OSVVM in a Nutshell,
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

SynthWorks

OSVVM in a Nutshell

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

OSVVM in a Nutshell

- Agenda
  - What is OSVVM?
  - OSVVM Verification Framework
  - Verification Components
  - Test Sequencer
  - Writing Directed Tests
  - Constrained Random Tests
  - Scoreboards
  - Functional Coverage
  - Intelligent Coverage Random
  - Protocol and Parameter Checks
  - Test Reporting
  - Scripts

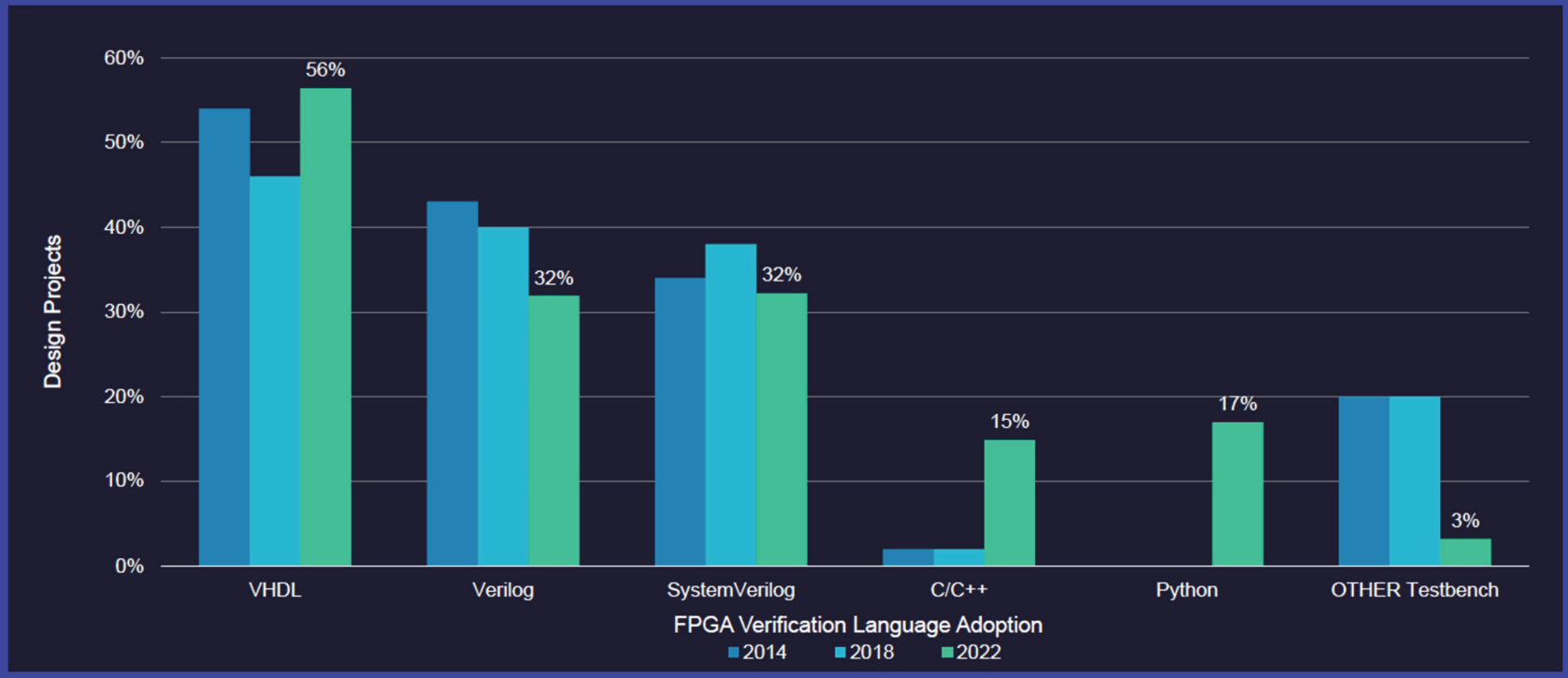
Background

- About Jim Lewis
  - 30 years: VHDL Design and Verification
  - 20+ years: Active in IEEE VHDL WGs
  - 15+ 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.

Why VHDL?

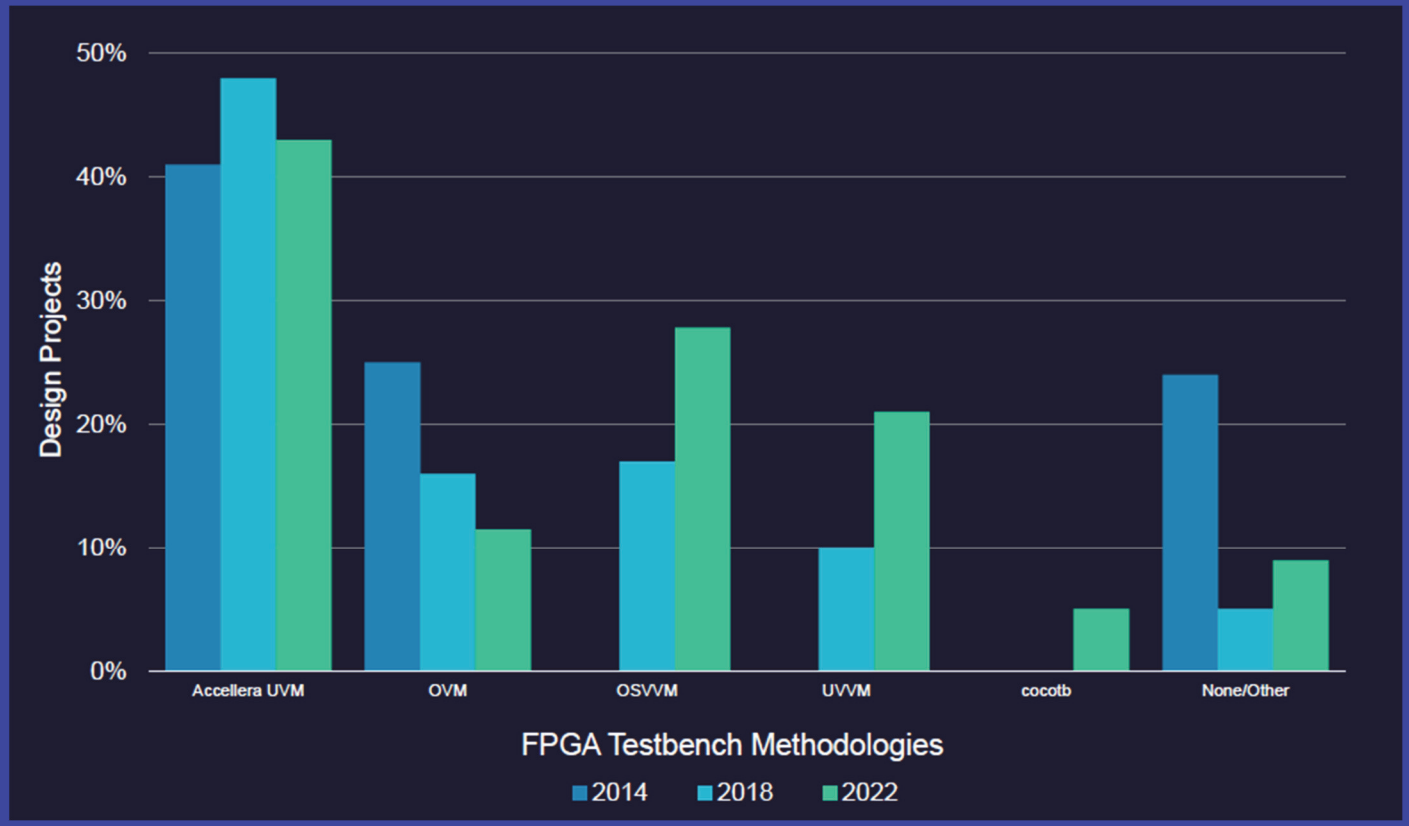
- VHDL is #1 for FPGA Design and Verification
- From Wilson Research Group 2022 Functional Verification Survey
  - For FPGA design: 66% worldwide use VHDL
  - For FPGA verification: 56% worldwide use VHDL



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

Why OSVVM?

