IEEE 200X Fast Track Change Proposal

ID: FT10B

Proposed By: Jim Lewis jim@synthworks.com
Analyzed By: Jim Lewis jim@synthworks.com
With significant input from John Ries

Status: Analyzed
Proposed: 12/03
Analyzed: 07/11

Enhancement Summary:

Make conditional signal assignment work for sequential signal and variable assignments.
Make selected signal assignment work in sequential signal and variable assignments

Revisions:

Rev4: BNF update for left-recursion in conditional signal assignment.
Revised transformation to include else
Rev3: Split into 10A n-nary expressions and 10B sequential assignments
Rev2: Initial pdf/word release. Changed conditional_expression to ternary_expression.
Removed proposed changes to conditional waveforms due to ambiguity.
Combined conditional assignments with selected assignments.
Rev1: Initial release

Related issues: 10A

Relevant LRM sections:

8.4, 8.4.1, 8.5, 9.5, 9.5.1, 9.5.2

Enhancement Detail:

The target of this enhancement is to simplify code of the form:

```
if (FP = '1') then
  NextState <= FLASH ;
else
  NextState <= IDLE ;
end if ;
```

By allowing conditional signal assignments in a process, the above code can be rewritten as:

```
NextState <= FLASH when (FP = '1') else IDLE ;
```

Analysis:

Choices:

1. Don't do this change in consideration of n-nary expressions.
2. Do this change to give consistency between concurrent and sequential VHDL

Approach Overview:

In section 8, redefine sequential signal assignments to allow for simple signal assignment, conditional signal assignment, and selected signal assignment. Much of the current section 8.4 will be moved to 8.4.1 simple signal assignment. This section is needed since other signal assignments are transformed to a simple signal assignment. Move the current 8.4.1 to 8.4.1.1. Add section 8.4.2 conditional signal assignment. Much of the text will come from the current section on conditional signal assignments. Changes were made to the BNF to distinguish it from simple signal assignment. Add section 8.4.3 selected signal assignment.

Move syntax for guarded to 8.4.1, but also note that it is an error to use it in the sequential form of signal assignments.

Redefine variable assignments to allow for simple variable assignment and selected variable assignment. Conditional variable assignment can be handled by ternary expressions (see further study). Define transformations as done for signals.

In section 9, redefine concurrent signal assignments to allow for simple signal assignment, conditional signal assignment, and selected signal assignment. Text will describe the transformation from concurrent signal assignment to the equivalent process, but will no longer need transformations for conditional and selected signal assignments as they are covered in 8.4.

Further Study

Should we change the name signal_assignment_statement to sequential_signal_assignment_statement (to distinguish it from concurrent_signal_assignment_statement). It would make some of the text easier to read.

LRM Changes

Change Clause 8.4 to:

8.4 Signal assignment statement

A signal assignment statement modifies the projected output waveforms contained in the drivers of one or more signals (see 12.6.1).

```
signal_assignment_statement ::=
      [label :] simple_signal_assignment
    | [label :] conditional_signal_assignment
    | [label :] selected_signal_assignment
```

Move remainder of clause 8.4 to 8.4.1 (unchanged except as noted)

8.4 Simple signal assignment

```
simple_signal_assignment ::=
  target <= options waveform ;

options ::= [guarded][delay_mechanism]
```

```
delay_mechanism ::=
    transport
  | [ reject time_expression ] inertial

target ::=
  name
  | aggregate

waveform ::=
  waveform_element { , waveform_element }
  | unaffected
```

Remove restrictions from unaffected in 8.4 (old) / 8.4.1(new)

Remove (Bottom p121 in 1076-2002)

~~It is an error if the reserved word unaffected appears as a waveform in a (sequential) signal assignment statement.~~

Notes:

~~1-The reserved word unaffected must only appear as a waveform in concurrent signal assignment statements (see 9.5.1).~~

Add (same place):

A simple signal assignment of the form

```
target <= [delay_mechanism] unaffected ;
```

shall have the same effect as replacing the given assignment with a null statement (not a null assignment).

Add (same place):

It is an error if the reserved word guarded appears as an option in the sequential form of the signal assignment statement.

