Consistency and Satisfiability of Waveform Timing Specifications*

J. A. Brzozowski, T. Gahlinger, and F. Mavaddat
Department of Computer Science and VLSI Group, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

Manufacturers often use digital waveforms to specify critical device timing. In this paper, we study two problems related to the use of such specifications. First, we are interested in verifying that the timing information is consistent to begin with. Second, given waveform specifications of two devices that are to be linked, we wish to know whether one satisfies the other's timing requirements. We construct a model of the timing information conveyed by the waveform convention and show how both problems can be solved efficiently with optimization techniques. To illustrate our arguments, we compare the write-cycle timing of a typical CPU with that of a RAM device.

1. INTRODUCTION

The proper timing of digital signals is a major concern in the design of reliable systems, especially in the light of dramatic increases in the speed and complexity of digital components. In essence, *improper* timing occurs when signals arrive at device inputs too early or too late to be recognized. From among the numerous recent studies that deal with such issues, we cite Hitchcock's work on verifying that circuit delays meet specified clock requirements [2] and Unger and Tan's analysis of optimal clock speeds for given known circuit delays [10].

To date, the main thrust of the research in digital timing, including the aforementioned, has been concerned with the *design* of reliable components. Typically, the investigator has at his/her disposal a comprehensive body of information related to the circuits under study, with the capability to modify the circuitry to resolve timing problems.

In this paper, we are interested in the types of timing problems a system engineer encounters when *connecting* prefabricated components. In this case, the engineer depends on the information specified in the manufacturer's data sheets. Not only is his/her knowledge of device behavior limited to timing specifications of the signal behavior at device ports, but the specifications are, by their very nature, often *partial* descriptions of the timing of events at the ports.

*This research was supported by the Natural Sciences and Engineering Council of Canada under grant A0871.

CCC 0028-3045/91/01091-017$04.00

The main question we wish to address here is: Can a set of devices be connected reliably given the manufacturers' timing specifications? A similar line of investigation was undertaken recently by Borriello [1], who developed several software tools for synthesizing transducer circuitry. However, the approach is rather informal and lacks a rigorous mathematical treatment of the subject.

The work reported here is a first step toward a more general formal treatment of timing in device connections. In particular, we restrict ourselves to timing specifications that can be formulated as conjunctions of linear inequalities. We also emphasize at the outset that we ignore physical and electrical aspects of the connections and that these must ultimately be taken into consideration to ensure the proper timing of device connections; an extensive list of references to such matters may be found in Ruehli and Ostapko [6].

In the next section, we present an informal description of waveform timing specifications and introduce the problems of timing consistency and satisfiability. In Section 3, we formulate the specifications in terms of linear inequalities. In Section 4, we formulate the timing problems as linear programming problems and show how their restricted nature leads to efficient network solutions. We then apply these methods, in Section 5, to verify the timing of a set of devices and the connections between them.

2. TIMING OF CONNECTED DEVICES

Data sheets for a digital device usually include the critical timing of activities at device ports. Consider a typical microprocessor chip, such as the Zilog Z80B, an eight-bit CPU [12]. For the sake of discussion, we will focus on the essential timing of the data-write operation. Figure 1 shows the main ports involved in this operation.

A general discussion of timing specifications, and the derivation of timing parameters, can be found in Wiatrowski and House [11]. However, in this paper, we are more concerned with how such information is specified, and how the specifications are used when connecting devices, than we are with how the timings were derived. Figure 2 shows the write-cycle timing of the Z80B; the device is designed to operate with a maximum clock frequency of 6 MHz.

Some of the conventions used in such specifications are explained in Figure 3 and warrant a brief digression. Additional references to the subject can be found in Rony [5] and Springer [7].

A waveform describes port behavior as an alternating sequence of *steady-state* and *transition* phases, indicated by horizontal and diagonal line segments, respectively. Waveforms (a) and (b) are used to represent single signal transitions.

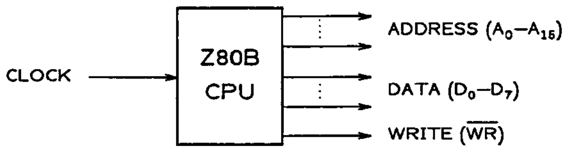

FIG. 1. Basic write cycle ports of a typical microprocessor.

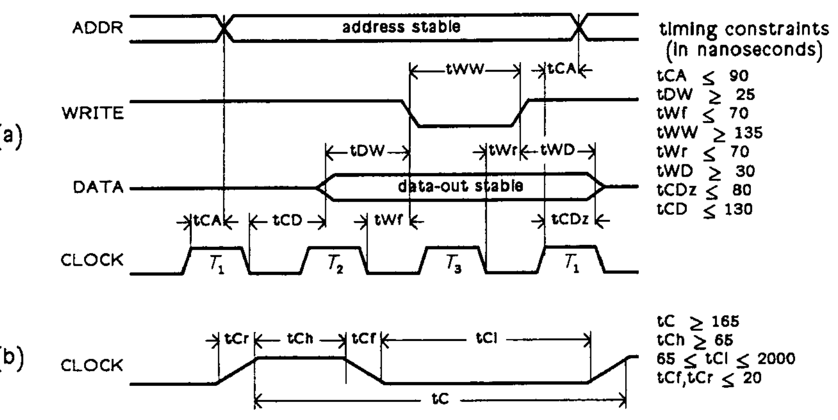

FIG. 2. (a) Write cycle timing for the Z80B CPU. (b) Clock timing.

