Why Your Team Should be Using VHDL+OSVVM for Verification

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

SynthWorks

Why VHDL+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

Why VHDL+OSVVM?

Topics

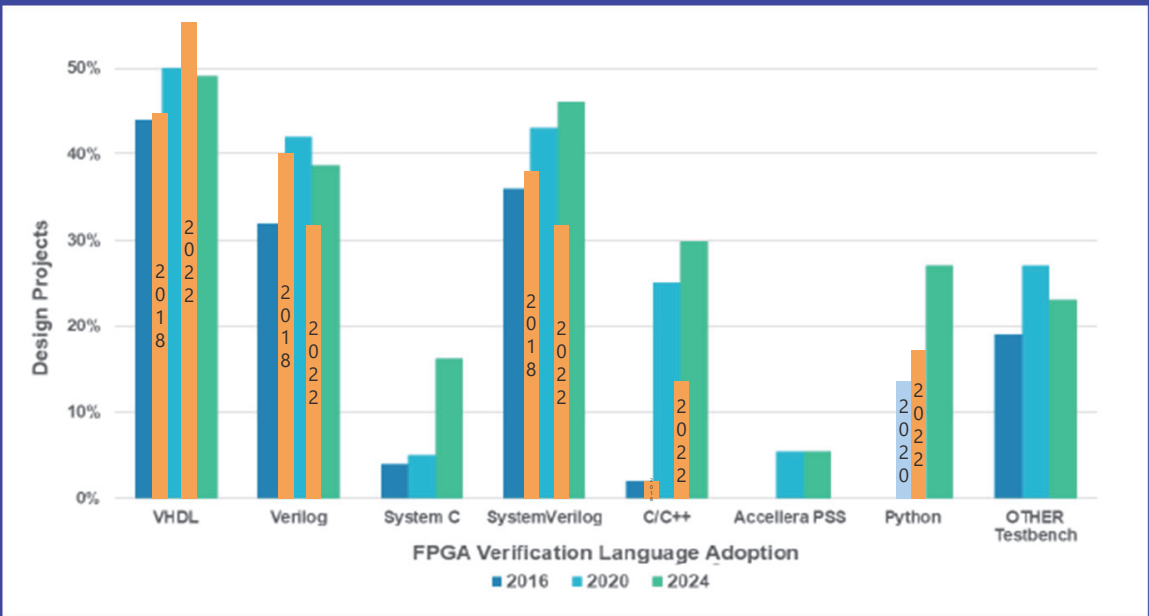
- Why VHDL? Why OSVVM?
- Why not create my own methodology.
- OSVVM Is a Complete Solution
- OSVVM Framework
- Model Independent Transactions
- Verificaiton Components
- Test Sequencer
- Scoreboards
- Functional Coverage
- Sharing Data Structures
- Requirements: Local vs. Tracked
- Reuse
- Scripts
- C Language Interface

Background

- About Jim Lewis
  - 30+ years: VHDL Design and Verification
  - 25+ 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

Why VHDL?

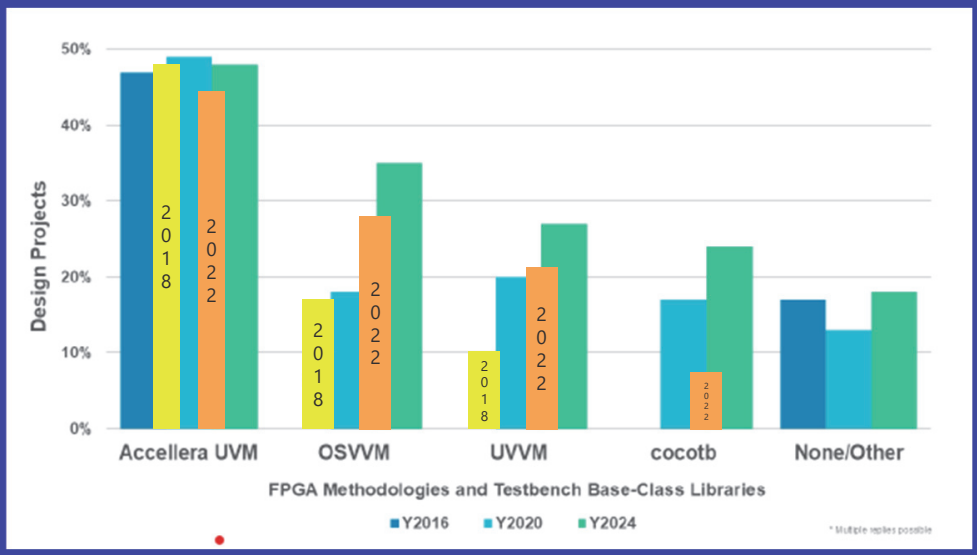
- VHDL is #1 for FPGA Design and Verification
  - For FPGA verification: 48% worldwide use VHDL.



