Getting Started with OSVVM

VHDL's #1 Verification Methodology

by
Jim Lewis
OSVVM Chief Architect
IEEE 1076 VHDL Working Group Chair
VHDL Training Expert at SynthWorks
Jim@SynthWorks.com

Getting Started with OSVVM

This material is derived from SynthWorks' class, VHDL Testbenches and Verification

This material is updated from time to time and the latest copy of this is available at
http://www.SynthWorks.com/papers

Contact Information
Jim Lewis, President
SynthWorks Design Inc
11898 SW 128th Avenue
Tigard, Oregon 97223
503-590-4787
jim@SynthWorks.com

www.SynthWorks.com

SynthWorks

Getting Started with OSVVM

- Agenda
  - VHDL is #1 for FPGA Design and Verification
  - What is OSVVM?
  - OSVVM Verification Framework
  - Verification Components
  - Self-Checking Tests Made Easy
  - Simplifying Test Printing with OSVVM Logs
  - Constrained Random Tests
  - Scoreboards
  - Functional Coverage & Intelligent Coverage Random
  - Protocol and Parameter Checks
  - Test Watch Dog Timers
  - Test Reporting

SynthWorks

Background

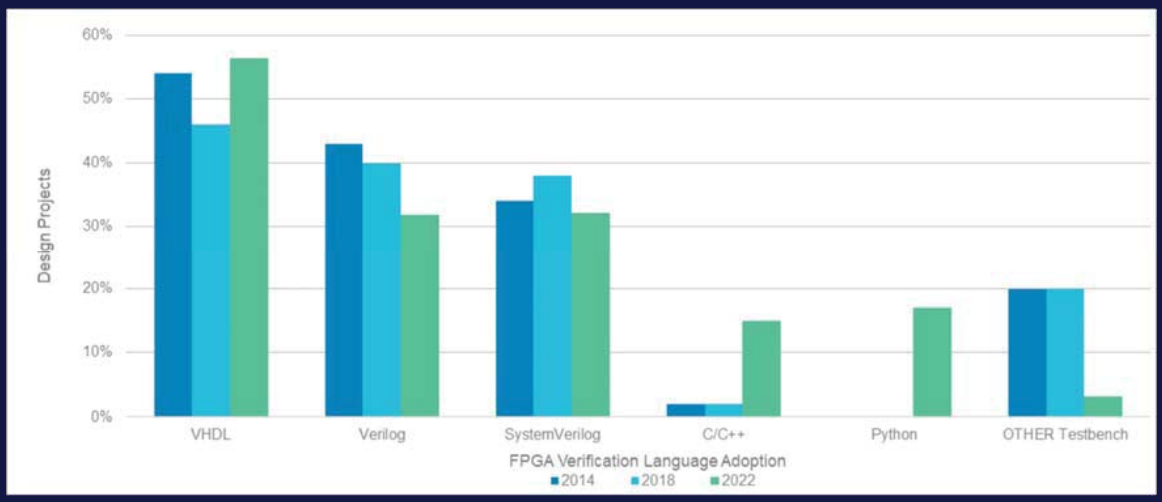
- About Jim Lewis
  - 30 years: VHDL Design and Verification
  - 20+ years: Active in IEEE VHDL WGs
  - 14 years: IEEE VHDL WG chair
  - Chief Architect of OSVVM
  - VHDL Consultant and Trainer for SynthWorks
- SynthWorks provides VHDL Training
  - Comprehensive VHDL Introduction
  - VHDL Coding for Synthesis
  - Advanced VHDL Testbenches and Verification – OSVVM Bootcamp

OSVVM is developed by the same VHDL experts who have helped develop VHDL standards. We have used our expert VHDL skills to create advanced verification capabilities.

SynthWorks

VHDL is #1 in FPGA Design & Verification

- From Wilson Research Group 2022 Functional Verification Survey
  - For FPGA design: 66% worldwide use VHDL
  - For FPGA verification: 56% worldwide use VHDL
- For FPGA VHDL is use more than Verilog / SystemVerilog

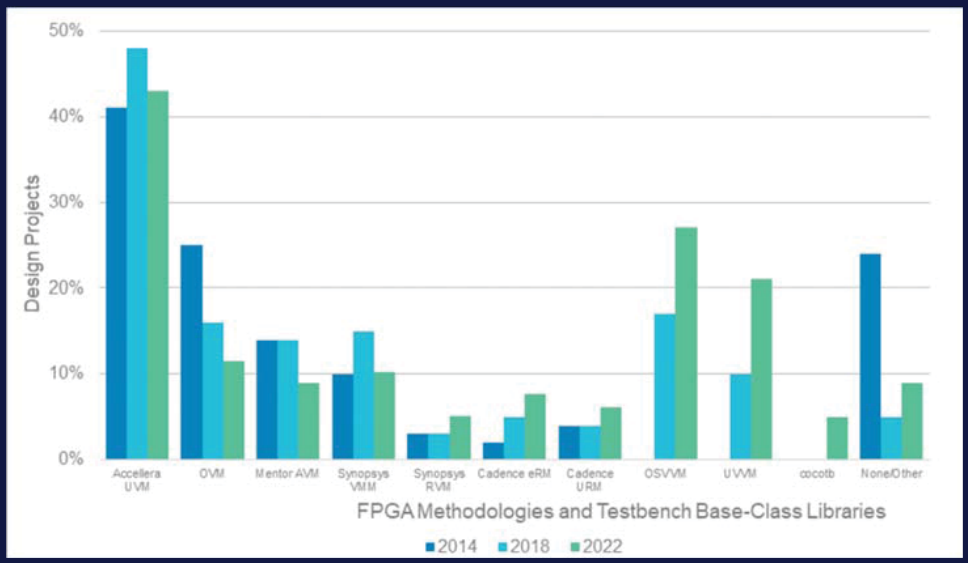


- © Siemens 2022 https://blogs.sw.siemens.com/verificationhorizons/2022/11/21/part-6-the-2022-wilson-research-group-functional-verification-study/

SynthWorks

OSVVM is VHDL's #1 Verification Methodology

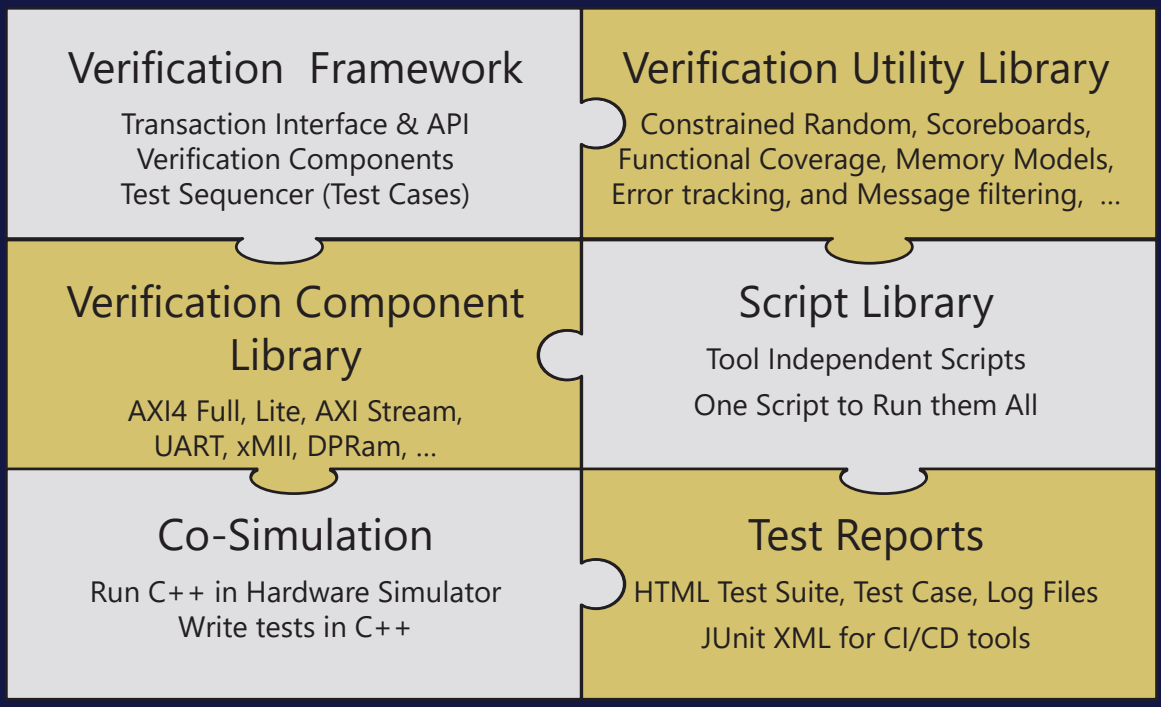
- For FPGA Verification Libraries,
  - Worldwide: 28% use OSVVM = 50% of the VHDL FPGA users



- © Siemens 2022 https://blogs.sw.siemens.com/verificationhorizons/2022/11/21/part-6-the-2022-wilson-research-group-functional-verification-study/

What is OSVVM?

