

# CBV Semantics (Draft)

NAVIT FEDIDA, JOHN HAVLICEK, NISSAN LEVI, AND HILLEL MILLER  
Motorola, Inc.  
7700 W. Parmer Lane  
Austin, TX 78729

13 January 2002

## ABSTRACT

A precise definition is given of the semantics of the specification-writing subset of CBV, excluding “if-tasks”. Familiarity with CBV syntax is assumed.

## 0. NOTATIONS

Let  $\mathbb{B}$  denote the set  $\{0, 1\}$ . A *binary variable* consists of an identifier and a positive integer width. We refer to a binary variable by its identifier. If  $s$  is a binary variable of width  $k \geq 1$ , a *valuation* of  $s$  is a pair  $(s, v)$ , where  $v \in \mathbb{B}^k$ . A binary variable of width 1 will be called a *boolean variable*.

Let  $\mathfrak{S}$  be a finite set of binary variables. A *valuation* of  $\mathfrak{S}$  is a set of valuations of the variables of  $\mathfrak{S}$  that contains exactly one valuation for each variable of  $\mathfrak{S}$ . A *partial valuation* of  $\mathfrak{S}$  is a subset of a valuation of  $\mathfrak{S}$ . The *domain* of a partial valuation of  $\mathfrak{S}$  is the set of binary variables in  $\mathfrak{S}$  for which the partial valuation contains variable valuations. We use function notation for partial valuations of  $\mathfrak{S}$ . For example, if  $\mathfrak{S} = \{s, t, u\}$ , where  $s, u$  are boolean and  $t$  has width 2, then  $f = \{(s, 0), (t, (1, 0))\}$  is a partial valuation of  $\mathfrak{S}$ . We write  $f(s) = 0$ ,  $f(t) = (1, 0)$ , and  $\text{dom}(f) = \{s, t\}$  in this case.

A *trace* of  $\mathfrak{S}$  is a sequence (finite or infinite) of valuations of  $\mathfrak{S}$ . If  $x$  is a trace, we generally write  $x = x_0x_1\dots$ , where  $x_0, x_1, \dots$  are the individual valuations of the sequence.

Let  $x = x_0x_1\dots$  be a trace of  $\mathfrak{S}$ , let  $s$  be a boolean variable in  $\mathfrak{S}$ , and let  $i > 0$  be an integer. We say that

- $x, i \models \text{posedge}(s)$  iff  $x_i(s) > x_{i-1}(s)$ .
- $x, i \models \text{negedge}(s)$  iff  $x_i(s) < x_{i-1}(s)$ .
- $x, i \models \text{anyedge}(s)$  iff  $x_i(s) \neq x_{i-1}(s)$ .

An *augmented trace* of  $\mathfrak{S}$  is a pair  $(f, x)$ , where  $f$  is a partial valuation of  $\mathfrak{S}$  and  $x$  is a trace of  $\mathfrak{S}$ .

Let  $x = x_0x_1\dots$  be a trace of  $\mathfrak{S}$ , let  $s$  be a boolean variable in  $\mathfrak{S}$ , and let  $f$  be a partial valuation of  $\mathfrak{S}$  such that  $s \in \text{dom}(f)$ . Then we say that

- $(f, x), 0 \models \text{posedge}(s)$  iff  $x_0(s) > f(s)$ .

- $(f, x), 0 \models \text{negedge}(s)$  iff  $x_0(s) < f(s)$ .
- $(f, x), 0 \models \text{anyedge}(s)$  iff  $x_0(s) \neq f(s)$ .

For  $i > 0$  and  $e \in \{\text{posedge}, \text{negedge}, \text{anyedge}\}$ ,  $(f, y), i \models e(s)$  iff  $y, i \models e(s)$ .

Let  $e$  be a binary expression with support in  $\mathfrak{S}$  and let  $f$  be a partial valuation of  $\mathfrak{S}$ . By  $e[f]$  we mean the expression obtained from  $e$  by replacing all occurrences in  $e$  of all variables in  $\text{dom}(f)$  with the corresponding values assigned to those variables in  $f$ . If  $g$  is also a partial valuation of  $\mathfrak{S}$ , then by  $e[f][g]$  we mean the iteration  $(e[f])[g]$ .

If  $e$  is of width  $k$  and  $f$  is a partial valuation that defines every variable in  $e$ , then  $e[f]$  is a constant expression equivalent to an element of  $\mathbb{B}^k$ , and we will identify  $e[f]$  with this value in  $\mathbb{B}^k$ . For  $s \in \mathfrak{S}$ , notice that  $s[f] = f(s)$  or  $s[f] = s$  according as  $s \in \text{dom}(f)$  or  $s \notin \text{dom}(f)$ .

