Dynamic Modeling of Logic Gate Circuits

Karem A. Sakallah

CSE-TR-253-95

July 31, 1995

THE UNIVERSITY OF MICHIGAN

Computer Science and Engineering Division
Department of Electrical Engineering and Computer Science
Ann Arbor, Michigan 48109-2122
USA

Dynamic Modeling of Logic Gate Circuits

Karem A. Sakallah

Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan 48109-2122

July 31, 1995

Abstract

We propose a dynamic model of logic gate circuits that integrates their functional and temporal aspects in a single unified framework based on a calculus of symbolic logic waveforms. Using a continuous time model, appropriate time derivatives that capture the dynamic behavior of digital signals are defined. These derivatives provide a natural link between the functional and timing components of gate models and highlight the conditional nature of gate delay. The framework subsumes all known dynamic models of logic gates and—through a formal process of functional and temporal abstraction—provides a sound foundation for functional timing analysis.

1 Introduction

Most dynamic models of logic gate circuits are premised on the separability of the circuits' functional and temporal behaviors. Indeed, the temporal behavior is almost always posited as an "add-on" to the Boolean algebra-based functional behavior in the form of lumped propagation delays. In reality, these two aspects of behavior are not entirely separable. Manifestations of the link between functionality and timing abound and have led to a proliferation of ad hoc extensions to Boolean algebra that were aimed at improving the temporal accuracy of the models. The first concession to the strict separation between timing and function was to allow for different rise/fall gate delays. This immediately led to the consideration of inertial filtering mechanisms to block the transmission of the narrow pulses that are produced as a result of asymmetric delays. The recently proposed state-dependent delay models [2, 4] intertwine functionality and timing even more closely by distinguishing among the various input conditions that produce functionally similar but temporally different output responses. The advent of "deep submicron" integrated circuit technology has introduced further complications including the need to account for the finite slopes of switching signals and for the degree of simultaneity among temporally adjacent transitions on different gate inputs. The increasing dominance of interconnect delay is also challenging the standard assumptions about the causes of delay as well as its intrinsic nature.

It is rather remarkable that—unlike the solid foundation that supports functional modeling—no universally accepted theoretical foundation exists for modeling the timing behavior of digital circuits. The plethora of models that exist today evolved in response to the need, real or perceived, for greater accuracy and tend to be specific to particular technologies or design styles. The primary goal of this paper is to take a first step towards evolving a consistent theory for the dynamic behavior of digital circuits. The centerpiece of this theory is a differential calculus that allows us to describe the transient behavior of logic waveforms in terms of appropriate time derivatives based on a continuous time model. These derivatives form the link between the functional and timing aspects of waveform behavior by establishing a functional framework for temporal modeling. The framework highlights the intrinsically conditional nature of signal delay and can be shown to encompass most delay models proposed to date.

The rest of this paper is organized in five sections. In Section 2 we briefly cover some basics to help enhance the clarity of the development in later sections. In particular, we describe our system of notation and introduce the topics of conditional arithmetic and partially specified Boolean functions. In Section 3 we propose a model for symbolic logic waveforms, investigate a number of its properties and nuances, and provide the computational recipes required for its algebraic manipulation. In Section 4 we define the time derivatives of these symbolic logic waveforms and relate them to several dynamic aspects of the waveform behavior; the development is restricted to completely specified waveforms because of space limitations. Signal delay through logic functions is defined in Section 5 and related to the time derivatives. Additional aspects of delay modeling, such as propagation through logic functions and inertial filtering are also discussed. The paper concludes in Section 6 by hinting at the application of this model and its underlying theory to the field of timing analysis.

2 Preliminaries

We will generally denote scalar quantities using lower-case symbols. Aggregates (vectors and sets) will be either denoted using upper-case roman or lower-case bold type. Calligraphic type will denote the carriers of algebraic structures: $\mathcal{B}$ for Boolean algebra, $\mathcal{R}$ for the real number system, $\mathcal{S}$ for the set of intervals on $\mathcal{B}$, and $\mathcal{U}_n$ for the universe of n-variable Boolean functions. Unless explicitly stated otherwise, when we speak of Boolean variables and functions we mean variables and functions in the 2-element Boolean (switching) algebra. Thus, $x_1, x_2, \ldots, x_n$ refer to switching variables, f, g, h denote switching functions, $X = (x_1, x_2, \ldots, x_n)$ represents a vector of switching variables, and F, G, H denote sets of switching functions.

The elements of S will be equivalently denoted as $\{[0, 0], [0, 1], [1, 1]\}$ or as $\{0, U, 1\}$. Logic waveforms will be indicated by lower-case bold type such as $\boldsymbol{w}$, $\boldsymbol{x}_i$, $\boldsymbol{y}_j$ or $\boldsymbol{z}$. An m-input single-output logic gate with input waveforms $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m$ and output waveform $\boldsymbol{z}$ is characterized by its combinational switching function f and an appropriate set of conditional propagation delays $\Delta_{\boldsymbol{y}_1\boldsymbol{z}}, \Delta_{\boldsymbol{y}_2\boldsymbol{z}}, \ldots, \Delta_{\boldsymbol{y}_m\boldsymbol{z}}$. The *instantaneous* (delayless) output waveform for such a gate will be denoted by $\boldsymbol{z}^*$.

To distinguish between Boolean and arithmetic operators, we will reserve the "+" symbol for arithmetic addition and the "$\vee$" symbol for logical OR. Logical AND will be denoted by " $\wedge$ " or by juxtaposition.

2.1 Conditional Arithmetic

Conditional arithmetic is a hybrid algebraic system that combines the deductive power of switching algebra with the computational facilities of regular arithmetic and emerges as a natural vehicle for capturing the functional dependencies of gate delay and signal transition times.

A *conditional number* A is a mapping (function) that assigns to every element $X \in \mathcal{B}^n$ an element $A(X) \in \mathcal{R}$. Conditional numbers can be specified by function tables that are analogous to the truth tables of Boolean functions or as sets of tuples that associate a real number with each possible combination of the Boolean vector X. Algebraically, a conditional number can be expressed in the following *disjoint sum-of-products (DSOP)* form:

$$A(x_1, x_2, \ldots, x_n) = (\varphi_1(X) \wedge a_1) \oplus (\varphi_2(X) \wedge a_2) \oplus \ldots \oplus (\varphi_k(X) \wedge a_k) \equiv \bigoplus_{i=1}^{k} (\varphi_i(X) \wedge a_i) \tag{2.1}$$

where $\{\varphi_1(X), \ldots, \varphi_k(X)\}$ is an orthonormal set of n-variable Boolean functions [1] and $a_1, \ldots, a_k \in \mathcal{R}$ are the corresponding real values of A. The set $\{\varphi_i(X) \,|\, 1 \le i \le k\}$ will be referred to as the Boolean *basis* of A. The logical AND and XOR operators in (2.1) are extended to handle the reals according to the rules:

$$y \wedge u \equiv \begin{cases} 0 \text{ if } y = 0 \\ u \text{ if } y = 1 \end{cases} \qquad u \oplus v \equiv \begin{cases} 0 & \text{if } u = 0 \text{ and } v = 0 \\ u & \text{if } u \neq 0 \text{ and } v = 0 \\ v & \text{if } u = 0 \text{ and } v \neq 0 \\ \text{undefined} & \text{if } u \neq 0 \text{ and } v \neq 0 \end{cases} \tag{2.2}$$

where $y \in \mathcal{B}$ and $u, v \in \mathcal{R}$. A DSOP form for conditional number A is *minimal* if each a_i is a distinct real number. The minimal DSOP form of A is unique and can be obtained from any DSOP form of A by repeated application of the identity:

$$(\varphi_i(X) \wedge a_i) \oplus (\varphi_j(X) \wedge a_j) = (\varphi_i(X) \oplus \varphi_j(X)) \wedge a_i = (\varphi_i(X) \vee \varphi_j(X)) \wedge a_i \tag{2.3}$$

whenever $a_j = a_i$.

To avoid potential ambiguities in interpreting algebraic expressions of conditional numbers, the elements of the Boolean basis and the corresponding real-valued components will be distinguished by enclosing the latter in angle brackets. This convention leads to the following *standard* DSOP form of (2.1):

$$A(x_1, x_2, \ldots, x_n) = \varphi_1(X)\langle a_1\rangle \oplus \varphi_2(X)\langle a_2\rangle \oplus \ldots \oplus \varphi_k(X)\langle a_k\rangle \equiv \bigoplus_{i=1}^{k} \varphi_i(X)\langle a_i\rangle \tag{2.4}$$

where, for the sake of compactness, juxtaposition is used to indicate logical AND.

Let $A(X) = \bigoplus_{i=1}^{k} \varphi_i(X)\langle a_i\rangle$ and $B(X) = \bigoplus_{i=1}^{m} \psi_i(X)\langle b_i\rangle$ be two conditional numbers expressed in terms of two independent bases $\{\varphi_1(X), \ldots, \varphi_k(X)\}$, $\{\psi_1(X), \ldots, \psi_m(X)\}$ and let $*$ stand for any binary arithmetic operation (addition, subtraction, multiplication, division, maximization, etc.) The conditional number $C(X) = A(X) * B(X)$ is defined by the following DSOP expression:

$$C(X) = \bigoplus_{i=1}^{k} \bigoplus_{j=1}^{m} \varphi_i(X)\psi_j(X)\langle a_i * b_j\rangle \tag{2.5}$$

Equation (2.5) also applies when $*$ is a relational operator ($<$, $=$, $>$, $\geq$ etc.) yielding a logical predicate $P(X): \mathcal{B}^n \to \mathcal{B}$. Since the terms in the above XOR double summation are orthogonal, the following simpler form of such predicates (using inclusive ORs) is immediately available:

$$P(X) = A(X) * B(X) = \bigvee_{i=1}^{k} \bigvee_{j=1}^{m} \varphi_i(X)\psi_j(X)(a_i * b_j) \tag{2.6}$$

2.2 Partially Specified Boolean Functions

Partial specification of Boolean functions provides a convenient mechanism for modeling functional uncertainty in logic waveforms. Uncertainty will be seen as an inherent ingredient in the logic waveform abstraction we introduce in Section 3. In addition, the judicious introduction of uncertainty often leads to significant savings in the computations required for performing dynamic analyses of logic circuits with only minor reductions in accuracy.

