SynthWorks

VHDL Training Experts

AlertLogPkg and ReportPkg Quick Ref

1. Package References

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

2. Simple vs Hierarchy Mode

Simple mode tracks alerts, logs, and affirmations using the default AlertLogID, ALERTLOG_DEFAULT_ID.

Hierarchy mode tracks alerts, logs, and affirmations using a specified AlertLogID. Each ID is created in a shared variable inside of AlertLogPkg using GetAlertID.

All alerts, logs, and affirmations support both simple and hierarchy mode. In hierarchy mode, the AlertLogID is always specified as the first parameter.

3. SetTestName

Sets the test name. Printed by ReportAlerts.

```
SetTestName("Uart1") ;
```

4. NewID: Create Hierarchy

Create an AlertLogId and associate it with the name.

```
Constant UartID : AlertLogIDType :=
 --                    Name,       Parent AlertLogID
    NewID("UART_1", ALERTLOG_BASE_ID);
```

Note that ALERTLOG_BASE_ID is the default

5. PathTail

PathTail is used in conjunction with PATH_NAME to determine an instance name for an entity.

```
Constant UartID : AlertLogIDType :=
    NewID(PathTail(UART'PATH_NAME));
```

6. Alert Levels

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

7. Alerts

Alerts are for error conditions. ERROR is the default. Alerts are enabled by default.

```
-- First Parameter, UartID is optional
Alert(UartID, "Uart Parity", ERROR) ;
AlertIf(UartID, Break='1', "Uart Break", ERROR) ;
AlertIfNot(UartID, ReadValid, "Read", FAILURE);
AlertIfDiff(UartID, "A.txt","B.txt", "A /= B", ERROR);
```

http://www.SynthWorks.com jim@SynthWorks.com

8. EndOfTestReports

Print text (via ReportAlerts) and YAML based alert reports.

```
EndOfTestReports ;
```

See AlertLog_user_guide.pdf for additional parameters.

9. ReportAlerts

Print a report of alerts and affirmations.

```
ReportAlerts ;
```

10. ReportNonZeroAlerts

Like ReportAlerts, except does not print a hierarchy element when it has zero errors.

11. SetAlertEnable

Enable or disable all alerts of a particular alert level:

```
--                  Level,      Enable
SetAlertEnable(WARNING, FALSE) ;
```

Enable or disable alerts for an AlertLogID and its children if the DescendHierarchy parameter is TRUE.

```
--                AlertLogID, Level, Enable, DescendHierarchy
SetAlertEnable(UartID, WARNING, FALSE, TRUE) ;
```

12. Alert Stop Counts

A simulation stops when an alert stop count is reached. Use SetAlertStopCount without an AlertLogID to set the stop count for the top level of the AlertLog hierarchy.

```
--                        Level,          Count
SetAlertStopCount(ERROR,      20) ;
```

Use SetAlertStopCount with an AlertLogID, to control the stop count for a particular verification component and all of its ancestors (parents).

```
--                  AlertLogID,  Level,          Count
SetAlertStopCount(UartID, ERROR,      20) ;
```

13. AlertCountType

Alerts are stored as a value of AlertCountType.

```
type AlertCountType is array (AlertType) of integer ;
```

14. GetAlertCount

Get count of all alerts as either AlertCountType or integer.

```
ErrorCount := GetAlertCount ;
```

Get count of alerts for a particular AlertLogID:

```
UartErrorCount := GetAlertCount(UartID) ;
```

15. GetEnabledAlertCount

GetEnabledAlertCount is similar to GetAlertCount except it returns 0 for disabled alert levels.

```
TopEnabledErrors := GetEnabledAlertCount ;
UartEnabledErrors := GetEnabledAlertCount(UartID) ;
```

16. GetDisabledAlertCount

GetDisabledAlert count sums up the disabled alerts for either the entire design hierarchy or a particular ID.

```
DisabledErrorCount := GetDisabledErrorCount ;
DisabledErrorCount := GetDisabledErrorCount(UartID) ;
```

17. Math on AlertCountType

```
TotalAlertCount := Phase1Count + Phase2Count ;
TotalErrors := GetAlertCount - ExpectedErrors ;
NegateErrors := -ExpectedErrors ;
```

18. SumAlertCount

SumAlertCount sums up the WARNING, ERROR, and FAILURE values into a single integer value.

```
ErrorCountInt := SumAlertCount(AlertCount) ;
```

19. SetAlertLogJustify

Justifies the name field of Alert and Log.

```
SetAlertLogJustify(TRUE) ;
```

20. ClearAlerts

Reset alert counts to 0.

```
ClearAlerts ;
```

21. ClearAlertStopCounts

Reset alert stop counts to their default.

```
ClearAlerts ;
```

22. SetGlobalAlertEnable

Alerts can be globally disabled. The intent is to be able to disable all alerts until the system goes into reset.

```
constant En : boolean := SetGlobalAlertEnable(FALSE);
. . .
SetGlobalAlertEnable(TRUE)  ;
```

23. SetAlertLogPrefix, SetAlertLogSuffix

Print prefix just before alert or log message. Print suffix just after alert or log message.

```
SetAlertLogPrefix(AlertLogID, "Test A") ;
```

24. UnSetAlertLogPrefix, UnSetAlertLogSuffix

Deallocate the prefix/suffix so it no longer prints.

