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

Michael Mendler
Department of Mathematics and Computer Science, University of Passau
Passau, Germany

Matt Fairtlough
Department of Computer Science, The University of Sheffield
Sheffield, United Kingdom

Abstract

We prove the equivalence between the ternary circuit model and a notion of intuitionistic stabilization bounds. This formalizes in a mathematically precise way the intuitive understanding of the ternary model as a level intermediate between the static Boolean model and the (discrete) real-time behaviour of circuits. 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 Acknowledgements

Michael Mendler is supported by a Deutsche Forschungsgemeinschaft fellowship, and Matt Fairtlough by a grant from the University of Sheffield research development fund.

2 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

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

of ternary simulation has been developed further by numerous authors, e.g. at the gate level by Brzozowski and Yoeli in [14, 5], and Malik in [10], or at the transistor 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 not so obvious. 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 and how can it be formalized? In ternary simulation the three-valued model usually is introduced as a refinement of the abstract Boolean view rather than as an abstraction of real-time behaviour. As long as the concern is more with algorithms and data structures this is the most convenient approach. However, when it comes to correctness and completeness issues this is not sufficient. We are forced to cross the t's and dot the i's and nail down the exact relationship the abstract model has to the real-time behaviour. We must make precise the intuitive reading of the new third value, be it one of "oscillation", "transient", "undefined binary value", "don't care" or all of them.

In this paper 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. We prove that the operational semantics is sound and complete with respect to 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.

3 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 and discrete time. 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, we may use transport or inertial delay assumptions 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 a three-valued domain.

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 some or all of "oscillation", "instability", "undefined", "don't care". The exact meaning usually is left unspecified, only implicitly 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

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

seen in Fig. 1. Our goal is to understand this ternary behaviour not only as a refinement of the two-valued

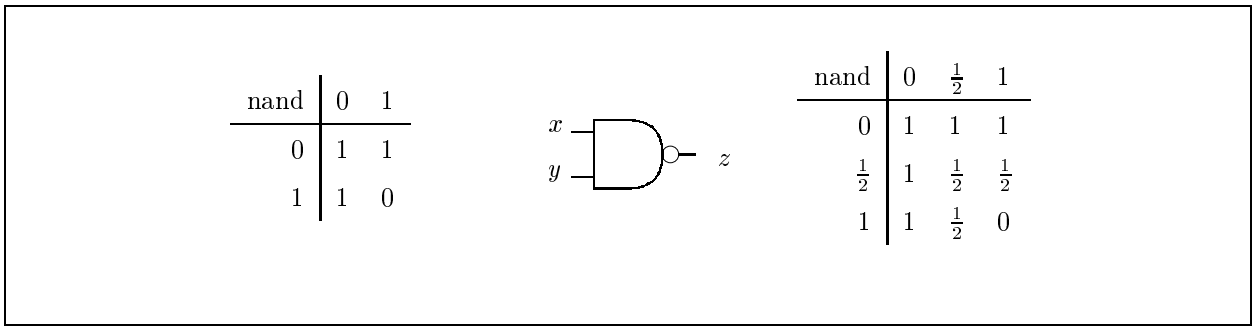


Figure 1: Ternary Refinement of the nand Gate.

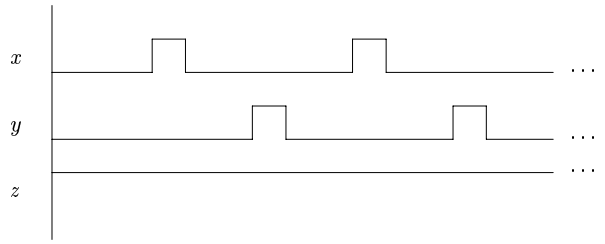
model, but conversely as an abstraction of the real-time behaviour of nand. We expect that this ternary abstraction be compatible with the standard Boolean abstraction, which applies to the final stable state of a circuit. Thus, it would appear natural also to try to take the ternary function table as a statement about the final stationary state of the circuit. Such a connection 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 $a \in \mathbb{S}$,

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

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 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

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

Though this is compatible with the static Boolean semantics, this 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. Even for loop-free circuits, or when oscillation is no issue, there

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

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. In some sense the difficulties seem to stem from the wish to interpret $\mathbb{K}$ as representing concrete signal values, signals, or properties of individual signals.