Notes:

1-The reserved word guarded must only appear as an option in concurrent signal assignment statements (see 9.5).

Move current clause 8.4.1 to 8.4.1.1 and change the BNF as shown below:

8.4.1.1 Updating a projected output waveform

The effect of execution of a signal assignment statement is defined in terms of its effect upon the projected output waveforms (see 12.6.1) representing the current and future values of drivers of signals.

```
waveform_element ::=
  expression [ after time_expression ]
  | null [ after time_expression ]
```

Add Clause 8.4.2 Conditional Signal Assignments (see editing notes that follow)

8.4.2 Conditional signal assignments

The conditional signal assignment statement is a short hand notation that can be replaced by an if statement and a set of simple signal assignments.

```
conditional_signal_assignment ::=
    target <= options conditional_waveforms ;

conditional_waveforms ::=
   { waveform when condition else }
   waveform when condition
   [ else waveform ]
```

The options for a conditional signal assignment statement are discussed in 8.4.1.

Conditional signal assignment is defined in terms of the following transformation. If the conditional signal assignment is of the form

```
target <= options
  waveform1  when condition1 else
  waveform2  when condition2 else
    •
    •
    •
  waveformN-2 when conditionN-2 else
  waveformN-1 when conditionN-1 else
  waveformN ;
```

then the equivalent if statement and simple signal assignments are of the form

```
if condition1 then
  target <= options waveform1 ;  -- simple signal assignment
elsif condition2 then
  target <= options waveform2 ;
    •
    •
    •
elsif conditionN-2 then
  target <= options waveformN-2 ;
elsif conditionN-1 then
  target <= options waveformN-1 ;
else
  target <= options waveformN ;
end if ;
```

If the conditional waveforms do not contain a final else clause, then the transformation will contain an else statement of the form:

```
else
   target <= options unaffected ;
```

If a label appears on the conditional signal assignment, then the same label appears on the corresponding if statement. If the delay mechanism option appears in the conditional signal assignment, then the same delay mechanism appears in every simple signal assignment statement of the transformed assignment.

The characteristics of the waveforms and conditions in the conditional assignment statement must be such that the if statement in the transformed code is a legal statement.

```
Example:
S <= unaffected when Input_pin = S'DrivingValue else
Input_pin after Buffer_Delay;
```

NOTE—The wave transform of a waveform of the form **unaffected** is the null statement, not the null transaction.

Editing notes for 8.4.2

Changed transformation to include an else. Added text to explain lack of else is the same as an assignment to unaffected.

The text on waveform transforms was removed because the equivalent assignment is a simple signal assignment and is already defined in this section so no further transformations are required.

Unaffected was removed since its transformation is now explained in simple signal assignment.

Add Clause 8.4.3 Selected Signal Assignments

8.4.3 Selected signal assignments

The selected signal assignment statement is a short hand notation that can be replaced by a case statement and a set of simple signal assignments.

```
selected_signal_assignment ::=
  with expression select
    target <= options selected_waveforms ;

selected_waveforms ::=
    { waveform when choices , }
    waveform when choices
```

The options for a selected signal assignment statement are discussed in 8.4.1.

Selected signal assignment is defined in terms of the following transformation. If the selected signal assignment is of the form

```
with expression select
  target <= options waveform1 when choice_list1 ,
                    waveform2 when choice_list2 ,
                       •
```

```
                    .
                    .
                waveformN-1 when choice_listN-1,
                waveformN when choice_listN ;
```

then the equivalent case statement and simple signal assignments are of the form

```
case expression is
  when choice_list1 =>
    target <= options waveform1 ;  -- simple signal assignment

  when choice_list2 =>
    target <= options waveform2 ;
        .
        .
        .
  when choice_listN-1 =>
    target <= options waveformN-1 ;
  when choice_listN =>
    target <= options waveformN ;
end case ;
```

If a label appears on the selected signal assignment, then the same label appears on the corresponding case statement. If the delay mechanism option appears in the selected signal assignment, then the same delay mechanism appears in every simple signal assignment statement of the transformed assignment.

The characteristics of the select expression, the waveforms, and the choices in the selected assignment statement must be such that the case statement in the transformed code is a legal statement.

