

MemoryPkg User Guide

User Guide for Release 2016.01

By

Jim Lewis

SynthWorks VHDL Training

Jim@SynthWorks.com

<http://www.SynthWorks.com>

Table of Contents

1	AlertLogPkg Overview.....	3
2	AlertLogPkg Use Models	Error! Bookmark not defined.
3	Simple Mode: Global Alert Counter	3
4	Hierarchy Mode: Separate Alert Counters.....	Error! Bookmark not defined.
5	Method Reference	4
5.1	Package References	4
5.2	AlertType	Error! Bookmark not defined.
5.3	Simple Alerts	Error! Bookmark not defined.
5.4	Creating Hierarchy: GetAlertLogID.....	Error! Bookmark not defined.
5.5	FindAlertLogID: Find an AlertLogID	Error! Bookmark not defined.
5.6	Hierarchical Alerts	Error! Bookmark not defined.
5.7	ReportAlerts: Reporting Alerts.....	Error! Bookmark not defined.
5.8	SetAlertLogName: Setting the Test Name	5
5.9	SetGlobalAlertEnable: Alert Global Enable / Disable	Error! Bookmark not defined.
5.10	SetAlertEnable: Alert Enable / Disable.....	Error! Bookmark not defined.
5.11	SetAlertStopCount: Alert Stop Counts	Error! Bookmark not defined.
5.12	AlertCountType.....	Error! Bookmark not defined.
5.13	GetAlertCount	Error! Bookmark not defined.
5.14	GetEnabledAlertCount	Error! Bookmark not defined.
5.15	GetDisabledAlertCount.....	Error! Bookmark not defined.
5.16	ClearAlerts: Reset Alert and Stop Counts.....	Error! Bookmark not defined.
5.17	Math on AlertCountType	Error! Bookmark not defined.
5.18	SumAlertCount: AlertCountType to Integer Error Count	Error! Bookmark not defined.
5.19	SetAlertLogJustify	Error! Bookmark not defined.
5.20	LogType.....	Error! Bookmark not defined.
5.21	Simple Logs.....	Error! Bookmark not defined.
5.22	Hierarchical Logs	Error! Bookmark not defined.
5.23	SetLogEnable: Enable / Disable Logging	Error! Bookmark not defined.
5.24	IsLoggingEnabled	Error! Bookmark not defined.
5.25	OsvvmOptionsType	Error! Bookmark not defined.
5.26	SetAlertLogOptions: Configuring Report Options.....	Error! Bookmark not defined.
5.27	DeallocateAlertLogStruct.....	Error! Bookmark not defined.
5.28	InitializeAlertLogStruct	Error! Bookmark not defined.
6	Compiling AlertLogPkg and Friends.....	6
7	About AlertLogPkg	6
8	Future Work	7

1 MemoryPkg Overview

The AlertLogPkg provides a protected type and methods to simplify the creation of data structures required to model larger memories.

2 Simple Mode: Global Alert Counter

By default, there is a single global alert counter. All designs that use alert or log need to reference the package AlertLogPkg.

```
use osvvm.AlertLogPkg.all ;
architecture Test1 of tb is
```

Use Alert to flag an error, AlertIf to flag an error when a condition is true, or AlertIfNot to flag an error when a condition is false (similar to assert). Alerts can be of severity FAILURE, ERROR, or WARNING.

```
--          message,          level
When others => Alert("Illegal State", FAILURE) ;
. . .
--          condition,          message,          level
AlertIf(ActualData /= ExpectedData, "Data Miscompare ...", ERROR) ;
. . .
read(Buf, A, ReadValid) ;
--          condition, message,          level
AlertIfNot( ReadValid, "read of A failed", FAILURE) ;
```

The output for an alert is as follows. Alert adds the time at which the log occurred.

```
%% Alert ERROR   Data Miscompare ... at 20160 ns
```

When a test completes, use ReportAlerts to provide a summary of errors.

```
ReportAlerts ;
```

When a test passes, the following message is generated:

```
%% DONE  PASSED  t1_basic  at 120180 ns
```

When a test fails, the following message is generated (on a single line):

```
%% DONE  FAILED  t1_basic  Total Error(s) = 2  Failures: 0  Errors: 1  Warnings: 1
at 120180  ns
```

Similar to assert, by default, when an alert FAILURE is signaled, a test failed message (see ReportAlerts) is produced and the simulation is stopped. This action is controlled by a stop count. The following call to SetAlertStopCount, causes a simulation to stop after 20 ERROR level alerts are received.

```
SetAlertStopCount(ERROR, 20) ;
```

Alerts can be enabled by a general enable, `SetGlobalAlertEnable` (disables all alert handling) or an enable for each alert level, `SetAlertEnable`. The following call to `SetAlertEnable` disables WARNING level alerts.

```
SetGlobalAlertEnable(TRUE) ; -- Default
SetAlertEnable(WARNING, FALSE) ;
```

Logs are used for verbosity control. Log level values are ALWAYS, DEBUG, FINAL, and INFO.

```
Log ("A message", DEBUG) ;
```

