Ternary Simulation: A Refinement of Binary Functions or an Abstraction of Real-Time Behaviour?

Michael Mendler* Matt Fairtlough†

Abstract

We prove the equivalence between the ternary circuit model and a notion of intuitionistic stabilization bounds. The results are obtained as an application of the timing interpretation of intuitionistic propositional logic presented in [12]. We show that if one takes an intensional view of the ternary model then the delays that have been abstracted away can be completely recovered. Our intensional soundness and completeness theorems imply that the extracted delays are both correct and exact; thus we have developed a framework which unifies ternary simulation and functional timing analysis. Our focus is on the combinational behaviour of gate-level circuits with feedback.

1 Motivation

When a binary digital circuit, say a network composed of AND, OR, INV gates etc., does not contain feedback loops its static behaviour can be dealt with completely and adequately by standard Boolean two-valued analysis. However, when one is interested in delay-related phenomena such as *e.g.* hazards, races, glitches, or when feedback loops cannot be avoided, as *e.g.* in asynchronous circuits, the two-valued Boolean model is no longer adequate. The *ternary model* has been introduced as a natural extension of the two-valued model to analyse circuits in the presence of propagation delays and oscillations. A third value is added to give a minimum extra capacity for accommodating time-related features of real circuits, without entering the descriptive and algorithmic complexity of a full real-time analysis.

Viewed as an extension of classical propositional logic the ternary model occurs already in Kleene's work on partial recursive functions [8]. As a three-valued signal algebra the ternary model was introduced by Yoeli and Rinon [15] to analyse static hazards. Eichelberger [6] extended the method to handle general hazards in combinational circuits, and races and oscillations in sequential circuits. Later the theory and application of *ternary*

*Department of Mathematics and Computer Science, University of Passau, Innstraße 33, D-94032 Passau. Email: mendler@fmi.uni-passau.de

†Department of Computer Science, The University of Sheffield, Regent Court, 211 Portobello St., Sheffield S1 4DP. Email: m.fairtlough@dcs.shef.ac.uk

simulation has been developed further by various authors, *e.g.* at the gate level by Brzozowski and Yoeli in [14, 5], and Malik in [10], or at the transitor level by Bryant [4].

While the relationship between the static two-valued model and the physical real-time behaviour of a binary circuit is rather straightforward and well understood, the corresponding relationship for the ternary model is far from being clear. Surely, given it is used in the right way, it will recover certain time-related features of real circuits. But just what kind of extra timing information is it that is captured? The published literature does not provide an answer to this question since the ternary model invariably is introduced as a *refinement* of the abstract Boolean view rather than an *abstraction* of real-time behaviour. The issue of a formal abstraction relationship is not addressed. To bridge the gap ternary simulation appeals merely to an intuitive interpretation of the three signal values in terms of real signal waveforms. The new, third, value is given various informal readings like "oscillation", "transient", "undefined binary value", "don't care." The exact meaning is left unspecified, present only implicitly in the way this new value is used in the ternary modelling or ternary simulation.

In this report we take a new look at the ternary circuit model. We present it as a result of reducing real-time information, rather than as a result of enriching an abstract two-valued model. Concretely, we obtain a formal link between ternary simulation and an intuitionistic axiomatics of real-time behaviour. In this approach the third value represents the *absence* of definite information about the *bounded stabilization* of a signal.

We interpret the standard ternary function tables of binary gates and binary gate networks both as programs and as logic specifications. A formal language of ternary function tables is introduced with associated operational and axiomatic semantics. The operational semantics corresponds to a simple form of ternary simulation, while the axiomatic semantics explains the function tables as specifications of bounded real-time stabilization behaviour. We prove that the operational semantics is sound and complete w.r.t. the axiomatic one, thus establishing a formal link between ternary simulation and real-time behaviour. Moreover, we can show that this correspondence does not only hold in the extensional, *i.e.* ternary sense, but also in an intensional sense. This means that we can maintain and manipulate exact real-time delay information in the process of ternary simulation, and thus naturally combine, in the ternary model, both functional and timing analysis of binary gate networks. Both aspects are traditionally treated as separate tasks.

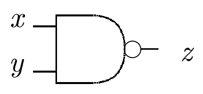
As a side-result of this report a natural generalization of the ternary model is suggested. We prove that ternary function tables may be seen as the first-order Horn fragment of (a theory of) Propositional Lax Logic [7]. In this sense the ternary model can be embedded into a full-fledged propositional logic in which we can express higher-order stabilization behaviour that goes beyond the three-valued model.

2 Introduction

We are interested in gate-level circuits, *i.e.* networks built from components like INV, AND, OR, NAND gates, *etc.* A *signal* a is a timed Boolean-valued function $a \in \mathbb{N} \to \mathbb{B}$, time being represented by the natural numbers. For convenience we will fix a countably infinite number of signal *names* $\mathbb{S} = \{a, b, c, c_1, c_2, \ldots, x, y, z\}$ throughout. A *waveform* is a mapping $V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ which assigns to every signal name a concrete signal. When V is understood we may confuse a signal name $a \in \mathbb{S}$ with its associated signal $V(a) \in \mathbb{N} \to \mathbb{B}$. Finally, a circuit is conceived as a subset $C \subseteq \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ of waveforms which constrains the behaviour on (a finite number of) signals. The elements $V \in C$ might be called *observable behaviours*, or *executions* of C. This nails down our low-level real-time model of behaviours. Compared to the kind of models used in dynamic system theory the model is rather abstract in the sense that it builds on discrete data values, discrete time, and initial state. On the other hand it does not constrain the behaviour of gates in a lot of ways. For instance, we may model the behaviour of a physical gate as a deterministic function or a nondeterministic relation, use transport or inertial delay assumption and we may have data and input dependent delays or fixed delays. In the sequel we will be concerned with ways of abstracting this fine-grained model into two and three-valued domains.

The three-valued domain arises in the specification of binary gate networks when the global assumption of stable signals is lifted. Let $\mathbb{K} = \{0, \frac{1}{2}, 1\}$ be the three-valued domain extending the Booleans $\mathbb{B}$ by an additional value $\frac{1}{2}$, which depending on the context, may have different intuitive readings. In typical interpretations $\frac{1}{2}$ would stand for "oscillation", "instability", "undefined", "don't care". The exact meaning usually is left unspecified, only intensionally present in the way $\frac{1}{2}$ is used. Following [5], for instance, the ternary function table for the NAND gate is obtained as a canonical refinement of the standard Boolean function table, as seen in Fig. 1. Our goal is to understand this ternary behaviour

NAND	0	1
0	1	1
1	1	0



NAND	0	$\frac{1}{2}$	1
0	1	1	1
$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$
1	1	$\frac{1}{2}$	0

Figure 1: Ternary Refinement of the NAND Gate.

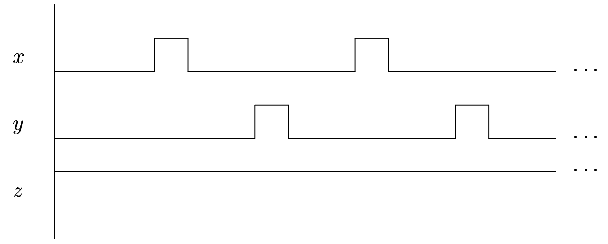
not as a refinement of the two-valued model, but conversely as an abstraction of the real-time behaviour of NAND. We expect that this ternary abstraction be compatible with the standard Boolean abstraction, which applies to the final stable state of a circuit. Thus, it would appear natural also to try to take the ternary function table as a statement about the final stationary state of the circuit. Such a link is suggested by the fact that in the stationary case there are exactly three possible behaviours for every binary signal: it can stabilize to 1, stabilize to 0, or oscillate. Formally, given a waveform $V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ the *stationary state* assumed by V may be defined as the mapping $V^\infty \in \mathbb{S} \to \mathbb{K}$, so that for all $a \in \mathbb{S}$,

$$V^{\infty}(a) = \begin{cases} 0 & \text{if } \exists t.\ \forall s.\ s \geq t \Rightarrow V(a)(s) = 0 \\ 1 & \text{if } \exists t.\ \forall s.\ s \geq t \Rightarrow V(a)(s) = 1 \\ \frac{1}{2} & \text{otherwise.} \end{cases}$$

In this reading the value $V^{\infty}(a) \in \mathbb{K}$ records the stabilization behaviour of signal a in V in such a way that $\frac{1}{2}$ denotes oscillation. If all signals in stabilize in V then V^{∞} captures the static binary behaviour in the usual sense. Now it is easy to interpret the ternary function table as a specification of the NAND's real-time behaviour: V is an execution of the NAND iff

$$V^{\infty}(z) = \text{NAND}(V^{\infty}(x), V^{\infty}(y)).$$

Though this is compatible with the static Boolean semantics, this way of going about is not the most useful way of explaining the meaning of the ternary function table. For it would imply that oscillating inputs necessarily produce an oscillation at the output of the NAND. This is rather too strict an account of the NAND's real-time behaviour as we may well have oscillating inputs but constant output, provided the inputs are interlaced in the right way:



Moreover, in some hardware structures such as dynamic memories the oscillation of a refresh signal is in fact a prerequisite for the memory signal to remain stable. Finally, the presence or absence of oscillation in general depends (among other things) on the relative differences between the propagation delays in different parts of a circuit. So, the attempt to use the ternary function tables to predict and to reason about oscillation algebraically would be quite unnatural. In some sense the difficulties stem from the wish to interpret $\frac{1}{2}$ as some specific signal behaviour. Even for loop-free circuits, or when oscillation is no issue, there does not seem to be a satisfactory real-time interpretation of $\frac{1}{2}$. See, *e.g.* the paper by Breuer [3] for a discussion of some of the problems.

A rather different way of interpreting $\mathbb{K}$ is not as a set of concrete signal behaviours but as a *domain of information*, in which $\frac{1}{2}$ stands for "unknown". The third value is given a special status and no longer on a par with 0 and 1. This is the original reading of Kleene [8], and it is implicit in most ternary simulation approaches like [5, 10, 4]. In this view the ternary table specifies a continuous function in the complete partial ordering $(\mathbb{K}, \sqsubseteq)$ with $\frac{1}{2} \sqsubseteq 1$, $\frac{1}{2} \sqsubseteq 0$. The ternary simulation of a binary network, then, corresponds to a least fixed-point computation. This domain-theoretic intepretation of the ternary function table, however, does not answer our question. It does not assign any concrete real-time operational meaning to the ternary NAND gate. What does the abstract fixed-point computation have to do with the real-time executions of the circuit? In this report we offer one possible connection using a real-time interpretation of ternary function tables, based on a notion of bounded stabilization.

3 The Ternary Model

To begin with, let us recall the basic elements of the standard ternary extension of binary gate modeling. Building on Kleene's three valued logic it was used originally by Yoeli, Rinon, Eichelberger, and Brzozowski [15, 6, 5] in order to analyse transient circuit behaviour. We follow the notation of these papers closely in this section.

