Timing Analysis for Extended Burst-Mode Circuits *

Supratik Chakraborty†
supratik@celery.stanford.edu

David L. Dill†
dill@cs.stanford.edu

Kenneth Y. Yun‡
kyy@paradise.ucsd.edu

Kun-Yung Chang†
kunyung@montana.stanford.edu

Abstract

We describe an efficient timing analysis technique for extended burst-mode circuits implemented according to the 3D design style. Gate-level 3D circuits with uncertain component delays are analyzed, and safe bounds on timing constraints for correct circuit operation are computed. We employ two passes of multi-valued logic simulation to precisely identify gates where timing constraint violations manifest themselves. Signal propagation delay bounds from the primary inputs to these gates are then used to compute global timing constraints for correct circuit operation. Timing constraints identified by our tool represent conservative approximations to the true timing requirements in the worst-case. In practice, our results are accurate on almost all of the 3D benchmarks we have experimented with.

1. Introduction

As designers strive to improve the performance of hardware systems, there has been a renewed interest in asynchronous design methodologies [24, 8, 20, 19, 2, 26, 25, 16, 15]. One promising design style is *burst-mode* design developed by Yun, Dill and Nowick [21, 27]. Burst-mode design begins with a state-machine specification, somewhat like conventional synchronous state-machine synthesis methods. However, the transitions in the machine are governed by the inputs themselves, not by a clock. The most general form of burst-mode specification, called *extended burst-mode*, also allows the sampling of level signals, called *conditionals*, to determine the choice of next state and output in the machine.

Unlike some design styles [8, 2], all burst-mode implementation methods generate circuits that depend on certain timing constraints for their correct operation. The precise values of these constraints depend on the actual delays of gates and wires in the circuit implementation. However, statistical variations in IC processing conditions, operating conditions, etc. result in uncertainties in gate and wire delays in a chip. Consequently, the problem of determining safe timing constraints for correct operation of burst-mode circuits in the presence of uncertain component delays is an important and interesting problem.

In this paper, we describe an efficient timing analysis technique for determining global timing constraints of extended burst-mode circuits implemented according to the *3D* design style [26]. We choose to analyze 3D circuits because we believe that this is currently the most practical implementation method for extended burst-mode circuits. We consider gate-level circuits, where the delays of gates and wires are allowed to vary within specified intervals. This insures that our analysis takes into account the effect of uncertain component delays. Our goal is to compute *safe* bounds on global timing constraints required for correct operation of the 3D circuit. The computed constraints should be safe in the sense that, when satisfied, they should *guarantee* correct operation of the 3D circuit for all possible variations of component delays within the specified bounds. The challenge is to compute the constraints as accurately as possible (so that the performance of the 3D machine does not suffer) while remaining efficient.

The work described in this paper assumes that 3D design is free of bugs. This implies that a spurious transition or hazard cannot appear at an output of a 3D circuit if all global timing constraints are satisfied. However, the circuit can have spurious transitions at internal gates even when all timing constraints are met. 3D design *guarantees* that these transitions are eventually masked out before reaching the primary outputs of the circuit. Our technique for timing analysis of 3D circuits is premised on this observation.

Our timing analysis technique essentially works by applying the following two-step procedure for *each* transition in the extended burst-mode specification:

*This work was supported by a grant from the Semiconductor Research Corporation, ref. 95-DJ-389

†Computer Systems Laboratory, Stanford University, Stanford, CA 94305, USA

‡Dept. of Electrical and Computer Engg., University of California, San Diego, La Jolla, CA 92093-0407, USA

- We first identify the gates where a global timing constraint violation potentially manifests itself. We choose to call such gates as *problem gates*. Note that, in general, the set of problem gates is *a subset* of the set of all gates with spurious transitions or hazards on their output. We will illustrate this by means of an example later in the paper.

- We then apply a min-max timing simulation algorithm to determine bounds on the signal propagation delays from each transitioning primary and feedback input to these problem gates.

Global timing constraints required for the circuit to correctly execute an extended burst-mode state transition are then obtained by comparing signal propagation delays from the relevant primary and feedback inputs to the problem gates.

In order to identify problem gates, we use an abstract 13-valued signal algebra. Although the details are somewhat different from previous work, we are essentially following a long tradition of hazard analysis using multi-valued logic [9, 10, 3, 14, 5]. The identification of problem gates is accomplished by performing 13-valued simulations of the 3D circuit under two different conditions. We first simulate the circuit under the assumption that all global timing constraints are satisfied. Then we relax our assumption about global timing constraints being satisfied and resimulate the circuit. Gates which did not have spurious transitions or hazards on their outputs in the first simulation pass but exhibit potential hazards in the second pass are the problem gates where global timing constraint violations potentially manifest themselves.

The problem of computing exact bounds on signal propagation delays from the circuit inputs to the problem gates is, in general, computationally intractable [11]. However, we feel that it is very important that a timing analysis tool be able to analyze all reasonable sized circuits fast enough. Existing polynomial-time approximate algorithms [3, 22] are not well-suited for our purpose because they are known to produce extremely pessimistic results in the presence of reconvergent fanouts. Consequently, we have designed our own approximate min-max timing simulation algorithm (this work is reported in a companion paper [4]). Our algorithm performs an efficient and effective reconvergent fanout analysis, thereby producing more accurate results than [3, 22], and has a worst-case complexity that is polynomial in the circuit-size. Nevertheless, the approximate nature of our timing simulation technique implies that in the worst-case scenario, timing constraints identified by our tool represent conservative approximations to the true timing requirements. For example, fundamental-mode constraints computed by our timing analyzer could be larger than the true minimum delay needed between successive input transitions, in the worst-case scenario. Experiments on a suite of 3D benchmarks, however, indicate that our analysis is reasonably accurate, while being fast enough even on moderately large circuits.

The primary contributions of the work reported in this paper can be summarized as follows:

- A method for fast and reasonably accurate timing analysis of extended burst-mode circuits (implemented according to the 3D design style) is presented.

