SynthWorks

VHDL Training Experts

Scoreboard Package Quick Reference

1. Generic Package Interface

```
package ScoreboardGenericPkg is
generic (
  type ExpectedType ;
  type ActualType ;
  function Match(Actual : ActualType ;
     Expected : ExpectedType)
     return boolean ;
  function expected_to_string(
     A : ExpectedType) return string ;
  function actual_to_string(
     A : ActualType) return string
) ;
```

ExpectedType is the input to the scoreboard. ActualType is the value to be checked against the oldest ExpectedType value. ExpectedType and ActualType may be the same type. Match is a function that compares ExpectedType with ActualType. Match is often mapped to "=". expected_to_string converts ExpectedType to a string value (for reports). actual_to_string converts ActualType to a string.

2. Package Instance

```
library ieee ;
  use ieee.std_logic_1164.all ;
  use ieee.numeric_std.all ;

package ScoreBoardPkg_slv is new
work.ScoreboardGenericPkg
  generic map (
    ExpectedType => std_logic_vector,
    ActualType   => std_logic_vector,
    Match              => std_match,
    expected_to_string  => to_hstring,
    actual_to_string    => to_hstring
) ;
```

Note, that ExpectedType and ActualType do not need to be constrained types.

3. Compatibility Package

ScoreboardPkg_slv_c.vhd contains equivalent subtypes and aliases to work like the above package instance.

4. Basic Operations

4.1 Scoreboard = Shared Variable

A scoreboard is a shared variable.

```
shared variable SB : ScoreBoardPType ;
```

If using more than one scoreboard package instance, disambiguate the type using a fully selected name.

```
shared variable SB_uart :
 work.ScoreBoardPkg_Uart.ScoreBoardPType;
```

4.2 Push

Add expected value (ExpectedType) to the scoreboard.

```
SB.Push(ExpectedVal) ;
```

4.3 Check

Check a received value (ActualType) with value in scoreboard. Reports PASSED/ERROR via the AlertLogPkg. Can also be called as a function that returns TRUE when PASSED.

```
SB.Check(ReceiveVal) ;
```

4.4 Pop

Use scoreboard as FIFO, get oldest value. Uses an out mode variable parameter of ExpectedType.

```
SB.Pop(ExpectedVal) ;
```

4.5 SetAlertLogID

Create an AlertLogID. Use the ID for reporting errors.

```
SB.SetAlertLogID(
  Name => "SB_UART",
  ParentID => OSVVM_ALERTLOG_ID ) ;
```

4.6 GetAlertLogID

Get the AlertLogID from the scoreboard internals.

```
SB_ID := SB.GetAlertLogID ;
```

4.7 Find

Return the ItemNumber of the oldest expected value that matches the received value. Find + Flush support systems that drop items before they are synchronized.

```
ItemNum := SB.Find(ReceiveVal) ;
```

4.8 Flush

Quietly drop all values whose item number is less than the specified item number. Find + Flush support systems that drop items before they are synchronized.

```
SB.Flush(ItemNum) ;
```

4.9 Empty

Check if the Scoreboard is empty.

```
if not SB.Empty then …
```

4.10 GetErrorCount

Only intended for non-alert flows. If not using separate AlertLogIDs and ReportAlerts, GetErrorCount returns the current error count.

```
ErrCnt := SB.GetErrorCount ;
```

4.11 GetPushCount / GetItemCount

Get number of items pushed into the scoreboard.

```
print("…" & to_string(SB.GetItemCount));
```

4.12 GetCheckCount

Get number of items checked by the scoreboard.

```
print("…" & to_string(SB.GetCheckCount));
```

4.13 GetPopCount

Get number of items popped out of the scoreboard.

```
print("…" & to_string(SB.GetPopCount));
```

4.14 GetDropCount

Get number of items dropped by the scoreboard.

```
print("…" & to_string(SB.GetDropCount));
```

4.15 GetFifoCount

Get number of items in the FIFO inside the scoreboard.

```
print("…" & to_string(SB.GetFifoCount));
```

4.16 SetName

Gives the scoreboard a name for reporting. Use if using a single ALertLogID for multiple items (scoreboards or other).

```
SB.SetName("Uart Scoreboard") ;
```

4.17 GetName

Get the scoreboard name

```
print("…" & SB.GetName) ;
```


SynthWorks Design Inc.
VHDL Design and Verification Training
11898 SW 128th Ave. Tigard OR 97223 (800)-505-8435
http://www.SynthWorks.com jim@synthworks.com

SynthWorks

VHDL Training Experts

5. Tagged Scoreboards

Tagged Scoreboards are used for systems that allow transactions to execute out of order.