Any boolean function $f \in \mathbb{B}^n \to \mathbb{B}$ may (for $n > 0$) be given a *ternary extension* $f^* \in \mathbb{K}^n \to \mathbb{K}$ as follows. First define a metric d_n on $\mathbb{K}^n$ by $d_n((u_1, \ldots, u_n), (v_1, \ldots, v_n)) := \max_i\{|u_i - v_i|\}$. Two states $\vec{u}$, $\vec{v}$ in $\mathbb{K}^n$ are called *adjacent* if $d_n(\vec{u}, \vec{v}) \leq \frac{1}{2}$. Also define a partial "averaging" function $\mu \in 2^{\mathbb{B}} \to \mathbb{K}$ given by $\mu\{0\} := 0$, $\mu\{1\} := 1$ and $\mu\{0,1\} := \frac{1}{2}$. Then

$$f^*(\vec{u}) \quad := \quad \mu\{f(\vec{v}) \mid \vec{v} \in \mathbb{B}^n \quad \& \quad d_n(\vec{u}, \vec{v}) \leq \tfrac{1}{2}\}.$$

Example *The ternary function table of the* NAND *gate in Fig. 1 is the graph of the ternary extension $f^* \in \mathbb{K}^2 \to \mathbb{K}$ of the Boolean function $f \in \mathbb{B}^2 \to \mathbb{B}$ with $f = \lambda(u, v).\, \overline{u \cdot v}$.* ∎

Now any network G of gates (a *binary gate network* in Yoeli and Brzozowski's terminology) with n primary inputs and $s \geq 1$ gates G_i has an associated Boolean transition function $F \in \mathbb{B}^{n+s} \to \mathbb{B}^s$. $F(\vec{u}, \vec{v})$ is the total gate state which arises from simultaneously replacing each gate output y_i with the appropriate Boolean function of its inputs. If $F(\vec{u}, \vec{v}) = \vec{v}$ then the configuration $(\vec{u}, \vec{v})$ is *stable*, and represents a possible steady state of the circuit, given that the inputs x are not changed. Note that the possibility of feedback means we must consider the gate outputs as inputs to the circuit in general. Then $F^* \in \mathbb{K}^{n+s} \to \mathbb{K}^s$ provides a ternary model of the behaviour of G. It can be used to analyse the combinational (as in [15, 10]) and the sequential behaviour of G (as in [6, 5].) In this report we are mainly interested in the combinational behaviour of circuits. We may paraphrase [10] in making a definition of "combinational" in terms of the ternary model F^*: A network G with associated transition function F is *combinational* for an input state $\vec{u} \in \mathbb{K}^n$ if

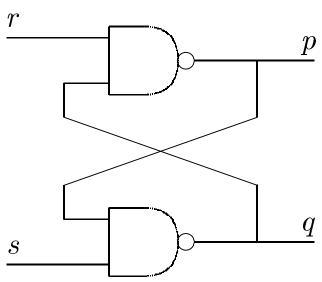
$$\mu.\, (\lambda \vec{w}.\, F^*(\vec{u}, \vec{w})) \quad \in \quad \mathbb{B},$$

where μ (from now on) denotes the operation of taking the least fixed point of the function $\lambda \vec{w}.\, F^*(\vec{u}, \vec{w}) \in \mathbb{K}^s \to \mathbb{K}^s$ over the complete partial ordering $(\mathbb{K}, \sqsubseteq)$. This least fixed point semantics is applied, implicitly, by Malik [10] and also by Bryant [4] at the transistor level. In the rest of this report, therefore, we will consider the ternary combinational behaviour of a circuit G to be defined by the function $\mu F^* \in \mathbb{K}^n \to \mathbb{K}^s$ given by

$$\mu F^*(\vec{u}) \quad = \quad \mu.\, (\lambda \vec{w}.\, F^*(\vec{u}, \vec{w})).$$

Intuitively, μF^* represents the input-output stabilization behaviour, *i.e.* the steady state response, of the circuit. It is this behaviour that we are going to tabulate in ternary function tables.

Example *Consider the RS-flipflop illustrated in Fig. 2. It has primary inputs* r, s *and outputs* p, q*. The associated Boolean function of this gate network is* $F \in \mathbb{B}^4 \to \mathbb{B}^2$ *with* $F = \lambda(r, s, p, q).\ (\overline{r \cdot q}, \overline{p \cdot s})$*. The ternary combinational behaviour* $\mu F^* \in \mathbb{K}^2 \to \mathbb{K}^2$ *is given by the ternary table in Fig. 2. From the table, three stable binary states may be read off, namely* $r = 0 \wedge s = 0 \wedge p = 1 \wedge q = 1$, $r = 0 \wedge s = 1 \wedge p = 1 \wedge q = 0$ *and* $r = 1 \wedge s = 0 \wedge p = 0 \wedge q = 1$*. The circuit is combinational (i.e, the outputs are Boolean and uniquely determined by the inputs) for the input states* $r = 0 \wedge s = 0$, $r = 0 \wedge s = 1$ *and* $r = 1 \wedge s = 0$*.* ∎



r	s	p	q
0	0	1	1
0	$\frac{1}{2}$	1	$\frac{1}{2}$
0	1	1	0
$\frac{1}{2}$	0	$\frac{1}{2}$	1
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$
1	0	0	1
1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
1	1	$\frac{1}{2}$	$\frac{1}{2}$

Figure 2: RS-flipflop.

4 A Language of Ternary Function Tables

To represent ternary function tables we wish introduce a simple and flexible formal language. There are several possible ways of concocting such a language, the basic choice being a programming language or a specification language. In our case both views actually coincide, but for reasons to become clear later, we have chosen to stress the logical aspect. The grammar of the language, which delineates a fragment of a propositional modal logic, is given by

$$
\begin{aligned}
\sigma &::= a = 1 \mid a = 0 \mid \sigma \wedge \sigma \\
\theta &::= \sigma \supset \bigcirc \sigma \mid \theta \wedge \theta,
\end{aligned}
$$

where a ranges over the set of signal names $\mathbb{S}$. When a is a signal name we will take $\mathbf{a}$ to stand for one of the atomic sentences, or *atoms*, $a = 1$ and $a = 0$. Elements of the syntactic class σ are called *(ternary) states*, those of form θ *(ternary) function tables*. As can be seen the general structure of a function table is

$$\theta \quad = \quad \bigwedge_{i=1}^{n} \sigma_i \supset \bigcirc \tau_i \quad = \quad \bigwedge_{i=1}^{n} (\bigwedge_{j=1}^{m_i} \mathbf{a}_i^j) \supset \bigcirc (\bigwedge_{k=1}^{l_i} \mathbf{b}_i^k)$$

where $n \geq 1$ and for all $i \leq n$, both $m_i \geq 1$ and $l_i \geq 1$. The components $\sigma_i \supset \bigcirc \tau_i$ we call the *transitions* of θ. When $n = 1$ then θ is a single transition. It will be convenient to identify a state $\sigma = \mathbf{a}_1 \wedge \cdots \wedge \mathbf{a}_m$ with the set $\{ \mathbf{a}_j \mid j \leq m \}$, since the ordering of atoms and the existence of duplications is not important.

We can represent every binary gate network by a function table in a canonical way. A binary gate G is given by a triple $G = (I, b, g)$ where $I = \{a_1, \ldots, a_m\} \subseteq \mathbb{S}$ is the set of input signals, $b \in \mathbb{S}$ the output signal, and $g \in \mathbb{B}^m \to \mathbb{B}$ a Boolean function. Let $g^* \in \mathbb{K}^m \to \mathbb{K}$ be the ternary extension of g. It is convenient, though not essential, to assume that g is not constant, whence we must have $g^*(\frac{1}{2}, \ldots, \frac{1}{2}) = \frac{1}{2}$. Then, the function table for G is formed as

$$\mathit{ft}(G) \quad = \bigwedge_{\vec{v} \in \mathbb{K}^m \;\&\; g^*(\vec{v}) \in \mathbb{B}} \Big(\bigwedge_{i \leq m \;\&\; v_i \in \mathbb{B}} a_i = v_i \Big) \supset \bigcirc (b = g^*(\vec{v})).$$

The ambiguity in this definition is not harmful since the actual permutation and bracketing of the conjuncts will be of no importance. If $G = G_1, \ldots, G_n$ is a network of binary gates we have the associated function table

$$\mathit{ft}(G) \quad = \quad \mathit{ft}(G_1) \wedge \cdots \wedge \mathit{ft}(G_n).$$

Our idea is that $\mathit{ft}(G)$ is a syntactic representation of the ternary behaviour of the network G. We will show in the next sections that this behaviour is determined by $\mathit{ft}(G)$ both in an operational and in an axiomatic sense. In other words, we may view $\mathit{ft}(G)$ both as a program and as a logic specification of real-time stabilization behaviour.

Example *Let us illustrate the syntactic presentation with our running example. Consider first the* NAND *gate* $N = (\{x, y\}, z, \lambda(u, v).\ \overline{u \cdot v})$ *of Fig. 1. The function table* $\mathit{ft}(N)$*, up to permutations, is the formula* $\textsc{nand}(x, y, z)$ *defined*

$$\begin{array}{rcccl}
\textsc{nand}(x,y,z) \ := & & & & \\
((x = 0 \wedge y = 0) \supset \bigcirc(z = 1)) & \wedge & (x = 0 \supset \bigcirc(z = 1)) & \wedge & ((x = 0 \wedge y = 1) \supset \bigcirc(z = 1)) \ \wedge \\
(y = 0 \supset \bigcirc(z = 1)) & \wedge & & & \\
((x = 1 \wedge y = 0) \supset \bigcirc(z = 1)) & \wedge & & & ((x = 1 \wedge y = 1) \supset \bigcirc(z = 0)).
\end{array}$$

Observe how this formula closely corresponds with the ternary function table of the NAND. *With* $\mathit{ft}(N)$ *at hand we form the function table for the RS-flipflop in Fig. 2 as*

$$\textsc{rs}(r, s, q, p) \quad := \quad \textsc{nand}(r, q, p) \wedge \textsc{nand}(p, s, q).$$

∎

5 Ternary Simulation As a Formal Calculus

The essence of traditional ternary simulation methods can be captured by proofs or derivations in a formal calculus on ternary function tables. We define a derivation relation $\theta_1 \vdash \theta_2$ between function tables to formalise an abstract understanding of ternary simulation. The calculus can be seen not only as a logical calculus but also as an operational semantics. From the logic point of view it essentially appears as a fragment of the sequent calculus for Propositional Lax Logic [7]. The relation $\vdash$ is the smallest relation closed under the rules shown in Fig. 3. In the rules $\supset\bigcirc L$ and $\bigcirc id$ it is to be understood that one or both of

$$\frac{}{\theta; \sigma_1 \wedge \mathbf{a} \wedge \sigma_2 \;\vdash\; \bigcirc\mathbf{a}}\bigcirc id \qquad \frac{\theta; \sigma \vdash \bigcirc\mathbf{a}_1 \quad \cdots \quad \theta; \sigma \vdash \bigcirc\mathbf{a}_m}{\theta; \sigma \vdash \bigcirc(\mathbf{a}_1 \wedge \cdots \wedge \mathbf{a}_m)}\bigcirc\!\wedge R$$

$$\frac{\theta_1 \;\wedge\; (\tau \supset \bigcirc(\sigma_1 \wedge \mathbf{b} \wedge \sigma_2)) \;\wedge\; \theta_2; \sigma \vdash \bigcirc\tau}{\theta_1 \;\wedge\; (\tau \supset \bigcirc(\sigma_1 \wedge \mathbf{b} \wedge \sigma_2)) \;\wedge\; \theta_2; \sigma \vdash \bigcirc\mathbf{b}}\supset\!\bigcirc L$$

$$\frac{\theta; \sigma \vdash \bigcirc(a = 1 \wedge a = 0)}{\theta; \sigma \vdash \bigcirc(b = v)}\bigcirc\mathrm{contr} \quad v \in \{0, 1\}$$

$$\frac{\theta; \sigma_1 \vdash \bigcirc\tau_1 \quad \cdots \quad \theta; \sigma_n \vdash \bigcirc\tau_n}{\theta \vdash \sigma_1 \supset \bigcirc\tau_1 \;\wedge\; \cdots \;\wedge\; \sigma_n \supset \bigcirc\tau_n}\supset\!\bigcirc R$$

Figure 3: A Sequent Calculus for Ternary Simulation.

the left and right side contexts σ_1, σ_2 may not be present, and similarly in the rule $\supset\bigcirc L$ the side contexts θ_1, θ_2 may be missing.

Strictly speaking, the system consists of two derivation relations. One is $\theta_1 \vdash \theta_2$ where both θ_1 and θ_2 are function tables. It is to represent the statement that function table θ_2 simulates table θ_1. In the calculus this relation is reduced to another relation $\theta; \sigma \vdash \bigcirc\tau$ where θ is a function table and σ, τ are states. This second relation is the actual simulation relation, stating that simulating the function table θ starting from initial state σ leads to the response τ, provided σ is maintained for long enough. The relationship between both relations is given by Prop. 5.1 below.