- A tool for automatic timing analysis of 3D circuits with bounded component delays is reported. Our tool automatically performs the following computations for each state transition in the extended burst-mode specification:
  - Checks whether feedback delays are large enough to prevent malfunctioning of the circuit. If it is detected that feedback paths do not have sufficient delays on them, our tool automatically determines additional delays to be inserted in the feedback paths.
  - Extracts setup and hold-time constraints for all conditional signals.
  - Extracts fundamental-mode constraints on transitioning input signals.

  These timing constraints are explained in more detail later in the paper.

- To the best of our knowledge, this is the first *completely automated* tool that analyzes gate-level 3D circuits with bounded component delays and extracts information about the global timing constraints required for correct circuit operation. Thus, our tool automatically extracts information about 3D circuits similar to the information that appears in the "data sheet" provided by a component manufacturer.

1.1. Related work

Time-symbolic simulation [13] and *coded time symbolic simulation* [12] are two well-known techniques for hazard detection and timing verification of asynchronous circuits. The work of [7] also addresses the problem of verifying a gate level implementation of an asynchronous circuit with bounded wire delays. Although these methods yield precise timing results, their worst-case simulation complexity is exponential in the circuit size for *each* set of input transitions simulated. Pruning techniques [6] can be used to reduce the number of delay inequality regions to a certain extent; however the effectiveness of pruning strongly depends on the interval during which the outputs are observed.

The work of [15] addresses the problem of synthesizing hazard-free asynchronous circuits from Signal Transition

Graphs (STG) using the bounded wire delay model. The authors of [15] also describe a technique for determining delays that need to be inserted between STG signal transitions in order to eliminate any hazards that remain after logic synthesis. Although our timing analysis technique is specifically geared for analyzing extended burst mode circuits, the basic simulation strategy could be useful in determining STG signal timing constraints in circuits synthesized according to [15]. This is because these circuits are hazard-free regardless of component delays if timing constraints between STG signal transitions are met. Therefore, our two-step technique of identifying gates where a timing constraint violation manifests itself could probably work here too. However, the analysis of such circuits is beyond the scope of this paper.

In [18], Linderman describes a sophisticated min-max timing simulator, called MTV, that has a worst case simulation complexity that is exponential in the circuit size. Simulators like MTV are not suitable for timing analysis of extended burst-mode circuits 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 the primary input transition times that guarantee correct functioning of the circuit; and (b) they do not generate signal propagation delays from each primary input to each internal gate, which is vital for determining timing constraints for correct circuit operation.

2. Extended burst-mode specifications and 3D design

In this section, we briefly review extended burst-mode specifications [27] and the 3D design style [26]. Several concepts and definitions from [27] and [26] that are used later in the paper are briefly described here.

2.1. Extended burst-mode specification

Figure 1 shows an example of an extended burst mode specification of a circuit which 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 and x remains 0.

Signals like c which are not enclosed within angle brackets are called *edge* signals. Signals enclosed within angle brackets (like $\langle s^+ \rangle$) represent *conditional* or *level* signals. An edge signal with a $+$ or a $-$ symbol represents a *terminating* transition, while one with a $*$ symbol represents a a *directed don't care* (not shown in Figure 1). A terminating transition, like $c+$, indicates that c must transition from low to high, or if it is already high, it must stay there. The interpretation of $c-$ is similar. When a signal x has a directed

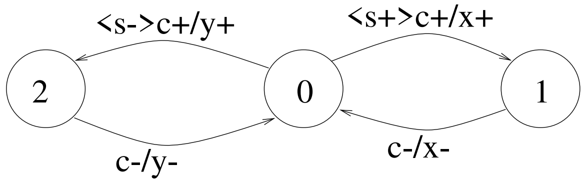

Figure 1. Example of extended burst mode specification

don't care specified on it (x^*), there must exist a sequence of state transitions in the extended burst-mode specification labeled with x^*. This sequence may have one or more state transitions labeled with x^* and must end in a state transition labeled with a terminating transition on x ($x+$ or $x-$). Such a sequence indicates that the signal x must undergo the terminating transition exactly once at some point during the sequence of state transitions. The exact point when the signal transitions is not specified. However, if we have a terminating transition immediately followed by another terminating transition on the same signal (e.g., the signal c in $S_0 \overset{\langle s+\rangle c+/x+}{\rightarrow} S_1 \overset{c-/x-}{\rightarrow} S_0$ in Figure 1), then the second transition ($c-$) *must* occur during the state transition it labels. A transition of this type is therefore called a *compulsory* transition. The interpretation of conditional signals is as follows: a conditional signal like $\langle s+\rangle$ is sampled by other edge signals and represents the condition "if s is high". The interpretation of $\langle s-\rangle$ is similar. If a state transition in the extended burst-mode specification is not labeled with a level signal, the signal may change freely during the transition. However, if an edge signal does not appear in the label of a state-transition, it is not allowed to change.

When the machine implementing the extended burst-mode specification is in a given state, and all conditional signals are stable at their desired values, the arrivals of all the terminating edges of an input burst cause the corresponding output burst to be produced and the machine transitions to the specified next state. The edges in an input burst can appear in any temporal order. Conditional signals must however stabilize to their desired values a certain *setup time* before the first *compulsory* transition, and remain stable until a certain *hold time* after the last *terminating* transition. Output transitions may be generated in any order. The environment is assumed to wait for the machine to stabilize before the compulsory edges of the next input burst are applied at the primary inputs. This is the classical *fundamental mode environmental constraint* of Unger [23]. For example, if we were to represent the behavior of a D flip-flop as an extended burst-mode specification, then the data input would be a conditional (level) input, while the clock would be an edge input.

2.2. 3D implementation of burst-mode

A 3D asynchronous finite state machine is formally represented as a 4-tuple (X, Y, Z, δ) [26], where X is a set of primary input symbols, Y a set of primary output symbols, Z a possibly empty set of state variable symbols, and $\delta : X \times Y \times Z \rightarrow Y \times Z$ is a *next-state function*. The hardware implementation of a 3D finite state machine consists of a combinational network implementing the next state function, with some of its outputs fed back as inputs to the network. The synthesis method developed in [26] guarantees that if the feedback paths are opened, the combinational logic network implementing the next state function is free of function and logic hazards assuming an *unbounded wire delay* model. A 3D implementation of the extended burst mode specification of Figure 1 is shown in Figure 2.

