Temporal Specification Verification via Causal Reasoning

Alan R. Martello and Steven P. Levitan

Department of Electrical Engineering
University of Pittsburgh
Pittsburgh, PA 15261

Abstract

We present a technique for verifying the timing specifications of the interfaces between digital systems. The verification process takes as input the timing protocols of each component as well as the connectivity between the components. The technique proceeds in three steps. First, a graph is built, which describes the causal relationships of events which can occur in the complete system. Second, a set of requirements (from the specifications) are used to identify pairs of events which must (or must not) happen with a particular temporal relationship. Third, for each such requirement, the sequences of events which might lead to such a requirement violation are identified and traced to determine if the requirement is violated or satisfied. The technique supports protocols with time ranges on transitions, and conditional events based on dynamic sensitivity to system state.

Introduction: The Verification Problem

Attempts to reason about digital systems have been ongoing for many years [1–3]. Various methods for modeling time delays have been proposed and formal verification methods investigated [4–7]. Methods based on simulation have also been used to verify circuits[8,9]. Recent work has taken a symbolic approach to attempt to reason about circuits [10,11] as well as our previous work [12,13] which investigated a simpler verification technique, upon which this work is based.

As in our previous work, our goal is to provide a technique for automatically verifying the temporal protocols used in the interfaces between digital systems. The technique uses three kinds of information about the system it is verifying. First, are the external actions of each of the component sub-systems, or modules. These are (generally) taken by the module in response to external events from other modules. Second, are the temporal constraints, or *requirements*, which each module places on its environment in order for it to operate correctly. Taken together, these comprise the temporal protocols of the system. Third, are the actual interconnections between the inputs and outputs of the modules in the system. Given this information the technique is used to verify that *no* sequence of actions (events) taken by the modules will violate any of the requirements specified for the system.

In this work, the verification process is dependent on an underlying assumption that the system is completely specified (i.e., all information regarding the system is known). This assumption is necessary and sufficient for the verification process to occur. It is necessary since permitting the possibility of unknown information in the system affecting system operation results in incorrect reasoning. An incompletely specified system is one where not all information is known about how all control signals in the system interact. Taken to an extreme, this implies that signals may change values in an unpredictable manner, at any time, precluding the possibility of verifying correct behavior. The assumption that the system is fully specified is sufficient for verification. If all information is known about the system, then, within the scope of the model used, no unanticipated action can occur in the system which could affect the verification process.

The verification methodology analyzes relationships between events on signals.[1] A signal is a physical entity (e.g., a wire) or a virtual identifier (e.g., a variable) which can have two values, high and low. Restricting signals to two values can be done without loss of generality in the verification

[1]This section provides an overview of the terminology which will be used; formal definitions appear in [12,14].

process since we can use variables to represent wires with arbitrary values. The transition of a signal from high to low is called a falling event and the transition from low to high is a rising event. Although the system supports time ranges for when events occur, the events themselves occur instantaneously. The syntax `A/` (`A`$\uparrow$) and `A\` (`A`$\downarrow$) denotes rising and falling events on signal `A` respectively.

Within this framework, the two kinds of statements which a designer can use to describe the interface protocols are *causality* statements and *requirement* statements. These are analogous to "recommended conditions" and "switching characteristics" in published data sheets. They are also similar to Seitz's *functional relations* and *domain relations* [15]. These statements describe relationships between events.

The first type of statement describes causality: "$e_1[when\ w_1] \rightarrow e_2[when\ w_2]\ (A, B)$." In this expression: e_1 and e_2 are events (such as rising or falling); w_1 and w_2 are boolean expressions called *enabling expressions*; and (A, B) is a min/max time range. The meaning of this expression is that if event e_1 happens when w_1 is true then event e_2 will occur provided w_2 is true within the time range defined by A and B.