SynthWorks



- Simple, readable, powerful capabilities that can be incrementally adopted.
- Unmatched reuse through the entire verification process.

OSVVM History

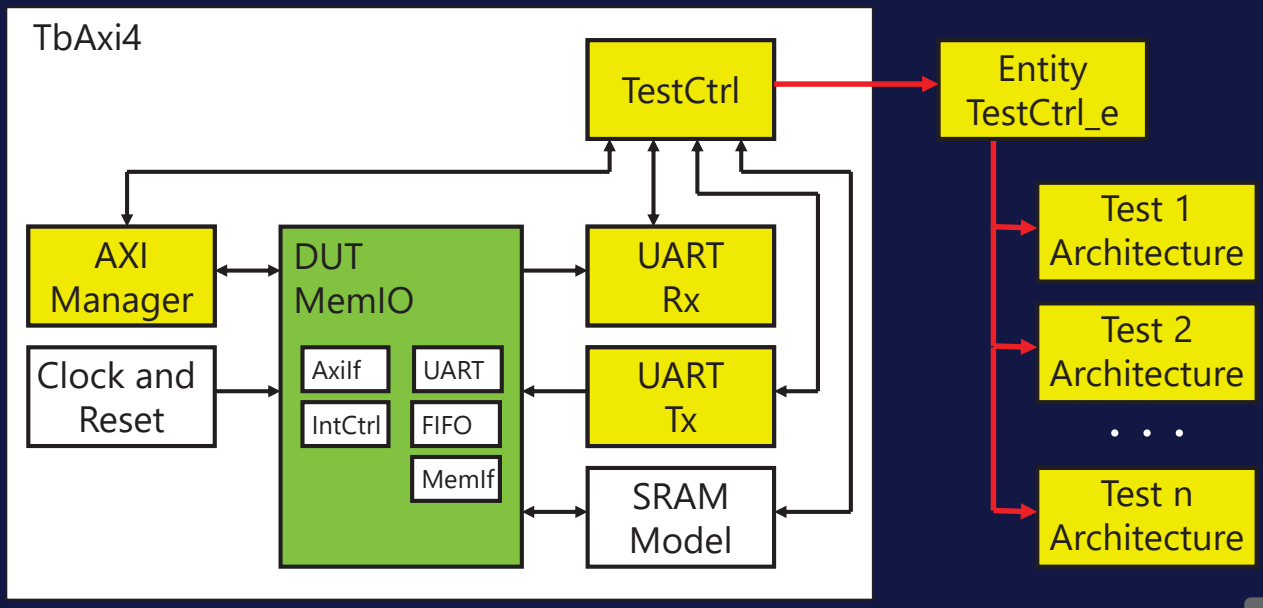
SynthWorks

SynthWorks Classes	1997	Transaction Framework, TbUtilPkg
	2006	RandomPkg, ResolutionPkg, ScoreboardPkg, MemoryPkg
	2010	CoveragePkg
OSVVM and SynthWorks Classes	2011	RandomPkg, CoveragePkg
	2015	AlertLogPkg, TranscriptPkg, MemoryPkg
	2016	ScoreboardGenericPkg, TbUtilPkg, ResolutionPkg
	2018	Axi4Lite, AxiStream, UART
	2020	Scripting, Specification Tracking, MIT, Virtual Interfaces, Axi4 Full and AxiStream, both with Bursting
	2021	Singleton Data Structures, HTML & JUnit XML reports
	2022	Ethernet VC, HTML Log & Scoreboard Reports, Arrays of Transaction Interfaces, Code Coverage Merging
	2023	Co-simulation of C++ Software in a Hardware Simulator

SynthWorks

OSVVM Verification Framework

- Looks identical to a SystemVerilog framework:
  - Verification components implement interface signaling
  - Test case = sequences of transactions in test sequencer (TestCtrl)
  - Each test case is a separate architecture of TestCtrl



SynthWorks

Verification Framework

```
library osvvm, osvvm_Axi4 ;
  context osvvm.OsvvmContext ;
. . .
entity  TbAxi4  is
end entity TbAxi4 ;
architecture TestHarness of TbAxi4 is
 . . .
 signal ManagerRec : AddressBusRecType (
    Address      (AXI_ADDR_WIDTH-1 downto 0),
    DataToModel  (AXI_DATA_WIDTH-1 downto 0),
    DataFromModel(AXI_DATA_WIDTH-1 downto 0)
  ) ;
begin
  osvvm.TbUtilPkg.CreateClock(Clk, tperiod_Clk) ;
  osvvm.TbUtilPkg.CreateReset(nReset, . . .) ;

  DUT_1: DUT ( . . . ) ;

  Axi4Manager_1 : Axi4Manager (ManagerRec, . . . ) ;
  UartRx_1      : UartRx(RxRec, . . . ) ;
  UartTx_1      : UartTx(TxRec, . . . ) ;

  TestCtrl_1    : TestCtrl (TxRec, RxRec, ManagerRec, nReset) ;
end TestHarness ;
```

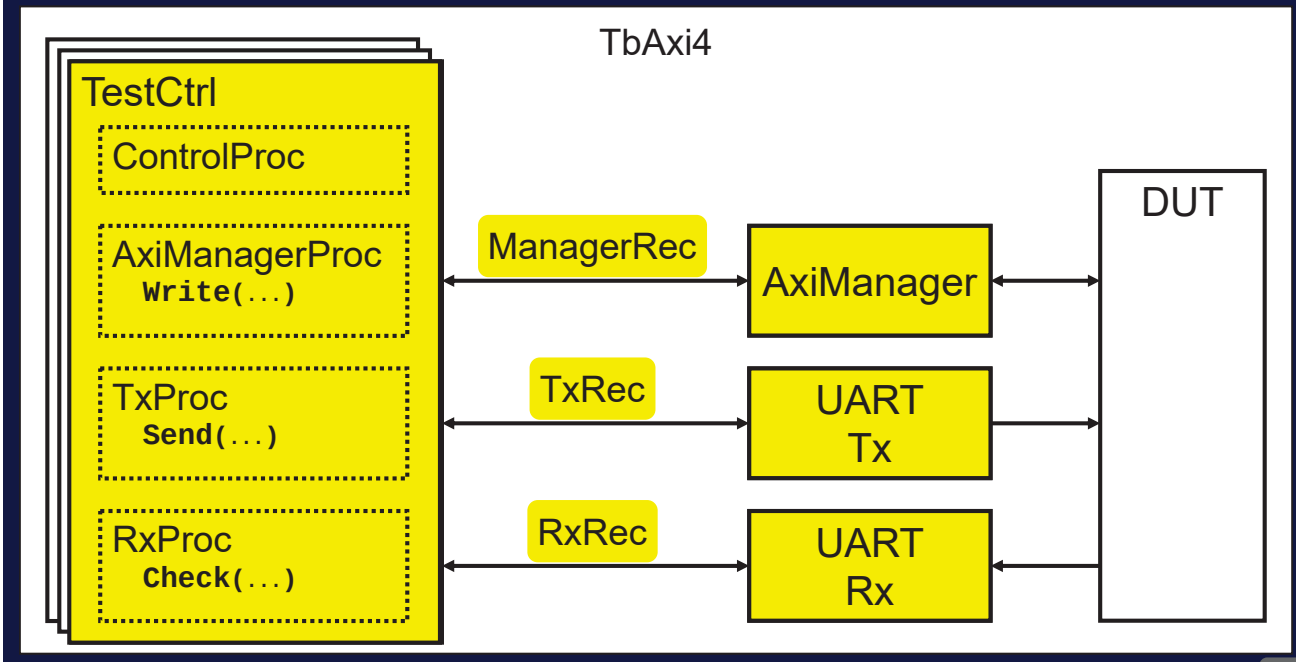
Synonyms
Structural Code
Test Harness
Netlist

Plugs together just like RTL

SynthWorks

Verification Framework – Elements

- Transaction Interface (records) + Transaction API (procedures)
- Verification components
- Test Sequencer (TestCtrl)



SynthWorks

Transaction Interface = Record

```
type AddressBusRecType is record
  Rdy             : RdyType ;
  Ack             : AckType ;
  Operation       : AddressBusOperationType ;
  Address         : std_logic_vector_max_c ;
  AddrWidth       : integer_max ;
  DataToModel     : std_logic_vector_max_c ;
  DataFromModel   : std_logic_vector_max_c ;
  DataWidth       : integer_max ;
  . . .
end record AddressBusRecType ;
```

Control / Handshaking

Transaction Information

- The record is an "inout" port
  - The "magic" is in the types and resolution functions (from ResolutionPkg)

Long term plan is to migrate to VHDL-2019 Interfaces
Interfaces replace OSVVM types with mode view declarations