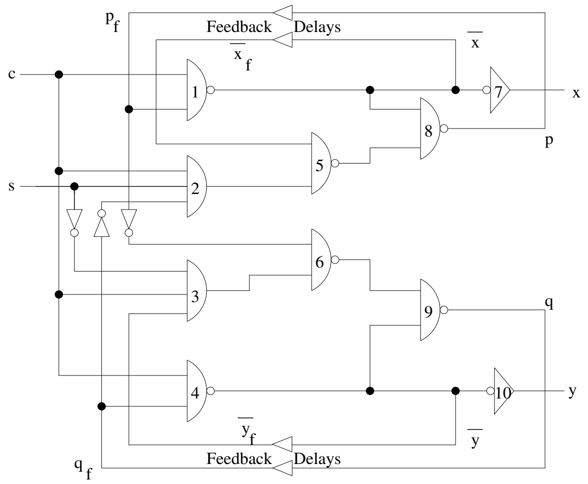

Figure 2. 3D Implementation

Given an input burst, the response of a 3D machine can be considered to be comprised of two or three phases depending on the circuit implementation (Figure 3). During the first phase, transitions applied to the primary inputs propagate forward through the circuit and give rise to transitions on the primary outputs and/or state signals. These signal transitions are then fed back as inputs to the 3D circuit, and the second phase of operation commences. During this phase, signal transitions which were fed back as inputs at the end of the first phase propagate forward through the combinational circuit. If the delays in the feedback paths are not sufficiently large, these feedback transitions also potentially interact with primary input transitions from the first phase still propagating through the circuit. If the 3D circuit has only two phases of operation, the circuit stabilizes after the second phase. However, if there are three phases of operation, transitions on some primary outputs or state signals are produced at the end of the second phase as well. These are again fed back as inputs to the circuit, and the third phase of operation ensues. This phase is quite similar to the second phase of operation and all signals in the circuit stabilize at the end of the third phase.

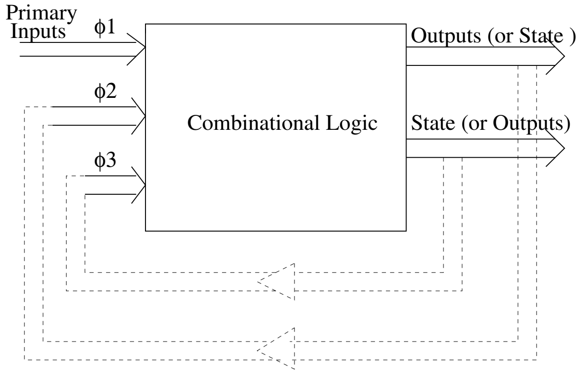

Figure 3. Phases in the operation of a 3D machine

There are certain global timing constraints that need to be satisfied when the sequential behavior of a 3D machine is considered. These are summarized below:

- *Fundamental-mode requirements,* which gives the minimum interval of time that must elapse between the last primary output transition of the present burst and the first compulsory input transition of the next burst.
- *Minimum feedback delay requirements,* which gives the magnitude of additional delays that need to be inserted in the feedback paths to avoid *essential hazards* [23],
- *Setup and hold-time requirements* for conditional signals, which give the minimum interval of time before the first compulsory transition and after the last terminating transition in an input burst, when all sampled conditional signals should remain stable.

3. Timing analysis of 3D circuits

In this section, we first mention a special property of 3D circuits that proves extremely useful in the design of our timing analysis technique. We then give a brief overview of a 13-valued signal algebra, and show how it can be used to efficiently identify problem gates in a 3D circuit. Finally, we briefly describe our polynomial-time min-max timing simulation algorithm and show how it can be used to determine timing constraints for proper operation of the 3D circuit. Details of the 13-valued algebra and min-max timing simulation algorithm may be found in a companion paper [4].

3.1. A special property of 3D circuits

We describe a special property of 3D circuits that proves extremely useful in designing a simple, fast and reasonably accurate technique for determining global timing constraints for correct operation. Consider an extended burst-mode specification and its 3D implementation. If we break open the feedback loops in the 3D circuit and analyze the resulting combinational circuit for each phase of an extended burst-mode state transition, then 3D design *guarantees* the following:

- Any hazard appearing on the primary outputs of the circuit during the first phase of operation is due to a setup-time constraint violation.
- If we assume that no setup-time violations have occurred, any hazard appearing on the primary outputs during the second and third phases of operation of the machine is either due to a minimum feedback delay constraint violation or a hold-time constraint violation.
- If we assume that there are no setup-time, feedback-delay or hold-time constraint violations, any hazard appearing on a primary output when two consecutive input bursts are applied is due to a fundamental-mode constraint violation.

The above property implies that it is advantageous to consider the combinational part of a 3D circuit and analyze the phases in an extended burst-mode state transition one at a time. This would enable us to isolate the interactions of multiple timing constraint violations and identify each type of timing constraint relatively easily. We use this key idea in designing an efficient and reasonably accurate timing analysis technique for 3D circuits.

3.2. 13-valued signal algebra

In this subsection, we outline an abstract 13-valued signal algebra similar to that used in [5]. Our timing analysis tool uses this algebra to succinctly represent signal transitions and analyze hazards in 3D circuits. Details of our 13-valued algebra may be found in a companion paper [4].

The waveform on each signal (wire or gate output) is represented using a triple: $\langle b, m, e \rangle$. b and e represent the initial and final states of the signal and are assigned values from 1, 0 and X (representing an unknown signal value which could be 0, 1 or changing repeatedly). The m component represents the intermediate behavior of the signal and is assigned a value from 1, 0, $\uparrow$, $\downarrow$ and X. An m component of X indicates that the signal potentially has multiple transitions on it. 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 $\langle 0, \uparrow, 1 \rangle$; two *static hazards* $\langle 0, X, 0 \rangle$ and $\langle 1, X, 1 \rangle$; two *dynamic hazards* $\langle 0, X, 1 \rangle$ and $\langle 1, X, 0 \rangle$; one *completely undefined* signal, $\langle X, X, X \rangle$; two signals which are undefined to begin with but eventually settle to constant binary values, $\langle X, X, 1 \rangle$ and $\langle X, X, 0 \rangle$; and two signals which have constant 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 [4]. The 13-valued output of a Boolean function represents the worst-case waveform that can appear at the output for any ordering of the input waveforms represented by the 13-valued inputs (here, worst-case means a hazard waveform is produced if there exists *any* input ordering that actually produces a hazard at the output). For efficiency, a table of the behavior of standard gates on all 13-valued inputs can be precomputed.