Proposition 5.1 *Let θ_1 and θ_2 be function tables. Then, $\theta_1 \vdash \theta_2$ iff, for all states σ and τ, $\theta_2; \sigma \vdash \bigcirc\tau$ implies $\theta_1; \sigma \vdash \bigcirc\tau$.*

Proposition 5.1 can be rephrased as saying that $\theta_1 \vdash \theta_2$ iff θ_1 can be used to simulate θ_2, *i.e.* every input-output response of θ_2 is also a response of θ_1. In this case we say that θ_1 *simulates* θ_2. When both $\theta_1 \vdash \theta_2$ and $\theta_2 \vdash \theta_1$ then θ_1 and θ_2 are said to be (simulation) *equivalent.*

Example *The* NAND*-gate's function table is equivalent to the simpler table*

$$\mathrm{NAND}'(x, y, z) \;:=$$

$$(x = 0 \supset \bigcirc(z = 1)) \;\wedge\; (y = 0 \supset \bigcirc(z = 1)) \;\wedge\; ((x = 1 \wedge y = 1) \supset \bigcirc(z = 0)).$$

Regarding the direction $\text{NAND}'(x, y, z) \vdash \text{NAND}(x, y, z)$ *one has to verify the six sequents*

$$\begin{array}{llcllcl}
\text{NAND}'(x,y,z) & ; & x = 0 & \wedge & y = 0 & \vdash \bigcirc z = 1 & (i) \\
\text{NAND}'(x,y,z) & ; & x = 0 & & & \vdash \bigcirc z = 1 & (ii) \\
\text{NAND}'(x,y,z) & ; & x = 0 & \wedge & y = 1 & \vdash \bigcirc z = 1 & (iii) \\
\text{NAND}'(x,y,z) & ; & & & y = 0 & \vdash \bigcirc z = 1 & (iv) \\
\text{NAND}'(x,y,z) & ; & x = 1 & \wedge & y = 0 & \vdash \bigcirc z = 1 & (v) \\
\text{NAND}'(x,y,z) & ; & x = 1 & \wedge & y = 1 & \vdash \bigcirc z = 0 & (vi)
\end{array}$$

Each sequent has a short proof, e.g. for (vi) we have

$$\dfrac{\dfrac{\dfrac{}{\text{NAND}'(x,y,z);\sigma \vdash \bigcirc(x=1)}\,\bigcirc id \qquad \dfrac{}{\text{NAND}'(x,y,z);\sigma \vdash \bigcirc(y=1)}\,\bigcirc id}{\text{NAND}'(x,y,z);\sigma \vdash \bigcirc(x=1 \wedge y=1)}\,\bigcirc\!\wedge R}{\text{NAND}'(x,y,z);\sigma \vdash \bigcirc(z=0)}\,\supset\!\bigcirc L$$

where $\sigma \equiv x = 1 \wedge y = 1$. *The other direction* $\text{NAND}(x, y, z) \vdash \text{NAND}'(x, y, z)$ *is also easy to establish.* ∎

It will be useful to have some shorthand for derivations in our simulation calculus. Derivation terms are just linear notations for derivation trees with multiple conclusions. They are generated by the following simple grammar:

$$\begin{array}{rcl}
\mathit{Deduction} & ::= & \mathit{Rule} \cdot \mathit{Deduction} \mid (\mathit{Deduction}, \mathit{Deduction}) \\
\mathit{Rule} & ::= & \mathit{id}(i) \mid \bigcirc\!\wedge R \mid \supset\!\bigcirc R \mid \supset\!\bigcirc L(i,j) \mid \bigcirc\text{contr},
\end{array}$$

where i in $id(i)$ is a natural number recording the index of the atomic sentence $\mathbf{a}$ picked out by an application of the rule $\bigcirc id$. In $\supset\bigcirc L(i, j)$, i is the index of the table entry $\tau \supset \bigcirc(\sigma_1 \wedge \mathbf{b} \wedge \sigma_2)$ and j the index of $\mathbf{b}$ within the state $\sigma_1 \wedge \mathbf{b} \wedge \sigma_2$. If D is a derivation of $\theta_1 \vdash \theta_2$ we will denote this by $D : \theta_1 \vdash \theta_2$. Similarly, we write $D : \theta; \sigma \vdash \bigcirc\tau$.

Example *Consider the RS-flipflop of Fig. 2 with function table* $\text{RS}'(r, s, q, p) = \text{NAND}'(r, q, p) \wedge \text{NAND}'(p, s, q)$. *The output response* $p = 1 \wedge q = 0$ *for the input state* $\sigma(01) := r = 0 \wedge s = 1$ *is exhibited by a proof that* $D : \text{RS}'(r, s, p, q); \sigma(01) \vdash \bigcirc(p = 1 \wedge q = 0)$ *with* $D = \bigcirc\wedge R \cdot (D_1, D_2)$, *where the subproofs* D_1 *and* D_2 *are constructed as follows:*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\text{RS}'(r,s,p,q);\sigma(01) \vdash \bigcirc(r=0)}\,\bigcirc id}{\mathbf{D_1} : \text{RS}'(r,s,p,q);\sigma(01) \vdash \bigcirc(p=1)}\,\supset\!\bigcirc L \qquad \dfrac{}{\text{RS}'(r,s,p,q);\sigma(01) \vdash \bigcirc(s=1)}\,\bigcirc id}{\text{RS}'(r,s,p,q);\sigma(01) \vdash \bigcirc(p=1 \wedge s=1)}\,\bigcirc\!\wedge R}{\mathbf{D_2} : \text{RS}'(r,s,p,q);\sigma(01) \vdash \bigcirc(q=0)}\,\supset\!\bigcirc L$$

We may read off the proof terms $D_1 = \supset\bigcirc L(1, 1) \cdot \bigcirc id(1)$ *and* $D_2 = \supset\bigcirc L(6, 1) \cdot \bigcirc\wedge R \cdot (\supset\bigcirc L(1, 1) \cdot \bigcirc id(1), \bigcirc id(2))$. ∎

Before we elaborate on the connection with the algebraic approach to ternary simulation let us mention two fundamental properties of the simulation calculus.

Proposition 5.2 *The relation $\vdash$ is transitive, i.e. if $\theta; \sigma_1 \vdash \bigcirc\sigma_2$ and $\theta; \sigma_2 \vdash \bigcirc\sigma_3$, then $\theta; \sigma_1 \vdash \bigcirc\sigma_3$. Furthermore, if $\sigma \subseteq \sigma'$ as sets of atoms, then $\theta; \sigma \vdash \bigcirc\tau$ implies $\theta; \sigma' \vdash \bigcirc\tau$.*

Proof: By case analysis and induction on the structure of derivations. The first part of the proposition corresponds to cut-elimination in logic, and the second part to the rules of weakening, permutation, and contraction. ∎

Proposition 5.3 *Let $\theta = \theta_1 \wedge \cdots \wedge \theta_n$, and for each $i \leq n$, $\theta_i = \tau_i \supset \bigcirc(\mathbf{b}_i^1 \wedge \cdots \mathbf{b}_i^{m_i})$. Further let σ and τ be states such that $\theta; \sigma \vdash \bigcirc\tau$. Then, there exists a derivation D : $\theta; \sigma \vdash \bigcirc\tau$ such for every $i \leq n$ and $j \leq m_i$, D does not contain more than one application of $\supset\bigcirc L(i,j)$ in every branch. That is, there are no nested applications of $\supset\bigcirc L$ with the same indices.*

Proposition 5.3 identifies a very useful normalization property, which relates to the well-known fact that in ternary simulation a single traversal through all paths is sufficient to compute the (combinational) response for a single input stimulus. As an immediate consequence we can weaken the rule $\supset\bigcirc L$ (*cf.* Fig. 3) by dropping the atom $\mathbf{b}$ in the premiss, *i.e.* replace $\supset\bigcirc L$ by the rule

$$\frac{\theta_1 \ \wedge \ (\tau \supset \bigcirc(\sigma_1 \wedge \sigma_2)) \ \wedge \ \theta_2; \sigma \vdash \bigcirc\tau}{\theta_1 \ \wedge \ (\tau \supset \bigcirc(\sigma_1 \wedge \mathbf{b} \wedge \sigma_2)) \ \wedge \ \theta_2; \sigma \vdash \bigcirc\mathbf{b}} \supset\bigcirc L^*$$

to obtain an equivalent calculus. This new calculus has the property that for every end sequent there are only a *finite* number of *finite* derivations, if there are any at all. Thus, we could mechanize the calculus directly in Prolog to obtain a naive but executable implementation.

5.1 Ternary Simulation

Let us see how our calculus provides an operational semantics of the ternary behaviour of binary gate networks. We first consider a single gate $G = (I, b, g)$ with $I = \{a_1, \ldots, a_m\}$ and $b \notin I$, with the associated function table $ft(G)$. How do we simulate G, on the basis of its description $ft(G)$, say for a given ternary input pattern $\vec{v} = (v_1, \ldots, v_m) \in \mathbb{K}^m$ (in which not all entries are $\frac{1}{2}$)? We form the ternary state

$$\sigma(\vec{v}) \ := \ \bigwedge_{i \leq m \ \& \ v_i \in \mathbb{B}} a_i = v_i$$

and compute the output value by looking at what we can derive in the calculus about the output signal b. Thus, we are interested in the set

$$R(G, \vec{v}) \ := \ \{ \mathbf{b} \mid ft(G); \sigma(\vec{v}) \vdash \bigcirc(\mathbf{b}) \},$$

that we will abbreviate temporarily by $R(\vec{v})$. There are three possibilities:

- $R(\vec{v}) = \{b = 1\}$. In this case the resulting output value for b is 1, *i.e.* $g^*(\vec{v}) = 1$.

- $R(\vec{v}) = \{b = 0\}$. This is the dual case indicating that the output value for b is 0, *i.e.* $g^*(\vec{v}) = 0$.

- $R(\vec{v}) = \emptyset$. This means that nothing definite can be inferred about the output value, corresponding to $g^*(\vec{v}) = \frac{1}{2}$.

One can show that the case $R(\vec{v}) = \{b = 1, b = 0\}$ is excluded because $b \notin I$ and the fact that $ft(G)$ does not contain conflicting transitions for b. We can sum up the situation as follows:

Proposition 5.4 *Let $G = (I, b, g)$ be a single gate with $I = \{a_1, \ldots, a_m\}$, $b \notin I$, and g not constant. Then for all $\vec{v} \in \mathbb{K}^m$ and $e \in \mathbb{B}$, $g^*(\vec{v}) = e$ iff $b = e \in R(G, \vec{v})$.*

Hence, the ternary extension g^* of g can be determined completely by our ternary calculus. Note, however, that the "third" value $\frac{1}{2}$ now is represented indirectly: it stands for the absence of information. Formally, we have $g^*(\vec{v}) = \frac{1}{2}$ iff $ft(G); \sigma(\vec{v}) \not\vdash \bigcirc(b = 1)$ and $ft(G); \sigma(\vec{v}) \not\vdash \bigcirc(b = 0)$.
The relationship between $ft(G)$ and g^* generalizes to arbitrary binary networks without feedback. Here $ft(G)$ is the conjunction of the gates' function tables and g the functional composition of the gates' Boolean functions. In this way we are recovering essentially the ternary combinational analysis of Yoeli and Rinon [15] and Eichelberger [6] in a non-algebraic setting.

The situation becomes more complicated when feedback is allowed. Consider again a single gate $G = (I, b, g)$, but this time we allow $b \in I$, say $I = \{a_1, \ldots, a_m, b\}$. If the goal is to capture the behaviour of G as a combinational device, then we view G as a function from the *primary inputs* $\{a_1, \ldots, a_m\}$ to *output* signal b, *i.e.* as a function $G^* \in \mathbb{K}^m \to \mathbb{K}$. As mentioned before this function is obtained by taking

$$\mu G^*(\vec{v}) \quad = \quad \mu.\, (\lambda w.\, g^*(\vec{v}, w)),$$

by fixed point construction. Again, we may recover μG^* in our calculus from $R(G, \vec{v})$ defined exactly as above.

Proposition 5.5 *Let $G = (I, b, g)$ be a single gate with $I = \{a_1, \ldots, a_m, b\}$, and g not constant. Then for all $\vec{v} \in \mathbb{K}^m$ and $e \in \mathbb{B}$, $\mu G^*(\vec{v}) = e$ iff $b = e \in R(G, \vec{v})$.*

The proposition generalizes to any binary network $G = G_1, \ldots, G_n$, where $\{a_1, \ldots, a_m\}$ is the set of all its primary inputs and b one of its gate outputs. In this manner we may recover the essence of Malik's ternary combinational analysis in a non-algebraic setting. For instance, we can give a characterization of Malik's notion of "combinational" circuits as follows:

Proposition 5.6 *A binary gate network $G = G_1, \ldots, G_n$ with primary inputs $\{a_1, \ldots, a_m\}$ is combinational for a given output b, in the sense of Malik [10], iff for all binary input states $\vec{v} \in \mathbb{B}^m$, either $b = 1 \in R(G, \vec{v})$ or $b = 0 \in R(G, \vec{v})$.*

In other words, G is combinational for output b if for every Boolean assignment to its primary inputs our simulation calculus allows us to derive a definite Boolean response for b.

