Towards a Formal Semantics of Verilog Using Duration Calculus

Gerardo Schneider* and Qiwen Xu

International Institute for Software Technology
United Nations University
P.O.Box 3058, Macau
{gss,qxu}@iist.unu.edu

Abstract. We formalise the semantics of V^-, a simple version of Verilog hardware description language using an extension of Duration Calculus. The language is simple enough for experimenting formalisation, but contains sufficient features for being practically relevant. V^- programs can exhibit a rich variety of computations, and it is therefore necessary to extend Duration Calculus with several features, including Weakly Monotonic Time, infinite intervals and fixed point operators. The semantics is compositional and can be used as the formal basis of a formal theory of Verilog.

1 Introduction

Modern hardware design typically uses hardware description languages to express designs at various levels of abstraction. A hardware description language is a high level programming language, with the usual programming constructs such as assignments, conditionals and iterations, and appropriate extensions for real-time, concurrency and data structures suitable for modelling hardware. The common approach is to first build a high level design using programming constructs. The high level design is then recoded using a subset of the description language which is closer to implementation. This process may be repeated several times until the design is at a sufficiently lower level such that the hardware can be synthesised from it.

For ensuring correctness of the development, precise understanding of the description language used is apparently important. Verilog is a hardware description language widely used in industry, but its standard semantics [7] is informal. A formal semantics will also be the basis of further formal support for the language. This includes methods to prove that the highest level description satisfies the overall requirements and that a lower level description correctly implements a higher level one.

Verilog programs can exhibit a rich variety of computations, when several features of the language are intertwined. The features include

* On leave from Catholic University of Pelotas, Cx.Postal 402 (96010-000), Pelotas-RS, Brazil. Email: gerardo@atlas.vcpel.tche.br

- shared states, updated by possibly instantaneous assignments from different processes;
- delay statements;
- synchronisation by waiting for some conditions to become true;
- recursion.

It is therefore a non-trivial task to give an adequate semantics to the language. In contrast to many attempts to formalise semantics of VHDL, another popular but reportedly less used hardware description language, there is only a little work on formal semantics of Verilog [6, 13].

In this paper, we give a formal semantics to a large subset of Verilog using an extension of the Duration Calculus, aiming to achieve a satisfactory level of abstraction and a more faithful modelling of concurrency. The logic is called Duration Calculus of Weakly Monotonic Time with Infinite Intervals and Fixed Point Operator, abbreviated as μWDCI, and it has incorporated features from several recent extensions of Duration Calculus [16, 15, 17, 20].

This paper is organised as follows. In Section 2, we introduce μWDCI. The semantics of V^- is presented in Section 3. The paper is concluded with a short discussion of related work.

2 Duration Calculus of Weakly Monotonic Time with Infinite Intervals and Fixed Point Operators

Duration Calculus [1, 4], is an extension of Interval Temporal Logic (ITL) [11] to dense time domains. The classical Duration Calculus [1], abbreviated as DC, was developed to reason about piece-wise continuous Boolean functions of time (called states in literature), which model the status of the system. This view is not adequate for describing semantics of Verilog programs. The standard semantics of Verilog [7], being defined for simulation, uses a discrete event execution model. In such a model, discrete events do not take any real-time, and subsequently several of them may happen at the same real-time point governed possibly by a causal order. Such abstraction, proposed also in the work on synchronous languages, provides substantial simplification in verification of real-time systems. To reason about discrete events and their compositions, one needs a more involved logic. Koymans [9] suggested that a time point can be defined as a pair (r, n), with r denoting the real-time and n the causal order, and following the terminology in the work on synchronous languages, we call r the macro-time and n the micro-time. A variant of Duration Calculus, called Weakly Monotonic Time Duration Calculus, abbreviated as WDC, was formed by Pandya and Dang over such time structures [15]. A similar logic was suggested by Liu, Ravn and Li [10].

Both DC and WDC are defined over finite intervals. However, timed systems in general and Verilog programs in particular often exhibit infinite behaviours. In [2], an extension of DC was studied by Zhou, Dang and Li, where infinite behaviours are described by their finite approximations. This approach avoids

direct interpretation of formulae over infinite intervals but has the disadvantage that properties are somewhat cumbersome to express. As an alternative formulation, infinite intervals have been directly included in DC [19] following an approach by Moszkowski [12] for ITL. In [20], WDC is extended with infinite intervals along both macro-time and micro-time. To describe recursive computations, Pandya and Ramakrishna [16] introduced fixed point operators in DC. Properties of the fixed points have been further studied in [17]. Our logic, μWDCI, is obtained by incorporating all these features.

We next give an overview of μWDCI.