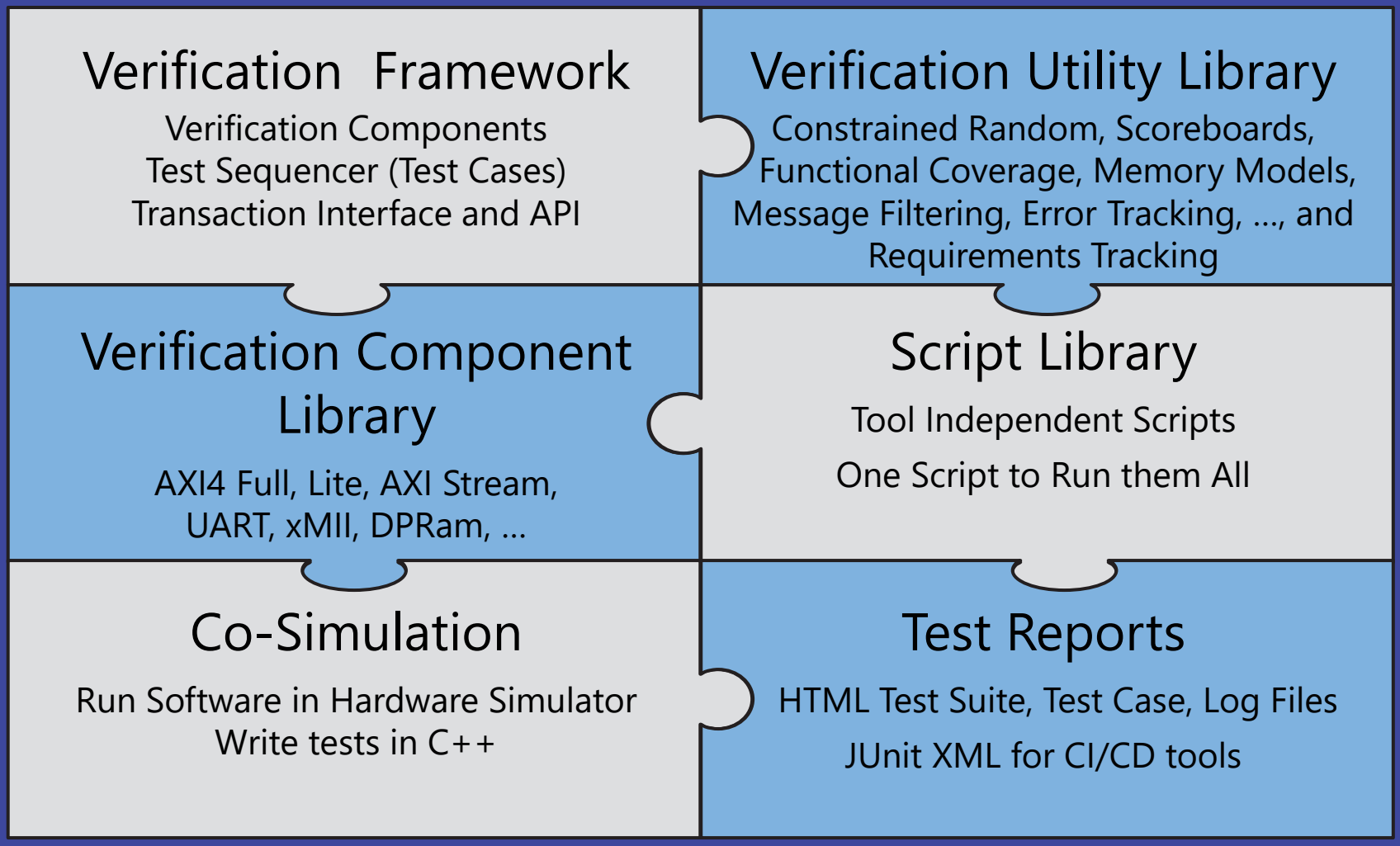
- OSVVM is VHDL's #1 Verification Methodology
- For FPGA Verification,
  - Worldwide: 28% use OSVVM = 50% of the VHDL FPGA users



- © 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



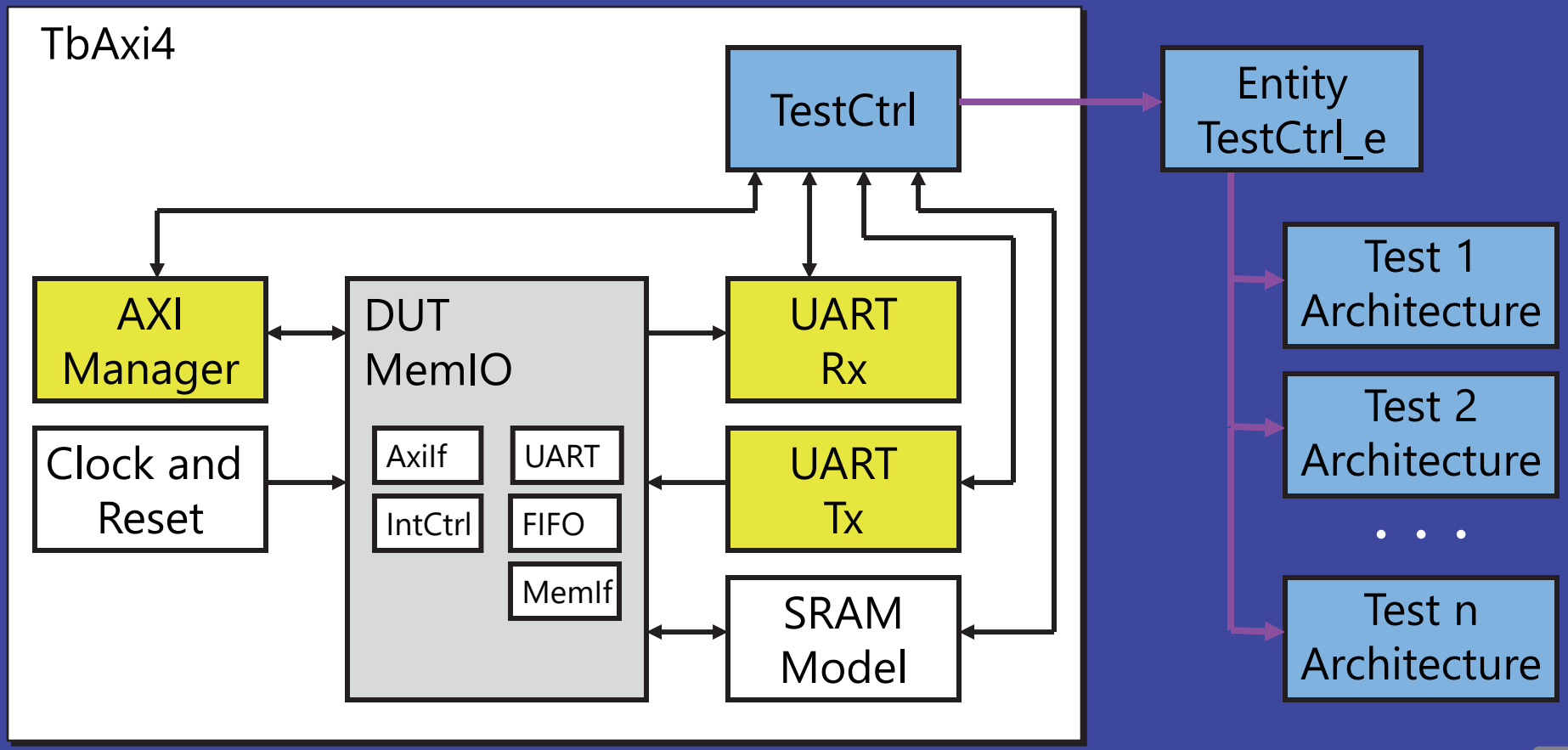
- OSVVM is Free, Open Source
- Developed by the same VHDL experts who helped with VHDL Standards