Generally speaking, a *partially specified n-variable Boolean function* $F(x_1, x_2, \ldots, x_n)$ is a nonempty subset of the function space $\mathcal{U}_n(x_1, x_2, \ldots, x_n)$ representing all 2^{2^n} completely specified *n*-variable Boolean functions. A partially specified function $F(X)$ is a *function interval* [1] if it can be expressed as

$$F(X) = \{ f(X): \mathcal{B}^n \to \mathcal{B} \,|\, l(X) \leq f(X) \leq u(X)\} \tag{2.7}$$

where $l(X)$ and $u(X)$ are two *n*-variable Boolean functions such that $l(X) \leq u(X)$. A function interval $F(X)$ can be viewed as a mapping $\mathcal{B}^n \to S$, where S is the set of intervals on $\mathcal{B}$, and can thus be equivalently expressed as $F(X) = [l(X), u(X)]$. It is easy to show that $l(X)$ and $u(X)$ are, respectively, the *greatest lower bound* (glb) and *least upper bound* (lub) of the functions contained in $F(X)$. Denoting these bounds by $\lfloor F(X) \rfloor$ and $\lceil F(X) \rceil$ allows us to express $l(X)$ and $u(X)$ in terms of $F(X)$:

$$\begin{aligned} l(X) &= \lfloor F(X) \rfloor = \bigwedge_{f(X) \in F(X)} f(X) \\ u(X) &= \lceil F(X) \rceil = \bigvee_{f(X) \in F(X)} f(X) \end{aligned} \tag{2.8}$$

The *interval operator* [ ] is a unary operator that returns the smallest function interval containing a given partially specified function. It is defined by the formula:

$$[F(X)] \equiv [\lfloor F(X) \rfloor, \lceil F(X) \rceil] \tag{2.9}$$

When $F(X)$ is itself a function interval, $[F(X)] = F(X)$.

The main advantage of function intervals over unrestricted partially specified functions is that they do not require an explicit enumeration of their member functions. Instead, the elements of a function interval are implicitly defined by the interval's glb and lub along with the partial ordering relation "$\leq$". This advantage may not, in general, be preserved when function intervals are combined by set operators. Specifically, the union, intersection, and difference of two function intervals are not necessarily function intervals. On the other hand, applying Boolean operators to function intervals will always yield function intervals. This property leads to an *interval Boolean algebra* that can be used to manipulate and simplify expressions involving function intervals. The extension of the three basic Boolean operators NOT, AND, and OR to function intervals is provided by the following theorem.

Theorem 2.1 Boolean Operations on Function Intervals) *Let $F(X)$ and $G(X)$ be two function intervals. Then $F'(X)$, $F(X) \wedge G(X)$, and $F(X) \vee G(X)$ are also function intervals that are given by the following identities:*

$$F'(X) = [\lceil F \rceil', \lfloor F \rfloor']$$

$$F(X) \wedge G(X) = [\lfloor F \rfloor \wedge \lfloor G \rfloor, \lceil F \rceil \wedge \lceil G \rceil]$$

$$F(X) \vee G(X) = [\lfloor F \rfloor \vee \lfloor G \rfloor, \lceil F \rceil \vee \lceil G \rceil]$$ □

Equality of function intervals derives from their interpretation as function sets. Thus, the function intervals $F(X)$ and $G(X)$ are equal if and only if $F(X) \subseteq G(X)$ and $G(X) \subseteq F(X)$. Equivalently:

$$(F = G) \Leftrightarrow (\lfloor F \rfloor = \lfloor G \rfloor) \wedge (\lceil F \rceil = \lceil G \rceil) \tag{2.10}$$

i.e., $F(X)$ and $G(X)$ are identical if their respective bounds are equal.

As mentioned earlier, set operations on function intervals do not, in general, yield function intervals. In such cases, the interval operator "[ ]" can be used to convert the result of a set operation to the smallest enclosing function interval. The function interval containing the union of two function sets (not necessarily function intervals) is particularly useful and will be called the *expansion* or *interval union* of the two function sets. Denoting this operation by the symbol "$\Diamond$", it is defined by the following expression:

$$F(X) \Diamond G(X) \equiv [F(X) \cup G(X)] \tag{2.11}$$

Equivalently, we may express the interval union in terms of logical AND and OR rather than set union. This alternative form, given in Theorem 2.2 below, is usually more convenient for algebraic manipulations.

Theorem 2.2 $F(X) \Diamond G(X) = [\lfloor F \rfloor \wedge \lfloor G \rfloor, \lceil F \rceil \vee \lceil G \rceil]$ □

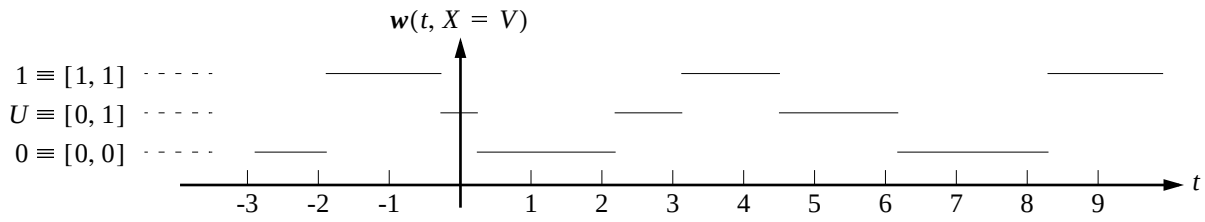


Figure 3-1: A typical plot of $w(t, X)$ versus t for $X = V$

3 Symbolic Logic Waveforms

We define a logic waveform as a discrete $(n + 1)$-variable function $\boldsymbol{w}(t, X)$ that takes values from the three-element set S. The function arguments represent the independent time variable $t \in \mathcal{R}$ and a suitable vector of n independent Boolean variables $X \in \mathcal{B}^n$. In other words, $\boldsymbol{w}$ is an element of the function space defined by the mapping:

$$\boldsymbol{w}: \mathcal{R} \times \mathcal{B}^n \rightarrow S \tag{3.1}$$

Several points are worth noting about this definition:

1. The domain of this function is partly discrete ($\mathcal{B}^n$) and partly continuous ($\mathcal{R}$). While the time axis could have been discretized to obtain a purely discrete waveform function, treating time as a continuous variable has certain advantages as will become obvious in the sequel. An immediate consequence of choosing a continuous time model is that, when X is set to some fixed value V, $\boldsymbol{w}(t, V)$ is a discrete (discontinuous) function of t (see Figure 3-1.) The role that t plays in the representation of $\boldsymbol{w}$ is, thus, limited to that of an interval *selector*. The waveform in Figure 3-1, for example, can be expressed as follows:

$$\begin{aligned}\boldsymbol{w}(t, X = V) = {} & (-\infty < t < -1.9) \wedge (0) \vee (-1.9 < t < -0.3) \wedge (1) \vee (-0.3 < t < 0.2) \wedge (U) \vee \\ & (0.2 < t < 2.2) \wedge (0) \vee (2.2 < t < 3.1) \wedge (U) \vee (3.1 < t < 4.5) \wedge (1) \vee \\ & (4.5 < t < 6.2) \wedge (U) \vee (6.2 < t < 8.3) \wedge (0) \vee (8.3 < t < \infty) \wedge (1)\end{aligned}$$

   where each inequality involving t is considered to be a predicate that evaluates to 1 when t falls within the specified interval and that evaluates to 0 otherwise.

2. The discontinuous nature of a logic waveform allows us to characterize its temporal behavior at any given time instant as either *stable* or *changing*. The waveform in Figure 3-1, for example, is stable at $t = 4$ and changing at $t = 6.2$. The times at which the waveform is changing are called *transition* instants; the waveform is also said to have an *event* at each of those times.

3. The codomain of the function is the ternary set S rather than the binary set $\mathcal{B}$. This choice is motivated by the need to model uncertainty in waveform value. To appreciate this need consider the question of assigning an appropriate value to a binary signal at a transition instant between logic 0 and logic 1. The available options are shown in Figure 3-2. Options (a) and (b) assign a binary value to the signal by including the transition instant with either the preceding or the succeeding time interval. These two options are somewhat arbitrary, however, since there is no clear basis for preferring one to the other. The third option basically excludes the transition instant from the function domain by declaring the signal value to be undefined (i.e. neither 0 nor 1.) This option is also unsatisfactory since it forces different waveforms to have different time domains; the resulting proliferation of time domains causes unwarranted complexity when manipulating collections of waveforms. The

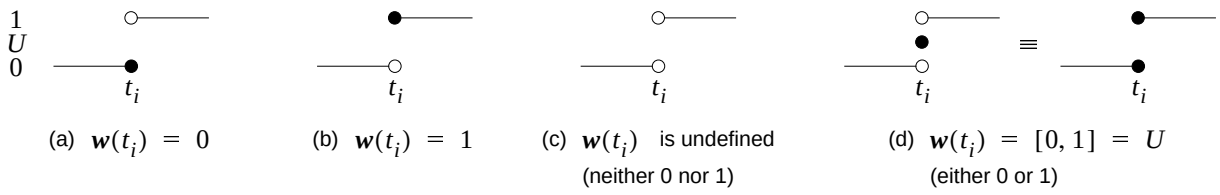


Figure 3-2: Options for waveform value at the transition instant t_i

last option assigns the value $[0, 1]$ or U to the waveform at the transition instant. Semantically, this choice conforms to the intuition that the signal value is "unknown" at the transition instant and can appropriately be interpreted as either 0 or 1. In other words, at the transition instant we lose the certainty of knowing the precise value of the waveform and must assume that it becomes multi-valued. It is important to note that the third value U does not model an intermediate voltage level between the low and high voltages that correspond to logic 0 and logic 1. Rather, it represents the set $\{0, 1\}$ and suggests that the transition instant is included in *both* the preceding and succeeding time intervals.