Definition 1. *A* time domain *is a total order* $(\mathcal{T}, <)$*, where*

- $\mathcal{T} \subseteq (\Re^+ \cup \{\infty\}) \times (\aleph \cup \{\infty\})$*, with* $(0,0) \in \mathcal{T}$*,*
- $(r_1, n_1) < (r_2, n_2)$
  iff $(r_1 \neq \infty \wedge r_1 \leq r_2 \wedge n_1 < n_2) \vee (r_1 < r_2 \wedge n_1 \neq \infty \wedge n_1 \leq n_2)$*,*
- *for any* $t \notin \mathcal{T}$*, there exists* $t' \in \mathcal{T}$*, such that* $t \not\leq t'$ *and* $t' \not\leq t$*,*
- $lim((r_i, n_i)) \in \mathcal{T}$*.*

where $\Re^+$ is the set of non-negative real numbers and $\aleph$ is the set of natural numbers.

The first and second conditions define the set and the order relationship $(<)$. The third condition says that a time domain is maximal, in the sense that adding any other time point will cause the set to be no longer a total order. By the definition, exactly one of (r, ∞), (∞, n) and (∞, ∞) is in a time domain and it is the maximal element. The last condition implies that if the time domain has an infinite sequence of time points of the form $(r, n_0), (r, n_1), \ldots, (r, n_n), \ldots$ for the same real number r, then (r, ∞) must be in the sequence. This in turn by the third condition ensures that in this case any other infinite time point, either (r', ∞) where $r \neq r'$ or (∞, ∞) is not in the domain. For any $t = (r, n)$, let $\pi_1(t) \stackrel{\text{def}}{=} r$ and $\pi_2(t) \stackrel{\text{def}}{=} n$. We write $t = \infty$ iff $n = \infty$ or $r = \infty$. An interval on $\mathcal{T}$ is a pair of time points $[b, e]$ of $\mathcal{T}$, where $b \neq \infty$ and $b \leq e$.

The logic contains the following sets of symbols: global variables $GVar = \{x, y, \ldots\}$, state variables $SVar = \{P, Q, \ldots\}$ and temporal propositional letters $PLetter = \{X, Y, \ldots\}$. State expressions are defined as

$$S \ ::= \ c \mid x \mid P \mid op_1 \ S \mid \ S_1 \ op_2 \ S_2$$

where c is a constant, $op_1 \in \{-, \neg\}$ and $op_2 \in \{\wedge, \vee, +, -, *, =, <\}$ are unary and binary operators of appropriate types. The syntax of terms is

$$T \ ::= \ c \mid x \mid \int S \mid \hat{\int} S \mid T_1 \ op \ T_2 \mid \mathbf{b}.S \mid \mathbf{e}.S$$

where $op \in \{\wedge, \vee, +, -, *\}$, $\mathbf{b}.S$ and $\mathbf{e}.S$ denote respectively the values of S at the beginning and the end of the interval. When S is Boolean, represented as $\{0, 1\}$, $\int S$ and $\hat{\int} S$ denote respectively durations of the expression along macro-time and micro-time.

The syntax of μWDCI formulae is

$$X \mid t_1 = t_2 \mid t_1 < t_2 \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \exists x \,.\, \phi \mid \phi_1 \frown \phi_2 \mid \mu X.\phi$$

where x is a global variable, and the occurrence of bounded propositional letter X in the fixed point formula is positive (i.e., X is preceded by even number of negation symbols).

Definition 2. *A* model **M** *is a tuple* $((\mathcal{T},<),\mathcal{I},\mathcal{J},\mathcal{V},[b,e])$, *where*

- $(\mathcal{T},<)$ *is a time domain,*
- $\mathcal{I}$: SVar $\rightarrow \mathcal{T} \rightarrow$ Values *is an interpretation of state variables where* Values $= \Re^+ \cup \aleph \cup$ Bool,
- $\mathcal{J}$: Pletter $\rightarrow$ *Intv* $\rightarrow$ Bool *is an interpretation of propositional letters, where Intv is the set of all intervals,*
- $\mathcal{V}$: GVar $\rightarrow$ Values *is an interpretation of global variables,*
- $[b,e]$ *is an interval on* $\mathcal{T}$.

Let t be a time point. The interpretation of state expressions is

- $[\![x]\!](\mathcal{T},\mathcal{I},\mathcal{V},t) \stackrel{\text{def}}{=} \mathcal{V}(x)$
- $[\![P]\!](\mathcal{T},\mathcal{I},\mathcal{V},t) \stackrel{\text{def}}{=} \mathcal{I}(P)(t)$
- $[\![op_1\ S]\!](\mathcal{T},\mathcal{I},\mathcal{V},t) \stackrel{\text{def}}{=} op_1^*\ [\![S]\!](\mathcal{T},\mathcal{I},\mathcal{V},t)$
- $[\![S_1\ op_2\ S_2]\!](\mathcal{T},\mathcal{I},\mathcal{V},t) \stackrel{\text{def}}{=} [\![S_1]\!](\mathcal{T},\mathcal{I},\mathcal{V},t)\ op_2^*\ [\![S_2]\!](\mathcal{T},\mathcal{I},\mathcal{V},t)$

