Design Rules and Abstractions (from branching and real time) DRAFT

Peter Sewell
The Computer Laboratory, University of Cambridge
Cambridge CB2 3QG, UK
Peter.Sewell@cl.cam.ac.uk

September 5, 1996

Abstract

Three simple models of synchronous hardware are given; using linear discrete, branching discrete and branching real time. A simple notion of abstraction is introduced, motivated by the need to ultimately view such models as scientific theories that make empirical predictions. It makes the significance of design rules explicit.

Two abstractions from the branching discrete to the linear discrete model are given. They shed some light on the roles of consistency, deadlock and determinacy. The stronger of the two depends on a notion of dynamic type for processes which ensures deadlock freedom.

A reasonably strong abstraction from the branching real to the branching discrete model is given. This depends on a finer notion of type which is a reasonably physically plausible formalisation of the timing properties of certain real components.

Contents

1 Introduction

In this paper we investigate some issues involved in the formal modelling of hardware. It is widely accepted that models at various levels of abstraction are required. When reasoning about a system it is desirable to use the most abstract model possible. It is a truism that actual hardware cannot be formally verified and therefore that our formal models must be related to the real world by some scientific work, enabling empirical predictions to be confidently made about the behaviour of actual hardware from the results of formal reasoning. This appears to be most straightforward for reasonably concrete models. We define a notion of abstraction between two models that has only enough structure to lift an empirical understanding of the more concrete to the more abstract.

A model is typically only valid for circuits that satisfy certain design rules. It is often unclear, however, exactly what is guaranteed by a particular design rule. This can sometimes be answered by a precise result in a more concrete model that is more widely valid. The design rule, together with assumptions on the concrete models of primitive components, can then be seen as necessary for a particular abstraction to exist.

These general statements are given some precise meaning by considering three models of simple synchronous circuits and the abstractions between them. In §2 and §3 we give a syntax suitable for describing circuits at the gate level and a standard linear discrete time model. In §4 we discuss design rules and abstractions between models in general.

In §5–7 we give a branching discrete time model and an abstraction that depends on the design rule 'All cycles contain a sequential gate'. This involves a class of deadlock-free parallel compositions. In §8 we recall the informal timing properties given in the manufacturer's databook [TI84] for certain 74LS devices. In §9–12 we give a branching real time model and an abstraction that depends on the design rule 'There are no long sequences of combinational gates' and a physically plausible formalisation of the timing properties.

It should be noted that we are only attempting to explore a few of the issues involved in relating abstract formal hardware models to actual hardware. In particular we consider only very simple synchronous circuits and quite abstract models. Our more concrete models have not been related to the concrete models in use, although they seem to be physically plausible. Most proofs are deferred to Appendix A. A short version of this work appeared at the 3rd Workshop on Designing Correct Circuits (DCC '96) [Sew96].

2 Circuit Syntax

We take a simple syntax of circuits with terms that are either primitive components from some set *prim* or the parallel composition of two terms:

$$\mathit{circ} ::= p \;\Big|\; \mathit{circ} \,\|\, \mathit{circ} \qquad p \in \mathit{prim}.$$

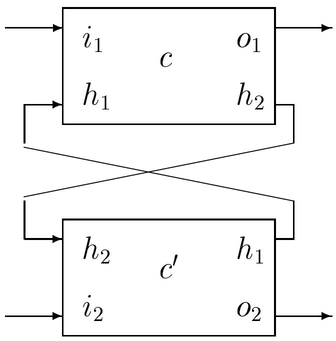
To each term we associate a pair of disjoint finite sets of port names, drawn from a set N, representing the available input and output ports. These are given for primitive components by a function sort $: \mathit{prim} \rightarrow (\mathcal{P}_{\text{fin}} N) \times (\mathcal{P}_{\text{fin}} N)$ which is lifted to all terms via a parallel composition of sorts:

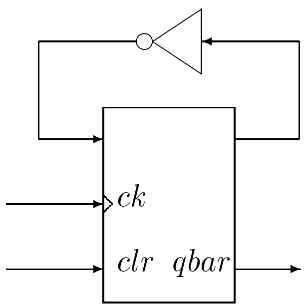
$$\begin{array}{rcl} \mathrm{sort}\, c \,\|\, c' & \stackrel{\mathit{def}}{=} & \mathrm{sort}\, c \,\|\, \mathrm{sort}\, c' \\ \text{where } \langle i, o\rangle \,\|\, \langle i', o'\rangle & \stackrel{\mathit{def}}{=} & \langle (i - o') \cup (i' - o), (o - i') \cup (o' - i)\rangle. \end{array}$$

We will consider only terms for which any input (resp. output) port is connected to at most one other port, which must be an output (resp. input). This condition, of 1:1 directional connection, is captured by D_0, where

$$D_0(c'') \iff \text{if } c \,\|\, c' \text{ is a subterm of } c'' \text{ then } \mathrm{sort}\, c \text{ and } \mathrm{sort}\, c' \text{ are composable}$$

and sorts $\langle i, o\rangle$, $\langle i', o'\rangle$ are composable iff $i \cap i' = o \cap o' = \{\}$. Set union between sets of ports will usually be elided. We will often refer to pairs of sorts of the form $\langle i_1 h_1, o_1 h_2\rangle$, $\langle h_2 i_2, h_1 o_2\rangle$, in which the sets are all supposed disjoint. The composition of terms c, c' with these sorts could be depicted as on the left below.





For example the primitive components might include gates (parameterised by ports) such as Not_{ab}, Nand_{abc} and DType with

$$
\begin{array}{rclrcl}
\mathrm{sort}\,\mathrm{Fork}_{abc} & \stackrel{def}{=} & \langle\{a\},\{b,c\}\rangle & \mathrm{sort}\,\mathrm{Pwr}_{a} & \stackrel{def}{=} & \langle\{\},\{a\}\rangle \\
\mathrm{sort}\,\mathrm{Nand}_{abc} & \stackrel{def}{=} & \langle\{a,b\},\{c\}\rangle & \mathrm{sort}\,\mathrm{Gnd}_{a} & \stackrel{def}{=} & \langle\{\},\{a\}\rangle \\
 & & & \mathrm{sort}\,\mathrm{CheckGnd}_{a} & \stackrel{def}{=} & \langle\{a\},\{\}\rangle \\
\mathrm{sort}\,\mathrm{Not}_{ab} & \stackrel{def}{=} & \langle\{a\},\{b\}\rangle & \mathrm{sort}\,\mathrm{DType} & \stackrel{def}{=} & \langle\{d,clr,ck\},\{q,qbar\}\rangle
\end{array}
$$

giving

$$\mathrm{sort}\,\mathrm{DType}\,\|\,\mathrm{Not}_{qd} \quad = \quad \langle\{ck,clr\},\{qbar\}\rangle$$

which could be depicted as on the right above.

This syntax is quite an abstract description of the structure of circuits — for certain problems, e.g. calculating capacitances between points, much more concrete geometrical descriptions would be required. We will be concerned with abstractions between the behaviours of terms in various models. We will not in this paper consider abstractions between different levels of structural description.

Henceforth we identify circuits with terms of the syntax. We will appeal to notions of the actual *instances* of a circuit c and of the *actual behaviour* of these instances. By the former we mean the physical objects that could be constructed following some 'standard practice' and that correspond to c. We will not attempt to be more precise about what this 'standard practice' is.

The choice of a binary parallel composition enables design rules, properties of ciruits and proofs of abstraction results to be expressed compositionally in a straightforward way. Parallel composition will be commutative and associative in all our models (up to a restriction on sorts), as expected. The choice of 1:1 directional connection is appropriate for the gate-based circuits we deal with. It would presumably not be appropriate for circuits with transistor-based structural descriptions. Having explicit sets of port names associated with each circuit simplifies our notation and definitions.

Definition A *model* $\mathcal{M}$ of the syntax will be a family of sets $\mathcal{M}_{io}$ indexed by the sorts and a sort-respecting function $[\![\_]\!]^{\mathcal{M}} : circ \to \mathcal{M}$. We will generally elide the indexing.

3 $\mathcal{L}$, a Linear Discrete Time Model

Work on formal hardware design commonly uses a linear discrete time model of circuit behaviour, here denoted by $\mathcal{L}$. In this model signals are modelled by functions from the naturals (representing time) to a set V of values and a circuit with n inputs and m outputs is modelled by a predicate over $(n+m)$-tuples of such functions. Composition is modelled by existential quantification and conjunction. To avoid a mass of notation port names and variables over signals will be identified — we suppose given some function $[\![\_]\!]^{\mathcal{L}}$ which for each primitive component gives a predicate over the variables in the sort thereof. This is extended to circuits by

$$[\![c \,\|\, c']\!]^{\mathcal{L}} \quad \stackrel{def}{=} \quad \exists a_1 \ldots a_n : \mathbb{N} \to V \quad . \, [\![c]\!]^{\mathcal{L}} \wedge [\![c']\!]^{\mathcal{L}}$$

where sort $c = \langle i_1 h_1, o_1 h_2 \rangle$, sort $c' = \langle i_2 h_2, o_2 h_1 \rangle$ and $h_1 h_2 = \{a_1 \ldots a_n\}$. For example, we might have the following, putting $V = \text{bool} = \{\text{t}, \text{f}\}$:

$$
\begin{array}{rcl}
[\![\text{Fork}_{abc}]\!]^{\mathcal{L}} & \stackrel{def}{=} & \forall t:\mathbb{N} \,.\, b(t) = a(t) \wedge c(t) = a(t) \\
[\![\text{Nand}_{abc}]\!]^{\mathcal{L}} & \stackrel{def}{=} & \forall t:\mathbb{N} \,.\, c(t) = \neg(a(t) \wedge b(t)) \\
[\![\text{Not}_{ab}]\!]^{\mathcal{L}} & \stackrel{def}{=} & \forall t:\mathbb{N} \,.\, b(t) = \neg a(t)
\end{array}
\qquad
\begin{array}{rcl}
[\![\text{Pwr}_{a}]\!]^{\mathcal{L}} & \stackrel{def}{=} & \forall t:\mathbb{N} \,.\, a(t) \\
[\![\text{Gnd}_{a}]\!]^{\mathcal{L}} & \stackrel{def}{=} & \forall t:\mathbb{N} \,.\, \neg a(t) \\
[\![\text{CheckGnd}_{a}]\!]^{\mathcal{L}} & \stackrel{def}{=} & \forall t:\mathbb{N} \,.\, \neg a(t)
\end{array}
$$

$$
\begin{array}{rcl}
[\![\text{DType}]\!]^{\mathcal{L}} & \stackrel{def}{=} & \forall t:\mathbb{N} \,.\, q(t+1) = (clr(t+1) \Rightarrow ((\neg ck(t) \wedge ck(t+1)) \Rightarrow d(t) \mid q(t)) \mid \text{f}) \\
& & \wedge\, \forall t:\mathbb{N} \,.\, qbar(t) = \neg q(t)
\end{array}
$$

Many combinational primitives can be described by a sort $\langle \{a_1 \ldots a_m\}, \{b_1 \ldots b_n\} \rangle$ and function $f : V^m \to V^n$ and given linear discrete time semantics

$$[\![\text{Comb}^{f}_{\vec{a}\vec{b}}]\!]^{\mathcal{L}} \stackrel{def}{=} \forall t:\mathbb{N} \,.\, \vec{b}(t) = f(\vec{a}(t)).$$

Many sequential primitives can be described by a sort $\langle \{a_1 \ldots a_m\}, \{b_1 \ldots b_n\} \rangle$, an $l:\mathbb{N}$ giving the size of the internal state, 'next-state' function $f : V^m \times V^l \to V^l$ and 'output' function $g : V^l \to V^n$. Their discrete time semantics is then

$$[\![\text{Seq}^{fg}_{\vec{a}\vec{b}}]\!]^{\mathcal{L}} \stackrel{def}{=} \exists \vec{h} : \mathbb{N} \to V^l \,.\, \forall t:\mathbb{N} \,.\, \vec{b}(t) = g(\vec{h}(t)) \wedge \vec{h}(t+1) = f(\vec{a}(t), \vec{h}(t)),$$

leaving the initial state unspecified.

4 Design Rules and Abstractions

The model $\mathcal{L}$ is quite abstract. This simplifies reasoning about the behaviour of large circuits but leaves the relationship between the model $[\![c]\!]^{\mathcal{L}}$ of a circuit c and the actual behaviour of instances of c somewhat unclear. In particular:

- We would only expect $[\![c]\!]^{\mathcal{L}}$ to correspond to the actual behaviour of instances of c for *some* c, i.e. those that satisfy *design rules* such as the following.

  0. All connections are 1:1 and directional.
  1. All cycles contain a sequential gate.
  2. There are no long sequences of combinational gates.
  3. No output is required to drive too many inputs.

  These are informal but easy to make precise (indeed 0 has been as D_0, 1 will be in §5 and 2 will be in §9). It is not clear exactly what is ensured by imposing these, nor whether they are sufficient.

- We have not said *how* $[\![c]\!]^{\mathcal{L}}$ corresponds to the actual behaviour of instances of c.

These points can be addressed by considering more concrete models and the abstraction relationships between them. We envisage a number of models, related to each other by some mathematical notion of abstraction. We suppose that one of these models is related to the actual behaviour of instances of circuits by some scientific work, giving us an 'empirical understanding' of it.

$$\begin{tikzcd}[row sep=large]
\text{Abstract model (e.g. } \mathcal{L}\text{)} \arrow[d, dash, "\textit{abstraction}"] \\
\text{Concrete model} \arrow[d, dash, "\textit{abstraction}"] \\
\text{Most Concrete model} \arrow[d, dash, "\textit{empirical understanding}"] \\
\textbf{Real world}
\end{tikzcd}$$

We will not discuss the basis or nature of this empirical understanding beyond formulating it in a way that focuses attention on the properties required of abstractions between models. We suppose that there is some set Ob of observations that may be made of the actual behaviour of instances of circuits. (We are not suggesting that it is feasible to define Ob, but only that it suggests a clear intuition about the desired statements of abstraction results.)

Definition An *empirical understanding* of a model $\mathcal{M}$ and design rule D (i.e. a subset of *circ*) is a function $obs_{\mathcal{M}} : \mathcal{M} \to \mathcal{P}(Ob)$ such that if $D(c)$ holds then the observations in $obs_{\mathcal{M}}([\![c]\!]^{\mathcal{M}})$ may with confidence (based on scientific work) be predicted of the actual behaviour of any instance of c.

For elements x of a model $\mathcal{M}$ we would like $obs_{\mathcal{M}}(x)$ to be as large as possible, so that we know as much as possible about the actual behaviour of instances of a circuit c simply from $[\![c]\!]^{\mathcal{M}}$. There is a trivial empirical understanding of any model, taking $obs_{\mathcal{M}}(x)$ to be empty.

Abstraction relationships can now be defined to enable an empirical understanding of a concrete model to be lifted to an abstract model.

Definition For models $\mathcal{A}$ and $\mathcal{C}$, design rules D_a and D_c, and a relation $Q \subseteq \mathcal{C} \times \mathcal{A}$ we say $\mathcal{A}, D_a$ is an *abstraction* of $\mathcal{C}, D_c$ along Q iff

$$\forall c \in circ \, . \, D_a(c) \Rightarrow (D_c(c) \wedge [\![c]\!]^{\mathcal{C}} \; Q \; [\![c]\!]^{\mathcal{A}})$$

It is then immediate that if $\mathcal{C}, D_c$ is empirically understood by $obs_{\mathcal{C}}(\_)$ and $\mathcal{A}, D_a$ is an abstraction of it along Q then $\mathcal{A}, D_a$ is empirically understood by

$$obs_{\mathcal{A}}(y) \quad \stackrel{def}{=} \quad \bigcap \{ obs_{\mathcal{C}}(x) \mid x \; Q \; y \}.$$

In other words, given the data below (which is not a commuting diagram)

$$\begin{tikzcd}
 & \mathcal{A} \arrow[dd, dash, "Q"] & \\
circ \arrow[ur, "{[\![_]\!]^{\mathcal{A}}}"] \arrow[dr, "{[\![_]\!]^{\mathcal{C}}}"'] & & \\
 & \mathcal{C} \arrow[r, "obs_{\mathcal{C}}(_)"'] & \mathcal{P}(Ob)
\end{tikzcd}$$

we can calculate the best empirical understanding $obs_{\mathcal{A}}(\_) : \mathcal{A} \to \mathcal{P}(Ob)$ of $\mathcal{A}, D_a$.