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 is no longer on a par with 0 and 1. This is the original reading of Kleene [8], which in fact is implicit in most ternary simulation approaches such as [5, 10, 4]. In this view the ternary table specifies a continuous function in the complete partial ordering $(\mathbb{K}, \leq)$. The ternary simulation of a binary network, then, corresponds to a least fixed-point computation. This domain-theoretic interpretation of the ternary function table, however, does not answer our question. It does not assign any concrete real-time 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 paper we offer one possible connection using a real-time interpretation of ternary function tables, based on a notion of bounded stabilization.

4 The Ternary Model

To begin with, let us recall the basic elements of the standard ternary extension of binary gate modelling. 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.

As mentioned, the ternary extension of Boolean functions rests on viewing $\mathbb{K} = \{0, 1, \frac{1}{2}\}$ as a domain of information for specifying the elements of $\mathbb{B}$ as "certainly 0," "certainly 1" and "unknown." Measuring information by subsets the element $v \in \mathbb{K}$ represents $\nu(v) \subseteq \mathbb{B}$ such that $\nu(1) := \{1\}$, $\nu(0) := \{0\}$ and $\nu(\frac{1}{2}) := \{0, 1\}$. The inverse of this representation function $\nu \in \mathbb{K} \to 2^{\mathbb{B}}$ is the partial "averaging" function $\mu \in 2^{\mathbb{B}} \to \mathbb{K}$. The representation function is naturally extended to ternary vectors as $\nu \in \mathbb{K}^n \to 2^{\mathbb{B}^n}$ by

$$\nu(u_1, \ldots, u_n) \quad := \quad \{\, (v_1, \ldots, v_n) \mid v_i \in \nu(u_i) \,\}.$$

Any Boolean function $f \in \mathbb{B}^n \to \mathbb{B}$ may then (for $n > 0$) be given a ternary extension $f^* \in \mathbb{K}^n \to \mathbb{K}$ by precomposing with ν and postcomposing with μ:

$$f^*(\vec{u}) \quad := \quad \mu\,(f\,(\nu\,(\vec{u}))),$$

where f is lifted to subsets in the standard way.

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 called stable, and intuitively represents a possible steady state of the circuit, given that the inputs remain unchanged. Then the extension $F^* \in \mathbb{K}^{n+s} \to \mathbb{K}^s$ 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 paper 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, in this case of the continuous function $\lambda\vec{w}.\ F^*(\vec{u}, \vec{w}) \in \mathbb{K}^s \to \mathbb{K}^s$, over the complete partial ordering $(\mathbb{K}, \leq)$. This least fixed point semantics

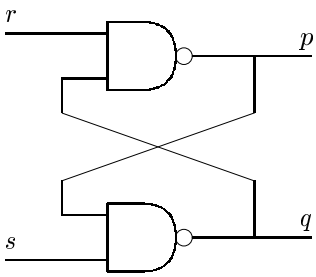
M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

is applied, implicitly, by Malik [10] and also by Bryant [4] at the transistor level. In the rest of this paper, 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, $\mu F^*(\vec{u})$ represents the steady state response of the circuit for input state $\vec{u}$. 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.

5 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

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

where a ranges over the set of signal names $\mathbb{S}$. When a is a signal name we will take **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. States specify (ternary) information about the state of some signals. $a = 0 \wedge b = 1$ means "signal a is surely 0 and b is surely 1", those signals not mentioned in a state are given the ternary value $\frac{1}{2}$. They might be represented by atoms $c = \frac{1}{2}$, but since "no information" is

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

logically equivalent to true these are redundant and thus systematically eliminated. The state $a = 1 \wedge a = 0$ is an inconsistent state and equivalent to false. 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. 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. Such a transition $\sigma \supset \bigcirc\tau$ represents the logical statement that state σ necessarily leads to state τ, provided this is consistent (with τ and the context in which the transition takes place). The modal symbol $\bigcirc$ is to indicate this constraint "provided it is consistent", but this will come up again later.