where op_i^* is the interpretation of the operator op_i. Let

$$[\![\bar{S}]\!](\mathcal{T},\mathcal{I},\mathcal{V},r) \stackrel{\text{def}}{=} \begin{cases} [\![S]\!](\mathcal{T},\mathcal{I},\mathcal{V},(r,n)) & \text{if } \{n|(r,n) \in \mathcal{T}\} \text{ is singleton} \\ 0 & \text{otherwise} \end{cases}$$

$$[\![\hat{S}]\!](\mathcal{T},\mathcal{I},\mathcal{V},n) \stackrel{\text{def}}{=} \begin{cases} [\![S]\!](\mathcal{T},\mathcal{I},\mathcal{V},(r,n)) & \text{if } (r,n),\ (r,n+1) \in \mathcal{T} \\ 0 & \text{otherwise.} \end{cases}$$

The first definition is well-formed because if $\{n|(r,n) \in \mathcal{T}\}$ is singleton, then obviously $n_1 = n_2$ for any $(r,n_1) \in \mathcal{T}$ and $(r,n_2) \in \mathcal{T}$. The second definition is also well-formed because for any $(r_1,n) \in \mathcal{T}$, $(r_2,n) \in \mathcal{T}$, $(r_1,n+1) \in \mathcal{T}$ and $(r_2,n+1) \in \mathcal{T}$, it is easy to prove $r_1 = r_2$. The interpretation of terms is

- $[\![\int S]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e]) \stackrel{\text{def}}{=} \int_{\pi_1(b)}^{\pi_1(e)} [\![\bar{S}]\!](\mathcal{T},\mathcal{I},\mathcal{V},r)dr$
- $[\![\hat{\int} S]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e]) \stackrel{\text{def}}{=} \sum_{n=\pi_2(b)}^{\pi_2(e)-1} [\![\hat{S}]\!](\mathcal{T},\mathcal{I},\mathcal{V},n)$
- $[\![T_1\ op\ T_2]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e]) \stackrel{\text{def}}{=} [\![T_1]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e])\ op^*\ [\![T_2]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e])$
- $[\![\mathbf{b}.S]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e]) \stackrel{\text{def}}{=} [\![S]\!](\mathcal{T},\mathcal{I},\mathcal{V},b)$
- $[\![\mathbf{e}.S]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e]) \stackrel{\text{def}}{=} [\![S]\!](\mathcal{T},\mathcal{I},\mathcal{V},e)$

where op^* is the interpretation of op. The lengths of an interval along macro-time and micro-time are denoted by $\ell \stackrel{\text{def}}{=} \int 1$ and $k \stackrel{\text{def}}{=} \hat{\int} 1$. It is easy to show that

- $[\![\ell]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e]) = \pi_1(e) - \pi_1(b)$
- $[\![k]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e]) = \pi_2(e) - \pi_2(b)$

To define the fixed point operators by Knaster-Tarski theorem, we give the semantics of μWDCI in the complete lattice $(2^{\mathrm{Intv}}, \subseteq)$, where Intv is the set of all intervals. In this setting, an interpretation of propositional letters $\mathcal{J}$ is regarded as a function from propositional letters to 2^{Intv}. For a given time domain $(\mathcal{T},<)$, an interpretation of state variables $\mathcal{I}$, an interpretation of propositional letters $\mathcal{J}$, a valuation of global variables $\mathcal{V}$, we define a function $\mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}$ from the set of μWDCI formulae to 2^{Intv}.

$$\mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(X) \stackrel{\text{def}}{=} \mathcal{J}(X)$$
$$\mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(T_1 = T_2) \stackrel{\text{def}}{=} \{[b,e] \mid \{b,e\} \subset \mathcal{T} \wedge [\![T_1]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e]) = [\![T_2]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e])\}$$
$$\mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(T_1 < T_2) \stackrel{\text{def}}{=} \{[b,e] \mid \{b,e\} \subset \mathcal{T} \wedge [\![T_1]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e]) < [\![T_2]\!](\mathcal{T},\mathcal{I},\mathcal{V},[b,e])\}$$
$$\mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\neg\phi) \stackrel{\text{def}}{=} 2^{\mathrm{Intv}} - \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi)$$
$$\mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi_1 \wedge \phi_2) \stackrel{\text{def}}{=} \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi_1) \cap \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi_2)$$
$$\mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\exists x.\phi) \stackrel{\text{def}}{=} \bigcup_{a\in \mathsf{Values}} \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V}(x\mapsto a),\mathcal{T}}(\phi)$$
$$\mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi_1 \frown \phi_2) \stackrel{\text{def}}{=} \{[b,e] \mid \exists m \,.\, \{b,e,m\} \subset \mathcal{T} \wedge b \le m \le e \wedge (m \neq \infty \wedge [b,m] \in \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi_1) \wedge [m,e] \in \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi_2)) \vee (e = \infty \wedge [b,e] \in \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi_1))\}$$
$$\mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\mu X.\phi) \stackrel{\text{def}}{=} \bigcap\{A \mid \mathcal{E}^{\mathcal{I},\mathcal{J}(X\mapsto A)}_{\mathcal{V},\mathcal{T}}(\phi) \subseteq A\}$$

