SynthWorks

VHDL Training Experts

AlertLogPkg 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. SetAlertLogName

Sets the test name. Printed by ReportAlerts.

```
SetAlertLogName("Uart1") ;
```

4. GetAlertLogID: Create Hierarchy

Create an AlertLogId and associate it with the name.

```
Constant UartID : AlertLogIDType :=
 --                  Name,      Parent AlertLogID
     GetAlertLogID("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 :=
     GetAlertLogID(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);
```

8. ReportAlerts

Prints a report of alerts and affirmations.

```
ReportAlerts ;
```

See AlertLog_user_guide.pdf for additional parameters.

9. ReportNonZeroAlerts

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

10. 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) ;
```

11. Alert Stop Counts

A simulation stops when an alert stop count is reached. When used without AlertLogID, SetAlertStopCount sets the alert stop count for the top level to the specified value if the current count is integer'right, otherwise, it sets it to the specified value plus the current count.

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

When used with an AlertLogID, SetAlertStopCount sets the value for the specified AlertLogID and all of its ancestors (parents). At each level, the current alert stop count is set to the specified value when the current count is integer'right, otherwise, the value is set to the specified value plus the current count.

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

12. AlertCountType

Alerts are stored as a value of AlertCountType.

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

13. GetAlertCount

Get count of all alerts as either AlertCountType or integer.

```
ErrorCount := GetAlertCount ;
```

Get count of alerts for a particular AlertLogID:

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

14. GetEnabledAlertCount

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

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

15. GetDisabledAlertCount

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

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

16. Math on AlertCountType

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

17. SumAlertCount

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

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

18. SetAlertLogJustify

Justifies the name field of Alert and Log.

```
SetAlertLogJustify ;
```

19. ClearAlerts

Reset alert counts to 0 and stop counts to their default.

```
ClearAlerts ;
```

20. 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)  ;
```

21. Log Levels

```
type LogType is (DEBUG, PASSED, INFO, ALWAYS) ;
```

22. 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) ;
```

23. 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) ;
```

24. ReadLogEnables

Read log enables from a file.

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

25. IsLogEnabled / GetLogEnable

IsLogEnabled returns true when a log level is enabled.

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

26. 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") ;
```

27. 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.

28. SetAlertLogOptions

Configure the output of Alert, Log, and ReportAlerts with SetAlertLogOptions. Values shown are the default.

```
SetAlertOptions (
  FailOnWarning          => Enabled,
  FailOnDisabledErrors   => Enabled,
  ReportHierarchy        => Enabled,
  WriteAlertLevel        => Enabled,
  WriteAlertName         => Enabled,
  WriteAlertTime         => Enabled,
  WriteLogLevel          => Enabled,
  WriteLogName           => Enabled,
  WriteLogTime           => Enabled,
  AlertPrefix            => "%% Alert",
  LogPrefix              => "%% Log  ",
  ReportPrefix           => "%% ",
  DoneName               => "DONE",
  PassName               => "PASSED",
  FailName               => "FAILED"
) ;
```

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

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

After setting a value, an OsvvmOptionsType value can be reset using OPT_USE_DEFAULT and a string value can be reset using OSVVM_STRING_USE_DEFAULT.

29. Options for ReportAlerts

FailOnWarning Warnings are test errors.
FailOnDisabledErrors Disabled errors are test errors.
ReportHierarchy Print alert summary for all levels.
ReportPrefix. Prefix for each line of ReportAlerts.
DoneName. Printed after ReportPrefix on first line of ReportAlerts.
PassName. Printed when no alerts.
FailName.   Printed when alerts.

30. Options for Alert

WriteAlertLevel Print level.
WriteAlertName Print name.
WriteAlertTime Alerts print time.
AlertPrefix. Prefix for alert.

31. Options for Log

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

32. DeallocateAlertLogStruct

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

33. InitializeAlertLogStruct

Recreates the alert structure after it was deallocated.

34. 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 (800)-505-8435
http://www.SynthWorks.com jim@synthworks.com