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

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

Note that the actual ordering and bracketing of the conjuncts will be of no importance. If $G = G_1, \ldots, G_n$ is a network of binary gates the associated function table is

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

Our idea is that $\mathrm{ft}(G)$ 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 $\mathrm{ft}(G)$ both in an operational and in an axiomatic sense. In other words, we may view $\mathrm{ft}(G)$ 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 $\mathrm{ft}(N)$, up to permutations, is the formula $\mathrm{nand}(x, y, z)$ defined

$$\begin{array}{rl} & \mathrm{nand}(x, y, z) \ := \\ ((x = 0 \wedge y = 0) \supset \bigcirc(z = 1)) & \wedge \quad (x = 0 \supset \bigcirc(z = 1)) \quad \wedge \quad ((x = 0 \wedge y = 1) \supset \bigcirc(z = 1)) \wedge \\ (y = 0 \supset \bigcirc(z = 1)) & \wedge \\ ((x = 1 \wedge y = 0) \supset \bigcirc(z = 1)) & \wedge \qquad\qquad\qquad\qquad ((x = 1 \wedge y = 1) \supset \bigcirc(z = 0)). \end{array}$$

Observe how this formula closely corresponds with the ternary function table of the nand in Fig. 1. The rule is that every entry in the table yields one conjunct, but entries $\frac{1}{2}$ and inputs $\frac{1}{2}$ are systematically dropped. This is because in the logic $a = \frac{1}{2}$ represents the formula true which is redundant. With $\mathrm{ft}(N)$ at hand we form the function table for the RS-flipflop in Fig. 2 as

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

∎

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

We will view the ternary simulation of a gate network G as an abstract computation manipulating the ternary transition table $\mathrm{ft}(G)$. In practice, however, simulation algorithms represent the ternary values concretely,

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

say by double rail binary encoding, and perform the simulation by algebraic operations. Yet, even though the simulation may 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 $\sigma \equiv a_1 = v_1 \wedge \cdots \wedge a_m = v_m$ 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. $s > t$, 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 artificially but it makes matters more uniform if we include it. We observe that the interpretation of σ means the syntactic construct $\wedge$ is construed 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. 3, 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 partitions into 8 adjacent subintervals $[t_i, t_{i+1}]$, $i = 0, \ldots, 7$, in which it assumes specified states. For instance,

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

∎

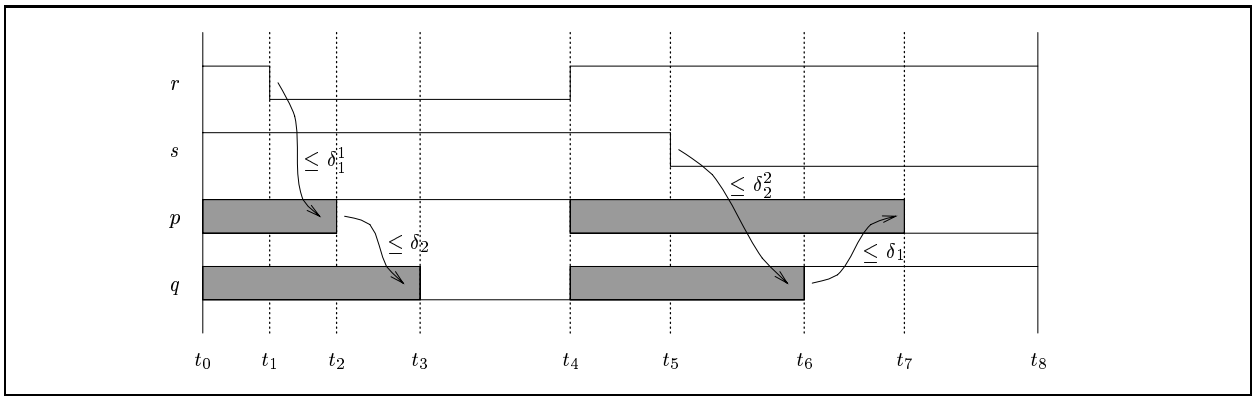


Figure 3: Some Waveforms Operating the RS-flipflop.