where $\mathcal{J}(X \mapsto A)$ is the interpretation of the propositional letters that is the same as $\mathcal{J}$ except mapping X to A. The greatest fixed point operator can be defined from the the least fixed point in the usual way

$$\nu X.\phi \stackrel{\text{def}}{=} \neg\mu Y.\neg\phi[\neg Y/X]$$

where $\phi[\neg Y/X]$ is the substitution of $\neg Y$ for all the occurrences of the propositional letter X in formula ϕ.

The notions of *satisfaction* and *validity* are defined as follows:

- $(\mathcal{T},\mathcal{I},\mathcal{J},\mathcal{V},[b,e]) \models \phi$ iff $[b,e] \in \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi)$
- $\models \phi$ iff $\mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi) = 2^{\mathrm{Intv}}$ for any $\mathcal{T},\mathcal{I},\mathcal{J},\mathcal{V}$.

The following two theorems give the semantics of two fixed point formulae which will be used in defining the semantics of iteration statement.

Theorem 1. *Let ϕ_1, ϕ_2 be two μWDCI formulae. If both ϕ_1 and ϕ_2 do not contain any free occurrence of the propositional variable symbol X, then for any $\mathcal{T},\mathcal{I},\mathcal{J},\mathcal{V}$, $[b,e] \in \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\mu X.(\phi_1 \frown X)) \vee \phi_2)$ iff there is a natural number n (may be 0) and a non-descending sequence of time points $b_0, b_1, \ldots, b_n$ such that*

- $b_0 = b, b_1, \ldots, b_n \in \mathcal{T} - \{\infty\}$,
- $[b_i, b_{i+1}] \in \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi_1)$ *for all* $i < n$,

– $[b_n, e] \in \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi_2)$, *or* $e = \infty$ *and* $[b_n, e] \in \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi_1)$.

Theorem 2. *Let* ϕ_1, ϕ_2 *be two* μ*WDCI formulae. If both* ϕ_1 *and* ϕ_2 *do not contain any free occurrence of the propositional variable symbol* X*, then for any* $\mathcal{T}, \mathcal{I}, \mathcal{J}, \mathcal{V}$,

$$\mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}\Big(\nu X.((\phi_1 \frown X) \vee \phi_2)\Big) = \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}\Big(\mu X.((\phi_1 \frown X) \vee \phi_2)\Big) \cup E$$

where $[b, e] \in E$ *iff there is an infinite non-descending sequence of time points* $b_0, b_1, \ldots, b_n, \ldots$ $(n \geq 0)$ *such that:*

– $b_0 = b$, $b_n \leq e$ *for all* $n \geq 0$,
– $[b_n, b_{n+1}] \in \mathcal{E}^{\mathcal{I},\mathcal{J}}_{\mathcal{V},\mathcal{T}}(\phi_1)$ *for all* $n \geq 0$.

Details about these theorems and other properties about fixed points can be found in [18]. We next introduce some derived modalities that will be useful when defining the semantics of V^-.

$$\Diamond A \stackrel{\text{def}}{=} \mathsf{true} \frown A \frown \mathsf{true}$$
$$\Box A \stackrel{\text{def}}{=} \neg\Diamond\neg A.$$

A model satisfies $\Diamond A$ and $\Box A$ if respectively a sub-interval and all sub-intervals satisfy A. Let

$$\mathsf{fin} \stackrel{\text{def}}{=} \exists x.\, (\ell < x \wedge k < x)$$
$$\widehat{\mathsf{fin}} \stackrel{\text{def}}{=} \exists x.\, k < x$$
$$\mathsf{inf} \stackrel{\text{def}}{=} \neg\mathsf{fin}$$
$$\mathsf{point} \stackrel{\text{def}}{=} \ell = 0 \wedge k = 0.$$

They characterise intervals which respectively are finite, finite on micro-time, infinite and points. For a Boolean expression S, define

$$\lceil S \rceil \stackrel{\text{def}}{=} \neg((\ell > 0 \vee k > 0) \frown (\mathsf{point} \wedge \neg \mathbf{b}.S) \frown (\ell > 0 \vee k > 0)).$$

This denotes that S holds everywhere inside the interval. Let

$$\lceil\!\lceil S \rceil\!\rceil \stackrel{\text{def}}{=} \lceil S \rceil \wedge \mathbf{b}.S \wedge \mathbf{e}.S.$$

This specifies that S holds everywhere inside and at both the beginning and the end of the interval. Let

$$\mathsf{dint} \stackrel{\text{def}}{=} \ell = 0 \wedge k = 1$$
$$\mathsf{cint} \stackrel{\text{def}}{=} \ell > 0 \wedge k = 0.$$

Intervals that satisfy cint and dint are called respectively continuous and discrete.