Waveforms (c) and (d), in our examples, represent bus signals. A critical separation between two transitions is referred to as a *timing constraint*, indicated by parameter t_d in (e). Timing constraints are specified as *worst-case* upper or lower bounds on transition delay times and are measured from a given percentage of the transition between voltage levels (50% and 10%/90% levels are typical). For the sake of discussion, when transition timings are not specified, we will assume that all constraints are measured from the 50% level of transition.

An illustration of the form shown in Figure 2 is variously referred to as a *timing diagram*, *delay specification*, or *waveform specification*. We will shorten the latter expression to *wavespec* and, henceforth, use this term to refer to such diagrams. The first problem that we study in this paper is the following:

Timing Consistency: Verify that the timing information contained in a wavespec is consistent.

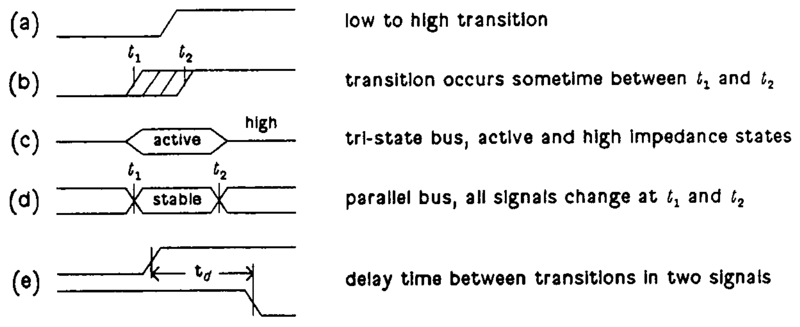
(a)
low to high transition
(b)
t_1
t_2
transition occurs sometime between t_1 and t_2
(c)
active
high
tri-state bus, active and high impedance states
(d)
t_1
t_2
stable
parallel bus, all signals change at t_1 and t_2
(e)
t_d
delay time between transitions in two signals

FIG. 3. Timing diagram conventions.

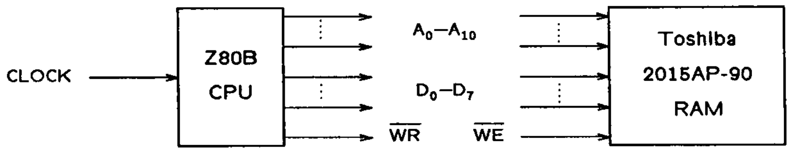

FIG. 4. Connecting the CPU to a RAM device.

Assume now that a designer wants to connect the Z80B CPU to a suitable memory device, such as the Toshiba 2015AP-90, a 2K-word eight-bit static RAM [9] (see Fig. 4). He/she would connect address ports A_0–A_{10} of the CPU to ports A_0–A_{10} of the RAM and port $\overline{WR}$ of the CPU to port $\overline{WE}$ of the RAM; the data port connections would be made in the obvious way.

For the RAM to function properly, the address has to reach the memory early enough to ensure writing to the proper location and both address and data have to be stable sufficiently long for the write to be performed correctly. These conditions are specified in the write cycle wavespec for the RAM, shown in Figure 5.

If the connection is to work properly, the designer must verify that the CPU and RAM wavespecs are "compatible" in some sense. One sense of compatibility is that corresponding port signal phases must match; for example, both CPU data-out and RAM data-in phases represent periods of stable data, etc. Indeed, it can be shown that all respective phases of the two wavespecs match, but a discussion of this is beyond the scope of this paper and we assume that the designer has established this using other methods.

The other sense of compatibility concerns the timing of corresponding phases. Note that, aside from the clock, all CPU ports are outputs for the write operation. We interpret the CPU wavespec to mean that, if the clock input meets the timing specified in Figure 2(b), the CPU produces output that meets the timing specified in Figure 2(a). Next, note that all RAM ports are input ports for the write operation. We interpret the RAM wavespec to mean that the RAM will perform a proper write operation if a connected device, such as the CPU, meets the timing specified in Figure 5.

For convenience, we refer to the timing constraints on the input signals of a

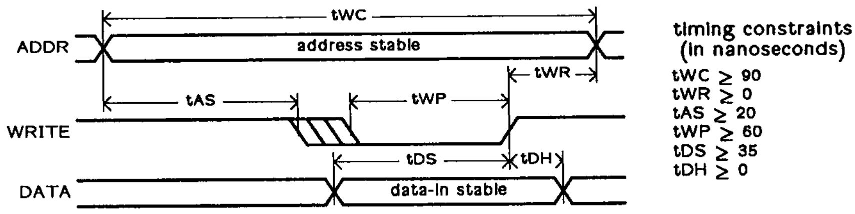

FIG. 5. Write cycle wavespec for the Toshiba RAM.

device as *required timing* and constraints on the output signals as *produced timing*. The second problem that we study in this paper can now be stated as follows:

Timing Satisfiability: Verify that the produced timings of one device *meet* or *satisfy* the required timings of another device.

Typically, a solution to the satisfiability problem requires a careful inspection and comparison of bounds between corresponding pairs of transition times. For example, it is clear that the CPU constraints tWW $\geq$ 135 satisfies the corresponding RAM constraint tWP $\geq$ 60. However, the RAM wavespec contains the minimum write-recovery-time constraint tWR $\geq$ 0, for which the CPU wavespec has no corresponding bounds. The designer then has to see whether corresponding bounds can be deduced from other CPU constraints, for instance, from tWr, tCA, and the clock constraints.

In the next section, we abstract the essential timing information contained in wavespecs.