Next 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 in an operational way. 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 σ remains stable. Note the implicit constraint: in order to conclude that V necessarily enters the state τ we must have that the interval $[s,t]$ with state σ has at least the length δ. This amounts to a conservative view of the I/O behaviour of a combinational circuit. If the input stimulus does not persist for long enough nothing can be concluded about the output reaction.

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

Example With transition formulas we can specify the upper bounds indicated in the timing diagram of Fig. 3. 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

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

∎

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, 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 is lost for the time being, we will see that it can be recovered in an exact way. This is possible if we take validity not only in the extensional sense, viz. the mere fact that a waveform V satisfies a transition $\sigma \supset \bigcirc\tau$, but also in an intensional sense, viz. just how V satisfies $\sigma \supset \bigcirc\tau$. This "how" is measured by the smallest δ such that $V \models_\delta \sigma \supset \bigcirc\tau$.

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 again the waveforms of Fig. 3, where the t_i now are assumed to be specific 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, $V \models_\delta r = 1 \supset \bigcirc(s = 0)$ iff $\delta \geq \max\{t_1 - t_0, t_5 - t_4\}$. ∎

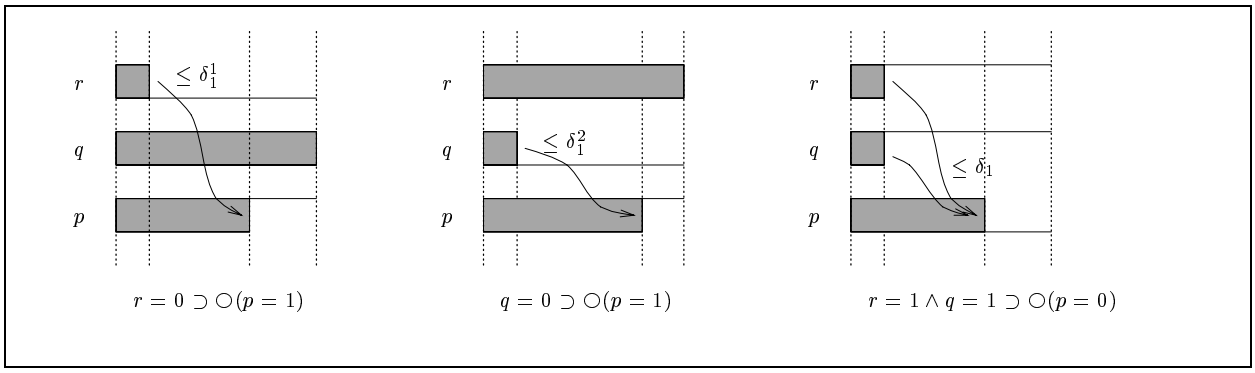


Figure 4: Stabilization Behaviour of nand(r, q, p).

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

$$\text{nand}(r, q, p) \quad = \quad r = 0 \supset \bigcirc p = 1 \wedge q = 0 \supset \bigcirc p = 1 \wedge (r = 1 \wedge q = 1) \supset \bigcirc p = 0.$$

$V \models \text{nand}(r, q, p)$ 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. 4 are satisfied by V. A similar interpretation applies to the lower 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 $\text{rs}(r, s, p, q) =$

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

$\mathrm{nand}(r,q,p) \wedge \mathrm{nand}(p,s,q)$ of the RS-flipflop we obtain a real time specification that contains the timing diagram of Fig. 3.

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. 3 is long enough for the state $s=1 \wedge p=1 \wedge q=0$ 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. In order to get this memory effect we would also need to allow lower bounds, or negative delays.

As 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 situation is somewhat analogous to the modelling of sequential circuits, which also can be described completely in terms of their combinational state transition function, assuming no state variables are hidden. ∎

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. The simplest example is 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,

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

is expressing that V exhibits bounded oscillation as seen in Fig. 5, for some upper bounds δ_1, δ_2.