3 Semantics of V^-

Gordon [6] suggested a simple version of Verilog which he called V. In this paper, we consider a subset of it which we denote by V^-, that has essentially the same statements as V except the function definition, the non-blocking assignment, the `disable` statement and the assignment statements with delays. The language is simple enough for experiments in formalisation, but contains sufficient features for being practically relevant.

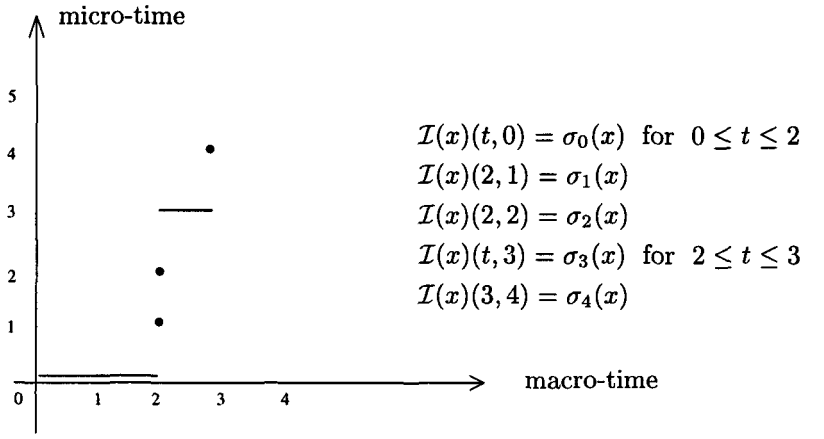
In the literature, the semantics usually is considered as for the program concerned defining a set of *runs*, each of which is a sequence

$$\sigma : (\sigma_0, r_0)(\sigma_1, r_1) \cdots (\sigma_i, r_i) \cdots$$

where σ_i is a valuation of variables and r_i is a real number denoting the macro-time that the variables are updated. Between two consecutive time points, variables are not changed. If there are several discrete events happening at the same macro-time, they will have the same time stamps and micro-time is denoted by the order in the sequence. A run σ can be regarded as the interpretation of the variables over a time domain. For example, assume the variable is x, and for a run

$$\sigma : (\sigma_0, 0)(\sigma_1, 2)(\sigma_2, 2)(\sigma_3, 2)(\sigma_4, 3) \cdots$$

the corresponding time domain is illustrated by the following diagram



Time domain corresponding to σ

Therefore a μWDCI formula can be regarded as characterising a set of runs, and consequently, can be used to define the semantics of V^- programs. This gives the abstraction and reasoning facilities provided by a logic system.

At the top level, a V^- program can be considered as a collection of concurrent processes, communicating through shared variables. When there are discrete transitions enabled, one of them will be selected for execution. When none of the discrete transitions are enabled, time may advance until a discrete transition is enabled.

It is desirable to define the semantics compositionally, that is, define the semantics of a compound statement as a function of the semantics of the components. The standard way to give a compositional semantics to shared variable programs, suggested first by Aczel (cited e.g., in [5]), is to define the semantics as a set of labelled state transition sequences, where the label records whether the transition is from the environment or from the component. This can be expressed in the setting of μWDCI by introducing a variable, say ∂, in the state, to record the transition agent. More precisely, let $\{P_i \mid i \in I_{Proc}\}$ be the set of processes, where I_{Proc} is a set of indexes. If there is a transition at $[(r,n),(r,n+1)]$, then the transition is from process P_i iff $\mathcal{I}(\partial)(r,n+1) = i$. To define the semantics compositionally, a component process P_i should be viewed as an open system, with its semantics containing all the potential actions from its environment processes. Runs with unmatching environment actions are removed by the semantics of the parallel composition.

Let Var be the set of variables from the concerned program and $Var^+ \stackrel{\text{def}}{=} Var \cup \{\partial\}$. The semantics contains a general formula

$$\mathbf{Ax_1} : \forall x \in Var^+ \,.\, \Box(\mathsf{cint} \Rightarrow \mathbf{b}.x = \mathbf{e}.x)$$

This says that none of the variables are changed if there are no transitions. The semantics of the whole system, considered as a closed system contains the formula:

$$\mathbf{Ax_2} : \exists i \in I_{Proc} \,.\, \Box(\mathsf{dint} \Rightarrow \mathbf{e}.\partial = i).$$

This says that any transition is caused by one of the constituent process.

