in Fig. 14 (the delay bounds shown alongside the gates and wires will be needed later in Section V-E). This circuit is the 3D circuit of Fig. 12 with the feedback loops cut open. We wish to simulate the state transition $S_1 \xrightarrow{c-/x-} S_0$ (see Fig. 11) and identify problem gates where minimum feedback delay violations are manifested, under the assumption that $p_f = \overline{y_f} = \langle 1,1,1 \rangle$ and $q_f = \overline{x_f} = \langle 0,0,0 \rangle$ in state S_1 (it can be easily verified that these are the unique stable values of the feedback inputs in state S_1).

The first phase of operation of the 3D machine is simulated as follows. Since no conditionals appear in the input burst of $S_1 \xrightarrow{c-/x-} S_0$, we set the conditional input s to $\langle X,X,X \rangle$ to model the fact that it can transition freely. Edge input c is set to $\langle 1,\downarrow,0 \rangle$ to model a falling transition on it. The result of our 13-valued simulation is shown in Fig. 14. Gates 2 and 3 potentially have spurious transitions $\langle X,X,0 \rangle$ on their outputs, although no global timing constraints have been violated. Therefore, we mark these gates (shown shaded in Fig. 14).

Then, we simulate the second phase of operation of the machine. The $\langle 1,\downarrow,0 \rangle$ transition on feedback output p in Fig. 14

is copied over to p_f and, similarly, the $\langle 0, \uparrow, 1\rangle$ on $\overline{x}$ is copied over to $\overline{x_f}$. The assumption that all global timing constraints are satisfied implies, in this case, that the delays in the feedback paths are sufficient. Consequently, input c would have settled to $\langle 0, 0, 0\rangle$ by the time the transitions on p and $\overline{x}$ reach p_f and $\overline{x_f}$. Therefore, we assign $\langle 0, 0, 0\rangle$ to c and $\langle X, X, X\rangle$ to s and re-simulate the circuit. The simulation results are not shown here due to space constraints, but it can be easily verified that no spurious transitions or hazards are generated in this phase. Therefore, no gates are marked in this phase.

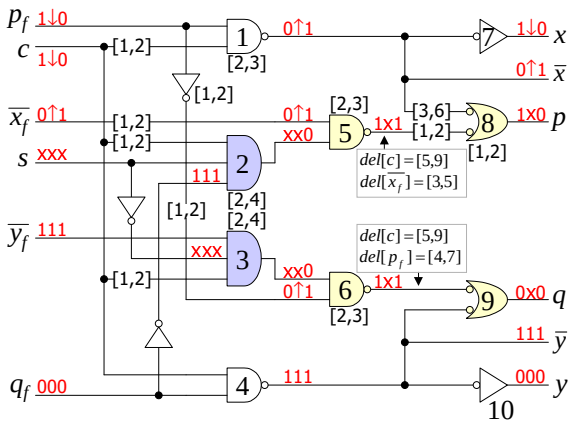


Fig. 15. Identifying problem gates, and min-max timing analysis. Gate and wire delays shown are [min, max].

Finally, we relax our assumption about feedback paths having sufficient delays and re-simulate the circuit with a $\langle 1, \downarrow, 0\rangle$ transition on c, a $\langle 0, \uparrow, 1\rangle$ transition on $\overline{x_f}$ and a $\langle 1, \downarrow, 0\rangle$ transition on p_f. This models the fact that the transition on c could potentially interact with the transitions on the feedback inputs $\overline{x_f}$ and p_f at some internal gate if the feedback paths were too fast. The result of our simulation is shown as 13-valued annotations in Fig. 15. Gates 5, 6, 8 and 9 were not marked in earlier passes, but exhibit potential hazards now. Therefore, these are the problem gates where insufficient feedback delays can cause spurious transitions during the state transition $S_1 \rightarrow S_0$.