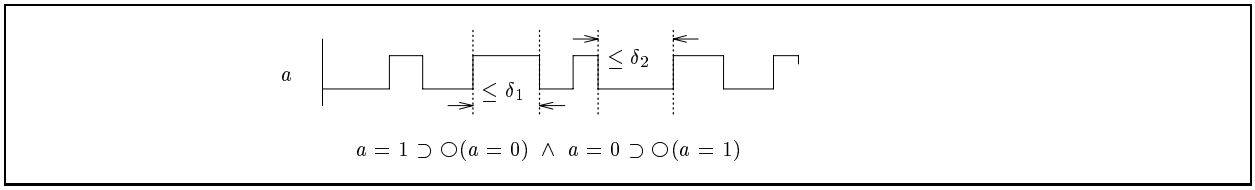


Figure 5: Bounded Oscillation.

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

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

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 definition of $V \models x=0 \supset \bigcirc(z=1)$ yields the statement

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

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

A weaker way of abstracting from the delay would be the condition

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

where we have swapped 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 to subtle physical phenomena such as the metastable operation of circuits [11, 13] this weaker delay model might be more appropriate. 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. In the weakest version the nand gate then might look like

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

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. Lower bounds can be obtained with negative propagation delays. For instance,

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

implies that whenever (e.g. output) x goes to 0 then (e.g. input) z must have been stable at 1 for at least a period δ. This is a lower bound on z and an inertiality for the reaction of x to z. 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.

Each of these choices for a delay model yields a different real-time interpretation, and in each case the question arises of how it might relate to the abstract ternary model.

7 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 constitutes 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. 6. In the rules $\supset\bigcirc L$ and $\bigcirc$id it is to be understood that one or both of 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 represents 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, expressing that simulating the function table θ starting from initial state σ leads to the response τ. The rules $\bigcirc\wedge R$ and $\supset\bigcirc L$ are familiar from Prolog. In fact,

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

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

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

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

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

Figure 6: A Sequent Calculus for Ternary Simulation.

they are essentially the rules known as the decomposition and backchaining rule, respectively, of the Prolog SLD-resolution mechanism [9]. The rule $\bigcirc$id is a special case of backchaining for atomic clauses. These three rules intuitively serve to follow the propagation of signal values through the circuit. Signal values are introduced as input with rule $\bigcirc$id, propagated through a gate by activating a particular transition with $\supset\bigcirc L$, and bundled together by $\bigcirc\wedge R$. Pushing this analogy, the simulation relation $\theta;\sigma \vdash \bigcirc\tau$ could be seen as a Propositional Lax Logic version of the SLD-resolution relation with program $\theta;\sigma$ and answer goal τ. The analogy breaks down, however, with rule $\bigcirc$contr, which is special to our calculus. It formalizes a notion of inconsistency, saying that if a function table θ (starting from state σ) is shown to lead to an inconsistent response $a = 1 \wedge a = 0$, then we can derive any response $b = v$ for it. Thus, the state $a = 1 \wedge a = 0$ has the same role as the formula false in logic. This means that our notion of simulation also involves keeping track of inconsistencies. As a consequence the relation $\theta;\sigma \vdash \bigcirc\tau$ must be read as the constrained assertion that θ starting from initial state σ leads to the response τ, provided σ can be maintained for long enough without producing an inconsistency. This relativization is crucial to cope with oscillations.
The relationship between the two relations $\theta_1 \vdash \theta_2$ and $\theta;\sigma \vdash \bigcirc\tau$ is given by Prop. 7.1 below.

Proposition 7.1 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 7.1 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

$$\mathrm{nand}'(x,y,z) \quad := \quad (x = 0 \supset \bigcirc(z = 1)) \ \wedge\ (y = 0 \supset \bigcirc(z = 1)) \ \wedge\ ((x = 1 \wedge y = 1) \supset \bigcirc(z = 0)).$$

Using the rules of the calculus one derives the relations

$$\mathrm{nand}'(x,y,z) \vdash \mathrm{nand}(x,y,z) \qquad \mathrm{nand}(x,y,z) \vdash \mathrm{nand}'(x,y,z).$$

The derivations verify essentially that the transition $x = 0 \supset \bigcirc(z = 1)$ is logically equivalent to the conjunction $((x = 0 \wedge y = 0) \supset \bigcirc(z = 1)) \wedge (x = 0 \supset \bigcirc(z = 1)) \wedge ((x = 0 \wedge y = 1) \supset \bigcirc(z = 1))$ and that $y = 0 \supset \bigcirc(z = 1)$ is equivalent to $((x = 0 \wedge y = 0) \supset \bigcirc(z = 1)) \wedge (y = 0 \supset \bigcirc(z = 1)) \wedge ((x = 1 \wedge y = 0) \supset \bigcirc(z = 1))$. ∎