Transaction API = VHDL Procedure

SynthWorks

```
procedure Read (
  signal   TransRec     : InOut AddressBusRecType ;
           iAddr        : In    std_logic_vector ;
  variable oData        : Out   std_logic_vector ;
           StatusMsgOn  : In    boolean := FALSE
) is
begin
  -- Put Transaction into Record
  TransRec.Operation     <= READ_OP ;
  TransRec.Address       <= SafeResize(iAddr, TransRec.Address'length) ;
  TransRec.AddrWidth     <= iAddr'length ;
  TransRec.DataWidth     <= oData'length ;
  TransRec.StatusMsgOn   <= StatusMsgOn ;

  -- Handshake with Verification Component
  RequestTransaction(Rdy => TransRec.Rdy, Ack => TransRec.Ack) ;

  -- Get Results
  oData := SafeResize(TransRec.DataFromModel, oData'length) ;
end procedure Read ;
```

Basic Flow

- Put information for VC into record
- Handshake with VC
- Get results from VC from record

OSVVM Model Independent Transactions

SynthWorks

- Codifies the Transaction Interface and Transaction API for
  - Address Bus Interfaces (AXI4, Avalon, …)
  - Streaming Interfaces (AxiStream, UART, …)
- Address Bus Model Independent Transactions (MIT) – basic capability:

```
type AddressBusRecType is record  . . . ;

Write(     AddrRec, iAddr, iData [, StatusMsgOn]) ;
Read (     AddrRec, X"1111_1110", oData) ;
ReadCheck( AddrRec, X"AAAA_AAA0", X"55") ;
```

- Stream Model Independent Transactions (MIT) – basic capability:

```
type StreamRecType is record  . . . ;

Send (TxRec, iData [, iParam]) ;
Get  (RxRec, oData [, oParam]) ;
Check(RxRec, iData [, iParam]) ;
```

Benefit: Simplifies VC development and facilitates reuse of test cases / sequences

Verification Components

SynthWorks

```
entity Axi4Manager is
generic (
  tperiod_Clk     : time := 10 ns ;
  . . .
  tpd_Clk_RReady  : time := 2 ns
) ;
port (
  -- Globals
  Clk          : in   std_logic ;
  nReset       : in   std_logic ;

  -- AXI Master Functional Interface
  AxiBus       : inout Axi4RecType ;

  -- Testbench Transaction Interface
  TransRec     : inout AddressBusRecType
) ;
```

DUT Interface

Transaction Interface Record type defined by OSVVM MIT

Verification Components

SynthWorks

```
TransactionHandler : process
begin
  WaitForTransaction(
    Clk => Clk,
    Rdy => TransRec.Rdy,
    Ack => TransRec.Ack
  ) ;

  -- Decode and execute the transaction
  case TransRec.Operation is
    when WRITE_OP =>
      AxiWrite(TransRec.Address, TransRec.Data, …);
    when READ_OP =>
      AxiRead (TransRec.Address, TransRec.Data, …);
    when . . . =>
      -- Other Transactions
  end case ;
end process TransactionHandler ;
```

Find Transaction in Record

Do the Transaction

Use of OSVVM MIT allows VC developer to just write the functionality

TestCtrl = Test Sequencer

SynthWorks

```
entity TestCtrl is
port (
  TxRec         : InOut StreamRecType ;
  RxRec         : InOut StreamRecType ;

  ManagerRec    : InOut AddressBusRecType;

  nReset        : In    std_logic
) ;
end TestCtrl ;
```

Transaction Interface Record types defined by OSVVM MIT

```
architecture UartTx1 of TestCtrl is
  . . .
begin
  ControlProc : process
  begin
    . . .
    WaitForBarrier(TestDone, 5 ms) ;
    EndOfTestReports ;
    std.env.stop;
  end process ;

  AxiManagerProc : process
  begin
    wait until nReset = '1' ;
    Write(. . .) ;
    WaitForBarrier(CpuRdy);
    . . .
    WaitForBarrier(TestDone) ;
  end process ;

  TxProc : process
  begin
    WaitForBarrier(CpuRdy);
    Send(. . .) ;
    . . .
    WaitForBarrier(TestDone) ;
  end process;
  . . .
```

Aspects of a Test Sequencer

- Whole test in one file
- Control Process
  - Initialize & finalize test
- One process per interface
  - Concurrent, just like design
- Tests = calls to transactions
  - Easy to write and read
- Easy to add
  - Directed Tests
  - Functional Coverage
  - Constrained Random
  - Mix of test approaches
- Synchronization
- Error Reporting & Messaging

Increased readability since similar VC use the same transaction API calls

Test Initialization in ControlProc

SynthWorks

```
ControlProc : process
begin
  SetTestName("UartRx1") ;

  TranscriptOpen("UartRx1.log") ;
  SetTranscriptMirror(TRUE) ;

  TBID <= NewID("TB") ;
  RxID <= NewID("UartRx_1") ;
  SB   <= NewID("SB", ModelID) ;

  SetLogEnable(PASSED, TRUE) ;
  SetLogEnable(RxID, INFO, TRUE) ;

  WaitForBarrier(TestDone, 5 ms) ;

  . . . -- Test Finalization
```

Set Test Name

Open Results File
+ Write to Console

Create AlertLog and
Scoreboard IDs

Enable Logs
Message Filtering

Stop until Test Done
or 5 ms has passed
= WatchDog timer

OSVVM Makes Self-Checking Tests Easy

SynthWorks

- Send, Get, and Check transactions are used to implement test cases

```
TxProc : process

begin
  Send (TxRec, X"10") ;

  Send (TxRec, X"11") ;

  . . .
  WaitForBarrier(...) ;
end process TxProc ;
```

```
RxProc : process
  variable RxD : ByteType;
begin
  Get(RxRec, RxD) ;
  AffirmIfEqual(TBID, RxD, X"10");

  Check(RxRec, X"11");

  . . .
  WaitForBarrier(TestDone);
end process RxProc ;
```

- Test Output for AffirmIfEqual

```
%% Alert ERROR  In TB, Received: 08 /= Expected: 10 at 2150 ns
```

```
%% Log   PASSED In TB, Received: 10 at 2150 ns
```

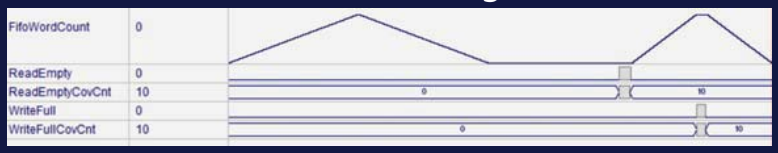
  - Check output will be similar, except it will say "In UartRx_1"

Can reuse test cases/sequences since similar VC use the same API

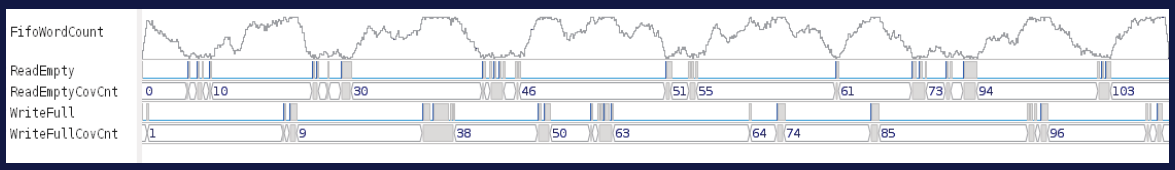
SynthWorks

OSVVM Makes Randomization Easy

- Why Randomize?
  - Directed test of a FIFO (tracking words in FIFO):



  - Constrained Random test of a FIFO:



- Key Benefits:
  - Generates realistic stimulus in a timely fashion (to write)
  - Ideal for large variety of similar items
    - Modes, sequences, network packets, processor instructions, …

SynthWorks

OSVVM Makes Randomization Easy

- Randomize a value in an inclusive range, 0 to 15, except 5 & 11

```
Data1 := RV.RandInt(Min => 0, Max => 15) ;
Data2 := RV.RandInt(0, 15, Exclude => (5,11) ) ;
```

- Randomize a value within the set (1, 2, 3, 5, 7, 11), except 5 & 11

```
Data3 := RV.RandInt( (1,2,3,5,7,11) ) ;
Data4 := RV.RandInt( (1,2,3,5,7,11), Exclude => (5,11) ) ;
```

- Weighted Randomization: Weight, Value = 0 .. N-1

```
Data5 := RV.DistInt ( (7, 2, 1) ) ;
```

- Weighted Randomization: Value + Weight

```
. . .                    -- ((val1, wt1), (val2, wt2), ...)
Data6 := RV.DistValInt( ((1,7), (3,2), (5, 1)) ) ;
```

By itself, this is not constrained random

OSVVM Constrained Random Tests

SynthWorks

```
TxProc : process
  variable RV : RandomPType ;
  . . .
for I in 1 to 10000 loop
  case RV.DistInt( (70, 10, 10, 5, 5) ) is
    when 0  =>    -- Nominal case   70%
      Operation := UARTTB_NO_ERROR ;
      TxD:= RV.RandSlv(0, 255, Data'length) ;

    when 1  =>    -- Parity Error   10%

    when 2  =>    -- Stop Error   10%
      Operation := UARTTB_STOP_ERROR ;
      TxD:= RV.RandSlv(1, 255, Data'length) ;

    when  . . .  -- (3 and 4)
  end case ;

  Send(TxRec, TxD, Operation) ;
end loop ;
 . . .
```

Randomize Operation

Nominal 70%

Stop Error 10%

Do Transaction

Constrained Random = Randomization + Code + Transaction Calls

Constrained Random and Checking?