E. Timing constraints for 3D circuits

We now use the min-max timing simulation algorithm of Section IV to compute signal propagation delay bounds from the primary and feedback inputs to the problem gates identified above. In general, problem gate G has a potential hazard on its output if transitions propagating from two circuit inputs i and j (primary or feedback) do not reach the gate in the correct order. However, (using the notation of Section IV-A) the transition from j can be made to reach gate G before the transition from i by requiring that j transition at least $d = G.del[j].T - G.del[i].t$ time units before i. Since the delay bounds computed by our min-max timing simulation algorithm are conservative, this guarantees that the transition from j reaches G no later than that from i.

As an example, consider the delay annotations in the circuits in Figs. 14 and 15. Fig. 14 shows signal propagation delay bounds for gates 1 and 8 computed during the first phase of simulation, and Fig. 15 shows signal propagation delay bounds for gates 5 and 6 computed during the final phase of simulation.[2] The bounds in Fig. 15 indicate that a transition on primary input c takes a maximum of 9 time units to reach the output of gate 5, whereas a transition on $\overline{x_f}$ takes a minimum of 3 time units to reach the output of the same gate. In order to ensure that the transition on c reaches the output of gate 5 before the transition on feedback input $\overline{x_f}$ (which is required to meet the feedback delay constraint), $\overline{x_f}$ must transition at least $9 - 3 = 6$ time units after c transitions. However, $\overline{x}$ itself transitions at least 3 time units after c transitions (obtained from Fig. 14). Therefore, the additional feedback delay needed in the $\overline{x} \rightarrow \overline{x_f}$ path is $6 - 3 = 3$ time units. A similar analysis for gate 6 indicates that p_f must transition at least 5 time units after c transitions. However, we know from Fig. 14 that p transitions at least 7 time units after c transitions. Consequently, we do not need any additional delay in the $p \rightarrow p_f$ path. This shows how inherent circuit delays may obviate the need for inserting additional feedback delays. Note that although gates 8 and 9 as well as 5 and 6 are the designated problem gates in Fig. 15, ensuring the absence of hazards at gates 5 and 6 automatically eliminates any hazards at gates 8 and 9. Therefore, gates 8 and 9 do not give rise to additional feedback delay constraints.

The method for determining minimum feedback delays described above can be extended to determine setup-time and hold-time constraints in 3D circuits. Setup-time constraints are obtained by ensuring that the desired values of sampled conditionals reach every problem gate before the sampling edges. Hold-time constraints are analyzed by ensuring that an edge that de-sensitizes a problem gate to a transition propagating from a conditional input, reaches the gate before a change in the conditional signal (from its sampled value to an undefined value) propagates to the same gate. The de-sensitizing edge must propagate through a feedback path before it reaches the problem gate, so we express the hold-times in terms of signal propagation delays through the combinational circuit and the feedback paths involved. Once all state transitions in the extended burst-mode specification have been analyzed and the feedback delays determined, the computed feedback delays are used to determine the actual values of the hold-time constraints. To determine fundamental-mode constraints, we simulate each pair of consecutive state transitions, $S_0 \rightarrow S_1 \rightarrow S_2$, in the extended burst-mode specification and model interactions between transitioning feedback signals of the $S_0 \rightarrow S_1$ transition and transitioning primary inputs of the $S_1 \rightarrow S_2$ transition. Here, too, the constraints are first expressed in terms of feedback path delays; once the delays are known, the actual values of the fundamental-mode constraints are readily determined.

F. 3D timing analysis tool

The ideas described above have been implemented in a timing analysis tool for 3D asynchronous finite state machines. The structure of the tool is shown in Fig. 16.

The top-level analyzer cuts the feedback loops to obtain a combinational network. For each phase of each extended burst-mode state transition, it then determines the 13-valued stimuli to

[2] For clarity, not all delay bounds are shown in the figures.