13-valued simulation of a combinational circuit proceeds in the obvious way: we process gates in topological order, and evaluate the 13-valued waveform at each gate output from the 13-valued waveforms on its inputs. As an example, consider the circuit in Figure 4, and let us concentrate only on the 13-valued annotations shown in the figure. This is the same circuit as that in Figure 2 with the feedback wires separated into feedback outputs and feedback inputs. Let us assume that feedback inputs p_f and $\overline{y_f}$ are held at $\langle 111 \rangle$, while feedback inputs $\overline{x_f}$ and q_f are held at $\langle 000 \rangle$. We now apply a falling transition on edge input c, while level input s is allowed to transition in an unconstrained way (we will see later that this essentially simulates the first phase of operation of the 3D machine during the $S_1 \overset{c-/x-}{\longrightarrow} S_0$ transition of Figure 1). The result of performing a 13-valued simulation is shown as 13-valued annotations on the gate outputs in Figure 4.

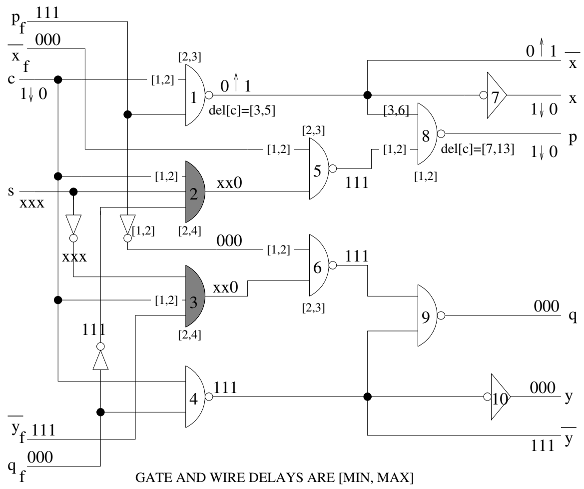

Figure 4. First phase of operation of 3D machine.

3.3. Identifying *problem* gates in 3D circuits

We now describe how 13-valued simulation can be used to efficiently identify *problem* gates in a 3D circuit. Recall that a *problem* gate is one where a global timing constraint violation (say, setup-time violation during an extended burst-mode transition involving sampling of level signals) manifests itself. In order to identify problem gates, we break open the feedback loops in the 3D circuit and perform 13-valued simulation of the underlying combinational circuit under the following two conditions.

1. First, we assume that there are no global timing constraint violations. The corresponding 13-valued waveforms at the inputs of the combinational circuit are determined and a 13-valued simulation is performed. All gates with potential spurious transitions or hazards on their outputs are marked.

2. Next, we relax our assumption about the global timing constraint (the one in which we are interested) being satisfied, determine the corresponding 13-valued input waveforms and resimulate the circuit.

Gates which were not marked in step 1, but have potential hazards on their outputs in step 2 are identified as the problem gates where a timing constraint violation potentially manifests itself. In order to see the rationale behind this, notice that gates which were marked in step 1 had spurious transitions or hazards despite all global timing constraints being satisfied. 3D design *guarantees* that these transitions will get masked before reaching the primary outputs of the circuit. Thus, spurious transitions or hazards on these gates are not manifestations of the timing constraint violation that we simulated in step 2 above. Consequently, gates which were not marked in the first pass, but have potential hazards on their outputs in the second pass are exactly the gates where the timing constraint violation potentially manifests itself. Step 1, therefore, acts as the reference or control simulation which helps us weed out the gates which potentially have spurious transitions or hazards on their outputs, but which do not really exhibit a timing constraint violation effect.

As an example, consider the circuit in Figure 4. We wish to simulate the transition $S_1 \overset{c-/x-}{\rightarrow} S_0$ (see Figure 1) on this 3D machine and identify problem gates where minimum feedback delay constraint violations manifest themselves. Let us assume that the stable values of the feedback inputs in state S_1 are: $p_f = \overline{y_f} = \langle 111 \rangle$ and $\overline{x_f} = q_f = \langle 000 \rangle$ (given the extended burst-mode specification of Figure 1 and the 3D implementation of Figure 2, it can easily be verified that these are the unique stable values of the feedback inputs in state S_1).

We now simulate the first phase of operation of the 3D machine. Since the extended burst-mode transition that we are simulating does not involve sampling of any conditional signal, we set the conditional input s to $\langle XXX \rangle$. This models the fact that s can transition freely during the current state transition. 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 as 13-valued annotations in Figure 4. Clearly, gates 2 and 3 potentially have spurious transitions on their outputs despite no global timing constraints being violated. Therefore, we mark these gates (shown shaded in dark in Figure 4). Note that, by virtue of the 3D design style, these spurious transitions get masked before reaching the outputs of the circuit.

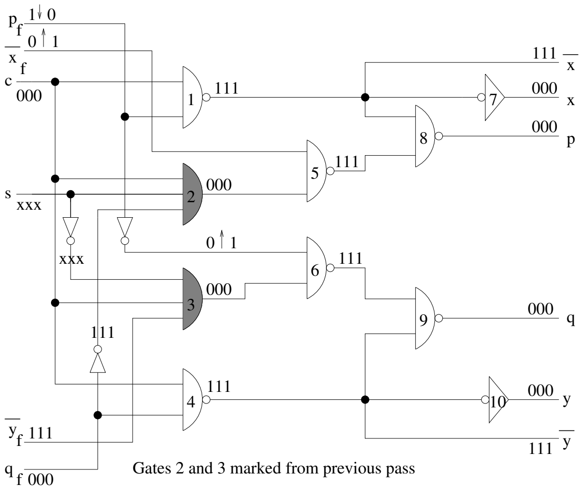

Figure 5. Second phase of operation of 3D machine.