Log formats the output as follows.

```
%% Log    DEBUG  A Message  at 15110 ns
```

Each log level is independently enabled or disabled. This allows the testbench to support debug or final report messages and only enable them during the appropriate simulation run. The log ALWAYS is always enabled, all other logs are disabled by default. The following call to `SetLogEnable` enables DEBUG level logs.

```
SetLogEnable(DEBUG, TRUE) ;
```

3 Method Reference

3.1 Package References

Using `MemoryPkg` requires the following package references:

```
library osvvm ;
use osvvm.MemoryPkg.all ;
```

3.2 MemInit

Alert levels can be FAILURE, ERROR, or WARNING.

```
type AlertType is (FAILURE, ERROR, WARNING) ;
```

3.3 MemWrite

Simple alerts accumulate alerts in the default `AlertLogID` (`ALERTLOG_DEFAULT_ID`). It supports the basic overloading and usage:

```
procedure Alert( Message : string ; Level : AlertType := ERROR ) ;
. . .
Alert("Uart Parity") ; -- ERROR by default
```

3.4 MemRead Procedure

Each level in a hierarchy is referenced with an `AlertLogID`. The function, `GetAlertLogID`, creates a new `AlertLogID`. If an `AlertLogID` already exists for the specified name,

GetAlertLogID will return its AlertLogID. It is recommended to use the instance label as the Name. The interface for GetAlertLogID is as follows.

```
impure function GetAlertLogID(Name : string ;
    ParentID : AlertLogIDType := ALERTLOG_BASE_ID ) return AlertLogIDType ;
    GetAlertLogID("UART_1", ALERTLOG_BASE_ID);
```

3.5 MemRead Function

The function, FindAlertLogID, finds an existing AlertLogID. If the AlertLogID is not found, ALERTLOG_ID_NOT_FOUND is returned. The interface for FindAlertLogID is as follows.

```
impure function FindAlertLogID(Name : string ; ParentID : AlertLogIDType)
    return AlertLogIDType ;
impure function FindAlertLogID(Name : string ) return AlertLogIDType ;
```

3.6 MemErase

Hierarchical alerts require the AlertLogID to be specified in the call to alert. It supports the basic overloading and usage:

```
procedure alert(
    AlertLogID : AlertLogIDType ;
    Message : string ;
    Level : AlertType := ERROR
) ;
. . .
Alert(UartID, "Uart Parity", ERROR) ;
```

3.7 Deallocate

At test completion alerts are reported with ReportAlerts.

```
procedure ReportAlerts (
    Name : string := "" ;
    AlertLogID : AlertLogIDType := ALERTLOG_BASE_ID ;
    ExternalErrors : AlertCountType := (others => 0)
) ;
. . .
ReportAlerts ;
```

3.8 ReadMemB and ReadMemH

Designed to mimic Verilog system functions \$readmemb and readmemh.

```
procedure ReadMemB (
    Name : string ;
    StartAddr : natural := 0 ;
    FinishAddr : natural := natural'right
) ;
```

The file shall contain only of the following:

- White space (spaces, new lines, tabs, and form-feeds)

- Comments (either //, #, or --)
- @ character (designating the adjacent number is an address)
- Binary or hexadecimal numbers

Addresses specified in the call to ReadMemB or ReadMemH provide both an initial starting address and a range of valid addresses for memory operations. Addressing advances from StartAddr to FinishAddr. If FinishAddr is greater than StartAddr, then the next address is one larger than the current one, otherwise, the next address is one less than the current address.

Addresses may also be specified in the file in the format '@' followed by a hexadecimal number as shown below. There shall not be any space between the '@' and the number. The address read must be between StartAddr and FinishAddr or a FAILURE is generated.

@hhhhh

Values not preceded by an '@' character are data values. Data values must be separated by white space or comments from other values. ReadMemB requires values compatible with read for std_ulogic. ReadMemH requires hexadecimal values compatible with hread for std_ulogic_vector. If more digits are read than are required by the memory, left hand characters will be dropped provided they are 0. If fewer digits are read than are required by the memory, left hand characters will be 0.

4 Compiling AlertLogPkg and Friends

Use of AlertLogPkg requires use NamePkg and OsvvmGlobalPkg. The compile order is: NamePkg.vhd, OsvvmGlobalPkg.vhd, TranscriptPkg.vhd, and AlertLogPkg.vhd. Compiling the packages requires VHDL-2008.

5 About AlertLogPkg

AlertLogPkg was developed and is maintained by Jim Lewis of SynthWorks VHDL Training. It originated as an interface layer to the BitVis Utility Library (BVUL). However, it required a default implementation and that default implementation grew into its own project.

Please support our effort in supporting AlertLogPkg and OSVVM by purchasing your VHDL training from SynthWorks.

AlertLogPkg 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 <http://www.synthworks.com/downloads>. It will be updated from time to time. Currently there are numerous planned revisions.

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](http://www.osvvm.org).

6 Future Work

AlertLogPkg.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.

7 About the Author - Jim Lewis

Jim Lewis, the founder of SynthWorks, has twenty-eight 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.