3D Benchmark	States	State Transitions	Inputs	Gates	Analysis Times(s)	Min. feedback	Setup Time	Hold Time	Fnd-mode constrnt.
ircv	16	22	8	107	1.375	0.000	1.361	7.140	4.789
ircv-bm	8	10	7	58	0.309	0.000	0.000	4.488	1.959
ircv-csm	8	10	7	64	0.339	0.000	0.519	4.015	1.497
isend	24	32	8	247	7.738	0.000	1.028	8.189	3.996
isend-bm	12	15	7	104	1.052	0.000	0.492	7.231	3.751
isend-csm	12	15	7	81	0.698	0.000	0.924	6.226	3.589
trcv	16	22	8	96	1.318	0.000	0.995	5.594	2.715
trcv-bm	8	10	7	58	0.332	0.243	0.080	5.501	2.073
trcv-csm	8	10	7	59	0.328	0.000	0.681	3.775	1.395
tsend	22	30	8	182	4.761	0.000	2.632	7.777	3.403
tsend-bm	11	14	7	65	0.466	0.000	0.371	5.860	2.600
tsend-csm	11	14	7	69	0.492	0.000	0.757	3.984	1.102
biu-dma2fifo	12	15	6	81	0.711	0.000	0.213	7.390	2.550
biu-fifo2dma	11	13	6	77	0.590	0.298	0.592	4.753	1.357
scsi-init-send	7	8	5	33	0.105	0.000	0.213	2.564	0.044
scsi-targ-send	7	8	5	41	0.155	0.000	0.000	4.172	0.540
pscsi	45	62	11	350	17.171	0.000	0.000	0.000	3.505

TABLE I

TIMING ANALYSIS OF 3D CIRCUITS. ALL CONSTRAINTS ARE IN TERMS OF DELAY OF INVERTER WITH FANOUT 4.

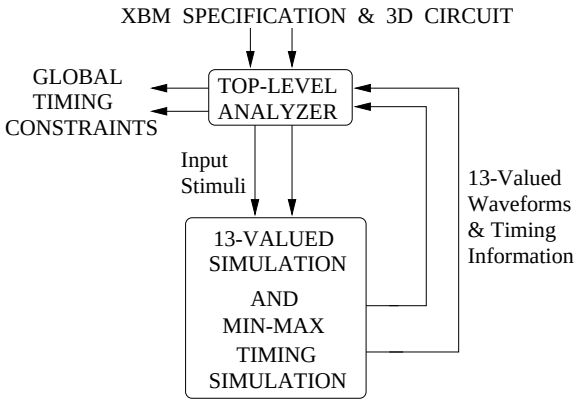


Fig. 16. Timing analysis tool for 3D circuits.

apply to the inputs of the combinational circuit, and invokes the 13-valued simulator. Directed don't cares and non-compulsory terminating edges are modeled assuming they are transitioning in the appropriate direction, since that would lead to the worst case conditions as far as the violation of timing constraints is concerned. Problem gates for each type of global timing constraint violation are identified; then the min-max timing simulator is invoked. Timing information and 13-valued output waveforms computed by the simulator are fed back to the top-level analyzer, which uses them to determine: 1) timing constraints for correct circuit operation and 2) 13-valued waveforms and timing information to copy from the outputs to the corresponding feedback inputs. These are passed on to the 13-valued simulator and the min-max timing simulator for the next pass of analysis. The iteration continues until the tool has examined every extended burst-mode state transition for setup, hold and feedback timing constraint violations, and also examined every pair of consecutive state transitions for fundamental-mode timing constraint violations. The timing analysis is completely automated: given a 3D circuit with delay annotations on the gates and wires and the corresponding extended burst-mode specification, the tool produces timing constraint values that guarantee correct operation of the circuit.

The complexity of our timing analysis technique is $O(n^2_{transitions}.C_{min-max})$, where $n_{transitions}$ represents the number of state transitions in the extended burst-mode specification and $C_{min-max}$ is the complexity of our min-max timing simulation algorithm.

VI. EXPERIMENTAL RESULTS

We have evaluated our tool by applying it to a suite of extended burst-mode benchmarks synthesized according to the 3D design style. The goal of the experiments was to evaluate the speed and accuracy of the tool. While one might argue about the input parameters used, e.g. percentage variations of delays, the algorithm is not terribly sensitive to them.