The second kind of statement expresses the constraints that a device puts on the rest of the environment in order for it to operate correctly. This is a requirement on the temporal relationships of signals to ensure correct operation of the device. During verification, a requirement is checked by comparing the temporal relationship between its two events and determining if the required timing is violated. There are two types of requirements in the system: positive requirements and negative requirements. The positive requirement of "$e_1[when\ w_1] \mid e_2[when\ w_2]\ (A, B)$" means that event e_2 and event e_1 are related such that event e_2 must occur no less than A and no more than B after event e_1. If the enabling expressions w_1 or w_2 are present, then w_1 or w_2 must be true when e_1 or e_2 occurs for the requirement to be verified, otherwise the requirement is ignored.

The symmetric case to the positive requirement, called the negative requirement, is also supported. The negative requirement is expressed as "$e_1[when\ w_1] <> e_2[when\ w_2](A, B)$" and means that event e_2 and event e_1 are related such that event e_2 must *not* occur within the range $A \leq t \leq B$ after event e_1. If the enabling expressions w_1 or w_2 are present, then w_1 or w_2 must be true when e_1 or e_2 occurs for the requirement to be verified, otherwise the requirement is ignored.

The enabling expressions on all three types of statements are conjunctions of signal values and negated signal values. For statements with enabling expressions, the values of signals in the system must be consistent with the expression in order for the event to be active.

The tracing of the causalities to determine the relative time of events is complicated by two properties: time ranges on events and system state affecting enabling expressions. The handling of these properties correctly is both difficult and necessary for the verification of non-trivial systems.

The remainder of this paper provides a description of the verification process. First, the analysis procedure based on graph construction and traversal is presented. Next, the search process and system history creation is described. Finally, the verification process using a complete system history is discussed.

Event Graph Construction and Analysis

Given a set of causalities which define the system to be verified, a causal event graph is built where the nodes of the graph are the events and the directed arcs in the graph are the causal relationships. Each arc is annotated with the relative time range, Δ, and the two enabling expressions, w_1 and w_2. The requirements are not kept explicitly in the graph but are used to guide the search process. Each requirement specifies a timing relationship which must be maintained between two events, e_1 and e_2, known as the requirement nodes or events. If these events ever happen, then they appear in *some* causality and therefore the events are represented as nodes in the event graph.

Once the event graph is built, each requirement can be verified by tracing paths through the graph which lead to the two events constrained by the requirement. To verify the timing relationship, a common time reference must be established between e_1 and e_2. This is accomplished by finding all nodes in the graph which are ancestors to the requirement nodes. Ancestors of the requirement nodes are all of the nodes in the graph which can cause the requirement event through one or multiple causalities. All events in the graph which are ancestors to *both* requirement events are called the $ParentSet$. Only nodes in the $ParentSet$ can provide a common time reference between e_1 and e_2 and in a fully specified system, only nodes in the $ParentSet$ can possibly *cause* a conflict.

For each node $e \in ParentSet$, the graph is analyzed to determine if a single occurrence of e, the common reference event, can cause both e_1 and e_2 in the system. Although e may cause e_1 or e_2 in isolation, it may not be possible for e to result in both e_1 *and* e_2. The paths in the graph from e to e_1 and from e to e_2 may be mutually exclusive due to the enabling expressions associated with each causality along the path. The determination of possible paths through the graph is discussed in the next section.