The definition of R is based on derivability, and because of the finiteness of our calculus it can be computed effectively in a finite number of steps. Obviously, there are many ways of actually performing this computation. For instance, we could simply enumerate all derivations in a systematic way. A more sophisticated solution is given by an iterative process as follows: Suppose that $\mathit{ft}(G) = \bigwedge_{i=1}^{n}(\bigwedge_{j=1}^{m_i} \mathbf{a}_i^j) \supset \mathbf{b}_i$ is the function table of the circuit and $a_1, \ldots, a_l$ the primary inputs. An input bit-pattern assigning the value $v_i \in \mathbb{B}$ to input a_i, $i \leq l$ can be represented by the set of atoms

$$I \quad = \quad \{\, a_i = v_i \mid i \leq l \,\}.$$

The stationary response $R(G, \vec{v})$ of the circuit for input stimulus I can be computed iteratively by following Lloyd [9] in constructing the least Herbrand model of $\mathit{ft}(G)$:

$$\begin{aligned} R_0 &= I \\ R_{k+1} &= \{\, \mathbf{b}_i \mid i \leq n \ \&\ \forall j \leq m_i.\, \mathbf{a}_i^j \in R_k \,\}. \end{aligned}$$

We define $R(G, \vec{v}) = \bigcup_k R_k$. Obviously, the sequence R_k must become stationary, so that R can be computed in a finite number of steps. Note however that the delay information embedded in the full calculus is not recorded in this bottom-up construction of R, although it would not be difficult to amend the construction to record this information.

An even more sophisticated method would be not to construct $R(G, \vec{v})$ point-wise for every $\vec{v}$ but to compute the function $R(G) : \vec{v} \mapsto R(G, \vec{v})$ as a whole. This might be done by symbolic means, using BDDs and in a dual-rail coding of the ternary information involved. This solution is applied by the ternary simulation approaches of Malik [10] or Bryant [4].

6 Bounded Stabilization: A Real-Time Semantics for Ternary Function Tables

In the previous section ternary simulation of binary networks was viewed as an abstract calculus manipulating formal syntax. In practice a simulation algorithm may be represent the ternary values concretely, say by double rail binary encoding and perform the simulation by algebraic operations, say by BDD methods. Yet, even though the simulation might seem to handle concrete numbers, or sets of concrete numbers, rather than formal syntax, we are still lacking a precise connection to the real circuit. What is the real-time information we are handling in the ternary world? We will now conquer new land and establish such a connection. We will show how the ternary calculus arises in a very natural way by abstracting from bounded propagation delays.

Let $\sigma = a_1 = v_1 \wedge \cdots \wedge a_m = v_m$ be a state and $V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ a waveform. We say that V *assumes* σ in a given time interval $[s, t]$, written $V[s, t] \models \sigma$, if the signal a_i has the stable value v_i throughout the interval $[s, t]$. Formally,

$$V[s,t] \models \sigma \quad \text{iff} \quad \forall i.\, \forall r.\, s \leq r \leq t \Rightarrow V(a_i)(r) = v_i.$$

We allow the interval to be empty, *i.e.* $s \geq t$, in which case the condition becomes trivially true. In other words: In the empty interval V assumes every state. This pathological case could be eliminated but it makes matters simpler if we include it. We observe that the interpretation of σ interprets the syntactic construct $\wedge$ as logical conjunction: $V[s,t] \models \mathbf{a}_1 \wedge \mathbf{a}_2$ iff $V[s,t] \models \mathbf{a}_1$ and $V[s,t] \models \mathbf{a}_2$.

Example *Consider the timing diagram seen in Fig. 4, specifying a particular set of waveforms V operating the RS-flipflop, where the shaded areas indicate unconstrained behaviour. The diagram states that such a V contains 8 adjacent subintervals $[t_i, t_{i+1}]$, $i = 0, \ldots, 7$, in which it assumes particular states. For instance, it specifies*

$$V[t_1,t_4] \models r = 0 \wedge s = 1 \qquad V[t_3,t_4] \models r = 0 \wedge s = 1 \wedge p = 1 \wedge q = 0.$$

∎

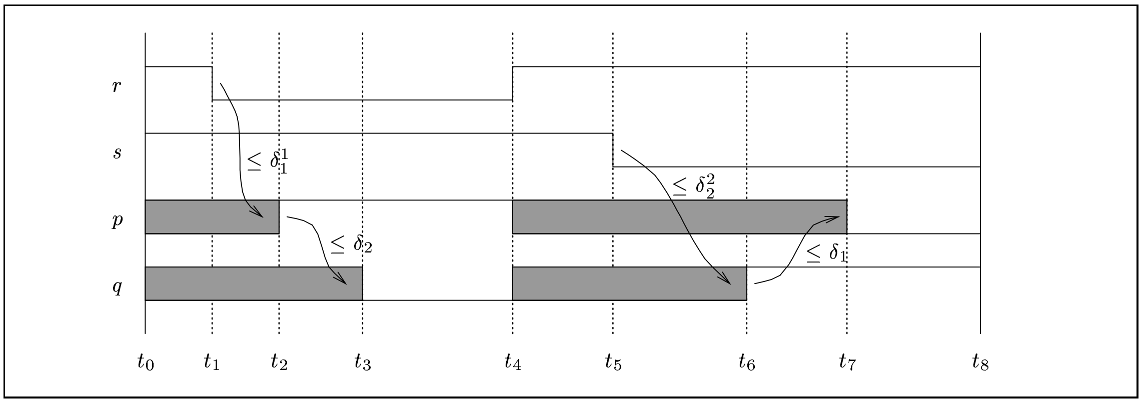


Figure 4: Some Waveforms Operating the RS-flipflop.

Now let us interpret a single transition as a bounded response constraint on a waveform V. We say that V satisfies the transition $\sigma \supset \bigcirc\tau$ with *stabilization bound* $\delta \in \mathbb{N}$, written $V \models_\delta \sigma \supset \bigcirc\tau$, if whenever V assumes state σ then it also will assume τ with a maximum delay δ. Formally,

$$V \models_\delta \sigma \supset \bigcirc\tau \quad \text{iff} \quad \forall s,t \in \mathbb{N}.\ V[s,t] \models \sigma \Rightarrow V[s+\delta,t] \models \tau.$$

We can read this more operationally. It says that whenever V enters σ at some point in time s and remains in this state long enough, then eventually, but no later that time $s+\delta$, state τ is entered too; and furthermore V remains in state τ as long as long as σ remains stable. This is exactly the behaviour one expects of a combinational circuit where σ refers to the input and τ to the output.

Example *With transition formulas we can specify the upper bounds indicated in the timing diagram of Fig. 4. The statement that a falling transition on r is followed by a rising transition on p with a maximal delay of δ_1^1, and that p must hold 1 at least as long as r remains 0, can be expressed as*

$$V \models_{\delta_1^1} r = 0 \supset \bigcirc(p = 1).$$

Similarly, the property that when both signals s and p become stable 1, signal q must fall to 0 with a maximal delay of δ_2, and thereafter keep its value as long as s and p do so, is expressed as

$$V \models_{\delta_2} (s = 1 \land p = 1) \supset \bigcirc(q = 0).$$

∎

We say that V *satisfies* $\sigma \supset \bigcirc\tau$, written $V \models \sigma \supset \bigcirc\tau$, if there exists a δ such that $V \models_\delta \sigma \supset \bigcirc\tau$. With the existential quantification the concrete stabilization delay, or propagation delay, is abstracted away completely, so that $\sigma \supset \bigcirc\tau$ becomes a purely qualitative specification. Though the quantitative aspect of the delay appears to be lost for the time being, we will see that it can be recovered in an exact way. This is possible because we are interested not only in extensional validity, *viz.* the mere fact *that* a waveform V satisfies a transition $\sigma \supset \bigcirc\tau$, but also in the intensional sense, *viz.* just *how* V satisfies $\sigma \supset \bigcirc\tau$. This "how" is given by the smalles δ such that $V \models_\delta \sigma \supset \bigcirc\tau$.

Example *Consider again the waveforms of Fig. 4, where the t_i now are assumed to be fixed points in time with $t_0 = 0$. We also assume that t_8 represents ∞, i.e. that the last interval is $[t_7, \infty)$. Then, $\delta \geq t_3 - t_1$ iff for all waveforms in the specified set we have $V \models_\delta r = 0 \supset \bigcirc(q = 0)$. Or, for any particular V, $V \models_\delta r = 1 \supset \bigcirc(q = 0)$ iff $\delta \geq max\{t_1 - t_0, t_5 - t_4\}$.* ∎

Finally, let

$$\theta \;=\; \bigwedge_{i=1}^{n} \sigma_i \supset \bigcirc\tau_i$$

be a function table. Then, V satisfies θ, written $V \models \theta$, iff for all $i = 1, \ldots, n$, $V \models \sigma_i \supset \bigcirc\tau_i$.

Example *Consider the upper* NAND *gate in Fig. 2 specified by the formula*

$$\mathrm{NAND}(r, q, p) \;=\; r = 0 \supset \bigcirc p = 1 \land q = 0 \supset \bigcirc p = 1 \land (r = 1 \land q = 1) \supset \bigcirc p = 0.$$

$V \models \mathrm{NAND}(r, q, p)$ means that there exist delay parameters $\delta_1^1, \delta_1^2, \delta_1 \in \mathbb{N}$ such that the timing diagrams shown in Fig. 5 are satisfied by V. A similar interpretation applies to the lower $\mathrm{NAND}(p, s, q)$ of Fig. 2, say with corresponding abstracted delays $\delta_2^1, \delta_2^2, \delta_2 \in \mathbb{N}$. If we combine both to the function table $\mathrm{RS}(r, s, p, q) = \mathrm{NAND}(r, q, p) \land \mathrm{NAND}(p, s, q)$ of the RS-flipflop we obtain a real time specification that contains the timing diagram of Fig. 4.

It is important to realize that $\mathrm{RS}(r, s, p, q)$ by itself only covers the combinational behaviour of the RS-flipflop. The sequential behaviour, viz. the fact that state can be stored, depends on assuming lower bounds on delays, and in particular nonzero inertiality. Suppose the interval $[t_3, t_4]$ in Fig. 4 is long enough for the state $s = 1 \land p = 1 \land q = 0$ to propagate through both NAND *gates. Then, under nonzero inertiality this state stabilizes itself and remains stable throughout the interval $[t_4, t_5]$, even though the "reset" signal r returns back to the idle state $r = 1$ at time t_4. This memory effect is not visible in $\mathrm{RS}(r, s, p, q)$ which explains the shaded areas in $[t_4, t_5]$. The combinational view allows for arbitrary behaviour on the output p as soon as input r returns to 1 and gives up functional control of p.*

However, as was shown by Eichelberger [6] and Brzozowski and Yoeli [14], the sequential behaviour can be recovered, to a certain extent, from the (ternary) combinational behaviour. One introduces the notion of stable state and simulates the combinational response from one stable state to the next. The details are beyond the scope of this report. The situation is somewhat analogous to the modeling of sequential circuits, which also can be described completely in terms of their combinational state transition function, assuming no state variables are hidden. ∎

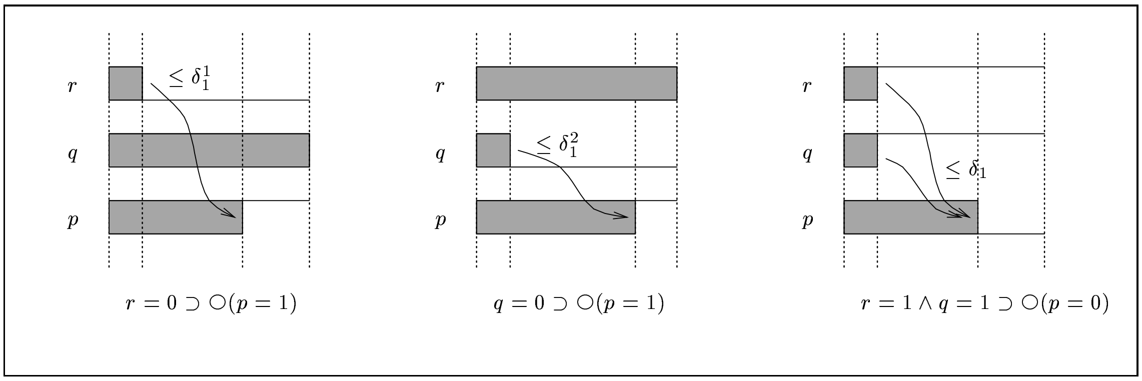


Figure 5: Stabilization Behaviour of NAND(r, q, p).