The number of states, state transitions, primary inputs and gate count of each benchmark are presented in the first four columns of Table I. The column labeled "Analysis Time" lists the time taken by our tool to exercise all extended burst-mode state transitions and determine global timing constraints for each benchmark. The CPU times shown are on a DEC 5000/240 machine. These times do not include the time required to read in the extended burst-mode specification and 3D circuit description, and the time to topologically sort the gates, which are not significant.

The nominal gate delays are estimated using a Hitachi CMOS gate library [41]. The gate delays used in our experiments are assumed to vary within $(0.9d, 1.1d)$, where d is the nominal gate delay. Since we do not have post-layout information about wire

delays, wire delays are assumed to be zero. The reader may wonder about the usefulness of considering $\pm 10\%$ delay variations, but it does not matter much for the experiments. In fact, the percentage variation is a tunable parameter of the tool that can be changed to examine the effects of other delay variations.

The timing constraints obtained for each benchmark are reported in the final four columns of Table I. All constraints are normalized with respect to the delay of an inverter with fanout of 4 (FO4 delay). "Min. feedback" is obtained by determining the minimum delay required in each feedback path for each extended burst-mode state transition, and then taking the maximum of these delays over all state transitions and all feedback paths. "Setup Time" is obtained by determining the setup time for each sampled conditional signal in each extended burst-mode state transition, and then taking the maximum of these values over all state transitions and all conditional inputs. The "Hold Time" constraint is similarly computed. To obtain fundamental-mode constraints, we determine the minimum delay that must elapse between the last output transition of the present burst and the first compulsory input transition of the next burst, for each pair of consecutive state transitions. "Fnd-mode constraint" is then obtained by taking the maximum of all these values over all pairs of consecutive state transitions.

We have manually checked the accuracy of our results for all benchmarks except "pscsi" by running the timing analysis tool in a debugging mode. In this mode, whenever a timing constraint is detected, and is larger than was previously detected, the tool prints out the state of the entire circuit with delay annotations. We then manually check to see whether the timing constraint reported by our tool is correct. Although this process is very painstaking, it is feasible for the benchmarks shown since the sizes of the circuits involved are not huge. The benchmark "pscsi", however, was too large for manual inspection. We have found that timing constraints identified by our tool are accurate for all the 3D benchmarks except "pscsi", which could not be checked. Even for "pscsi", our tool *conservatively* estimates that no additional delays need to be inserted in the feedback paths. Therefore, feedback delays are definitely not needed. The fundamental-mode constraint for "pscsi" also seems quite reasonable (approximately 3.5 inverter delays).

To evaluate the gain in accuracy obtained with our reconvergent fanout analysis technique, we also ran our tool on several 3D benchmarks with the reconvergent fanout analysis turned off. Table II shows the results of our experiments on some of the benchmarks which produced differences in results. In this table, we report hold time (HT) constraints and fundamental-mode (FM) constraints of 3D circuits obtained using our reconvergent fanout analysis technique ("-r") and without it ("-b"). The bold-faced entries indicate that our reconvergent fanout analysis results in less conservative constraints than those obtained with the basic algorithm.

As an interesting aside, our analysis shows that for practical 3D circuits, additional feedback delays are almost never needed – the inherent circuit delays are sufficient to ensure that essential hazards do not occur. Setup-time constraints seem reasonable; however, the maximum hold-times are large because the machine often has to wait for feedback signals in the third phase of operation to de-sensitize internal gates to the conditionals.

3D Benchmark	HT-b	HT-r	FM-b	FM-r
ircv	7.140	7.140	**4.998**	**4.789**
ircv-bm	**5.130**	**4.488**	**2.995**	**1.959**
trcv	**6.525**	**5.594**	**3.201**	**2.715**
trcv-bm	**5.637**	**5.501**	2.073	2.073
tsend-bm	**5.915**	**5.860**	**2.737**	**2.600**
biu-dma2fifo	7.390	7.390	**3.402**	**2.550**
biu-fifo2dma	4.753	4.753	**2.074**	**1.357**
scsi-targ-send	4.172	4.172	**0.628**	**0.540**

TABLE II
COMPARISON OF RESULTS ON 3D BENCHMARKS. ALL CONSTRAINTS ARE IN TERMS OF DELAY OF INVERTER WITH FANOUT 4.