3. ABSTRACTION OF WAVESPEC TIMING

We model the timing of a wavespec in terms of the waveform transition times, which we refer to as *events*, and by sets of linear constraints on the events. We begin by identifying the events and then derive the constraints from the timing specifications.

3.1. Transition Events

Let $T = \{t_1, \ldots, t_n\}$ be a set of integer-valued variables representing transition times on waveforms. Note that the use of integers to represent transition times is not a severe restriction in practice, since the necessary resolution can be attained with a suitable scaling factor.

We identify waveform transition events in one of two ways. If the transition timing is specified, we mark the transition with consecutive events, t_i and t_{i+1}, as shown in Figure 6(a). Otherwise, we abstract the transition time as the instant between two steady-state phases and mark this with an event, t_i, as shown in Figure 6(b); note that this abstraction eliminates the transition phase.

In our examples, all transitions, aside from those of the CPU clock, are assumed to be instantaneous. Figure 7 illustrates the abstracted CPU and RAM

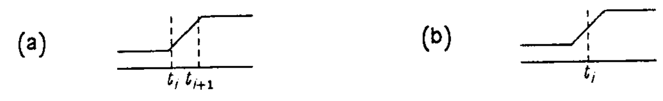

FIG. 6. Waveform transitions: (a) duration > 0. (b) instantaneous.

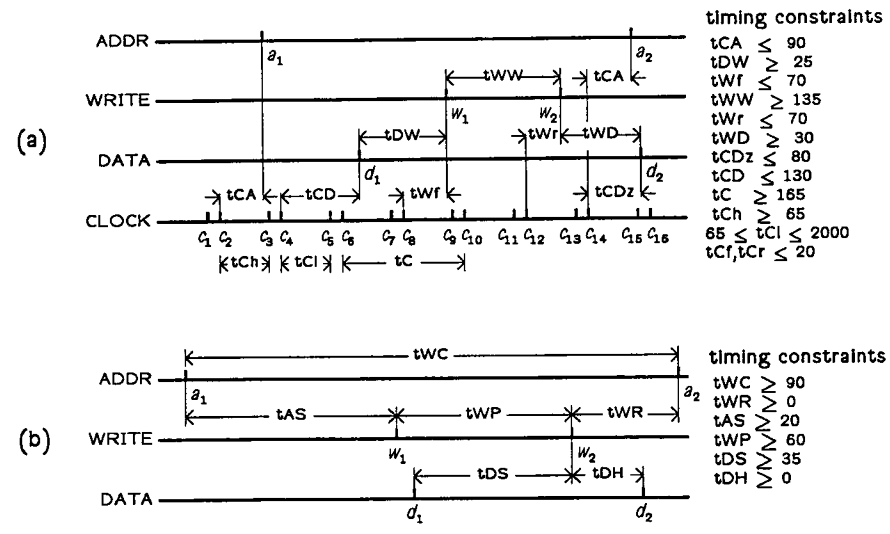

FIG. 7. Abstracted write-cycle wavespecs: (a) CPU. (b) RAM.

wavespecs, with sets of events

$$T^{\text{cpu}} = \{a_1,a_2,w_1,w_2,d_1,d_2,c_1, \ldots ,c_{16}\}$$

and

$$T^{\text{ram}} = \{a_1,a_2,w_1,w_2,d_1,d_2\}.$$

Following established practice, we use common names to indicate interconnected ports. After connection, the nonclock events of the CPU coincide with those of the RAM; hence, we use the same symbols in both wavespecs.

3.2. Constraints on Transition Events

We now formulate the wavespecs as sets of linear constraints on pairs of transition events. The constraints are derived from the sequencing of events within a waveform and from the specified timing parameters. We use R and P to distinguish between required and produced timings. For our purposes, a bound on a pair of input events is a required timing and a bound on a pair containing an output event is a produced timing.

Let R_{seq} and P_{seq} be the sets of required and produced sequencing constraints, respectively. It is reasonable to assume that steady-state phases have positive duration. Since transition phases that are preserved by the event abstraction have positive duration as well, it should be clear that every waveform event sequence is strictly ordered.

The definition of the sequencing constraints for our examples follows directly from these assumptions:

$$R_{\text{seq}}^{\text{cpu}} = \{c_i < c_{i+1},\ i = 1, \ldots, 15 \text{ (CLOCK)},$$

$$P_{\text{seq}}^{\text{cpu}} = \begin{cases} a_1 < a_2 & \text{(ADDR)} \\ w_1 < w_2 & \text{(WRITE)} \\ d_1 < d_2 & \text{(DATA)}, \end{cases}$$

and

$$R_{\text{seq}}^{\text{ram}} = \begin{cases} a_1 < a_2 & \text{(ADDR)} \\ w_1 < w_2 & \text{(WRITE)} \\ d_1 < d_2 & \text{(DATA)}, \end{cases}$$

Let R_{tim} and P_{tim} be the sets of specified required and produced timings. Under our transition-time abstraction, each constraint specifies a bound on the difference between two events; for example, tCD ≤ 130 is represented by $d_1 - c_4 \leq 130$. In addition, we assume a lower bound of zero for pairs bounded from above; thus $d_1 - c_4 \geq 0$.

