

# The VHDL Interface to WAVES

James P. Hanna  
Design and Diagnostics Branch  
Microelectronics Reliability Division  
Rome Laboratory  
Griffiss AFB, NY 13441-5700

## Abstract

The Waveform and Vector Exchange Specification (WAVES, IEEE standard 1029.1) provides powerful support for concurrent engineering principles as the Industry standard representation for digital stimulus and response for both design and test communities. Although WAVES is a subset of VHDL, no special support for using WAVES in a VHDL environment is defined within the language. The WAVES-VHDL Interface package has been developed to provide a software interface to support the use of WAVES in a VHDL environment.

## Summary

In December of 1991 the Waveform and Vector Exchange Specification (WAVES) was approved by the balloting community of the IEEE as the Industry standard representation and exchange format for digital stimulus and response data for both design and test activities. WAVES will provide a powerful support mechanism for concurrent engineering practices by allowing this information to be freely exchanged between multiple simulation and test platforms. Although WAVES is defined as a subset of VHDL, there are no special provisions in the language which facilitate its use in a VHDL modeling environment. Therefore, developing WAVES testbenches for VHDL models is not as straight-forward as one might assume. Providing a seamless interface between WAVES data sets and VHDL models (or any specific application) is beyond the scope of developing an interchange format; the intent of WAVES.

Developing a WAVES testbench for a VHDL model requires an understanding of the implementation of the WAVES port list. The WAVES port list provides signals that are used to drive the model and signals that describe the expected response of the model. The interface to the WAVES data set is through the WAVES port list. These signals are provided in terms of the WAVES concept of a "logic value" which describes events on a given signal as sets of state, strength, direction, and relevance. All of this event information may not be applicable to the VHDL model, some may only have meaning in the context of physical hardware. In order to use WAVES in a VHDL modeling environment, the WAVES logic values must be translated into the simulator codes that can be used by the model. No mechanism for achieving this translation is provided by the WAVES language definition since this is a VHDL-specific implementation issue.

The WAVES Language Reference Manual (LRM) does not define the data structure implementation of the WAVES port list because particular environments (various testers, various simulators, etc.) may place different constraints and requirements on the underlying implementation of the WAVES port list. The WAVES standard is not dependent on any particular environment or usage scenario. This provides for maximum flexibility when interfacing WAVES to various applications and environments. This flexibility does not come without a price. This poses concern when using WAVES in a VHDL modeling environment since different VHDL simulators are free to implement the WAVES port list in very different ways.

The WAVES-VHDL Interface (WVI) was developed to define and provide a software interface to ease the use of WAVES in the VHDL environment. In particular, the WVI addresses two areas of concern: connecting the WAVES port list signals to the VHDL model and testbench portability. If WAVES testbenches are to be portable among a variety of VHDL simulators without regard to a particular implementation of the WAVES standard, some "standard practice" interface to the WAVES port list must be defined. The WVI provides a nomenclature and a set of calling conventions for an implementation independent functional interface to the WAVES port list. These functional interfaces also provide the mechanisms needed to support the WAVES port list to VHDL model port connections in a flexible and easy to use manner. The functions and procedures included in the WVI package include support for comparing the actual response of the VHDL model with the expected response given in the WAVES dataset.

The WVI uses the VHDL packaging concepts to separate the functional interface from the functional implementation details. The WVI package specification serves as the “standard practice” that defines these interfaces, while the WAVES implementation dependent details remain “hidden” in the package body. This approach allows testbenches written based on the WVI to be portable among multiple WAVES-VHDL environments. VHDL environments with differing WAVES implementations will need to provide a WVI package body based on the local WAVES implementation. Since the implementation of the WVI package body is only dependent on the implementation of the WAVES port list, modifying these functions and procedures is a relatively trivial task.

The WVI also provides the needed WAVES logic value to VHDL simulator code translation mechanisms. The WVI “uses” a user supplied VHDL package which defines the mappings between the WAVES logic values and the simulator codes that the VHDL model expects and generates. This user supplied package allows the WVI to support VHDL models with arbitrary signal types.