SynthWorks

For checking, RxProc could repeat the randomization from TxProc, however, this is tedious and potentially error prone.

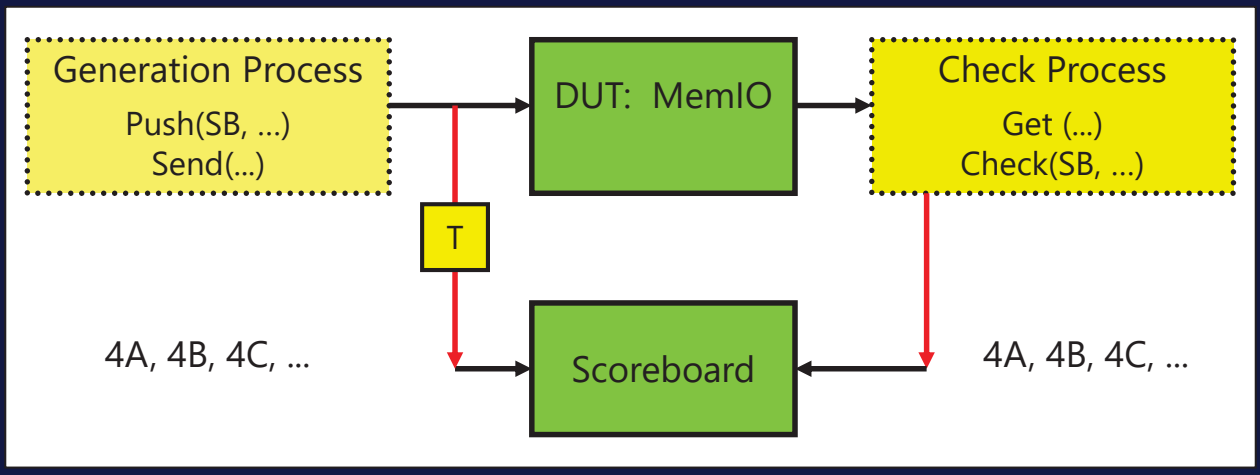
```
TxProc : process
 variable TxD : ByteType;
 variable RV  : RandomPType;
begin
  for I in 1 to 10000 loop
    case RV.DistInt((. . .)) is
      . . .
    end case ;

    Send(TxRec, TxD, Op);
  end loop ;

  . . .

  WaitForBarrier(TestDone);
end process TxProc ;
```

```
RxProc : process
  variable ExpD : ByteType;
  variable RV   : RandomPType;
begin
  for I in 1 to 10000 loop
    case RV.DistInt((. . .)) is
      . . .
    end case ;

    Check(TxRec, ExpD, Op);
  end loop ;

  . . .

  WaitForBarrier(TestDone);
end process RxProc ;
```

Scoreboards

SynthWorks

- Simplify self-checking when data is minimally transformed



- FIFO + Checker
- Uses package generics to support different types
- Handles small data transformations
- Handles out of order execution
- Handles dropped values

OSVVM Generic Scoreboards

SynthWorks

```
package ScoreBoardGenericPkg is
  generic(
    type ExpectedType ;
    type ActualType ;
    function Match( . . . ) return boolean ;
    function expected_to_string( . . . ) return string ;
    function actual_to_string  ( . . . ) return string
  ) ;
  type ScoreboardIDType is ... ;
  procedure NewID (…) ;
  procedure Push (…) ;
  procedure Check (…) ;
  procedure Pop (…) ;
  impure function Pop (…) return ExpectedType ;
  impure function Empty (…) return boolean ;
  . . .

  type ScoreBoardPType is protected
    . . .
  end protected ScoreBoardPType ;
end ScoreBoardGenericPkg ;
```

Parameterized with Generics

Scoreboard / FIFO API

OSVVM Scoreboards: Generic Instance

SynthWorks

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

package ScoreBoardPkg_slv is new osvvm.ScoreBoardGenericPkg
  generic map (
    ExpectedType        =>  std_logic_vector,
    ActualType          =>  std_logic_vector,
    match               =>  MetaMatch,
    expected_to_string  =>  to_hxstring,
    actual_to_string    =>  to_hxstring
  ) ;
```

```
package ScoreBoardPkg_int is new osvvm.ScoreBoardGenericPkg
  generic map (
    ExpectedType        =>  integer,
    ActualType          =>  integer,
    match               =>  "=",
    expected_to_string  =>  to_string,
    actual_to_string    =>  to_string
  ) ;
```

Both in OSVVM Library

Using OSVVM's Scoreboard is Easy

SynthWorks

```
use osvvm.ScoreboardPkg_slv.all ;
signal SB : ScoreboardIDType ;
. . .
SB <= NewID("SB", ModelID) ;  -- Constructor in ControlProc
```

```
TxProc : process
  . . .
begin
  for I in 1 to 10000 loop
    case RV.DistInt((. . .)) is
      . . .
    end case ;
    Push(SB,(TxD, Op));
    Send(TxRec, TxD, Op);
  end loop ;
  . . .
```

```
RxProc : process
  variable RxD  : ByteType;
  variable RV   : RandomPType;
begin
  for I in 1 to 10000 loop

    Get(RxRec, RxD, RxOp);
    Check(SB,(RxD, RxOp));
  end loop ;
  . . .
  WaitForBarrier(TestDone);
```

OSVVM Scoreboards, FIFOs, Functional Coverage, and Alerts use singletons

- Singletons use ordinary types and constructors (NewID)
- Easier than our older methodology which uses protected types.

Functional Coverage

SynthWorks

- What: Code that tracks that items in the test plan occur
  - Tracks requirements, features, and boundary conditions
- Why?
  - With Randomization, how do you know what the test did?
  - Test Done = Functional Coverage and Code Coverage @ 100 %
- Item Coverage (aka Point Coverage)
  - Track relationships within a single object
  - Bin transfer sizes into: 1, 2, 3, 4-127, 128-252, 253, 254, 255
- Cross Coverage
  - Track relationships between multiple objects
  - Has each set of registers been used with each input of an ALU?
- Why not just use code coverage?
  - Code coverage tracks code execution
  - Misses anything not in code (bins, uncorrelated items)

CoveragePkg

SynthWorks

- CoveragePkg simplifies coverage definition, collection, and reporting
  - Internally it has a data structure and configuration parameters
  - Implemented as a singleton in CoveragePkg
  - The singleton API defines the coverage capabilities

```
function GenBin ( . . . ) return CovBinType ;

type CoverageIDType is . . . ;
impure function NewID(Name : string; ...)
  return CoverageIDType ;

procedure AddBins (ID : CoverageIDType; CovBin : CovBinType ) ;
procedure AddCross(ID : CoverageIDType;  Bin1, Bin2, ... : CovBinType );

procedure ICover (ID : CoverageIDType; val : integer ) ;
procedure ICover (ID : CoverageIDType; val : integer_vector ) ;

impure function IsCovered (ID : CoverageIDType) return boolean ;

procedure WriteBin        (ID : CoverageIDType) ;
procedure WriteCovHoles   (ID : CoverageIDType) ;
  . . .
```

OSVVM Functional Coverage is Easy

SynthWorks

- For the UART, we track the following items

Condition	Status Register Values: Break Error	Stop Error	Parity Error	Done Flag	Integer Value(s)
Normal Transfer	0	0	0	1	1
Parity Error	0	0	1	1	3
Stop Error	0	1	0	1	5
Parity & Stop Error	0	1	1	1	7
Break Error	1	-	-	1	9-15

OSVVM Functional Coverage is Easy

SynthWorks

```
architecture CR_1 of TestCtrl is
  signal RxCov : CoverageIdType ;
```

Coverage Object

```
RxProc : process
  . . .
begin
  RxCov <= NewID("RxCov", TB_ID) ;
  wait for 0 ns ;

  AddBins(RxCov, GenBin(1) ) ;          -- Normal
  AddBins(RxCov, GenBin(3) ) ;          -- Parity Error
  AddBins(RxCov, GenBin(5) ) ;          -- Stop Error
  AddBins(RxCov, GenBin(7) ) ;          -- Parity + Stop
  AddBins(RxCov, GenBin(9, 15, 1) ) ; -- Break

  for I in 1 to 10000 loop
    Get(RxRec, RxD, RxOp);
    Check(SB,(RxD, RxOp));

    ICover(RxCov, RxOp ) ;
  end loop ;
  . . .
  WriteBin(RxCov) ;
```

Initialize Data Structure

Define coverage model

Collect Coverage

Functional Coverage with OSVVM is as simple and concise as language syntax.

OSVVM Intelligent Coverage Randomization

SynthWorks

Intelligent Coverage = Randomize using functional coverage holes