In the following, we assume that the considered sequential statements are from process P_i. As the semantics of an open system, it is necessary to include possible behaviours from other processes. The formula

$$\mathsf{idle}_i \stackrel{\text{def}}{=} \Box(\mathsf{dint} \Rightarrow \mathbf{e}.\partial \neq i)$$

says that over the interval, process P_i does not contribute any discrete transitions. There is no restriction over transitions by processes other than P_i, or in other words, transitions from other processes can be arbitrary.

Procedural Assignment. This is the usual assignment, denoted by $v = e$. It is called *procedural* in Verilog to distinguish it from other forms of assignments such as the continuous assignment which will be discussed later. Evaluation of expression e is assumed to take one micro-time, and the value is assigned to variable v, with other variables unchanged. Before the assignment is executed, its environment processes may perform an arbitrary, including infinite, number of instantaneous actions.

$$\mathcal{M}(v = e) \stackrel{\text{def}}{=} \ell = 0 \wedge (\mathsf{idle}_i \frown (\mathsf{dint} \wedge \mathbf{e}.v = \mathbf{b}.e \wedge \mathit{unchanged}_{Var-\{v\}} \wedge \mathbf{e}.\partial = i))$$

where $\mathit{unchanged}_{Var-\{v\}} \stackrel{\text{def}}{=} \bigwedge_{x \in Var-\{v\}} \exists a.\lceil x = a \rceil$, denoting that any variable in $Var - \{v\}$ is not changed. A vacuous assignment like $x = x$ is denoted as `skip`.

Sequential Composition. The execution of sequential composition `begin` $S_1; \ldots;$ S_n `end` is that of S_1 followed by the execution of `begin` $S_2; \ldots; S_n$ `end`.

$$\mathcal{M}(\texttt{begin } S_1; S_2; \ldots; S_n \texttt{ end}) \stackrel{\text{def}}{=} \mathcal{M}(S_1) \frown \mathcal{M}(\texttt{begin } S_2; \ldots; S_n \texttt{ end})$$

Parallel Composition. The parallel composition of the processes is defined roughly as the conjunction

$$\mathcal{M}(P_1 \parallel \ldots \parallel P_n) \stackrel{\text{def}}{=} \bigvee_{i=1}^{n} ((\mathcal{M}(P_1) \frown \mathsf{idle}_1) \wedge \ldots \wedge \mathcal{M}(P_i) \wedge \ldots \wedge (\mathcal{M}(P_n) \frown \mathsf{idle}_n)).$$

Boolean Expressions. Successful evaluation of a Boolean expression is defined as:

$$\mathcal{M}(eb) \stackrel{\text{def}}{=} \ell = 0 \wedge (\mathsf{idle}_i \frown (\mathbf{b}.eb \wedge \mathsf{dint} \wedge \mathit{unchanged}_{Var} \wedge \mathbf{e}.\partial = i)).$$

Like a procedural assignment, successful evaluation of a Boolean expression takes one micro-time, and its environment processes may perform an arbitrary number of instantaneous actions before the expression is evaluated. Evaluation has no side-effect. For Boolean constants true and false, we have

- $\mathcal{M}(\mathsf{true}) = (\ell = 0 \wedge (\mathsf{idle}_i \frown (\mathsf{dint} \wedge \mathit{unchanged}_{Var} \wedge \mathbf{e}.\partial = i)))$
- $\mathcal{M}(\mathsf{false}) = (\ell = 0 \wedge (\mathsf{idle}_i \frown \mathsf{false})) = (\ell = 0 \wedge \mathsf{idle}_i \wedge \mathsf{inf})$.

Conditional. In `if` (eb) S_1 `else` S_2, the Boolean expression eb is evaluated first. If eb is successfully evaluated, then S_1 is selected, and if instead $\neg eb$ is successfully evaluated, then S_2 is selected.

$$\mathcal{M}(\texttt{if } (eb)\, S_1 \texttt{ else } S_2) \stackrel{\text{def}}{=} (\mathcal{M}(eb) \frown \mathcal{M}(S_1)) \vee (\mathcal{M}(\neg eb) \frown \mathcal{M}(S_2)).$$

Instantaneous environment actions are allowed between the evaluation of the Boolean expression and the execution of the statements that follows.

Delay. The delay statement in Verilog is denoted by $\#n$. Its meaning is that when this instruction is reached, the process is blocked for n macro-time units. During this period, other processes may execute discrete state transitions.

$$\mathcal{M}(\#n) \stackrel{\text{def}}{=} \mathsf{idle}_i \wedge \ell \leq n \wedge (\widehat{\mathsf{fin}} \Rightarrow \ell = n).$$

Note that it is possible that before n macro-time units have passed, other processes can execute an infinite number of instantaneous transitions and cause the micro-time to reach infinity.