For our examples, the timing constraints are

$$R_{\text{tim}}^{\text{cpu}} = \begin{cases} 165 \leq c_i - c_{i-4},\ i = 5, \ldots, 16 & \text{(tC)} \\ 65 \leq c_{4i-1} - c_{4i-2},\ i = 1, \ldots, 4 & \text{(tCh)} \\ 65 \leq c_{4i+1} - c_{4i} \leq 2000,\ i = 1, \ldots, 3 & \text{(tCl)} \\ 0 \leq c_{2i} - c_{2i-1} \leq 20,\ i = 1, \ldots, 8 & \text{(tCf,tCr)}, \end{cases}$$

$$P_{\text{tim}}^{\text{cpu}} = \begin{cases} 0 \leq a_1 - c_2 \leq 90 & \text{(tCA)} \\ 0 \leq a_2 - c_{14} \leq 90 & \text{(tCA)} \\ 25 \leq w_1 - d_1 & \text{(tDW)} \\ 0 \leq w_1 - c_8 \leq 70 & \text{(tWf)} \\ 135 \leq w_2 - w_1 & \text{(tWW)} \\ 0 \leq w_2 - c_{12} \leq 70 & \text{(tWr)} \\ 30 \leq d_2 - w_2 & \text{(tWD)} \\ 0 \leq d_2 - c_{14} \leq 80 & \text{(tCDz)} \\ 0 \leq d_1 - c_4 \leq 130 & \text{(tCD)}, \end{cases}$$

and

$$R_{\text{tim}}^{\text{ram}} = \begin{cases} 90 \leq a_2 - a_1 & \text{(tWC)} \\ 0 \leq a_2 - w_2 & \text{(tWR)} \\ 20 \leq w_1 - a_1 & \text{(tAS)} \\ 60 \leq w_2 - w_1 & \text{(tWP)} \\ 35 \leq w_2 - d_1 & \text{(tDS)} \\ 0 \leq d_2 - w_2 & \text{(tDH)}. \end{cases}$$

3.3. Abstracted Wavespecs

For notational convenience, we transform the constraints of the previous section into constraints of the form

$$t_j - t_i \leq a_{ij},$$

where t_i and t_j are transition events and a_{ij} is an integer bound. To do so, we make use of the following transformations:

$$t_j < t_i \rightarrow t_j - t_i < 0 \rightarrow t_j - t_i \leq -1 \tag{1}$$

$$t_j \leq t_i \rightarrow t_j - t_i \leq 0 \tag{2}$$

$$t_i - t_j \geq a_{ij} \rightarrow t_j - t_i \leq -a_{ij} \tag{3}$$

$$t_j - t_i = a_{ij} \rightarrow t_j - t_i \leq a_{ij} \text{ and } t_i - t_j \leq -a_{ij}. \tag{4}$$

Transformation (1) is based on the assumption that steady-state phases have a minimum duration of one time unit (nanosecond [ns] in our examples); choosing a smaller unit simply requires scaling all bounds accordingly. The remaining transformations should be self-evident.

If the application of (1)–(4) generates multiple bounds on a difference $t_j - t_i$, we discard the redundant ones. For example, the RAM timing constraint $a_2 - a_1 \geq 90$ transforms to $a_1 - a_2 \leq -90$, and the sequencing constraint $a_1 < a_2$ transforms to $a_1 - a_2 \leq -1$. Since $a_1 - a_2 \leq -90$ implies $a_1 - a_2 \leq -1$, we discard $a_1 - a_2 \leq -1$.

We say that a set of constraints is in *canonical form* if it has been modified in the foregoing way. Subsequently, we assume that all sets of constraints are in canonical form.

Definition 1. An (*abstracted*) *wavespec*, Ω, is a triple, $\Omega = (T,R,P)$, where T is the set of transition events in Ω, and R and P are canonical forms of the required timing, $R_{\text{seq}} \cup R_{\text{tim}}$, and produced timing, $P_{\text{seq}} \cup P_{\text{tim}}$, respectively. ■

For our examples, the abstracted wavespecs are defined as

$$\Omega^{\text{cpu}} = (T^{\text{cpu}},R^{\text{cpu}},P^{\text{cpu}})$$

and

$$\Omega^{\text{ram}} = (T^{\text{ram}},R^{\text{ram}},\emptyset).$$

By way of illustration, R^{ram}, in canonical form, is given as follows:

$$R^{\text{ram}} = \begin{cases} a_1 - a_2 \le -90 & (-1) \\ a_1 - w_1 \le -20 & \\ w_1 - w_2 \le -60 & (-135) \\ w_2 - a_2 \le 0 & \\ w_2 - d_2 \le 0 & (-30) \\ d_1 - w_2 \le -35 & \\ d_1 - d_2 \le -1 & (-1). \end{cases}$$

The numbers between parentheses show the corresponding bounds (if any) in P^{cpu}. We want to verify that the specified timings of each device are consistent and that the CPU timings meet the RAM requirements. Some of the RAM requirements are explicitly met by the CPU inequalities. Thus, for example, the CPU "guarantees" $w_1 - w_2 \le -135$, which "satisfies" the RAM requirement $w_1 - w_2 \le -60$. On the other hand, in the case of $a_1 - a_2$, the only inequality for the CPU is $a_1 - a_2 \le -1$, which arises from the sequencing constraint $a_1 < a_2$. This is not sufficient to satisfy the RAM requirement $a_1 - a_2 \le -90$. Finally, in the case of $a_1 - w_1$, there is no inequality at all in P^{cpu}. In the following sections, we show that the given CPU inequalities imply additional ones that *do* indeed satisfy the RAM requirements.

4. CONSISTENCY AND SATISFIABILITY

In the last section, we modeled wavespecs as sets of linear constraints. We now formulate the problems of consistency and satisfiability in terms of such constraints. In so doing, we digress somewhat from the examples under study, to develop the basic principles for testing the compatibility of device timings. We return to apply these principles to the CPU and RAM wavespecs in the following sections.