Change Clause 8.5 to:

8.5 Variable assignment statement

A variable assignment statement replaces the current value of a variable with a new value specified by an expression. The named variable and the right-hand side expression must be of the same type.

```
variable_assignment_statement ::=
      [label :] simple_variable_assignment
    | [label :] conditional_variable_assignment
    | [label :] selected_variable_assignment
```

Move rest of clause 8.5 to 8.5.1 and change the term variable assignment to simple variable assignment.

8.5.1 Simple variable assignment statement

```
Simple_variable_assignment ::=
```

```
[ label : ] target := expression ;
```

Move clause 8.5.1 to 8.5.1.1

Add section 8.5.2

The conditional variable assignment statement is a short hand notation that can be replaced by an if statement and a set of simple variable assignments.

```
conditional_variable_assignment ::=
    target := conditional_variable_expression ;

conditional_variable_expression ::=
   expression when condition
   { else expression when condition }
   [ else expression]
```

Conditional variable assignment is defined in terms of the following transformation. If the conditional variable assignment is of the form

```
target :=
  expression1    when condition1 else
  expression2    when condition2 else
    •
    •
    •
  expressionN-1  when conditionN-1 else
  expressionN    when conditionN;
```

then the equivalent if statement and simple variable assignments are of the form

```
if condition1 then
  target := expression1 ;  -- simple variable assignment
elsif condition2 then
  target := expression2 ;
    •
    •
    •
elsif conditionN-1 then
  target := expressionN-1 ;
elsif conditionN then
  target := expressionN ;
end if ;
```

If a label appears on the conditional variable assignment, then the same label appears on the corresponding if statement. The characteristics of the expressions and conditions in the conditional assignment statement must be such that the if statement in the transformed code is a legal statement.

Add section 8.5.3

The selected variable assignment statement is a short hand notation that can be replaced by a case statement and a set of simple variable assignments.

```
selected_variable_assignment ::=
  with expression select
    target <= options selected_expression ;

selected_expression ::=
    { expression when choices , }
    expression when choices
```

Selected variable assignment is defined in terms of the following transformation. If the selected variable assignment is of the form

```
with expression select
  target := waveform1 when choice_list1 ,
            waveform2 when choice_list2 ,
                 •
                 •
                 •
            waveformN-1 when choice_listN-1,
            waveformN when choice_listN ;
```

then the equivalent case statement and simple variable assignments are of the form

```
case expression is
  when choice_list1 =>
    target := expression1 ;  -- simple variable assignment

  when choice_list2 =>
    target := expression2 ;
       •
       •
       •
  when choice_listN-1 =>
    target := expressionN-1 ;
  when choice_listN =>
    target := expressionN ;
end case ;
```

If a label appears on the selected variable assignment, then the same label appears on the corresponding case statement.

The characteristics of the select expression, the waveforms, and the choices in the selected assignment statement must be such that the case statement in the transformed code is a legal statement.

Changes to Clause 9.5:

9.5 Concurrent signal assignment statements

The concurrent signal assignment statement is a short hand notation for an equivalent process statement with the corresponding sequential form of the signal assignment statement.

```
Concurrent_signal_assignment_statement ::=
      [label :] [ postponed ] simple_signal_assignment
    | [label :] [ postponed ] conditional_signal_assignment
    | [label :] [ postponed ] selected_signal_assignment
```

Each concurrent signal assignment has an identically formatted sequential signal assignment. The concurrent forms will be defined by a transformation to an equivalent process that uses the corresponding sequential signal assignment.

The primary difference in syntax between the concurrent and sequential forms of signal assignment is the support of the **guarded** option. The option **guarded** specifies that the signal assignment statement is executed when a signal GUARD changes from FALSE to TRUE, or when that signal has been TRUE and an event occurs on one of the signal assignment statement's inputs. (The signal GUARD must be either one of the implicitly declared GUARD signals associated with block statements that have guard expressions, or it must be an explicitly declared signal of type Boolean that is visible at the point of the concurrent signal assignment statement.) The delay mechanism option specifies the pulse rejection characteristics of the signal assignment statement (see 8.4.1).