OSVVM History

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	HTML Log & Scoreboard Reports, Code Coverage Reports, Ethernet VC, Arrays of Transaction Interfaces
	2023	Co-simulation of C++ Software in a Hardware Simulator VC with delay randomization, Specification Tracking Part 2

OSVVM Verification Framework

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



Verification Framework - Implementation


```
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 (MRec, . . . ) ;
  UartRx_1      : UartRx(RxRec, . . . ) ;
  UartTx_1      : UartTx(TxRec, . . . ) ;

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

Structural Code
Plugs together just like RTL

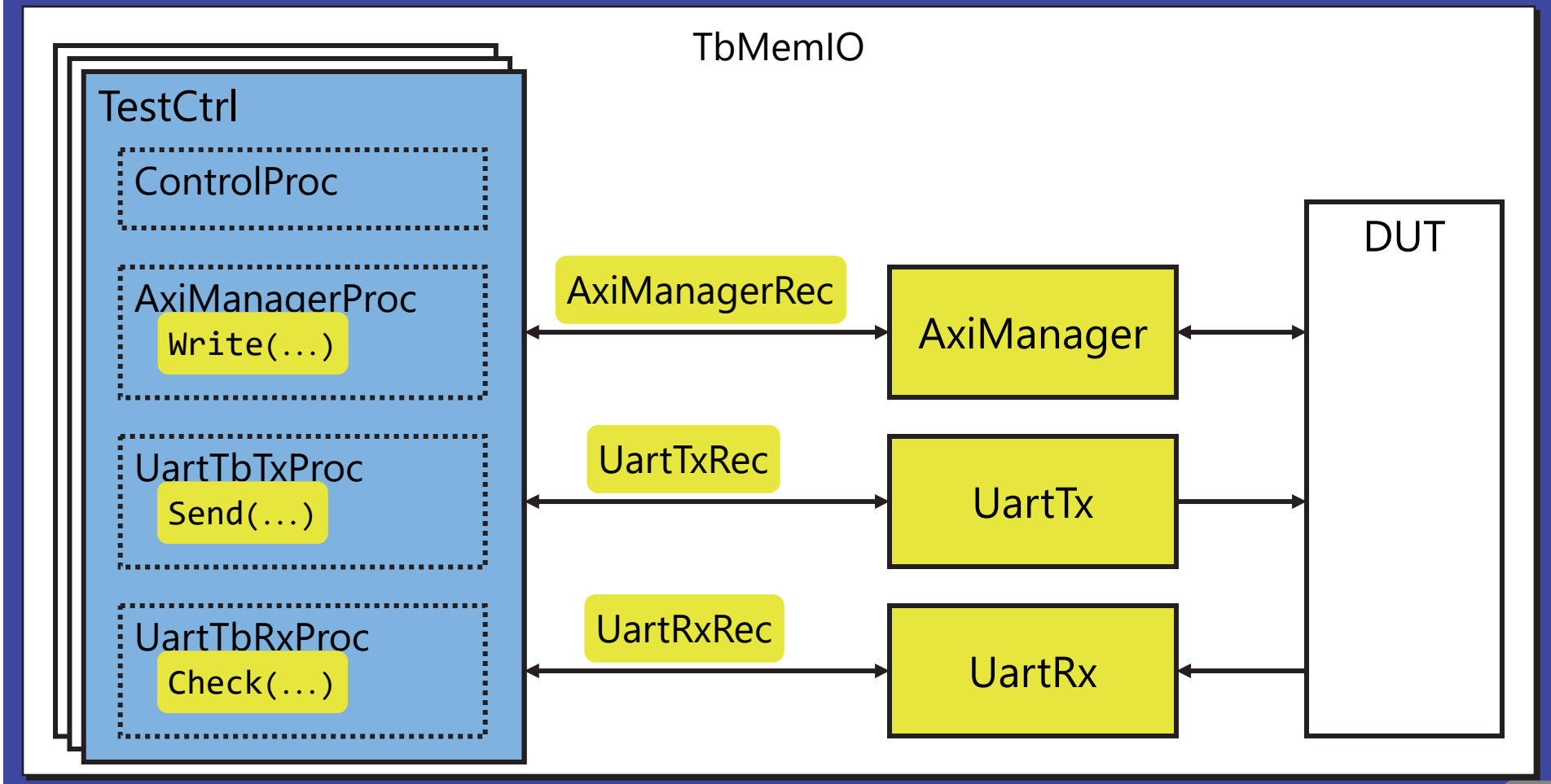
DUT

Verification Components

Test Sequencer

Verification Framework - Elements

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



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
- Only requires a mode view declaration for the record

Transaction API = VHDL Procedure


```
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.Dat
end procedure Read ;
```

Independent of VC

- Put transaction into record
- Handshake with VC
- Get results from record

OSVVM's Easy Approach

- Observation: Some interfaces do the same transactions
  - Address Bus Interfaces (AXI4, Avalon, …) do read and write
  - Streaming Interfaces (AxiStream, UART, …) do send and get
- For these interfaces, Model Independent Transactions (MIT) define
  - Transaction Interface (record) and Transaction API (procedures)
- … Address Bus MIT (basic subset)

```
type AddressBusRecType is record  . . . ;

Write(AddrRec, iAddr, iData) ;
Read (AddrRec, X"1111_1110", oData) ;
. . .
```

- … Stream MIT (basic subset)

```
type StreamRecType is record  . . . ;

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

Benefits of OSVVM MIT

- Accelerates Verification Component Development
  - Use the transaction interface and API are defined by MIT
  - Focus on writing VC behavior
  - A basic VC is as simple as writing a procedure
- Accelerates Test Case Development
  - Similar VC use the same transaction API
  - Share transaction sequences between similar VC
- Co-Simulation is Supported
  - OSVVM Co-Simulation supports all MIT based VC
- Accelerates Documentation
  - Only need to identify which transactions a VC supports
- More Information is in user guides in OsvvmLibraries/Documentation/
  - Address_Bus_Model_Independent_Transactions_user_guide.pdf
  - Stream_Model_Independent_Transactions_user_guide.pdf

Verification Components


```
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

Verification Components


```
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

Benefit: Coding OSVVM VC is well within the capabilities of any VHDL engineer

TestCtrl = Test Sequencer


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

  ManagerRec    : InOut AddressBusRecType ;

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

Ports =
Transaction Interfaces

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

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

Aspects of a Test Case

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

Test Initialization in ControlProc


```
  signal TbID, RxID : AlertLogIDType ;
  signal ReqID1     : AlertLogIDType ;
  signal SB         : ScoreboardIDType ;
  . . .
ControlProc : process
begin
  SetTestName("UartRx1") ;

  TranscriptOpen ;
  SetTranscriptMirror(TRUE) ;

  TBID   <= NewID("TB") ;
  RxID   <= NewID("UartRx_1") ;
  SB     <= NewID("SB", ModelID) ;
  ReqID1 <= GetReqID("Req1") ;

  wait for 0 ns ;
  SetLogEnable(PASSED, TRUE) ;
  SetLogEnable(RxID, INFO, TRUE) ;

  WaitForBarrier(TestDone, 5 ms) ;

  . . . -- Test Finalization
```

Declare IDs for OSVVM's singleton data structures

Set Test Name

Open Transcript File + Write to Console

Construct AlertLog, Requirements, and Scoreboard data structures

Enable Logs/Message Filters

Stop until Test Done or 5 ms passes = WatchDog timer

A Test Case

- Test Case = Send, Get, and Check transactions + affirmations (checks)

```
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 ;
```

- Affirmations signal pass/fail as well as count errors and checks

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

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

Benefit: Improves readability. Simplifies writing self-checking tests.