Next, we simulate the second phase of operation of the 3D machine. The $\langle 1 \downarrow 0 \rangle$ transition on feedback output p is copied over to feedback input p_f and the $\langle 0 \uparrow 1 \rangle$ transition on $\overline{x}$ is copied over to feedback input $\overline{x_f}$. We must now perform a 13-valued simulation of the circuit. However, we need to decide what 13-valued waveform should be assign to c before we simulate the circuit. If we assume that all global timing constraints are satisfied, then we are essentially assuming that there is sufficient delay in the feedback paths. Consequently, input c would settle to $\langle 000 \rangle$ by the time the transitions on p and $\overline{x}$ reach p_f and $\overline{x_f}$. The resulting situation is shown in Figure 5. 13-valued simulation now indicates that there are no gates with spurious transitions or hazards on their outputs, and consequently, no additional gates are marked in this step.

Finally, we relax our assumption about the feedback paths having sufficient delays and resimulate 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. This could happen, for example, if the delays in the feedback paths were not large enough. The result of our 13-valued simulation is shown in Figure 6. Gates which were not marked in the earlier simulation passes and exhibit potential

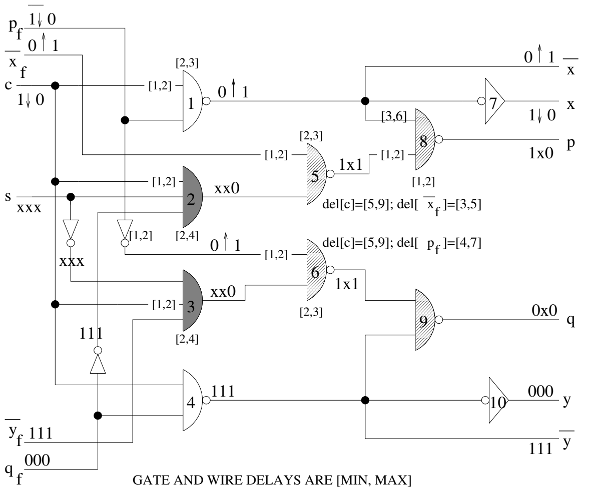

Figure 6. Identifying problem gates, and min-max timing analysis

hazards on their outputs in this pass are gates 5, 6, 8 and 9 (shown lightly shaded Figure 6). We therefore identify these gates as the problem gates where a feedback delay constraint violation potentially manifests itself. Notice that gates 2 and 3 still potentially have spurious transitions on their outputs. However, these were marked in an earlier pass, and are consequently not identified as problem gates by our algorithm. This clearly shows that in general, the set of problem gates is a *subset* of the set of all gates with spurious transitions or hazards on their outputs.

On examining the situation in Figure 6 more closely, we find that indeed, if we have sufficient delay in the feedback path from $\overline{x}$ to $\overline{x_f}$, the $\langle 0 \uparrow 1 \rangle$ transition on $\overline{x_f}$ will reach gate 5 after the $\langle 1 \downarrow 0 \rangle$ transition on c reaches gate 2 and stabilizes its output to 0. As a result, gates 5 and 8 will not have any hazard on their outputs. Similarly, if the feedback path $p \rightarrow p_f$ has a large enough delay, the $\langle 1 \downarrow 0 \rangle$ transition on p_f will reach gate 6 (after inversion) after the $\langle 1 \downarrow 0 \rangle$ transition on c propagates to gate 3 and stabilizes its output to 0. This, in turn, will eliminate the hazards on gates 6 and 9. Consequently, we really need to analyze signal propagation delays from the primary and feedback inputs to gates 5 and 6 in order to determine our required feedback delay constraints.

In this subsection, we have demonstrated how 13-valued simulation can be used to efficiently isolate the problem gates in a 3D circuit. Our next mission is to compute signal propagation delays bounds from the circuit inputs to these gates.

3.4. Min-max timing simulation

In this subsection, we outline the input-output characteristics of our min-max timing simulation algorithm. The details of the algorithm may be found in a companion paper [4].

There are two inputs to our algorithm. The first is a combinational circuit with minimum and maximum delay annotations on each gate. The second is an input stimulus consisting of 13-valued signals associated with the circuit inputs. The output of our algorithm is an assignment of a 13-valued signal to each gate, G, modeling the output waveform resulting from the input. In addition, G is annotated with a set of intervals, one *for each circuit input* i, representing the shortest and longest propagation delays from i to the output of G, for the given input stimulus.

Our min-max timing simulator incorporates an efficient and effective reconvergent fanout analysis algorithm [4]. Our reconvergent fanout analysis technique is able to detect several instances of signal transition orderings that go undetected with previously published polynomial-time [3, 22] and some exponential-time [18] algorithms. This enables us to compute signal propagation delays from the circuit inputs to the gates with increased accuracy. The worst-case complexity of our min-max timing simulation algorithm is : $O(n_g^2.n_{pi}.n_{fanin}^2)$, where n_g = number of gates in the circuit, n_{pi} = number of circuit inputs, and n_{fanin} = maximum fanin of a gate. However, in reality, we have a multiplicative factor of at most $10.n_g$ instead of n_g^2 and n_{fanin} is usually restricted to 4 or 5 for technological reasons.

3.5. Determining timing constraints for correct circuit operation

Once we have identified the problem gates in a 3D circuit, signal propagation delays from the primary and feedback inputs to these gates need to be compared to determine timing constraints for correct circuit operation. The general idea behind this technique is that a problem gate G has a potential hazard on its output because transitions propagating from two circuit inputs i and j (primary or feedback) reach the gate in the wrong order. Therefore, we must determine constraints on the transition times of i and j to insure that transitions propagating from these inputs reach gate G in the correct order. In this subsection, we show how this can be accomplished utilizing the information computed by our min-max timing simulation algorithm.