An abstraction result thus reduces the question of how $[\![c]\!]^{\mathcal{A}}$ corresponds to the actual behaviour of instances of c to the (presumably better understood) question of how $[\![c]\!]^{\mathcal{C}}$ does. It lets us work only at the abstract level — typically $[\![c]\!]^{\mathcal{A}}$ will be easier to calculate than $[\![c]\!]^{\mathcal{C}}$ and having done so we know $[\![c]\!]^{\mathcal{C}} \in \{\, x \in \mathcal{C} \mid x \; Q \; [\![c]\!]^{\mathcal{A}} \,\}$. Note that there may be many abstraction relations between two models. In general one would be preferred if for c such that $D_a(c)$ the set $\{\, x \in \mathcal{C} \mid x \; Q \; [\![c]\!]^{\mathcal{A}} \,\}$ is small and/or easy to describe.

In the remainder of this paper we give two abstractions from a branching discrete time model $\mathcal{B}$ to $\mathcal{L}$ and one from a branching real time model $\mathcal{R}$ to $\mathcal{B}$.

$\mathcal{L}$, a linear discrete time model
$\prec$
$\prec^{nt}$ requiring 'All cycles contain a sequential gate.'
$\mathcal{B}$, a branching discrete time model
$<$ requiring 'There are no long sequences of combinational gates.'
$\mathcal{R}$, a branching real time model

One of the former ($\prec$) is trivial. The others have similar developments. In each case we formalise a design rule (1 and 2 respectively) and recall the informal reason why imposing it ensures that circuits behave correctly. This involves properties of the behaviour of primitive components. We introduce more concrete models ($\mathcal{B}$ and $\mathcal{R}$ respectively) in which these properties can be expressed. The desired abstraction relations are stated and the proofs that they are indeed abstraction relations make the informal reasoning precise. We thus give a precise understanding of what is ensured by imposing design rules 1 and 2.

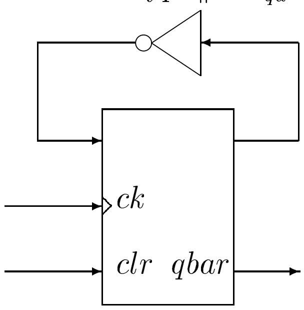
5 Design Rule 1: 'All cycles contain a sequential gate.'

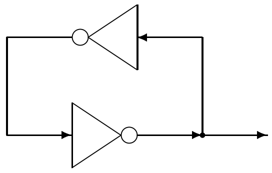
Simple gates can be divided into two groups, the combinational (e.g. Nand_{abc}, Not_{ab}) and the sequential (e.g. a latch or a DType in certain environments). The behaviours of these gates have the following informal property:

> In any clock period any values may be applied to the inputs of a gate. Further, for sequential gates the values on outputs in a given clock period are independent of the inputs in that clock period. (1)

Consider the circuits below.

$c \stackrel{def}{=} \text{DType} \parallel \text{Not}_{qd}$ $\qquad$ $c' \stackrel{def}{=} \text{Not}_{bd} \parallel \text{Not}_{da} \parallel \text{Fork}_{abc}$





The first, c, obeys Design Rule 1, as the only cycle contains a sequential gate — the DType. This, together with property (1), suggests that in each clock period there can be consistent values on the internal ports q and d. We would thus expect $[\![c]\!]^{\mathcal{L}}$ to correspond to the actual behaviour of instances of c. On the other hand the actual behaviour of instances of c' may not correspond to any element of $\mathcal{L}$, let alone $[\![c']\!]^{\mathcal{L}}$, as c' does not obey Design Rule 1.

In the rest of this section we make Design Rule 1 precise. In §6 and §7 we formalise property (1) and make the above informal reasoning precise.

The informal statement of Design Rule 1 is global, referring to all cycles within a circuit. It is more convenient to have a definition that is inductive on the structure of circuits. To state this we introduce a simple notion of type. The types of a circuit with sort $\langle i, o\rangle$ will be relations $T \subseteq i \times o$, thought of as giving the admissible direct (i.e., via combinational gates) connectivity within the circuit. More precisely, for a circuit c and $a \in i$, $b \in o$, if c has type T and $\neg(a\ T\ b)$ then there will be no direct connection from a to b within c. The types $i \times o$ and $\{\}$ correspond to purely combinational and sequential circuits respectively. We write T^+ for the transitive closure of a relation T.

Definition If T, T' are types for sorts $\langle i_1 h_1, o_1 h_2\rangle, \langle i_2 h_2, o_2 h_1\rangle$ then they are *composable* if $\neg\exists a\ .\ a\ (T \cup T')^+\ a$, in which case $T \parallel T' \stackrel{def}{=} (T \cup T')^+ \cap (i_1 i_2 \times o_1 o_2)$.

We suppose that each primitive component is given some type. The rules for typing circuits are those below.

$$\frac{}{p : T}\, p : T \text{ given} \qquad \frac{c : T \quad c' : T'}{c \parallel c' : T \parallel T'}\, T, T' \text{ composable}$$

For example if $\mathrm{Not}_{ab} : \{\langle a, b\rangle\}$ and $\mathrm{DType} : \{\langle clr, q\rangle, \langle clr, qbar\rangle, \langle ck, q\rangle, \langle ck, qbar\rangle$ are given then $\mathrm{DType} \parallel \mathrm{Not}_{qd} : \{\langle ck, qbar\rangle, \langle clr, qbar\rangle\}$.

Design Rule 1, 'All cycles contain a sequential gate', can now be expressed:

$$D_1(c) \iff D_0(c) \text{ and } c \text{ is typable.}$$

For example $D_1(\mathrm{DType} \parallel \mathrm{Not}_{qd})$ holds whereas $D_1(\mathrm{Not}_{bd} \parallel \mathrm{Not}_{da} \parallel \mathrm{Fork}_{abc})$ does not.

We will use the following fact about composable types.

Lemma 1 *If types T, T' for sorts $\langle i_1 h_1, o_1 h_2\rangle, \langle i_2 h_2, o_2 h_1\rangle$ are composable then the set of ports $h_1 h_2 o_1 o_2$ can be ordered $c_1 \ldots c_n$ such that $\forall p, q \in 1 \ldots n\ .\ p < q \Rightarrow \neg(c_q\ (T \cup T')^+\ c_p)$.*

PROOF Trivial. □

6 $\mathcal{B}$, a Branching Discrete Time Model

In order to express property (1) we introduce a more concrete model.

The model $\mathcal{L}$ abstracts from many aspects of circuit behaviour — real time, voltages, impedances etc. In particular it abstracts from the internal states of circuits. This might be contrasted with the behaviour of a simulator which maintains a state, calculating a/the state for a succeeding time step from the current state and inputs. Work on concurrency has considered a variety of models that abstract from internal state to different extents, from linear time (or trace based) to branching time (or bisimulation) — see [Gla90] for an overview. In this section a branching time model is given. We assume familiarity with transition systems, bisimulation and SCCS [Mil83, Mil89].

Definition A transition system S over labels L is a pair of a set S (of states) and a family of relations $\xrightarrow{l} \subseteq S \times S \mid l \in L$. We write $\longrightarrow$ for $\cup_{l \in L} \xrightarrow{l}$. Strong bisimulation $\sim$ over S is the largest equivalence relation over S such that if $s \sim t$ and $s \xrightarrow{l} s'$ then $\exists t' \, . \, t \xrightarrow{l} t' \sim s'$.

Definition For a set $\mathcal{V}$ the model $BR(\mathcal{V})$ consists of the transition systems $BR(\mathcal{V})_{io}$, where these are the strong bisimulation quotients of sufficiently large[1] transition systems over the labels $io \rightarrow \mathcal{V}$. Parallel composition is basically SCCS synchronisation and hiding, i.e. for s, s' of sorts $\langle i_1 h_1, o_1 h_2 \rangle$, $\langle i_2 h_2, o_2 h_1 \rangle$ the transitions of $s \parallel s'$ are given by

$$\frac{s \xrightarrow{\alpha} \hat{s} \quad s' \xrightarrow{\beta} \hat{s}'}{s \parallel s' \xrightarrow{\alpha \parallel \beta} \hat{s} \parallel \hat{s}'} \forall a \in h_1 h_2 \, . \, \alpha(a) = \beta(a),$$

where the composition $\alpha \parallel \beta : i_1 i_2 o_1 o_2 \rightarrow \mathcal{V}$ of $\alpha : i_1 h_1 o_1 h_2 \rightarrow \mathcal{V}$ and $\beta : i_2 h_2 o_2 h_1 \rightarrow \mathcal{V}$ is defined by

$$\begin{aligned} (\alpha \parallel \beta)(a) &= \alpha(a) \text{ for } a \in i_1 o_1 \\ (\alpha \parallel \beta)(a) &= \beta(a) \text{ for } a \in i_2 o_2. \end{aligned}$$

Given $\alpha : j \rightarrow \mathcal{V}$ and $\beta : j' \rightarrow \mathcal{V}$ for disjoint sets of names j, j' we write $\alpha\beta$ for their source tupling.

Definition The branching discrete time model is $\mathcal{B} \stackrel{def}{=} BR(V)$ with a sort-respecting semantic function $[\![\_]\!]^{\mathcal{B}}$ supposed given for primitive components. $\mathcal{B}_{io}$ is thus a transition system with labels that give a value for each port in $i \cup o$. The transition labels will be ranged over by $\alpha, \beta, \gamma, \mu, \nu$.

SCCS notation will be used for examples, taking $\langle Act, \times, 1, \_ \rangle$ to be the free abelian group over $N \times V$ and writing states with sort $\langle i, o \rangle$ as processes with sort (in the sense of [Mil83]) $In_i \times Out_o$, where

$$\begin{aligned} In_{\vec{a}} &\stackrel{def}{=} \{(a_1)_{v_1} \times \ldots \times (a_n)_{v_n} \mid \vec{v} \in V^n\} \\ Out_{\vec{b}} &\stackrel{def}{=} \{\overline{(b_1)_{v_1}} \times \ldots \times \overline{(b_m)_{v_m}} \mid \vec{v} \in V^m\}. \end{aligned}$$

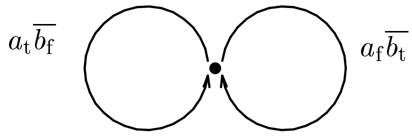
Parallel composition is just $s \parallel s' = (s \times s') \restriction In_{i_1 i_2} \times Out_{o_1 o_2}$. The isomorphisms between $In_{\vec{a}}$ (resp. $Out_{\vec{b}}$) and the set of functions from $\vec{a}$ to V (resp. $\vec{b}$ to V) will be elided.

[1] i.e., containing the denotations of all primitive components and closed under parallel composition

For example we might have

$$
\begin{array}{rcl@{\qquad}rcl}
[\![\mathrm{Fork}_{abc}]\!]^{\mathcal{B}} & \overset{def}{=} & \mathit{fix}\ X.\sum_{x\in\mathrm{bool}} a_x\overline{b_x c_x} : X & [\![\mathrm{Pwr}_a]\!]^{\mathcal{B}} & \overset{def}{=} & \mathit{fix}\ X.\overline{a_\mathrm{t}} : X \\
[\![\mathrm{Nand}_{abc}]\!]^{\mathcal{B}} & \overset{def}{=} & \mathit{fix}\ X.\sum_{x,y\in\mathrm{bool}} a_x b_y\overline{c_{\neg(x\wedge y)}} : X & [\![\mathrm{Gnd}_a]\!]^{\mathcal{B}} & \overset{def}{=} & \mathit{fix}\ X.\overline{a_\mathrm{f}} : X \\
[\![\mathrm{Not}_{ab}]\!]^{\mathcal{B}} & \overset{def}{=} & \mathit{fix}\ X.\sum_{x\in\mathrm{bool}} a_x\overline{b_{\neg x}} : X & [\![\mathrm{CheckGnd}_a]\!]^{\mathcal{B}} & \overset{def}{=} & \mathit{fix}\ X.a_\mathrm{f} : X \\
[\![\mathrm{DType}]\!]^{\mathcal{B}} & \overset{def}{=} & \sum_{xyz} D_{xyz} \\
\text{where}\quad D_{xyz} & \overset{def}{=} & \sum_{w',x',y'\in\mathrm{bool}} \mathit{clr}_{w'}\mathit{ck}_{x'}d_{y'}\overline{q_{z'}\mathit{qbar}_{\neg z'}} : D_{x'y'z'} \\
\text{and} & & z' = (w' \Rightarrow ((\neg x \wedge x') \Rightarrow y \mid z) \mid \mathrm{f}).
\end{array}
$$

The transition system $[\![\mathrm{Not}_{ab}]\!]^{\mathcal{B}}$ could be depicted as below.



For the general combinational and sequential primitives we take

$$
\begin{array}{rcl}
[\![\mathrm{Comb}^{f}_{\vec{a}\vec{b}}]\!]^{\mathcal{B}} & \overset{def}{=} & \mathit{fix}\ X.\ \sum_{v\in V^m} \vec{a}_v\overline{\vec{b}_{f(v)}} : X \\
[\![\mathrm{Seq}^{fg}_{\vec{a}\vec{b}}]\!]^{\mathcal{B}} & \overset{def}{=} & \sum_{s\in V^l} P_s \quad \text{where} \quad P_s \overset{def}{=} \sum_{v\in V^m} \vec{a}_v\overline{\vec{b}_{g(s)}} : P_{f(v,s)}.
\end{array}
$$

This $[\![\mathrm{Seq}^{fg}_{\vec{a}\vec{b}}]\!]^{\mathcal{B}}$ is only nondeterministic at the first state so we are not using much of the expressive power of the branching time model. Modelling arbiters or faulty components would make more use of the expressiveness of the model.

The set of infinite traces of a state s will be written $\mathit{itr}(s)$. We elide the isomorphism between the sets of infinite traces for sort $\langle i, o\rangle$ (i.e. subsets of $\mathbb{N}\to io \to V$) and the predicates on the variables (of type $\mathbb{N}\to V$) in io. For the above examples of primitive components $\mathit{itr}([\![p]\!]^{\mathcal{B}}) = [\![p]\!]^{\mathcal{L}}$.

It is well known that it is not necessary to use as fine an equivalence as bisimulation in order to be sensitive to deadlock and be a congruence for parallel composition. Bisimulation is chosen to make the definitions in the next section and the relationship with the real time model simple. We will give abstractions from $\mathcal{B}$ all the way to the linear time $\mathcal{L}$, skipping over intermediates such as failure equivalence classes of transition systems. One can argue that $\mathcal{B}$ is closer than $\mathcal{L}$ to the behaviour that might be exhibited by a stochastic circuit simulator. $\mathcal{L}$ abstracts from the internal states of circuits, whereas we might expect a simulator to maintain them — in a given state, (perhaps randomly) choosing a succeeding state and valuation for ports from those allowed by the primitive components in that state. Such a simulator would therefore detect (probabilistically) the deadlocks of the example in §7.4. One might also argue that a proper treatment of nondeterminism will be essential when we get to very concrete physical models.

7 Abstraction from $\mathcal{B}$ to $\mathcal{L}$

There may be many abstraction relations between two models, with differing strengths and requiring different design rules. For example there are the following natural ones between $\mathcal{B}$ and $\mathcal{L}$.

Definition For $s \in \mathcal{B}$ and $x \in \mathcal{L}$ of the same sort

$$\begin{array}{rcl} s \prec x & \textit{iff} & itr(s) = x \\ s \prec^{nt} x & \textit{iff} & itr(s) = x \text{ and } s \text{ is nonterminating.} \end{array}$$

For an abstraction result along $\prec$ no design rules or properties of primitive components are required.

Theorem 1 *If for all primitive components p we have $[\![p]\!]^{\mathcal{B}} \prec [\![p]\!]^{\mathcal{L}}$ then $\mathcal{L}, D_0$ is an abstraction of $\mathcal{B}, D_0$ along $\prec$.*

Proof Trivial induction on circuits using the fact that for states s, s' of composable sorts in $\mathcal{B}$ we have $itr(s \parallel s') = itr(s) \parallel itr(s')$. $\square$

For an abstraction result along $\prec^{nt}$ we impose Design Rule 1 and a formalisation of property (1).

