OSVVM and Error Reporting

Jim Lewis SynthWorks VHDL Training

SynthWorks

accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Agenda

- OSVVM
- Motivations
- Basic Alerts
- Hierarchical Alerts
- Logs
- Affirmations
- Transcript Files
- Conclusions

Open Source VHDL Verification Methodology

- OSVVM = Free, Open Source Library + Methodology for:
  - Constrained Random (CR)
  - Functional Coverage (FC)
  - Intelligent Coverage - Random using FC holes
  - Memory Modeling (2015.06)
  - Alerts for error reporting (2015.01)
  - Logs for verbosity control (2015.01)
  - Transcripts for test wide printing (2015.01)
- Rivals other languages in capability and conciseness

Motivations

- How do we signal and count errors?
  - VHDL Assert signals error messages, but does not count
  - Often ad-hoc methods collect counts from separate sources
  - OSVVM solution: Alerts
- How do we conditionally print test (verbosity control)?
  - VHDL Assert note – turn on/off using simulator GUI
  - OSVVM solution: Logs
- How do we print all test information to a named file?
  - OSVVM solution: TranscriptFiles

Basic Alerts

- Alerts both signal and count errors

```
Alert ("Illegal State") ;
AlertIf    ( Data /= ExpectedData, "Data Error …");
AlertIfNot( ReadValid,             "Read Failed", FAILURE) ;
AlertIfDiff("./File1.txt", "./File2.txt") ;
```

  - Alert Levels: FAILURE, ERROR (default), WARNING

```
%% Alert ERROR   Illegal State at 5000 ns
```

- By default, errors are accumulated in a global error counter

Basic Alerts

- Summarizing Errors

```
ReportAlerts(Name => "tb1_basic") ;
std.env.stop ;
```

```
%% DONE  PASSED  tb1_basic  at 100000 ns
```

```
%% DONE  FAILED  tb1_basic  Total Error(s) = 2  Failures:
0  Errors: 1  Warnings: 1  at 100000 ns
```

Hierarchical Alerts

- Alert IDs allow each model to accumulate errors separately

```
Alert (CpuBaseID, "Illegal State") ;
AlertIf (CpuBaseID, Data /= ExpectedData, "Error …");
```

- Alert IDs are allocated

```
signal CpuBaseID : AlertLogIDType ;
signal DataErrID  : AlertLogIDType ;
. . .

CpuBaseID <= GetAlertLogID("Cpu_1") ;
DataErrID  <= GetAlertLogID("Cpu_1 Data Error", CpuBaseID);
. . .
```

Hierarchical Alerts

- Hierarchical Alerts report errors for all levels of the hierarchy

```
ReportAlerts (Name => "Test_UartRx_1") ;
```

```
%% DONE   PASSED    Test_UartRx_1  at 100100100 ns
```

```
%% DONE   FAILED    Test_UartRx_1  Total Error(s) = 10  Failures: 0
Errors: 10  Warnings: 0  at 100100100 ns
%%    Default                    Failures: 0  Errors: 2  Warnings: 0
%%    OSVVM                      Failures: 0  Errors: 0  Warnings: 0
%%    Cpu_1                      Failures: 0  Errors: 5  Warnings: 0
%%      Cpu_1 Data Error         Failures: 0  Errors: 4  Warnings: 0
%%      Cpu_1 Protocol Error     Failures: 0  Errors: 1  Warnings: 0
%%    UART_TX_1                  Failures: 0  Errors: 0  Warnings: 0
%%    UART_RX_1                  Failures: 0  Errors: 3  Warnings: 0
```

Stopping on Alert Counts

- Setting a Stop Count

```
SetAlertStopCount(ERROR, 20) ;  -- Any error
SetAlertStopCount(CpuBaseID, ERROR, 20) ;  -- errors in CPU
```

  - Stops the test on the 20th Alert ERROR generated
  - Default for ERROR and WARNING is 2**32-1
  - Default for FAILURE is 0

Enabling / Disabling Alerts

- Enable / Disable an Alert level

```
SetAlertEnable(WARNING, FALSE) ;
SetAlertEnable(CpuBaseID, ERROR, FALSE) ;
```

  - Alerts are enabled by default

- Global Enable/Disable

```
SetGlobalAlertEnable( TRUE ) ;
```

- Clear All Alerts

```
ClearAlerts( TRUE ) ;
```

Logs

- Logs allow output to be conditionally printed

```
Log("Test 1 Starting") ;
...
Log(CpuBaseID, "Entered Hold State", DEBUG) ;
```

  - Levels: ALWAYS, DEBUG, INFO, FINAL, PASSED
  - There is no relationship between levels (> or <)

```
%% Log    ALWAYS   Test 1 Starting at 1770 ns
%% Log    DEBUG    In Cpu_1, Entered Hold State at 31000 ns
```

Enabling / Disabling Logs

- Enable / Disable Logs

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

  - All logs, except ALWAYS, are disabled by default.
  - ALWAYS cannot be disabled.
  - Enabling one level does not enable any other level

- Enables can also be read from a file

```
ReadLogEnables("./Test1_LogEnables.txt") ;
```

```
U_CpuModel DEBUG
U_UART_TX DEBUG INFO
U_UART_RX FINAL INFO DEBUG
```

Affirmations

- Affirmations combine alerts and logs

```
AffirmIf ( CpuBaseID, Data = Expected,
                  "Data: " & to_string(Data) &
                  " = Expected: " & to_string(Expected)) ;
```

- AffirmIf uses Alert Error when it Fails

```
%% Alert ERROR   In Cpu_1, Data: 5 = Expected: 6 at …
```

- AffirmIf uses Log with level PASSED when it passes

```
%% Log   PASSED  In Cpu_1, Data: 5 = Expected: 5 at …
```

Transcript Files

- OSVVM prints to TranscriptFile if open, otherwise, OUTPUT
  - This includes, Alerts, Logs, Coverage, …
- Opening and Closing Transcripts

```
TranscriptOpen("./results/test1.txt") ;
TranscriptClose ;
```

  - Opens or closes file TranscriptFile
- Mirroring: Printing to TranscriptFile and OUTPUT

```
SetTranscriptMirror(TRUE) ;
```

Transcript Files

- Printing using Transcript Files

```
print("A String") ;      -- Direct to file, newline added
print("") ;              -- Print a blank line
writeline( WriteBuf );   -- Using textio
```

- Print & VHDL-2008

```
signal Count : integer ;
signal Data  : std_logic_vector(15 downto 0) ;
. . .
print("State = "   & to_string(State) &
      ", Count = " & to_string(Count) &
      ", Data = "  & to_hstring(Data) &
      " at time "  & to_string(NOW, 1 ns) ) ;
```

Package References

- To Use Alerts, Logs, and Transcripting:

```
Library OSVVM ;
use OSVVM.TranscriptPkg.all ;
use OSVVM.AlertLogPkg.all ;
```

- To use all of OSVVM:

```
Library OSVVM ;
Context OSVVM.OsvvmContext ;
```

Conclusions

- Alerts
  - Signal, Count, Control, and Report Error Sources
  - No need for ad-hoc approaches
- Logs
  - Conditional printing that can be turned on/off for either a level or a level in a specific model
  - Very different from the levels in SV
- Transcript Files
  - Simplified test wide printing

References

- Jim's OSVVM Blog: www.synthworks.com/blog/osvvm
- OSVVM Forums and Blog: www.osvvm.org
  - Currently there are over 1400 members
- Get the packages: www.osvvm.org/downloads
  - Includes User Manuals

Questions