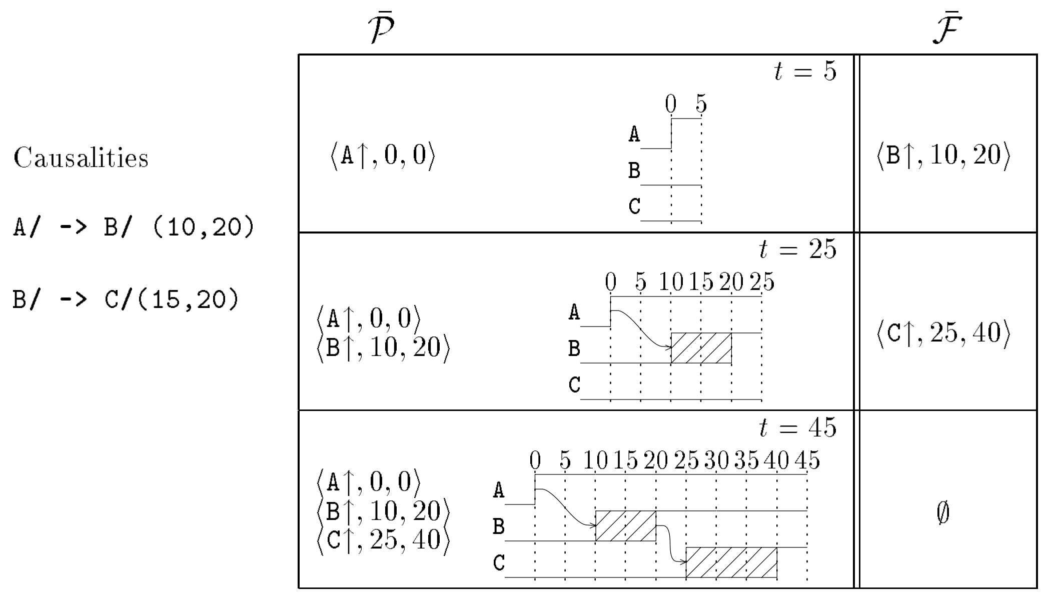
Creating System Histories

To verify a requirement, an analysis must be performed to determine if there is a path from $e \in ParentSet$ to each of the requirement nodes, e_1 and e_2. If a path can occur from e to one of the requirement nodes, then that path is said to be *active*. Only if there are two active paths, one from e to each requirement node, do e_1 and e_2 have a common time reference in e. This is the only case where the relative time between e_1 and e_2 needs to be checked for requirement violation. If both paths are not active, then it is not possible for a single occurrence of e to generate both e_1 and e_2 and therefore it cannot cause a violation.

The forward analysis in time from a single event transition, e, is referred to as a system history, $\mathcal{H}$. A system history is a particular set of paths (or tree) through the graph representing a distinct evolution of events based on a subset of all possible signal interactions. There are many possible system histories which can be constructed from the occurrence of e. The histories may appear identical before they "branch" into unique sequences of transitions. Some histories may contain e_1 and e_2 and other histories may not.

The size of the search space of all possible system histories from a single reference event e can be extremely large for even a small system. The nature of this problem leads to a tree-like search space which has many alternatives or branch points. The reduction of the size of the search space by minimizing the number of branch points is key to keeping the problem within a manageable scope. This reduction is performed by: using a directed search based on the requirements, performing reasoning with symbolic time ranges, and minimizing the bifurcation by splitting only when needed.

A system history consists of two sets of events: events which have been fully traced are kept in $\bar{\mathcal{P}}$ and their descendants which have not yet been fully traced are kept in $\bar{\mathcal{F}}$. System histories are constructed as follows. As each event is moved from $\bar{\mathcal{F}}$ to $\bar{\mathcal{P}}$, all causalities which have that event as their left-hand side (LHS) *and* whose conditional is true have their right-hand side (RHS) event added to $\bar{\mathcal{F}}$. As events are traced through the graph, histories are built up by moving events incrementally from $\bar{\mathcal{F}}$ to $\bar{\mathcal{P}}$.



$\bar{\mathcal{P}}$
$\bar{\mathcal{F}}$
Causalities
A/ -> B/ (10,20)
B/ -> C/(15,20)
$t = 5$
$\langle A\uparrow, 0, 0\rangle$
0 5
A
B
C
$\langle B\uparrow, 10, 20\rangle$
$t = 25$
$\langle A\uparrow, 0, 0\rangle$
$\langle B\uparrow, 10, 20\rangle$
0 5 10 15 20 25
$\langle C\uparrow, 25, 40\rangle$
$t = 45$
$\langle A\uparrow, 0, 0\rangle$
$\langle B\uparrow, 10, 20\rangle$
$\langle C\uparrow, 25, 40\rangle$
0 5 10 15 20 25 30 35 40 45
$\emptyset$

Figure 1: Illustration of system history creation

Figure 1 shows an example of the creation of a system history. The system consists of two causalities and an event, A$\uparrow$ which occurs at $t = 0$. The figure illustrates how events are inserted into $\bar{\mathcal{F}}$ and then moved from $\bar{\mathcal{F}}$ to $\bar{\mathcal{P}}$ as the system history is constructed. At a given time, the combination of $\bar{\mathcal{P}}$ and $\bar{\mathcal{F}}$ is a system history which uniquely describes the state of the system.