Note that this is an artifact of the design style, not of the timing analysis tool. Fundamental-mode timing constraints for the 3D benchmarks also seem reasonable, since the environment will require a few gate delays' time to react to the primary output transitions.

VII. CONCLUSION

We believe that for maximum performance, asynchronous designs of tomorrow must exploit knowledge of the timing of their components and the environment. Hence, timing analyzers will be essential asynchronous design tools. However, exact timing analysis with bounded component delays is computationally difficult. This paper proposed a polynomial-time technique for approximate min-max timing simulation of combinational circuits, for use in timing analysis of asynchronous circuits. As an application, an efficient timing analysis tool for extended burst-mode circuits implemented in the 3D design style has been described. Although our results represent conservative approximations to the true timing requirements in the worst case, experiments indicate that they are fairly accurate in practice. This suggests that polynomial-time approximate timing analysis techniques hold some promise for efficient timing analysis of asynchronous circuits. As an interesting aside, the results indicate that environmental timing requirements of practical 3D circuits are usually not very severe.

REFERENCES

[1] K. Y. Yun, *Synthesis of asynchronous controllers for heterogeneous systems*, Ph.D. thesis, Stanford University, 1994.

[2] D. E. Muller and W. S. Bartky, "A theory of asynchronous circuits," in *Annals of Computing Laboratory of Harvard University*, 1959, pp. 204–243.

[3] W. A. Clark, "Macromodular computer systems," in *Proceedings of the Spring Joint Computer Conference (AFIPS)*, Apr. 1967, vol. 30, pp. 335–336.

[4] C. H. van Berkel and R. W. J. J. Saeijs, "Compilation of communicating processes into delay-insensitive circuits," in *Proceedings of the International Conference on Computer Design: VLSI in Computers and Processors*, Nov. 1988, pp. 157–162.

[5] C. Berthet and E. Cerny, "Synthesis of speed-independent circuits from algebraic specifications," in *Proceedings of International Symposium on Circuits and Systems*, 1988, vol. 2, pp. 1869–1872.

[6] E. Brunvand and R. F. Sproull, "Translating concurrent programs into delay-insensitive circuits," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, Nov. 1989, pp. 262–265.

[7] A. J. Martin, "Programming in VLSI: From communicating processes to delay-insensitive VLSI circuits," in *Developments in Concurrency and Communications, UT Year of Programming Series*, C. A. R. Hoare, Ed. Addison-Wesley, Reading, MA, 1990.

[8] J. C. Ebergen, "A formal approach to designing delay-insensitive circuits," *Distributed Computing*, vol. 5, no. 3, pp. 107–119, 1991.

[9] P. Beerel and T. H.-Y. Meng, "Automatic gate-level synthesis of speed-independent circuits," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, Nov. 1992, pp. 581–586.

[10] C. J. Myers and T. H.-Y. Meng, "Synthesis of timed asynchronous circuits," *IEEE Transactions on VLSI Systems*, vol. 1, no. 2, pp. 106–119, June 1993.

[11] L. Lavagno, K. Keutzer, and A.L. Sangiovanni-Vincentelli, "Synthesis of hazard-free asynchronous circuits with bounded wire delays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 1, pp. 61–86, Jan. 1995.

[12] W. S. Coates, A. L. Davis, and K. S. Stevens, "The Post Office experience: Designing a large asynchronous chip," *INTEGRATION, the VLSI Journal*, vol. 15, no. 3, pp. 341–366, 1993.

[13] S. M. Nowick, *Automatic Synthesis of Burst-mode Asynchronous Controllers*, Ph.D. thesis, Stanford University, 1993.

[14] W.K.C. Lam and R.K. Brayton, *Timed Boolean Functions: A Unified Formalism for Exact Timing Analysis*, Kluwer Academic Publishers, Norwell, MA, 1994.

[15] D. A. Huffman, "The synthesis of sequential switching circuits," *Journal of the Franklin Institute*, pp. 161–190,275–303, March, April 1954.