Requirements Tracking

- For requirements, affirmations also count errors and checks

```
RxProc : process
  variable RxD : ByteType;
begin
  . . .
  AffirmIfEqual(ReqID1, RxD, X"10") ;
  . . .
  AffirmIf(ReqID1, TRUE, "Req1 Passed") ;
  . . .
  AffirmIf("Req2.1", A=B,
    "Require that A: " & to_string(A) &
    " = B: " & to_string(B) ) ;
  . . .
  WaitForBarrier(TestDone);
end process RxProc ;
```

Use a Requirements ID

Create a Requirements ID and use it

Note:
A failed requirement is both a requirements failure and a test failure

Alerts Simplify Protocol Checks

- If error, print message and count error in the AlertLog data structure

```
SimultaneousAccessCheck: process
begin
  wait on iCE, iWE, iOE ;
  AlertIf((iCE and iWE and iOE) = '1', "iWE and iOE both active");
end process SimultaneousAccessCheck ;
```

```
%%   5500 ns   Alert ERROR   iWE and iOE both active
```

  - Alert Levels: FAILURE, ERROR (default), WARNING
  - Alerts are used for protocol and parameter checks

- Controls: StopCount, PrintCount, Enable/Disable

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

  - Alerts are enabled by default and rarely disabled

Logs Simplify Debug

- Logs are conditional printing (messaging)

```
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

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

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

Test Finalization

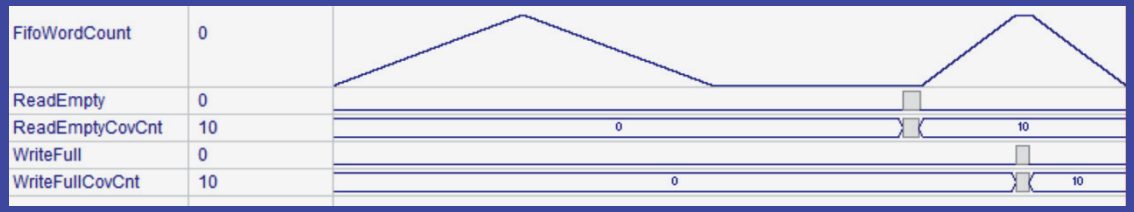

```
ControlProc : process
begin
  . . .  -- Test Initialization
  WaitForBarrier(TestDone, 5 ms) ;

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

  TranscriptClose ;
  AffirmIfTranscriptsMatch("ValidatedResults") ;

  AffirmIf("UartRx1: Is Self-Checking", GetAffirmCount > 0,
    "AffirmCount = " & to_string(GetAffirmCount) & ", Expect > 0");

  AffirmIf(GetTestName & ": Is Covered", GetCov = 100.0,
    "Coverage = " & to_string(GetCov) & ", Expect 100.0") ;

  AffirmIf(GetTestName & ": Passes", GetAlertCount = 0,
    "ErrorCount = " & to_string(GetAlertCount) & ", Expect 0") ;

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

Stop until Test Done or 5 ms has passed

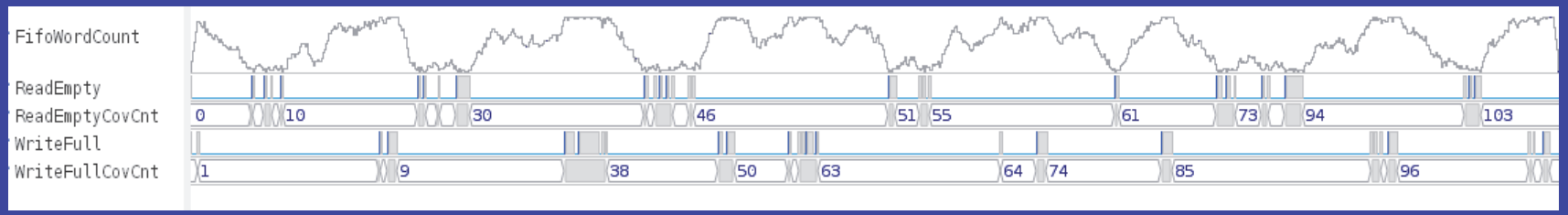
Create Reports

Constrained Random Testing

- 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, …

OSVVM Randomization Library

- 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

SynthWorks

= Code Pattern + Randomization + Transaction Calls

```
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 and Checking?

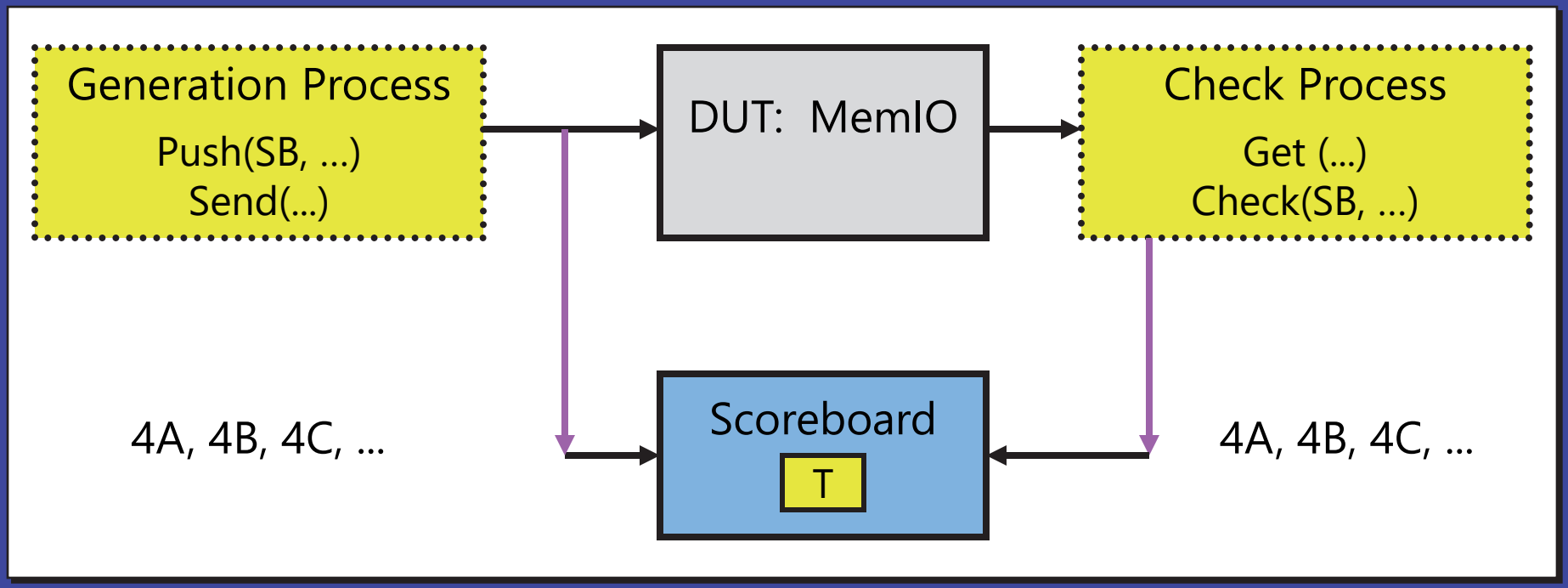
For checking, RxProc could repeat the randomization from TxProc, however, this is tedious and potentially error prone.