Definition 2. Let A be a set of constraints on a set T of n events. Let S_A be the solution space of A, namely, the set of n-tuples in the event space defined by T that satisfy A. We say that

A is *consistent* if and only if $S_A \ne \emptyset$.

Let B be a set of constraints on a set U of events, where $U \subseteq T$. Let $S_A|_U$ denote the projection of S_A on the set of events in U. Assume A and B are consistent; thus, $S_A|_U \ne \emptyset$ and $S_B \ne \emptyset$. We say that

A *satisfies* B, written $A \Rightarrow_s B$, if and only if $S_A|_U \subseteq S_B$. ■

Thus, by definition, to solve consistency and satisfiability, we need to characterize the solution spaces of sets of linear constraints. It is well known that such systems can be solved with optimization techniques; the reader is referred to

Papadimitriou and Steiglitz [4] for greater detail. In the next section we show how consistency and satisfiability can be formulated as linear programming (LP) problems. Following this, we argue that the restricted nature of the problems also permits a more efficient combinatorial approach.

4.1. LP Formulation of Consistency and Satisfiability

An LP problem involves optimizing the value of a linear objective function subject to a system of linear constraints. Let x be a vector of n variables. Let f_x be a linear function on x, and let L_x be a system of linear constraints on x. The general form of an LP problem is posed as follows:

maximize f_x

subject to L_x.

From a geometric perspective, each constraint in L_x defines an n-dimensional half-space. The intersection of half-spaces defines a (possibly empty or unbounded) convex polyhedron consisting of all solutions to L_x. An optimal solution is one that maximizes the value of the objective function; it can be shown that an optimal solution must lie on the boundary of the polyhedron. The objective function corresponds to a family of parallel hyperplanes in n-space. To search for an optimal solution, the hyperplane is moved in a maximizing direction across the polyhedron; the boundary where it exits represents optimality. If no solution exists, the LP problem is said to be *infeasible*; if the problem has solutions that permit the objective function to be made arbitrarily large, then it is said to be *unbounded*. The fundamental theorem of linear programming states that every LP problem that is neither infeasible nor unbounded has an (attainable) optimal solution.

The standard algorithm for solving LP problems is called the *Simplex method*. We will not go into a description of this method. For our purposes, it suffices to assume the existence of a function, **maximize**, defined as follows:

$$\textbf{maximize}(f_x,L_x) = \begin{cases} \phi, & \text{if infeasible} \\ \infty, & \text{if } f_x \text{ is unbounded} \\ f_{\hat{x}}, & \text{if } \hat{x} \text{ is an optimal solution.} \end{cases}$$

Consider now the set A of constraints on $T = \{t_1, \ldots, t_n\}$. By Definition 2, A is consistent if $S_A \neq \emptyset$. Observe that the nature of the objective function in an LP problem is irrelevant if the problem is infeasible, since feasibility is a characteristic of the sytem of constraints. In particular, consider the *zero* objective function:

$$f_t^0 = 0 \cdot t_1 + \cdots + 0 \cdot t_n.$$

It is evident that $f_s^0 = 0$ for every feasible solution $s = (s_1, \ldots, s_n)$. From the definition of **maximize**, it then follows that

$$S_A \neq \emptyset \Leftrightarrow \textbf{maximize}(f_t^0, A) = 0.$$

We now consider the satisfiability problem. As before, let B be a set of constraints on a set U of events. Without loss of generality, assume U consists of the first k elements of T. Assume A and B are consistent. By Definition 2, A satisfies B if $S_A|_U \subseteq S_B$.

Let $s = (s_1, \ldots, s_n)$ be a solution in S_A, and let $s_{\leq k}$ denote the first k components of s, namely,

$$s_{\leq k} = (s_1 \ \ldots, s_k).$$

Clearly,

$$(A \Rightarrow_s B) \Leftrightarrow (s \in S_A \Rightarrow s_{\leq k} \in S_B). \tag{5}$$

Thus, to verify satisfiability, we need to verify the right-hand side of (5).

We now proceed to show that statement (5) above—involving the comparison of the solution spaces S_A and S_B of the sets of inequalities A and B—can be converted to an equivalent statement concerning a comparison of bounds of inequalities derived from A and B. In Section 3.3 we have transformed a given set of timing constraints on a set $T = \{t_1, \ldots, t_n\}$ of event variables to a set of inequalities of the form $t_j - t_i \leq a_{ij}$, where we may set $a_{ij} = \infty$ if no explicit bound for $t_j - t_i$ exists. Thus, we may assume that the first set of timing constraints is

$$A = \{t_j - t_i \leq a_{ij} | 1 \leq i, j \leq n\}. \tag{6}$$

A particular bound a_{ij} in A may not be "achievable," in the sense that other constraints in A may imply a tighter bound $a_{ij}^* < a_{ij}$ on $t_j - t_i$. To illustrate this, consider the following constraints for one period of the clock waveforms for the Z80B CPU:

$$c_2 - c_1 = 20, \quad c_3 - c_2 \geq 65, \quad c_4 - c_3 = 20,$$

$$c_5 - c_4 \geq 65, \quad \text{and} \quad c_5 - c_1 \geq 165.$$

The equations $c_2 - c_1 = 20$ and $c_4 - c_3 = 20$ represent a possible interpretation of the footnote on p. 24 of [12]. It is easily seen that the first four conditions imply $c_5 - c_1 \geq 20 + 65 + 20 + 65 = 170$. Under this interpretation of the Z80B specifications, the bound 165 is never achievable and should be replaced by the tighter bound 170.