## 1. SEMANTICS

Let  $M$  be a CBV module. Let  $\mathfrak{D}$  be the set of design signals of  $M$ , let  $\mathfrak{A}$  be the set of CBV assign variables of  $M$ , let  $\mathfrak{V}$  be the set of CBV var variables of  $M$ , and let  $\mathfrak{L}$  be the set of CBV local variables of  $M$ . These are all sets of binary variables.  $\mathfrak{D}$ ,  $\mathfrak{A}$ , and  $\mathfrak{V}$  are disjoint. [A CBV variable overshadows a design signal with the same identifier.] Write  $\mathfrak{S}$  to denote the union  $\mathfrak{D} \cup \mathfrak{A} \cup \mathfrak{V}$ .  $\mathfrak{L}$  may have nonempty intersection with  $\mathfrak{S}$ .

Let  $\mathfrak{C} \subseteq \mathfrak{S}$  denote the set of boolean variables that are declared as clocks in  $M$ .

If  $e \in \{\text{posedge}, \text{negedge}, \text{anyedge}\}$  and  $c \in \mathfrak{C}$ , then the pair  $(e, c)$  is a *CBV clock event* for  $M$ . There is a default CBV clock event associated with  $M$ .

A CBV assign variable definition in  $M$  is of the form

$$\text{assign } a = a_{\text{now}};$$

where  $a_{\text{now}}$  is a binary expression with support in  $\mathfrak{S}$ . Cyclic dependencies among assign variables are illegal.

A CBV var variable definition in  $M$  is of the form

$$\text{var calculate } @(e\ c)\ v = v_{\text{next}};$$

where  $(e, c)$  is a CBV clock event for  $M$  and  $v_{\text{next}}$  is a binary expression with support in  $\mathfrak{S}$ . If the “calculate @” is omitted, then the var variable is understood to be calculated at the default clock event for  $M$ . [CBV allows definition of a finest clock by

$$\text{var calculate } @(\text{anyedge } F)\ F = !F;$$

$F$  must be set in motion by an initialization that produces its first edge.]

Let  $x$  be a trace of  $\mathfrak{D}$ , let  $f$  be a valuation of  $\mathfrak{C}$ , and let  $V$  be a valuation of  $\mathfrak{V}$ . There is a unique extension of  $x$  to an augmented trace  $(f, \bar{x})$  of  $\mathfrak{S}$  such that:

1.  $\bar{x}_i(d) = x_i(d)$  for all  $d \in \mathfrak{D}$  and  $i \geq 0$ .
2.  $\bar{x}_i(a) = a_{\text{now}}[\bar{x}_i]$  for all  $a \in \mathfrak{A}$  and  $i \geq 0$ . [Well-defined because cyclic dependencies among assign variables are excluded.]
3.  $\bar{x}_0(v) = V(v)$  for all  $v \in \mathfrak{V}$ .

4. If  $v \in \mathfrak{V}$  with clock event  $(e, c)$  and  $i > 0$ , then  $\bar{x}_i(v) = v_{next}[\bar{x}_{i-1}]$  or  $\bar{x}_i(v) = \bar{x}_{i-1}(v)$  according as  $(f, \bar{x}), i-1 \models e(c)$  or  $(f, \bar{x}), i-1 \not\models e(c)$ .

We refer to  $(f, \bar{x})$  as the extension of  $x$  to  $\mathfrak{G}$  that is *coherent* with  $M$  and  $V$ .

Let  $y$  be a trace of  $\mathfrak{G}$ , and let  $f$  be a valuation of  $\mathfrak{C}$ . Let  $x$  be the trace of  $\mathfrak{D}$  obtained by restriction of  $y$ , and let  $V$  be the restriction of  $y_0$  to  $\mathfrak{V}$ . We say that  $(f, y)$  is *coherent* with  $M$  provided  $(f, y)$  is the extension of  $x$  to  $\mathfrak{G}$  that is coherent with  $M$  and  $V$ . When the CBV module is understood, we say simply that  $(f, y)$  is *coherent*.

