July 30, 2003

Mr Stephen Bailey
Chairman, IEEE 1076 Working Group
6664 Cherokee Court
Niwot, CO 80503

Dear Mr. Bailey,

Cadence Design Systems hereby grants permission to the Institute of Electrical and Electronics Engineers (IEEE) to reprint and/or modify material attached to this letter as Appendix A.

It is understood that if the material is modified by IEEE, IEEE will be the sole owner of the copyright to the resulting document, namely IEEE Std 1076 – IEEE Standard VHDL Reference Manual, being developed by the IEEE 1076 Working Group, subject to Cadence's underlying right to the unmodified material.

If requested acknowledgement of the original work will be placed in the front matter of the new work.

Very truly yours,

Rahul Razdan,

Vice President
Design & Verification R&D
Cadence design Systems, Inc.

APPENDIX A

1.ncmirror documentation

The following paragraphs describe the functionality provided by the ncmirror package.

ncmirror is a set of foreign procedures that you can call from a VHDL architecture to access VHDL signals at any level of hierarchy in the design. The procedures are intended to be used primarily in a VHDL testbench so that you have visibility of signals at lower levels in the hierarchy. However, it is possible to reference signals at any level, higher or lower.

Hierarchical paths to signals are specified as strings. A "." is used as the hierarchy separator. Either full hierarchical paths or relative paths may be used. Full hierarchical paths begin with a ":" (eg. ":inst1.dut.data_bus"), while relative paths begin with instance labels (eg. "dut.data_bus"). A "^" in the hierarchy string means moving one level up in the hierarchy (eg. "^.^.inst2.signal1").

The ncmirror package comprises four procedures. These are:

`a.nc_mirror`

Lets you trace the value of any VHDL signal lying anywhere in the design from within the VHDL architecture. This procedure can also be used to mirror the value of a source signal onto a destination signal.

`b.nc_deposit`

Lets you deposit a value on a VHDL signal lying anywhere in the design.

`c.nc_force`

Lets you force a value on a VHDL signal lying anywhere in the design.

`d.nc_release`

Lets you release a force on a VHDL signal.

The following sections describe each of these routines in detail.

a)nc_mirror

The `nc_mirror` procedure may be used to trace the value of any VHDL lying anywhere in the design from within a VHDL architecture. This procedure establishes a link so that the value of a signal that is not in the current architecture can be read.

Syntax:

```
procedure nc_mirror (
                      destination : IN string := "",
                      source      : IN string,
                      verbose     : IN boolean := false
                     );
```

Parameters:

Destination: The full hierarchical path of a VHDL signal to which the value of the source signal is to be mirrored. This parameter is optional. If you do not specify a destination signal, the new value of the source will be printed on the standard output whenever it's value changes.

Source: The hierarchical path to the signal that is to be mirrored.

Verbose: Optional parameter used for status reporting. If true, a message is displayed stating that the source object value has been mirrored onto the destination object.

The procedure performs syntax and error checks to ensure that the two signals are of a compatible type, that the number of elements match, that specified ranges are valid, and so on. This foreign procedure does not create a driver for the destination signal. Instead, it directly sets the effective value of the destination whenever the source changes.

Examples:

In the following examples, a VHDL signal called `dest` has been declared in the top-level VHDL architecture. The `nc_mirror` routine is used to mirror the value of a signal called `src`, which is in a lower-level design unit called `I1`. In the first example, full hierarchical paths are used, and the verbose parameter defaults to false.

```
nc_mirror(destination => ":dest",
          source      => ":I1:src");
```

In the next example, relative paths are used.

```
nc_mirror(destination => "dest",
          source      => "I1:src",
          verbose     => true);
```

In the following example, the destination is not specified, so the value of the source signal is mirrored to the standard output.

```
nc_mirror(source => ":I1:src");
```

b)nc_deposit

The `nc_deposit` procedure deposits a value on a VHDL signal.

The `nc_deposit` procedure does not create a driver for the VHDL signal. The effective value of the specified object is set, and behaviors that are sensitive to value changes on the object run when simulation resumes, just as if the value change was caused by VHDL code.

