ResolutionPkg Package User Guide

User Guide for Release 2016.11

By

Jim Lewis

SynthWorks VHDL Training

Jim@SynthWorks.com

http://www.SynthWorks.com

Table of Contents

1 ResolutionPkg Overview 3
2 Maximum Resolution Functions 3
 2.1 Resolved_max 3
 2.2 Subtypes based on Resolved_max 3
 2.3 Use Model 3
 2.4 Compatibility Mode Types 4
3 Summing Resolution 5
 3.1 Resolved_sum 5
 3.2 Subtypes 5
 3.3 Compatibility Mode Types 5
4 Legacy Resolution Functions 5
 4.1 Resolved 5
 4.2 Subtypes based on Resolved 6
 4.3 Use Model 6
5 Compiling ResolutionPkg and Friends 7
6 About ResolutionPkg 7
7 Future Work 7
8 About the Author - Jim Lewis 7

1 ResolutionPkg Overview

ResolutionPkg provides resolution functions used to resolve elements of a record. OSVVM uses records as a means of implementing its transaction interfaces.

2 Maximum Resolution Functions

2.1 Resolved_max

A function named resolved_max is defined for the types std_ulogic, bit, integer, time, real, character, and Boolean.

```
function resolved_max ( s : std_ulogic_vector) return std_ulogic ;
```

2.2 Subtypes based on Resolved_max

The following subtypes are defined in ResolutionPkg:

```
subtype  std_logic_max is resolved_max std_ulogic ;
subtype  std_logic_vector_max is (resolved_max) std_ulogic_vector ;
subtype  unsigned_max is (resolved_max) unresolved_unsigned ;
subtype  signed_max   is (resolved_max) unresolved_signed ;
subtype  bit_max is resolved_max bit ;
subtype  bit_vector_max is (resolved_max) bit_vector ;
subtype  integer_max is resolved_max integer ;
subtype  integer_vector_max is (resolved_max) integer_vector ;
subtype  time_max is resolved_max time ;
subtype  time_vector_max is (resolved_max) time_vector ;
subtype  real_max is resolved_max real ;
subtype  real_vector_max is (resolved_max) real_vector ;
subtype  character_max is resolved_max character ;
subtype  string_max is (resolved_max) string ;
subtype  boolean_max is resolved_max boolean ;
subtype  boolean_vector_max is (resolved_max) boolean_vector ;
```

2.3 Use Model

OSVVM uses records as transaction interfaces. Since the record is inout of more than one entity, one option is to use the above resolved subtypes as elements of the record.

```
package InterfacePkg is
  type InterfaceRecType is record
    Rdy         : std_logic_max ;
    Ack         : std_logic_max ;
    Operation   : integer_max ;
    Address     : integer_max ;
    Data_in     : integer_max ;
    Data_out    : integer_max ;
  end record InterfaceRecType ;

  procedure DoTrans (IntefaceRec : inout InterfaceRecType ; . . . ) ;
  .  .  .
End package InterfacePkg ;
```

```
entity TestCtrl is
  port (
    InterfaceRec : inout InterfaceRecType ;
    . . .
  ) ;
end entity TestCtrl ;

entity IntefaceModel is
  port (
    InterfaceRec : inout InterfaceRecType ;
    . . .
  ) ;
end entity TestCtrl ;
```

The default value on each record element is type'left. The resolved_max resolution function returns the maximum value driven on the record. Undriven record elements from a particular model contribute the value type'left, and hence, if the other model is driving the corresponding element of the record, its value is the maximum value.

By using a subtype, the value will convert automatically to other types that are a subtype of the base type. Going further it will also convert to the base type since it is a subtype of itself.

2.4 Compatibility Mode Types

Subtypes of arrays, such as std_logic_vector_max, that use the resolution function of its elements is a new feature for VHDL-2008. There are simulators that do not currently support this capability. For that reason, the package also provides the following type definitions.

```
type     std_logic_vector_max_c is array (natural range <>) of std_logic_max ;
type     unsigned_max_c is array (natural range <>) of std_logic_max ;
type     signed_max_c is array (natural range <>) of std_logic_max ;
type     bit_vector_max_c is array (natural range <>) of bit_max ;
type     integer_vector_max_c is array (natural range <>) of integer_max ;
type     time_vector_max_c is array (natural range <>) of time_max ;
type     real_vector_max_c is array (natural range <>) of real_max ;
type     string_max_c is array (positive range <>) of character_max ;
type     boolean_vector_max_c is array (natural range <>) of boolean_max ;
```

These work well, however, not as well as subtypes. With a type, the conversion between string_max_c is no longer automatic with string. In this case, a type conversion (type cast can be used).

```
signal FileName : string(1 to 10) ;
type InterfaceRecType is record
  FileName : string_max_c ;
  . . .
end record InterfaceRecType ;
signal InterfaceRec : InterfaceRecType ;
. . .
InterfaceRec.FileName <= string_max_c(FileName) ;
```

```
    . . .
    FileName <= string(InterfaceRec.FileName) ;
```

3 Summing Resolution

3.1 Resolved_sum

A function named resolved_sum is defined for the types integer, time, and real.

```
    function resolved_sum ( s : integer_vector ) return integer ;
    function resolved_sum ( s : time_vector ) return time ;
    function resolved_sum ( s : real_vector ) return real ;
```