Let $G.del[i]$ be a tuple (t, T) containing a lower and an upper bound of the signal propagation delay from circuit input i to gate G. These tuples are computed by our min-max timing simulator for each gate G in the circuit, and for each transitioning circuit input i. A sufficient condition to insure that the transition from circuit input j reaches problem gate G before that from i, is that j transitions at least $G.del[j].T - G.del[i].t$ time units before i transitions. Note that the length of a sensitized path from i to G depends on the stimulus applied to the circuit inputs, in general. Consequently, we need to recompute the del tuples for each gate each time a new input stimulus is applied to the circuit.

As an example, let us consider the circuits in Figures 4 and 6. We now concentrate on the gate and wire delay annotations shown in these figures. Figure 4 shows signal propagation delay bounds for gates 1 and 8 computed during the first phase of simulation. Similarly, Figure 6 shows signal propagation delay bounds for gates 5 and 6 computed during the final phase of simulation. The delay bounds in Figure 6 indicate that a transition on circuit input c takes a maximum of 9 time units to reach the output of gate 5, while a transition on $\overline{x_f}$ takes a minimum of 3 time units to reach the output of the same gate. Consequently, in order to insure that the transition on c reaches gate 5 before the transition on $\overline{x_f}$ does, $\overline{x_f}$ 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 Figure 4). 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 $9 - 4 = 5$ time units after c transitions. However, we know from Figure 4 that p transitions at least 7 time units after c transitions. Consequently, we do not need to insert any additional delay in the $p \rightarrow p_f$ in order to insure that the machine correctly executes the $S-1 \stackrel{c-x-}{\rightarrow} S_0$ state transition. This shows how the inherent delays of circuit components often obviate the need for inserting additional feedback delays.

3.6. 3D timing analysis tool

The method for determining minimum feedback constraints outlined above can easily be extended to determine setup and hold-time constraints for 3D circuits. In order to identify fundamental-mode timing constraints, each pair of adjacent state transitions, $S_0 \rightarrow S_1 \rightarrow S_2$, in the extended burst-mode specification needs to be simulated. The basic technique for determining fundamental-mode timing constraints is the same as that for the other timing constraints. The only difference is that we now 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.

These ideas have been incorporated in a timing analysis tool for 3D asynchronous finite state machines. The structure of the tool is shown in Figure 7. The top-level analyzer first breaks the feedback loops in a gate-level 3D circuit to obtain a combinational network. It then looks at the extended burst-mode specification and analyzes the combinational circuit obtained above for each phase of each extended

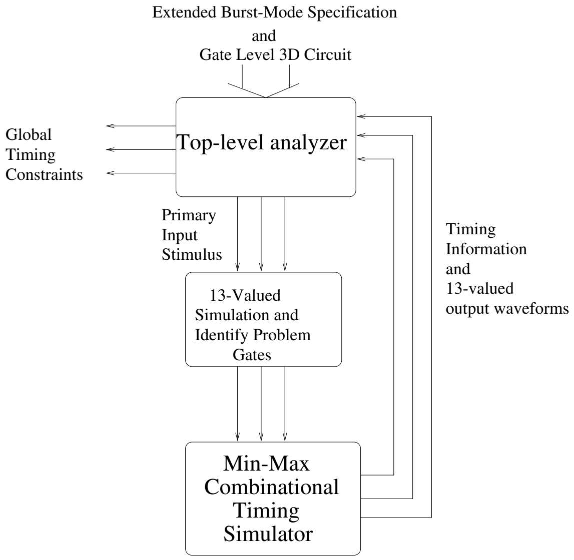

Figure 7. Timing analysis tool for 3D circuits

burst-mode state transition. This involves determining the 13-valued stimuli to apply to the inputs of the combinational circuit and performing a 13-valued simulation of the circuit. Signals with directed don't-care transitions are assumed to be transitioning since this would lead to the worst-case situation as far as violation of timing constraints is concerned. Problem gates for each type of global timing constraint violation are identified and then our min-max timing simulator is invoked. The results of the timing simulator are fed back to the top-level analyzer. Timing information and 13-valued output waveforms computed by the simulator are used by the top-level analyzer to: (a) determine timing constraints for correct circuit operation and (b) determine 13-valued waveforms and timing information to copy from the feedback outputs to the corresponding feedback inputs. These are then passed on to the 13-valued simulator and min-max timing simulator for another pass of analysis. The iteration continues until we have examined each extended burst-mode state transition for setup, hold and feedback timing constraint violations, and also examined each pair of adjacent state transitions for fundamental-mode timing constraint violations.

The complexity of our timing analysis technique is bounded above by $(n^2_{transitions} \times C_{min-max})$, where $n_{transitions}$ represents the number of state transitions in the extended burst-mode specification and $C_{min-max}$ represents the complexity of the min-max timing simulation algorithm. Since $C_{min-max}$ for our simulator is polynomial in the circuit-size, the overall complexity of our timing-analysis method is polynomial in the circuit-size and in the size of the extended burst-mode specification.

4. Experimental results

We have applied our timing analysis tool to a suite of asynchronous benchmarks synthesized according to the 3D style. The benchmark characteristics are shown in Table 1.

3D Benchmark	States	State Trans.	Inputs	Gates
ircv	16	22	8	107
ircv-bm	8	10	7	58
ircv-csm	8	10	7	64
isend	24	32	8	247
isend-bm	12	15	7	104
isend-csm	12	15	7	81
trcv	16	22	8	96
trcv-bm	8	10	7	58
trcv-csm	8	10	7	59
tsend	22	30	8	182
tsend-bm	11	14	7	65
tsend-csm	11	14	7	69
biu-dma2fifo	12	15	6	81
biu-fifo2dma	11	13	6	77
scsi-init-send	7	8	5	33
scsi-targ-send	7	8	5	41
pscsi	45	62	11	350

Table 1. Benchmark Characteristics

Table 2 gives the actual time taken by our tool to exercise all the extended burst-mode state transitions specified for a 3D benchmark and determine the global timing constraints required for correct circuit operation. The CPU times shown are on a DEC 5000/240 machine. The "Analysis Time" for a 3D benchmark reported in Table 2 does not include the time required to read in the circuit description and the extended burst-mode specification, nor does it include the time required to topologically sort the gates.