7.1 Interpretation of types in $\mathcal{B}$

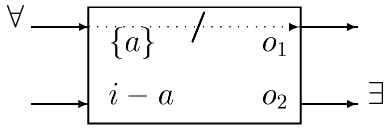
Types can be interpreted in the branching time model, formalising property (1) and generalising it to arbitrary circuits:

Definition If $T \subseteq i \times o$ we say a state $s \in \mathcal{B}_{io}$ *inhabits* T if for all $a \in i$, if

$$s \longrightarrow^* s' \xrightarrow{\alpha\alpha_2\beta_1\beta_2}$$

$$\text{where} \quad \begin{array}{ll} \alpha : \{a\} \to V & \beta_1 : \{b \mid \neg(a\ T\ b)\} \to V \\ \alpha_2 : (i - \{a\}) \to V & \beta_2 : \{b \mid a\ T\ b\} \to V \end{array}$$

then for all $\alpha' : \{a\} \to V$ there exists $\beta_2' : \{b \mid a\ T\ b\} \to V$ such that $s' \xrightarrow{\alpha'\alpha_2\beta_1\beta_2'}$. This could be depicted as below, in which $o_1 = \{b \mid \neg(a\ T\ b)\}$ and $o_2 = \{b \mid a\ T\ b\}$.



For combinational circuits with type $i \times o$ this definition is similar to strong consistency, saying that for all inputs there is an output. For sequential circuits with type $\{\}$ it is a form of receptivity, saying that for any output that can occur all inputs are permitted. More precisely, suppose $s \longrightarrow^* s' \xrightarrow{\alpha\beta}$. If s inhabits $i \times o$ then $\forall \alpha' \,.\, \exists \beta' \,.\, s' \xrightarrow{\alpha'\beta'}$ and if s inhabits $\{\}$ then $\forall \alpha' \,.\, s' \xrightarrow{\alpha'\beta}$.

Note that if $T \subseteq T' \subseteq i \times o$ and a nonterminating state s inhabits T then it also inhabits T' but that states may not inhabit unique minimal types. For example the state with sort $\langle \{a\}, \{b, c\} \rangle$ defined by

$$\textit{fix}\ X.\ \sum_{x,y \in \text{bool}} a_x \overline{b_y} \overline{c_{(x \operatorname{xor} y)}} : X$$

inhabits $\{\langle a,b\rangle\}$ and $\{\langle a,c\rangle\}$ but not $\{\}$. Note also that determinacy (in any of the senses of §7.5) and nontermination are not implied by inhabitation.

The formalisation of property (1) for a primitive component $p\!:\!T$ is that $[\![p]\!]^{\mathcal{B}}$ inhabits T and is nonterminating. This holds for the examples given.

In this definition and henceforth we implicitly identify the units of discrete time with the period of some global clock.

Remark This notion of type does not seem to fit into the interaction category framework of Abramsky [Abr94], essentially because the interpretations of types are all distinct and composability is not an involution on types. Consider compositions of sorts $\langle i_1h_1, o_1h_2\rangle, \langle i_2h_2, o_2h_1\rangle$. If A is a set of types for one of these we define $A^{\perp}$ to be $\{T' \mid \forall T \in A \; . \; T,T' \text{ are composable}\}$ and note that for a type T the set $\{T\}^{\perp}$ may *not* be a singleton.

7.2 Abstraction Result

The important property of states inhabiting certain types is the following.

Lemma 2 *If states s, s' of sorts $\langle i_1h_1, o_1h_2\rangle, \langle i_2h_2, o_2h_1\rangle$ inhabit composable types T, T' then $s \,\|\, s'$ inhabits $T \,\|\, T'$. Further, if s and s' are nonterminating then so is $s \,\|\, s'$.*

PROOF Sketch: consider $a \in i_1i_2$. By Lemma 1 the set $\{c \in h_1h_2 \mid a \; (T \cup T')^+ \; c\}$ can be linearly ordered $c_1 \ldots c_n$ such that $p < q \Rightarrow \neg c_q \; (T \cup T')^+ \; c_p$. For any particular transition $s \,\|\, s' \longrightarrow^* \hat{s} \,\|\, \hat{s}' \xrightarrow{\gamma}$ and new value at a the values at $c_1 \ldots c_n$ can be chosen successively. □

Theorem 2 *If for all primitive components $p\!:\!T$ we have $[\![p]\!]^{\mathcal{B}} \prec^{nt} [\![p]\!]^{\mathcal{L}}$ and $[\![p]\!]^{\mathcal{B}}$ inhabits T then $\mathcal{L}, D_1$ is an abstraction of $\mathcal{B}, D_0$ along $\prec^{nt}$.*

PROOF An easy induction on the structure of c using Lemma 2 shows that for all circuits c, if $c\!:\!T$ (so $D_1(c)$ holds), then $[\![c]\!]^{\mathcal{B}}$ is nonterminating, inhabits T and has the infinite traces $[\![c]\!]^{\mathcal{L}}$. □

7.3 Consistency

Design rules D_0 and D_1 are expressed almost directly on circuits. An alternative is to express design rules as conditions on the models of circuits. It has been noted (in e.g. [CGM87, BNWV92]) that some circuits have 'inconsistent' denotations in $\mathcal{L}$. For example

$$[\![\mathrm{Pwr}_a \,\|\, \mathrm{CheckGnd}_a]\!]^{\mathcal{L}} \;\; = \;\; \mathrm{f}$$

(the more usual $\mathrm{Pwr}_a \,\|\, \mathrm{Gnd}_a$ is outlawed by D_0).This is particularly problematic when using logical implication as an implementation relation, all specifications being satisfied by such a circuit. Accordingly it has been suggested (in [Mel93, §4.2.2]) that a finer implementation relation that requires 'consistency' is used, i.e. c implements *Spec* iff $([\![c]\!]^{\mathcal{L}} \neq \mathrm{f}) \wedge ([\![c]\!]^{\mathcal{L}} \Rightarrow \mathit{Spec})$. This can be equivalently viewed as imposing a design rule

$$D_{con}(c) \quad \Longleftrightarrow \quad [\![c]\!]^{\mathcal{L}} \neq \mathrm{f}$$

— the model is only regarded as valid for circuits c such that $D_{con}(c)$. In our setting with distinguished inputs and outputs it is more natural to use, for c of sort $\langle \vec{a}, \vec{b} \rangle$, the stronger condition (suggested by Fourman)

$$D_{dcon}(c) \iff \forall \vec{a} \,.\, \exists \vec{b} \,.\, [\![c]\!]^{\mathcal{L}}.$$

This is strictly weaker than D_1 (assuming for all primitives $p\!:\!T$ that $[\![p]\!]^{\mathcal{L}} = itr([\![p]\!]^{\mathcal{B}})$, $[\![p]\!]^{\mathcal{B}}$ inhabits T and $[\![p]\!]^{\mathcal{B}}$ is nonterminating). It does not rule out all pathologies, for example D_{dcon} holds of $\mathrm{Not}_{bd} \,\|\, \mathrm{Not}_{da} \,\|\, \mathrm{Fork}_{abc}$.

7.4 Deadlock

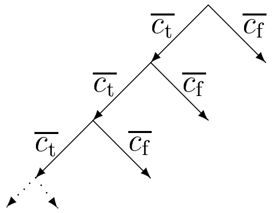
We give another pathological example circuit to illustrate the difference between $\mathcal{L}$ and $\mathcal{B}$. Take a primitive component Q with sort $\langle \{a\}, \{b\} \rangle$ and semantics

$$\begin{array}{rcl} [\![\mathrm{Q}]\!]^{\mathcal{B}} & \stackrel{def}{=} & \mathit{fix}\, X.a_{\mathrm{t}}\overline{b_{\mathrm{t}}} : X + a_{\mathrm{f}}\overline{b_{\mathrm{f}}} : (\mathit{fix}\, Y.a_{\mathrm{t}}\overline{b_{\mathrm{f}}} : Y + a_{\mathrm{f}}\overline{b_{\mathrm{t}}} : Y) \\ [\![\mathrm{Q}]\!]^{\mathcal{L}} & \stackrel{def}{=} & itr([\![\mathrm{Q}]\!]^{\mathcal{B}}) \end{array}$$

Intuitively Q behaves like a wire from a to b until it receives a value f, thereafter behaving like an inverter. Consider a circuit $\mathrm{Q} \,\|\, \mathrm{Fork}_{bac}$. Then

$$\begin{array}{rcl} [\![\mathrm{Q} \,\|\, \mathrm{Fork}_{bac}]\!]^{\mathcal{B}} & = & \mathit{fix}\, Z.\overline{c_{\mathrm{t}}} : Z + \overline{c_{\mathrm{f}}} : 0 \\ [\![\mathrm{Q} \,\|\, \mathrm{Fork}_{bac}]\!]^{\mathcal{L}} & = & \forall t\!:\!\mathbb{N} \,.\, c(t) \end{array}$$

the first of which could be depicted as



Note that $D_{dcon}(\mathrm{Q} \,\|\, \mathrm{Fork}_{bac})$ holds, however $D_1(\mathrm{Q} \,\|\, \mathrm{Fork}_{bac})$ cannot — the only types inhabited by $[\![\mathrm{Q}]\!]^{\mathcal{B}}$,$[\![\mathrm{Fork}_{bac}]\!]^{\mathcal{B}}$ are $\{\langle a, b \rangle\}$, $\{\langle b, a \rangle, \langle b, c \rangle\}$ respectively and compositions of components with these types do not satisfy D_1.

The deadlocks of behaviours such as $[\![\mathrm{Q} \,\|\, \mathrm{Fork}_{bac}]\!]^{\mathcal{B}}$ can be interpreted in two ways. In one they are regarded as symptoms of the fact that this is not a circuit that we expect the model $\mathcal{B}$ to be valid for, so $[\![\mathrm{Q} \,\|\, \mathrm{Fork}_{bac}]\!]^{\mathcal{B}}$ may be nonsense. This is the view we adopted for the abstraction result, where deadlocks were viewed as arising from the existence of feedback loops containing no delay elements. In the other view the model is taken to be valid until the occurrence of a deadlock. This allows the failure of the validity of the model (perhaps due to some violation of constraints on the environment of a component) to be encoded within $[\![\_]\!]^{\mathcal{B}}$.

Note that D_1 is not a necessary condition for deadlock freedom, e.g. $[\![\mathrm{Not}_{ab} \,\|\, \mathrm{Not}_{ba}]\!]^{\mathcal{B}}$ is nonterminating. We surmise that it is necessary in the following sense.

Conjecture 3 *If $\neg D_1(c)$ then there is some choice of $[\![\_]\!]^{\mathcal{B}}$ for the primitive components such that for each $p\!:\!T$ $[\![p]\!]^{\mathcal{B}}$ inhabits T and is nonterminating but $[\![c]\!]^{\mathcal{B}}$ is terminating.*

7.5 Determinacy

The branching time model admits various intuitive definitions of *determinacy*. These differ widely so it seems worthwhile to discuss them briefly for reference.

Definition A state s_0 with sort $\langle i,o\rangle$ is:

1. SCCS-determinate iff $s_0 \longrightarrow^* s \xrightarrow{\gamma} s' \wedge s \xrightarrow{\gamma} s'' \Rightarrow s' = s''$, i.e. if the state after a transition is determined by the state before and the values on inputs and outputs. This is the standard concurrency-theoretic definition. (Recall that we are working up to strong bisimulation.)

2. early-determinate iff $s_0 \longrightarrow^* s \Rightarrow \exists! \beta : o \to V \;.\; \exists \alpha : i \to V \;.\; s \xrightarrow{\alpha\beta}$, i.e. if in any state there is a unique tuple of values that can appear on the outputs.

3. late-determinate iff $s_0 \longrightarrow^* s \Rightarrow \forall \alpha : i \to V \;.\; \exists! \beta : o \to V \;.\; s \xrightarrow{\alpha\beta}$, i.e. if in any state, for any values on the inputs, there is a unique tuple of values that can appear on the outputs.

4. ω-late-determinate iff $\forall \vec{a} : \mathbb{N} \to V \;.\; \exists! \vec{b} : \mathbb{N} \to V \;.\; s_0 \xrightarrow{\vec{a}_0 \vec{b}_0} \xrightarrow{\vec{a}_1 \vec{b}_1} \cdots$, taking $i = \vec{a}$ and $o = \vec{b}$. I.e., if for any infinite sequence of values on the inputs there is a unique infinite sequence of values that can appear on the outputs. This is purely a property of the infinite traces of s_0 . A slightly weaker determinacy property, $\forall \vec{a} \;.\; \exists_{\leq 1} \vec{b} \;.\; s_0 \xrightarrow{\vec{a}_0 \vec{b}_0} \xrightarrow{\vec{a}_1 \vec{b}_1} \cdots$, is introduced in [BNWV92].

Proposition 4 *The only implications between the above, assuming that s_0 is nonterminating and inhabits type $i \times o$, are $1 \Leftarrow 4 \Rightarrow 3 \Leftarrow 2$.*

PROOF Straightforward. $\square$

For example, $[\![\mathrm{Q}]\!]^{\mathcal{B}}$, $[\![\mathrm{Fork}_{bac}]\!]^{\mathcal{B}}$ and $[\![\mathrm{Q} \,\|\, \mathrm{Fork}_{bac}]\!]^{\mathcal{B}}$ are SCCS-determinate and ω-late-determinate. $[\![\mathrm{Q}]\!]^{\mathcal{B}}$ and $[\![\mathrm{Fork}_{bac}]\!]^{\mathcal{B}}$ are also late-determinate. $[\![\mathrm{Comb}^{f}_{\vec{a}\vec{b}}]\!]^{\mathcal{B}}$ is SCCS-determinate, late-determinate and ω-late-determinate; depending on f it may also be early-determinate. $[\![\mathrm{Seq}^{fg}_{\vec{a}\vec{b}}]\!]^{\mathcal{B}}$ may not be determinate by any of the definitions, depending on f and g, although after the first transition it will be determinate by all four.

We expect that for any typable circuit c composed of standard components (e.g. instances of $\mathrm{Comb}^{f}_{\vec{a}\vec{b}}$ and $\mathrm{Seq}^{fg}_{\vec{a}\vec{b}}$) with standard branching time models (e.g. $[\![\mathrm{Comb}^{f}_{\vec{a}\vec{b}}]\!]^{\mathcal{B}}$ and $[\![\mathrm{Seq}^{fg}_{\vec{a}\vec{b}}]\!]^{\mathcal{B}}$) $[\![c]\!]^{\mathcal{B}}$ will be SCCS-, late- and ω-late- determinate after its first transition. For these circuits the only source of nondeterminacy is under-initialisation. More interesting components, e.g. arbiters, and/or more interesting models of standard components, e.g. models of components that may fail, would introduce nondeterminacy at all transitions.

Given nonterminating SCCS-determinate states s and s' it is standard that $itr(s) = itr(s')$ iff $s = s'$. It follows that for nonterminating SCCS- (or ω-late-) determinate states the predicate 'inhabits T' can be expressed simply in terms of their infinite trace sets. However, none of the above notions of determinacy are preserved by parallel composition.

8 Informal Timing Properties

In the remainder of this paper we consider Design Rule 2: 'There are no long sequences of combinational gates'. Imposing this ensures that, loosely speaking, the timing behaviour of actual circuits will be correct, e.g. that setup and hold times will be met. To obtain a convincing abstraction result depending on Design Rule 2 we must, therefore, consider the timing properties of primitive components. The basic problem is to state properties of the real-time behaviour of circuits that:

- For primitive components are expected to hold of standard device physics models, and are loose enough to admit manufacturing variations.
- Are strong enough to admit a strong abstraction result to a discrete-time model.
- Are composable (for circuits satisfying the design rule).

The abstraction result given below (Theorem 3) should be regarded only as an approximation to the kind of result desired — as will be discussed in the conclusion, the first condition has been relaxed slightly in order to obtain a reasonably tractable proof.

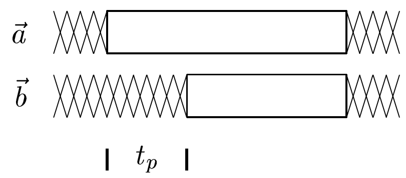
For simplicity we suppose that primitive components are either purely combinational, of the form $\mathrm{Comb}^{f}_{\vec{a}\vec{b}}$, or purely sequential, of the form $\mathrm{Seq}^{fg}_{\vec{a}\vec{b}}$. We will give these timing properties based on those informally specified in the databook [TI84] for two particular TTL devices, a 74LS00 NAND gate and a 74LS171 D-type (with 'clear' input tied to V_{cc} and 'clock' tied to a global square-wave clock). In this section we recall these informal specifications.

8.1 Combinational Primitives

We assume that combinational primitives have a maximum propagation delay of $t_p > 0ns$ and a minimum propagation delay of $0ns$. Informally:

> If the inputs of a combinational primitive are stable for an interval of length t_p or more then its outputs will be stable, and have the correct values, for the subinterval starting t_p later. (2)

This can be depicted by the timing diagram below, for a primitive with sort $\langle\vec{a},\vec{b}\rangle$. In these diagrams a double line denotes a requirement or guarantee of a stable signal, hatched lines denote 'don't care'.