When σ refers to the input state and τ to the output state of a circuit then $\sigma \supset \bigcirc\tau$ would capture a particular aspect of the combinational input-output behaviour of the circuit. When feedback is present, however, σ and τ might refer to the same signals, and then $\sigma \supset \bigcirc\tau$ specifies a slightly more general form of stabilization behaviour, *viz.* transients. Consider the transition $a = 1 \supset \bigcirc(a = 0)$. Unrolling the semantic definition we find that $V \models_\delta a = 1 \supset \bigcirc(a = 0)$ iff V only ever assumes $a = 1$ for at most δ time steps, *i.e.* iff the 1 phases of a are transient with upper bound δ. Thus,

$$V \models a = 1 \supset \bigcirc(a = 0) \ \wedge \ a = 0 \supset \bigcirc(a = 1)$$

expressing that V exhibits bounded oscillation as seen in Fig. 6, for some δ_1, δ_2.

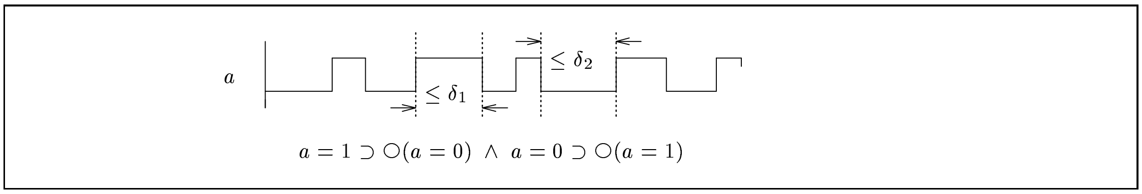


Figure 6: Bounded Oscillation.

If $G = G_1, \ldots, G_n$ is a binary network of gates and $ft(G) = ft(G_1) \wedge \cdots ft(G_n)$ the associated function table, then

$$[\![ft(G)]\!] \quad = \quad \{\, V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B} \mid V \models ft(G) \,\}$$

is our *real-time semantics* for G. It specifies the set of waveforms that we consider valid executions, or observations, of the network. It is a real-time model essentially for the combinational behaviour of arbitrary binary networks.

There are many other options to assign a real-time semantics to binary networks, more optimistic as well as more conservative ones. The main choice lies in the delay model. A more conservative approach, for instance, might allow for unbounded propagation delays. Take the NAND gate seen in Fig. 1 again. Considering an input state $x = 0$, our interpretation of $x = 0 \supset \bigcirc(z = 1)$ yields the statement

$$\exists \delta.\, \forall s, t.\, V[s,t) \models x = 0 \Rightarrow V[s+\delta, t] \models z = 1.$$

A weaker way of abstracting from the delay would be to stipulate

$$\forall s.\, \exists \delta.\, \forall t.\, V[s,t) \models x = 0 \Rightarrow V[s+\delta, t) \models z = 1,$$

i.e. we swap the quantifiers $\exists\delta$ and $\forall s$. In this case the delay δ is unbounded and may depend on the time s when the input changes. In normal circumstances this is not what we want since we want the delay to be a property of the NAND gate not of a particular use of it. In other words, the NAND gate's input x may change arbitrarily many times within a waveform, but the maximum propagation delay is fixed throughout the whole execution. However, if the ambition or need was to be faithful with subtle physical phenomena such as the metastable operation of circuits [11, 13] this weaker delay model might be more adequate. Another direction for being more conservative would have been to ignore the data and input dependency of delays and to assume only a single delay for every gate. Also, we could have enforced the stability of non-controlling inputs. Taking the weakest version the NAND gate then might look like

$$\forall s.\, \exists \delta.\, \forall t.\, \forall v, w.\;\; V[s,t) \models x = v \wedge y = w \;\;\Rightarrow\;\; V[s+\delta, t) \models z = \overline{v \cdot w}.$$

In this case the delay is unbounded, does not depend on input data, and in every observation interval all inputs are required to be stable, regardless whether they functionally determine the output value or not.

In the other direction one could also strengthen the delay model to arrive at more powerful descriptions that, for instance, also capture the sequential behaviour of circuits. This could be achieved by introducing inertial delays, which specify how long an output is guaranteed to hold its value after the input has changed, or for how long an input must at least persist in order for the output to respond. Real-time gate models with inertiality can be found in the work of [2] for instance. An example of a rather strong real-time model is the inertial delay model of Brzozowski and Yoeli [14]. In their model a gate has unbounded propagation delay but infinite inertial delay. This means that if a gate is excited through an input transition the output will follow eventually (unboundedly,) but does not change at all if the input returns to the stable position before.

In a sense, this discussion points to fact that there are a lot of choices to pick a suitable real-time model of binary circuits. Our aim is to show that among these choices the semantics presented in this report holds a distinguished position. We show that

- $\models$ is a natural intuitionistic Kripke semantics for propositional modal logic (Sec. 8).
- $\models$ is equivalent to the traditional ternary model, as formalized by $\vdash$, both in an extensional and in an intensional sense (Sec. 7).

The first result means that the language of function tables can be enriched naturally into a full-fledged propositional modal logic, which leads us beyond the ternary model. The second result implies that we can obtain exact timing information from derivations in the logic.

7 Soundness and Completeness, Intensionally

As in ordinary logic we can define a semantical consequence relation between function tables, $\theta_1 \models \theta_2$, abbreviating the condition $\forall V.\, V \models \theta_1 \Rightarrow V \models \theta_2$, *i.e.* the condition that every model of θ_1 is a model also of θ_2. In this section we will make explicit the intensional contents of the operational and axiomatic semantics of ternary function tables, and prove the equivalence

$$\theta_1 \models \theta_2 \quad \Leftrightarrow \quad \theta_1 \vdash \theta_2$$

the strong sense, showing that the underlying quantitative delay information is preserved in both directions as well. But before we can state the theorems we need to uncover the intensional contents of $\models$ and $\vdash$.

Let us assume, throughout this section, that $\theta_1 = \bigwedge_{i=1}^{m} \sigma_1^i \supset \bigcirc\tau_1^i$ and $\theta_2 = \bigwedge_{i=1}^{n} \sigma_2^i \supset \bigcirc\tau_2^i$. We re-introduce the stabilization information into θ_1 and θ_2. For $\vec{\delta}_1 \in \mathbb{N}^m$ we form the "timed" statement $\vec{\delta}_1 : \theta_1$ that relativizes θ_1 to $\vec{\delta}_1$. We define $V \models \vec{\delta}_1 : \theta_1$ iff $V \models_{\vec{\delta}_1} \theta_1$. The same convention we introduce for θ_2. Of course, this is no more than a syntactic convention reflecting a shift of stress: We consider $\vec{\delta}$ not as part of the validity of θ but as part of the formula. In this spirit, we define

$$\vec{\delta}_1 : \theta_1 \models \vec{\delta}_2 : \theta_2 \quad \text{iff} \quad \forall V.\, V \models_{\vec{\delta}_1} \theta_1 \;\Rightarrow\; V \models_{\vec{\delta}_2} \theta_2.$$

Now, on the other side, what is the quantitative delay information hidden in a derivation $\theta_1 \vdash \theta_2$? It turns out that, quite naturally, we can translate every derivation $D : \theta_1 \vdash \theta_2$ into a delay function $[\![D]\!] \in \mathbb{N}^m \to \mathbb{N}^n$ as follows:

$$\begin{aligned}
[\![\supset\!\bigcirc R \cdot (D_1, \ldots, D_n)]\!](\vec{\delta}) &= ([\![D_1]\!](\vec{\delta}), \ldots, [\![D_n]\!](\vec{\delta})) \\
[\![id(i)]\!](\vec{\delta}) &= 0 \\
[\![\supset\!\bigcirc L(i,j) \cdot D]\!](\vec{\delta}) &= [\![D]\!](\vec{\delta}) + \delta_i \\
[\![\bigcirc\!\wedge R \cdot (D_1, \ldots, D_m)]\!](\vec{\delta}) &= max\{[\![D_1]\!](\vec{\delta}), \ldots, [\![D_m]\!](\vec{\delta})\} \\
[\![\bigcirc\text{contr} \cdot D]\!](\vec{\delta}) &= [\![D]\!](\vec{\delta}).
\end{aligned}$$

Based on this extracted delay function we can enrich the syntactic simulation calculus by explicit delay information:

$$\vec{\delta}_1 : \theta_1 \vdash \vec{\delta}_2 : \theta_2 \quad = \quad \exists D.\, D : \theta_1 \vdash \theta_2 \;\wedge\; [\![D]\!](\vec{\delta}_1) \leq \vec{\delta}_2,$$

where $\leq$ is defined component-wise.

Our soundness and completeness theorems state that the delay-enriched versions of $\vdash$ and $\models$ are equivalent. More precisely, if $\mathbb{N}_0 = \mathbb{N} \setminus \{0\}$ is the set of positive numbers, then for all $\vec{\delta}_1 \in \mathbb{N}_0^m$ and $\vec{\delta}_2 \in \mathbb{N}_0^n$,

$$\vec{\delta}_1 : \theta_1 \models \vec{\delta}_2 : \theta_2 \quad \Leftrightarrow \quad \vec{\delta}_1 : \theta_1 \vdash \vec{\delta}_2 : \theta_2.$$

These intensional versions are of great importance from a practical point of view. They can be seen as quantitative soundness and quantitative completeness of the ternary calculus. Let θ_1 describe the implementation of a circuit in terms of a set of primitive components, and θ_2 a specification for the composite circuit. Then the soundness direction $\Leftarrow$ implies that from every D verifying that the circuit θ_1 simulates specification θ_2 we can extract a delay function $[\![D]\!]$ that translates the delay bounds $\vec{\delta}_1$ of the primitive components of θ_1 into delay bounds $[\![D]\!](\vec{\delta}_1)$ for the composite circuit θ_2. Formally, for all $\vec{\delta}_1$, $\vec{\delta}_1 : \theta_1 \models [\![D]\!](\vec{\delta}_1) : \theta_2$. The completeness direction $\Rightarrow$ in turn implies that, if we consider all possible derivations, then we can actually get the optimal delays for the composite circuit. Formally, for every $\vec{\delta}_1$, for which there exists at all a $\vec{\delta}_2$ satisfying $\vec{\delta}_1 : \theta_1 \models \vec{\delta}_2 : \theta_2$ we actually can find a derivation D such that $[\![D]\!](\vec{\delta}_1)$ is the minimal $\vec{\delta}_2$ with this property. So, in order to get the minimal delay we can compute all proofs and take the minimum of the extracted delays. This is always possible since there are only a finite number of different derivations[1].

At the heart $[\![\_]\!]$ is local translation of our derivation rules as arithmetical operations on delays, *viz.*

$$\bigcirc id \mapsto 0 \qquad\qquad \bigcirc\wedge R \mapsto max \qquad\qquad \supset\bigcirc L \mapsto +,$$

which means that the bounds are terms in the *max-plus delay algebra* $(\mathbb{N}, 0, max, +)$, see [1]. This observation has been worked out in more detail in [12], about the timing extraction from constructive proofs. It uses the general setting of Propositional Lax Logic [7], but with a simpler real-time semantics.

We should remark that completeness, in general, need not hold for delays $\vec{\delta}_1$ in which some or all components take the value 0. For instance, if $\theta_1 = a = 1 \supset \bigcirc(a = 0) \ \wedge \ a = 0 \supset \bigcirc(a = 1)$ and $\vec{\delta}_1 = (0, 0)$, then $\vec{\delta} : \theta_1$ amounts to an inconsistent specification. There is no waveform V such that $V \models_{\vec{\delta}} \theta_1$. This means that $\vec{\delta}_1 : \theta_1 \models \vec{\delta}_2 : \theta_2$ trivially holds for all $\vec{\delta}_2$ and θ_2. But there is no sound extension of the calculus in which there would be a derivation $D : \theta_1 \vdash \theta_2$ for arbitrary θ_2. In other words: with zero delays in $\vec{\delta}_1$, $\vec{\delta}_1 : \theta_1$ may be a singular delay-*dependent* inconsistency that cannot be detected by the calculus, which is supposed to handle only delay-*independent* functional information.

Let us observe first that the standard extensional soundness and completeness theorems follow from the intensional ones.

Theorem 7.1 (Extensional Soundness and Completeness) *$\theta_1 \vdash \theta_2$ iff $\theta_1 \models \theta_2$.*

[1] In the equivalent system with rule $\supset\bigcirc L^*$.

Proof: Given $\theta_1 \vdash \theta_2$, and let D be a derivation. We claim $\theta_1 \models \theta_2$. So, let $V \models \theta_1$, *i.e.* there exists $\vec{\delta}_1$ such that $V \models_{\vec{\delta}_1} \theta_1$. We can now use the intensional soundness theorem Thm. 7.3, instantiating $\vec{\delta}_2 = [\![D]\!](\vec{\delta}_1)$ to conclude that $\vec{\delta}_1 : \theta_1 \models [\![D]\!](\vec{\delta}_1) : \theta_2$. But this means, $V \models_{[\![D]\!](\vec{\delta}_1)} \theta_2$, whence $V \models \theta_2$ as desired.