The ability to model uncertainty in waveform value is not restricted to transition instants between logic 0 and logic 1. For example, it may be desirable in some cases to ignore the precise value of the waveform over one or more finite time intervals. These intervals may correspond to the finite rise/fall times of the analog voltage waveform that is idealized by the logic waveform. In addition, it may be desirable to deliberately ignore the functional dependence of the waveform on one or more of its Boolean variables. It turns out that such *functional abstraction*, along with the complementary mechanism of temporal abstraction (see Section 3.2), form the basis that underlies the field of timing analysis.

4. The temporal and Boolean aspects of the waveform function are *separable* in the sense that the functional dependence on t and X are distinctly different [3, p. 23]. As noted earlier, when the value of X is fixed $\boldsymbol{w}$ becomes a ternary function of t. This can be expressed formally by the mapping

$$\boldsymbol{w}: \mathcal{B}^n \rightarrow (\mathcal{R} \rightarrow S) \tag{3.2}$$

which assigns a ternary function of time to each element $X \in \mathcal{B}^n$ and allows us to "separate" the arguments of $\boldsymbol{w}$ by expressing it as $\boldsymbol{w}(X)(t)$. In this form, the notation $\boldsymbol{w}(X)$ is interpreted to mean a function of time that assigns to each element in $\mathcal{R}$ a unique element from S. Alternatively, fixing the value of t makes $\boldsymbol{w}$ a partially specified Boolean function of X. Formally, this is expressed by the alternate mapping:

$$\boldsymbol{w}: \mathcal{R} \rightarrow (\mathcal{B}^n \rightarrow S) \tag{3.3}$$

which assigns a Boolean function interval to each element $t \in \mathcal{R}$ and allows us to express the waveform function as $\boldsymbol{w}(t)(X)$. The notation $\boldsymbol{w}(t)$ in this case is interpreted to mean a function interval in the space of Boolean functions of X. We will mostly use this second form and refer to $\boldsymbol{w}$ as an n-variable logic waveform[1] where n is the size of the Boolean vector X. In addition, for the sake of brevity, we may on occasion drop $\boldsymbol{w}$'s explicit dependence on either t or X or both; in such cases it must be assumed that this dependence is implied.

1. Strictly speaking, $\boldsymbol{w}$ is an $(n + 1)$-variable function. However, since t is a common argument to all logic waveforms, it is convenient to characterize specific waveforms by the number of their Boolean variables.

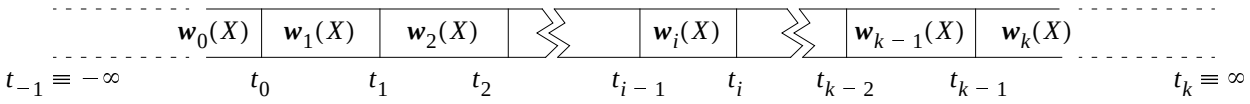


Figure 3-3: Schematic Representation of the Logic Waveform $w(t)(X)$

3.1 Logic Waveform Specification

A logic waveform $w(t)(X)$ can be represented in many different ways. A particularly convenient representation is to specify it in terms of a two-tuple (T_w, V_w) whose components are defined as follows (see Figure 3-3):

- A finite ordered set T_w of k distinct time instants $T_w = \{t_i \in \mathcal{R} | t_0 < t_1 < \ldots < t_{k-1}\}$ referred to as the *time grid* or *time base* of w; the elements of T_w are referred to as *grid points*. T_w induces a partition of the time axis

  $$\{(-\infty, t_0), [t_0, t_0], (t_0, t_1), [t_1, t_1], \ldots, (t_{k-2}, t_{k-1}), [t_{k-1}, t_{k-1}], (t_{k-1}, \infty)\} \tag{3.4}$$

  that consists of $k+1$ finite open time intervals and k time instants. We will find it convenient for later manipulations to augment T_w with two artificial grid points defined as $t_{-1} \equiv -\infty$ and $t_k \equiv \infty$.

- A corresponding set $V_w = \{w_0(X), w_1(X), \ldots, w_k(X)\}$ of $k+1$ partially specified Boolean functions. The ith such function $w_i(X): \mathcal{B}^n \rightarrow S$ is the *value* of w over the interval (t_{i-1}, t_i). We will refer to $w_0(X)$ and $w_k(X)$ as the *initial* and *final* values of w.

As suggested by Figure 3-2, a transition instant is best modeled by including it in both the preceding and succeeding time intervals. Generalizing, we define the value of w at grid point t_i as the interval union of w's value over the intervals (t_{i-1}, t_i) and (t_i, t_{i+1}). Specifically:

$$w(t_i)(X) \equiv w_i(X) \lozenge w_{i+1}(X) \qquad 0 \le i \le k-1 \tag{3.5}$$

A waveform w will be called *completely specified* if the $k+1$ Boolean functions in V_w are completely specified; if one or more of these functions is partially specified, w will be referred to as a *partially specified* waveform. When necessary, we will write $w = (T_w, V_w)$ to indicate that w is defined in terms of a particular time grid T_w and an associated value set V_w.

Stability, Functional Uncertainty, and Temporal Uncertainty The notion of stability is fundamental to our understanding of the dynamic behavior of logic waveforms. As mentioned earlier, at any given time instant a completely specified waveform is either stable or changing for any given combination of the Boolean vector X. On the other hand, when the waveform is partially specified, i.e. when the waveform has some *functional uncertainty*, its temporal behavior may become ambiguous.

Let us associate with a waveform w a ternary function $\sigma_w(t) \in S$, called its *stability predicate*. This predicate evaluates to 1 at a given time instant if it is certain that the waveform cannot have a transition at that instant under any combination of the Boolean vector X. The predicate evaluates to 0 if it is certain that there is at least one combination of the Boolean vector X that causes a transition to occur at that instant. It evaluates to U if we are uncertain about whether transitions are possible at that time. Stated another way, $\sigma_w(t) = 1$ means w is always stable at t; $\sigma_w(t) = 0$ means w will experience one or more tran-

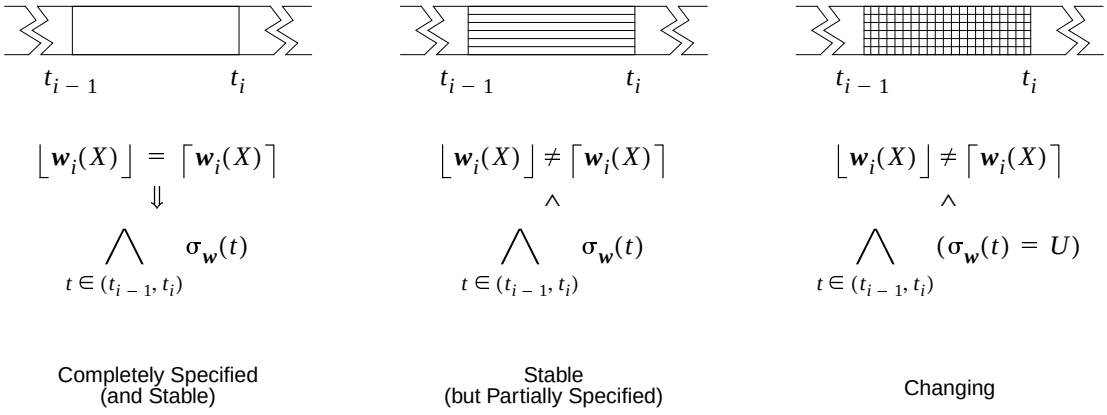


Figure 3-4: The three types of waveform intervals

sitions at t; and $\sigma_w(t) = U$ means we cannot tell whether w will or won't have transitions at t. We refer to this last condition as *temporal uncertainty.*

Consider now a completely specified waveform $w = (T_w, V_w)$. Clearly, $\sigma_w(t) = 1$ within the time intervals defined by T_w. On the other hand, $\sigma_w(t_i)$, where t_i is a grid point, may be either 0 or 1. A grid point at which $\sigma_w(t_i) = 1$ is *redundant* since it does not correspond to an actual signal transition.

Consider next a partially specified waveform $w = (T_w, V_w)$. At the grid points, we must assume that transitions can occur even if the partially specified functions over the adjoining intervals are identical. To understand this it is best to recall the interpretation of partially specified Boolean functions as *sets* of completely specified Boolean functions. If the two sets w_i and w_{i+1} are disjoint, then we can state with certainty that w will change at t_i; thus $\sigma_w(t_i) = 0$. If the two sets are not disjoint, we must conclude that it is possible for w to change as well as not to change; thus $\sigma_w(t_i) = U$. Thus the presence of uncertainty in w_i and/or w_{i+1} causes $\sigma_w(t_i)$ to be uncertain (except when the sets are disjoint.) In any case, $\sigma_w(t_i) \neq 1$. Within partially specified intervals, we have two cases. We either know that the waveform is stable, i.e. $\sigma_w(t) = 1$, or we don't know for sure that it is stable, i.e. $\sigma_w(t) = U$. In any case, $\sigma_w(t) \neq 0$ at an internal point since that would imply that we have certain knowledge that w will change at that time.

The preceding discussion suggests that a plausible definition of the stability predicate might be as follows:

$$\sigma_w(t) \equiv \forall X \cdot (\lfloor w(t)(X) \rfloor \odot \lceil w(t)(X) \rceil) \tag{3.6}$$

This definition yields the expected results when w is a completely specified waveform. For partially specified waveforms, the definition always yields 0 for those time instants at which $w(t)$ is partially specified. Thus, it fails to make the distinction noted above between intervals over which the waveform value is uncertain but it is known to be stable and those intervals over which both the waveform value and stability are uncertain.

In summary, we note that functional certainty implies temporal certainty but that the converse is not true. Thus, the stability predicate must be made part of the waveform specification for partially specified waveforms. In those cases, we will indicate a waveform by a three-tuple $w = (T_w, V_w, \sigma_w)$. Figure 3-4 illustrates the three types of waveform intervals that result from the interaction of functional and temporal uncertainty.