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

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

grammar:

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

where i in $\text{id}(i)$ 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 derivation $D : \text{rs}'(r,s,p,q); \sigma(01) \vdash \bigcirc(p = 1 \wedge q = 0)$ with $D = \bigcirc\wedge R \cdot (D_1, D_2)$, where the subproofs D_1 and D_2 are constructed as follows:

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

We read off the proof terms

$$D_1 = \supset\bigcirc L(1,1) \cdot \bigcirc\text{id}(1)$$

and

$$D_2 = \supset\bigcirc L(6,1) \cdot \bigcirc\wedge R \cdot (\supset\bigcirc L(1,1) \cdot \bigcirc\text{id}(1), \bigcirc\text{id}(2)).$$

∎

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

Proposition 7.2 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 7.3 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 7.3 identifies a very useful normalization property, which relates to the 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. 6) by dropping the atom $\mathbf{b}$ in the premiss, i.e. replace $\supset\bigcirc L$ by the rule

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

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 an executable, albeit naive, implementation. The direct implementation of the calculus is not our primary concern, however. The purpose of the calculus is to act as a reference system for reasoning about about correctness and completeness of real-time semantics, and as an interface between the logic and algebraic viewpoints.

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

7.1 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 $b \notin I$, and the associated function table $\mathrm{ft}(G)$. How do we simulate G, on the basis of its description $\mathrm{ft}(G)$, 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

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

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

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

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

- $R(\vec{v}) = \{b = 1\}$. This is the case if 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 $\mathrm{ft}(G)$ does not contain conflicting transitions for b. We can sum up the situation as follows:

Proposition 7.4 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 is determined completely by our ternary calculus. Note, that the value $\frac{1}{2}$ is represented indirectly: it stands for the absence of information. Formally, we have

$$g^*(\vec{v}) = \frac{1}{2} \quad \text{iff} \quad \mathrm{ft}(G); \sigma(\vec{v}) \not\vdash \bigcirc(b = 1) \text{ and } \mathrm{ft}(G); \sigma(\vec{v}) \not\vdash \bigcirc(b = 0).$$

The relationship between $\mathrm{ft}(G)$ and g^* as in Prop. 7.4 generalizes to arbitrary binary networks without feedback. Here $\mathrm{ft}(G)$ is the conjunction of the gates' function tables and g the functional composition of the gates' Boolean functions. In this way we find included 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 7.5 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 have included the essence of Malik's ternary combinational analysis in a logical framework. For instance, we can give a characterization of Malik's notion of "combinational" circuits as follows:

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

Proposition 7.6 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 (with rule $\supset\!\circ L^*$) 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 $\mathrm{ft}(G) = \bigwedge_{i=1}^{n}(\bigwedge_{j=1}^{m_i} \mathbf{a}_i^j) \supset \circ\mathbf{b}_i$ 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, $i \leq l$ is represented by the set of atoms $\sigma(\vec{v}) = \{\, a_i = v_i \mid i \leq l \,\}$. The stationary response $R(G, \vec{v})$ of the circuit for input stimulus $\sigma(\vec{v})$ can be computed iteratively by following Lloyd [9] in constructing the least Herbrand model of $\mathrm{ft}(G)$ together with $\sigma(\vec{v})$:

$$\begin{array}{lcl} R_0 & = & \sigma(\vec{v}) \\ R_{k+1} & = & \{\, \mathbf{b}_i \mid i \leq n \;\&\; \forall j \leq m_i.\, \mathbf{a}_i^j \in R_k \,\}. \end{array}$$

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 that we will find 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 can 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].