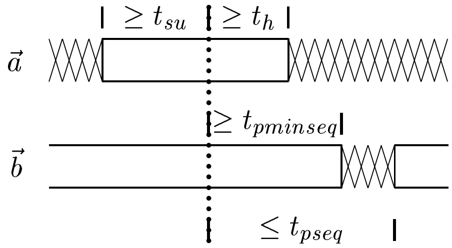
For a 74LS00 NAND the databook [TI84] specifies maximum propagation delays $t_{PLH} = 15ns$, $t_{PHL} = 15ns$ for low-to-high and high-to-low input transitions. For our results to be relevant to actual circuits constructed using these devices we would therefore require $t_p \geq \max(t_{PLH}, t_{PHL}) = 15ns$. [TI84] does not specify a minimum propagation delay. Our choice of $0ns$ is uncontroversial.

8.2 Sequential Primitives

We assume that sequential primitives have minimum setup and hold times t_{su} and t_h (for the data inputs w.r.t. clock edges) and maximum and minimum propagation delays t_{pseq} and $t_{pminseq}$ (from clock edges to data outputs). We take a clock period t_c and suppose that these quantities are non-negative and satisfy $t_{pminseq} < t_{pseq}$, $t_{pseq} + t_p + t_{su} \leq t_c$ and $0 < t_h \leq t_{pminseq}$. Informally:

> If the inputs of a sequential primitive are stable from t_{su} before to t_h after a clock tick (and similarly for all preceding clock ticks) then the outputs will be stable, and have the correct values, from t_{pseq} after to $t_c + t_{pminseq}$ after that clock tick. Moreover, the values on the outputs in a given clock period are independent of the values on the inputs in that clock period. (3)

The first clause can be depicted by the timing diagram below, for a primitive with sort $\langle \vec{a}, \vec{b} \rangle$.

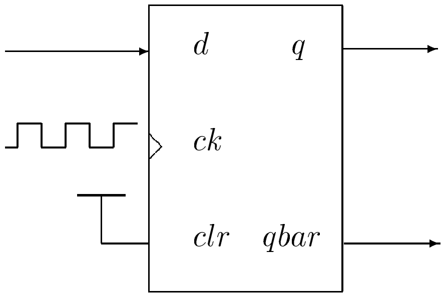


This is based on the specification in [TI84] of an edge triggered D-type from a 74LS171 device, tying the 'clear' input to V_{cc} and the 'clock' input to a global clock with period t_c. There $t_{su} = 20ns$ and $t_h = 5ns$. Two propagation delays $t_{PLH} = 25ns$, $t_{PHL} = 30ns$ are given so we could take $t_{pseq} = 30ns$. The minimum propagation delay $t_{pminseq}$ is not specified. We require $t_h \leq t_{pminseq}$, for example $t_{pminseq} = 5ns$, which seems reasonably plausible. We take for example a clock period $t_c = 100ns$. We take such a D-type to be a new primitive component DType′, with sort $\langle \{d\}, \{q, qbar\} \rangle$ and discrete time semantics

$$\llbracket \mathrm{DType}' \rrbracket^{\mathcal{L}} \stackrel{def}{=} \forall t : \mathbb{N} \,.\, q(t+1) = d(t) \wedge qbar(t+1) = \neg d(t)$$

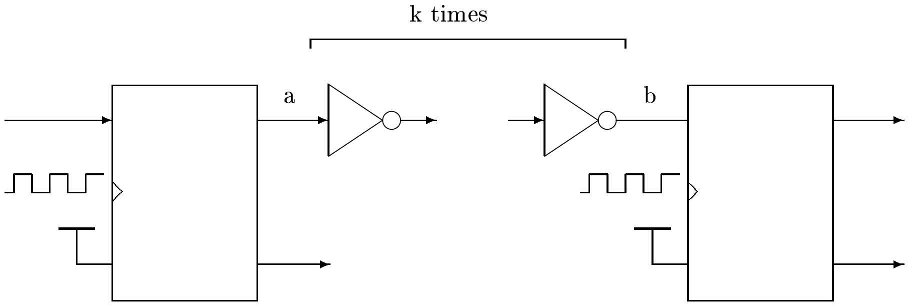
$$\llbracket \mathrm{DType}' \rrbracket^{\mathcal{B}} \stackrel{def}{=} \sum_{y} D_y \quad \text{where} \quad D_y \stackrel{def}{=} \sum_{y' \in \mathrm{bool}} d_{y'} \overline{q_y} \overline{qbar_{\neg y}} : D_{y'}$$

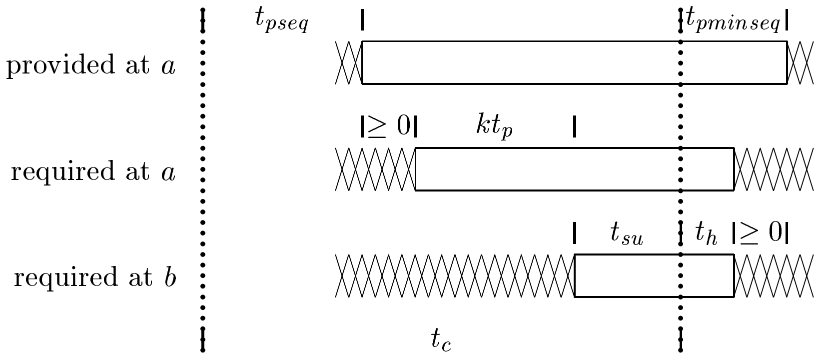
(i.e. $\mathrm{DType}' = \mathrm{Seq}^{fg}_{\{d\},\{q,qbar\}}$ for $f(y, y') \stackrel{def}{=} y'$ and $g(y') \stackrel{def}{=} y'$). We draw a DType′ as below.



8.3 Informal Timing Reasoning

Intuitively, circuits built from these primitives will work correctly if the total propagation delay between any sequential component output and input is small enough. There are no edge-triggered primitive components so no edges or hazards can be significant. The timing for an example circuit for a single clock period is shown below.





From this we need $k \leq (t_c - t_{pseq} - t_{su})/t_p$ and expect that any circuit with no path of more than this many combinational primitives will operate correctly. We take some $K : \mathbb{N}$ satisfying $1 \leq K \leq (t_c - t_{pseq} - t_{su})/t_p$ (for example $1 \leq K \leq 3$) and another condition given later.

9 Design Rule 2: 'There are no long sequences of combinational gates.'

To express Design Rule 2: 'There are no long sequences of combinational gates.' (where a sequence is long if it contains more than K gates) precisely we use a finer notion of type that records the maximum lengths of sequences of combinational primitives. It is again defined inductively on circuit structure, making the definition a little complex but easy to reason about. For checking the design rule a direct characterization on a graph representation of circuits might be more appropriate.

Definition $\mathbb{N}_*, +, \sqcup$ is the naturals extended by a negative infinite element $*$ with addition and maximum operations defined by

$$\begin{array}{c|cc} + & * & n \\ \hline * & * & * \\ m & * & m+n \end{array} \qquad \begin{array}{c|cc} \sqcup & * & n \\ \hline * & * & n \\ m & m & \max(m,n) \end{array} .$$

The operations $+$ and $\sqcup$ are both associative and commutative. They have units 0 and $*$ respectively and $+$ distributes over $\sqcup$. We extend $\sqcup$ to a partial function from subsets of $\mathbb{N}_*$ to $\mathbb{N}_*$ in the obvious way. It is convenient to use matrices over $\mathbb{N}_*$, defining $+, \sqcup$ pointwise and a matrix product $\circ$ by $(A \circ B)_{ac} = \sqcup_b A_{ab} + B_{bc}$.

Definition The *refined types* associated with a sort $\langle i, o \rangle$ are functions

$$R : (i \times o \cup i \cup o) \rightarrow \{*, 0 \dots K\}$$

with, for $a \in i$ and $b \in o$,

- R_{ab} encoding the maximum path lengths of combinational primitives between input a and output b
- R_a encoding the maximum length from input a to anywhere
- R_b encoding the maximum length from anywhere to output b.

These must satisfy $R_{ab} \neq 0$ and $R_a \neq * \neq R_b$.

Definition The refined type of a combinational primitive is R^{comb}, where

$$R^{\text{comb}}_{ab} = 1, \qquad R^{\text{comb}}_{a} = 1 \quad \text{and} \quad R^{\text{comb}}_{b} = 1.$$

The refined type of a sequential primitive is R^{seq}, where

$$R^{\text{seq}}_{ab} = *, \qquad R^{\text{seq}}_{a} = 0 \quad \text{and} \quad R^{\text{seq}}_{b} = 0.$$

For compositions $c \parallel c'$ we first say what it means for two refined types to be composable — intuitively that the composition of any circuits with those refined types will contain no path of combinational primitives of length $> K$. Consider sorts $\langle i_1 h_1, o_1 h_2 \rangle$, $\langle i_2 h_2, o_2 h_1 \rangle$ and refined types R, R' for them. Let $j = i_1 i_2 o_1 o_2 h_1 h_2$ and define $G, O : j \times j \rightarrow \mathbb{N}_*$; $H, J : j \rightarrow \mathbb{N}_*$ by

$$G_{ab} = \left| \begin{array}{ll} R_{ab} & \text{, if } a \in i_1 h_1 \wedge b \in o_1 h_2 \\ R'_{ab} & \text{, if } a \in i_2 h_2 \wedge b \in o_2 h_1 \\ * & \text{, otherwise} \end{array} \right. \qquad H_a = \left| \begin{array}{ll} R_a & \text{, if } a \in i_1 h_1 \\ R'_a & \text{, if } a \in i_2 h_2 \\ * & \text{, otherwise} \end{array} \right. \qquad J_b = \left| \begin{array}{ll} R_b & \text{, if } b \in o_1 h_2 \\ R'_b & \text{, if } b \in o_2 h_1 \\ * & \text{, otherwise} \end{array} \right.$$

$$O_{ab} = \left| \begin{array}{ll} 0 & \text{, if } a = b \\ * & \text{, otherwise} \end{array} \right.$$

Let $C \stackrel{def}{=} \sqcup_{n \geq 1} \overbrace{G \circ \dots \circ G}^{n \text{ times}}$, $A \stackrel{def}{=} (C \sqcup O) \circ H$ and $B \stackrel{def}{=} J \circ (C \sqcup O)$. We say that R, R' are composable if C exists, C, A, B contain no element $> K$ and $\forall c \in h_1 h_2 \,.\, A_c + B_c \leq K$. The refined type $R \parallel R'$ is then given by

$$(R \parallel R')_{ab} \stackrel{def}{=} C_{ab}, \qquad (R \parallel R')_a \stackrel{def}{=} A_a \quad \text{and} \quad (R \parallel R')_b \stackrel{def}{=} B_b.$$

The rules for typing circuits are below.

$$\frac{}{p : R^{\text{comb}}} p \text{ combinational} \qquad \frac{}{p : R^{\text{seq}}} p \text{ sequential} \qquad \frac{c : R \quad c' : R'}{c \parallel c' : R \parallel R'} R, R' \text{ composable}$$

Design Rule 2: 'There are no long sequences of combinational gates', can now be expressed:

Definition $D_2(c) \iff D_0(c)$ and c is typable in the system above.

For simplicity this is slightly stronger than necessary — it forbids all long paths of combinational primitives, not just those which feed to sequential primitives. These refined types above are related to the types of §5 as follows.

Definition If R is a refined type for sort $\langle i, o\rangle$ then $T(R) \subseteq i \times o$ is given by

$$a\; T(R)\; b \iff R_{ab} \neq *.$$

Proposition 5 *If refined types R, R' are composable then types $T(R), T(R')$ are composable and $T(R \,\|\, R') = T(R) \,\|\, T(R')$.*

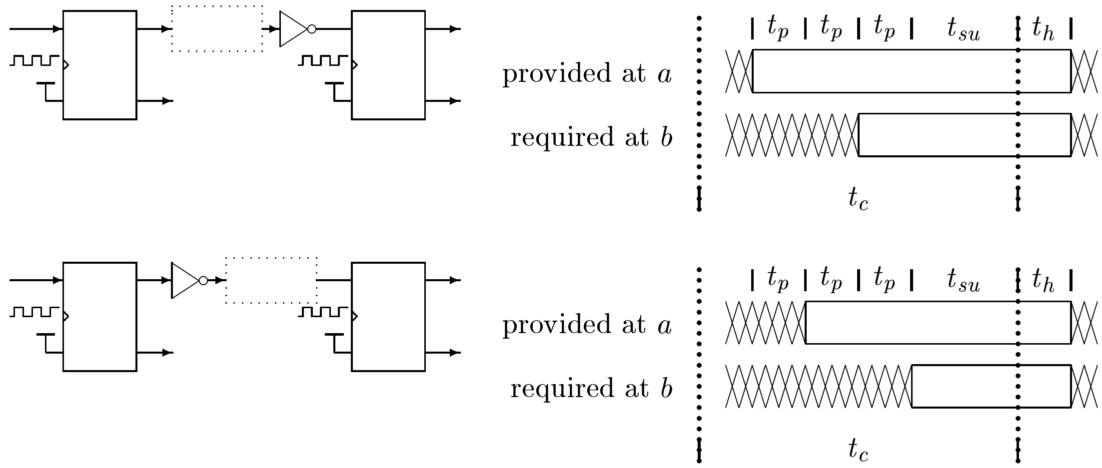
10 Timing Disciplines

In this section we discuss the timing disciplines that a circuit may be required to obey. This simply generalises the informal calculation of §8.3 to arbitrary circuits. We have not yet given a real time model and so cannot yet state what it means for a real time model of a circuit to obey a timing discipline.

Definition A *timing discipline* Δ is a function from the port names N to non-empty subsets of the interval $[0, t_c)$ (for our simple circuits we will only use open sub-intervals of $[0, t_c)$).

Intuitively such a Δ defines, for each port a, a part $\Delta(a)$ of each clock period for which the value on that port will be or must be constant. If we were using a richer model of values this would have to be replaced by a more interesting condition, e.g. that the value on a is above or below certain thresholds in each interval, rather than simply constant. For later convenience we take clock ticks to be at $\{(k+1)t_c - t_h \mid k : \mathbb{N}\}$. A timing discipline may thus refer to intervals containing clock ticks.

A circuit may be placed in contexts that require it to obey different, indeed incomparable, timing disciplines. For example, suppose $K = 3$ and consider the circuit $c = \mathrm{Not}_{ad} \,\|\, \mathrm{Not}_{db}$ in the two contexts on the left below. In these, c must obey the timing disciplines depicted on the right.



Definition For $0 \leq k \leq K$ let δ_k be the interval $(t_c - t_h - t_{su} - (K-k)t_p, t_c)$. If R is a refined type for sort $s = \langle i, o \rangle$ then $td(R)$ is a set of timing disciplines given by $\Delta \in td(R)$ if $\forall a \in i \,.\, \forall b \in o \,.$

$$\begin{array}{ll} & \Delta_a \in \{\delta_k \mid 0 \leq k \leq K - R_a\} \\ \wedge & \Delta_b \in \{\delta_k \mid R_b \leq k \leq K\} \\ \wedge & R_{ab} \neq * \wedge \Delta_a = \delta_k \wedge \Delta_b = \delta_{k'} \Rightarrow k + R_{ab} \leq k' \end{array}$$

The example c above has sort $\langle \{a\}, \{b\} \rangle$ and refined type R, where

$$R_{ab} = 2, \qquad R_a = 2 \quad \text{and} \quad R_b = 2.$$

We have $td(R) = \{\Delta^1, \Delta^2, \Delta^3\}$ where

$$\begin{array}{lll} \Delta^1_a = \delta_0 & \Delta^2_a = \delta_1 & \Delta^3_a = \delta_0 \\ \Delta^1_b = \delta_2 & \Delta^2_b = \delta_3 & \Delta^3_b = \delta_3. \end{array}$$

For any typable compositions we can find a timing discipline that can be agreed upon by both components.