Algebraic Representation The functional dependence of waveform $\boldsymbol{w} = (T_{\boldsymbol{w}}, V_{\boldsymbol{w}}, \sigma_{\boldsymbol{w}})$ on its arguments t and X can be expressed in a variety of ways. A particularly useful algebraic form for $\boldsymbol{w}$ is the standard *sum-of-intervals* (SOI) expression given by:

$$\boldsymbol{w}(t)(X) = \left(\bigvee_{0 \le i \le k} (t_{i-1} < t < t_i) \wedge \boldsymbol{w}_i(X) \right) \vee \left(\bigvee_{0 \le i \le k-1} (t = t_i) \wedge (\boldsymbol{w}_i(X) \Diamond \boldsymbol{w}_{i+1}(X)) \right) \tag{3.7}$$

This expression can be viewed as an expansion of $\boldsymbol{w}$ using the partition of the time axis given by (3.4) as a basis. The expansion has $2k+1$ disjoint terms corresponding to the $k+1$ intervals and k grid points. Each such term is the conjunction of an interval or grid point selector (a predicate on t) and a corresponding waveform value (a partially specified Boolean function of X.) Since waveform values at the grid points are completely determined by their values over the adjoining intervals, it is often sufficient in algebraic manipulations to express a waveform just by the first OR sum in (3.7). In such cases, it is to be understood that the unspecified waveform values at the grid points can be obtained by applying (3.5).

3.2 Temporal Operations on Logic Waveforms

The time grid used to specify a logic waveform $\boldsymbol{w}$ is not unique. We examine in this section the effect of changing the time grid on the value and stability of the waveform.

Refinement Consider a waveform $\boldsymbol{w} = (T_{\boldsymbol{w}}, V_{\boldsymbol{w}}, \sigma_{\boldsymbol{w}})$, and let $T'_{\boldsymbol{w}}$ be another time grid such that $T_{\boldsymbol{w}} \subset T'_{\boldsymbol{w}}$. In other words, $T'_{\boldsymbol{w}}$ is obtained from $T_{\boldsymbol{w}}$ by adding more grid points. The effect of adding a grid point t_* such that $t_{i-1} < t_* < t_i$ is to *split* the interval (t_{i-1}, t_i) into two subintervals (t_{i-1}, t_*) and (t_*, t_i). Clearly, such a split can neither affect the value nor the stability of $\boldsymbol{w}$ at any instant within (t_{i-1}, t_i).

Abstraction Consider a waveform $\boldsymbol{w} = (T_{\boldsymbol{w}}, V_{\boldsymbol{w}}, \sigma_{\boldsymbol{w}})$, and let $T'_{\boldsymbol{w}}$ be another time grid such that $T'_{\boldsymbol{w}} \subset T_{\boldsymbol{w}}$. In other words, $T'_{\boldsymbol{w}}$ is obtained from $T_{\boldsymbol{w}}$ by removing one or more grid points. Removal of the grid point t_i from $T_{\boldsymbol{w}}$ causes the intervals (t_{i-1}, t_i) and (t_i, t_{i+1}) to *merge* into a single interval (t_{i-1}, t_{i+1}). This merger causes the value and stability of $\boldsymbol{w}$ for $t \in (t_{i-1}, t_{i+1})$ to change according to the following rules:

$$\begin{aligned} \boldsymbol{w}(t)(X) &= \boldsymbol{w}(t_i)(X) \\ \sigma_{\boldsymbol{w}}(t) &= \begin{cases} 1 & \text{if } \sigma_{\boldsymbol{w}}(t_i) = 1 \\ U & \text{if } \sigma_{\boldsymbol{w}}(t_i) \neq 1 \end{cases} \end{aligned} \tag{3.8}$$

In other words, unless the waveform was stable at t_i merging the two intervals adjoining at t_i causes the value of $\boldsymbol{w}$ at t_i to "spread" to all time instants within the merged interval and its stability to become uncertain. In general, temporal abstraction increases the uncertainty in both the functional and temporal aspects of the waveform. The only exception is when $\sigma_{\boldsymbol{w}}(t_i) = 1$ before the merger indicating that the grid point t_i does not correspond to an actual transition instant. Obviously, removal of such a "redundant" grid point has no effect on the value or stability of the waveform (see discussion of minimal waveforms below.)

Example 3.1 An example of temporal abstraction is shown in Figure 3-5. Removal of the grid points at $t = 5$ and $t = 11$ from the time grid of the completely specified waveform $\boldsymbol{w}$ leads to the creation of two merged intervals and results in a new abstracted waveform $\tilde{\boldsymbol{w}}$. Prior to the mergers, the value and stability predicate of $\boldsymbol{w}$ at these two grid points were:

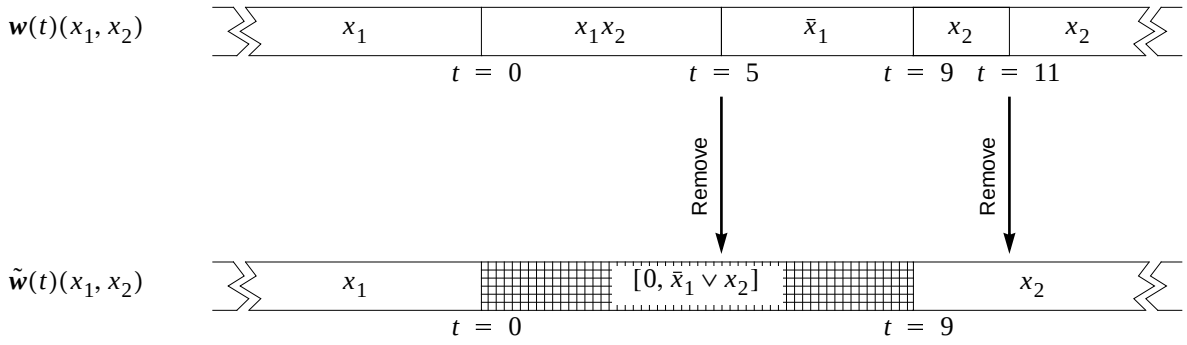


Figure 3-5: Temporal abstraction (see Example 3.1)

$$w(5)(x_1, x_2) = (x_1x_2) \Diamond (\bar{x}_1) = [0, \bar{x}_1 \vee x_2]$$

$$\sigma_w(5) = \forall x_1 \cdot \forall x_2 \cdot ((x_1x_2) \odot \bar{x}_1) = \forall x_1 \cdot \forall x_2 \cdot (x_1\bar{x}_2) = 0$$

$$w(11)(x_1, x_2) = (x_2) \Diamond (x_2) = [x_2, x_2] = x_2$$

$$\sigma_w(11) = \forall x_1 \cdot \forall x_2 \cdot (x_2 \odot x_2) = \forall x_1 \cdot \forall x_2 \cdot (1) = 1$$

Thus, the abstraction of $t = 5$ causes $\tilde{w}$ to be partially specified and changing over the merged interval $(0, 9)$. In contrast, the abstraction of $t = 11$ has no effect since w is always stable at that time. Note that the removal of the grid point at $t = 5$ represents an irreversible loss of information; starting from $\tilde{w}$ it is not possible to recover the original waveform w by merely adding back the grid point at $t = 5$. □

Minimal Waveforms A grid point whose temporal abstraction does not change the value or the stability of the waveform is *redundant* and can be removed to yield a more compact representation of the waveform. A waveform is minimal if it is based on a minimal time grid, i.e. a grid that does not have any redundant grid points.

For a completely specified waveform, a grid point is redundant if its stability predicate is 1. A completely specified waveform is therefore minimal if the stability predicate is 0 at all grid points. Any given completely specified waveform can be minimized by temporally abstracting all grid points at which the stability predicate is 1.

For partially specified waveforms, a grid point is redundant if the two time intervals adjoining it are changing intervals whose partially specified Boolean functions are equal. Specifically, the grid point t_i is redundant if (t_{i-1}, t_i) and (t_i, t_{i+1}) are changing intervals such that $w_i(X) = w_{i+1}(X)$.

3.3 Logical Operations on Waveforms

The complement of a waveform w is simply obtained by complementing each of the components in its value set V_w. Let w and y be arbitrary waveforms and let f be a binary operator defined over partially specified Boolean functions. The application of f to w and y yields another waveform z that can be obtained by the following procedure:

Re-express $\boldsymbol{w}$ and $\boldsymbol{y}$ in terms of the common time grid $T_{\boldsymbol{wy}} = T_{\boldsymbol{w}} \cup T_{\boldsymbol{y}}$.

- Apply f to the transformed waveforms interval-by-interval. Note that this may cause the stability predicate at certain grid points to change from 0 to 1 (e.g. a 0 on one input of an AND can eliminate a possible transition on another input).
- Minimize the resulting waveform by temporally abstracting redundant grid points.

4 Time Derivatives of Completely Specified Waveforms

To analyze the dynamics of logic gate circuits we propose the use of derivative operators that are similar in spirit to the derivative operator D of the calculus of real numbers. The time derivatives of a symbolic logic waveform $\boldsymbol{w}(t)(X)$ are themselves symbolic logic waveforms that capture the conditions, both temporal and logical, under which $\boldsymbol{w}$ might be stable or changing. The derivatives allow the derivation of differentiation formulas for arbitrary Boolean combinations of waveforms; these formulas are used, in Section 5, as the basis for linking the functional and temporal aspects of combinational circuits. In addition, we show how a waveform's first and last event times are related to its derivatives. Due to space limitations, the discussion in this section is restricted to completely specified waveforms.