```
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

- Simplify self-checking when data is minimally transformed



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

OSVVM Generic Scoreboards


```
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


```
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 Scoreboards


```
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);
```

Benefit: The checker process (RxProc) stays the same, even when the generator process (TxProc) changes

Functional Coverage

- 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 independent 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)

Functional Coverage: CoveragePkg

- 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) ;
  . . .
```

Functional Coverage: Defining a Model

- For the UART, we track the following items

	Status Register Values				
Condition	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

Writing Functional Coverage


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

Coverage Object

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

Construct the Data Structure

Define coverage model

```
  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, to_integer(RxOp));
  end loop ;
  . . .
```

Collect Coverage

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

OSVVM's Intelligent Coverage Randomization

SynthWorks

= 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", 1000,  GenBin(3) ) ;
  AddBins(StimCov, "STOP",   1000,  GenBin(5) ) ;
  . . .
  for I in 1 to 10000 loop
    iOperation := GetRandPoint(StimCov) ;
    case iOperation is
      when 1  =>    . . .  -- Nominal 70%
      when 3  =>    . . .  -- Parity  10%
      . . .
    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

Intelligent Coverage goes beyond what SV does

Using OSVVM Libraries

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

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

-- 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


```
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?

- 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

Scripting

- Procedure based API that runs on top of TCL

```
library  osvvm_TbUart

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

analyze  ./testbench/TbUart_SendGet1.vhd
simulate TbUart_SendGet1
```

Activate Library

Compile

Simulate & Run TbUart_SendGet1

- Benefits
  - Simple, just like a list of source files …
    - … Except we also get the power of TCL
  - Settings (like current working library) are remembered
  - Paths are relative to script location to facilitate moving pieces of projects
- One Simulator Script API for
  - GHDL, NVC, Aldec, Siemens, Synopsys VCS, Cadence Xcelium

Calling Scripts

- build - Call a *.pro file from command line or CI

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

  - Build + EndOfTestReports generates the HTML and Junit reports

- include - Call a *.pro file from another *.pro file

```
# AXI4.pro
include ./common/common.pro
include ./Axi4Lite/Axi4Lite.pro
include ./AxiStream/AxiStream.pro
include ./Axi4/Axi4.pro
```

- Use Build and Include rather than TCL's source or EDA vendor's do
  - Used to make paths relative to directory from which the scripts run
  - Supports $ARGC, $ARGV0, $ARGV(1), $ARGV(2), ...

OSVVM Creates Unmatched Test Reports

- OSVVM Test Completion Message
- Build Summary Mini-Report – Text
- Build Summary Report – HTML
  - Summary of entire build
  - Summary of each test suite in the build
  - Summary of each test case within a test suite
- Requirements Summary – HTML and CSV
- Test Case Reports – HTML
  - Reports on Alert, Functional Coverage, and Scoreboards
  - Links to HTML simulation transcript and test file output
- HTML'ized simulation transcript / log file (simulator output)
  - Errors shown in red
- JUnit Report – XML for Continuous Integration Tools
- For more see: https://osvvm.github.io/Overview/Osvvm3Reports.html

Test Completion Message

- EndOfTestReports produces a summary and if errors a detailed report

```
EndOfTestReports ;
```

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

```
%% 100100100 ns  DONE  FAILED   Test_UartRx_1  Total Error(s) = 7  Failures: 0
Errors: 7  Warnings: 0  Passed: 41  Affirmations Checked: 48
%%    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
%%      Data Err      Failures: 0  Errors: 7  Warnings: 0  Passed: 41
%%      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
```

Build Summary Mini-Report

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

```
BuildError: Sim_Demo FAILED,  Passed: 149,  Failed: 3,  Skipped: 0,
Analyze Errors: 0,  Simulate Errors: 0,  Build Error Code: 0
```

  - If there are errors, this is the first place we see an indication

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
TbAxi4_RandomReadWrite	PASSED	3000 / 3000	0	0 / 0	-	0	5.333

Created by Scripts + EndOfTestReports

Build Status

- Summarizes all tests run
- Link to Simulation Transcript
  - Both text and html
- Link to code coverage

Test Suite Summaries

Test Suite = multiple test cases
Summarizes Pass/Fail+

Test Case Summaries

Test Case = Testbench
Identify Failing Test(s)
Links to Test Case Reports

Build Summary Report with Errors

SynthWorks

Simplifies debug of regression tests

sim_OsvvmDemo Build Summary Report

Build	sim_OsvvmDemo
Status	FAILED
PASSED	149
FAILED	3
SKIPPED	0
Analyze Failures	0
Simulate Failures	0
Elapsed Time (h:m:s)	0:42:39
Elapsed Time (seconds)	2558.991
Start Time	2023-05-07T16:20-0700
Simulator	QuestaSim -gui
Simulator Version	QuestaSim-2022.01
OSVVM Version	2023.04
Simulation Transcript	sim_OsvvmDemo.log
HTML Simulation Transcript	sim_OsvvmDemo_log.html
Code Coverage	Code Coverage Results
Finish Time	2023-05-07T17:02-0700

▼ sim_OsvvmDemo Test Suite Summary

TestSuites	Status	PASSED	FAILED	SKIPPED	Requirements passed / goal	Disabled Alerts	Elapsed Time
Axi4Lite	PASSED	11	0	0	0 / 0	0	197.229
Axi4Full	FAILED	59	3	0	0 / 0	0	1042.115
AxiStream	PASSED	63	0	0	0 / 0	0	997.658
Uart	PASSED	9	0	0	0 / 0	0	153.148
DpRam	PASSED	1	0	0	0 / 0	0	8.976
Ethernet	PASSED	6	0	0	0 / 0	0	113.837

▸ Axi4Lite Test Case Summary

▼ Axi4Full Test Case Summary

Test Case	Status	Checks passed / checked	Errors	Requirements passed / goal	Functional Coverage	Disabled Alerts	Elapsed Time
TbAxi4_DemoMemoryReadWrite1	PASSED	334 / 334	0	0 / 0	43.75	0	12.847
TbAxi4_DemoErrorMemoryReadWrite1	FAILED	300 / 302	2	0 / 0	100.00	0	3.378
TbAxi4_BasicReadWrite	PASSED	60 / 60	0	0 / 0	-	0	15.823
TbAxi4_RandomReadWrite	PASSED	3000 / 3000	0	0 / 0	-	0	16.387
TbAxi4_RandomReadWriteByte1	PASSED	3000 / 3000	0	0 / 0	-	0	13.892
TbAxi4_SubordinateReadWrite1	PASSED	60 / 60	0	0 / 0	-	0	12.095
TbAxi4_SubordinateReadWrite2	PASSED	60 / 60	0	0 / 0	-	0	11.955
TbAxi4_SubordinateReadWrite3	PASSED	60 / 60	0	0 / 0	-	0	11.529
TbAxi4_ReadWriteAsync1	PASSED	60 / 60	0	0 / 0	-	0	11.919