To meet the goal of verification, we must construct all of the possible *complete* system histories which may cause a requirement violation. A complete system history is a system history which has reached one of two termination conditions: either $\bar{\mathcal{F}}$ is empty (as in Figure 1), or $\bar{\mathcal{F}}$ is identical to a previous instance of $\bar{\mathcal{F}}$ for that history. In the second case, the system history has cycled and no new states will be explored by continuing to move events from $\bar{\mathcal{F}}$ to $\bar{\mathcal{P}}$.

The construction of system histories is complicated by the interaction of enabling expressions of the causalities. The enabling expressions can cause a bifurcation or split in the system history search process and therefore result in multiple system histories.

Figure 2 illustrates the causality `A/ when C -> B/ (10,10)` knowing that `A/` occurs from $0 \leq t \leq 20$, `C` falls at $t = 5$ and `C` rises at $t = 15$. In the analysis of this causality, there are three cases to consider producing a three-way branch in the system history. The first case is if `A` rises in the range $0 \leq t \leq 5$. In this case, `A` will cause `B` to rise as shown by `B'`. The second case is if `A` rises in

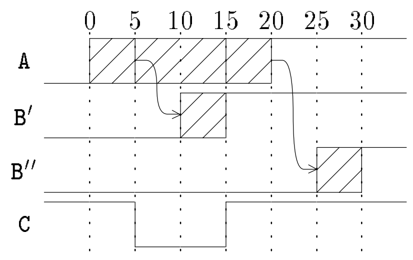


Figure 2: `A/ when C -> B/ (10,10)`

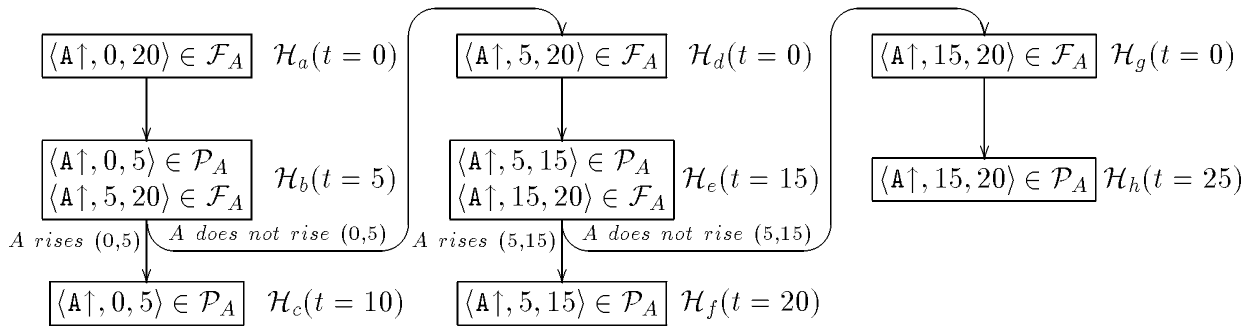

$\langle A\uparrow, 0, 20\rangle \in \mathcal{F}_A$
$\mathcal{H}_a(t = 0)$
$\langle A\uparrow, 5, 20\rangle \in \mathcal{F}_A$
$\mathcal{H}_d(t = 0)$
$\langle A\uparrow, 15, 20\rangle \in \mathcal{F}_A$
$\mathcal{H}_g(t = 0)$
$\langle A\uparrow, 0, 5\rangle \in \mathcal{P}_A$
$\langle A\uparrow, 5, 20\rangle \in \mathcal{F}_A$
$\mathcal{H}_b(t = 5)$
$\langle A\uparrow, 5, 15\rangle \in \mathcal{P}_A$
$\langle A\uparrow, 15, 20\rangle \in \mathcal{F}_A$
$\mathcal{H}_e(t = 15)$
$\langle A\uparrow, 15, 20\rangle \in \mathcal{P}_A$
$\mathcal{H}_h(t = 25)$
A rises (0,5)
A does not rise (0,5)
A rises (5,15)
A does not rise (5,15)
$\langle A\uparrow, 0, 5\rangle \in \mathcal{P}_A$
$\mathcal{H}_c(t = 10)$
$\langle A\uparrow, 5, 15\rangle \in \mathcal{P}_A$
$\mathcal{H}_f(t = 20)$