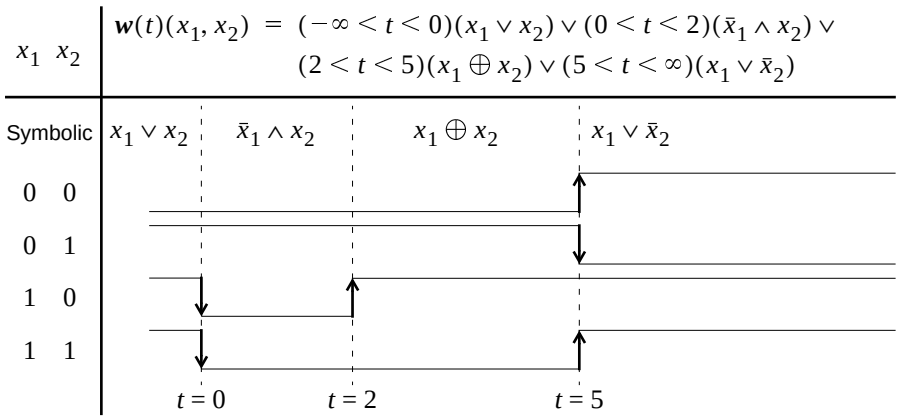
4.1 Basic Derivatives

The four time derivatives of $\boldsymbol{w}(t)(X)$ with respect to t are termed its *rising derivative* $R\boldsymbol{w}(t)(X)$, its *falling derivative* $F\boldsymbol{w}(t)(X)$, its *high derivative* $H\boldsymbol{w}(t)(X)$, and its *low derivative* $L\boldsymbol{w}(t)(X)$, and are defined by the following equations:

$$\begin{aligned} R\boldsymbol{w}(t)(X) &\equiv \lim_{\epsilon \to 0} [\bar{\boldsymbol{w}}(t-\epsilon)(X) \wedge \boldsymbol{w}(t+\epsilon)(X)] \\ F\boldsymbol{w}(t)(X) &\equiv \lim_{\epsilon \to 0} [\boldsymbol{w}(t-\epsilon)(X) \wedge \bar{\boldsymbol{w}}(t+\epsilon)(X)] \\ H\boldsymbol{w}(t)(X) &\equiv \lim_{\epsilon \to 0} [\boldsymbol{w}(t-\epsilon)(X) \wedge \boldsymbol{w}(t+\epsilon)(X)] \\ L\boldsymbol{w}(t)(X) &\equiv \lim_{\epsilon \to 0} [\bar{\boldsymbol{w}}(t-\epsilon)(X) \wedge \bar{\boldsymbol{w}}(t+\epsilon)(X)] \end{aligned} \tag{4.1}$$

Substituting the definition of $\boldsymbol{w}$ from (3.7) yields the following derivative expressions:

$$\begin{aligned} R\boldsymbol{w}(t)(X) &= \bigvee_{0 \le i \le k-1} (t = t_i) \wedge [\bar{\boldsymbol{w}}_i(X) \wedge \boldsymbol{w}_{i+1}(X)] \\ F\boldsymbol{w}(t)(X) &= \bigvee_{0 \le i \le k-1} (t = t_i) \wedge [\boldsymbol{w}_i(X) \wedge \bar{\boldsymbol{w}}_{i+1}(X)] \\ H\boldsymbol{w}(t)(X) &= \boldsymbol{w}(t)(X) \vee \left(\bigvee_{0 \le i \le k-1} (t = t_i) \wedge [\boldsymbol{w}_i(X) \wedge \boldsymbol{w}_{i+1}(X)] \right) \\ L\boldsymbol{w}(t)(X) &= \bar{\boldsymbol{w}}(t)(X) \vee \left(\bigvee_{0 \le i \le k-1} (t = t_i) \wedge [\bar{\boldsymbol{w}}_i(X) \wedge \bar{\boldsymbol{w}}_{i+1}(X)] \right) \end{aligned} \tag{4.2}$$



$$R\boldsymbol{w}(t)(x_1, x_2) = (t = 2)(x_1 \wedge \bar{x}_2) \vee (t = 5)(x_1 \odot x_2)$$
$$F\boldsymbol{w}(t)(x_1, x_2) = (t = 0)(x_1) \vee (t = 5)(\bar{x}_1 \wedge x_2)$$

Figure 4-1: Example waveform and its rising and falling derivatives

Note that $R\boldsymbol{w}$ and $F\boldsymbol{w}$ can be nonzero only at the grid points used to specify $\boldsymbol{w}$. We say that a *rising event* occurs at $(t = t_i)$ if $\overline{w}_i(X) \wedge w_{i+1}(X) = 1$. Similarly, a *falling event* is said to occur at $(t = t_i)$ if $w_i(X) \wedge \overline{w}_{i+1}(X) = 1$. An example showing the application of the derivative formulas in (4.2) is shown in Figure 4-1.

It is worth noting that at any given time instant, the four time derivatives of a waveform $\boldsymbol{w}$ form an orthonormal set. In addition, the derivatives at two consecutive grid points t_i and t_{i+1} are related by causality constraints; for example, $R\boldsymbol{w}(t_i)R\boldsymbol{w}(t_{i+1}) = 0$ and $R\boldsymbol{w}(t_i)L\boldsymbol{w}(t_{i+1}) = 0$ since a waveform cannot rise or be low at t_{i+1} if it was rising at t_i and remained stable until t_{i+1}.

4.2 Other Derivatives

If we are interested in just knowing whether a completely specified waveform is stable or changing at a particular time instant, it is useful to introduce two additional derivatives: $C\boldsymbol{w}(t)(X)$, the *changing derivative* of $\boldsymbol{w}$ with respect to t; and $S\boldsymbol{w}(t)(X)$, the *stable derivative* of $\boldsymbol{w}$ with respect to t. They are defined as follows:[2]

$$\begin{aligned} C\boldsymbol{w} &\equiv R\boldsymbol{w} \vee F\boldsymbol{w} \\ S\boldsymbol{w} &\equiv \bar{C}\boldsymbol{w} = \bar{R}\boldsymbol{w} \wedge \bar{F}\boldsymbol{w} \end{aligned} \tag{4.3}$$

The rising, falling, and changing derivatives can be evaluated over any subset of the time axis by simply ORing their values at all time instants in the given subset. Let $\mathcal{T} \subseteq \mathcal{R}$. Then,

2. For completely specified waveforms, orthonormality allows us to equivalently express the stable derivative as $S\boldsymbol{w} = H\boldsymbol{w} \vee L\boldsymbol{w}$. The more general definition in (4.3) is still required for partially specified waveforms, though.

$$
\begin{aligned}
R\boldsymbol{w}(\mathcal{T})(X) &\equiv \bigvee_{t \in \mathcal{T}} R\boldsymbol{w}(t)(X) \\
F\boldsymbol{w}(\mathcal{T})(X) &\equiv \bigvee_{t \in \mathcal{T}} F\boldsymbol{w}(t)(X) \\
C\boldsymbol{w}(\mathcal{T})(X) &\equiv \bigvee_{t \in \mathcal{T}} C\boldsymbol{w}(t)(X) \;=\; R\boldsymbol{w}(\mathcal{T})(X) \vee F\boldsymbol{w}(\mathcal{T})(X)
\end{aligned}
\tag{4.4}
$$

These definitions are consistent with the intuitive notion that a waveform should be considered rising, falling, or changing over a given set of time points if it rises, falls, or changes, at one or more of these time points. In contrast, the high, low, and stable derivatives are extended to sets of time points by ANDing their values at all time points:

$$
\begin{aligned}
H\boldsymbol{w}(\mathcal{T})(X) &\equiv \bigwedge_{t \in \mathcal{T}} H\boldsymbol{w}(t)(X) \\
L\boldsymbol{w}(\mathcal{T})(X) &\equiv \bigwedge_{t \in \mathcal{T}} L\boldsymbol{w}(t)(X) \\
S\boldsymbol{w}(\mathcal{T})(X) &\equiv \bigwedge_{t \in \mathcal{T}} S\boldsymbol{w}(t)(X) \;=\; [R\boldsymbol{w}(\mathcal{T})(X)]' \wedge [F\boldsymbol{w}(\mathcal{T})(X)]'
\end{aligned}
\tag{4.5}
$$

Thus, a waveform is considered to be high (low) over a given set of time points if and only if it is high (low) at each time point in the set. It is considered to be stable over the given set of time points if it is guaranteed that it neither rises nor falls at any time point in the set.

4.3 First and Last Event Times

The times at which the first and last event of a waveform $\boldsymbol{w}$ occur are frequently needed. Indeed, it is fair to say that the field of timing analysis is primarily concerned with the determination of these event times under a variety of simplifying approximations. In this section we derive formulas for these two quantities that show them to be conditional numbers whose conditions are related to a waveform's derivatives.

Let's consider the time at which the last event of $\boldsymbol{w}$ occurs. We begin by noting that this quantity is going to be a conditional number. Its possible values come from the time grid $T_{\boldsymbol{w}}$ used to specify $\boldsymbol{w}$. Each grid point t_i is a candidate value for the time of the last event if $\boldsymbol{w}$ changes at t_i and is guaranteed to remain stable after t_i. This requirement is easily expressed by:

$$C\boldsymbol{w}(t_i) \wedge S\boldsymbol{w}((t_i, \infty)) \tag{4.6}$$

Denoting the time of the last event of $\boldsymbol{w}$ by $A_{\boldsymbol{w}}$, it can be succinctly expressed as follows:

$$A_{\boldsymbol{w}} \;=\; \bigoplus_{-1 \le i \le k-1} (C\boldsymbol{w}(t_i) \wedge S\boldsymbol{w}((t_i, \infty)))\langle t_i \rangle \tag{4.7}$$

where $C\boldsymbol{w}(-\infty) \equiv 1$. Recalling that $t_{-1} = -\infty$, the above expression allows for the possibility of a waveform that has no events (i.e., that is stable for all points on the time axis) by assigning $-\infty$ as the time of the last event (the event that never took place!)

Similar reasoning yields the following expression for $a_{\boldsymbol{w}}$, the time of the first event[3] of $\boldsymbol{w}$:

Table 4-1: Differentiation Formulas

$z^*(t)(X) = f(w(t)(X), y(t)(X))$				
z^*	Rz^*	Fz^*	Hz^*	Lz^*
$\bar{w}$	Fw	Rw	Lw	Hw
wy	$RwRy \vee RwHy \vee HwRy$	$FwFy \vee FwHy \vee HwFy$	$HwHy$	$Lw \vee Ly \vee RwFy \vee FwRy$
$w \vee y$	$RwRy \vee RwLy \vee LwRy$	$FwFy \vee FwLy \vee LwFy$	$Hw \vee Hy \vee RwFy \vee FwRy$	$LwLy$
$w \oplus y$	$LwRy \vee FwHy \vee$ $HwFy \vee RwLy$	$LwFy \vee FwLy \vee$ $HwRy \vee RwHy$	$LwHy \vee HwLy \vee$ $FwRy \vee RwFy$	$LwLy \vee HwHy \vee$ $FwFy \vee RwRy$

$$a_w = \bigoplus_{0 \le i \le k} (Sw((-\infty, t_i)) \wedge Cw(t_i))\langle t_i \rangle \tag{4.8}$$

In this case, the time of the first event is set to ∞ when the waveform is stable at all time points (the event that will never occur!)