3.2 Subtypes

Requests a transaction from the test initiation (client) to the model (TLM or VVC).

```
    subtype  integer_sum is resolved_sum integer ;
    subtype  integer_vector_sum is (resolved_sum) integer_vector ;
    subtype  time_sum is resolved_sum time ;
    subtype  time_vector_sum is (resolved_sum) time_vector ;
    subtype  real_sum is resolved_sum real ;
    subtype  real_vector_sum is (resolved_sum) real_vector ;
```

3.3 Compatibility Mode Types

Suspends a model until a transaction is requested via RequestTransaction.

```
    type     integer_vector_sum_c is array (natural range <>) of integer_sum ;
    type     time_vector_sum_c is array (natural range <>) of time_sum ;
    type     real_vector_sum_c is array (natural range <>) of real_sum ;
```

4 Legacy Resolution Functions

The legacy resolution functions were the first resolution functions added to ResolutionPkg. Usage of these included using std_logic_vector. The disadvantage to this is that the default initialization of std_logic is 'U' and the resolved value of 'U' and any other value is 'U'. As a result, the ports require initialization to a non-driving state, such as 'Z'.

4.1 Resolved

A function named resolved is defined for the types integer, time, real, character, and boolean. These resolution functions require a non-driving value of 0, 0 ns, 0.0, NUL, and FALSE respectively. If there is more than one non-driving value, these functions will generate an assert FAILURE. Note though that in most simulators, this will not cause the normal stopping action.

```
    function resolved ( s : integer_vector ) return integer ;
```

4.2 Subtypes based on Resolved

The following subtypes are defined in ResolutionPkg:

```
subtype  resolved_integer is resolved integer ;
subtype  resolved_time is resolved time ;
subtype  resolved_real is resolved real ;
subtype  resolved_character is resolved character ;
type resolved_string is array (positive range <>) of resolved_character;
subtype  resolved_boolean is resolved boolean ;
```

4.3 Use Model

OSVVM uses records as transaction interfaces.   Since the record is inout of more than one entity, one option is to use the above resolved subtypes as elements of the record.

```
package InterfacePkg is
  type InterfaceRecType is record
    Rdy         : std_logic ;
    Ack         : std_logic ;
    Operation   : resolved_integer ;
    Address     : resolved_integer ;
    Data_in     : resolved_integer ;
    Data_out    : resolved_integer ;
  end record InterfaceRecType ;

  constant INTERFACE_INIT : InterfaceRecType := (
    Rdy         => 'Z',
    Ack         => 'Z',
    Operation   => 0,
    Address     => 0,
    Data_in     => 0,
    Data_out    => 0
  ) ;

  procedure DoTrans (IntefaceRec : inout InterfaceRecType ; . . . ) ;
  .  .  .
End package InterfacePkg ;
entity TestCtrl is
  port (
    InterfaceRec : inout InterfaceRecType := INTERFACE_INIT ;
    . . .
  ) ;
end entity TestCtrl ;

entity IntefaceModel is
  port (
    InterfaceRec : inout InterfaceRecType := INTERFACE_INIT ;
    . . .
  ) ;
end entity TestCtrl ;
```

When a value is undriven by a model, the model will contribute the value from the default initialization for that record element.

Since the types used here are subtypes, values convert automatically.

5 Compiling ResolutionPkg and Friends

See OSVVM_release_notes.pdf for the current compilation directions. Rather than referencing individual packages, we recommend using the context declaration:

```
library OSVVM ;
  context osvvm.OsvvmContext ;
```

6 About ResolutionPkg

ResolutionPkg was developed and is maintained by Jim Lewis of SynthWorks VHDL Training. Prior to its release to OSVVM it was used in SynthWorks' VHDL classes.

Please support our effort in supporting the OSVVM library of packages by purchasing your VHDL training from SynthWorks.

ResolutionPkg is released under the Perl Artistic open source license. It is free (both to download and use - there are no license fees). You can download it from osvvm.org or from our development area on GitHub.

If you add features to the package, please donate them back under the same license as candidates to be added to the standard version of the package. If you need features, be sure to contact us. I blog about the packages at http://www.synthworks.com/blog. We also support the OSVVM user community and blogs through http://www.osvvm.org.

Find any innovative usage for the package? Let us know, you can blog about it at osvvm.org.

7 Future Work

ResolutionPkg.vhd is a work in progress and will be updated from time to time.

Caution, undocumented items are experimental and may be removed in a future version.

8 About the Author - Jim Lewis

Jim Lewis, the founder of SynthWorks, has thirty plus years of design, teaching, and problem solving experience. In addition to working as a Principal Trainer for

SynthWorks, Mr Lewis has done ASIC and FPGA design, custom model development, and consulting.

Mr. Lewis is chair of the IEEE 1076 VHDL Working Group (VASG) and is the primary developer of the Open Source VHDL Verification Methodology (OSVVM.org) packages. Neither of these activities generate revenue. Please support our volunteer efforts by buying your VHDL training from SynthWorks.

If you find bugs these packages or would like to request enhancements, you can reach me at jim@synthworks.com.