```
TxProc : process
  variable StimCov : CoverageIdType ;
begin
  StimCov :=  NewID("StimCov", TB_ID) ;
  wait for 0 ns ;
  AddBins(StimCov, "NORMAL", 7000,  GenBin(1) ) ;
  AddBins(StimCov, "PARITY", 1100,  GenBin(3) ) ;
  AddBins(StimCov, "STOP",   1100,  GenBin(5) ) ;
  . . .
  for I in 1 to 10000 loop
    iOperation := GetRandPoint(StimCov) ;
    case iOperation is
      when 1  =>    . . .  -- Nominal 70%
      when 3  =>    . . .  -- Parity  11%
      . . .
    end case ;
    Push(SB, (Data, Operation) ) ;
    Send(TxRec, Data, Operation) ;
    ICover(StimCov, iOperation ) ;
    wait for Idle * UART_BAUD_PERIOD_115200 ;
  end loop ;
```

Coverage Object

Constructor

Coverage Goals = Randomization Weights

Randomize

Similar actions to constrained random

Do transaction & Collect coverage

OSVVM Protocol and Parameter Checkers

SynthWorks

- Protocol Checks (in a Memory Model)

```
SimultaneousAccessCheck: process
begin
  wait on iCE, iWE, iOE ;
  AlertIf(SramAlertID, (iCE and iWE and iOE) = '1',
      "nCE, nWE, and nOE are all active") ;
end process SimultaneousAccessCheck ;
```

- Parameter Checks (in OSVVM packages)

```
AlertIf (OSVVMID, Max < Min, "RandInt: Max < Min") ;
```

- Controls: Enable/Disable, StopCount, PrintCount

```
SetAlertEnable(WARNING, FALSE) ;          -- Disable Alerts
SetAlertStopCount(ERROR, 20) ;            -- Stop when 20
SetAlertPrintCount(CpuID, ERROR, 10) ;    -- Limit printing
```

OSVVM Logs Simplify Test Printing

- Logs are for conditional test printing

```
TxProc : process
begin
   . . .
  Log(TbID, "Sequence 1 Starting", ALWAYS) ;
   . . .
  Log(TbID, "Test Last Failed Here", DEBUG) ;  -- Disabled
```

  - Log Levels: ALWAYS (default), DEBUG, INFO, FINAL, PASSED
  - Logs only print when enabled.

- Controls: Enable/Disable

```
SetLogEnable(DEBUG, FALSE) ;        -- Disable Alerts
```

- Log output for above

```
%% Log   ALWAYS In TB, Sequence 1 Starting at 2200 ns
```

  - Message with level DEBUG does not print since it is disabled

Test Finalization


```
ControlProc : process
begin
  SetAlertLogName("UartRx1") ;

  . . . – Test Initialization

  WaitForBarrier(TestDone, 5 ms) ;

  AlertIf(TBID, NOW >= 5 ms, "Test timed out") ;

  AlertIf(TBID, not Empty(SB), "Scoreboard not empty") ;

  AlertIf(TBID, GetAffirmCount < 1, "Checked < 1 items") ;

  AffirmIfNotDiff("UartRx1.log", "Checked/UartRx1.log") ;

  EndOfTestReports ;
  std.env.stop ;
  wait ;
end process ControlProc
```

Stop until Test Done or 5 ms has passed

Create Reports

SynthWorks

OSVVM Test Watch Dog

- Purpose: Stop a process until all processes have reached the barrier

```
signal TestDone : integer_barrier := 1;
```

```
ControlProc : process
begin
  SetAlertLogName("UartRx1") ;
  . . .
  WaitForBarrier(TestDone,5 ms);
  . .
  ReportAlerts ;
  std.env.stop(GetAlertCount) ;
end process ControlProc ;
```

```
TestProc1 : process
 . . .
 WaitForBarrier(TestDone);
 wait ;
```

```
TestProc2 : process
 . . .
 WaitForBarrier(TestDone);
 wait ;
```

- Benefit
  - With "TestDone", simulator scripts do not need to know run length
  - The 5 ms is a time out – aka watch dog timer on the test

SynthWorks

OSVVM Test Wide Reporting

- EndOfTestReports produces a summary and if errors a detailed report

```
EndOfTestReports ;
```

```
%% DONE  PASSED  Test_UartRx_1  Passed: 48  Affirmations Checked: 48
at 100100100 ns
```

```
%% DONE  FAILED   Test_UartRx_1  Total Error(s) = 7  Failures: 0  Errors: 7
Warnings: 0  Passed: 41  Affirmations Checked: 48  at 100100100 ns
%%    Default                  Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    OSVVM                    Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    TB                       Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    UART_SB                  Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    AxiMaster_1              Failures: 0  Errors: 7  Warnings: 0  Passed: 0
%%      AxiMaster_1 Data Err   Failures: 0  Errors: 7  Warnings: 0  Passed: 41
%%      AxiMaster_1 Protocol   Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    UartRx_1                 Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    UartTx_1                 Failures: 0  Errors: 0  Warnings: 0  Passed: 0
```

EndOfTestReports is also essential to creating OSVVM's
- Build Summary and
- Test Case Reports

Including the OSVVM Library is Easy

SynthWorks

- OSVVM Includes numerous packages.
  - To simplify this, OSVVM library provides context declarations

```
-- OSVVM Utility Library, Random, Coverage, …
library osvvm ;
  context osvvm.OsvvmContext ;          -- All OSVVM packages

-- AXI4 Model Library
library osvvm_axi4 ;
  context osvvm_axi4.Axi4Context ;      -- AXI4 Full VC
  context osvvm_axi4.Axi4LiteContext ;  -- AXI4 Lite VC
  context osvvm_axi4.AxiStreamContext ; -- AXI Stream VC

-- UART Model Library
Library osvvm_uart ;
  context osvvm_uart.UartContext ;      -- UART VC

-- DPRAM Model Library
Library osvvm_dpram ;
  context osvvm_dpram.DpRamContext ;    -- DpRam VC
```


My Scripts Before OSVVM

SynthWorks

```
set DIR_SRC [file dirname [status file] ]
```

Source Location

```
set LIB_NAME osvvm_TbUart
if {![file isdirectory ./VHDL_LIBS/${LIB_NAME}} {
  vlib  ./VHDL_LIBS/${LIB_NAME}
}
```

Create Libraries & Map to them

```
vcom -2008 -work $LIB_NAME ${DIR_SRC}/TestCtrl_e.vhd
vcom -2008 -work $LIB_NAME ${DIR_SRC}/TbUart.vhd
vcom -2008 -work $LIB_NAME ${DIR_SRC}/TbUart_SendGet1.vhd
```

Compile

```
vsim –lib ${LIB_NAME} TbUart_SendGet1
add wave -r /TbLedFlasher/*
run 40 us
```

Simulate, add waves, & run

- Issues
  - Need help (TCL code) to find the source directory
  - Simulator Specific
  - Simulator API repeats the same information on many calls

Why is EDA Scripting Hard?

SynthWorks

- Some blame TCL
- Issues
  - Simulator needs to run in a specific directory
    - Settings and Library information are in a *.ini or *.cfg
      - If not, the library info must be respecified on tool start
    - Hence, if you use "cd", you loose this information
  - Scripts need to be co-located with verification IP
    - Hence, they need directory information
  - The simulator API fundamentally misunderstands the VHDL work library
    - Work is not a name for a library
    - Work is the shorthand for the current library

OSVVM Scripting

SynthWorks

```
library  osvvm_TbUart

analyze  ./testbench/TestCtrl_e.vhd
analyze  ./testbench/TbUart.vhd
analyze  ./testbench/TbUart_SendGet1.vhd

simulate TbUart_SendGet1
```

- Benefits
  - Simple, just like a list of source files …
    - … Except it is an API running on top of TCL
    - Get the power of TCL without the ugly parts.
  - Library is set and remembered by following commands
  - Paths are relative to the directory from which the scripts run
  - Supports GHDL, NVC, Aldec, Siemens, Synopsys VCS, Cadence Xcelium
- Run the Scripts using:

```
build ../OsvvmLibraries/OsvvmLibraries.pro
build ../OsvvmLibraries/RunDemoTestsWithCoverage.pro
```

Reports in the Simulation Transcript

- Each test that uses EndOfTestReports produces a PASSED/FAILED report

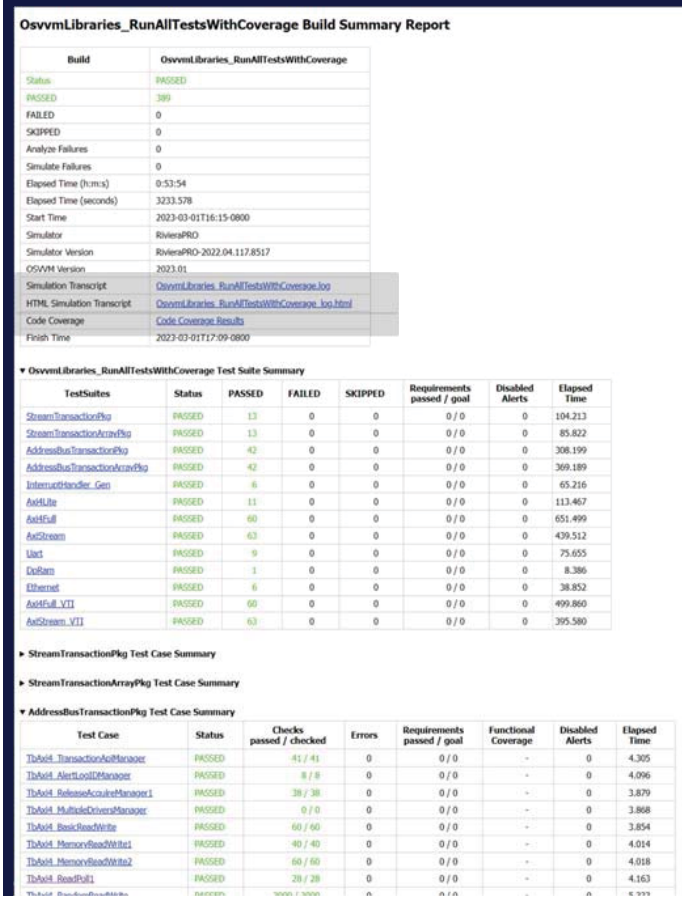
```
%% DONE  PASSED  Test_UartRx_1  Passed: 48  Affirmations Checked: 48
at 100100100 ns
```

```
%% DONE  FAILED   Test_UartRx_1  Total Error(s) = 7  Failures: 0  Errors: 7
Warnings: 0  Passed: 41  Affirmations Checked: 48  at 100100100 ns
%%    Default                  Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    OSVVM                    Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    TB                       Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    UART_SB                  Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    AxiMaster_1              Failures: 0  Errors: 7  Warnings: 0  Passed: 0
%%      AxiMaster_1 Data Err   Failures: 0  Errors: 7  Warnings: 0  Passed: 41
%%      AxiMaster_1 Protocol   Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    UartRx_1                 Failures: 0  Errors: 0  Warnings: 0  Passed: 0
%%    UartTx_1                 Failures: 0  Errors: 0  Warnings: 0  Passed: 0
```

- When build finishes, a single line, mini report is produced

```
Build: OSVVM_RunAllTests PASSED,  Passed: 389,  Failed: 0,
Skipped: 0,  Analyze Errors: 0,  Simulate Errors: 0
```


Build Summary Report



OsvvmLibraries_RunAllTestsWithCoverage Build Summary Report

Build	OsvvmLibraries_RunAllTestsWithCoverage
Status	PASSED
PASSED	389
FAILED	0
SKIPPED	0
Analyze Failures	0
Simulate Failures	0
Elapsed Time (h:m:s)	0:53:54
Elapsed Time (seconds)	3233.578
Start Time	2023-03-01T16:15:0800
Simulator	RivieraPRO
Simulator Version	RivieraPRO-2022.04.117.8517
OSVVM Version	2023.01
Simulation Transcript	OsvvmLibraries_RunAllTestsWithCoverage.log
HTML Simulation Transcript	OsvvmLibraries_RunAllTestsWithCoverage_log.html
Code Coverage	Code Coverage Results
Finish Time	2023-03-01T17:09-0800

▾ OsvvmLibraries_RunAllTestsWithCoverage Test Suite Summary

TestSuites	Status	PASSED	FAILED	SKIPPED	Requirements passed / goal	Disabled Alerts	Elapsed Time
StreamTransactionPkg	PASSED	13	0	0	0 / 0	0	104.213
StreamTransactionArrayPkg	PASSED	13	0	0	0 / 0	0	85.822
AddressBusTransactionPkg	PASSED	42	0	0	0 / 0	0	308.199
AddressBusTransactionArrayPkg	PASSED	42	0	0	0 / 0	0	369.189
InterruptHandler_Gen	PASSED	6	0	0	0 / 0	0	65.216
Axi4Lite	PASSED	11	0	0	0 / 0	0	113.467
Axi4Full	PASSED	60	0	0	0 / 0	0	651.499
AxiStream	PASSED	63	0	0	0 / 0	0	439.512
Uart	PASSED	9	0	0	0 / 0	0	75.655
DpRam	PASSED	1	0	0	0 / 0	0	8.386
Ethernet	PASSED	6	0	0	0 / 0	0	38.852
Axi4Full_VTI	PASSED	60	0	0	0 / 0	0	499.860
AxiStream_VTI	PASSED	63	0	0	0 / 0	0	395.580

▸ StreamTransactionPkg Test Case Summary

▸ StreamTransactionArrayPkg Test Case Summary

▾ AddressBusTransactionPkg Test Case Summary

Test Case	Status	Checks passed / checked	Errors	Requirements passed / goal	Functional Coverage	Disabled Alerts	Elapsed Time
TbAxi4_TransactionApiManager	PASSED	41 / 41	0	0 / 0	-	0	4.305
TbAxi4_AlertLogIDManager	PASSED	8 / 8	0	0 / 0	-	0	4.096
TbAxi4_ReleaseAcquireManager1	PASSED	38 / 38	0	0 / 0	-	0	3.879
TbAxi4_MultipleDriversManager	PASSED	0 / 0	0	0 / 0	-	0	3.868
TbAxi4_BasicReadWrite	PASSED	60 / 60	0	0 / 0	-	0	3.854
TbAxi4_MemoryReadWrite1	PASSED	40 / 40	0	0 / 0	-	0	4.014
TbAxi4_MemoryReadWrite2	PASSED	60 / 60	0	0 / 0	-	0	4.018
TbAxi4_ReadPoll1	PASSED	28 / 28	0	0 / 0	-	0	4.163

Created by OSVVM Scripts + EndOfTestReports

Build Status

- Build Pass / Fail
- Link to Simulation Transcript
  - Both text and html
- Link to code coverage

Test Suite Summary

Test Case Summaries

One for each test suite
Links to Test Case Reports
- Each test case is a separate testbench / test architecure

Test Case Report

SynthWorks

TbAxi4_MemoryReadWriteDemo1 Test Case Detailed Report

Available Reports
Alert Report
Functional Coverage Report(s)
ScoreboardPkg_slv Report(s)
Link to Simulation Results
TbAxi4_MemoryReadWriteDemo1.txt
OsvvmLibraries_RunAllTestsWithCoverage Build Summary

Links

Alert Report
Functional Coverage Report
Scoreboard Reports
Simulation Results
Test Case Transcript
Link to Build Summary

TbAxi4_MemoryReadWriteDemo1 Alert Report

▸ TbAxi4_MemoryReadWriteDemo1 Alert Settings

▸ TbAxi4_MemoryReadWriteDemo1 Alert Results

Alert Report

Settings (hidden)
Results (hidden)

TbAxi4_MemoryReadWriteDemo1 Coverage Report

Total Coverage: 43.75

▸ Cov1 Coverage Model Coverage: 37.5
▸ Cov2 Coverage Model Coverage: 37.5
▸ Cov1b Coverage Model Coverage: 50.0
▸ Cov2b Coverage Model Coverage: 50.0

Functional Coverage Report

Report for each FC model in testbench (each hidden)

TbAxi4_MemoryReadWriteDemo1 Scoreboard Report for Scoreboard_slv

Name	ParentName	ItemCount	ErrorCount	ItemsChecked	ItemsPopped	ItemsDropped	FifoCount
WriteAddressFIFO	memory_1	40	0	0	40	0	0
WriteDataFifo	memory_1	150	0	0	150	0	0
WriteResponseFifo	memory_1	40	0	0	40	0	0
ReadAddressFifo	memory_1	40	0	0	40	0	0
ReadDataFifo	memory_1	150	0	0	150	0	0
WriteResponse Scoreboard	manager_1	40	0	40	40	0	0
ReadResponse Scoreboard	manager_1	150	0	150	150	0	0
WriteAddressFIFO	manager_1	40	0	0	40	0	0
WriteDataFifo	manager_1	150	0	0	150	0	0
ReadAddressFifo	manager_1	40	0	0	40	0	0

Scoreboard Report

One Table for each Scoreboard type.
One row in table for each scoreboard.

Alert Report part of Test Case Report

SynthWorks

TbAxi4_MemoryReadWriteDemo1 Alert Report

▾ TbAxi4_MemoryReadWriteDemo1 Alert Settings

Setting		Value	Description
FailOnWarning		true	If true, warnings are a test error
FailOnDisabledErrors		true	If true, Disabled Alert Counts are a test error
FailOnRequirementErrors		true	If true, Requirements Errors are a test error
External	Failures	0	Added to Alert Counts in determine total errors
	Errors	0	
	Warnings	0	
Expected	Failures	0	Subtracted from Alert Counts in determine total errors
	Errors	0	
	Warnings	0	

Alert Settings

Alert Report

▾ TbAxi4_MemoryReadWriteDemo1 Alert Results

Name	Status	Checks		Total Errors	Alert Counts			Requirements		Disabled Alert Counts		
		Passed	Total		Failures	Errors	Warnings	Passed	Checked	Failures	Errors	Warnings
TbAxi4_MemoryReadWriteDemo1	PASSED	334	334	0	0	0	0	0	0	0	0	0
Default	PASSED	20	20	0	0	0	0	0	0	0	0	0
OSVVM	PASSED	0	0	0	0	0	0	0	0	0	0	0
:tbaxi4memory:memory_1::memory	PASSED	0	0	0	0	0	0	0	0	0	0	0
Cov1	PASSED	0	0	0	0	0	0	0	0	0	0	0
Cov2	PASSED	0	0	0	0	0	0	0	0	0	0	0
Cov1b	PASSED	0	0	0	0	0	0	0	0	0	0	0
Cov2b	PASSED	0	0	0	0	0	0	0	0	0	0	0
memory_1	PASSED	0	0	0	0	0	0	0	0	0	0	0
No response	PASSED	0	0	0	0	0	0	0	0	0	0	0
Data Check	PASSED	0	0	0	0	0	0	0	0	0	0	0
WriteAddressFIFO	PASSED	0	0	0	0	0	0	0	0	0	0	0
WriteDataFifo	PASSED	0	0	0	0	0	0	0	0	0	0	0
WriteResponseFifo	PASSED	0	0	0	0	0	0	0	0	0	0	0
ReadAddressFifo	PASSED	0	0	0	0	0	0	0	0	0	0	0
ReadDataFifo	PASSED	0	0	0	0	0	0	0	0	0	0	0

Functional Coverage Report

Uart7_Random_part3 Coverage Report

Total Coverage: 100.00

▼ UART_RX_STIM_COV Coverage Model **Coverage: 100.0**

▼ UART_RX_STIM_COV Coverage Settings

CovWeight	1
Goal	100.0
WeightMode	at_least
Seeds	824213985 792842968
CountMode	count_first
IllegalMode	illegal_on
Threashold	45.0
ThresholdEnable	0
TotalCovCount	100
TotalCovGoal	100

Coverage Model Settings

▼ UART_RX_STIM_COV Coverage Bins

Name	Type	Mode	Data	Idle	Count	AtLeast	Percent Coverage
NORMAL	Count	1 to 1	0 to 255	0 to 0	63	63	100.0
NORMAL	Count	1 to 1	0 to 255	1 to 15	7	7	100.0
PARITY	Count	3 to 3	0 to 255	2 to 15	11	11	100.0
STOP	Count	5 to 5	1 to 255	2 to 15	11	11	100.0
PARITY_STOP	Count	7 to 7	1 to 255	2 to 15	6	6	100.0
BREAK	Count	9 to 15	11 to 30	2 to 15	2	2	100.0
Total Percent Coverage:	100.0						

Coverage Results

▼ UART_RX_COV Coverage Model **Coverage: 100.0**

▶ UART_RX_COV Coverage Settings

▼ UART_RX_COV Coverage Bins

HTML Simulation Transcript