Figure 3: Events on signal `A` during the construction of three system histories

the range $5 \leq t \leq 15$. In this case, the causality is not triggered and `A/` does not cause `B` to rise. The third case is if `A` rises in the range $15 \leq t \leq 20$. In this case, `A` will cause `B` to rise as shown by `B″`. This simple example illustrates that although `A` rises in a simple time range, the next step in the system history must be considered by a case analysis of the three possibilities and denotes a system history branch point.

The creation of the three system histories hinges on the interaction of `A/` with signal `C`. Figure 3 illustrates how the `A` transition is moved from $\bar{\mathcal{F}}$ to $\bar{\mathcal{P}}$ incrementally. When an interaction with `C` occurs which requires a split in the system history, time is "rolled back" and the current `A/` transition is bifurcated. As such, the history $\mathcal{H}_b$ results in a split where `A/` is split into the region from $0 \leq t \leq 5$ and $5 \leq t \leq 20$; further processing causes the $\mathcal{H}_e$ to be split. The three complete histories where `A/` occurs in $0 \leq t \leq 5$, $5 \leq t \leq 15$, and $5 \leq t \leq 20$ can be seen in Figures 4, 5, and 6 respectively. These figures also identify the events as they are incrementally transferred from $\bar{\mathcal{F}}$ to $\bar{\mathcal{P}}$.

A troublesome problem for some systems is the correct handling of reconvergent paths. In our technique, reconvergent paths are handled during the construction of $\bar{\mathcal{P}}$. We enforce the restriction that the time ranges of events for a given signal on $\bar{\mathcal{P}}$ cannot overlap. This restriction is maintained through the bifurcation of system histories whenever the time ranges of two events on the same signal in $\bar{\mathcal{P}}$ would overlap when moving an event from $\bar{\mathcal{F}}$ to $\bar{\mathcal{P}}$.

Causality

`A/ when C -> B/ (10,10)`

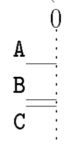
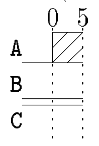
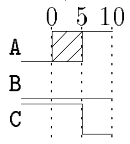
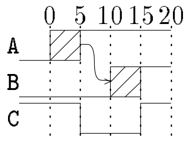
$\bar{\mathcal{P}}$		$\bar{\mathcal{F}}$
	$\mathcal{H}_a(t=0)$ 	$\langle$`A`$\uparrow, 0, 20\rangle$ $\langle$`C`$\downarrow, 5, 5\rangle$ $\langle$`C`$\uparrow, 15, 15\rangle$
$\langle$`A`$\uparrow, 0, 5\rangle$ $\langle$`C`$\downarrow, 5, 5\rangle$	$\mathcal{H}_b(t=5)$ 	$\langle$`A`$\uparrow, 5, 20\rangle$ $\langle$`B`$\uparrow, 10, 15\rangle$ $\langle$`C`$\uparrow, 15, 15\rangle$
$\langle$`A`$\uparrow, 0, 5\rangle$ $\langle$`C`$\downarrow, 5, 5\rangle$	$\mathcal{H}_c(t=10)$ 	$\langle$`B`$\uparrow, 10, 15\rangle$ $\langle$`C`$\uparrow, 15, 15\rangle$
$\langle$`A`$\uparrow, 0, 5\rangle$ $\langle$`B`$\uparrow, 10, 15\rangle$ $\langle$`C`$\downarrow, 5, 5\rangle$ $\langle$`C`$\uparrow, 15, 15\rangle$	$\mathcal{H}_{end}(t=20)$ 	$\emptyset$

Figure 4: Complete history resulting if `A` transitions in $0 \leq t \leq 5$

An example of a reconvergent path is shown in Figure 7. For this example, the "naive" waveform is what a designer might sketch where `A`$'$ and `A`$''$ are both transitions of signal `A` which overlap in time. Extending the concepts from the previous example with the added condition that transitions on the same signal cannot overlap results in three individual complete histories, $\mathcal{H}_x$, $\mathcal{H}_y$, and $\mathcal{H}_z$. These histories result from two bifurcations applied to the original transition `A/`. Thus, the reconvergent path for signal `A` is split as needed and each case is fully traced.