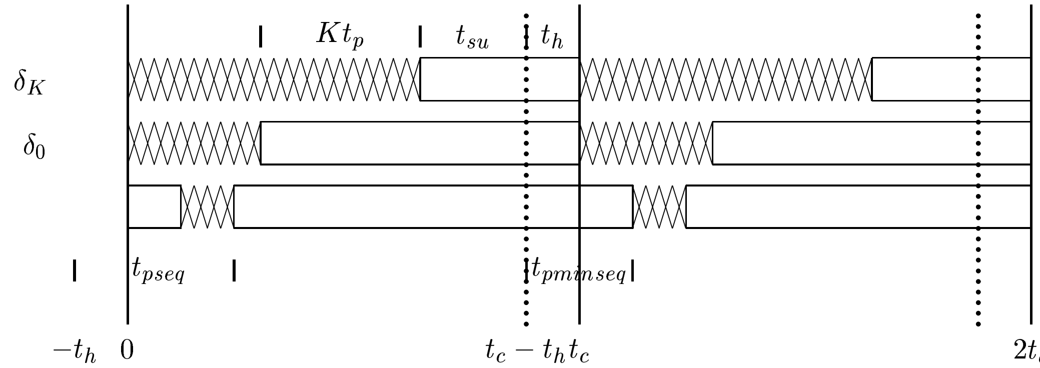
Proposition 6 *If R, R' are composable refined types and $\Delta \in td(R \parallel R')$ then there exists $\Delta' \in td(R) \cap td(R')$ such that $\forall d \in i_1 i_2 o_1 o_2 \,.\, \Delta'_d = \Delta_d$.*

Proposition 7 *For any refined type R the set $td(R)$ is a non-empty set of timing disciplines.*

11 $\mathcal{R}$, a Branching Real Time Model

In order to express timing properties (2) and (3) of primitive components we must use a real time model. We take $\mathcal{R}$ to be $BR([0, t_c) \to V)$, recalling the definition of $BR(\_)$ from §6. Thus $\mathcal{R}_{io}$ is again a transition system quotiented by strong bisimulation but now with labels from $io \to [0, t_c) \to V$ instead of $io \to V$, giving the values on ports *at each time* in a clock period. It is a hybrid of linear real and discrete branching time. This makes it easy to formalise a clean abstraction result and the required properties of primitive components. It is, however, rather ad-hoc. In principle we would prefer a continuously branching model. We will not make essential use of the reals (although we should note that we do not use induction over time). Using 'small-step' discrete time, with many units per clock cycle, would make little difference to the results given here although it would be more awkward to relate to yet more concrete models.

By choosing clock ticks at $\{(k+1)t_c - t_h \mid k:\mathbb{N}\}$ we can treat successive clock periods almost independently. Below we depict the intervals of the first two transitions, together with two of the time intervals of the form δ_k. Dotted vertical lines indicate clock ticks, which occur t_h before the end of each transition.



Transition labels, i.e. functions $j \to [0, t_c) \to V$ for sets of ports $j \subseteq N$, will be ranged over by U, V, W, A, B. The resolution of the overloaded V will hopefully be clear from context.

We say what it means for transitions $U, V : io \to [0, t_c) \to V$ to obey a timing discipline Δ on certain ports $j \subseteq io$, be equal on j and be abstractly equal on j:

Definition U obeys Δ on j, written $U{\downarrow}_j^\Delta$, if there exists some $abs_j^\Delta(U) : j \to V$ such that $\forall c \in j \,.\, \forall t \in \Delta(c) \,.\, U(c)(t) = abs_j^\Delta(U)(c)$.

$$\begin{array}{rcl} U =_j V & \stackrel{def}{=} & \forall c \in j \,.\, U(c) = V(c) \\ U \simeq_j^\Delta V & \stackrel{def}{=} & U{\downarrow}_j^\Delta \wedge V{\downarrow}_j^\Delta \wedge abs_j^\Delta(U) = abs_j^\Delta(V) \end{array}$$

Subscripts and superscripts will be omitted when clear from context.

12 Abstraction from $\mathcal{R}$ to $\mathcal{B}$

An abstraction relation between $\mathcal{R}$ and $\mathcal{B}$ must satisfy two criteria — it must be reasonably strong, to enable us to make interesting predictions about the behaviour of a circuit from $[\![c]\!]^{\mathcal{B}}$, and the abstraction result must be derivable from physically plausible assumptions on primitive components. In the first subsection we define a relation $<_\Delta$ between $\mathcal{R}$ and $\mathcal{B}$, parameterised by a timing discipline Δ. The relation $<$ between $\mathcal{R}$ and $\mathcal{B}$, for which we shall give an abstraction result, is defined in terms of $<_\Delta$ and a notion of a 'resonable' timing discipline. An alternative characterization of $<_\Delta$ is given using a partial equivalence relation $\cong_\Delta$ over $\mathcal{R}$ and a function $abs^\Delta(\_)$ from $\mathcal{R}$ to $\mathcal{B}$. In the second subsection we briefly consider a linear real time model $\mathcal{LR}$, giving some linear time consequences of $<_\Delta$. In the last two subsections we give an interpretation of the refined types in $\mathcal{R}$, formalising some plausible timing properties of primitives, and prove an abstraction result along $<$ that depends on them.

12.1 Abstraction Relations

The relation $<_\Delta$ is defined as follows.

Definition If Q is a sort-respecting relation between $\mathcal{R}$ and $\mathcal{B}$, $s \in \mathcal{R}_{io}$ and $t \in \mathcal{B}_{io}$ then $s\ \mathcal{E}_\Delta(Q)\ t$ iff for all $AB : io \to [0, t_c) \to V$

1. $s \xrightarrow{AB} \wedge A\!\downarrow\, \Rightarrow B\!\downarrow$
2. $AB\!\downarrow \wedge\, s \xrightarrow{AB} s' \Rightarrow \exists t' \,.\, t \xrightarrow{abs(AB)} t' \wedge s'\ Q\ t'$
3. $AB\!\downarrow \wedge\, t \xrightarrow{abs(AB)} t' \Rightarrow \exists B', s' \,.\, s \xrightarrow{AB'} s' \wedge B' \simeq B \wedge s'\ Q\ t'$

Definition $<_\Delta$ is the greatest fixed point of $\mathcal{E}_\Delta$. It contains all pre-fixed points of $\mathcal{E}_\Delta$.

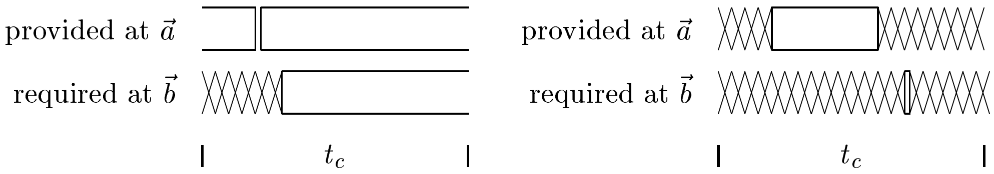
Proposition 8 *$<_\Delta$ is well defined and contains all pre-fixed points of $\mathcal{E}_\Delta$.*

Loosely, in any state accessible by supplying inputs obeying Δ, any real transition can be matched by the unique corresponding discrete one and any discrete transition can, for any corresponding real inputs, be matched by a real one. More precisely, $s <_\Delta t$ if for the first clock period:

1. If s has a behaviour with input signals $A : i \to [0, t_c) \to V$ and output signals $B : o \to [0, t_c) \to V$ and, for each input $a \in i$, $A(a)$ is stable during $\Delta(a)$, then for each output $b \in o$ $B(b)$ is stable during $\Delta(b)$.
2. If s has a behaviour with signals $AB : io \to [0, t_c) \to V$ leaving it in state s' and, for each port $c \in io$, $(AB)(c)$ is stable during $\Delta(c)$, then t has a behaviour with values $abs(AB) : io \to V$ leaving it in a state t'. Moreover, $s' <_\Delta t'$.
3. If t has a behaviour with values $\alpha\beta : io \to V$ leaving it in a state t' then, for any input signals $A : i \to [0, t_c) \to V$ that are stable during Δ and satisfy $abs(A) = \alpha$ there are output signals $B' : o \to [0, t_c) \to V$ that are stable during Δ and satisfy $abs(B) = \beta$ such that s has a behaviour with signals AB' leaving it in a state s'. Moreover $s' <_\Delta t'$.

In the proof of the abstraction result we will show that if c is a circuit with refined type R and Δ is a timing discipline associated with a context in which such a circuit may be placed (i.e. $\Delta \in td(R)$) then $[\![c]\!]^\mathcal{R} <_\Delta [\![c]\!]^\mathcal{B}$. This statement is a little heavy, requiring the definitions of refined types and $td(\_)$. When working with a discrete time model, say $[\![c]\!]^\mathcal{B}$, of a circuit c it will often suffice to show that there is *some* timing discipline for which the real time model $[\![c]\!]^\mathcal{R}$ has well defined abstract behaviour that is equal to $[\![c]\!]^\mathcal{B}$. We therefore pick out a large class of timing disciplines and define the abstraction relation $<$ as the union over these of $<_\Delta$.

Consider the (pathological) timing disciplines depicted below for sort $\langle \vec{a}, \vec{b} \rangle$. The left is unresonable in that the input signal is not realisable — there is insufficient switching time allowed. The right is unreasonable in that the output signal is not observable — it is not required to stay constant for long enough to observe.



Accordingly:

Definition A timing discipline Δ is *reasonable* for sort $\langle i, o\rangle$ if it allows a significant switching time t_{sw} between clock periods for inputs and ensures that outputs are constant for at least a noticeable time t_{small}, i.e. taking for simplicity each Δ_a to be an open subset of $[0, t_c)$:

$$\forall a \in i \;.\; \Delta_a = (t, t') \text{ for some } t, t' \text{ such that } t' - t \leq t_c - t_{sw}$$
$$\forall b \in o \;.\; \Delta_b = (t, t') \text{ for some } t, t' \text{ such that } t' - t \geq t_{small}$$

For the following proposition we require $t_c - (t_h + t_{su} + Kt_p) \geq t_{sw}$ and $t_h + t_{su} \geq t_{small}$, e.g. $t_{sw} \leq 30ns, t_{small} \leq 25ns$.

Proposition 9 *For any refined type R the set $td(R)$ contains only reasonable timing disciplines.*

Finally we can state the abstraction relation.

Definition For $s \in \mathcal{R}$ and $t \in \mathcal{B}$ of the same sort

$$s < t \quad \textit{iff} \quad \exists \text{ reasonable } \Delta \;.\; s <_\Delta t.$$

The abstraction result will thus be of the form $\forall c \in \mathit{circ} \;.\; D_2(c) \Rightarrow [\![c]\!]^{\mathcal{R}} < [\![c]\!]^{\mathcal{B}}$ under some restrictions on $[\![p]\!]^{\mathcal{R}}$ for primitives p.

We conclude this subsection by giving a different characterization of $<_\Delta$, showing that $s <_\Delta t$ if the behaviour of s is closed under small variations of input signals (up to small variations of output signals and resulting states) and moreover a discretization of the behaviour of s is bisimilar to t.

Definition If Q is a sort-respecting relation on $\mathcal{R}$ and $s, s' \in \mathcal{R}_{io}$ then $s \;\mathcal{F}_\Delta(Q)\; s'$ iff for all $AB : io \to [0, t_c) \to V$

1. $s \xrightarrow{AB} \hat{s} \wedge A{\downarrow} \Rightarrow B{\downarrow} \wedge \forall A' \simeq A \;.\; \exists B' \simeq B, \hat{s}' \;.\; s' \xrightarrow{A'B'} \hat{s}' \wedge \hat{s} \;Q\; \hat{s}'$
2. $s' \xrightarrow{AB} \hat{s}' \wedge A{\downarrow} \Rightarrow B{\downarrow} \wedge \forall A' \simeq A \;.\; \exists B' \simeq B, \hat{s} \;.\; s \xrightarrow{A'B'} \hat{s} \wedge \hat{s} \;Q\; \hat{s}'$

Definition $\cong_\Delta$ is the greatest fixed point of $\mathcal{F}_\Delta$. It is a partial equivalence relation containing all pre-fixed points of $\mathcal{F}_\Delta$.

Definition We define $abs^\Delta(\_) : \mathcal{R} \to \mathcal{B}$ by

$$abs^\Delta(s) \xrightarrow{\mu} abs^\Delta(s') \quad \textit{iff} \quad \exists U \;.\; s \xrightarrow{U} s' \wedge U{\downarrow} \wedge abs(U) = \mu$$

Proposition 10 *$\cong_\Delta$ is well defined and is a partial equivalence relation containing all pre-fixed points of $\mathcal{F}_\Delta$.*

Proposition 11 $s \cong_\Delta s' \Rightarrow abs^\Delta(s) = abs^\Delta(s')$

Proposition 12 $s <_\Delta t \iff s \cong_\Delta s \wedge abs^\Delta(s) = t$

12.2 $\mathcal{LR}$, a Linear Real Time Model

When considering the real time behaviour of circuits one might naturally consider a linear real time model $\mathcal{LR}$, modelling signals by functions from the non-negative reals to V and circuits by predicates over such functions, i.e. with $[\![c]\!]^{\mathcal{LR}} \subseteq io \to \mathbb{R}_{\geq 0} \to V$ for c of sort $\langle i, o\rangle$. The set $itr(s)$ of infinite traces of a state $s \in \mathcal{R}_{io}$ can be considered as such a predicate, eliding the isomorphism between $\mathbb{N} \to io \to [0, t_c) \to V$ and $io \to \mathbb{R}_{\geq 0} \to V$ that 'glues together' the signals in each clock period. We can thus give linear time consequences of the relation $<_\Delta$. This will lend support to our claim that $<$ is a suitable abstraction relation.

It is straightforward to give plausible naive linear real time semantics to the components $\mathrm{Comb}^{f}_{\vec{a}\vec{b}}$ and $\mathrm{Seq}^{fg}_{\vec{a}\vec{b}}$:

$$[\![\mathrm{Comb}^{f}_{\vec{a}\vec{b}}]\!]^{\mathcal{LR}} \stackrel{def}{=} \forall t, t'' : \mathbb{R}_{\geq 0} \,.\, \forall \vec{v} : V^m \,.\, \begin{array}{l} ((t'' - t \geq t_p) \wedge (\forall t' \in (t, t'') \,.\, \vec{a}(t') = \vec{v})) \\ \Rightarrow \\ (\forall t' \in (t + t_p, t'') \,.\, \vec{b}(t') = f(\vec{v})). \end{array}$$

Definition A signal $a : \mathbb{R}_{\geq 0} \to V$ *obeys* a timing discipline Δ if it is constant within $\Delta(a)$ in all clock periods, i.e. if there is some function $f : \mathbb{N} \to V$ such that $\forall n : \mathbb{N} \,.\, \forall t : \mathbb{R}_{\geq 0} \,.\, t - nt_c \in \Delta(a) \Rightarrow a(t) = f(n)$. When it exists, f will be unique and denoted $abs^\Delta(a)$.

The linear real time semantics of $\mathrm{Seq}^{fg}_{\vec{a}\vec{b}}$ can now be stated in terms of a timing discipline Δ, where

$$\begin{array}{rcl} \forall p \in 1..m \,.\, \Delta_{a_p} & = & (t_c - t_h - t_{su}, t_c) \\ \forall q \in 1..n \,.\, \Delta_{b_q} & = & (t_{pseq} - t_h, t_c) \end{array}$$

$$\begin{array}{rl} [\![\mathrm{Seq}^{fg}_{\vec{a}\vec{b}}]\!]^{\mathcal{LR}} \stackrel{def}{=} & (\vec{a} \textit{ obeys } \Delta \Rightarrow \vec{b} \textit{ obeys } \Delta) \\ & \wedge\, (\vec{a}\vec{b} \textit{ obeys } \Delta \Rightarrow \exists \vec{h} : \mathbb{N} \to V^l \,.\, \forall n : \mathbb{N} \,. \\ & \quad abs^\Delta(\vec{b})(n) = g(\vec{h}(n)) \wedge \vec{h}(n+1) = f(abs^\Delta(\vec{a})(n), \vec{h}(n))). \end{array}$$

These seem, however, to be too weak to obtain a useful abstraction result, for example one based on the following linear time analogue of $<_\Delta$.

Definition Suppose $P \in \mathcal{LR}_{io}$ and $Q \in \mathcal{L}_{io}$, i.e. $P \subseteq io \to \mathbb{R}_{\geq 0} \to V$ and $Q \subseteq io \to \mathbb{N} \to V$. We say $P <^{lin}_\Delta Q$ iff:

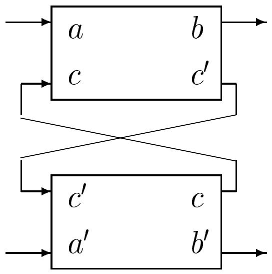
1. $\forall ab : io \to \mathbb{R}_{\geq 0} \to V \,.\, (a \textit{ obeys } \Delta \wedge P(ab)) \Rightarrow b \textit{ obeys } \Delta$.
2. $\forall a : i \to \mathbb{R}_{\geq 0} \to V \,.\, \forall \beta : o \to \mathbb{N} \to V \,.\, a \textit{ obeys } \Delta \Rightarrow$
   $(\exists b \,.\, P(ab) \wedge abs^\Delta(b) = \beta \wedge b \textit{ obeys } \Delta \iff Q(abs^\Delta(a)\beta))$.