# **WAVES/VHDL Interface**

**James P. Hanna  
Microelectronics Reliability Division  
Rome Laboratory  
525 Brooks Road  
Griffiss AFB, NY 13441-4505  
hanna@rl.af.mil**

Rome Laboratory, USAF

## **Objectives**

- **VHDL Test Benches Utilizing WAVES Must Be Portable**
  - **Independent of WAVES Implementation**
  - **Independent of WAVES System Package**
- **VHDL Users of WAVES Should Not Need To Know the "Nuts & Bolts" of the WAVES Implementation**
  - **Ease Insertion of WAVES in VHDL Environment**
  - **Assist in Automatic Test Bench Generation**

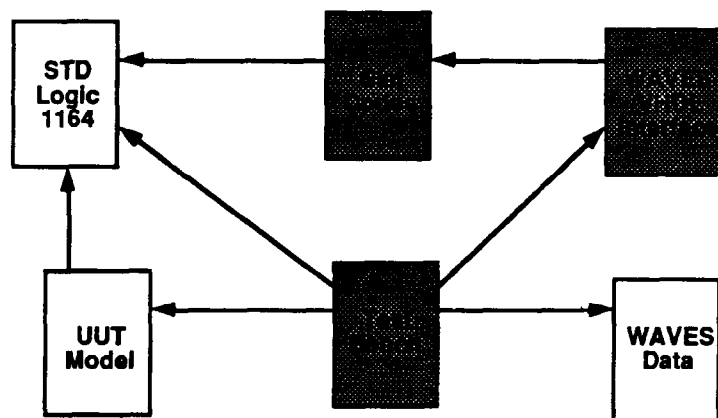
Rome Laboratory, USAF

## Approach

- **Specify an Explicit Mapping Between Simulator Codes and Logic Values**
- **Provide Procedural and Functional Interfaces to the WAVES Port List**
  - **Provide Stimulus to the Model**
  - **Compare the Model Output to Expected Results**
- **Use WAVES Test\_Pins and Pinset as WPL Selectors**
  - **Allow Single Pin Wiring**
  - **Allow Bus Wiring**

Rome Laboratory, USAF

## Topology



Rome Laboratory, USAF

## Package Simulator\_Codes

- Provides Explicit Mapping Between Simulator Codes and Logic Values
- Required Declarations
  - Sim\_Code : Enumerates Legal Simulator Codes
  - TRANSLATE : Function (Logic\_Value to Sim\_Code)
  - IS\_EQUAL : Function (Compare Logic\_Value to Sim\_Code)

Rome Laboratory, USAF

## Package Simulator\_Codes

```
use WORK.STD_Logic_1164.all;
use WORK.MM54HC181_WAVES_logic.all;
package Simulator_Codes is

  --
  -- Declare Sim_code.
  --
  subtype Sim_code is Std_ulogic;

  --
  -- Declare Sim_code_vector.
  --
  subtype Sim_code_vector is Std_ulogic_vector;
```

Rome Laboratory, USAF

## Package Simulator\_Codes

```
--  
-- Declare Sim_code_array.  
-- UNKNOWN DONT_CARE DRIVE_0 DRIVE_1 SENSE_0 SENSE_1  
--  
type Sim_code_array is array ( Logic_value'left to Logic_value'right )  
  of Sim_code;  
--  
-- Declare Boolean_matrix.  
--  
type Boolean_matrix is array ( Sim_code'left to Sim_code'right,  
  Logic_value'left to Logic_value'right )  
  of Boolean;  
--  
-- Define TRANSLATE table.  
--  
Constant TRANSLATE : Sim_code_array := ( 'U','-', '0', '1', 'L', 'H' );
```

Nome Laboratory, USAF

## Package Simulator\_Codes

```
--  
-- Define IS_EQUAL table.  
--  
constant IS_EQUAL : Boolean_matrix :=  
--  
-- UNKNOWN DONT_CARE DRIVE_0 DRIVE_1 SENSE_0 SENSE_1  
--  
{ { TRUE, TRUE, FALSE, FALSE, FALSE, FALSE }, -- 'U'  
  { TRUE, TRUE, FALSE, FALSE, FALSE, FALSE }, -- 'X'  
  { FALSE, TRUE, TRUE, FALSE, TRUE, FALSE }, -- '0'  
  { FALSE, TRUE, FALSE, TRUE, FALSE, TRUE }, -- '1'  
  { FALSE, TRUE, FALSE, FALSE, FALSE, FALSE }, -- 'Z'  
  { TRUE, TRUE, FALSE, FALSE, FALSE, FALSE }, -- 'W'  
  { FALSE, TRUE, TRUE, FALSE, TRUE, FALSE }, -- 'L'  
  { FALSE, TRUE, FALSE, TRUE, FALSE, TRUE }, -- 'H'  
  { TRUE, TRUE, TRUE, TRUE, TRUE, TRUE } }; -- '-'  
end Simulator_Codes;
```