System History Verification

Once a complete system history is built, the timing relationship between events can be evaluated to determine if the particular requirement being tested is satisfied or violated. Since the requirement being verified specifies two events, e_1 and e_2 with their associated enabling expressions w_1 and w_2, then the verification process compares all combinations of e_1 *when* w_1 and e_2 *when* w_2 present in the complete system history. Each comparison directly yields whether this requirement has been satisfied or violated. The entire system has been verified once each complete system history generated for each requirement has been tested.

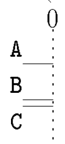
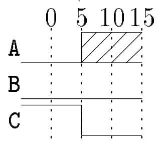
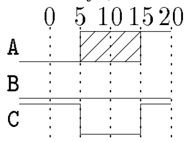
	$\bar{\mathcal{P}}$		$\bar{\mathcal{F}}$
		$\mathcal{H}_d(t=0)$ 	$\langle$A$\uparrow, 5, 20\rangle$ $\langle$C$\downarrow, 5, 5\rangle$ $\langle$C$\uparrow, 15, 15\rangle$
Causality `A/ when C -> B/ (10,10)`	$\langle$A$\uparrow, 5, 15\rangle$ $\langle$C$\downarrow, 5, 5\rangle$ $\langle$C$\uparrow, 15, 15\rangle$	$\mathcal{H}_e(t=15)$ 	$\langle$A$\uparrow, 15, 20\rangle$ $\langle$B$\uparrow, 15, 25\rangle$
	$\langle$A$\uparrow, 5, 15\rangle$ $\langle$C$\downarrow, 5, 5\rangle$ $\langle$C$\uparrow, 15, 15\rangle$	$\mathcal{H}_f(t=20)$ 	$\emptyset$

Figure 5: Complete history resulting if `A` transitions in $5 \leq t \leq 15$

Conclusion

This paper has presented a technique for a temporal verification system for digital interfaces. The verification process searches the space of possible system timing behaviors, looking for timing violations. This search process is guided by the knowledge of the timing relationships that must not be violated to ensure proper operation. This reduces the complexity of the search problem with the tradeoff that the verification process performs only safety tests and not liveness tests.

The advantages of this approach are that it does not differentiate between synchronous and asynchronous systems and works equally well for both. In addition, the system does not need additional "hints", user intervention, or require specific design styles. Finally, the technique supports protocols with time ranges on transitions, and conditional events based on dynamic sensitivity to system state.

Causality

`A/ when C -> B/ (10,10)`

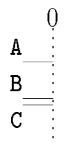
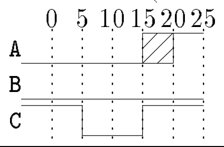
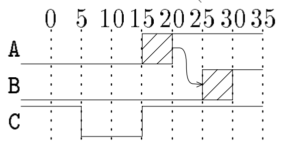
$\bar{\mathcal{P}}$		$\bar{\mathcal{F}}$
	$\mathcal{H}_g(t=0)$ 	$\langle$A$\uparrow, 15, 20\rangle$ $\langle$C$\downarrow, 5, 5\rangle$ $\langle$C$\uparrow, 15, 15\rangle$
$\langle$A$\uparrow, 15, 20\rangle$ $\langle$C$\downarrow, 5, 5\rangle$ $\langle$C$\uparrow, 15, 15\rangle$	$\mathcal{H}_h(t=25)$ 	$\langle$B$\uparrow, 25, 30\rangle$
$\langle$A$\uparrow, 15, 20\rangle$ $\langle$B$\uparrow, 25, 30\rangle$ $\langle$C$\downarrow, 5, 5\rangle$ $\langle$C$\uparrow, 15, 15\rangle$	$\mathcal{H}_{end}(t=35)$ 	$\emptyset$