Example 4.1 Using (4.9) and (4.8), the first and last events of the waveform in Figure 4-1 can be calculated as follows:

$$a_w = \begin{cases} 0 \text{ if } Sw((-\infty, 0)) \wedge Cw(0) \\ 2 \text{ if } Sw((-\infty, 2)) \wedge Cw(2) \\ 5 \text{ if } Sw((-\infty, 5)) \wedge Cw(5) \\ \infty \text{ if } (Sw((-\infty, \infty))) \end{cases} = x_1\langle 0 \rangle \oplus \bar{x}_1\langle 5 \rangle \tag{4.9}$$

$$A_w = \begin{cases} 5 \quad \text{if } Cw(5) \wedge Sw((5, \infty)) \\ 2 \quad \text{if } Cw(2) \wedge Sw((2, \infty)) \\ 0 \quad \text{if } Cw(0) \wedge Sw((0, \infty)) \\ -\infty \text{ if } Sw((-\infty, \infty)) \end{cases} = (\bar{x}_1 \vee x_2)\langle 5 \rangle \oplus (x_1 \wedge \bar{x}_2)\langle 2 \rangle \tag{4.10}$$

The correctness of these results is easily validated by inspection of the waveforms in Figure 4-1. □

4.4 Differentiation Formulas

The time derivatives of any Boolean function of logic waveforms are easily found by direct, if somewhat tedious, application of the definitions in (4.1). The derivative sets for a sampling of one- and two-variable functions are given in Table 4-1. Not surprisingly, these derivative formulas have simple intuitive explanations. For example, the rising derivative of wy indicates that wy rises at some time t if either input is rising at that time while the other input is either rising or is high. There are some

3. The "a" in A_w and a_w stands for the "arrival" time of the corresponding event.

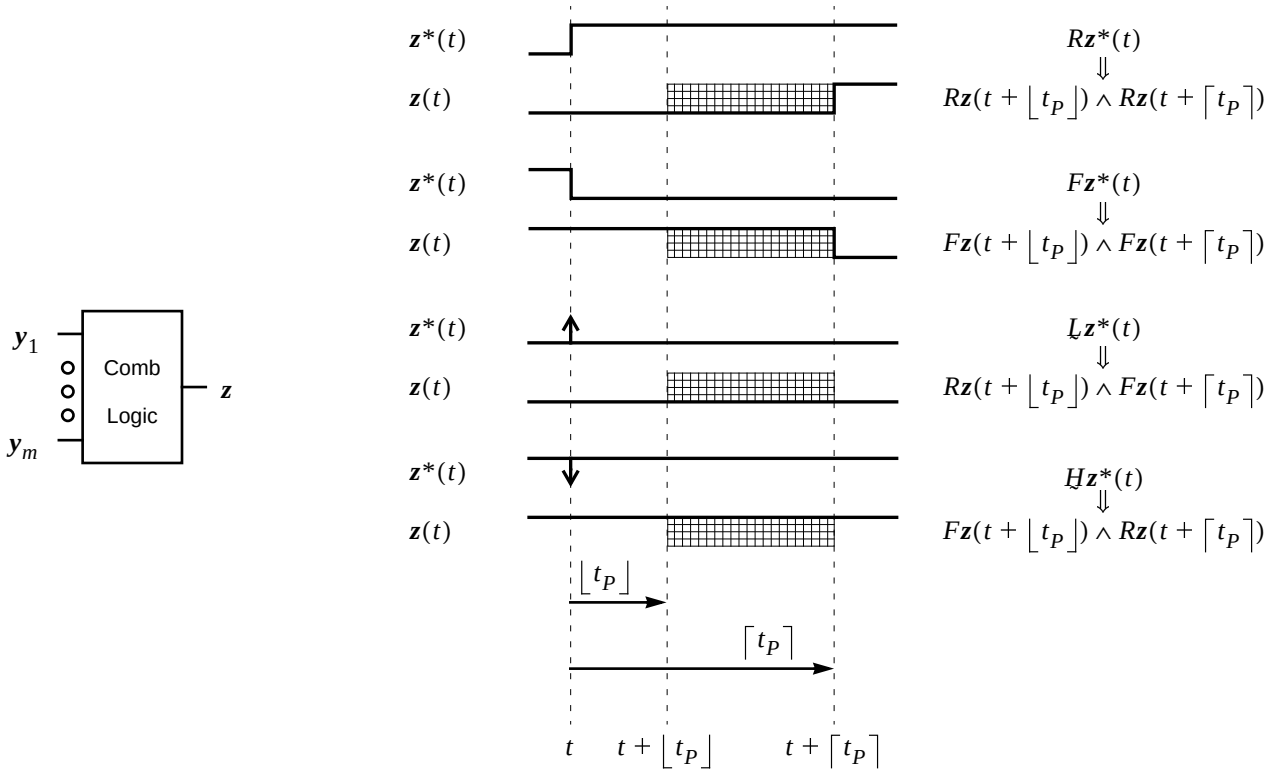


Figure 5-1: The link between time derivatives and signal delay. z^* is the ideal instantaneous output waveform; z is the actual output waveform.

subtleties, however. Specifically, note that the low derivative of $\boldsymbol{wy}$ consists of four disjoint terms indicating that a sufficient condition for $\boldsymbol{wy}$ to be low is that either input is low or that the two inputs are undergoing opposite transitions. However, it should be clear that the appearance of rising and falling derivatives in an expression for a low or high derivative is an indication of a static hazard that might result in a glitch. When necessary, we will indicate these glitch-inducing derivatives with a tilde under the appropriate derivative operator. Thus, for the AND function we would write:

$$\underset{\sim}{L}(\boldsymbol{wy}) = R\boldsymbol{w}F\boldsymbol{y} \vee F\boldsymbol{w}R\boldsymbol{y} \Rightarrow L(\boldsymbol{wy}) = L\boldsymbol{w} \vee L\boldsymbol{y} \vee \underset{\sim}{L}(\boldsymbol{wy})$$

Another subtlety is that these derivatives pertain to the instantaneous (delayless) waveform resulting from application of the Boolean function f. In other words, these derivatives are for $\boldsymbol{z}^*(t)(X) = f(\boldsymbol{w}(t)(X), \boldsymbol{y}(t)(X))$; they indicate the conditions on the input waveforms at time *t* that would cause the instantaneous output waveform to rise, fall, remain high, or remain low. What happens to the actual waveform is discussed next.

5 Signal Delay

The link between the functional and temporal aspects of logic gate models can be inferred by noting that a change indication for the instantaneous output waveform $\boldsymbol{z}^*$ at time *t* implies corresponding changes on the real output waveform $\boldsymbol{z}$ after some *propagation time* t_P. The relation between the four possible change indications on $\boldsymbol{z}^*$ and the changes they imply on $\boldsymbol{z}$ is illustrated in Figure 5-1. Note that, in addition to $R\boldsymbol{z}^*(t)$ and $F\boldsymbol{z}^*(t)$, the glitch-inducing derivatives $\underset{\sim}{L}\boldsymbol{z}^*(t)$ and $\underset{\sim}{H}\boldsymbol{z}^*(t)$

must be considered as change indicators. In making this link between the functional (given by the derivatives) and temporal (given by the propagation times) components of a gate model, it is important to emphasize that no assumptions were made about the implementation of the gate function f. Thus, different implementations could yield markedly different propagation times. More to the point, perhaps, is the fact that all we can say when, for example, a rise indication exists for $\boldsymbol{z}^*$ at time *t*, is that the real output will *eventually* rise. It may, depending on implementation, have a single rising event or a sequence of rising and falling transitions that end in a final rising event. To accommodate these variations, the propagation time t_P is best considered as a range that bounds the changing interval on $\boldsymbol{z}$; the lower and upper bounds of this range will be denoted by $\lfloor t_P \rfloor$ and $\lceil t_P \rceil$. It is important to note that the range in propagation time introduced here is meant to model the possible existence of several structural propagation paths within the logic gate and not any statistical effects caused by random variations in processing or operating conditions. Indeed, the special case of a degenerate range (i.e. $\lfloor t_P \rfloor = \lceil t_P \rceil$) may be used to differentiate between structurally *primitive* and *complex* components. In the remainder of this section we will assume that t_P is a degenerate range (a single point) in order to keep the waveforms completely specified.

5.1 Delay Specification

Having established the link between waveform derivatives and propagation times, we can now label each of these times based on the logical conditions that cause its manifestation. To illustrate, consider a 2-input AND gate with inputs $\boldsymbol{y}_1$, $\boldsymbol{y}_2$ and output $\boldsymbol{z}$. Consulting the derivative formulas in Table 4-1 we can identify six potential propagation times from the inputs of the gate to its output $\boldsymbol{z}$ corresponding to the six different rise and fall indications on the instantaneous output $\boldsymbol{z}^*$. A possible systematic labeling of these six propagation times, thus, is[4] $t_{PRH}^{R\boldsymbol{z}}$, $t_{PHR}^{R\boldsymbol{z}}$, $t_{PRR}^{R\boldsymbol{z}}$, $t_{PFH}^{F\boldsymbol{z}}$, $t_{PHF}^{F\boldsymbol{z}}$, $t_{PFF}^{F\boldsymbol{z}}$.