Nome Laboratory, USAF

## Interface Package

- **Exported Types**
  - **Logic\_Vector** (Logic\_Value Vector)
- **Procedures and Functions (15)**
  - **Sim\_Code\_Of** - **Sim\_Code\_Vector\_Of**
  - **Logic\_Value\_Of** - **Logic\_Vector\_Of**
  - **Compare** (4 overloaded functions)  
(2 overloaded procedures)
  - **Match\_Assign** (2 overloaded procedures)
  - **Get\_Tag** (3 overloaded procedures)

Home Laboratory, USAF

## Interface Package

```
use WORK.MM54HC181_WAVES_DUT.all;
use WORK.MM54HC181_WAVES_Logic.all;
use WORK.Waveform_Generator.all;
use WORK.WAVES_Objects.all;
use WORK.Simulator_Codes.all;
package WAVES_VHDL_Interface is

--
-- Declare a type for vectors of user defined type 'Logic_value'.
--
type Logic_vector is array ( natural range <> ) of Logic_value;
```

Home Laboratory, USAF

## Interface Package

```
-- This function returns the Sim_code that corresponds to the WAVES
-- Logic_value of the given pin in the waves port list.
--
function sim_code_Of( signal WPL: WAVES_port_list;
                    PIN : Test_pins ) return Sim_code;
```

Rome Laboratory, USAF

## Interface Package

```
--
-- This function returns a Sim_code_vector that corresponds - in length
-- and order - to the WAVES Logic values in the WAVES port list elements
-- that are members of the given pin set.
--
function sim_code_Vector_Of( signal WPL          : WAVES_port_list;
                          PIN_SET      : Pinset := ALL_PINS;
                          ASCENDING    : Boolean := TRUE )
    return Sim_code_vector;
```

Rome Laboratory, USAF

## Interface Package

- This function returns the WAVES Logic\_value of the given pin in the
- WAVES port list

```
-  
function Logic_Value_Of( signal WPL      : WAVES_port_list;  
                        PIN              : Pinset := ALL_PINS )  
    return Logic_value;
```

Rome Laboratory, USAF

## Interface Package

- This function returns a Logic\_vector that corresponds - in length
- and order - to WAVES logic values in the WAVES port list elements
- that are members of the given pin set.

```
-  
function Logic_Vector_Of( signal WPL      : WAVES_port_list;  
                        PIN_SET         : Pinset := ALL_PINS;  
                        ASCENDING       : Boolean := TRUE )  
    return Logic_vector;
```

Rome Laboratory, USAF

## Interface Package

- 
- This function compares a `Sim_code` to the corresponding element
- (given by `PIN`) of the WAVES port list.
- 

```
function Compare( signal WPL : WAVES_port_list;  
                 CODE : Sim_code;  
                 PIN   : Test_pins )  
    return Boolean;
```

Rome Laboratory, USAF

## Interface Package

- 
- This function compares the `Sim_codes` in the `Sim_code_vector` to the
- corresponding elements (given by `PIN_SET`) of the WAVES port list.
- 

```
function Compare( signal WPL      : WAVES_port_list;  
                 VECTOR : Sim_code_vector;  
                 PIN_SET : Pinset := ALL_PINS )  
    return Boolean;
```

Rome Laboratory, USAF

## Interface Package

```
--  
-- This function compares a Sim_code to an element (given by PIN) of  
-- the WAVES port list and returns a WAVES_match_list indicating the  
-- result.  
--  
function Compare( signal WPL : WAVES_port_list;  
                 CODE : Sim_code;  
                 PIN : Test_pins )  
    return WAVES_match_list;
```

Rome Laboratory, USAF

## Interface Package

```
--  
-- This function compares each Sim_code in the Sim_code_vector to the  
-- corresponding elements (given by PIN_SET) of the given WAVES port  
-- list and returns a WAVES_match_list indicating the results.  
--  
function Compare( signal WPL : WAVES_port_list;  
                 VECTOR : Sim_code_vector;  
                 PIN_SET : Pinset := ALL_PINS )  
    return WAVES_match_list;
```

Rome Laboratory, USAF

## Interface Package

```
--  
-- This procedure assigns the element (given by PIN) of the WAVES  
-- match list to the values specified in VALUE, also given by PIN.  
--  
procedure Match_Assign( signal WML   : inout WAVES_match_list;  
                        VALUE : in   WAVES_match_list;  
                        PIN    : in   Test_Pins );
```