A *CBV context* for  $M$  is a tuple

$$((e, c), \lambda, (\alpha, A), \sigma),$$

where:

- $(e, c)$  is a CBV clock event for  $M$ .
- $\lambda$  is a partial valuation of  $\mathfrak{L}$ .
- $\alpha$  is a boolean expression with support in  $\mathfrak{G} \cup \text{dom}(\lambda)$ , and  $A$  is a CBV statement.  $(\alpha, A)$  is called the *abort subcontext*.
- $\sigma$  is a boolean expression with support in  $\mathfrak{G} \cup \text{dom}(\lambda)$ .  $\sigma$  is called the *suspend condition*.

Let  $S$  be a CBV statement in  $M$ , let  $C = ((e, c), \lambda, (\alpha, A), \sigma)$  be a CBV context for  $M$ , let  $f$  be a valuation of  $\mathfrak{C}$ , let  $Y = (f, y)$  be a coherent augmented trace of  $\mathfrak{G}$ , and let  $i \geq 0$  be an index within the domain of  $y$  (in case  $y$  is finite). We give an inductive definition of

$$Y, i, C \models S.$$

Note the following about the rules in the inductive definition:

- We say that  $S$  is *equivalent* to another CBV statement  $T$  provided  $Y, i, C \models S$  iff  $Y, i, C \models T$  for all  $Y, i, C$ .
  - The symbols  $S', T, T', T_i$  represent CBV statements;  $\alpha', \sigma', \varepsilon$  represent boolean expressions; and  $C'$  represents a CBV context.
  - Rules 1 and 2 take precedence over the others. In other words, if  $Y, i \models e(c)$  and either  $\alpha[\lambda][y_i] = 1$  or  $\sigma[\lambda][y_i] = 1$ , then one of rules 1 and 2 must be applied. Otherwise, the form of  $S$  will determine which of the remaining rules can be used.
1. If  $Y, i \models e(c)$  and  $\alpha[\lambda][y_i] = 1$ , then  $Y, i, C \models S$  iff  $Y, i, C' \models A$ , where  $C' = ((e, c), \lambda, (0, 1), \sigma)$ .
  2. If  $Y, i \models e(c)$ ,  $\alpha[\lambda][y_i] = 0$ , and  $\sigma[\lambda][y_i] = 1$ , then  $Y, i, C \models S$  iff  $Y, i+1, C \models S$  or  $i$  is the largest index in the domain of  $y$  (in case  $y$  is finite).
  3. Let  $S$  have the form “ $\varepsilon$ ;”, where  $\varepsilon$  has support in  $\mathfrak{G} \cup \text{dom}(\lambda)$ . Then  $Y, i, C \models S$  iff  $\varepsilon[\lambda][y_i] = 1$ .
  4. Let  $S$  have the form “ $+(n) : T$ ”, where  $n \geq 0$ .