Denoting the signal propagation delay to $\boldsymbol{z}$ at time *t* by $\Delta_{*\boldsymbol{z}}(t)(X)$, it can now be expressed as a conditional number whose conditions are the relevant derivative combinations and whose values are the corresponding propagation times:

$$\begin{aligned}\Delta_{*\boldsymbol{z}}(t)(X) \;=\; & Ry_1Hy_2\langle t_{PRH}^{R\boldsymbol{z}}\rangle \oplus Hy_1Ry_2\langle t_{PHR}^{R\boldsymbol{z}}\rangle \oplus Ry_1Ry_2\langle t_{PRR}^{R\boldsymbol{z}}\rangle \oplus \\ & Fy_1Hy_2\langle t_{PFH}^{F\boldsymbol{z}}\rangle \oplus Hy_1Fy_2\langle t_{PHF}^{F\boldsymbol{z}}\rangle \oplus Fy_1Fy_2\langle t_{PFF}^{F\boldsymbol{z}}\rangle \oplus \\ & (Ry_1Hy_2 \vee Hy_1Ry_2 \vee Ry_1Ry_2 \vee Fy_1Hy_2 \vee Hy_1Fy_2 \vee Fy_1Fy_2)'\langle 0\rangle\end{aligned} \tag{5.1}$$

Note that the delay is set to zero when none of the change-inducing conditions apply; this merely states the fact that if no events occur on the inputs to the gate, then there is nothing to delay! Note also that the delays from each of the inputs to the output can be "extracted" from the above composite delay equation by selecting only those conditions that involve a change in that particular input. Thus, the delays from $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ to $\boldsymbol{z}$ are immediately seen to be:

$$\begin{aligned}\Delta_{\boldsymbol{y}_1\boldsymbol{z}}(t)(X) \;=\; & Ry_1Hy_2\langle t_{PRH}^{R\boldsymbol{z}}\rangle \oplus Ry_1Ry_2\langle t_{PRR}^{R\boldsymbol{z}}\rangle \oplus Fy_1Hy_2\langle t_{PFH}^{F\boldsymbol{z}}\rangle \oplus Fy_1Fy_2\langle t_{PFF}^{F\boldsymbol{z}}\rangle \oplus \\ & (Ry_1Hy_2 \vee Ry_1Ry_2 \vee Fy_1Hy_2 \vee Fy_1Fy_2)'\langle 0\rangle \\ \Delta_{\boldsymbol{y}_2\boldsymbol{z}}(t)(X) \;=\; & Hy_1Ry_2\langle t_{PHR}^{R\boldsymbol{z}}\rangle \oplus Ry_1Ry_2\langle t_{PRR}^{R\boldsymbol{z}}\rangle \oplus Hy_1Fy_2\langle t_{PHF}^{F\boldsymbol{z}}\rangle \oplus Fy_1Fy_2\langle t_{PFF}^{F\boldsymbol{z}}\rangle \oplus \\ & (Hy_1Ry_2 \vee Ry_1Ry_2 \vee Hy_1Fy_2 \vee Fy_1Fy_2)'\langle 0\rangle\end{aligned} \tag{5.2}$$

4. This notation is meant to resemble the traditional notation for specifying delay parameters in databooks. The "*t*" stands for time, the "*P*" for propagation (thus, propagation time.) The remaining subscripts denote the input events assuming a particular input ordering. The superscript indicates the waveform (gate label) to which the delay parameter applies and the type of event involved.

5.2 Delay Application

The preceding development clearly shows the conditional nature of logic gate delay and highlights its relationship to the Boolean function of the gate. To complete our evolving dynamic logic gate model, we must lastly examine how such conditional delays affect the propagation of symbolic logic waveforms. Consider a generic waveform $\boldsymbol{w}(t)(X)$ as defined in (3.7) and let $\Delta(t)(X) = \bigvee_{0 \le i \le k-1}(t = t_i)\langle\Delta_i(X)\rangle$ where $\Delta_i(X)$ are conditional numbers (this is the form that results from substituting the expressions for the time derivatives in (5.1)). The waveform that results from delaying $\boldsymbol{w}(t)(X)$ by $\Delta(t)(X)$ is evaluated according to:

$$\boldsymbol{w}(t - \Delta(t)(X))(X) \equiv \bigvee_{0 \le i \le k-1} (t_{i-1} + \Delta_{i-1}(X) < t < t_i + \Delta_i(X)) \wedge \boldsymbol{w}_i(X) \tag{5.3}$$

where the addition and relational operations are carried out using the conditional arithmetic rules of Section 2.1. Several points are worth noting about this delay operation:

- Let's assume for a moment that the Δ_i's are unconditional numbers, i.e. they are independent of X. In that case, (5.4) indicates that the delay operation transforms the original grid $\{t_0, t_1, \ldots, t_{k-1}\}$ to a new grid $\{t_0 + \Delta_0, t_1 + \Delta_1, \ldots, t_{k-1} + \Delta_{k-1}\}$. In addition, if the delay values are equal (uniform delay), then the grid is merely shifted by the delay amount. If, however, the delay values are not equal, the spacing of the new grid will be different from the original grid. Thus, some intervals might shrink while others might expand, and as a result the new intervals may overlap. The value of the delayed waveform in these overlap regions is the logical OR of the respective $\boldsymbol{w}_i$'s.

- In extreme cases, intervals might shrink to zero or negative width. Their contribution to the value of the delayed waveform in such cases reduces to zero because the temporal predicate $(t_{i-1} + \Delta_{i-1} < t < t_i + \Delta_i)$ becomes 0. In some sense, this is a form of inertial filtering with a minimum pulse width of 0 and it is taken care of automatically. This also suggests a possible approach to general inertial delay modeling. Suppose that there is a minimum pulse width $t_{MPW} > 0$ that must be satisfied by all signals. The delay operation defined in (5.4) can be extended to handle this case as follows:

$$\boldsymbol{w}(t - \Delta(t)) = \bigvee_{0 \le i \le k-1} (t_{i-1} + \Delta_{i-1} + t_{MPW} < t_i + \Delta_i) \wedge (t_{i-1} + \Delta_{i-1} < t < t_i + \Delta_i) \wedge \boldsymbol{w}_i \tag{5.4}$$

  The additional predicate eliminates the contribution of $\boldsymbol{w}_i$ to the value of the delayed waveform if the transformed interval is narrower than the specified minimum pulse width.

- When the Δ_i's are allowed to be conditional numbers (functions of X), the above observations still hold. The transformation to the time grid, however, is more involved. Each original grid point now generates several new grid points depending on how many delay values and associated conditions are associated with that grid point. The conditional arithmetic machinery takes care of this automatically.

5.3 Complete Integrated Functional and Temporal Gate Model

If the functional dependence of $\boldsymbol{z}$ on $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m$ is given by the Boolean function f, then the complete waveform at $\boldsymbol{z}$ is calculated from:

$$\boldsymbol{z}(t) \;=\; f(\boldsymbol{y}_1(t - \Delta_{\boldsymbol{y}_1 z}(t)),\, \boldsymbol{y}_2(t - \Delta_{\boldsymbol{y}_2 z}(t)),\, \ldots,\, \boldsymbol{y}_m(t - \Delta_{\boldsymbol{y}_m z}(t))) \tag{5.5}$$

This equation may imply a front-end delay model. It can be written in an equivalent form, however, that looks like a back-end delay model:

$$\boldsymbol{z}(t) \;=\; f(\boldsymbol{y}_1, \boldsymbol{y}_2 \ldots, \boldsymbol{y}_m)(t - \Delta_{*z}(t)) \;=\; \boldsymbol{z}^*(t - \Delta_{*z}(t)) \tag{5.6}$$

Thus, as long as we treat delay properly, we get the right answer in one of two ways: 1) apply the static logic function to the inputs, then delay the result by Δ_{*z}; 2) delay each of the inputs by the portion of Δ_{*z} that pertains to it, then apply the static logic function to the result. This is an interesting conclusion and suggests that the traditional dichotomy between front- and back-end delay models is an artifact of how delay was postulated. It is clear from the above analysis that there is no difference between these two styles of modeling logic delay as long as they derive from a consistent causal definition of delay.

6 Conclusions and Future Work

In this paper we have sketched the outlines of a proposed framework for the dynamic modeling of logic gate circuits. The thesis underlying this framework is that functional and temporal properties are inherently related. By contrast, most existing logic gate models treat these two facets independently and somewhat inconsistently. The unification provided by our framework is not merely aesthetic; it allows us to explore at a fundamental level the nature of logic delay and forms the basis for more rigorous timing analysis models than are currently available. Specifically, through a systematic functional and temporal abstraction procedure we can derive early and late signal arrival time equations that mirror in their form the topological equations at the core of the shortest and longest critical path method (CPM). To illustrate, consider an *m*-input single-output gate in a combinational circuit. Denoting the late topological arrival times at the inputs and outputs of the gate by $A_{\boldsymbol{y}_1}, \ldots, A_{\boldsymbol{y}_m}$ and $A_{\boldsymbol{z}}$, and the (unconditional) delays from each of these inputs to the output by $\Delta_{\boldsymbol{y}_1 \boldsymbol{z}}, \ldots, \Delta_{\boldsymbol{y}_m \boldsymbol{z}}$, the basic computation in CPM is given by the equation:

$$A_z \;=\; \max_{1 \le i \le m} [A_{y_i} + \Delta_{y_i z}] \tag{6.1}$$

The corresponding equation when the arrival times and signal delays are conditional numbers that are functions of an independent Boolean vector *X* is then simply:

$$A_z(X) \;=\; \max_{1 \le i \le m} [A_{y_i}(X) + \Delta_{y_i z}(A_{y_i})(X)] \tag{6.2}$$

where the rules of conditional arithmetic apply. The significance of the above equation lies in its universal applicability to any logic function f augmented with the most comprehensive delay model proposed to date. It clearly supersedes the plethora of *local sensitization criteria* that have been proposed in the last decade to deal with the timing analysis problem and rationalizes a field that has so far been treated by ad hoc methods.

Space limitations have precluded discussion of several related topics including the modeling of such analog effects as signal slope and the near-simultaneous switching of inputs. We also hinted at but did not fully develop the model of partially specified waveforms. We plan to describe these ideas in future publications.

References

[1] F. M. Brown, *Boolean Reasoning*. ed. J. Allen. 1990, Kluwer Academic Publishers.

[2] C. T. Gray, W. Liu and R. K. Cavin III, "Exact Timing Analysis Considering Data Dependent Delays," Technical Report NCSU-VLSI-92-04, North Carolina State University, December 1992.
[3] W. K.-C. Lam, *Algebraic Methods for Timing Analysis and Testing in High Performance Designs*, Ph.D. Thesis, University of California, Berkeley, 1994.
[4] S.-Z. Sun, D. H. C. Du and H.-C. Chen, "Efficient Timing Analysis for CMOS Circuits Considering Data Dependent Delays," in *Proc. IEEE International Conference on Computer Design (ICCD)*, pp. 156-159, October 1994, Cambridge, Massachusetts.