Vice versa let $\theta_1 \models \theta_2$. One can show, regardless of what θ_1 looks like, that there must always exist a waveform V with $V \models_{\vec{1}} \theta_1$. In other words: Every function table is realizable with unit delays. Hence, there must exist $\vec{\delta}_2$ with $V \models_{\vec{\delta}_2} \theta_2$. Now we invoke the intensional completeness to obtain a derivation $D : \theta_1 \vdash \theta_2$ with $[\![D]\!](\vec{1}) \leq \vec{\delta}_2$. Thus, $\theta_1 \vdash \theta_2$, as desired.

∎

As an application of Theorem 7.1 in combination with Proposition 5.6 we otain a semantical characterization of combinational circuits in the sense of Malik [10]. It establishes a rigorous correspondence between this abstract ternary property and the intuitive understanding of combinational behaviour.

Theorem 7.2 *Let G be a binary gate network with primary inputs $I = \{a_1, \ldots, a_n\}$ and (remaining) gate output signals $b_1, \ldots, b_s$. Let $ft(G)$ be the associated ternary function table. Then, G is combinational in the sense of Malik [10] iff for every $i \leq s$ there exists a Boolean function $g_i \in \mathbb{B}^n \to \mathbb{B}$ such that for all waveforms V with $V \models ft(G)$ we have $V \models ft(I, b_i, g_i)$.*

Theorem 7.3 (Intensional Soundness) *For all $\vec{\delta}_1 \in \mathbb{N}_0^m$ and $\vec{\delta}_2 \in \mathbb{N}_0^n$, $\vec{\delta}_1 : \theta_1 \vdash \vec{\delta}_2 : \theta_2$ implies $\vec{\delta}_1 : \theta_2 \models \vec{\delta}_2 : \theta_2$.*

Example *Consider the RS-flipflop of Fig. 2. In the example on page 9 we verified that if the input is held in state $r = 0 \wedge s = 1$ long enough, then the output must stablilize in state $p = 1 \wedge q = 0$. As a proof of the sequent*

$$\mathrm{RS}'(r, s, p, q) \ ; \ r = 0 \wedge s = 1 \quad \vdash \quad \bigcirc(p = 1 \wedge q = 0)$$

we obtained the derivation

$$D \ = \ \bigcirc\!\wedge\! R \cdot (\supset\!\bigcirc L(1,1) \ \cdot \ \bigcirc id(1) \ , \ \supset\!\bigcirc L(6,1) \ \cdot \ \bigcirc\!\wedge\! R \ \cdot \ (\supset\!\bigcirc L(1,1) \ \cdot \ \bigcirc id(1), \bigcirc id(2))).$$

This yields a simulation of function tables

$$\supset\!\bigcirc R \cdot D \quad : \quad \mathrm{RS}'(r, s, p, q) \vdash (r = 0 \wedge s = 1) \supset \bigcirc(p = 1 \wedge q = 0).$$

The function table RS′ *consists of six transitions, namely three for each* NAND *gate. Accordingly, the delay parameter of* RS′ *is a six-tuple $\vec{\delta} = (\delta_1^1, \delta_1^2, \delta_1, \delta_2^1, \delta_2^2, \delta_2)$. Each component of $\vec{\delta}$ characterizes the stabilization bound for a particular transition in a particular* NAND *gate. In order to get the stabilization bound for the verified transition $(r = 0 \wedge s = 1) \supset \bigcirc(p = 1 \wedge q = 0)$ of the composite RS-flipflop, all we need to do* *is to extract the delay information from* $\supset\!\bigcirc R \cdot D$*:*

$$
\begin{array}{rcl}
[\![\supset\!\bigcirc R \cdot D]\!](\vec{\delta}) & = & [\![D]\!](\vec{\delta}) \\
& = & [\![\bigcirc\!\wedge\! R \cdot (\supset\!\bigcirc L(1,1) \ \cdot\cdots\ , \ \supset\!\bigcirc L(6,1) \ \cdot\cdots)]\!](\vec{\delta}) \\
& = & max\{[\![\supset\!\bigcirc L(1,1) \ \cdot \ \bigcirc id(1)]\!](\vec{\delta}) \ , \ [\![\supset\!\bigcirc L(6,1) \ \cdot \ \bigcirc\!\wedge\! R \ \cdot\cdots]\!](\vec{\delta})\} \\
& = & max\{[\![\bigcirc id(1)]\!](\vec{\delta}) + (\vec{\delta})_1 \ , \ [\![\bigcirc\!\wedge\! R \ \cdot\cdots]\!](\vec{\delta}) + (\vec{\delta})_6\} \\
& = & max\{0 + \delta_1^1 \ , \ max\{[\![\supset\!\bigcirc L(1,1) \ \cdot \ \bigcirc id(1)]\!](\vec{\delta}) \ , \ [\![\bigcirc id(2)]\!](\vec{\delta})\} + \delta_2\} \\
& = & max\{0 + \delta_1^1 \ , \ max\{0 + \delta_1^1 \ , \ 0\} + \delta_2\} \\
& = & \delta_1^1 + \delta_2 .
\end{array}
$$

By our soundness theorem, we know that if $V \models_{\vec{\delta}} \mathrm{RS}'$*, then* $V \models_{\delta_1^1+\delta_2} (r = 0 \wedge s = 1) \supset \bigcirc(p = 1 \wedge q = 0)$*. Note that* $\delta_1^1 + \delta_2$ *is a data-dependent stabilization bound that picks out precisely those delays from the* NAND*-gates' transitions that are actually relevant to produce the input-output response* $(r = 0 \wedge s = 1) \supset \bigcirc(p = 1 \wedge q = 0)$ *for the RS-flipflop.* ∎

Proof: Given $\vec{\delta}_1 \in \mathbb{N}_0^m$ and $\vec{\delta}_2 \in \mathbb{N}_0^n$ with $\vec{\delta}_1 : \theta_1 \vdash \vec{\delta}_2 : \theta_2$. This means there exists a derivation $D : \theta_1 \vdash \theta_2$ such that $[\![D]\!](\vec{\delta}_1) \leq \vec{\delta}_2$. Assume $V \models_{\vec{\delta}_1} \theta_1$. We prove $V \models_{\vec{\delta}_2} \theta_2$ by induction on D. This proves $\vec{\delta}_1 : \theta_2 \models \vec{\delta}_2 : \theta_2$.

• Let $D = \supset\!\bigcirc R \cdot (D_1, \ldots, D_n)$ such that $D_i : \theta_1; \sigma_2^i \vdash \bigcirc\tau_2^i$. By induction hypothesis we know that $V \models_{[\![D_i]\!](\vec{\delta}_1)} \sigma_2^i \supset \bigcirc\tau_2^i$. But, by definition, $[\![D]\!](\vec{\delta}_1) = ([\![D_1]\!](\vec{\delta}_1), \ldots, [\![D_n]\!](\vec{\delta}_1))$, whence $V \models_{[\![D]\!](\vec{\delta}_1)} \theta_2$.

• If $D = id(i)$ then $\sigma = \sigma_1 \wedge \mathbf{a} \wedge \sigma_2$ and $\tau = \mathbf{a}$. Thus, we must show $V \models_{[\![D]\!](\vec{\delta}_1)} \sigma_1 \wedge \mathbf{a} \wedge \sigma_2 \supset \bigcirc\mathbf{a}$. But this is trivial since $[\![D]\!](\vec{\delta}_1) = 0$ and for all $s, t \in \mathbb{N}$, $V, [s, t] \models \sigma_1 \wedge \mathbf{a} \wedge \sigma_2$ implies $V, [s + 0, t] \models \mathbf{a}$.

• Let $D = \bigcirc\!\wedge\! R \cdot (D_1, \ldots, D_k)$ where for all $i = 1 \ldots, m$, $D_i : \theta; \sigma \vdash \bigcirc\mathbf{a}_i$ and $\tau = \bigwedge_i \mathbf{a}_i$. By induction hypothesis we may assume that $V \models_{[\![D_i]\!](\vec{\delta}_1)} \sigma \supset \bigcirc\mathbf{a}_i$ for all i, which means that for all $s, t \in \mathbb{N}$,

$$V, [s, t] \models \sigma \quad \Rightarrow \quad V, [s + [\![D_i]\!](\vec{\delta}_1), t] \models \mathbf{a}_i. \tag{1}$$

Now observe that if $[s_1, t_1] \subseteq [s_2, t_2]$ then $V, [s_2, t_2] \models \mathbf{a}$ implies $V, [s_1, t_1] \models \mathbf{a}$. Thus, since $[\![D]\!](\vec{\delta}_1) = max\{[\![D_1]\!](\vec{\delta}_1), \ldots, [\![D_k]\!](\vec{\delta}_1)\}$ we can infer from (1)

$$V, [s, t] \models \sigma \quad \Rightarrow \quad V, [s + [\![D]\!](\vec{\delta}_1), t] \models \bigwedge_i \mathbf{a}_i$$

as desired.

• Let $D = \supset\!\bigcirc L(i, j) \cdot E$. This means that $\tau = \mathbf{a}$, $\theta = \theta_1 \ \wedge \ (\tau' \supset \bigcirc(\sigma_1 \wedge \mathbf{a} \wedge \sigma_2)) \ \wedge \ \theta_2$, and $E : \theta_1 \ \wedge \ (\tau' \supset \bigcirc(\sigma_1 \wedge \mathbf{a} \wedge \sigma_2)) \ \wedge \ \theta_2 \vdash \bigcirc\tau'$. By induction hypothesis, therefore we have that for all $s, t \in \mathbb{N}$,

$$V, [s, t] \models \sigma \quad \Rightarrow \quad V, [s + [\![E]\!](\vec{\delta}_1), t] \models \tau'. \tag{2}$$

We also know that $V \models_{\vec{\delta}_1} \theta$, *i.e.* in particular $V \models_{\delta_i} \tau' \supset \bigcirc(\sigma_1 \wedge \mathbf{a} \wedge \sigma_2)$. By definition of $\models$ this obtains

$$V, [s + [\![E]\!](\vec{\delta}_1), t] \models \tau' \quad \Rightarrow \quad V, [s + [\![E]\!](\vec{\delta}_1) + \delta_i, t] \models \sigma_1 \wedge \mathbf{a} \wedge \sigma_2. \tag{3}$$

Now, observe that $[\![D]\!](\vec{\delta}_1) = [\![E]\!](\vec{\delta}_1) + \delta_i$, so that taking (2) and (3) together yields

$$V, [s, t] \models \sigma \quad \Rightarrow \quad V, [s + [\![D]\!](\vec{\delta}_1), t] \models \mathbf{a},$$

which was to be shown.

- The case $D = \bigcirc\mathrm{contr} \cdot E$ is trivial. ∎

Theorem 7.3 ensures that ternary simulation provides safe timing information. But does it also provide complete, *i.e.* exact, information about real-time delays? The answer is yes, by the following intensional completeness theorem. Thus, the ternary calculus is "tight" w.r.t. delay bounds.

Theorem 7.4 (Intensional Completeness) *For all $\vec{\delta}_1 \in \mathbb{N}_0^m$ and $\vec{\delta}_2 \in \mathbb{N}_0^n$, $\vec{\delta}_1 : \theta_1 \models \vec{\delta}_2 : \theta_2$ implies $\vec{\delta}_1 : \theta_1 \vdash \vec{\delta}_2 : \theta_2$.*

Proof: Let $\vec{\delta}_1 \in \mathbb{N}_0^m$ and $\vec{\delta}_2 \in \mathbb{N}_0^n$ be fixed. Unraveling the definitions, we must show that if

$$\forall V.\, V \models_{\vec{\delta}_1} \theta_1 \quad \Rightarrow \quad V \models_{\vec{\delta}_2} \theta_2,$$

there exists a derivation $D : \theta_1 \vdash \theta_2$ such that $[\![D]\!](\vec{\delta}_1) \leq \vec{\delta}_2$, where the ordering is component-wise.