In general, the maximally tightened bound for any $t_j - t_i$ of A can be conveniently found by invoking the **maximize** function. For $1 \leq i$, $j \leq n$, let

$a_{ij}^* = \textbf{maximize}(t_j - t_i, A)$, and let

$$A^* = \{t_j - t_i \leq a_{ij}^* \mid 1 \leq i, j \leq n\}. \tag{7}$$

We refer to A^* as the *tightening* of A, and we say that A^* is a *tight* set of inequalities. Obviously, A is consistent if and only if A^* is consistent, and the solution space S_{A^*} of A^* is the same as that of A.

Returning now to the question of satisfiability of B by A, we need to verify equivalently that A^* satisfies B, where A^* is given by (7) and B has the form

$$B = \{t_j - t_i \leq b_{ij} \mid (i,j) \in X_B\}, \tag{8}$$

where X_B is an indexing set for the event pairs constrained by B. Note that B need not be tight. We have the following result:

Proposition 1.

$$s \in S_A \Rightarrow s_{\leq k} \in S_B \Leftrightarrow a_{ij}^* \leq b_{ij} \textit{ for all } (i,j) \in X_B.$$

Proof. ($\Leftarrow$) Consider any $s \in S_A$. By definition of a_{ij}^*, we have $s_j - s_i \leq a_{ij}^*$ for all $(i,j) \in X_B$. Assume now that $a_{ij}^* \leq b_{ij}$ for all $(i,j) \in X_B$. Thus, $s_j - s_i \leq b_{ij}$ for all $(i,j) \in X_B$, and $s_{\leq k} \in S_B$, proving the claim. ($\Rightarrow$) Suppose that $s \in S_A \Rightarrow s_{\leq k} \in S_B$; then $s_j - s_i \leq b_{ij}$ for all $(i,j) \in X_B$. Assume now that $a_{ij}^* > b_{ij}$ for some $(i,j) \in X_B$. By the definition of a_{ij}^*, there exists an $s \in S_A$ such that $s_j - s_i = a_{ij}^*$. Thus, $s_j - s_i > b_{ij}$, contradicting the first assumption above. Hence, $s \in S_A \Rightarrow s_{\leq k} \in S_B$ implies $a_{ij}^* \leq b_{ij}$ for all $(i,j) \in X_B$. ■

Note that one could use an "intermediate" set of inequalities between A and A^*, where only those $t_j - t_i$ are maximized for which the corresponding (i,j) is in X_B. Thus, by Proposition 1, we can verify satisfiability with $|X_B|$ maximizations, where $|X_B|$ is the number of constraints in B. However, the worst-case complexity of each optimization using the simplex method is exponential. Although there are polynomially bounded, general-purpose LP algorithms to perform the optimizations, we now show that our problems can be solved more quickly, both empirically and theoretically, than can general LP problems.

4.2. A Combinatorial Formulation

In the previous section, given the set A of constraints on T, each call to **maximize** represents an LP problem of the form

$$\begin{aligned} &\text{maximize } t_j - t_i \\ &\text{subject to } A. \end{aligned} \tag{9}$$

We note that both the objective function and the constraints are defined in

maximize $t_j - t_i$	find minimum-length $p(i, j)$
infeasible	negative cycle
unbounded	vertices i and j unconnected
optimal solution	minimum-length $p(i, j)$

FIG. 8. Correspondence between the LP and shortest path problems.

terms of pair-differences. It is well known that an LP problem with this restricted nature can be formulated in network terms as a shortest path problem; in particular, (9) is the dual of a restricted form of the *minimum-cost flow problem*, which is interpreted in network terms as finding a path of minimum length between two vertices [4]. It suffices to give a brief description of the shortest path formulation of (9).

Let $G_A = (V,E)$ be a directed graph, with vertices $V = \{1, \ldots, n\}$ and edges $E = \{e_{ij} | t_j - t_i \leq a_{ij} \in A\}$ where, associated with each edge e_{ij}, oriented from vertex i to vertex j, is an integral length a_{ij}. We also use (i,j) to refer to edge e_{ij}. A *path*, $p(i,j)$, between vertices i and j in G, is a sequence of edges $p(i,j) = (i,i_1), (i_1,i_2), \ldots, (i_k,j)$. A *cycle* in G is a path $p(i,i)$; vertex i is *connected to* vertex j if there is a path $p(i,j)$. The *length* of a path is the sum of its edge lengths. A *negative cycle* is a cycle $p(i,i)$ with length < 0.

The relationship between the outcomes of LP problem (9) and the shortest path problem is shown in Figure 8.

A convenient procedure for computing shortest paths is the Floyd–Warshall method [8], which computes shortest paths between *all* pairs of n vertices of a network, or detects a negative cycle, in time $O(n^3)$. Initially, each edge e_{ij} is assigned a length a_{ij} if $t_j - t_i \leq a_{ij}$ is in A, 0 if $i = j$, and ∞ otherwise. Let l_{ij} be the length of the shortest path from i to j, given a limited set of intermediate vertices. The algorithm consists of repeating the following step for every vertex i in the network:

If $l_{ii} < 0$, the network contains a negative cycle. Otherwise, for each pair (j,k), j, $k = 1, \ldots, n$, if $l_{jk} > l_{ji} + l_{ik}$, then $l_{jk} \leftarrow l_{ji} + l_{ik}$.