Tags are represented as string values (since most types convert to string using to_string). A tag value is specified as the first value in the calls to push, check, and pop, such as shown below. In all examples, ExpectedVal has the type ExpectedType, and ReceiveVal has the type ActualType.

```
SB.Push("WriteOp", ExpectedVal) ;
SB.Check("WriteOp", ReceiveVal) ;
SB.Pop("WriteOp", ExpectedVal) ;

if SB.Empty("MyTag")  then   …
```

For Check (and Pop), the item checked (or returned) is the oldest item with the matching tag.

```
ItemNum := SB.Find("ReadOp", ReceiveVal);
SB.Flush("ReadOp", ItemNum) ;
```

For Flush, only items matching the tag are removed. In some systems, it may be appropriate to do the Find with the tag and the flush without the tag.

6. Indexed Scoreboards

Indexed scoreboards emulates arrays of protected types, since the language does not support this.

Indexed scoreboards are for systems, such as a network switch that have multiple scoreboards that are most conveniently represented as an array.

6.1 Setting Array Indices

Use SetArrayIndex to create the array indices. The following creates an array with indices 1 to 5:

```
SB.SetArrayIndex(5) ;
```

To create array indices with a different range, such as 3 to 8, use the following.

```
SB.SetArrayIndex(3, 8) ;
```

Slicing and null arrays of scoreboards are not supported. Negative indices are supported.

6.2 Getting Array Indices

Use GetArrayIndex to get the indices as an integer_vector.

```
Index_IV := SB.GetArrayIndex ;
```

Use GetArrayLength to determine the number of scoreboards (effectively the length of the array).

```
Index_int := SB.GetArrayLength ;
```

6.3 Arrays of Scoreboards

The following operations are appropriate for any array of scoreboards. Procedures and functions not documented here are from AlertLogPkg.

```
-- Create 3 indexed scoreboards
SB.SetArrayIndex(1, 3);

-- TB_ID via AlertLogPkg
TB_ID := GetAlertLogID("TB") ;
SB.SetAlertLogID(1, "SB1", TB_ID) ;
SB.SetAlertLogID(2, "SB2", TB_ID) ;
SB.SetAlertLogID(3, "SB3", TB_ID) ;

-- display PASSED logs via AlertLogPkg
SetLogEnable(TB_ID, PASSED, TRUE) ;

-- Turn off Error messages for SB1
SB1_ID := GetAlertLogID(1) ;
SetAlertEnable(SB1_ID, ERROR, FALSE) ;

-- Check at least 100 items and
-- Finish Empty
SB.CheckFinish(1, 100, TRUE) ;
SB.CheckFinish(2, 100, TRUE) ;
SB.CheckFinish(3, 100, TRUE) ;

-- test completion via AlertLogPkg
ReportAlerts ;

-- Getting Error Counts (non-Alert)
TotalErrorCount :=
  SB.GetErrorCount(1) +
  SB.GetErrorCount(2) +
  SB.GetErrorCount(3);

TotalErrorCountAlt := SB.GetErrorCount ;
```

6.4 Arrays of Simple Scoreboards

The following are operations appropriate for arrays of simple scoreboards. In all examples, 4 is the index, ExpectedVal has the type ExpectedType, and ReceiveVal has the type ActualType.

```
SB.Push(4, ExpectedVal) ;
SB.Check(4, ReceiveVal) ;
SB.Pop(4, ExpectedVal) ;

if SB.Empty(4)  then   …

ItemNum := SB.Find(4, ReceiveVal);
SB.Flush(4, ItemNum) ;
```

6.5 Arrays of Tagged Scoreboards

The following are operations appropriate for arrays of tagged scoreboards. In all examples, 4 is the index, values in quotes are the tag value, ExpectedVal has the type ExpectedType, and ReceiveVal has the type ActualType. Operations where either using a tag or not is appropriate are marked with "**".

```
SB.Push(4, "WriteOp", ExpectedVal) ;
SB.Check(4, "WriteOp", ReceiveVal) ;
SB.Pop(4, "WriteOp", ExpectedVal) ;

if SB.Empty(4, "MyTag")  then   …  -- **
if SB.Empty(4)  then   …  -- **

ItemNum := SB.Find(4, "Red", ReceiveVal);
-- two possible alternatives
SB.Flush(4, "Red", ItemNum) ; -- **
SB.Flush(4, ItemNum) ; -- **
```


SynthWorks Design Inc.
VHDL Design and Verification Training
11898 SW 128th Ave. Tigard OR 97223 (800)-505-8435
http://www.SynthWorks.com jim@synthworks.com