We prove the theorem by contraposition. To get rid of some indices let us agree to rename $\vec{\delta}_1$ by $\vec{\delta}$ and $\vec{\delta}_2$ by $\vec{\xi}$. We assume $\vec{\delta} \in \mathbb{N}_0^m$ and $\vec{\xi} \in \mathbb{N}_0^n$, $\delta_i \geq 1$, $\xi_i \geq 1$, are such that for all derivations D,

$$D : \theta_1 \vdash \theta_2 \quad \Rightarrow \quad [\![D]\!](\vec{\delta}) \not\leq \vec{\xi}.$$

This implies that there must exists $l \leq n$ such that for all derivations D_l,

$$D_l : \theta_1; \sigma_2^l \vdash \bigcirc\tau_2^l \quad \Rightarrow \quad [\![D_l]\!](\vec{\delta}) > \xi_l.$$

For otherwise, if for all $i \leq n$ there was a derivation $D_i : \theta_1; \sigma_2^i \vdash \bigcirc\tau_2^i$ with $[\![D_i]\!](\vec{\delta}) \leq \xi_i$, we could form the derivation $D = \supset\!\bigcirc R \cdot (D_1, \ldots, D_n)$ of $\theta_1 \vdash \theta_2$ for which $[\![D]\!](\vec{\delta}) \leq \vec{\xi}$. This contradicts the assumption. We claim that there is a waveform V such that

$$V \models_{\vec{\delta}} \theta_1 \tag{4}$$
$$V \not\models_{\xi_l} \sigma_2^l \supset \bigcirc\tau_2^l. \tag{5}$$

This, then, proves our theorem. Here we can see that the restriction $\delta_i \geq 1$ is necessary in general, as with $\delta_i = 0$, $V \models_{\vec{\delta}} \theta_1$ might force zero-delay oscillations, which cannot exist. So, (4) could not be satisfied.

For simplicity let us abbreviate σ_2^l by σ', τ_2^l by τ' and ξ_l by ξ' throughout the rest of the proof. Before we embark on the construction of V we introduce some general concepts, using the possibility that for the given set of delays $\vec{\delta}$ we can evaluate derivations as concrete delay values. We define a transition relation

$$\sigma \xrightarrow{t} \tau \quad \text{iff} \quad t = min\{ [\![D]\!](\vec{\delta}) \mid D : \theta_1 ; \sigma \vdash \bigcirc\tau \}$$

between arbitrary states σ, τ. As a special case we include $\sigma \xrightarrow{\infty} \tau$, *viz.* if there is no derivation D for $\theta_1 ; \sigma \vdash \bigcirc\tau$, adopting the convention that $min(\emptyset) = \infty$. The unique t such that $\sigma \xrightarrow{t} \tau$ is called the *transition time* from σ to τ, written $d(\sigma, \tau)$. This value satisfies the triangle inequality

$$d(\sigma, \tau) + d(\tau, \rho) \geq d(\sigma, \rho)$$

as one shows without difficulties, using the transitivity of $\vdash$ (*cf.* Prop. 5.2). We also find that

$$d(\sigma, \tau_1 \wedge \tau_2) = max\{d(\sigma, \tau_1), d(\sigma, \tau_2)\} \quad \text{and} \quad d(\sigma_1 \wedge \mathbf{a} \wedge \sigma_2, \mathbf{a}) = 0.$$

Two ternary states σ, τ are called *inconsistent* if there exists a signal a such that both $a = 1$ and $a = 0$ are contained in $\sigma \cup \tau$. One verifies that if τ_1 and τ_2 are both inconsistent with σ, then $d(\sigma, \tau_1) = d(\sigma, \tau_2)$, which is a consequence of our rule $\bigcirc$contr and its delay interpretation. This suggests to define, for every ternary state σ, the *transient time*

$$\delta(\sigma) \quad = \quad d(\sigma, a = 1 \wedge a = 0),$$

for some arbitrary a. Then, $\delta(\sigma) = d(\sigma, \tau)$ for any τ inconsistent with σ. Again, this includes the special case $\delta(\sigma) = \infty$ meaning there is no inconsistent state reachable from σ. We note that $d(\sigma, \tau) \leq \delta(\sigma)$ for arbitrary σ, τ, while equality holds if σ and τ are inconsistent.

The assumption, that for all derivations D of $\theta_1 ; \sigma' \vdash \bigcirc\tau'$, $[\![D]\!](\vec{\delta}) > \xi'$, implies $\xi' < d(\sigma', \tau')$. Thus, we may assume that $d(\sigma', \tau') \geq 1$. Our idea is to construct an infinite sequence

$$\vec{\rho} \quad = \quad \rho_0, \rho_1, \ldots, \rho_j, \rho_{j+1}, \ldots$$

of binary states $\rho_i \in \mathbb{S} \to \mathbb{B}$, in which, starting from a binary state $\rho_0 \in \mathbb{S} \to \mathbb{B}$ that extends σ', *i.e.* such that $\rho_0 \models \sigma'$, we let all signals oscillate as long as possible without violating the transition relation $\sigma \xrightarrow{t} \tau$. From the sequence we obtain the desired counter model V by putting $V(a)(s) = \rho_s(a)$.

The first step is to choose a suitable initial state ρ_0. To this end we observe that for every signal a there must exists a value $v_a \in \mathbb{B}$ such that $d(\sigma', a = 1 - v_a) \geq \delta(\sigma')$ (*i.e.* infinite or equal to $\delta(\sigma')$). For otherwise, if for some a both $d(\sigma', a = 0) < \delta(\sigma')$ and $d(\sigma', a = 1) < \delta(\sigma')$, this would contradict the property of $\delta(\sigma')$. This choice makes sure that, if for an atom $\mathbf{a}$ we have $d(\sigma', \mathbf{a}) < \delta(\sigma')$, then the atom must be $a = v_a$.

The choice of a $v_a \in \mathbb{B}$ for each $a \in \mathbb{S}$ is not unique: if both $d(\sigma', a = 1) \geq \delta(\sigma')$ and $d(\sigma', a = 0) \geq \delta(\sigma')$ we have two possibilities $v_a = 0$ and $v_a = 1$. For our purposes it does not matter which alternative we take, with one exception. There must at least one atom $\mathbf{c} \in \tau'$ such that $d(\sigma', \mathbf{c}) = d(\sigma', \tau')$. We select one such signal, and call it c. If for this particular signal we have both $d(\sigma', c = 1) \geq \delta(\sigma')$ and $d(\sigma', c = 0) \geq \delta(\sigma')$ then we choose v_c such that $c = v_c$ is the atom $\mathbf{c} \in \tau'$. This choice satisfies the condition that if $\mathbf{c}$ is different from $c = v_c$ then $d(\sigma', c = v_c) < \delta(\sigma')$. Henceforth we assume a choice of $\mathbf{c}$ and v_a for every a has been made according to these rules.

The sequence $\vec{\rho}$ now is constructed such that every signal a oscillates continuously until time $d(\sigma', a = v_a)$, whereupon it stabilizes to v_a until $\delta(\sigma') - 1$. Finally, from $\delta(\sigma')$ onwards all signals oscillate forever. More precisely, we want $\vec{\rho}$ to satisfy

$$\begin{aligned} \forall s.\ s+1 \leq d(\sigma', a = v_a) \ &\Rightarrow\ \rho_{s+1}(a) \ = \ 1 - \rho_s(a) &(6)\\ \forall s.\ d(\sigma', a = v_a) \leq s < \delta(\sigma') \ &\Rightarrow\ \rho_s(a) \ = \ v_a &(7)\\ \forall s.\ s+1 \geq \delta(\sigma') \ &\Rightarrow\ \rho_{s+1}(a) \ = \ 1 - \rho_s(a). &(8) \end{aligned}$$

This definition also includes the special cases $d(\sigma', a = v_a) = \infty$ or $\delta(\sigma') = \infty$ by the convention that $\forall s$ only ranges over finite elements. We can obtain such a $\vec{\rho}$ concretely, by taking as the initial state

$$\rho_0(a) = \begin{cases} v_a & \text{if } d(\sigma', a = v_a) \geq \delta(\sigma') \\ (v_a + d(\sigma', a = v_a))\ mod\ 2 & \text{otherwise,} \end{cases}$$

and then define

$$\rho_{j+1}(a) = \begin{cases} v_a & \text{if } d(\sigma', a = v_a) \leq j + 1 < \delta(\sigma') \\ 1 - \rho_j(a) & \text{otherwise.} \end{cases}$$

The sequence $\vec{\rho}$ has the properties (6)–(8), from which it follows that for all $s < t$,

$$V[s, t] \models \mathbf{a} \quad \text{iff} \quad d(\sigma', \mathbf{a}) \leq s \ \&\ t < \delta(\sigma'). \tag{9}$$

Direction $\Rightarrow$ is immediate from the conditions (6)–(8). For because of $s < t$ there are at least two successive time points in the interval $[s, t]$, and thus $V[s, t] \models \mathbf{a}$ implies that a cannot oscillate in $[s, t]$. Hence the only possibility is that $\mathbf{a}$ is the atom $a = v_a$, $d(\sigma', a = v_a) \leq s$ and $t < \delta(\sigma')$. The other direction $\Leftarrow$ follows because $d(\sigma', \mathbf{a}) \leq s$ & $t < \delta(\sigma')$ implies $d(\sigma', \mathbf{a}) < \delta(\sigma')$ and thus $\mathbf{a}$ must be the atom $a = v_a$ by the choice of v_a. Then, $V[s, t] \models \mathbf{a}$ by (7). Note, the direction $\Leftarrow$ also holds if $s = t$.

We claim that V is the desired waveform satisfying (4) and (5).

We begin with (4), and show for all $i \leq m$, $V \models_{\delta_i} \sigma_1^i \supset \bigcirc \tau_1^i$. Suppose $[s, t]$ is an interval such that $V[s, t] \models \sigma_1^i$. We must show that $V[s + \delta_i, t] \models \tau_1^i$. If $s + \delta_i > t$ then $[s + \delta_i, t]$ is the empty interval which satisfies τ_1^i trivially. Hence, we only need to do any work in the case that $s + \delta_i \leq t$. Since $\delta_i \geq 1$, this means that $s < t$ so that by property (9) $V[s, t] \models \sigma_1^i$ implies

$$\forall \mathbf{b} \in \sigma_1^i.\ d(\sigma', \mathbf{b}) \leq s \quad \& \quad t < \delta(\sigma').$$

So, again by property (9), it suffices to show that for all $\mathbf{a} \in \tau_1^i$, $d(\sigma', \mathbf{a}) \leq s+\delta_i$. Let $\mathbf{a} \in \tau_1^i$. It is not difficult to see that there is a derivation $D : \theta_1; \sigma_1^i \vdash \bigcirc\mathbf{a}$ with $[\![D]\!](\vec{\delta}) = \delta_i$. Hence, $d(\sigma_1^i, \mathbf{a}) \leq \delta_i$. Thus, by the triangle inequality, we are done if we can show $d(\sigma', \sigma_1^i) \leq s$. But this follows from the fact that for all $\mathbf{b} \in \sigma_1^i$, $d(\sigma', \mathbf{b}) \leq s$. This completes the proof of (4).

It remains to show (5), *i.e.* $V \not\models_{\xi'} \sigma' \supset \bigcirc\tau'$. Clearly, $V[0,t] \models \sigma'$, by (9), for all t such that $t < \delta(\sigma')$. So, all we need to find is a suitable t with $\xi' \leq t < \delta(\sigma')$ such that $V[\xi', t] \not\models \tau'$.

Suppose $d(\sigma', \tau') = \delta(\sigma') = \infty$. Then, for the atom $\mathbf{c} \in \tau'$ distinguished above, we have $d(\sigma', \mathbf{c}) = d(\sigma', \tau') = \infty$. If $\mathbf{c}$ is the same as $c = v_c$ then, since there is no $j+1$ with $d(\sigma', \mathbf{c}) \leq j+1 < \delta(\sigma')$ the signal c keeps oscillating throughout $\vec{\rho}$. Thus, $V[\xi', \xi'+1] \not\models \mathbf{c}$ and thence $V[\xi', \xi'+1] \not\models \tau'$ as desired. If $\mathbf{c}$ is different from $c = v_c$ this means that $d(\sigma', c = v_c) < \delta(\sigma')$, and thus by (7) signal c remains stable at v_c forever as from time $d(\sigma', c = v_c)$ onwards. So, we only need to choose $t > \xi'$ and get $V[\xi', t] \not\models \tau'$.