Proposition 13 *If $s \in \mathcal{R}_{io}$ and $t \in \mathcal{B}_{io}$ then $s <_\Delta t$ implies $itr(s) <^{lin}_\Delta itr(t)$.*

The branching time models and relation $<_\Delta$ have an additional advantage in that they give a direct understanding of the behaviour of circuits for finite durations, i.e. where they are supplied with input signals that only obey the relevant timing disciplines for a finite number of clock periods. Further, they appear to be technically simpler to work with. The author initially attempted to give a good abstraction result between $\mathcal{LR}$ and $\mathcal{L}$. It turned out to be much more natural to formalise the required real-time properties of components in $\mathcal{R}$ rather than in $\mathcal{LR}$.

12.3 Interpretation of Refined Types in $\mathcal{R}$

For an abstraction result along $<$ we need to find conditions on the real time behaviours of circuits (i.e., an interpretation of the refined types in $\mathcal{R}$) under which $<_\Delta$ is a conguence for (refined-typable) parallel composition. Equivalently, by Proposition 12, under which $\cong_\Delta$ and $abs^\Delta(\_)$ are congruences. There are two problems. Firstly, consider a composition of circuits as below. For its real time model to satisfy the first clause of the definition of $<_\Delta$ we must show that, if Δ is obeyed during a clock period on a and a', then Δ must be obeyed on b and b'. However, from the first clause of the definition applied to the subcircuits we cannot infer that Δ is obeyed on c and c'.



Our proof that it *is* obeyed will be based on the following property of a state $s \in \mathcal{R}_{io}$ in the branching real time model, taking $U : io \to [0, t_c) \to V$ and a type $T \subseteq i \times o$.

> If s has a transition $s \xrightarrow{U} s'$ that obeys the timing discipline on all inputs that are directly connected to an output b (i.e. on $\{\, a \in i \mid a\ T\ b \,\}$) then U obeys the timing discipline on b. (4)

(See Lemma 19, Corollary 20 and Corollary 21 in Appendix A.)This is essentially the combination of property (2) and the first sentence of property (3), generalised to arbitrary circuits and without the part dealt with by the definition of $td(\_)$. Secondly, consider $s <_\Delta t$, $s' <_\Delta t'$ and a discrete transition of $t \parallel t'$, i.e. $t \xrightarrow{\mu} \hat{t}$, $t' \xrightarrow{\nu} \hat{t}'$ and $\mu =_{h_1 h_2} \nu$. By the definition of $<_\Delta$ we have $s \xrightarrow{U} \hat{s}$ and $s' \xrightarrow{V} \hat{s}'$ with $U \simeq_{h_1 h_2} V$. To show $s \parallel s' <_\Delta t \parallel t'$, however, there must be a transition $s \parallel s' \xrightarrow{U \parallel V} \hat{s} \parallel \hat{s}'$ so we need $U =_{h_1 h_2} V$. Our proof of this will be based on the property:

> If s has a transition $s \xrightarrow{U} s'$ that obeys the timing discipline then, for any input a, the signal on a may be changed slightly without doing more than slightly change the signals on the outputs that are directly connected to a (i.e. on $\{\, b \in o \mid a\ T\ b \,\}$) and slightly change the resulting state. (5)

(See Lemma 22, Corollary 23 and Corollary 24 in Appendix A.)This is an amplification of the second sentence of property (3), generalised to arbitrary circuits. In the rest of this subsection we make these precise, giving an interpretation of the refined types in the branching real time model $\mathcal{B}$. We first state the properties for a single clock period and fixed timing discipline Δ. We then define a modified form of bisimulation, strengthening $\cong_\Delta$ by incorporating these properties at every clock period. This *will* be a congruence for parallel composition.

Definition If $\sim$ is a relation on $\mathcal{R}_{io}$ and $T \subseteq i \times o$ then $s \in \mathcal{R}_{io}$ inhabits 1-step type T for $\sim$ if

1. $\forall b \in o \,.\, s \xrightarrow{U} s' \wedge U{\downarrow}_{\{a \in i | a T b\}} \Rightarrow U{\downarrow}_{\{b\}}$
2. $\forall a \in i \,.\, \forall U, U' : io \to [0, t_c) \to V \,.\quad s \xrightarrow{U} s' \wedge U{\downarrow} \wedge U \simeq_{\{a\}} U'$
$$\Rightarrow$$
$$\exists U'', s'' \,.\, s \xrightarrow{U''} s'' \sim s' \wedge \quad U'' =_{\{a\}} U' \wedge \\ U'' =_{i_B o_A} U \wedge \\ U'' \simeq_{o_B} U$$
where $i_B \stackrel{def}{=} i - \{a\}$, $o_A \stackrel{def}{=} \{b \in o \mid \neg a \; T \; b\}$ and $o_B \stackrel{def}{=} o - o_A$.
3. $\forall U : io \to [0, t_c) \to V \,.\, s \xrightarrow{U} s' \wedge U{\downarrow} \Rightarrow s' \sim s'$

Clauses 1 and 2 formalize properties (4) and (5) respectively. Clause 3 ensures that inhabitation of refined types will be preserved by all transitions that obey the timing discipline.

The interpretation of refined types in $\mathcal{R}$ and the relation $\sim$, which will turn out to be a partial equivalence relation, used above must be defined simultaneously. We need a partial equivalence which is finer than the bisimulation $\cong_\Delta$ defined above and also preserves refined types.

Definition If $\sim$ is a relation on $\mathcal{R}_{io}$, $T \subseteq i \times o$ and $s, s' \in \mathcal{R}_{io}$ then $s \; \mathcal{G}_{T,\Delta}(\sim) \; s'$ iff

1. $s \; \mathcal{F}_\Delta(\sim) \; s'$
2. s, s' both inhabit 1-step type T for $\sim$.

Definition $\sim_{T\Delta}$ is the greatest fixed point of $\mathcal{G}_{T,\Delta}$. It is a partial equivalence relation containing all pre-fixed points of $\mathcal{G}_{T,\Delta}$.

Proposition 14 *$\sim_{T\Delta}$ is well defined and is a partial equivalence relation containing all pre-fixed points of $\mathcal{G}_{T,\Delta}$.*

These relations are congruences for parallel composition in the following sense.

Proposition 15 *If $s \sim_{T\Delta} \hat{s}$, $s' \sim_{T'\Delta} \hat{s}'$ and T, T' are composable in the sense of §5 then $(s \parallel s') \sim_{(T \parallel T')\Delta} (\hat{s} \parallel \hat{s}')$. Further, $abs^\Delta(s \parallel s') = abs^\Delta(\hat{s}) \parallel abs^\Delta(\hat{s}')$.*

Proof Sketch: Both parts are proved by coinduction, making use of some composability results for states inhabiting 1-step types. These results are proved by induction along the internal and output ports, ordered using Lemma 1. $\square$

Corollary 16 *If $s \sim_{T\Delta} s <_\Delta t$, $s' \sim_{T'\Delta} s' <_\Delta t'$ and T, T' are composable then $(s \parallel s') \sim_{(T \parallel T')\Delta} (s \parallel s') <_\Delta (t \parallel t')$.*

Proof By Proposition 12 $s \cong_\Delta s$, $s' \cong_\Delta s'$, $abs^\Delta(s) = t$ and $abs^\Delta(s') = t'$. By Proposition 15 $abs^\Delta(s \,\|\, s') = t \,\|\, t'$ and $s \,\|\, s' \sim_{(T \,\|\, T')\Delta} s \,\|\, s'$. It then suffices to note that $\forall T, \Delta \,.\! \sim_{T\Delta} \subseteq \cong_\Delta$. □

Finally we can interpret the refined types in $\mathcal{R}$.

Definition If $s \in \mathcal{R}_{io}$ and R is a refined type for $\langle i, o \rangle$ then s inhabits R iff $\forall \Delta \in td(R)$. $s \sim_{T(R),\Delta} s$.

12.4 Abstraction Result

The following proposition is used only implicitly, within the proof of the abstraction result. We state it anyway.

Proposition 17 *If s, s' inhabit composable refined types R, R' then $s \,\|\, s'$ inhabits $R \,\|\, R'$.*

Proof Consider $\Delta \in td(R \,\|\, R')$. By Proposition 6 w.l.g. $\Delta \in td(R) \cap td(R')$. By Proposition 5 $T(R)$, $T(R')$ are composable and $T(R \,\|\, R') = T(R) \,\|\, T(R')$. As s, s' inhabit R, R' we have $s \sim_{T(R)\,\Delta} s$ and $s' \sim_{T(R')\,\Delta} s'$. By Corollary 16 $s \,\|\, s' \sim_{T(R) \,\|\, T(R')\,\Delta} s \,\|\, s'$ hence $s \,\|\, s'$ inhabits $R \,\|\, R'$. □

And state the abstraction theorem.

Theorem 3 *If for all primitive components $p\!:\!R$ we have $\forall \Delta \in td(R)$. $[\![p]\!]^{\mathcal{R}} <_\Delta [\![p]\!]^{\mathcal{B}}$ and $[\![p]\!]^{\mathcal{R}}$ inhabits R then $\mathcal{B}, D_2$ is an abstraction of $\mathcal{R}, D_2$ along $<$.*

Proof We first show by induction on circuits that if $c\!:\!R$ then $\forall \Delta \in td(R)$. $[\![c]\!]^{\mathcal{R}} <_\Delta [\![c]\!]^{\mathcal{B}}$ and $[\![c]\!]^{\mathcal{R}}$ inhabits R. The base case is trivial. For $c \,\|\, c'$ suppose that $c\!:\!R$ and $c'\!:\!R'$. By assumption R, R' are composable so by Proposition 5 $T(R)$ and $T(R')$ are composable and $T(R \,\|\, R') = T(R) \,\|\, T(R')$. Consider $\Delta \in td(R \,\|\, R')$. By Proposition 6 w.l.g. $\Delta \in td(R) \cap td(R')$. By induction $[\![c]\!]^{\mathcal{R}} \sim_{T(R)\,\Delta} [\![c]\!]^{\mathcal{R}} <_\Delta [\![c]\!]^{\mathcal{B}}$ and $[\![c']\!]^{\mathcal{R}} \sim_{T(R')\,\Delta} [\![c']\!]^{\mathcal{R}} <_\Delta [\![c']\!]^{\mathcal{B}}$. By Corollary 16

$$[\![c \,\|\, c']\!]^{\mathcal{R}} \sim_{T(R \,\|\, R')\,\Delta} [\![c \,\|\, c']\!]^{\mathcal{R}} <_\Delta [\![c \,\|\, c']\!]^{\mathcal{B}}.$$

It then suffices to note that, by Propositions 7 and 9, $<_\Delta \subseteq <$. □

This can be composed with the earlier result to give an abstraction from the real branching time model to the discrete linear time model.

Corollary 18 *Given $[\![\_]\!]^{\mathcal{L}}$ and $[\![\_]\!]^{\mathcal{R}}$, if there exists $[\![\_]\!]^{\mathcal{B}}$ such that for all primitive components $p\!:\!R$*

- $[\![p]\!]^{\mathcal{B}} \prec^{nt} [\![p]\!]^{\mathcal{L}}$
- $[\![p]\!]^{\mathcal{B}}$ *inhabits* $T(R)$
- $\forall \Delta \in td(R)$. $[\![p]\!]^{\mathcal{R}} <_\Delta [\![p]\!]^{\mathcal{B}}$
- $[\![p]\!]^{\mathcal{R}}$ *inhabits* R

then $\mathcal{L}, D_2$ is an abstraction of $\mathcal{R}, D_2$ along $< \circ \prec^{nt}$.

Proof Immediate from Theorem 2, Theorem 3 and Proposition 5 (which entails that $D_2(c)$ implies $D_1(c)$). □

13 Conclusion

We have given a simple framework for understanding the significance of abstract hardware models and design rules in terms of abstraction results to more concrete models, in which important properties of component behaviour can be expressed. This was instantiated by two abstraction results for models of some synchronous circuits, making use of two natural design rules.

From the concurrency-theoretic point of view the abstraction results can be seen as interesting uses of bisimulation-based semantics and as uses of two notions of type (capturing some properties of dynamic behaviour) supported by delicate rely-guarantee condition reasoning. Expressing properties such as our inhabitation of refined types in $\mathcal{R}$ would perhaps be a useful test example for real-time process calculi and logics over them.

There are some obvious defects in the work. In particular:

- The required real time properties of primitive components (i.e. that $[\![p]\!]^{\mathcal{B}}$ inhabits R^{comb} or R^{seq} for combinational or sequential p) have not been shown to hold of accepted component models. In fact, requirements such as

  > The values on the outputs of a sequential primitive in a given clock period are independent of the values on the inputs in that clock period.

  (part of property (3)) might not be expected to hold *exactly* in a real time/voltage model. More subtle formalised properties than our interpretations of refined types would then be needed for an abstraction result. It is not clear whether the proof of this result could follow the same form as the one given, particularly for the analogue of (the key) Proposition 15.

- The model $\mathcal{R}$ is rather ad-hoc. The hybrid of linear real time (in each clock cycle) and branching discrete time (with branching only at the ends of clock cycles) was chosen to simplify the statement and proof of the abstraction result. We would prefer a model that allows branching at all times.

- The circuits considered have been extremely simple and the primitive components rather complex (and not heavily used in current VLSI design).

We conclude by mentioning some related work, pointing out some differences without attempting a full survey or discussion.

In [Win87] Winskel considers abstraction between two models of the steady state behaviour of transistors. An abstraction result is given in terms of an adjunction between partial orders of specifications. Related work is presented in [Mel93, Chapter 7], where Melham shows that for circuits satisfying a design rule Wb (defined inductively on circuits but also using the more concrete semantic function) two steady state transistor models agree. When we get to more concrete timed models, such as the $\mathcal{R}$ defined here, it is not clear what a good class of specifications of circuits should be. It seems more straightforward, therefore, to work only with the models of circuits, whose significance is given by abstraction results along particular relations, as we have done.

In [Her88, Her89] Herbert takes a linear small-step discrete time model and considers the implementation of certain flip-flops using gates. The timing properties of components differ from those expressed by inhabitation of refined types — the latter are composable, even for cyclic networks of sequential primitives, but would not suffice to show correctness of a flip-flop implementation using combinational primitives. Related work by Hanna and Daeche is presented in [HD86], where the authors consider a specification of a D-type flipflop and show that it can be implemented using NAND gates. The D-type specification seems to be similar to the linear real time models $[\![\mathrm{Seq}^{fg}_{\vec{a}\vec{b}}]\!]^{\mathcal{LR}}$ of §12.2 (albeit with a more sophisticated treatment of the clock input, i.e. not assuming a global clock).

In [Fou95] Fourman considers a discrete time model of sequential circuits. Starting with a lattice B of four values that may appear on a wire (1, 0, under-driven and over-driven) circuits are modelled by functions of type $(\mathbb{N} \rightarrow B) \rightarrow (\mathbb{N} \rightarrow B)$ satisfying certain conditions. This allows a property of 'non-zero delay' to be stated. It would be interesting to have a precise connection between this and the inhabitation of types in $\mathcal{B}$.

In [Han94] Hanna considers the steady state analogue behaviour of devices, giving a verification of a transistor implementation of a Not gate. He also mentions the possibility of proving design rules *correct*. This notion of correctness is given in terms of the commutation of certain behavioural abstraction and structure reification functions. The proof of Theorem 3 essentially shows such a result, where the behavioural abstraction function is the $abs^{\Delta}(\_) : \mathcal{R} \rightarrow \mathcal{B}$ defined in §12.1 and the structure reification function is the identity (we have considered analogue and digital behaviour of a single representation of circuit structure).

Acknowledgements I would like to thank Michael Fourman and Anthony McIsaac for their comments on drafts of this work, Stuart Anderson for suggesting the case study that was its original stimulus, and an anonymous referee. I acknowledge support from SERC studentship 90311819, ESPRIT BRA 6454 'CONFER' and the EPSRC grant 'Action Structures and the Pi Calculus'.

A Deferred Proofs

This appendix contains the proofs of propositions required for the abstraction from $\mathcal{R}$ to $\mathcal{B}$. It is divided into two parts. The first contains composability results for 1-step types and so also for the relations $\sim_{T\Delta}$, giving a proof of Proposition 15. This is the larger part of the formalisation in $\mathcal{R}$ of the informal timing reasoning. The second contains the remainder — the proofs that the various coinductive relations used are well defined, results about the timing disciplines associated with refined types, the characterisation of $<_{\Delta}$ using $\cong_{\Delta}$ and $abs^{\Delta}(\_)$ and the relationship between $<_{\Delta}$ and $<^{lin}_{\Delta}$.