Syntax:

```
procedure nc_deposit (
            target     : IN string,
            value      : IN string,
            delay      : IN time,
            delay_mode : IN nc_delay_type :=
NC_TRANSPORT,
            absolute   : IN boolean := false,
            verbose    : IN boolean := false
           );
```

Parameters:

Target: Full hierarchical path to a VHDL the target signal.
Value: The value to be deposited on the target signal.
Delay: The simulation delay associated with the deposit. The default is to treat this as a relative delay, unless overridden by setting the absolute parameter to true. In the absolute delay mode, if the time specified by the parameter is less than the current simulation time, the deposit is ignored, and an error message is generated.
Delay_mode: The delay mechanism to be used. You can specify NC_INERTIAL or NC_TRANSPORT, which map to inertial and transport delays respectively. The default is NC_TRANSPORT.
Absolute: Specifies that the delay is to be treated as an absolute time at which the deposit should happen.
Verbose: Optional parameter used for status reporting. If true, a message is displayed stating that the deposit has been applied to the target signal.

Examples:

The following examples deposit the specified value on signal `:I1:src`.

```
nc_deposit (source     => ":I1:src",
            value      => "11001110",
            delay      => 10 ns);

nc_deposit (source     => ":I1:src",
            value      => "11001110",
            delay      => 10 ns,
            delay_mode => NC_INERTIAL,
            absolute   => true,
            verbose    => true);

nc_deposit (":I1:src", "11001110", 10 ns);
```

c)nc_force

The `nc_force` procedure may be used to force a value onto a VHDL signal.

The new value takes effect immediately and propagates throughout the hierarchy. The transaction is processed in the simulation cycle that follows the call. The simulator schedules a new delta cycle, if necessary, to process the transaction.

The forced value can be released by calling the `nc_release` procedure.

Syntax:

```
procedure nc_force (
                    target  : IN string,
                    value   : IN string,
                    verbose : IN boolean := false
                   );
```

Parameters:

Target: The full hierarchical path to a VHDL signal. This parameter is required.
Value: Value to be forced on the source object. This parameter is required.

Verbose: Optional parameter used for status reporting. If true, a message is displayed stating that the source signal was forced to the specified value.

Examples:

In the following example, a force is set on signal `:I1:src`. The `verbose` parameter displays a message reporting that the force has been set.

```
nc_force (source  => ":I1:src",
          value   => "11010101",
          verbose => true);
```

In the following example, the `verbose` parameter is omitted.

```
nc_force (":I1:src", "11010101");
```

d)nc_release

The `nc_release` procedure is used to release a force on an object. The forced value is released, and the value of the object immediately returns to the value that the object would have had if the force hadn't been blocking transactions. The `keepvalue` parameter can be used to specify that the forced value is to be retained.

Syntax:

```
procedure nc_release (
                      target    : IN string,
                      keepvalue : IN boolean := false,
                      verbose   : IN boolean := false
                     );
```

Parameters:

Target: A full hierarchical path of a VHDL signal or Verilog wire. This parameter is required.

Keepvalue: Release the forced object, but retain the forced value. The default is to revert to the value that the signal should have had without the forece.

Verbose: Optional parameter used for status reporting. If specified, a message is displayed stating that the force on the source signal has been released.

Examples:

In the following example, a previously set force on signal `:I1:src` is released. The `verbose` parameter displays a message reporting that the force has been released.

```
nc_release (
            target    => ":I1:src",
            keepvalue => false,
            verbose   => true);
```

The following example releases the force, but retains the forced value. The keepvalue and `verbose` parameters are omitted.

```
nc_release ( ":I1:src" );
```

2.Contents of the ncmirror package

```
library std;
use std.standard.all;

package NCUTILITIES is

  type nc_delay_type is ( NC_TRANSPORT, NC_INERTIAL );

  procedure nc_mirror (
    destination   : IN string := "";
    source        : IN string;
    verbose       : IN boolean := false
  );

  attribute FOREIGN of nc_mirror  : procedure is "NC_internal";


  procedure nc_deposit (
    target        : IN string;
    value         : IN string;
    delay         : IN time;
    delay_mode    : IN nc_delay_type := NC_TRANSPORT;
    absolute      : IN boolean := false;
    verbose       : IN boolean := false
  );

  attribute FOREIGN of nc_deposit : procedure is "NC_internal";


  procedure nc_force (
    target        : IN string;
    value         : IN string;
    verbose       : IN boolean := false
  );

  attribute FOREIGN of nc_force : procedure is "NC_internal";


  procedure nc_release (
    target        : IN string;
    keepvalue     : IN boolean := false;
    verbose       : IN boolean := false
  );

  attribute FOREIGN of nc_release : procedure is "NC_internal";


end NCUTILITIES;
```