Iteration. The meaning of a loop `while` $(eb)\ S$ is as usual: the statement is executed repeatedly until the boolean expression eb becomes false. Its semantics is defined as a greatest fixed point:

$$\mathcal{M}(\texttt{while}\ (eb)\ S) \stackrel{\text{def}}{=} \nu X.((\mathcal{M}(eb)\frown\mathcal{M}(S)\frown X)\vee\mathcal{M}(\neg eb)).$$

It follows from Theorem 2 that the semantics of the iteration statement contains two parts. The first part is defined by the least fixed point $\mu X.((\mathcal{M}(eb)\frown\mathcal{M}(S)\frown X)\vee\mathcal{M}(\neg eb))$. This describes finite number of iterations, either eb becomes false in the end, or the last iteration is infinite. The second part characterises infinite iterations of the loop. Since we assume Boolean evaluation takes one unit micro-time, infinite iterations take infinite micro-time. There are two cases regarding the macro-time. One case is that after some iterations, each iteration takes zero macro-time, then the macro-time will be 'stopped' at that point. The second case is that infinite iterations need infinite macro-time. In V^{-}, we assume a positive delay has a lower bound, therefore the so-called zeno behaviour is not possible.

The following are some examples about the semantics of iteration statements.

1. $\mathcal{M}(\texttt{while}\ (\mathsf{true})\ \texttt{skip}) = (\ell = 0 \wedge \mathsf{inf} \wedge \Box(\mathbf{e}.\partial = i \Rightarrow \mathit{unchanged}_{Var}))$
2. $\mathcal{M}(\texttt{while}\ (\mathsf{false})\ P) = \mathcal{M}(\texttt{skip})$
3. $\mathcal{M}(\texttt{while}\ (\mathsf{true})\ \texttt{skip} \parallel \#1) = \mathcal{M}(\texttt{while}\ (\mathsf{true})\ \texttt{skip}) \wedge \mathsf{idle}_2$.

Wait. The wait statement `wait` $(eb)\ S$ waits until Boolean expression eb becomes true, then the control is passed to the following statement S.

$$\mathcal{M}(\texttt{wait}\ (eb)\ S) \stackrel{\text{def}}{=} (\mathsf{idle}_i \wedge (\lceil\neg eb\rceil\frown(\mathsf{dint} \wedge \mathbf{e}.eb)))\frown\mathcal{M}(S)$$

Event Control. Synchronisation can also be achieved by waiting for some events (state changes) to occur. The statement is of the form

$$@(event)\ S$$

where *event* is considered as a binary state predicate, and the control is passed to S when *event* is satisfied. An event expression is of the following form

- v, indicating waiting for a change of the value in v,
- `posedge` v, indicating waiting for a positive change of the value in v,
- `negedge` v, indicating waiting for a negative change of the value in v,
- $(event_1 \texttt{or}\ \ldots \texttt{or}\ event_n)$, indicating waiting for any event to be true.

Overloading symbols a little, we define

$$@(v) \stackrel{\text{def}}{=} \mathbf{b}.v \neq \mathbf{e}.v$$
$$@(\texttt{posedge}\ v) \stackrel{\text{def}}{=} \mathbf{b}.v < \mathbf{e}.v$$
$$@(\texttt{negedge}\ v) \stackrel{\text{def}}{=} \mathbf{b}.v > \mathbf{e}.v$$
$$@(event_1 \texttt{or}\ \ldots \texttt{or}\ event_n) \stackrel{\text{def}}{=} @(event_1) \vee \ldots \vee @(event_n)$$

It is now easy to give the semantics of the statement

$$\mathcal{M}(@(event)\ S) \stackrel{\text{def}}{=} (\mathsf{idle}_i \wedge ((\Box\neg@(event))\frown(\mathsf{dint} \wedge @(event))))\frown\mathcal{M}(S)$$

Continuous assignment. A continuous assignment `assign` $w = e$ describes the connection of a wire w to an input. Like a channel, a wire does not store the value, and any change of the input is immediately propagated

$$\mathcal{M}(\texttt{assign } w = e) \stackrel{\text{def}}{=} \lceil w = \mathcal{M}(e) \rceil \wedge \mathsf{inf}.$$

The semantics of other V^- statements can be found in [18].

4 Discussion

In this paper, we have given a formal semantics to V^-, a simple version of Verilog. The language contains interesting features like concurrency, timing behaviours, discrete events as well as recursion. Formalising more advanced features of Verilog, such as various other assignments, needs further research. On the other hand, V^- has contained the basics of Verilog, and therefore it will be useful to develop other formal techniques for V^- based on the semantics. However, to support particular techniques, the semantics may need fine tuning; for example, for refinement, the semantics should probably be made insensitive to stuttering.