A.1 Proofs of 1-step type composability results

In order to prove Proposition 15 (the composition result for the interpretation of refined types in the real time model) it is convenient to use slightly more abstract definitions. In this subsection we take an arbitrary set $\mathcal{V}$ of values and a branching time model $\mathcal{S} \stackrel{def}{=} BR(\mathcal{V})$. We suppose that for each port name $a \in N$ there is a partial equivalence relation $\simeq_a$ over $\mathcal{V}$. This is lifted to functions from sets of ports to values as follows. If

$A, B, C \subseteq N$ and $U : A \to \mathcal{V},\ V : B \to \mathcal{V}$ then

$$\begin{array}{rll} U \simeq_C V & \textit{iff} & \forall c \in C \cap A \cap B \ .\ U(c) \simeq_c V(c) \\ U{\downarrow}_C & \textit{iff} & U \simeq_C U \\ U =_C V & \textit{iff} & \forall c \in C \cap A \cap B \ .\ U(c) = V(c). \end{array}$$

In each case if the subscript is omitted (and $A = B$) then it is taken to be A.

Definition If $\sim$ is a partial equivalence relation on $\mathcal{S}_{io}$ and $T \subseteq i \times o$ then $s \in \mathcal{S}_{io}$ inhabits 1-step type T for $\sim$ if

1. $\forall b \in o \ .\ s \xrightarrow{U} s' \wedge U{\downarrow}_{\{a \in i | aTb\}} \Rightarrow U{\downarrow}_{\{b\}}$
2. $$\begin{array}{l} \forall a \in i \ .\ \forall U, U' : io \to \mathcal{V} \ .\quad s \xrightarrow{U} s' \wedge U{\downarrow} \wedge U \simeq_{\{a\}} U' \\ \qquad\qquad \Rightarrow \\ \qquad\qquad \exists U'', s'' \ .\ s \xrightarrow{U''} s'' \sim s' \wedge \quad U'' =_{\{a\}} U' \wedge \\ \qquad\qquad\qquad\qquad\qquad\qquad\quad U'' =_{i_B o_A} U \wedge \\ \qquad\qquad\qquad\qquad\qquad\qquad\quad U'' \simeq U \end{array}$$
   where $i_B \stackrel{def}{=} i - \{a\}, o_A \stackrel{def}{=} \{b \in o \mid \neg a \ T \ b\}, o_B \stackrel{def}{=} o - o_A$.
3. $\forall U : io \to \mathcal{V} \ .\ s \xrightarrow{U} s' \wedge U{\downarrow} \Rightarrow s' \sim s'$

In the following we consider states $s \in \mathcal{S}_{\langle i_1 h_1, o_1 h_2 \rangle}, t \in \mathcal{S}_{\langle i_2 h_2, o_2 h_1 \rangle}$ that inhabit 1-step types T, T' for $\sim, \sim'$ respectively. We suppose that T and T' are composable.

Two lemmas (19 and 22) are shown using the 1-step type assumptions (using clauses 1 and 2 respectively). They are shown by induction along the hidden ports of a composition, ordered as in Lemma 1. For each there are two immediate corollaries. These are used to show a composability result for 1-step types and thereby prove Proposition 15.

Lemma 19 *If* $s \xrightarrow{U} s',\ t \xrightarrow{V} t',\ U =_{h_1 h_2} V,\ i_{11} \subseteq i_1,\ i_{21} \subseteq i_2,\ U{\downarrow}_{i_{11}}$ *and* $V{\downarrow}_{i_{21}}$ *then* $U{\downarrow}_D$ *and* $V{\downarrow}_D$*, where* $D \stackrel{def}{=} \{d \in h_1 h_2 o_1 o_2 \mid \forall a \in i_1 i_2 \ .\ a \ (T \cup T')^+ \ d \Rightarrow a \in i_{11} i_{21}\}$.

Proof By Lemma 1 D can be ordered as $d_0, \ldots, d_n$ such that $p < q \Rightarrow \neg d_q \ (T \cup T')^+ \ d_p$. We show by induction on $k \in 0 \ldots n$ that $U{\downarrow}_{i_{11} \cup \{d_0 \ldots d_k\}}$ and $V{\downarrow}_{i_{11} \cup \{d_0 \ldots d_k\}}$. The base case $k = 0$ is immediate. For $k + 1$ suppose that $d_{k+1} \in h_2 o_1$ (the reasoning for $d_{k+1} \in h_1 o_2$ is symmetric). It follows from the definition of D that

$$\{a \in i_1 h_1 \mid a \ T \ d_{k+1}\} \subseteq i_{11} \cup \{d_0 \ldots d_k\}$$

so by induction $U{\downarrow}_{\{a \in i_1 h_1 | a T d_{k+1}\}}$. By clause 1 of the definition of 1-step types $U{\downarrow}_{\{d_{k+1}\}}$ so $U{\downarrow}_{i_{11} \cup \{d_0 \ldots d_{k+1}\}}$. For V, if $d_{k+1} \in h_2$ then $V =_{\{d_{k+1}\}} U$ so $V{\downarrow}_{\{d_{k+1}\}}$. If $d_{k+1} \in o_1$ then vacuously $V{\downarrow}_{\{d_{k+1}\}}$. In either case $V{\downarrow}_{i_{21} \cup \{d_0 \ldots d_{k+1}\}}$. □

Corollary 20 *If* $s \xrightarrow{U} s',\ t \xrightarrow{V} t',\ U =_{h_1 h_2} V,\ U{\downarrow}_{i_1}$ *and* $V{\downarrow}_{i_2}$ *then* $U{\downarrow}$ *and* $V{\downarrow}$.

Proof Take $i_{11} = i_1$ and $i_{21} = i_2$, giving $D = h_1 h_2 o_1 o_2$. □

Corollary 21 *If* $s \xrightarrow{U} s',\ t \xrightarrow{V} t',\ U =_{h_1 h_2} V,\ b \in o_1,\ U{\downarrow}_{\{a \in i_1 | a (T \cup T')^+ b\}}$ *and* $V{\downarrow}_{\{a \in i_2 | a (T \cup T')^+ b\}}$ *then* $U{\downarrow}_{\{b\}}$.

PROOF Take $i_{11} = \{a \in i_1 \mid a\ (T \cup T')^+\ b\}$ and $i_{21} = \{a \in i_2 \mid a\ (T \cup T')^+\ b\}$ and note that this ensures $b \in D$. $\square$

Lemma 22 *If*

- $s \xrightarrow{U} s',\ t \xrightarrow{V} t',\ U\!\downarrow$ *and* $V\!\downarrow$
- $i_{11} \subseteq i_1$ *is empty or a singleton*
- $U' : i_{11} \to \mathcal{V}$ *and* $U' \simeq_{i_{11}} U$
- $h_< \subseteq \{c \in h_1 h_2 \mid \forall a \in i_{11}\ .\ \neg a\ (T \cup T')^+\ c\}$ *is such that if* $c \in h_<$ *and* $d \in h_1 h_2 - h_<$ *then* $\neg d\ (T \cup T')^+\ c$
- $U =_{h_<} V$
- $U \simeq_{h_1 h_2} V$

then with additional notation

- $h_> = \{c_1 \ldots c_n\} \stackrel{def}{=} h_1 h_2 - h_<$. *By Lemma 1 w.l.g.* $p < q \Rightarrow \neg c_q\ (T \cup T')^+\ c_p$.
- $i_{12} \stackrel{def}{=} i_1 - i_{11}$
- $o_{11} \stackrel{def}{=} \{b \in o_1 \mid \forall a \in i_{11} h_>\ .\ \neg a\ (T \cup T')^+\ b\}$ *and* $o_{12} \stackrel{def}{=} o_1 - o_{11}$
- $o_{21} \stackrel{def}{=} \{b \in o_2 \mid \forall a \in i_{11} h_>\ .\ \neg a\ (T \cup T')^+\ b\}$ *and* $o_{22} \stackrel{def}{=} o_2 - o_{21}$

it follows that for all $k \in 0 \ldots n$ *there exist* U_k, V_k, s_k, t_k *such that*

- $s \xrightarrow{U_k} s_k \sim s'$ *and* $t \xrightarrow{V_k} t_k \sim' t'$
- $U_k =_{i_{12} h_< o_{11}} U,\ U_k =_{i_{11}} U'$ *and* $V_k =_{i_2 h_< o_{21}} V$
- $U_k \simeq U$ *and* $V_k \simeq V$
- $U_k =_{h_< \cup \{c_1, \ldots, c_k\}} V_k$.

PROOF By induction on k.

Base case $k = 0$: Case $i_{11} = \{\}$: take $U_0 \stackrel{def}{=} U,\ s_0 \stackrel{def}{=} s',\ V_0 \stackrel{def}{=} V$ and $t_0 \stackrel{def}{=} t'$.

Case $i_{11} = \{a\}$: by the definition of 1-step type (clause 2) for s there exist U_0, s_0 such that

- $s \xrightarrow{U_0} s_0 \sim s'$
- $U_0 =_{i_{11}} U'$
- $U_0 =_{i_{12} h_1 \{b \in o_1 h_2 \mid \neg a T b\}} U$
- $U_0 \simeq U$.

By the assumption on $h_<$ and the definition of o_{11}

$$(i_{12} h_< o_{11}) \cap (i_1 h_1 o_1 h_2) \subseteq i_{12} h_1 \{b \in o_1 h_2 \mid \neg a\ T\ b\}$$

so $U_0 =_{i_{12} h_< o_{11}} U$. Taking $V_0 \stackrel{def}{=} V$ and $t_0 \stackrel{def}{=} t'$ the remaining conclusions are trivial.

Inductive step $k+1$:

Suppose $c_{k+1} \in h_1$. We apply the definition of 1-step type (clause 2) for s with premises

- $c_{k+1} \in i_1 h_1$
- $s \xrightarrow{U_k} s_k$
- $U_k\!\downarrow$
- $U_k \simeq_{\{c_{k+1}\}} V_k$

giving the existence of U_{k+1}, s_{k+1} such that

- $s \xrightarrow{U_{k+1}} s_{k+1} \sim s_k$
- $U_{k+1} =_{\{c_{k+1}\}} V_k$
- $U_{k+1} =_{i_1 \cup (h_1 - c_{k+1}) \cup \{b \in h_2 o_1 \mid \neg c_{k+1} T b\}} U_k$
- $U_{k+1} \simeq U_k$.

Taking $V_{k+1} \stackrel{def}{=} V_k$ and $t_{k+1} \stackrel{def}{=} t_k$ the conclusions are straightforward from the above and the observation that

$$i_{12} h_{<} o_{11} \subseteq i_1 \cup (h_1 - c_{k+1}) \cup \{b \in h_2 o_1 \mid \neg c_{k+1}\ T\ b\}.$$

For $c_{k+1} \in h_2$ the reasoning is similar. $\square$

Corollary 23 *If*

- $s \xrightarrow{U} s',\ t \xrightarrow{V} t',\ U\!\downarrow$ *and* $V\!\downarrow$
- $U \simeq_{h_1 h_2} V$

then there exist U_0, V_0, s_0, t_0 *such that*

- $s \xrightarrow{U_0} s_0 \sim s'$ *and* $t \xrightarrow{V_0} t_0 \sim' t'$
- $U_0 =_{h_1 h_2} V_0$
- $U_0 =_{i_1} U$ *and* $V_0 =_{i_2} V$
- $U_0 \simeq U$ *and* $V_0 \simeq V$.

Proof Take $i_{11} = \{\},\ h_{<} = \{\}$ and $k = n$. $\square$

Corollary 24 *If*

- $s \xrightarrow{U} s',\ t \xrightarrow{V} t',\ U\!\downarrow_{i_1}$ *and* $V\!\downarrow_{i_2}$
- $U =_{h_1 h_2} V$
- $a \in i_1,\ U' : i_1 h_1 o_1 h_2 \to \mathcal{V}$ *and* $U \simeq_{\{a\}} U'$

then with additional notation

- $o_{11} \stackrel{def}{=} \{b \in o_1 \mid \neg a\ (T \cup T')^{+}\ b\}$ *and* $o_{12} \stackrel{def}{=} o_1 - o_{11}$

- $o_{21} \stackrel{def}{=} \{b \in o_2 \mid \neg a\ (T \cup T')^+\ b\}$ *and* $o_{22} \stackrel{def}{=} o_2 - o_{21}$

there exist U'', V'', s'', t'' *such that*

- $s \xrightarrow{U''} s''$, $t \xrightarrow{V''} t''$, $s'' \sim s'$ *and* $t'' \sim' t'$
- $U'' =_{h_1 h_2} V''$
- $U'' =_{\{a\}} U'$
- $U'' =_{(i_1 - a) \cup o_{11}} U$
- $V'' =_{i_2 o_{21}} V$
- $U'' \simeq U$ *and* $V'' \simeq V$

Proof By Corollary 20 $U\!\downarrow$ and $V\!\downarrow$. The result follows from Lemma 22 taking $i_{11} = \{a\}$ and $h_< = \{c \mid \neg a\ (T \cup T')^+\ c\}$. □

Lemma 25 *If in addition* $\sim''$ *is a partial equivalence relation over* $\mathcal{S}_{\langle i_1 i_2, o_1 o_2 \rangle}$ *such that* $s_1 \sim s_2 \wedge t_1 \sim' t_2 \Rightarrow s_1 \| t_1 \sim'' s_2 \| t_2$ *then* $s \| t$ *has 1-step type* $T \| T'$ *for* $\sim''$.

Proof Clause 1 follows from Corollary 21, clause 2 follows from Corollary 24 and clause 3 from Corollary 20. □

Finally we return to the concrete definitions to re-state and prove the proposition.

Proposition 15 *If* $s \sim_{T\Delta} s', t \sim_{T'\Delta} t'$ *and* T, T' *are composable then*

1. $s \| t \sim_{(T \| T')\Delta} s' \| t'$
2. $abs^{\Delta}(s \| t) = abs^{\Delta}(s') \| abs^{\Delta}(t')$.

Proof Part 1: Let $Q \stackrel{def}{=} \{s \| t,\ s' \| t' \mid s \sim_{T\Delta} s',\ t \sim_{T'\Delta} t'\}$. We show below that $Q \subseteq \mathcal{F}_{\Delta}(Q)$. Lemma 25 entails that if $r\ Q\ r'$ then r, r' both have 1-step type $T \| T'$ for Q so then $Q \subseteq \mathcal{G}_{T \| T', \Delta}(Q)$, which suffices.

Consider $s \| t\ Q\ s' \| t'$ and a transition $s \| t \xrightarrow{W} s_1 \| t_1$ with $W\!\downarrow_{i_1 i_2}$. Suppose $W' \simeq_{i_1 i_2} W$.

By the definition of parallel composition there exist U, V such that $s \xrightarrow{U} s_1$, $t \xrightarrow{V} t_1$, $U =_{h_1 h_2} V$ and $U \| V = W$. By Corollary 20 $U\!\downarrow$ and $V\!\downarrow$. By $s \sim_{T\Delta} s'$ and $t \sim_{T'\Delta} t'$ there exist U', V', s'_1, t'_1 such that

- $s' \xrightarrow{U'} s'_1 \sim_{T\Delta} s_1$ and $t' \xrightarrow{V'} t'_1 \sim_{T'\Delta} t_1$
- $U' =_{i_1} W'$ and $V' =_{i_2} W'$
- $U' =_{h_1} U$ and $V' =_{h_2} V$
- $U' \simeq U$ and $V' \simeq V$.

It follows that $U' \simeq_{h_1 h_2} V'$, $U'\!\downarrow$ and $V'\!\downarrow$ so Corollary 23 can be applied to give the existence of U'_1, V'_1, s'_2, t'_2 such that

- $s' \xrightarrow{U'_1} s'_2 \sim_{T\Delta} s'_1$ and $t' \xrightarrow{V'_1} t'_2 \sim_{T'\Delta} t'_1$

- $U_1' =_{h_1h_2} V_1'$
- $U_1' =_{i_1} U'$ and $V_1' =_{i_2} V'$
- $U_1' \simeq_{o_1} U$ and $V_1' \simeq_{o_2} V$

so $s' \,\|\, t' \xrightarrow{U_1' \,\|\, V_1'} s_2' \,\|\, t_2'$ with $U_1' \,\|\, V_1' =_{i_1i_2} W'$ and $U_1' \,\|\, V_1' \simeq_{o_1o_2} W$. By Proposition 14 $s_1 \,\|\, t_1 \; Q \; s_2' \,\|\, t_2'$.