```
▶ build ../../OsvvmLibraries/RunAllTestsWithCoverage.pro
▶ include ../../OsvvmLibraries/RunAllTestsWithCoverage.pro
▶ library default C:/tools/sim_temp/RivieraPRO-2022.04.117.8517
▶ include OsvvmLibraries.pro
▶ include ./osvvm/osvvm.pro
▶ library osvvm C:/tools/sim_temp/RivieraPRO-2022.04.117.8517
▶ analyze TextUtilPkg.vhd
▶ analyze ResolutionPkg.vhd
▶ analyze NamePkg.vhd
▶ analyze OsvvmGlobalPkg.vhd
▶ analyze VendorCovApiPkg_Aldec.vhd
▶ analyze TranscriptPkg.vhd
▶ analyze AlertLogPkg.vhd
▶ analyze NameStorePkg.vhd
▶ analyze MessageListPkg.vhd
▶ analyze SortListPkg_int.vhd
▶ analyze RandomBasePkg.vhd
▶ analyze RandomPkg.vhd
▶ analyze RandomProcedurePkg.vhd
▶ analyze CoveragePkg.vhd
▶ analyze ScoreboardGenericPkg.vhd
▶ analyze ScoreboardPkg_slv.vhd
▶ analyze ScoreboardPkg_int.vhd
▶ analyze ResizePkg.vhd
▶ analyze MemoryPkg.vhd
▶ analyze TbUtilPkg.vhd
▶ analyze ReportPkg.vhd
▶ analyze OsvvmTypesPkg.vhd
▶ analyze OsvvmContext.vhd
▶ include ./Common/Common.pro
▶ library OSVVM_Common C:/tools/sim_temp/RivieraPRO-2022.04.117.8517
▶ analyze ./src/ModelParametersPkg.vhd
▶ analyze ./src/FifoFillPkg_slv.vhd
▶ analyze ./src/StreamTransactionPkg.vhd
▶ analyze ./src/AddressBusTransactionPkg.vhd
▶ analyze ./src/AddressBusResponderTransactionPkg.vhd
▶ analyze ./src/AddressBusVersionCompatibilityPkg.vhd
▶ analyze ./src/InterruptHandler.vhd
▶ analyze ./src/InterruptHandlerComponentPkg.vhd
▶ analyze ./src/OsvvmCommonContext.vhd
▶ include ./UART/UART.pro
▶ library osvvm_uart C:/tools/sim_temp/RivieraPRO-2022.04.117.8517
▶ analyze ./src/UartTbPkg.vhd
▶ analyze ./src/ScoreboardPkg_Uart.vhd
▶ analyze ./src/UartTxComponentPkg.vhd
▶ analyze ./src/UartRxComponentPkg.vhd
▶ analyze ./src/UartContext.vhd
▶ analyze ./src/UartTx.vhd
▶ analyze ./src/UartRx.vhd
▶ include ./AXI4/AXI4.pro
▶ include ./common/common.pro
▶ library osvvm_axi4 C:/tools/sim_temp/RivieraPRO-2022.04.117.8517
▶ analyze ./src/Axi4InterfaceCommonPkg.vhd
▶ analyze ./src/Axi4LiteInterfacePkg.vhd
▶ analyze ./src/Axi4InterfacePkg.vhd
▶ analyze ./src/Axi4CommonPkg.vhd
▶ analyze ./src/Axi4ModelPkg.vhd
▶ analyze ./src/Axi4OptionsPkg.vhd
```

Simulation Transcript

- Details are hidden
- Rotate triangle to see details
- Scan a file log file that is otherwise 100K+ lines in seconds

HTML Simulation Transcript


```
▶ RunTest TbAxi4_AxiXResp.vhd
▶ include ./Axi4/RunAllTests.pro
▶ TestSuite Axi4Full
▶ library osvvm_TbAxi4 C:/tools/sim_temp/RivieraPRO-2022.04.117.8517
▶ include ./testbench
▶ library osvvm_TbAxi4 C:/tools/sim_temp/RivieraPRO-2022.04.117.8517
▶ analyze ../TestCases/OsvvmTestCommonPkg.vhd
▶ analyze TestCtrl_e.vhd
▶ analyze TbAxi4.vhd
▶ analyze TbAxi4Memory.vhd
▶ include ./TestCases
▼ RunTest TbAxi4_MemoryReadWriteDemo1.vhd

#
# analyze TbAxi4_MemoryReadWriteDemo1.vhd
# vcom {*}-2008 -relax -work osvvm_TbAxi4 ../../OsvvmLibraries/AXI4/Axi4/TestCases/TbAxi4_MemoryReadWriteDemo1.vhd
# ACOM: Warning: DAGGEN_0523: The source is compiled without the -dbg switch. Line breakpoints and assertion debug will not be available.
# ACOM: File: ../../OsvvmLibraries/AXI4/Axi4/TestCases/TbAxi4_MemoryReadWriteDemo1.vhd
# ACOM: Compile Architecture "MemoryReadWriteDemo1" of Entity "TestCtrl"
# ACOM: Compile Configuration "TbAxi4_MemoryReadWriteDemo1"
# ACOM: Compile success 0 Errors 0 Warnings  Analysis time :  0.1 [s]
# ACDB: Closing Code Coverage session.
# VSIM: Simulation has finished.
# simulate TbAxi4_MemoryReadWriteDemo1
#