Let G_A^* denote the final graph obtained from G_A by the foregoing algorithm, and let l_{ij}^* be the length of the shortest path from i to j in G_A^*. We then have the following proposition:

Proposition 2. The set A of inequalities is inconsistent if G_A^* contains a negative cycle. Otherwise, for each (i,j) such that $1 \leq i, j \leq n$, $l_{ij}^* = a_{ij}^*$. In other words G_A^* corresponds precisely to the tightening A^* of A. ■

5. DEVICE TIMING AND CONNECTIVITY

In this section, we use the methods described above to test the accuracy of waveform specifications and to verify that the required timings of a device are satisfied by the produced timings of device connected to it. We use the CPU and RAM examples to illustrate our arguments.

Example 1. Consider the CPU constraints of Section 3.2. Let $C = \{c_1, \ldots, c_{16}\}$ be the set of clock events and let $U = \{a_1, a_2, w_1, w_2, d_1, d_2\}$ be the set of output events. Then, $T = C \cup U$ is the set of CPU events. Let A be the set of inequalities resulting from $R^{\text{cpu}}_{\text{seq}}, P^{\text{cpu}}_{\text{seq}}, R^{\text{cpu}}_{\text{tim}}$, and $P^{\text{cpu}}_{\text{tim}}$ after conversion to the form $t_j - t_i \leq a_{ij}$. Let A^* be the tightening of A.

The designer of the clock for the CPU is interested only in those inequalities of A^* that concern the clock events. Let $A^*_C = \{c_j - c_i \leq a^*_{ij} \mid 1 \leq i,j \leq 16\}$ be the set of inequalities. Part of A^*_C is shown in Figure 9; since each clock cycle has the same timings, we show the inequalities for the first cycle only. Entry (x,y) in location (i,j) of the table is to be interepreted as $x \leq t_j - t_i \leq y$. Table entries are shown only for $i \leq j$; by (4), the entry for location (j,i) is $(-y,-x)$. The reader will verify that all the original clock bounds are achievable. These bounds are indicated by "†." ■

Example 2. From the point of view of the RAM, only those inequalities of A^* that concern output events are of interest. Let $A^*_U = \{u_j - u_i \leq a^*_{ij} \mid u_i, u_j \in U\}$. The inequalities A^*_U are shown in Figure 10(a).

Note that the CPU timing parameter tDW: $w_1 - d_1 \geq 25$ is tightened to $w_1 - d_1 \geq 35$; thus, the original bound is not achievable. However, the tightening is likely due to our abstracting the WRITE-pulse falling transition as instantaneous, in the absence of a specified transition timing, rather than to a lack of precision in the specification. The discrepancy is resolved if we assume a transition time of 10 ns.

The tightening B^* of the inequalities in $B = R^{\text{ram}}$ of Section 3.3 is shown in Figure 10(b). Here, it turns out that all the original bounds of B arising from timing constraints are achievable. These RAM bounds are indicated by "†."

Comparing the † entries of Figure 10(b) with the corresponding entries of Figure 10(a), we see that the CPU satisfies the RAM requirements, except for $w_2 - a_2$. The RAM requires this to be less than 0, but the tightened CPU bound is 5. Here, also, the minor discrepancy may well be due to our simplifying

A^*_C	c_1	c_2	c_3	c_4	c_5
c_1	0,0	0,20†	65,∞	65,∞	165†,∞
c_2	—	0,0	65†,∞	65,∞	145,∞
c_3	—	—	0,0	0,20	65,2020
c_4	—	—	—	0,0	65†,2000†
c_5	—	—	—	—	0,0

FIG. 9. Inequalities A^*_C.

	A_U^*	a_1	a_2	w_1	w_2	d_1	d_2
	a_1	0,0	405,∞	140,∞	305,∞	-25,∞	405,∞
	a_2	—	0,0	-∞,-160	-2110,5	-∞,-265	-90,80
(a)	w_1	—	—	0,0	135,∞	-∞,-35	165,∞
	w_2	—	—	—	0,0	-∞,-200	30,2100
	d_1	—	—	—	—	0,0	265,∞
	d_2	—	—	—	—	—	0,0

	B^*	a_1	a_2	w_1	w_2	d_1	d_2
	a_1	0,0	90†,∞	20†,∞	80,∞	-∞,∞	80,∞
	a_2	—	0,0	-∞,-60	-∞,0†	-∞,-35	-∞,∞
(b)	w_1	—	—	0,0	60†,∞	-∞,∞	60,∞
	w_2	—	—	—	0,0	-∞,-35†	0†,∞
	d_1	—	—	—	—	0,0	35,∞
	d_2	—	—	—	—	—	0,0

FIG. 10. (a) Inequalities A_U^*. (b) Inequalities B^*.

assumption, in the absence of a specified timing, the WRITE-pulse transitions are instantaneous. ■

Example 3. In this example, we show how to handle delays in the linear inequalities model. Consider the CPU/RAM connection of Figure 11. Bus transceivers (MC6880A [3]) on the address and data lines strengthen the signals when connections are made to several memory devices; we assume that the transceiver delays have fixed values δ_a and δ_d, where $0 \le \delta_a, \delta_d \le 14$ ns.