The remaining case is that not both of $d(\sigma', \tau')$ and $\delta(\sigma')$ are infinite. Then $t' := d(\sigma', \tau')$ must be finite (and ≥ 1) since always $d(\sigma', \tau') \leq \delta(\sigma')$. Now, consider the nonempty interval $[0, t'-1]$. We claim that $V[\xi', t'-1] \not\models \tau'$, where we know that $\xi' < t' \leq \delta(\sigma')$. Thus, $[\xi', t'-1]$ is nonempty. We know that $d(\sigma', \mathbf{c}) = t' > 0$. If $d(\sigma', \mathbf{c}) < \delta(\sigma')$ then $\mathbf{c} \in \tau'$ must be $c = v_c$ by the choice of v_c. But then by condition (7) we get $\rho_{t'}(c) = v_c$ and by (6) $\rho_{t'}(c) = 1 - \rho_{t'-1}(c)$, so that $\rho_{t'-1}(c) = 1 - v_c$. However, this means that $V[t'-1, t'-1] \not\models \tau'$ since τ' requires $c = v_c$. Thus, $V[\xi', t'-1] \not\models \tau'$ as desired. The final case is that $d(\sigma', \mathbf{c}) = \delta(\sigma')$. Here we argue in a similar way as before. If $\mathbf{c}$ is the same as $c = v_c$ then, since there is no $j+1$ with $d(\sigma', \mathbf{c}) \leq j+1 < \delta(\sigma')$ the signal c keeps oscillating throughout $\vec{\rho}$. Thus, $V[\xi', \xi'+1] \not\models \mathbf{c}$ and thence $V[\xi', \xi'+1] \not\models \tau'$ as desired. If $\mathbf{c}$, on the other hand, is different from $c = v_c$ this means that $d(\sigma', c = v_c) < \delta(\sigma')$, and thus by (7) signal c remains stable at v_c between $d(\sigma', c = v_c)$ and $t'-1 < \delta(\sigma')$, *i.e.* $V[t'-1, t'-1] \models c = v_c$. Hence, $V[\xi', t'-1] \not\models \tau'$ for $\mathbf{c}$ is different from $c = v_c$. ∎

8 An Equivalent Intuitionistic Kripke Semantics

The reader may observe how our semantic definitions in Sec. 6 closely follow the structure of the transition formula $\sigma \supset \bigcirc\tau$. The $\supset$ has its direct counterpart in the semantical implication $\Rightarrow$, and $\bigcirc$ in the time shift $[s+\delta, t]$, and the symbol $\wedge$ comes out as logical conjunction. This, of course, is not accidental. In fact, transition tables are a fragment of Propositional Lax Logic [7] (PLL) and, as we will show next, the semantics presented above falls out as a particular class of models for PLL.

This section is a postscript to the main theme of this report. The reader may skip this section on first reading. We first recall some facts from [7].

Definition 8.1 (Kripke Constraint Model) *A (Kripke)* constraint model *is a quintuple $\mathcal{C} = (W, R_m, R_i, V, F)$, where W is a non-empty set, R_m, R_i are binary relations on W, $F \subseteq W$, and V is a map that assigns to every atom $\mathbf{a}$ a subset $V(\mathbf{a}) \subseteq W$. These data are subject to the following conditions:*

- *R_m, R_i are preorders, i.e. reflexive and transitive relations, and $R_m \subseteq R_i$,*
- *F and V are hereditary w.r.t. R_i, i.e. if aR_ib, then $a \in F$ implies $b \in F$, and $a \in V(A)$ implies $b \in V(A)$,*
- *V is full on F, i.e. $F \subseteq V(A)$.*

Kripke constraint models are a natural class of two-frame Kripke structures for PLL, which is the logic generated from the grammar

$$M \quad ::= \quad \mathbf{a} \mid \mathit{true} \mid \mathit{false} \mid M \wedge M \mid M \vee M \mid M \supset M \mid \neg M \mid \bigcirc M$$

where $\mathbf{a}$ ranges over a possibly infinite set of propositional atoms. If we let these constants to be of the form $a = 1$, $a = 0$ with $a \in \mathbb{S}$, then function tables θ are a simple fragment of PLL. $\neg M$ is taken as an abbreviation for $M \supset \mathit{false}$ and *true* abbreviates $a = 1 \supset a = 1$. Kripke constraint models provide a natural semantics to PLL formulas by virtue of the following semantical clauses:

Definition 8.2 (Validity) *Let $\mathcal{C} = (W, R_m, R_i, V, F)$ be a constraint model. Given a formula M and $c \in W$, M is* valid *at c in $\mathcal{C}$, written $\mathcal{C}, c \models_c M$ iff*

- *M is a propositional constant $\mathbf{a}$ and $c \in V(\mathbf{a})$;*
- *M is $N \wedge K$ and both $\mathcal{C}, c \models_c N$ and $\mathcal{C}, c \models_c K$;*
- *M is $N \vee K$ and $\mathcal{C}, c \models_c N$ or $\mathcal{C}, c \models_c K$;*
- *M is false and $c \in F$;*
- *M is $N \supset K$ and for all $a \in W$ such that cR_ia, $\mathcal{C}, a \models_c N$ implies $\mathcal{C}, a \models_c K$;*
- *M is of form $\bigcirc N$ and for all $b \in W$, cR_ib, there exists $a \in W$ with bR_ma such that $\mathcal{C}, a \models_c N$.*

A formula M is valid *in $\mathcal{C}$, written $\mathcal{C} \models_c M$, if for all $c \in W$, M is valid at c in $\mathcal{C}$; M is* valid*, written $\models_c M$, if M is valid in any constraint model $\mathcal{C}$.*

Now we can nail down the connection between our real-time semantics for function tables and Kripke constraint models. Given a waveform $V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ we define a concrete constraint model

$$\mathcal{C}(V) \quad = \quad (W, R_m, R_i, V, F)$$

such that

- W is the set of pairs (I, t) where $I \subseteq \mathbb{N} \times \mathbb{N}$ and $t \in \mathbb{N}$
- $(I_1, t_1)R_i(I_2, t_2)$ iff $I_2 \subseteq I_1$ and $t_1 \leq t_2$

- $(I_1, t_1) R_i (I_2, t_2)$ iff $I_1 = I_2$ and $t_1 \leq t_2$
- $(I, t) \in V(a = 0)$ iff $\forall (r, s) \in I.\ \forall \tau \in [r + t, s].\ V(a)(\tau) = 0$
- $(I, t) \in V(a = 1)$ iff $\forall (r, s) \in I.\ \forall \tau \in [r + t, s].\ V(a)(\tau) = 1$
- $(I, t) \in F$ iff $\forall (r, s) \in I.\ [r + t, s] = \emptyset$.

Theorem 8.3 *For all function tables θ, $V \models \theta$ iff $\mathcal{C}(V) \models_c \theta$.*

The theorem means that our real-time semantics $\models$ is a proper Kripke-style semantics, and that the ternary function tables can be extended naturally to a proper intuitionistic modal logic. In this modal logic many more things can be expressed, which are beyond the ternary model. For instance, we can specify that a signal a must stabilize eventually, either to 1 or to 0, by the formula

$$a \downarrow \;\; = \;\; ((a = 1 \supset \bigcirc \mathit{false}) \;\wedge\; (a = 0 \supset \bigcirc \mathit{false})) \supset \bigcirc \mathit{false}.$$

The behaviour $a \downarrow$ is not expressible in the ternary model. There is no ternary function table θ equivalent to $a \downarrow$. To give another example, the formulas of form $\neg\neg M$ express point-wise behaviour: $\mathcal{C}(V) \models_c \neg\neg M$ iff V satisfies condition M in a point-wise fashion. Thus, we can express arbitrary state invariants. Of interest are also higher-order function tables, *e.g.* expressions of the form $(\theta_1 \supset \theta_2) \supset (\theta_3 \supset \theta_4)$, where each θ_i is a ternary function table. A calculus based on these would amount to a notion of higher-order ternary simulation. All these extensions to ternary function tables, which become available in the full setting of PLL still need to be explored.

9 Conclusion and Future Work

We have introduced a natural real-time semantics for binary gate networks, which is sound and complete for the standard ternary gate model. This interpretation gives one way of closing an open abstraction gap that remains, in order to view the ternary model as a level intermediate between the static Boolean model and the (discrete) real-time behaviour of circuits. Our real-time semantics captures the combinational behaviour, in the sense of Malik [10], of arbitrary circuits.

On the syntactic side our results imply that ternary function tables can be seen as a fragment of a specific intuitionistic modal logic, Propositional Lax Logic. The operational semantics of function tables in this setting comes down to a formal logical calculus. The intuitionistic nature of the calculus makes it possible to extract data-dependent stabilization delays from derivations in the calculus. The intensional versions of our soundness and completeness theorems then guarantees that the extracted delays are both correct and exact, relative to the chosen real-time semantics. This shows that ternary simulation and functional timing analysis, which so far have been treated as unrelated methods (see *e.g.* [10]) can be unified in a single framework.

We aim to extend our results to more powerful logical calculi and to richer real-time semantics. As a first step we will consider full Propositional Lax Logic with special focus on higher-order function tables, *i.e.* formulas with arbitrary nesting of implications. In another direction it would appear natural to introduce Boolean expressions to arrive at a (restricted) first-order logical system, in which symbolic simulation can be represented. Concerning the semantics we will investigate more sophisticated delay models, involving upper as well as lower bounds. Our hope is that this would lead to a general framework for the extraction of timing constraints, such as set-up and hold times, for asynchronous sequential circuits in fundamental mode operation.

This report in the first place aims at a logical and semantical investigation of ternary simulation, not at making a contribution to the algorithmic side of the matter. Though our simulation calculus can be implemented it operates at too low a level to be efficient. The main disadvantage is that it manipulates concrete bits of binary information rather than a compact symbolic representation of ternary states, as done by other published simulation algorithms (like Bryant's). However, as our calculus is a calculus of logic, it can be made arbitrarily symbolic, simply by enriching the logical formalism. In this way it should be possible to extend our results to full ternary symbolic simulation, and in fact further to cover large amounts of first-order and higher-order intuitionistic theorem proving.

An important feature of our intuitionistic logic approach, as compared to the algebraic ternary modelling, is that it leaves open the interpretation of the "third" value. In fact it does not even freeze the assumption that there be only one additional value besides 0 and 1. The third value in our function tables is present, not explicitly as in the ternary algebra, but indirectly by the absence of any positive information about the Boolean behaviour. Thus, our approach is open to accommodate also a "forth", "fifth", "sixth" value, and so on. One can show that the semantics of higher-order function tables already requires more than three truth values. We believe that this openness in the non-Boolean dimension of the intuitionistic approach is very profitable for formal hardware verification, in order to preserve the validity of correctness arguments in passing from higher to lower levels of abstraction.

References

[1] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity.* John Wiley and Sons, 1992.

[2] J. C. Barros and B. W. Johnson. Equivalence of the arbiter, the synchronizer, the latch and the inertial delay. *IEEE Transactions on Computers*, C-32(7):603–614, July 1983.

[3] M. A. Breuer. A note on three-valued logic simulation. *IEEE Transactions on Computers*, C-21(4):399–402, April 1972.

[4] R. E. Bryant. Boolean analysis of MOS circuits. *IEEE Transactions on Computer-aided Design*, 6(4):634–649, July 1987.

[5] J. A. Brzozowski and M. Yoeli. On a ternary model of gate networks. *IEEE Transactions on Computers*, C-28:178–184, 1979.

[6] E. B. Eichelberger. Hazard detection in combinational and sequential switching circuits. *IBM J. Res. Div.*, 9:90–99, 1965.

[7] M. Fairtlough and M. Mendler. An intuitionistic modal logic with applications to the formal verification of hardware. In L. Pacholski and J. Tiuryn, editors, *Proceedings of the 1994 Annual Conference of the European Association for Computer Science Logic*, pages 354–368. Springer, LNCS 933, 1995.

[8] S. C. Kleene. *Introduction to Metamathematics*. North Holland, Amsterdam, 1952. Chap. XII, Par. 64.

[9] J. W. Lloyd. *Foundations of Logic Programming*. Springer, 1984.

[10] S. Malik. Analysis of cyclic combinational circuits. In *International Conference on Computer-aided Design*, pages 618–625. IEEE, 1993.

[11] L. R. Marino. General theory of metastable operation. *IEEE Transactions on Computers*, 30(2):107–115, February 1981.

[12] M. Mendler. A timing refinement of intuitionistic proofs and its application to the timing analysis of combinational circuits. Technical Report MIP-9518, University of Passau, November 1995.

[13] M. Mendler and T. Stroup. Newtonian arbiters cannot be proven correct. *Formal Methods in System Design*, 3(3):233–257, November/December 1993.

[14] M. Yoeli and J. A. Brzozowski. Ternary simulation of binary gate networks. In J. M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 41–50. D. Reidel, 1977.

[15] M. Yoeli and S. Rinon. Application of ternary algebra to the study of static hazards. *Journal of the ACM*, 11:84–97, 1964.