The timing parameters obtained for each benchmark are indicated in Table 3. For each benchmark circuit, gate delays are estimated for a Hitachi CMOS gate library [1]. Each gate has a nominal delay given by: $t = t_0 + K \times C$, where $C = 0.4\times$ *Effective fanout*, where *Effective fanout* $= \sum_{all\ fanout\ gates}(Normalized\ load\ of\ fanout\ gate)$. The t_0, K and *Normalized load* figures have been obtained from the Hitachi data book [1] and are different for rising and falling transitions. In our experiments, the actual gate delays are assumed to vary within $(0.9 \times nominal\_delay, 1.1 \times nominal\_delay)$. We also assume zero delays for the wires, since we do not have post-layout information about the delays of wires in the circuit.

It may be noted here that an inverter with a fanout of 4 has a rising delay of 2.12ns and a falling delay of 1.74ns in this library. The timing constraints shown in Table 3 are normalized with respect to the delay of an inverter with fanout 4 (FO4 delay), which is assumed to be $2ns$.

3D Benchmark	Analysis Time(s)
ircv	1.375
ircv-bm	0.309
ircv-csm	0.339
isend	7.738
isend-bm	1.052
isend-csm	0.698
trcv	1.318
trcv-bm	0.332
trcv-csm	0.328
tsend	4.761
tsend-bm	0.466
tsend-csm	0.492
biu-dma2fifo	0.711
biu-fifo2dma	0.590
scsi-init-send	0.105
scsi-targ-send	0.155
pscsi	17.171

Table 2. Analysis Times

The timing constraints reported in Table 3 are computed in the following manner. The "Minimum feedback" constraint is obtained by determining the minimum feedback delay required in each feedback path for each extended burst-mode state transition, and then taking the maximum of these values over all extended burst-mode state transitions and all feedback paths. The "Setup Time" constraint is obtained by determining the setup time constraint for each conditional signal being sampled in each extended burst-mode state transition, and then taking the maximum of these values over all extended burst-mode transitions and all conditional inputs. The "Hold Time" constraint is computed in a similar fashion using the hold times of conditional inputs. "Fund-mode constraint" is determined by computing the maximum fundamental mode timing constraint for each pair of consecutive state transitions in the extended burst-mode specification, and then taking the maximum of these values over all pairs of consecutive transitions. Note that this assumes that no other timing constraint violation (like setup or hold-time violation) has occurred in each of the pair of transitions. In addition, fundamental-mode constraints reported in the table give the minimum interval of time that must elapse *between the last primary output transition of the present burst and the first compulsory input transition of the next burst.*

It may be noted here that benchmark "pscsi" does not have any conditional input. Consequently, our tool computes the setup and hold-time constraints of this circuit to be 0.

We have checked the accuracy of our timing results for all of our benchmarks except "pscsi". Our strategy for checking the accuracy of our results consists of running the timing analysis tool in an interactive debugging mode. Whenever a timing constraint is detected, and is larger than was previously detected, our tool prints out the state of the entire

3D Benchmark	Minimum feedback (FO4 delays)	Setup Time (FO4 delays)	Hold Time (FO4 delays)	Fund-mode constraint (FO4 delays)
ircv	0.000	1.361	7.140	4.789
ircv-bm	0.000	0.000	4.488	1.959
ircv-csm	0.000	0.519	4.015	1.497
isend	0.000	1.028	8.189	3.996
isend-bm	0.000	0.492	7.231	3.751
isend-csm	0.000	0.924	6.226	3.589
trcv	0.000	0.995	5.594	2.715
trcv-bm	0.243	0.080	5.501	2.073
trcv-csm	0.000	0.681	3.775	1.395
tsend	0.000	2.632	7.777	3.403
tsend-bm	0.000	0.371	5.860	2.600
tsend-csm	0.000	0.757	3.984	1.102
biu-dma2fifo	0.000	0.213	7.390	2.550
biu-fifo2dma	0.298	0.592	4.753	1.357
scsi-init-send	0.000	0.213	2.564	0.044
scsi-targ-send	0.000	0.000	4.172	0.540
pscsi	0.000	0.000	0.000	3.505

Table 3. Results of Timing Analysis

circuit with the thirteen-valued waveforms and the delay bound annotations. We then check to see whether the timing constraint reported by our tool is conservative or exact. Although this process is very time-consuming and painstaking, it is feasible since the sizes of the circuits involved are not very large ("pscsi", on the other hand, is reasonably large for manual inspection). We have found that timing constraints identified by our tool are accurate for all of the 3D benchmarks other than "pscsi". Even with "pscsi", our tool reports that no additional delays need to be inserted in the feedback paths. Since this is a conservative approximation to the true feedback delay requirement, we can infer that additional feedback delays are actually not required in this circuit. The fundamental-mode constraint extracted by our tool for "pscsi" also seems quite realistic, and we know that this is a conservative approximation (an upper bound) of the true minimum fundamental-mode constraint. Thus, our timing analysis technique is fairly accurate.

Our results indicate that our timing analysis technique is fast enough even when considering 30 – 40 distinct state transitions and dealing with moderately large 3D circuits. Since all analysis algorithms used in our tool have complexity that is polynomial in the circuit size, we do not expect any significant performance degradation even with larger 3D circuits.

Our analysis shows that for practical 3D circuits, additional feedback delays are almost never needed – the inherent delays of gates are sufficient to ensure that essential hazards don't occur. Setup-time constraints are also quite reasonable (of the order of one gate delay). However, the maximum hold times are usually long since the machine often has to wait for feedback signal transitions in the third phase of operation for some internal gates to become desensitized to the conditionals. This can possibly be overcome by some amount of careful redesigning to insure that feedback signal transitions in the second phase desensitize all gates to the conditional inputs. Fundamental-mode timing constraints for our 3D benchmarks also appear to be reasonable, since the environment will take a few gate delays' time to react to the primary output transitions anyway. On the whole, timing requirements for 3D design do not seem very restrictive on the environment.