Requirements Report

SynthWorks

▼ sim_RunDemoTestsWithCov Requirement Results

Requirement	TestName	Status	Requirements		Checks			Alert Counts			Disabled Alert Counts		
			Goal	Passed	Total	Passed	Failed	Failures	Errors	Warnings	Failures	Errors	Warnings
1.1	Merged	PASSED	1	6	6	6	0	0	0	0	0	0	0
	TbAlertLog_t18_requirements	PASSED	1	1	1	1	0	0	0	0	0	0	0
	TbAlertLog_t19_requirements_print_passed	PASSED	1	1	1	1	0	0	0	0	0	0	0
	TbAlertLog_t20_requirements_explicit_hierarchy	PASSED	1	1	1	1	0	0	0	0	0	0	0
	TbAlertLog_t21_requirements_no_hierarchy	PASSED	1	1	1	1	0	0	0	0	0	0	0
	TbAlertLog_t22_requirements_w_errors	PASSED	1	1	1	1	0	0	0	0	0	0	0
	TbAlertLog_t23_read_requirements	PASSED	1	1	1	1	0	0	0	0	0	0	0
1.2	Merged	PASSED	2	12	12	12	0	0	0	0	0	0	0
	TbAlertLog_t18_requirements	PASSED	2	2	2	2	0	0	0	0	0	0	0
	TbAlertLog_t19_requirements_print_passed	PASSED	2	2	2	2	0	0	0	0	0	0	0
	TbAlertLog_t20_requirements_explicit_hierarchy	PASSED	2	2	2	2	0	0	0	0	0	0	0
	TbAlertLog_t21_requirements_no_hierarchy	PASSED	2	2	2	2	0	0	0	0	0	0	0
	TbAlertLog_t22_requirements_w_errors	PASSED	2	2	2	2	0	0	0	0	0	0	0
	TbAlertLog_t23_read_requirements	PASSED	2	2	2	2	0	0	0	0	0	0	0
1.3	Merged	FAILED	3	18	20	18	2	0	2	0	0	0	0
	TbAlertLog_t18_requirements	PASSED	3	3	3	3	0	0	0	0	0	0	0
	TbAlertLog_t19_requirements_print_passed	PASSED	3	3	3	3	0	0	0	0	0	0	0
	TbAlertLog_t20_requirements_explicit_hierarchy	PASSED	3	3	3	3	0	0	0	0	0	0	0
	TbAlertLog_t21_requirements_no_hierarchy	PASSED	3	3	3	3	0	0	0	0	0	0	0
	TbAlertLog_t22_requirements_w_errors	FAILED	3	3	5	3	2	0	2	0	0	0	0
	TbAlertLog_t23_read_requirements	PASSED	3	3	3	3	0	0	0	0	0	0	0
1.4	Merged	FAILED	4	24	26	24	2	0	2	0	0	0	0
	TbAlertLog_t18_requirements	PASSED	4	4	4	4	0	0	0	0	0	0	0
	TbAlertLog_t19_requirements_print_passed	PASSED	4	4	4	4	0	0	0	0	0	0	0
	TbAlertLog_t20_requirements_explicit_hierarchy	PASSED	4	4	4	4	0	0	0	0	0	0	0
	TbAlertLog_t21_requirements_no_hierarchy	PASSED	4	4	4	4	0	0	0	0	0	0	0
	TbAlertLog_t22_requirements_w_errors	FAILED	4	4	6	4	2	0	2	0	0	0	0
	TbAlertLog_t23_read_requirements	PASSED	4	4	4	4	0	0	0	0	0	0	0
2.1	TbAlertLog_t23_read_requirements	PASSED	2	2	2	2	0	0	0	0	0	0	0
2.2	TbAlertLog_t23_read_requirements	PASSED	2	2	2	2	0	0	0	0	0	0	0
2.3	TbAlertLog_t23_read_requirements	PASSED	2	2	2	2	0	0	0	0	0	0	0
2.4	TbAlertLog_t23_read_requirements	PASSED	2	2	2	2	0	0	0	0	0	0	0
3.1	TbAlertLog_t23_read_requirements	PASSED	1	1	1	1	0	0	0	0	0	0	0
3.2	TbAlertLog_t23_read_requirements	PASSED	1	1	1	1	0	0	0	0	0	0	0

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.

Test Case Report: Alert Report

SynthWorks

TbAxi4_RandomReadWrite Alert Report

▼ TbAxi4_RandomReadWrite 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_RandomReadWrite Alert Results

Name	Status	Checks		Total Errors	Alert Counts			Requirements		Disabled Alert Counts		
		Passed	Total		Failures	Errors	Warnings	Passed	Checked	Failures	Errors	Warnings
TbAxi4_RandomReadWrite	PASSED	3000	3000	0	0	0	0	0	0	0	0	0
Default	PASSED	1750	1750	0	0	0	0	0	0	0	0	0
OSVVM	PASSED	0	0	0	0	0	0	0	0	0	0	0
subordinate_1	PASSED	0	0	0	0	0	0	0	0	0	0	0
subordinate_1: Protocol Error	PASSED	0	0	0	0	0	0	0	0	0	0	0
subordinate_1: Data Check	PASSED	0	0	0	0	0	0	0	0	0	0	0
subordinate_1: No response	PASSED	0	0	0	0	0	0	0	0	0	0	0
manager_1	PASSED	0	0	0	0	0	0	0	0	0	0	0
manager_1: Protocol Error	PASSED	0	0	0	0	0	0	0	0	0	0	0
manager_1: Data Check	PASSED	250	250	0	0	0	0	0	0	0	0	0
manager_1: No response	PASSED	0	0	0	0	0	0	0	0	0	0	0
manager_1: WriteResponse Scoreboard	PASSED	500	500	0	0	0	0	0	0	0	0	0
manager_1: ReadResponse Scoreboard	PASSED	500	500	0	0	0	0	0	0	0	0	0
manager_1: WriteBurstFifo	PASSED	0	0	0	0	0	0	0	0	0	0	0

Test Case Report: Alert Report with Errors

SynthWorks

TbAxi4_DemoErrorMemoryReadWrite1 Alert Report

▼ TbAxi4_DemoErrorMemoryReadWrite1 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	

Status FAILED in table indicates

TbAxi4_DemoErrorMemoryReadWrite1 has 2 errors
The 2 errors were detected in the manager_1 VC
One error is in the manager_1::Data Check
One error is in manager1::ReadBurstFifo

▼ TbAxi4_DemoErrorMemoryReadWrite1 Alert Results

Name	Status	Checks		Total Errors	Alert Counts			Requirements		Disabled Alert Counts		
		Passed	Total		Failures	Errors	Warnings	Passed	Checked	Failures	Errors	Warnings
TbAxi4_DemoErrorMemoryReadWrite1	FAILED	300	302	2	0	2	0	0	0	0	0	0
Default	PASSED	0	0	0	0	0	0	0	0	0	0	0
OSVVM	PASSED	0	0	0	0	0	0	0	0	0	0	0
Testbench	PASSED	20	20	0	0	0	0	0	0	0	0	0
manager_1	FAILED	0	0	2	0	2	0	0	0	0	0	0
Protocol Error	PASSED	0	0	0	0	0	0	0	0	0	0	0
Data Check	FAILED	15	16	1	0	1	0	0	0	0	0	0
No response	PASSED	0	0	0	0	0	0	0	0	0	0	0
WriteResponse Scoreboard	PASSED	40	40	0	0	0	0	0	0	0	0	0
ReadResponse Scoreboard	PASSED	134	134	0	0	0	0	0	0	0	0	0
WriteBurstFifo	PASSED	0	0	0	0	0	0	0	0	0	0	0
ReadBurstFifo	FAILED	91	92	1	0	1	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
memory_1:memory	PASSED	0	0	0	0	0	0	0	0	0	0	0

Test Case Report: Functional Coverage

SynthWorks

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

Test Case Report: Scoreboards

SynthWorks

TbAxi4_DemoErrorMemoryReadWrite1 Scoreboard Report for Scoreboard_slv

Name	ParentName	ItemCount	ErrorCount	ItemsChecked	ItemsPopped	ItemsDropped	FifoCount
WriteResponse Scoreboard	manager_1	40	0	40	40	0	0
ReadResponse Scoreboard	manager_1	134	0	134	134	0	0
WriteAddressFIFO	manager_1	40	0	0	40	0	0
WriteDataFifo	manager_1	134	0	0	134	0	0
ReadAddressFifo	manager_1	40	0	0	40	0	0
ReadAddressTransactionFifo	manager_1	40	0	0	40	0	0
ReadDataFifo	manager_1	134	0	0	134	0	0
WriteAddressFIFO	memory_1	40	0	0	40	0	0
WriteDataFifo	memory_1	134	0	0	134	0	0
WriteResponseFifo	memory_1	40	0	0	40	0	0
ReadAddressFifo	memory_1	40	0	0	40	0	0
ReadDataFifo	memory_1	134	0	0	134	0	0
WriteBurstFifo	manager_1	102	0	0	102	0	0
ReadBurstFifo	manager_1	102	1	92	102	0	0

- A separate table is created for each scoreboard instance
- Each row in the table has statistics for a single scoreboard

- Tables for Scoreboard_slv and Scoreboard_int are automatically generated
- Use WriteScoreboardYaml to generate reports for user created scoreboards

HTML'ized 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 that is otherwise 100K+ lines in seconds