If the target of a concurrent signal assignment is a name that denotes a guarded signal (see 4.3.1.2), or if it is in the form of an aggregate and the expression in each element association of the aggregate is a static signal name denoting a guarded signal, then the target is said to be a *guarded target*. If the target of a concurrent signal assignment is a name that denotes a signal that is not a guarded signal, or if it is in the form of an aggregate and the expression in each element association of the aggregate is a static signal name denoting a signal that is not a guarded signal, then the target is said to be an *unguarded target*. It is an error if the target of a concurrent signal assignment is neither a guarded target nor an unguarded target.

For any concurrent signal assignment statement, there is an equivalent process statement with the same meaning. The process statement equivalent to a concurrent signal assignment statement whose target is a signal name is constructed as follows:

a) If a label appears on the concurrent signal assignment statement, then the same label appears on the process statement.

b) The equivalent process statement is a postponed process if and only if the concurrent signal assignment statement includes the reserved word **postponed**.

c) If the delay mechanism option appears in the concurrent signal assignment, then the same delay mechanism appears on the corresponding sequential signal assignment statement in the process statement

d) The statement part of the equivalent process statement consists of a statement transform [described in item e)].

e) If the option **guarded** appears in the concurrent signal assignment statement, then the concurrent signal assignment is called a *guarded assignment*. If the concurrent signal assignment statement is a guarded assignment, and if the target of the concurrent signal assignment is a guarded target, then the statement transform is as follows:

```
if GUARD then
  signal_transform
else
  disconnection_statements
end if ;
```

Otherwise, if the concurrent signal assignment statement is a guarded assignment, but if the target of the concurrent signal assignment is *not* a guarded target, then the statement transform is as follows:

```
if GUARD then
  signal_transform
end if ;
```

Finally, if the concurrent signal assignment statement is *not* a guarded assignment, and if the target of the concurrent signal assignment is *not* a guarded target, then the statement transform is as follows:

```
signal_transform
```

It is an error if a concurrent signal assignment is not a guarded assignment and the target of the concurrent signal assignment is a guarded target.

A *signal transform* is the replacement of the concurrent signal assignment statement with its corresponding sequential signal assignment statement.

f) If the concurrent signal assignment statement is a guarded assignment, or if any expression (other than a time expression) within the concurrent signal assignment statement references a signal, then the process statement contains a final wait statement with an explicit sensitivity clause. The sensitivity clause is constructed by taking the union of the sets constructed by applying the rule of 8.1 to each of the aforementioned expressions. Furthermore, if the concurrent signal assignment statement is a guarded assignment, then the sensitivity clause also contains the simple name GUARD. (The signals identified by these names are called the inputs of the signal assignment statement.) Otherwise, the process statement contains a final wait statement that has no explicit sensitivity clause, condition clause, or timeout clause.

Under certain conditions (see above) the equivalent process statement may contain a sequence of disconnection statements. A *disconnection statement* is a sequential signal assignment statement that assigns a null transaction to its target. If a sequence of disconnection statements is present in the equivalent process statement, the sequence consists of one sequential signal assignment for each scalar subelement of the target of the concurrent signal assignment statement. For each such sequential signal assignment, the target of the assignment is the corresponding scalar subelement of the target of the concurrent signal assignment, and the waveform of the assignment is a null waveform element whose time expression is given by the applicable disconnection specification (see 5.3).

If the target of a concurrent signal assignment statement is in the form of an aggregate, then the same transformation applies. Such a target must contain only locally static signal names; moreover, it is an error if any signal is identified by more than one signal name. It is an error if a null waveform element appears in a waveform of a concurrent signal assignment statement. Execution of a concurrent signal assignment statement is equivalent to execution of the equivalent process statement.

NOTES

1—A concurrent signal assignment statement whose waveforms and target contain only static expressions is equivalent to a process statement whose final wait statement has no explicit sensitivity clause, so it will execute once through at the beginning of simulation and then suspend permanently.

2—A concurrent signal assignment statement whose waveforms are all the reserved word **unaffected** has no drivers for the target, since every waveform in the concurrent signal assignment statement is transformed to the statement

null;

in the equivalent process statement (see 8.4.2).

Remove Clause 9.5.1 and 9.5.2

Most of the content was incorporated into the new clauses 8.4.2 and 8.4.3