To the best of our knowledge, the timing analysis tool reported in this paper is the first completely automated tool that analyzes 3D circuits with bounded component delays and determines global timing constraints required for correct circuit operation. Consequently, we are unable to present performance comparison results with other tools with similar capabilities.

5. Conclusion

We believe that fully speed-independent and delay-insensitive design styles are too inefficient for use in many applications. For maximum performance, asynchronous design styles will need to exploit some knowledge about the timing of their components and the environment. Hence, timing analyzers will be essential asynchronous design tools.

In this paper, we have presented an efficient timing analysis technique for determining global timing constraints of extended burst-mode circuits implemented according to the 3D design style. To the best of our knowledge, this is the first completely automated tool that performs such an analysis of gate-level 3D circuits with bounded component delays. Although timing constraints identified by our tool represent conservative approximations to the true timing requirements in the worst-case scenario, experimental results indicate that our results are accurate for almost all of the 3D benchmarks we analyzed. The efficiency and accuracy of our timing analysis technique makes it very easy for a designer to make incremental optimizations and hand-tunings to a 3D circuit and verify the timing properties almost immediately. This proves extremely helpful in the design cycle of practical 3D circuits.

The work reported in this paper can potentially be extended for other interesting applications. For example, one could modify our timing analysis algorithm to verify the correctness of a 3D design by simulation. Another interesting application could be to determine if the gate and wire-delays in a 3D circuit permit the elimination of some hardware required for insuring hazard-free behavior. This could be used, for example, to optimize the area of a 3D circuit. Subsequently, timing constraints required for correct operation of the modified circuit can be obtained using a modified version of the timing analysis algorithm reported in this paper.

References

[1] *Hitachi High Speed CMOS Gate Array: HG62E Series Design Manual*.

[2] P. Beerel and T. H.-Y. Meng. Automatic gate-level synthesis of speed-independent circuits. In *Proceedings of the 1992 IEEE/ACM International Conference on Computer Aided Design*, Nov. 1992.

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

[4] 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*, April 1997.

[5] 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 29th ACM/IEEE Design Automation Conference*, pages 165–72, 1992.

[6] 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*, 13(6):814–822, June 1994.

[7] S. Devadas, K. Keutzer, S. Malik, and A. Wang. Verification of asynchronous interface circuits with bounded wire delays. *IEEE Journal of VLSI Signal Processing*, 7(1-2):161–182, February 1994.

[8] J. C. Ebergen. A formal approach to designing delay-insensitive circuits. *Distributed Computing*, 5(3):107–119, 1991.

[9] E. B. Eichelberger. Hazard detection in combinational and sequential switching circuits. *IBM Journal of Research and Development*, 9, Mar. 1965.

[10] G. Fantauzzi. An algebraic model for the analysis of logical circuits. *IEEE Transactions on Computers*, C-23:576–581, June 1974.

[11] N. Ishiura. *Studies on Logic Simulation and Hardware Description Languages*. PhD thesis, Kyoto University, Kyoto, Japan, Dec. 1990.

[12] N. Ishiura, Y. Deguchi, and S. Yajima. Coded time-symbolic simulation using shared binary decision diagram. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pages 130–145, 1990.

[13] N. Ishiura, M. Takahashi, and S. Yajima. Time symbolic simulation for accurate timing verification of asynchronous behavior of logic circuits. In *Proceedings of the 26th ACM/IEEE Design Automation Conference*, pages 497–502, 1989.

[14] D. S. Kung. Hazard-non-increasing gate-level optimization algorithms. In *Proceedings of the 1992 IEEE/ACM International Conference on Computer Aided Design*, pages 631–634. IEEE Computer Society Press, November 1992.

[15] L. Lavagno, K. Keutzer, and A. Sangiovanni-Vincentelli. Synthesis of hazard-free asynchronous circuits with bounded wire delays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(1):61–86, Jan. 1995.

[16] K.-J. Lin, J.-W. Kuo, and C.-S. Lin. Direct synthesis of hazard-free asynchronous circuits from STGs based on lock relation and MG-decomposition approach. In *Proceedings of the European Design and Test Conference EDAC-ETC-EUROASIC*, pages 178–183, Mar. 1994.

[17] M. Linderman and M. Leeser. Simulation of digital circuits in the presence of uncertainty. In *Proceedings of the 1994 IEEE International Conference on Computer Aided Design*, pages 248–251, 1994.

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

[19] C. W. Moon. *Synthesis and Verification of Asynchronous Circuits from Graph Specifications*. PhD thesis, University of California, Berkeley, 1992.

[20] S. M. Nowick and D. L. Dill. Automatic synthesis of locally-clocked asynchronous state machines. In *Proceedings of the 1991 IEEE International Conference on Computer Aided Design*, pages 318–321, November 1991.

[21] S. M. Nowick and D. L. Dill. Synthesis of asynchronous state machines using a local clock. In *Proceedings of the 1991 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 192–197. IEEE Computer Society Press, October 1991.

[22] E. Ulrich, K. Lentz, S. Demba, and R. Razdan. Concurrent min-max simulation. In *EDAC. Proceedings of the European Conference on Design Automation*, pages 554–557, 1991.

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

[24] P. Vanbekbergen, F. Catthoor, G. Goossens, and H. De Man. Optimized synthesis of asynchronous control circuits from graph-theoretic specifications. In *Proceedings of the 1990 IEEE International Conference on Computer Aided Design*. IEEE Computer Society Press, 1990.

[25] A. V. Yakovlev, L. Lavagno, and A. Sangiovanni-Vincentelli. A unified signal transition graph model for asynchronous control circuit synthesis. *Proceedings of the 1992 IEEE International Conference on CAD*, Nov. 1992.

[26] K. Y. Yun and D. L. Dill. Automatic synthesis of 3D asynchronous finite-state machines. In *Proceedings of the 1992 IEEE/ACM International Conference on Computer Aided Design*, pages 576–580. IEEE Computer Society Press, November 1992.

[27] K. Y. Yun and D. L. Dill. Unifying synchronous/asynchronous state machine synthesis. In *Proceedings of the 1993 IEEE/ACM International Conference on Computer Aided Design*, pages 255–260. IEEE Computer Society Press, November 1993.