8 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

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

in 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 $\theta_1 = \bigwedge_{i=1}^{m} \sigma_1^i \supset \circ\tau_1^i$ and $\theta_2 = \bigwedge_{i=1}^{n} \sigma_2^i \supset \circ\tau_2^i$. 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 $[\![D]\!] \in \mathbb{N}^m \to \mathbb{N}^n$

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

as follows:

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

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

$$\circ\mathrm{id} \;\mapsto\; 0 \qquad\qquad \circ\!\wedge R \;\mapsto\; \max \qquad\qquad \supset\!\circ L \;\mapsto\; +,$$

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], on the timing extraction from constructive proofs. It uses the general setting of Propositional Lax Logic [7], but with a simpler real-time semantics.

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 now 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].

We should remark that completeness, in general, need not hold for delays $\vec{\delta}_1$ in which some components take the value 0. For instance, if $\theta_1 = a = 1 \supset \circ(a = 0) \;\wedge\; a = 0 \supset \circ(a = 1)$ and $\vec{\delta}_1 = (0,0)$, then $\vec{\delta}_1 : \theta_1$ amounts to an inconsistent specification. There is no waveform V such that $V \models_{\vec{\delta}_1} \theta_1$. 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 delay-dependent inconsistency, which cannot be detected by the calculus, as it 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 8.1 (Extensional Soundness and Completeness) $\theta_1 \vdash \theta_2$ iff $\theta_1 \models \theta_2$.

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

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

As an application of Theorem 8.1 in combination with Proposition 7.6 we obtain 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, according to which a circuit is combinational if its behaviour can be represented by a Boolean function.

Theorem 8.2 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 $\mathrm{ft}(G)$ 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 $V \models \mathrm{ft}(G)$ we have $V \models \mathrm{ft}(I, b_i, g_i)$.

Theorem 8.3 (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 12 we verified that if the input is held in state $r = 0 \wedge s = 1$ long enough, then the output must stabilize 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

$$D \quad = \quad \bigcirc\wedge R \cdot (\supset\bigcirc L(1,1) \cdot \bigcirc\mathrm{id}(1)\ ,\ \supset\bigcirc L(6,1) \cdot \bigcirc\wedge R \cdot (\supset\bigcirc L(1,1) \cdot \bigcirc\mathrm{id}(1), \bigcirc\mathrm{id}(2))).$$

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 $\supset\bigcirc R \cdot D$:

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

By the Soundness Theorem 8.3, it follows 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. ∎

Theorem 8.3 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" with respect to delay bounds, which applies not only to the derivation of proper combinational behaviour but also to the derivation of bounds for oscillations.

Theorem 8.4 (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 $\vec{\delta}_1 : \theta_1 \vdash \vec{\delta}_2 : \theta_2$.

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

9 Conclusion and Future Work

We have introduced a natural real-time semantics for binary gate networks, which is equivalent to 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 guarantee 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.

The more general importance of this work lies in capturing a low level real-time model by a simple formal calculus that may be used as a convenient reference system to substitute the real-time model. We can develop circuit simulation and synthesis techniques, and reason about their correctness and completeness, relative to this calculus, without every time having to go through the details of quantifier reasoning and temporal inequations again. A particular advantage of our logical setting, as opposed to e.g. an algebraic approach, is that it obtains a quite natural separation of the intensional aspect of timing from the extensional aspect of function in a mathematically very precise sense: The timing is part of the derivations or proofs, and the function is part of the formulas. We can deal with the intensional structure by standard proof-theoretic methods, and with the extensional structure by standard model-theoretic means. We believe that the correspondence

	Calculus		Real-Time Semantics
Extensional Aspect	Formula	$\leftrightarrow$	Function
Intensional Aspect	Proof	$\leftrightarrow$	Timing

brought up in this work is a useful concept that can be used in many other situations as well.

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 paper 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.

References

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

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,

[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. Timing refinement of intuitionistic proofs and its application to the timing analysis of combinational circuits. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, Proceedings of the 5th International Workshop on Theorem Proving with Analytic Tableaux and Related Methods, pages 261–277. Springer LNAI 1071, 1996.

[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.

M. Mendler, M. Fairtlough: Ternary simulation: Refinement of binary functions or abstraction of real-time behaviour?
In M. Sheeran and S. Singh (eds.), Proc. Workshop on Designing Correct Circuits (DCC'96),
Springer Electronic Workshops in Computing, October 1996,