# Simulate Start time 17:12:02
# vsim {*}-acdb_cov sbm -cc_all -t ps -lib osvvm_TbAxi4 TbAxi4_MemoryReadWriteDemo1
# ELBREAD: Elaboration process.
# ELBREAD: Elaboration time 0.0 [s].
# KERNEL: Main thread initiated.
# KERNEL: Kernel process initialization phase.
# ELAB2: Elaboration final pass...
# KERNEL: PLI/VHPI kernel's engine initialization done.
# PLI: Loading library 'C:\tools\Aldec\Riviera-PRO-2022.04-x64\bin\systf.dll'
# VHPI: Loading library 'systf.dll'
# ELAB2: Create instances ...
# KERNEL: Time resolution set to 1ps.
# ELAB2: Create instances complete.
# SLP: Started
# SLP: Elaboration phase ...
# SLP: Elaboration phase ... skipped, nothing to simulate in SLP mode : 0.0 [s]
# SLP: Finished : 0.0 [s]
# ELAB2: Elaboration final pass complete - time: 0.1 [s].
# ACDB: Code Coverage session started in hierarchical mode for all units.
# KERNEL: Kernel process initialization done.
# Allocation: Simulator allocated 44785 kB (elbread=427 elab2=11108 kernel=33249 sdf=0)
# KERNEL: ASDB file was created in location C:/SynthWorks/Dev/_osvvm/_sim/riviera/dataset.asdb
# VSIM: 87 object(s) traced.
# KERNEL: %% Log   PASSED  in Default, Default BurstMode is ADDRESS_BUS_BURST_WORD_MODE 0 at 110 ns
# KERNEL: %% Log   PASSED  in Default, BurstMode Received : 0 at 110 ns
# KERNEL: %% Log   ALWAYS  in Default, Write and Read. Addr = 0000.  16 words at 110 ns
# KERNEL: %% Log   INFO    in manager_1, Write Data.  WData: 00000001  WStrb: 1111  Operation# 1 at 110 ns
# KERNEL: %% Log   INFO    in manager_1, Write Address.  AWAddr: 00000010  AWProt: 000  Operation# 1 at 110 ns
# KERNEL: %% Log   DEBUG   in manager_1, Waiting for Write Response. at 110 ns
```

Simulation Transcript

When viewed from Test Case Report, it jumps to the simulation's results

Transcript File


```
%% Log   ALWAYS  in Default, Transmit 16 bytes.  Cover Random at 110 ns
%% Log   INFO    in transmitter_1, Axi Stream Send.  TData: 63200124  TStrb: 1111  TKeep: 1111  TID: 01  TDest: 2  TUser: 3  TLast: 0  Operation# 1 at 110 ns
%% Log   INFO    in transmitter_1, Axi Stream Send.  TData: 05022122  TStrb: 1111  TKeep: 1111  TID: 01  TDest: 2  TUser: 3  TLast: 0  Operation# 2 at 120 ns
%% Log   INFO    in transmitter_1, Axi Stream Send.  TData: 41072646  TStrb: 1111  TKeep: 1111  TID: 01  TDest: 2  TUser: 3  TLast: 0  Operation# 3 at 130 ns
%% Log   INFO    in transmitter_1, Axi Stream Send.  TData: 00276162  TStrb: 1111  TKeep: 1111  TID: 01  TDest: 2  TUser: 3  TLast: 1  Operation# 4 at 140 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 24   Item Number: 1 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 01   Item Number: 2 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 20   Item Number: 3 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 63   Item Number: 4 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 22   Item Number: 5 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 21   Item Number: 6 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 02   Item Number: 7 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 05   Item Number: 8 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 46   Item Number: 9 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 26   Item Number: 10 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 07   Item Number: 11 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 41   Item Number: 12 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 62   Item Number: 13 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 61   Item Number: 14 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 27   Item Number: 15 at 150 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 00   Item Number: 16 at 150 ns
%% Log   INFO    in receiver_1, Burst Check.  Operation# 1  Last Data: 00276162  TID: 01  TDest: 2  TUser: 3  TLast: 1 at 150 ns
%% Log   PASSED  in receiver_1, Burst Check WordCount Received : 16 at 150 ns
%% Log   PASSED  in receiver_1, ID Received : 01 at 150 ns
%% Log   PASSED  in receiver_1, DEST Received : 2 at 150 ns
%% Log   PASSED  in receiver_1, USER Received : 3 at 150 ns
%% Log   PASSED  in receiver_1, Last Received : 1 at 150 ns
%% Log   ALWAYS  in Default, Transmit 14 bytes. at 190 ns
%% Log   INFO    in transmitter_1, Axi Stream Send.  TData: 04230664  TStrb: 1111  TKeep: 1111  TID: 02  TDest: 3  TUser: 4  TLast: 0  Operation# 5 at 190 ns
%% Log   INFO    in transmitter_1, Axi Stream Send.  TData: 67604742  TStrb: 1111  TKeep: 1111  TID: 02  TDest: 3  TUser: 4  TLast: 0  Operation# 6 at 200 ns
%% Log   INFO    in transmitter_1, Axi Stream Send.  TData: 45034465  TStrb: 1111  TKeep: 1111  TID: 02  TDest: 3  TUser: 4  TLast: 0  Operation# 7 at 210 ns
%% Log   INFO    in transmitter_1, Axi Stream Send.  TData: XXXX4366  TStrb: 0011  TKeep: 0011  TID: 02  TDest: 3  TUser: 4  TLast: 1  Operation# 8 at 220 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 64   Item Number: 17 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 06   Item Number: 18 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 23   Item Number: 19 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 04   Item Number: 20 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 42   Item Number: 21 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 47   Item Number: 22 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 60   Item Number: 23 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 67   Item Number: 24 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 65   Item Number: 25 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 44   Item Number: 26 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 03   Item Number: 27 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 45   Item Number: 28 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 66   Item Number: 29 at 230 ns
%% Log   PASSED  in receiver_1: RxBurstFifo,    Received: 43   Item Number: 30 at 230 ns
%% Log   INFO    in receiver_1, Burst Check.  Operation# 2  Last Data: ----4366  TID: 02  TDest: 3  TUser: 4  TLast: 1 at 230 ns
%% Log   PASSED  in receiver_1, Burst Check WordCount Received : 14 at 230 ns
%% Log   PASSED  in receiver_1, ID Received : 02 at 230 ns
%% Log   PASSED  in receiver_1, DEST Received : 3 at 230 ns
%% Log   PASSED  in receiver_1, USER Received : 4 at 230 ns
%% Log   PASSED  in receiver_1, Last Received : 1 at 230 ns
%% Log   ALWAYS  in Default, Transmit 17 bytes. at 230 ns
%% Log   INFO    in transmitter_1, Axi Stream Send.  TData: 67234025  TStrb: 1111  TKeep: 1111  TID: 03  TDest: 4  TUser: 5  TLast: 0  Operation# 9 at
%% Log   INFO    in transmitter_1, Axi Stream Send.  TData: 20454520  TStrb: 1111  TKeep: 1111  TID: 03  TDest: 4  TUser: 5  TLast: 0  Operation# 10 at
```

Getting OSVVM & Running Scripts

- Documentation starts at:

```
https://osvvm.github.io/
```

- Get the sources:

```
git clone --recursive https://github.com/osvvm/OsvvmLibraries
```

  - Alternately, a zip file is at: osvvm.org/downloads

- Initialize the simulator – see Documentation/Scripts_user_guide.pdf

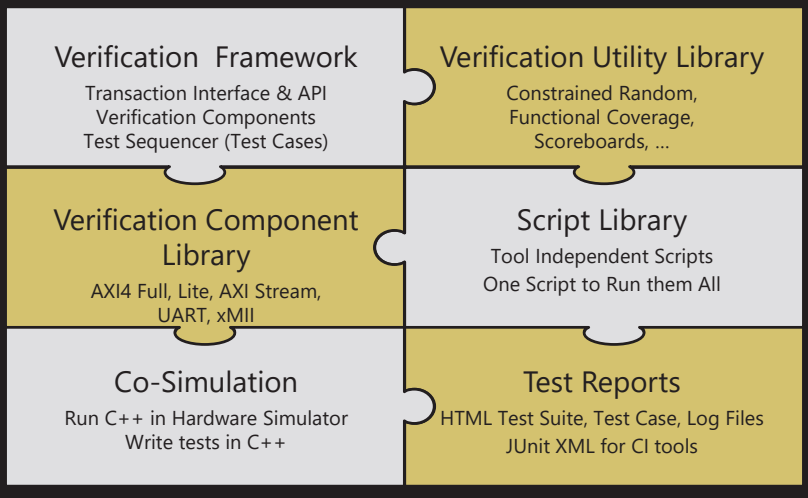
```
file mkdir sim ;   # In directory containing OsvvmLibraries
cd   sim
source ../OsvvmLibraries/Scripts/StartUp.tcl
```

- Build all OSVVM and Run AXI4 Tests

```
build ../OsvvmLibraries/OsvvmLibraries.pro
build ../OsvvmLibraries/AXI4/Axi4/RunAllTests.pro
```

  - There is a RunAllTests for each VC and the OsvvmLibraries
  - There is also a RunDemoTests

All you need is … OSVVM



- Benefits
  - Powerful and Concise – rivals other verification languages
  - Unmatched reuse through the entire verification process
  - Unmatched report capability with HTML for humans and JUnit XML for CI
  - Tests are Readable and Reviewable by All
  - Adopt incrementally as needed