Let the events occurring on the output of the top transceiver be $\bar{a}_1$ and $\bar{a}_2$, corresponding to CPU output events a_1 and a_2. Similarly, define events $\bar{d}_1$ and $\bar{d}_2$, corresponding to d_1 and d_2. It is clear that $\bar{a}_2 - \bar{a}_1 = (a_2 + \delta_a) - (\bar{a}_1 + \delta_a) = a_2 - a_1$. Similarly, $\bar{d}_2 - \bar{d}_1 = d_2 - d_1$. Next, consider $\bar{d}_2$ and $\bar{a}_2$ as an example

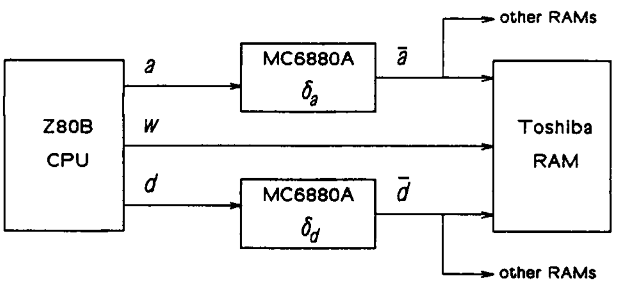

FIG. 11. The CPU/RAM connection with bus delays.

	$\bar{a}_1$	$\bar{a}_2$	w_1	w_2	$\bar{d}_1$	$\bar{d}_2$
$\bar{a}_1$	0,0	405,∞	126,∞	291,∞	-39,∞	391,∞
$\bar{a}_2$	—	0,0	-∞,-160	-2124,5	-∞,-251	-104,94
w_1	—	—	0,0	135,∞	-∞,-21	165,∞
w_2	—	—	—	0,0	-∞,-186	30,2086
$\bar{d}_1$	—	—	—	—	0,0	265,∞
$\bar{d}_2$	—	—	—	—	—	0,0

FIG. 12. The effects of delays.

of two events delayed by possibly different amounts δ_a and δ_d, respectively. Here, $\bar{d}_2 - \bar{a}_2 = (d_2 + \delta_d) - (a_2 + \delta_a) = (d_2 - a_2) + (\delta_d - \delta_a)$. Since $\delta_d - \delta_a$ could be as small as $0 - 14 = -14$, we can only conclude that $\bar{d}_2 - \bar{a}_2 \geq -90 - 14 = -104$. Similarly, $\bar{d}_2 - \bar{a}_2 \leq 80 + 14 = 94$. Finally, consider a difference like $\bar{d}_2 - w_2$, where $\bar{d}_2$ has been delayed, but w_2 has not. We have $\bar{d}_2 - w_2 = d_2 - w_2 + \delta_d$. Hence $\bar{d}_2 - w_2 \geq 30$ as before, but $\bar{d}_2 - w_2 \leq 2100 + 14 = 2114$.

The bounds on the remaining differences are computed in a similar fashion. The resulting modified bounds are shown in Figure 12. By comparing the "†" entries in Figure 10(b) with the corresponding entries in Figure 12, the reader will verify that the outputs $\bar{a}$, w, and $\bar{d}$ of the CPU and the two transceivers still satisfy the RAM requirements, except for the small discrepancy in the upper bound of $w_2 - \bar{a}_2$, which was already discussed in Example 2. ■

6. CONCLUSIONS

This paper presented two problems related to the use of waveform timing specifications. We began by showing how the timing specification of a device could be formulated in terms of linear inequalities. We then used optimization techniques to verify that a specification was consistent and to test whether the produced timings of one device satisfied the required timings of another; we showed that the restricted nature of the problems permits efficient solutions using well-known network techniques. To illustrate the arguments, we tested the accuracy of the write-cycle timings of a typical CPU and a RAM device.

The methods we described are applicable to waveform specifications that can be formulated as conjunctions of linear inequalities. These methods need to be generalized to deal with cases where the timing of an output event depends on several inputs. As an esample of such a case, consider a RAM read-cycle; typically, the timing of valid output data is specified with reference to both address the output-enable signals and cannot be formulated solely as a conjunction of inequalities. The extension of the methods to such cases is currently under investigation.

We appreciate the helpful comments of W. R. Pulleyblank, of the Department of Combinatorics and Optimization at the University of Waterloo, and J. K. Pachl, of IBM

Research, Zurich. We are also grateful to the referees for drawing our attention to a number of references on related work.

REFERENCES

[1] G. Borriello, A new interface specification methodology and its application to transducer synthesis. Ph.D. Thesis, Computer Science Division, EECS, University of California, Berkeley (May, 1988).
[2] R. B. Hitchcock, Timing verification and the timing analysis program. *19th Design Automation Conference, IEEE* (1982) 594–604.
[3] Motorola, Inc., *8-Bit Microprocessor and Peripheral Data* (1983).
[4] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, NJ (1982).
[5] P. Rony, Interfacing fundamentals: Timing diagram conventions. *Comput. Design* **January** (1980) 152–153.
[6] A. Ruehli and D. L. Ostapko, VLSI circuit analysis, timing verification and optimization," *VLSI CAD Tools and Applications* (W. Fichter and M. Morf, Eds.), Kluwer Boston (1987) 129–146.
[7] J. Springer, Making sense out of delay specs in semiconductor memories. *Electronics* **October** (1971) 82–88.
[8] R. E. Tarjan, *Data Structures and Network Algorithms*. SIAM, Philadelphia (1983).
[9] Toshiba, Inc., *MOS Memory Products Databook 82–83* (1984).
[10] S. H. Unger and C. J. Tan, Clocking schemes for high speed digital systems. *IEEE Trans. Comput.* **C-35** (1986) 880–895.
[11] C. Wiatrowski and C. House, *Logic Circuits and Microcomputer Systems*. McGraw-Hill, New York (1980).
[12] Zilog, Inc., *1982/1983 Data Book* (1983).

Received June 1988
Accepted March 1990