Figure 6: Complete history resulting if A transitions in $15 \leq t \leq 20$

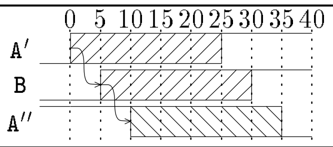
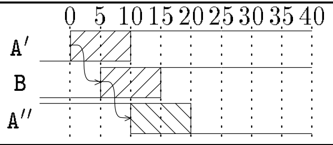
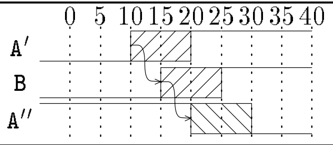
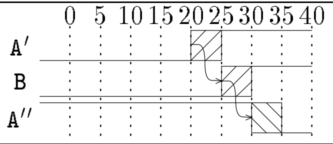
"Naive" Waveform	
$\mathcal{H}_x$	
$\mathcal{H}_y$	
$\mathcal{H}_z$	

Figure 7: Reconvergent Path with causalities `A/ -> B/ (5,5)` and `B/ -> A\(5,5)`

References

[1] Thomas M. McWilliams, "Verification of Timing Constraints on Large Digital Systems," *Proc. 17th Design Automation Conference* (June, 1980).

[2] Robert B. Hitchcock, Sr., "Timing Verification and the Timing Analysis Program," *Proc. 19th Design Automation Conference* (June, 1982).

[3] John J. Wallace, "On Automatic Verification of SLIDE Descriptions," Design Research Center, Carnegie-Mellon University, DRC-01-2-80, Pittsburgh, PA, Aug., 1979.

[4] David E. Wallace, "Abstract Timing Verification for Synchronous Digital Systems," Computer Science Division, Univ. of Calif. at Berkeley, UCB/CSD 88/425, Berkeley, CA, June 27, 1988.

[5] David L. Dill, "Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits," Dept. of Computer Science, Carnegie-Mellon University, CMU-CS-88-119, Pittsburgh, PA, Feb., 1988.

[6] M. Browne, E. Clarke, D. Dill & B. Mishra, "Automatic Verification of Sequential Circuits Using Temporal Logic," Dept. of Computer Science, Carnegie-Mellon University, CMU-CS-85-100, Pittsburgh, PA, Dec., 1984.

[7] Patrick C. McGeer & Robert K. Brayton, "Timing Analysis in Precharge / Unate Networks," *Proc. 27th Design Automation Conference* (June, 1990).

[8] Tod Amon & Gaetano Borriello, "On the Specification of Timing Behavior," *Proceedings of the 1990 ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU '90)*, New York, NY, SIGDA (Aug., 1990).

[9] Dimitrie Doukas & Andrea S. LaPaugh, "CLOVER: A Timing Constraints Verification System," *Proceedings of the 1990 ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU '90)*, New York, NY, SIGDA (Aug., 1990).

[10] F. Mavaddat & T. Gahlinger, "On Deducing Tight Bounds from Partial Timing Specifications," *Proceedings of the 1990 ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU '90)*, New York, NY, SIGDA (Aug., 1990).

[11] Michael C. McFarland, "CPA: Giving an Account of Timed Systme Behavior," *Proceedings of the 1990 ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU '90)*, New York, NY, SIGDA (Aug., 1990).

[12] Alan R. Martello & Steven P. Levitan, "Causal Timing Verification," *Proceedings of the 1990 ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU '90)*, New York, NY, SIGDA (Aug., 1990).

[13] Alan R. Martello, Steven P. Levitan & Donald M. Chiarulli, "Timing Verification Using HDTV," *Proc. 27th Design Automation Conference* (June, 1990).

[14] Alan R. Martello, "Temporal Specification Verification," Dept. of Elect. Eng., University of Pittsburgh, PhD. Dissertation (in preparation).

[15] Charles L. Seitz, "System Timing," in *Introduction to VLSI Systems*, Carver Mead & Lynn Conway, eds., Addison Wesley, Reading, MA, 1980, 245.