Part 2: Let $Q \stackrel{def}{=} \{abs^{\Delta}(s \,\|\, t),\ abs^{\Delta}(s') \,\|\, abs^{\Delta}(t') \mid s \sim_{T\Delta} s',\ t \sim_{T'\Delta} t'\}$. We check that Q is a strong bisimulation.

Consider $abs^{\Delta}(s \,\|\, t) \; Q \; abs^{\Delta}(s') \,\|\, abs^{\Delta}(t')$ and a transition

$$abs^{\Delta}(s') \,\|\, abs^{\Delta}(t') \xrightarrow{\alpha} abs^{\Delta}(s_1') \,\|\, abs^{\Delta}(t_1').$$

By the definition of $abs^{\Delta}(\_)$ there exist U', V', s_1', t_1' such that

- $s' \xrightarrow{U'} s_1'$, $t' \xrightarrow{V'} t_1'$, $U'{\downarrow}$ and $V'{\downarrow}$
- $U' \simeq_{h_1h_2} V'$
- $abs^{\Delta}(U' \,\|\, V') = \alpha$.

By $s \sim_{T\Delta} s'$ and $t \sim_{T'\Delta} t'$ there exist U, V, s_1, t_1 such that

- $s \xrightarrow{U} s_1 \sim_{T\Delta} s_1'$ and $t \xrightarrow{V} t_1 \sim_{T'\Delta} t_1'$
- $U =_{i_1h_1} U'$ and $V =_{i_2h_2} V'$
- $U \simeq U'$ and $V \simeq V'$

It follows that $U{\downarrow}$, $V{\downarrow}$ and $U \simeq_{h_1h_2} V$ so by Corollary 23 there exist U_1, V_1, s_2, t_2 such that

- $s \xrightarrow{U_1} s_2 \sim_{T\Delta} s_1$ and $t \xrightarrow{V_1} t_2 \sim_{T'\Delta} t_1$
- $U_1 =_{h_1h_2} V_1$
- $U_1 =_{i_1} U$ and $V_1 =_{i_2} V$
- $U_1 \simeq U$ and $V_1 \simeq V$

so $abs^{\Delta}(U_1 \,\|\, V_1) = \alpha$ and $abs^{\Delta}(s \,\|\, t) \xrightarrow{\alpha} abs^{\Delta}(s_2 \,\|\, t_2) \; Q \; abs^{\Delta}(s_1') \,\|\, abs^{\Delta}(t_1')$.

Now consider a transition

$$abs^{\Delta}(s \,\|\, t) \xrightarrow{\alpha} abs^{\Delta}(s_1 \,\|\, t_1).$$

By the definition of $abs^{\Delta}(\_)$ there exist U, V such that

- $s \xrightarrow{U} s_1$, $t \xrightarrow{V} t_1$
- $U =_{h_1h_2} V$
- $(U \,\|\, V){\downarrow}$ and $abs^{\Delta}(U \,\|\, V) = \alpha$.

By Corollary 20 $U{\downarrow}$ and $V{\downarrow}$ so by $s \sim_{T\Delta} s'$ and $t \sim_{T'\Delta} t'$ there exist U', V', s_1', t_1' such that

- $s' \xrightarrow{U'} s_1' \sim_{T\Delta} s_1$ and $t' \xrightarrow{V'} t_1' \sim_{T'\Delta} t_1$
- $U' =_{i_1 h_1} U$ and $V' =_{i_2 h_2} V$
- $U' \simeq U$ and $V' \simeq V$.

It follows that $U'{\downarrow}$, $V'{\downarrow}$ and $U' \simeq_{h_1 h_2} V'$ so by Corollary 23 there exist U_1', V_1', s_2', t_2' such that

- $s' \xrightarrow{U_1'} s_2' \sim_{T\Delta} s_1'$ and $t' \xrightarrow{V_1'} t_2' \sim_{T'\Delta} t_1'$
- $U_1' =_{h_1 h_2} V_1'$
- $U_1' =_{i_1} U'$ and $V_1' =_{i_2} V'$
- $U_1' \simeq U'$ and $V_1' \simeq V'$

so $abs^{\Delta}(U_1' \parallel V_1') = \alpha$ and

$$abs^{\Delta}(s') \parallel abs^{\Delta}(t') \xrightarrow{\alpha} abs^{\Delta}(s_2') \parallel abs^{\Delta}(t_2') \; Q^{-1} \; abs^{\Delta}(s_1 \parallel t_1).$$

□

A.2 Proofs of other propositions

Proposition 8 *$<_{\Delta}$ is well defined and contains all pre-fixed points of $\mathcal{E}_{\Delta}$.*

Proposition 10 *$\cong_{\Delta}$ is well defined and is a partial equivalence relation containing all pre-fixed points of $\mathcal{F}_{\Delta}$.*

Proposition 14 *$\sim_{T\Delta}$ is well defined and is a partial equivalence relation containing all pre-fixed points of $\mathcal{G}_{T,\Delta}$.*

Proof (of Propositions 8, 10 and 14) It is immediate that $\mathcal{E}_{\Delta}$, $\mathcal{F}_{\Delta}$, the functional $\mathcal{I}_{T\Delta}$ defined by

$$\mathcal{I}_{T\Delta}(\sim) \stackrel{def}{=} \{\, s \mid s \text{ inhabits 1-step type } T \text{ for } \sim \}$$

and $\mathcal{G}_{T\Delta}$ are all monotone. It follows that $<_{\Delta}$, $\cong_{\Delta}$ and $\sim_{T\Delta}$ are well-defined and contain all pre-fixed points of $\mathcal{E}_{\Delta}$, $\mathcal{F}_{\Delta}$ and $\mathcal{G}_{T\Delta}$ respectively. Moreover it is straightforward to show

$$\begin{array}{rcl}
\mathcal{F}_{\Delta}(Q^{-1}) & = & \mathcal{F}_{\Delta}(Q)^{-1} \\
Q \subseteq \mathcal{F}_{\Delta}(Q) & \Rightarrow & Q \circ Q \subseteq \mathcal{F}_{\Delta}(Q \circ Q) \\
Q \subseteq \mathcal{G}_{T\Delta}(Q) & \Rightarrow & Q^{-1} \subseteq \mathcal{G}_{T\Delta}(Q \cup Q^{-1}) \\
Q \subseteq \mathcal{G}_{T\Delta}(Q) & \Rightarrow & Q \circ Q \subseteq \mathcal{G}_{T\Delta}(Q \cup Q \circ Q)
\end{array}$$

from which it follows that $\cong_{\Delta}$ and $\sim_{T\Delta}$ are symmetric and transitive. □

Proposition 5 *If R, R' are composable then $T(R), T(R')$ are composable and $T(R \parallel R') = T(R) \parallel T(R')$.*

Proof For any matrix $A : i \times j \to \mathbb{N}_*$ we define a relation $T(A) \subseteq i \times j$ by

$$a\, T(A)\, b \quad \mathit{iff} \quad A_{ab} \neq *.$$

It is straightforward to check for matrices A, B that $T(A \circ B) = T(A)T(B)$ and $T(A \sqcup B) = T(A) \cup T(B)$. Furthermore, if $A_m \mid m \in M$ is a family of matrices and $\sqcup_{m \in M} A_m$ is defined then $T(\sqcup_{m \in M} A_m) = \cup_{m \in M} T(A_m)$. Now suppose that $C \stackrel{def}{=} \sqcup_{n \geq 1} G^n$ is defined. By the above observations $T(R \parallel R') = T(R) \parallel T(R')$. Now suppose that $T(R)$ and $T(R')$ are not composable, i.e. for some a

$$a\ (T(R) \cup T(R'))^+\ a.$$

It follows that for some n $(G^n)_{aa} \neq *$. Now, no element of G is equal to 0 so no element of G^n can be. Hence $(G^n)_{aa} \geq 1$ and the set $\{ (G^n)^m_{aa} \mid m \in \mathbb{N} \}$ is unbounded, contradicting the assumption that $\sqcup_{n \geq 1} G^n$ is defined. $\square$

Proposition 6 *If R, R' are composable refined types and $\Delta \in td(R \parallel R')$ then there exist $\Delta' \in td(R) \cap td(R')$ such that*

$$\forall d \in i_1 i_2 o_1 o_2 \,.\, \Delta'_d = \Delta_d$$

Proof We confuse the interval δ_k and the natural number k, regarding timing disciplines as functions from N to $\mathbb{N}_*$. We define Δ' by

$$\Delta'_c \stackrel{def}{=} \left| \begin{array}{ll} J_c \sqcup \bigsqcup_{a \in i_1 i_2} (\Delta_a + C_{ac}) \sqcup \bigsqcup_{d \in h_1 h_2} (J_d + C_{dc}), & \text{if } c \in h_1 h_2 \\ \Delta_c, & \text{if } c \in i_1 i_2 o_1 o_2. \end{array} \right.$$

This sets the starts of the timing disciplines on internal ports to the earliest possible times. As R, R' are composable we know:

0. $\forall c \in h_1 h_2 \,.\, A_c + B_c \leq K$.

From $\Delta \in td(R \parallel R')$ we know:

1. $\forall a \in i_1 i_2 \,.\, 0 \leq \Delta_a \leq K - A_a$
2. $\forall b \in o_1 o_2 \,.\, B_b \leq \Delta_b \leq K$
3. $\forall a \in i_1 i_2 \,.\, \forall b \in o_1 o_2 \,.\, C_{ab} \neq * \Rightarrow \Delta_a + C_{ab} \leq \Delta_b$

We check $\Delta' \in td(R)$, for which we need:

4. $\forall c \in h_1 \,.\, 0 \leq J_c \sqcup \bigsqcup_{a \in i_1 i_2} (\Delta_a + C_{ac}) \sqcup \bigsqcup_{d \in h_1 h_2} (J_d + C_{dc}) \leq K - R_c$
5. $\forall c \in h_2 \,.\, R_c \leq J_c \sqcup \bigsqcup_{a \in i_1 i_2} (\Delta_a + C_{ac}) \sqcup \bigsqcup_{d \in h_1 h_2} (J_d + C_{dc}) \leq K$
6. $\forall a \in i_1 h_1 \,.\, \forall b \in o_1 h_2 \,.\, R_{ab} \neq * \Rightarrow \Delta'_a + R_{ab} \leq \Delta'_b$

From the definitions of A and B, using the fact that, as R, R' are composable $\forall c \,.\, C_{cc} = *$, one can show

7. $\forall c \in j \,.\, A_c = H_c \sqcup \bigsqcup_{d \neq c} C_{cd} + H_d$

8. $\forall c \in j \,.\, B_c = J_c \sqcup \bigsqcup_{e \neq c} J_e + C_{ec}$

4 and 5 follow from 0,1,7,8 by simple inequational reasoning. For 6 we consider cases of a and b. If $a \in i_1$ it is trivial, noting that $R_{ab} \leq C_{ab}$. If $a \in h_1$ and $b \in h_2$ it is trivial, noting that $C_{ca} + C_{ab} \leq C_{cb}$. The remaining case, $a \in h_1$ and $b \in h_2$, follows from 2,3,8 by simple inequational reasoning. □

Proposition 7 *For any refined type R the set $td(R)$ is a non-empty set of timing disciplines.*

Proof If R is a refined type for sort $\langle i, o\rangle$ define a timing discipline Δ by

$$\Delta_c \stackrel{def}{=} \left| \begin{array}{ll} \delta_0, & \text{if } c \in i \\ \delta_K, & \text{if } c \in o \\ \delta_0, & \text{otherwise.} \end{array} \right.$$

It is clear that $\Delta \in td(R)$. □

Proposition 9 *For any refined type R the set $td(R)$ contains only reasonable timing disciplines.*

Proof Immediate from the definitions of $td(\_)$ and reasonableness. □

Proposition 11 $s \cong_\Delta s' \Rightarrow abs^\Delta(s) = abs^\Delta(s')$

Proof It suffices, and is straightforward, to check that $\{\, abs^\Delta(s), abs^\Delta(s') \mid s \cong_\Delta s' \,\}$ is a strong bisimulation. □

Proposition 12 $s <_\Delta t \iff s \cong_\Delta s \wedge abs^\Delta(s) = t$

Proof For the implication $s <_\Delta t \Rightarrow s \cong_\Delta s$ it suffices to check that $\{\, s, s' \mid \exists t \,.\, s <_\Delta t \wedge s' <_\Delta t \,\}$ is a pre-fixed point of $\mathcal{F}_\Delta$. For the implication $s <_\Delta t \Rightarrow abs^\Delta(s) = t$ it suffices to check that $\{\, abs^\Delta(s), t \mid s <_\Delta t \,\}$ is a strong bisimulation. For $s \cong_\Delta s \wedge abs^\Delta(s) = t \Rightarrow s <_\Delta t$ it suffices to check that $\{\, s, abs^\Delta(s) \mid s \cong_\Delta s \,\}$ is a pre-fixed point of $\mathcal{E}_\Delta$. These are all straghtforward. □

Proposition 13 *If $s \in \mathcal{R}_{io}$ and $t \in \mathcal{B}_{io}$ then $s <_\Delta t$ implies $itr(s) <^{lin}_\Delta itr(t)$.*

Proof Straightforward. □

References

[Abr94] Samson Abramsky. Interaction Categories and Communicating Sequential Processes. In A. W. Roscoe, editor, *A Classical Mind: Essays in Honour of C. A. R. Hoare*, pages 1–15. Prentice Hall International, 1994.

[BNWV92] J. Bormann, H. Nusser-Wehlan, and G. Venzl. Formal design in an industrial research laboratory: Lessons and perspectives. In J. Staunstrup and R. Sharp, editors, *Designing Correct Circuits: Proceedings of the Second IFIP WG 10.2/WG 10.5 Workshop, Lyngby, Denmark*, IFIP Transactions A: Computer Science and Technology, A-5. North-Holland, 1992.

[CGM87] Albert Camilleri, Mike Gordon, and Tom Melham. Hardware verification using higher-order logic. In D. Borrione, editor, *From HDL Descriptions to Guaranteed Correct Circuit Designs: Proceedings of the IFIP WG 10.2 Working Conference, Grenoble*, pages 43–67. North-Holland, 1987. See also Technical report 91, Computer Laboratory, University of Cambridge.

[Fou95] Michael P. Fourman. Proof and design, 1995. Draft notes for the Marktoberdorf Summer School.

[Gla90] R. J. van Glabeek. The linear time — branching time spectrum. In J. C. M. Baeten and J. W. Klop, editors, *Proceedings CONCUR 90, Amsterdam, LNCS 458*, pages 278–297, 1990.

[Han94] K. Hanna. Reasoning about real circuits. In T. F. Melham and J. Camilleri, editors, *Proceedings of the 7th International Workshop on Higher Order Logic Theorem Proving and Its Applications, LNCS 859*, pages 235–253, Valletta, Malta, September 1994. Springer-Verlag.

[HD86] F. K. Hanna and N. Daeche. Specification and verification using higher-order logic: A case study. *Formal Aspects of VLSI Design*, pages 179–213, 1986.

[Her88] John Herbert. Temporal abstraction of digital designs. In G. J. Milne, editor, *The Fusion of Hardware Design and Verification*. North-Holland, 1988. See also Technical report 122, Computer Laboratory, University of Cambridge.

[Her89] John Herbert. Formal reasoning about the timing and function of basic memory devices. In L. J. M. Claesen, editor, *Formal VLSI Correctness Verification; VLSI Design Methods II*, pages 379–398. North Holland, 1989. See also Technical report 124, Computer Laboratory, University of Cambridge.

[Mel93] Thomas Frederick Melham. *Higher Order Logic and Hardware Verification*. Number 31 in Cambridge Tracts In Theoretical Computer Science. Cambridge University Press, 1993. See also Technical report 201, Computer Laboratory, University of Cambridge.

[Mil83] Robin Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice Hall International, 1989.

[Sew96] Peter Sewell. Design rules and abstractions (from branching and real time). In *3rd Workshop on Designing Correct Circuits, Båstad, Sweden*, *electronic* Workshops in Computing. Springer-Verlag, September 1996. ISBN 3–540–76102–0. To appear.

[TI84] TI. *The TTL Data Book*. Texas Instruments, 1984.

[Win87] Glynn Winskel. Relating two models of hardware. In D. H. Pitt, A. Poigné, and D. E. Rydeheard, editors, *Category Theory and Computer Science, LNCS 283*, pages 98–113. Springer-Verlag, September 1987.