25. GetAlertLogPrefix, GetAlertLogSuffix

Return the string value of the prefix/suffix.

26. Log Levels
type LogType is (DEBUG, PASSED, INFO, ALWAYS) ;

27. Logs
Log prints when the LogLevel is enabled.

```
-- First Parameter, UartID is optional
Log(UartID, "Uart Parity Received", DEBUG) ;
```

Log also prints if the optional Enable is TRUE:

```
-- First Parameter, TestID is optional
Log(TestID, "TB out of reset", INFO, TRUE) ;
```

28. Log Enable / Disable
Enable all logs of a particular log level:

```
--                Level,  Enable
SetLogEnable(DEBUG,  TRUE) ;
```

Enable logs for an AlertLogID and log level. Also enable its children if the DescendHierarchy is TRUE.

```
--             AlertLogID,  Level,  Enable,  DescendHierarchy
SetLogEnable(UartID, WARNING, FALSE, FALSE) ;
```

29. ReadLogEnables
Read log enables from a file.

```
ReadLogEnables("InitEnables.txt") ;
```

30. IsLogEnabled / GetLogEnable
IsLogEnabled returns true when a log level is enabled.

```
-- First Parameter, UartID is optional
If IsLogEnabled(UartID, DEBUG) then
```

31. Affirmations
Affirmations combine Alerts and Logs and are intended for self-checking. A passing affirmation uses log PASSED and a failing affirmation uses Alert ERROR.

```
-- First Parameter, UartID is optional
AffirmIf(UartID,
   Data = Expected, "Data: " & to_string(Data),
   " /=  Expected: " & to_string(Expected)) ;
AffirmIfNot(UartID, ReadValid, "Reading Test.txt") ;
AffirmIfEqual(UartID, ReceivedData, ExpectedData) ;
AffirmIfDiff(UartID, "Uart1.txt", "validated/Uart1.txt") ;
```

32. OsvvmOptionsType
OsvvmOptionsType defines the values for options. User values are: OPT_DEFAULT, DISABLED, FALSE, ENABLED, TRUE. The values DISABLED and FALSE are handled the same. The values ENABLED and TRUE are treated the same.

33. SetAlertLogOptions
With release 2024.03, all the settings here get their default value from OsvvmSettingsPkg_default. If you want to change the settings for all of test cases, copy OsvvmSettingsPkg_default to OsvvmSettingsPkg_local and modify the contents.

Alternately a given test case can change the settings using SetAlertLogOptions. Values shown are the default.

```
SetAlertOptions (
  FailOnWarning                => Enabled,
  FailOnDisabledErrors         => Enabled,
  FailOnRequirementErrors      => Enabled,
  ReportHierarchy              => Enabled,
  WriteAlertErrorCount         => Disabled,
  WriteAlertLevel              => Enabled,
  WriteAlertName               => Enabled,
  WriteAlertTime               => Enabled,
  WriteLogErrorCount           => Disabled,
  WriteLogLevel                => Enabled,
  WriteLogName                 => Enabled,
  WriteLogTime                 => Enabled,
  PrintPassed                  => Enabled,
  PrintAffirmations            => Disabled,
  PrintDisabledAlerts          => Disabled,
  PrintRequirements            => Disabled,
  PrintIfHaveRequirements      => Enabled,
  DefaultPassedGoal            => 1,
  WriteTimeLast                => FALSE,
  TimeJustifyAmount            => 9
) ;
```

SetAlertOptions will change as AlertLogPkg evolves. Use named association to ensure future compatibility.

```
SetAlertLogOptions (
   FailOnWarning             => FALSE,
   FailOnDisabledErrors      => FALSE
) ;
```

34. Options for ReportAlerts
FailOnWarning Warnings are test errors.
FailOnDisabledErrors Disabled errors cause FAILED.
FailOnRequirementErrors –Req Error is a test error.
ReportHierarchy Print alert summary for all levels.
PrintPassed Print number of passed in summary
PrintAffirmations Print a count of affirmations in details
PrintDisabledAlerts Print disabled alert counts
PrintRequirements Print requirements in details
PrintIfHaveRequirements in summary
ReportPrefix. Prefix for each line of ReportAlerts.
DoneName. Printed on first line of ReportAlerts.
PassName. Printed when PASSED.
FailName. Printed when alerts.

35. Options for Alert
WriteAlertErrorCount
WriteAlertLevel Print level.
WriteAlertName Print name.
WriteAlertTime Alerts print time.
AlertPrefix. Prefix for alert.

36. Options for Log
WriteLogLevel Print level.
WriteLogName Print name.
WriteLogTime Print time.
LogPrefix. Prefix for log.

37. DeallocateAlertLogStruct
Removes all temporary storage allocated by AlertLogBasePkg. See also ClearAlerts.

38. InitializeAlertLogStruct
Recreates the alert structure after it was deallocated.

39. File IO / Transcript Files
Alert, Log, and ReportAlerts all use the common transcript file defined in TranscriptPkg. Hence, when TranscriptFile is open, all output goes to it, otherwise, output goes to std.textio.OUTPUT.

SynthWorks Design Inc.
VHDL Intro, RTL, and Verification Training
11898 SW 128th Ave. Tigard OR 97223 +1-503-590-4787
http://www.SynthWorks.com jim@synthworks.com