HTML'ized Simulation Transcript with Errors


```
▶ include ./AXI4/Axi4/TestCases/TestCasesWithError.pro
▶ RunTest TbAxi4_DemoMemoryReadWrite1.vhd
▶ RunTest TbAxi4_DemoErrorMemoryReadWrite1.vhd
%%         3210 ns    DONE    FAILED    TbAxi4_DemoErrorMemoryReadWrite1  Total Error(s) = 2  Failures: 0  Errors: 2  Warnings: 0  Passed: 300  Affirmations Checked: 302
▶ include TestCases_NoBurst.pro
▶ RunTest TbAxi4_BasicReadWrite.vhd
▶ RunTest TbAxi4_RandomReadWrite.vhd
▶ RunTest TbAxi4_RandomReadWriteByte1.vhd
▶ RunTest TbAxi4_SubordinateReadWrite1.vhd
▶ RunTest TbAxi4_SubordinateReadWrite2.vhd
▶ RunTest TbAxi4_SubordinateReadWrite3.vhd
▶ RunTest TbAxi4_ReadWriteAsync1.vhd
▶ RunTest TbAxi4_ReadWriteAsync2.vhd
▶ RunTest TbAxi4_ReadWriteAsync3.vhd
▶ RunTest TbAxi4_ReadWriteAsync4.vhd
▶ RunTest TbAxi4_SubordinateReadWriteAsync1.vhd
▶ RunTest TbAxi4_SubordinateReadWriteAsync2.vhd
▶ RunTest TbAxi4_MultipleDriversManager.vhd
▶ RunTest TbAxi4_MultipleDriversSubordinate.vhd
▶ RunTest TbAxi4_ReleaseAcquireSubordinate1.vhd
▶ RunTest TbAxi4_AlertLogIDManager.vhd
▶ RunTest TbAxi4_AlertLogIDSubordinate.vhd
▶ RunTest TbAxi4_TransactionApiSubordinate.vhd
▶ RunTest TbAxi4_ValidTimingManager.vhd
▶ RunTest TbAxi4_ValidTimingSubordinate.vhd
▶ RunTest TbAxi4_ReadyTimingSubordinate.vhd
▶ RunTest TbAxi4_AxiIfOptionsManagerSubordinate.vhd
▶ RunTest TbAxi4_AxiXResp.vhd
▶ RunTest TbAxi4_AxiXResp2_Enum.vhd
▶ RunTest TbAxi4_AxiXResp3_slv.vhd
▶ RunTest TbAxi4_TimeOutManager.vhd
▶ RunTest TbAxi4_TimeOutSubordinate.vhd
▶ RunTest TbAxi4_MemoryReadWrite1.vhd
▶ RunTest TbAxi4_MemoryReadWrite2.vhd
▶ RunTest TbAxi4_MultipleDriversMemory.vhd
▶ RunTest TbAxi4_ReleaseAcquireMemory1.vhd
▶ RunTest TbAxi4_AlertLogIDMemory.vhd
▶ RunTest TbAxi4_TimeOutMemory.vhd
▶ RunTest TbAxi4_TransactionApiManager.vhd
▶ RunTest TbAxi4_TransactionApiMemory.vhd
▶ RunTest TbAxi4_ValidTimingMemory.vhd
▶ RunTest TbAxi4_ReadyTimingManager.vhd
▶ RunTest TbAxi4_ReadyTimingMemory.vhd
▶ RunTest TbAxi4_MemoryAsync.vhd
▶ RunTest TbAxi4_DemoErrorMemoryReadWrite1.vhd
%%         3210 ns    DONE    FAILED    TbAxi4_DemoErrorMemoryReadWrite1  Total Error(s) = 2  Failures: 0  Errors: 2  Warnings: 0  Passed: 300  Affirmations Checked: 302
▶ include TestCases_Burst.pro
▶ RunTest TbAxi4_MemoryBurst1.vhd
▶ RunTest TbAxi4_MemoryBurstAsync1.vhd
▶ RunTest TbAxi4_MemoryBurstByte1.vhd
▶ RunTest TbAxi4_MemoryBurstPattern1.vhd
▶ RunTest TbAxi4_MemoryBurstPattern2.vhd
▶ RunTest TbAxi4_MemoryBurstBytePattern1.vhd
▶ RunTest TbAxi4_MemoryBurstAsyncPattern1.vhd
▶ RunTest TbAxi4_MemoryBurstAsyncPattern2.vhd
▶ RunTest TbAxi4_MemoryBurstSparse1.vhd
▶ RunTest TbAxi4_ReleaseAcquireManager1.vhd
▶ RunTest TbAxi4_AxSizeManagerMemory1.vhd
▶ RunTest TbAxi4_AxSizeManagerMemory2.vhd
▶ RunTest TbAxi4_AxiIfOptionsManagerMemory.vhd
▶ RunTest TbAxi4_TransactionApiManagerBurst.vhd
```

Errors are shown in red in the HTML'ized report

As a result, this report can be scanned for errors

HTML'ized Simulation Transcript – Test Case Information

SynthWorks

```
▶ RunTest TbAxi4_WriteOptions.vhd
▶ RunTest TbAxi4_MemoryReadWrite1.vhd
▶ RunTest TbAxi4_AxiXResp.vhd
▶ RunTest TbAxi4_AxiXResp2_Enum.vhd
▶ TestSuite Axi4Full
▶ include ./AXI4/Axi4/testbench
▶ analyze ../TestCases/OsvvmTestCommonPkg.vhd
▼ analyze TestCtrl_e.vhd

▶ analyze TbAxi4.vhd
▶ analyze TbAxi4Memory.vhd
▶ include ./AXI4/Axi4/TestCases/TestCasesWithError.pro
▶ RunTest TbAxi4_DemoMemoryReadWrite1.vhd
▼ RunTest TbAxi4_DemoErrorMemoryReadWrite1.vhd

TestName TbAxi4_DemoErrorMemoryReadWrite1
analyze TbAxi4_DemoErrorMemoryReadWrite1.vhd
End time: 16:24:28 on May 07,2023, Elapsed time: 0:00:15
Errors: 0, Warnings: 0
Simulation Start time 16:24:28
simulate TbAxi4_DemoErrorMemoryReadWrite1
vsim -t ps -lib osvvm_TbAxi4 TbAxi4_DemoErrorMemoryReadWrite1 -quiet -coverage -suppress 8683 -suppress 8684
Start time: 16:24:28 on May 07,2023
** Note: (vsim-8009) Loading existing optimized design _opt
%%          110 ns    Log    PASSED    in Testbench,                          Default BurstMode is ADDRESS_BUS_BURST_WORD_MODE 0
%%          110 ns    Log    PASSED    in Testbench,                          BurstMode Received : 0
%%          110 ns    Log    ALWAYS    in Testbench,                          Write and Read. Addr = 0000.  16 words
%%          110 ns    Log    INFO      in manager_1,                          Write Address.  AWAddr: 00000010  AWProt: 000  Operation# 1
%%          110 ns    Log    INFO      in manager_1,                          Write Data.  WData: 00000001  WStrb: 1111  Operation# 1

. . .

%%         1030 ns    Log    INFO      in manager_1,                          Read Address.  ARAddr: 000010F0  ARProt: 000  Operation# 31
%%         1040 ns    Log    INFO      in memory_1,                           Memory Read.  ARAddr: 000010F0  ARProt: 000  RData: 0000100F  Operation# 30
%%         1050 ns    Log    PASSED    in manager_1: ReadResponse Scoreboard,    Received: 0   Item Number: 31
%%         1050 ns    Log    PASSED    in manager_1: Data Check,              Read Data: 0000100F  Read Address: 000010F0  Prot: 0
%%         1050 ns    Log    INFO      in manager_1,                          Read Address.  ARAddr: 000010F0  ARProt: 000  Operation# 32
%%         1060 ns    Log    INFO      in memory_1,                           Memory Read.  ARAddr: 000010F0  ARProt: 000  RData: 0000100F  Operation# 31
%%         1070 ns    Log    PASSED    in manager_1: ReadResponse Scoreboard,    Received: 0   Item Number: 32
%%         1070 ns    Log    Alert     ERROR     in manager_1: Data Check,    Read Data: 0000100F  Read Address: 000010F0  Prot: 0   Expected: 00001010
%%         1070 ns    Log    ALWAYS    in Default,                            WriteBurst and ReadBurst.  Addr = 2000.  16 words
%%         1070 ns    Log    INFO      in manager_1,                          Write Address.  AWAddr: 00002000  AWProt: 000  Operation# 33
%%         1070 ns    Log    INFO      in manager_1,                          Write Data.  WData: 00002001  WStrb: 1111  Operation# 33
%%         1080 ns    Log    INFO      in memory_1,                           Memory Write.  AWAddr: 00002000  AWProt: 000  WData: 00002001  WStrb: 1111  Operation# 32

. . .

%%         1400 ns    Log    PASSED    in manager_1: ReadBurstFifo,              Received: 0000200D   Item Number: 13
%%         1400 ns    Log    PASSED    in manager_1: ReadBurstFifo,              Received: 0000200E   Item Number: 14
%%         1400 ns    Log    PASSED    in manager_1: ReadBurstFifo,              Received: 0000200F   Item Number: 15
%%         1400 ns    Log    Alert     ERROR     in manager_1: ReadBurstFifo,    Received: 00002010   Expected: 00002011   Item Number: 16
%%         1400 ns    Log    ALWAYS    in Testbench,                          Burst Vector.  Addr = 3000, 13 Words -- unaligned
%%         1400 ns    Log    INFO      in manager_1,                          Write Address.  AWAddr: 00003000  AWProt: 000  Operation# 34
%%         1400 ns    Log    INFO      in manager_1,                          Write Data.  WData: 0001UUUU  WStrb: 1100  Operation# 49
%%         1410 ns    Log    INFO      in memory_1,                           Memory Write.  AWAddr: 00003000  AWProt: 000  WData: 0001UUUU  WStrb: 1100  Operation# 33
%%         1410 ns    Log    INFO      in manager_1,                          Write Data.  WData: 00000003  WStrb: 1111  Operation# 50
%%         1420 ns    Log    INFO      in manager_1,                          Write Data.  WData: 00000005  WStrb: 1111  Operation# 51
```