[16] S. H. Unger, *Asynchronous Sequential Switching Circuits*, Wiley-Interscience, New York, NY, 1969.

[17] N. Ishiura, *Studies on Logic Simulation and Hardware Description Languages*, Ph.D. thesis, Kyoto University, Kyoto, Japan, 1990.

[18] M.A. Breuer and A.D. Friedman, *Diagnosis and Reliable Design of Digital Systems*, Computer Science Press, Rockville, MD, 1976.

[19] E. Ulrich, K.P. Lentz, S. Demba, and R. Razdan, "Concurrent min-max simulation," in *Proceedings of the European Conference on Design Automation*, 1991, pp. 554–557.

[20] T. M. McWilliams, "Verification of timing constraints on large digital systems," in *Proceedings of the ACM/IEEE Design Automation Conference*, 1980, pp. 139–147.

[21] K. Bowden, "Design goals and implementation techniques for time-based digital simulation and hazard detection," in *Proceedings of the International Test Conference*, 1982, pp. 147–152.

[22] D. Doukas and A. S. LaPaugh, "Clover: A timing constraints verification system," in *Proceedings of the ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digitial Systems*, 1990.

[23] A. R. Martello, *Temporal Analysis for Time Bounded Causal Digital Systems*, Ph.D. thesis, University of Pittsburgh, 1993.

[24] M. H. Linderman, *Simulation of Digital Circuits in the Presence of Uncertainty*, Ph.D. thesis, Cornell University, 1994.

[25] S. Devadas, K. Keutzer, S. Malik, and A. Wang, "Verification of asynchronous interface circuits with bounded wire delays," *IEEE Journal of VLSI Signal Processing*, vol. 7, no. 1–2, pp. 161–182, Feb. 1994.

[26] S. Devadas, K. Keutzer, S. Malik, and A. Wang, "Event suppression: Improving the efficiency of timing simulation for synchronous digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 6, pp. 814–822, June 1994.

[27] N. Ishiura, M. Takahashi, and S. Yajima, "Time symbolic simulation for accurate timing verification of asynchronous behavior of logic circuits," in *Proceedings of the ACM/IEEE Design Automation Conference*, 1989, pp. 497–502.

[28] N. Ishiura, Y. Deguchi, and S. Yajima, "Coded time-symbolic simulation using shared binary decision diagram," in *Proceedings of the ACM/IEEE Design Automation Conference*, 1990, pp. 130–135.

[29] E. B. Eichelberger, "Hazard detection in combinational and sequential switching circuits," *IBM Journal of Research and Development*, vol. 9, no. 2, pp. 90–99, Mar. 1965.

[30] D. S. Kung, "Hazard-non-increasing gate-level optimization algorithms," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, Nov. 1992, pp. 631–634.

[31] T. J. Chakraborty, V. D. Agrawal, and M. L. Bushnell, "Delay fault models and test generation for random logic sequential circuits," in *Proceedings of the ACM/IEEE Design Automation Conference*, June 1992, pp. 165–172.

[32] S. Chakraborty and D. L. Dill, "More accurate polynomial-time min-max timing simulation," in *Proceedings of the Third International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Apr. 1997, pp. 112–123.

[33] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, McGraw-Hill, New York, NY, 1990.

[34] S. M. Nowick and D. L. Dill, "Automatic synthesis of locally-clocked asynchronous state machines," in *Proceedings of the IEEE International Conference on Computer-Aided Design*, Nov. 1991, pp. 318–321.

[35] P. Kudva, G. Gopalakrishnan, and H. Jacobson, "A technique for synthesizing distributed burst-mode circuits," in *Proceedings of the ACM/IEEE Design Automation Conference*, June 1996, pp. 67–70.

[36] P. Kudva, *Synthesis of Asynchronous Systems Targeting Finite State Machines*, Ph.D. thesis, University of Utah, 1995.

[37] K. Y. Yun, D. L. Dill, and S. M. Nowick, "Synthesis of 3D asynchronous state machines," in *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, Oct. 1992, pp. 346–350.