Application of Conditional Delay to Symbolic Logic Waveforms (*A Correction to Section 5.2 of CSE-TR-253-95*)

In some cases, equation (5.3) on page 16 yields incorrect waveforms. It should be replaced with the following corrected version:

$$\boldsymbol{w}(t - \Delta(t)(X))(X) \equiv \bigvee_{0 \le i \le k-1} (t_{i-1} + \hat{\Delta}_{i-1}(X) < t < t_i + \hat{\Delta}_i(X)) \wedge \boldsymbol{w}_i(X) \tag{7.1}$$

where the conditional delay $\hat{\Delta}_i(X)$ that is applied at grid point t_i is a function of the actual conditional delay $\Delta_i(X)$ at t_i as well as the conditional delays at all preceding grid points. Let $\pi_i(X)$ denote the *propagation condition* associated with the conditional delay $\Delta_i(X)$. Thus, $\pi_i(X)$ is the union (OR) of all the conditions under which the delay is nonzero. As an example, the propagation condition for the delay $\Delta_{*\boldsymbol{z}}$ of a 2-input AND gate is (see equation (5.1)):

$$\pi_{*\boldsymbol{z}} = Ry_1Hy_2 \vee Hy_1Ry_2 \vee Ry_1Ry_2 \vee Fy_1Hy_2 \vee Hy_1Fy_2 \vee Fy_1Fy_2 \tag{7.2}$$

The basic idea now is that the amount of delay to apply at grid point t_i should reflect the cumulative effect of the conditional delays at previous grid points. Thus, when the propagation condition $\pi_i(X)$ is true, t_i is delayed by $\Delta_i(X)$; when $\pi_i(X)$ is false, however, t_i is delayed by $\hat{\Delta}_{i-1}(X)$:

$$\hat{\Delta}_i(X) = \pi_i(X) \wedge \Delta_i(X) \vee \bar{\pi}_i(X) \wedge \hat{\Delta}_{i-1}(X) \tag{7.3}$$

This equation can be viewed as constructing a new conditional number by splicing together portions from two other conditional numbers. The propagation condition and its complement are used to select the portions to be spliced. Defining the delay at grid point $t_{-1} \equiv -\infty$ to be $\hat{\Delta}_{-1}(X) = \Delta_{-1}(X) = 0$ allows this equation to be used recursively to determine $\hat{\Delta}_0(X), \hat{\Delta}_1(X), \ldots, \hat{\Delta}_{k-1}(X)$. Note that (7.3) reduces to $\hat{\Delta}_i(X) = \Delta_i(X)$ when the delays are unconditional, i.e. when $\pi_i(X) = 1$.

Example 7.1

$$\begin{aligned} \boldsymbol{y}_1(t)(x_1, x_2) &= (-\infty < t < 0)(x_1) \vee (0 < t < \infty)(x_2) \\ \boldsymbol{y}_2(t)(x_1, x_2) &= (-\infty < t < 1)(x_2) \vee (1 < t < \infty)(\bar{x}_1) \end{aligned}$$

First, compute the delayless output:

$$z^* = y_1 \wedge y_2 = (-\infty < t < 0)(x_1x_2) \vee (0 < t < 1)(x_2) \vee (1 < t < \infty)(\bar{x}_1x_2)$$

Next, compute its derivatives:

$$Rz^* = (t = 0)(\bar{x}_1x_2)$$
$$Fz^* = (t = 1)(x_1x_2)$$

Now express gate delay $\Delta_{*z}(t)$ in terms of these derivatives. The delay is nonzero only at two time instants:

$$\Delta_{*z}(0) = (\bar{x}_1x_2)\langle t_{PRH}^{Rz}\rangle \oplus (x_1 \vee \bar{x}_2)\langle 0\rangle$$
$$\Delta_{*z}(1) = (x_1x_2)\langle t_{PHF}^{Fz}\rangle \oplus (\bar{x}_1 \vee \bar{x}_2)\langle 0\rangle$$

The actual gate output z is now obtained by evaluating $z^*(t - \Delta_{*z}(t))$:

$$z(t) = (-\infty < t < \hat{\Delta}_{*z}(0))(x_1x_2) \vee (\hat{\Delta}_{*z}(0) < t < 1 + \hat{\Delta}_{*z}(1))(x_2) \vee (1 + \hat{\Delta}_{*z}(1) < t < \infty)(\bar{x}_1x_2)$$

where:

$$\hat{\Delta}_{*z}(0) = \Delta_{*z}(0) = (\bar{x}_1x_2)\langle t_{PRH}^{Rz}\rangle \oplus (x_1 \vee \bar{x}_2)\langle 0\rangle$$
$$\hat{\Delta}_{*z}(1) = (x_1x_2) \wedge \Delta_{*z}(1) \vee (\bar{x}_1 \vee \bar{x}_2) \wedge \hat{\Delta}_{*z}(0)$$
$$= (x_1x_2)\langle t_{PHF}^{Fz}\rangle \oplus (\bar{x}_1x_2)\langle t_{PRH}^{Rz}\rangle \oplus (\bar{x}_2)\langle 0\rangle$$

Thus,

$$\begin{aligned} z(t) &= (-\infty < t)[(\bar{x}_1x_2)(t < t_{PRH}^{Rz}) \vee (x_1 \vee \bar{x}_2)(t < 0)](x_1x_2) \vee \\ &\quad [(\bar{x}_1x_2)(t_{PRH}^{Rz} < t) \vee (x_1 \vee \bar{x}_2)(0 < t)][(x_1x_2)(t < 1 + t_{PHF}^{Fz}) \vee (\bar{x}_1x_2)(t < 1 + t_{PRH}^{Rz}) \vee (\bar{x}_2)(t < 1)](x_2) \\ &\quad [(x_1x_2)(1 + t_{PHF}^{Fz} < t) \vee (\bar{x}_1x_2)(1 + t_{PRH}^{Rz} < t) \vee (\bar{x}_2)(1 < t)](t < \infty)(\bar{x}_1x_2) \\ &= (-\infty < t < 0)(x_1x_2) \vee \\ &\quad [(\bar{x}_1x_2)(t_{PRH}^{Rz} < t) \vee (x_1x_2)(0 < t)][(x_1x_2)(t < 1 + t_{PHF}^{Fz}) \vee (\bar{x}_1x_2)(t < 1 + t_{PRH}^{Rz})] \vee \\ &\quad (1 + t_{PRH}^{Rz} < t < \infty)(\bar{x}_1x_2) \\ &= (-\infty < t < 0)(x_1x_2) \vee \\ &\quad (t_{PRH}^{Rz} < t < 1 + t_{PRH}^{Rz})(\bar{x}_1x_2) \vee (0 < t < 1 + t_{PHF}^{Fz})(x_1x_2) \vee \\ &\quad (1 + t_{PRH}^{Rz} < t < \infty)(\bar{x}_1x_2) \end{aligned}$$

Under the uniform delay assumption, $t_{PRH}^{Rz} = t_{PHF}^{Fz} = 3$ leading to:

$$\begin{aligned} z(t) &= (-\infty < t < 0)(x_1x_2) \vee (3 < t < 4)(\bar{x}_1x_2) \vee (0 < t < 4)(x_1x_2) \vee (4 < t < \infty)(\bar{x}_1x_2) \\ &= (-\infty < t < 3)(x_1x_2) \vee (3 < t < 4)(x_2) \vee (4 < t < \infty)(\bar{x}_1x_2) \end{aligned}$$

Under the rise/fall and state-dependent delay assumptions, $t_{PRH}^{\boldsymbol{Rz}} = 3$ and $t_{PHF}^{\boldsymbol{Fz}} = 1$ leading to:

$$\begin{aligned}\boldsymbol{z}(t) &= (-\infty < t < 0)(x_1x_2) \vee (3 < t < 4)(\bar{x}_1x_2) \vee (0 < t < 2)(x_1x_2) \vee (4 < t < \infty)(\bar{x}_1x_2)\\ &= (-\infty < t < 2)(x_1x_2) \vee (3 < t < \infty)(\bar{x}_1x_2)\end{aligned}$$

Had we used the delay application formula in (5.3) we would have obtained the following erroneous output waveform (for the uniform delay case):

$$\begin{aligned}\boldsymbol{z}(t) &= (-\infty < t)[(\bar{x}_1x_2)(t < 3) \vee (x_1 \vee \bar{x}_2)(t < 0)](x_1x_2) \vee\\ &\quad [(\bar{x}_1x_2)(3 < t) \vee (x_1 \vee \bar{x}_2)(0 < t)][(x_1x_2)(t < 4) \vee (\bar{x}_1 \vee \bar{x}_2)(t < 1)](x_2) \vee\\ &\quad [(x_1x_2)(4 < t) \vee (\bar{x}_1 \vee \bar{x}_2)(1 < t)](t < \infty)(\bar{x}_1x_2)\\ &= (-\infty < t < 0)(x_1x_2) \vee\\ &\quad [(\bar{x}_1x_2)(3 < t) \vee (x_1x_2)(0 < t)][(x_1x_2)(t < 4) \vee (\bar{x}_1x_2)(t < 1)] \vee\\ &\quad (1 < t < \infty)(\bar{x}_1x_2)\\ &= (-\infty < t < 0)(x_1x_2) \vee (0 < t < 4)(x_1x_2) \vee (1 < t < \infty)(\bar{x}_1x_2)\\ &= (-\infty < t < 1)(x_1x_2) \vee (1 < t < 4)(x_2) \vee (4 < t < \infty)(\bar{x}_1x_2)\end{aligned}$$

The basic cause of this anomalous result is that the propagation conditions at $t = 0$ and at $t = 1$ are mutually exclusive and that the latter condition incorrectly annihilated the effect of the earlier one. When the effect of the earlier condition is incorporated into the delay at $t = 1$, the anomaly disappears.