Simulation Transcript

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

VHDL Transcript File

SynthWorks

```
%% 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 2
%% 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
```

Created by TranscriptOpen
Test Case Report links to this file

Getting OSVVM & Running Scripts

- 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

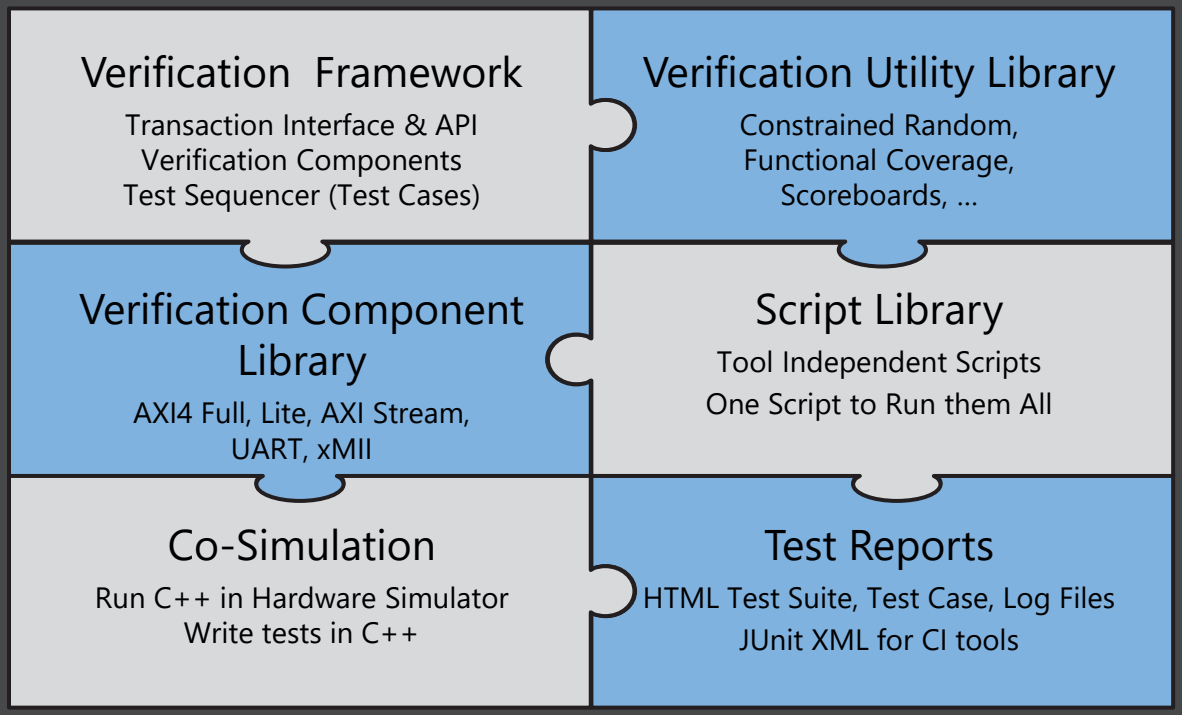
```
source <path-to-OsvvmLibs>/OsvvmLibraries/Scripts/StartUp.tcl
```

- Build all OSVVM and Run All VC Tests

```
build $OsvvmLibraries/OsvvmLibraries.pro
build $OsvvmLibraries/RunAllTests.pro
```

  - Each VC has a RunAllTests and 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
  - Tests and VC can be written by any VHDL Engineer

SynthWorks VHDL Classes

Comprehensive VHDL Introduction 4 Days - beginners class

http://www.synthworks.com/comprehensive_vhdl_introduction.htm

A design and verification engineer's introduction to VHDL syntax, RTL coding, and testbenches. Students get VHDL hardware experience with our FPGA based lab board.

Advanced VHDL Testbenches and Verification - OSVVM Boot Camp - 5 days

http://www.synthworks.com/vhdl_testbench_verification.htm

Learn the latest VHDL verification techniques including transaction based modeling, self-checking, scoreboards, memory modeling, functional coverage, directed, algorithmic, constrained random, and intelligent testbench test generation. Create a VHDL testbench environment that is competitive with other verification languages, such as SystemVerilog or 'e'. Our techniques work on VHDL simulators without additional licenses and are accessible to RTL engineers.

VHDL Coding for Synthesis 4 Days

http://www.synthworks.com/vhdl_rtl_synthesis.htm

Learn VHDL RTL (FPGA and ASIC) coding styles, methodologies, design techniques, problem solving techniques, and advanced language constructs to produce better, faster, and smaller logic.

OSVVM Resources

- Documentation
  - HTML: https://osvvm.github.io/Overview/Osvvm1About.html
  - PDF: OsvvmLibraries/Documentation - in OSVVM release
- Forum: https://osvvm.org
- Recorded Webinars
  - OSVVM: Leading Edge Verification for the VHDL Community
    - https://www.youtube.com/watch?v=KVmGDy_PHNI
  - Faster than Lite Verification Component Development with OSVVM
    - https://www.aldec.com/en/support/resources/multimedia/webinars/2187
  - OSVVM's Test Reports and Simulator Independent Scripting
    - https://www.aldec.com/en/support/resources/multimedia/webinars/2188
  - Advances in OSVVM's Verification Data Structures
    - https://www.aldec.com/en/support/resources/multimedia/webinars/2190
- Jump start your VHDL verification effort with training
  - Advanced VHDL Testbenches and Verification – OSVVM Boot Camp
  - https://synthworks.com/vhdl_testbench_verification.htm