- If  $n = 0$ , then  $Y, i, C \models S$  iff  $Y, i, C \models T$ .
  - If  $n > 0$  and there is no integer  $j > i$  such that  $Y, j \models e(c)$ , then  $Y, i, C \models S$ .
  - Otherwise, let  $j$  be the smallest integer greater than  $i$  such that  $Y, j \models e(c)$ . Then  $Y, i, C \models S$  iff  $Y, j, C \models +(n-1) : T$ .
5. Let  $S$  have the form “ $+(n \text{ to } m) : T$ ”, where  $m \geq n \geq 0$ .
- $S$  is equivalent to “ $+(n) : +(0 \text{ to } m-n) : T$ ”.
  - If  $m = n$ , then  $Y, i, C \models S$  iff  $Y, i, C \models +(n) : T$ .
  - If  $m > n = 0$ , then  $Y, i, C \models S$  iff  $Y, i, C \models T$  and  $Y, i, C \models +(1) : +(0 \text{ to } m-1) : T$ .
6. Let  $S$  have the form “begin  $T_1 \dots T_k$  end”. There are several cases.
- If  $k = 1$  and  $T_1$  is a local variable definition, then  $Y, i, C \models S$ .
  - If  $k = 1$  and  $T_1$  is not a local variable definition, then  $Y, i, C \models S$  iff  $Y, i, C \models T_1$ .
  - Suppose  $k > 1$  and  $T_1$  is a local variable definition of the form “ $\ell \leq \xi$ ”, where  $\ell \in \mathcal{L}$  and  $\xi$  is a binary expression with support in  $\mathfrak{G} \cup \text{dom}(\lambda)$ . Let  $\lambda'$  be the restriction of  $\lambda$  to  $\text{dom}(\lambda) - \{\ell\}$ . Then  $Y, i, C \models S$  iff  $Y, i, C' \models S'$ , where  $C' = ((e, c), \lambda' \cup \{(\ell, \varepsilon[\lambda][y_i])\}, (\alpha, A), \sigma)$  and  $S'$  is the statement “begin  $T_2 \dots T_k$  end”.
  - Suppose  $k > 1$  and  $T_1$  is not a local variable definition. Then  $Y, i, C \models S$  iff  $Y, i, C \models T_1$  and  $Y, i, C \models S'$ , where  $S'$  is the statement “begin  $T_2 \dots T_k$  end”.
7. Let  $S$  have the form “if  $(\varepsilon) T$ ”.  $S$  is equivalent to “if  $(\varepsilon) T$  else 1;”.
8. Let  $S$  have the form “if  $(\varepsilon) T_1$  else  $T_2$ ”, where  $\varepsilon$  has support in  $\mathfrak{G} \cup \text{dom}(\lambda)$ . If  $\varepsilon[\lambda][y_i] = 1$ , then  $Y, i, C \models S$  iff  $Y, i, C \models T_1$ . If  $\varepsilon[\lambda][y_i] = 0$ , then  $Y, i, C \models S$  iff  $Y, i, C \models T_2$ .
9. Let  $S$  have the form “if  $+(n) : (\varepsilon) T$ ”, where  $n \geq 0$ .  $S$  is equivalent to “ $+(n) : \text{if } (\varepsilon) T$ ”.
10. Let  $S$  have the form “if  $+(n) : (\varepsilon) T_1$  else  $T_2$ ”, where  $n \geq 0$ .  $S$  is equivalent to “ $+(n) : \text{if } (\varepsilon) T_1$  else  $T_2$ ”.
11. Let  $S$  have the form “if  $+(n \text{ to } m) : (\varepsilon) T$ ”, where  $m \geq n \geq 0$  and  $\varepsilon$  has support in  $\mathfrak{G} \cup \text{dom}(\lambda)$ .
- $S$  is equivalent to “ $+(n) : \text{if } +(0 \text{ to } m-n) : (\varepsilon) T$ ”.
  - If  $m = n$ , then  $Y, i, C \models S$  iff  $Y, i, C \models \text{if } +(n) : (\varepsilon) T$ .
  - Suppose  $m > n = 0$ . If  $\varepsilon[\lambda][y_i] = 0$ , then  $Y, i, C \models S$ . If  $\varepsilon[\lambda][y_i] = 1$ , then  $Y, i, C \models S$  iff  $Y, i, C \models +(1) : \text{if } +(0 \text{ to } m-1) : (\varepsilon) T$ .
12. Let  $S$  be a CBV task call statement. Let  $x_1, \dots, x_r$  be the formal value parameters of the task, and let  $\xi_1, \dots, \xi_r$  be the actual value parameters of the task call. Similarly, let  $z_1, \dots, z_s$  be the formal reference parameters of the task, and let  $\zeta_1, \dots, \zeta_s$  be the actual reference parameters of the task call. All of the actual parameters are required to be binary expressions with support in  $\mathfrak{G} \cup \text{dom}(\lambda)$ . Let  $T$  be the begin-end statement of the task. Let  $T'$  be the CBV statement that results from  $T$  by replacing all occurrences of  $z_1, \dots, z_s$  that are not overshadowed

by local variables of the task (i.e., formal value parameters of the task or local variables defined within  $T$ ) with the corresponding expressions  $\zeta_1, \dots, \zeta_s$ . Then  $Y, i, C \models S$  iff  $Y, i, C \models S'$ , where  $S'$  is the statement

begin  $\xi_1 \leq x_1; \dots \xi_r \leq x_r; T'$  end

13. Let  $S$  have the form “if  $+(n \text{ to } *) : (\varepsilon) T$ ”, where  $n \geq 0$ .  $S$  is equivalent to the statement “ $+(n) : \$loop(n);$ ”, where the *loop* task is defined by

```

$task loop(const STAR)
begin
  if ( $\varepsilon$ )
  begin
     $T$ 
     $+(1) : \$loop(STAR + 1) ;$ 
  end
end
endtask

```

The width of the parameter STAR is not shown, but it is determined at compile time as the maximum of the following:

- One more than the maximum bit index appearing in any indexed reference to STAR in  $\varepsilon$  or  $T$ .
  - The maximum width of any binary expression in  $\varepsilon$  or  $T$  to which STAR is compared
  - 1.
14. Let  $S$  have the form “eventually  $\varepsilon;$ ”, where  $\varepsilon$  has support in  $\mathfrak{G} \cup \text{dom}(\lambda)$ . If  $\varepsilon[\lambda][y_i] = 1$ , then  $Y, i, C \models S$ . Otherwise, consider the integers  $j > i$  such that  $Y, j \models e(c)$  and either of the following conditions holds:
- $\alpha[\lambda][y_j] = 1$
  - $\sigma[\lambda][y_j] = 0$  and  $\varepsilon[\lambda][y_j] = 1$ .
- If no such  $j > i$  exists, then  $Y, i, C \not\models S$ . Otherwise, if the smallest such  $j$  satisfies the first condition, then  $Y, i, C \models S$  iff  $Y, i, C' \models A$ , where  $C' = ((e, c), \lambda, (0, 1;), \sigma)$ . Otherwise, the smallest such  $j$  satisfies the second condition and not the first condition, and in this case  $Y, i, C \models S$ .
15. Let  $S$  have the form “sample  $@(e', c') T$ ”. Then  $Y, i, C \models S$  iff  $Y, i, C' \models T$ , where  $C' = ((e', c'), \lambda, (\alpha, A), \sigma)$ .
16. Let  $S$  have the form “ $@(e', c') T$ ”. If there is no integer  $j \geq i$  such that  $Y, j \models e'(c')$ , then  $Y, i, C \models S$ . Otherwise, let  $j$  be the smallest integer greater than or equal to  $i$  such that  $Y, j \models e'(c')$ . Then  $Y, i, C \models S$  iff  $Y, j, C \models T$ .
17. Let  $S$  have the form “sync  $@(e', c') T$ ”. If there is no integer  $j \geq i$  such that  $Y, j \models e'(c')$ , then  $Y, i, C \models S$ . Otherwise, let  $j$  be the smallest integer greater than or equal to  $i$  such that  $Y, j \models e'(c')$ . Then  $Y, i, C \models S$  iff  $Y, j, C' \models T$ , where  $C' = ((e', c'), \lambda, (\alpha, A), \sigma)$ .

18. Let  $S$  have the form “when  $\alpha' A'$  abort  $T$ ”, where  $\alpha'$  has support in  $\mathfrak{G} \cup \text{dom}(\lambda)$ . Then  $Y, i, C \models S$  iff  $Y, i, C' \models T$ , where  $C' = ((e, c), \lambda, (\alpha', A'), \sigma)$ .
19. Let  $S$  have the form “when  $\sigma'$  suspend  $T$ ”, where  $\sigma'$  has support in  $\mathfrak{G} \cup \text{dom}(\lambda)$ . Then  $Y, i, C \models S$  iff  $Y, i, C' \models T$ , where  $C' = ((e, c), \lambda, (\alpha, A), \sigma')$ .

This completes the definition of  $Y, i, C \models S$ .

Finally, we define  $x \models M$  for  $x = x_0 x_1 \dots$  a trace of  $\mathfrak{D}$ . Let  $f$  be the default initial valuation of  $\mathfrak{C}$  for  $M$ ; let  $V$  be the default initial valuation of  $\mathfrak{B}$  for  $M$ ; let  $(e, c)$  be the default clock event for  $M$ ; let  $(\alpha, A)$  be the default abort subcontext for  $M$ ; let  $\sigma$  be the default suspend condition for  $M$ ; and let  $S$  be the top-level begin-end statement of  $M$ . The supports of  $\alpha$  and  $\sigma$  must be in  $\mathfrak{G}$ . Let  $C$  be the CBV context  $((e, c), \{\}, (\alpha, A), \sigma)$ . [The partial valuation of the local variables is empty in  $C$ , while the other components of  $C$  match the corresponding defaults for  $M$ .] Let  $(f, y)$  be the extension of  $x$  to  $\mathfrak{G}$  that is coherent with  $M$  and  $V$ . Let  $I$  be the set of non-negative integers  $i$  that are in the domain of  $x$  (in case  $x$  is finite) and that satisfy  $(f, y), i \models e(c)$ . Then  $x \models M$  iff  $(f, y), i, C \models S$  for each  $i \in I$ .

**Remarks:**

- CBV can be extended to allow multiple top-level begin-end statements within a module. In this case, each top-level begin-end statement can be associated with its own initial CBV context.
- CBV if-tasks have been omitted. If-tasks utilize a return mechanism and require addition of a stack of return destination statements to the CBV context. Recursion of if-tasks is not allowed, so the stack can be bounded at compile time. Alternatively, the if-task code can be in-lined in a pre-compilation step.