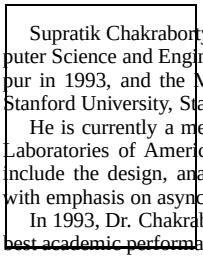
[38] K. Y. Yun and D. L. Dill, "Automatic synthesis of 3D asynchronous finite-state machines," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, Nov. 1992, pp. 576–580.

[39] K. Y. Yun and D. L. Dill, "Unifying synchronous/asynchronous state machine synthesis," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, Nov. 1993, pp. 255–260.

[40] E. J. McCluskey, *Logic Design Principles With Emphasis on Testable Semicustom Circuits*, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[41] *Hitachi High Speed CMOS Gate Array: HG62E Series Design Manual*, Hitachi Co.

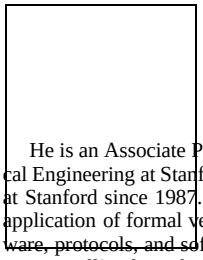
[42] S. Chakraborty, D. L. Dill, K. Y. Yun, and K.-Y. Chang, "Timing analysis for extended burst-mode circuits," in *Proceedings of the Third International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Apr. 1997, pp. 101–111.



Supratik Chakraborty (Member, IEEE) received the B. Tech. degree in Computer Science and Engineering from the Indian Institute of Technology, Kharagpur in 1993, and the M.S. and Ph.D. degrees in Electrical Engineering from Stanford University, Stanford, CA in 1995 and 1998 respectively.

He is currently a member of the Advanced CAD Research group at Fujitsu Laboratories of America, Inc., Sunnyvale, CA. His current research interests include the design, analysis, optimization and verification of digital systems, with emphasis on asynchronous systems and fast timing analysis algorithms.

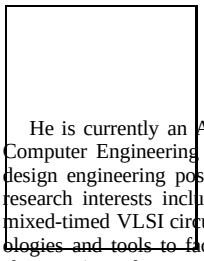
In 1993, Dr. Chakraborty has received the President of India Gold Medal for best academic performance at the Indian Institute of Technology, Kharagpur. He has received a Best Paper award at the International Conference on Computer Design in 1998.



David L. Dill (Member, IEEE) received the S.B. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, in 1979 and the M.S. and Ph.D. degrees from Carnegie-Mellon University, Pittsburgh, PA, in 1982 and 1987, respectively.

He is an Associate Professor of Computer Science and (by courtesy) Electrical Engineering at Stanford University, Stanford, CA. He has been on the faculty at Stanford since 1987. His primary research interests relate to the theory and application of formal verification techniques to system designs, including hardware, protocols, and software.

Dr. Dill's Ph.D. thesis, "Trace Theory for Automatic Hierarchical Verification of Speed Independent Circuits" was named as a Distinguished Dissertation by ACM and published as such by M.I.T. Press in 1988. He was the recipient of an Presidential Young Investigator award from the National Science Foundation in 1988, and a Young Investigator award from the Office of Naval Research in 1991. He has received Best Paper awards at International Conference on Computer Design in 1991 and the Design Automation Conference in 1993 and 1998. From July 1996 to September 1997 he was Chief Scientist of 0-In Design Automation.



Kenneth Y. Yun (Member, IEEE) received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, and the S.M. degree in electrical engineering and computer science from Massachusetts Institute of Technology (MIT), Cambridge.

He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at University of California, San Diego. He had held design engineering positions at TRW and Hitachi for six years. His current research interests include the design, synthesis, analysis, and verification of mixed-timed VLSI circuits and systems, in particular, interface design methodologies and tools to facilitate ultra-high-speed communications between synchronous/asynchronous modules. He has been working with Intel Corp. as a primary consultant on the Asynchronous Instruction Decoder Project.

Dr. Yun has organized ASYNC'98 as a Program Co-Chair. Dr. Yun is a recipient of 1996-99 National Science Foundation CAREER award and 1996 Hellman Faculty Fellowship, and a co-recipient of the Charles E. Molnar award for a paper that best bridges theory and practice of asynchronous circuits and systems at ASYNC'97. He has also received a Best Paper award at the International Conference on Computer Design in 1998.