There has been some related work where semantics of several languages have been formalised using Duration Calculus and its appropriate extensions. He [8], Zhou, Wang and Ravn [3] studied semantics of HCSP in classical DC, which is sufficient because every action is assumed to take some time. Semantics of sequential real-time programs, which may contain instantaneous actions, has been studied in [17] using Super Dense DC (SDC). SDC allows the input/output behaviours of a sequential program to be recorded, but not the intermediate states. SDC is similar to the Relational DC, which has been used by Pace to a subset of Verilog [13]. Hence, Pace also does not record the intermediate states of Verilog programs. To describe the semantics of concurrency adequately, Pandya and Dang proposed to use WDC [15] and applied to an ESTEREL-like language called SL [14]. WDC is extended with infinite intervals in [20] and applied to formalise the semantics of a toy language.

Acknowledgements We would like to thank He Jifeng for introducing Verilog to us, Gordon Pace for discussing his work, and Dang Van Hung, Paritosh Pandya and Zhou Chaochen for their useful comments in the initial stage of this work.

References

1. Zhou Chaochen, C.A.R. Hoare, and A.P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.
2. Zhou Chaochen, Dang Van Hung, and Li Xiaoshan. A duration calculus with infinite intervals. In *Fundamentals of Computation Theory, Horst Reichel (Ed.)*, pages 16–41. LNCS 965, Springer-Verlag, 1995.
3. Zhou Chaochen, Wang Ji, and A.P. Ravn. A formal description of hybrid systems. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III: Verification and Control*, pages 511–530. LNCS 1066, Springer–Verlag, 1995.

4. Zhou Chaochen, A.P. Ravn, and M.R. Hansen. An extended duration calculus for hybrid systems. In *Hybrid Systems, R.L. Grossman, A. Nerode, A.P. Ravn, H. Rischel (Eds.)*, pages 36–59. LNCS 736, Springer-Verlag, 1993.
5. W.-P. de Roever. The quest for compositionality. In *Proc. of IFIP Working Conf., The Role of Abstract Models in Computer Science.* Elsevier Science B.V. (North-Holland), 1985.
6. M.J.C. Gordon. The Semantic Challenge of Verilog HDL. In *Tenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, pages 136–145, June 1995.
7. IEEE Computer Society. *IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language (IEEE std 1364-1995*, 1995.
8. He Jifeng. From CSP to hybrid systems. In A.W. Roscoe, editor, *A Classical Mind, Eassy in Honour of C.A.R. Hoare*, pages 171–189. Prentice-Hall International, 1994.
9. R. Koymans. *Specifying Message Passing and Time-Critical Systems with Temporal Logic.* LNCS 651, Springer–Verlag, 1992.
10. Z. Liu, A.P. Ravn, and X.-S. Li. Verifying duration properties of timed transition systems. In *Proc. IFIP Working Conference PROCOMET'98.* Chapman & Hall, 1998.
11. B. Moszkowski. A temporal logic for multilevel reasoning about hardware. *IEEE Computer*, 18(2):10–19, 1985.
12. B. Moszkowski. Compositional reasoning about projected and infinite time. In *Proc. the First IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'95)*, pages 238–245. IEEE Computer Society Press, 1995.
13. G.J. Pace and J.-F. He. Formal reasoning with Verilog HDL. In *Proc. the Workshop on Formal Techniques for Hardware and Hardware-like Systems*, Sweden 1988.
14. P.K. Pandya. A compositional semantics of SL. Technical report, DeTfoRS Group, UNU/IIST, October 1997.
15. P.K. Pandya and Dang Van Hung. Duration calculus with weakly monotonic time. This volume.
16. P.K. Pandya and Y.S. Ramakrishna. A recursive duration calculus. Technical report, CS-95/3, Computer Science Group, TIFR, Bombay, 1995.
17. P.K. Pandya, H.-P. Wang, and Q.-W. Xu. Towards a theory of sequential hybrid programs. In *Proc. IFIP Working Conference PROCOMET'98.* Chapman & Hall, 1998.
18. G. Schneider and Q.-W. Xu. Towards a Formal Semantics of Verilog using Duration Calculus. Technical Report 133, UNU/IIST, P.O.Box 3058, Macau, February 1998.
19. H.-P. Wang and Q.-W. Xu. Infinite duration calculus with fixed-point operators. Technical Report draft, UNU/IIST, P.O.Box 3058, Macau, September 1997.
20. Q.-W. Xu and M. Swarup. Compositional reasoning using assumption - commitment paradigm. In H. Langmaack, A. Pnueli, and W.-P. de Roever, editors, *International Symposium, Compositionality – The Significant Difference.* Springer–Verlag, 1998.