Home Laboratory, USAF

## Interface Package

```
--  
-- This procedure assigns all of the elements (given by PIN_SET) of  
-- the WAVES match list to the values specified in VALUE that are  
-- also given by PIN_SET.  
--  
procedure Match_Assign( signal WML   : inout WAVES_match_list;  
                        VALUE : in   WAVES_match_list;  
                        PIN_SET : in   Pinset := ALL_PINS );
```

Home Laboratory, USAF

## Interface Package

```
--  
-- This procedure compares a Sim_code to an element (given by PIN) of  
-- the WAVES port list and assigns the corresponding element of the  
-- WAVES match list to the result.  
--  
procedure Compare( signal WPL : inout WAVES_port_list;  
                  signal WML : inout WAVES_match_list;  
                  CODE : in   Sim_code;  
                  PIN   : in   Test_pins );
```

Rome Laboratory, USAF

## Interface Package

```
--  
-- This procedure compares each Sim_code in the Sim_vector to the  
-- elements (given by PIN_SET) of the WAVES port list and assigns  
-- the corresponding elements of the WAVES match list to the results.  
--  
procedure Compare( signal WPL      : inout WAVES_port_list;  
                  signal WML      : inout WAVES_match_list;  
                  VECTOR : in   Sim_code_vector;  
                  PIN_SET : in   Pinset := ALL_PINS );
```

Rome Laboratory, USAF

## Interface Package

```
--  
-- This procedure returns the tag string associated with the given  
-- pin from the WAVES port list.  
--
```

```
procedure Get_Tag( signal WPL : inout WAVES_port_List;  
                  TAG : out String;  
                  PIN : in Test_pins );
```

Rome Laboratory, USAF

## Interface Package

```
--  
-- This procedure returns the tag string associated with the given  
-- pin set from the WAVES port list.  
--
```

```
procedure Get_Tag( signal WPL : inout WAVES_port_List;  
                  TAG : out String;  
                  PIN_SET : in Pinset := ALL_PINS );
```

Rome Laboratory, USAF

## Interface Package

```
--  
-- This procedure returns the tag string associated with the current  
-- WAVES port list.  
--  
procedure Get_Tag( signal WPL : inout WAVES_port_List;  
                  TAG : out String );  
  
end WAVES_VHDL_Interface;
```

Rome Laboratory, USAF

## Test Bench Example

```
architecture ALU of Test_Bench is  
  
  --  
  -- Define a signals for receiving outputs of model etc...  
  --  
begin  
  
  --  
  -- Process to generate the waveform.  
  --  
  WAVES : Waveform( CONNECT );
```

Rome Laboratory, USAF

## Test Bench Example

```
--  
-- Connect the DUT model.  
--  
ALU : FOUR_BIT_ALU  
  
port map ( A_INPUTS    => Sim_Code_Vector_Of( CONNECT, A_PINS ),  
          B_INPUTS    => Sim_Code_Vector_Of( CONNECT, B_PINS ),  
          SELECT_LINES => Sim_Code_Vector_Of( CONNECT, S_PINS ),  
          CN           => Sim_Code_Of( CONNECT, CN ),  
          M           => Sim_Code_Of( CONNECT, M ),  
          F_OUTPUTS   => FUNCTION_OUTPUTS,  
          AEQB        => A_EQUAL_B,  
          NOTP        => NOT_P,  
          CN_4        => CN_4,  
          NOTG        => NOT_G );
```

Home Laboratory, USAF

## Test Bench Example

```
MONITOR : process  
begin  
  wait on CONNECT;  
  assert ( Compare( CONNECT, FUNCTION_OUTPUTS, F_PINS )  
    and Compare( CONNECT, A_EQUAL_B, AEQB )  
    and Compare( CONNECT, NOT_P, NOTP )  
    and Compare( CONNECT, CN_4, CN_4 )  
    and Compare( CONNECT, NOT_G, NOTG ) )  
  report "==> Error in functional vector."  
  severity WARNING;  
end process;  
  
end ALU;
```

Home Laboratory, USAF

## **Status & Caveats**

- **Implemented, Tested on 4-Bit ALU Example**
  - **Need to Thoroughly Test & Evaluate the Interface package**
- **Order of the Test\_Pins Declaration is Critical to the Ability to Use the Bus Functions of the Interface**
- **Allows Users to Write Portable VHDL/WAVES Test Benches**
- **Eases the use of WAVES in VHDL Environment**