- Graph from Wilson Research Group 2024 Functional Verification Survey
  © Siemens 2024 https://verificationacademy.com/topics/planning-measurement-and-analysis/2024-siemens-eda-and-wilson-research-group-functional-verification-study/

Why OSVVM?

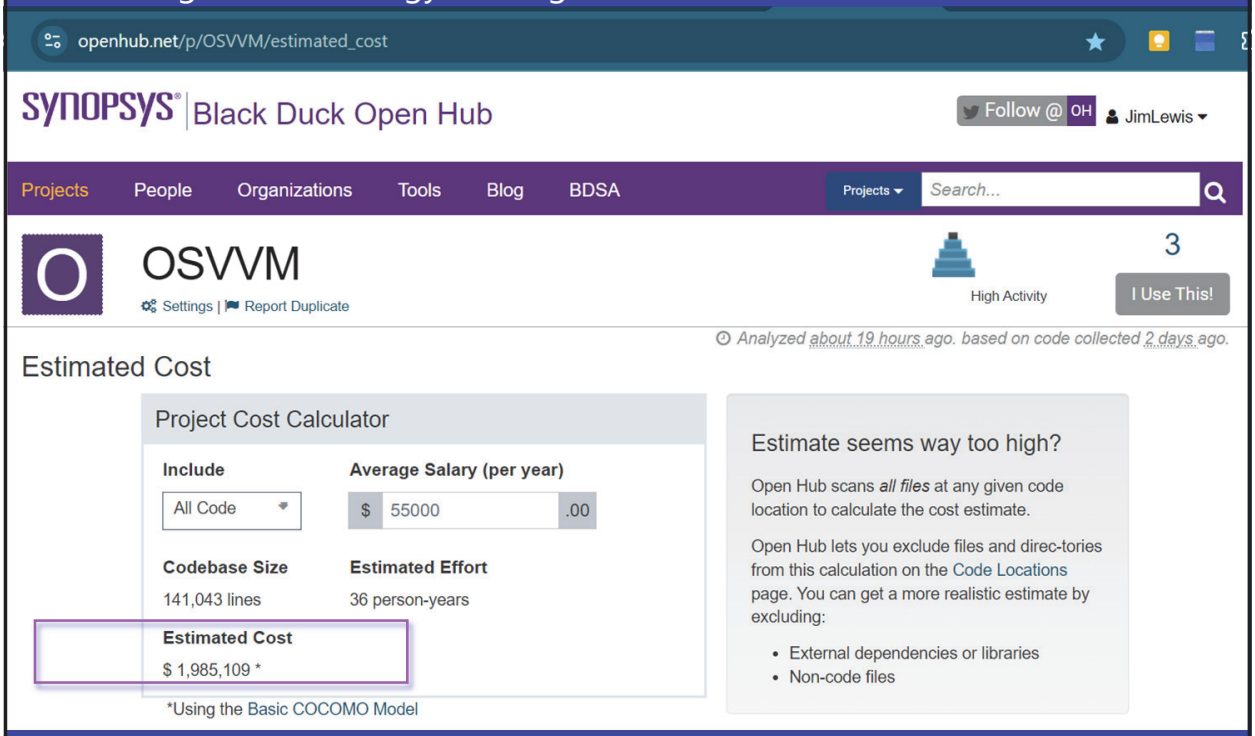
- OSVVM is VHDL's #1 Verification Methodology
  - For FPGA: 35% worldwide use OSVVM = 70% of VHDL users



- Graph from Wilson Research Group 2024 Functional Verification Survey
  © Siemens https://verificationacademy.com/topics/planning-measurement-and-analysis/2024-siemens-eda-and-wilson-research-group-functional-verification-study/

Why not create my own methodology?

- Creating a methodology is a large investment



- Data from openhub.net © Synopsys 2024

OSVVM is a Complete Solution

Verification Framework
Transaction Interface and API
Verification Components
Test Sequencer (Test Cases)

Verification Utility Library
Constrained Random, Scoreboards, Functional Coverage, Memory Models, Message Filtering, Error Tracking, ..., and Requirements Tracking

Verification Component Library
AXI4 Full, Lite, AXI Stream, UART, xMII, DPRam, ...

Script Library
Tool Independent Scripts
One Script to Run them All

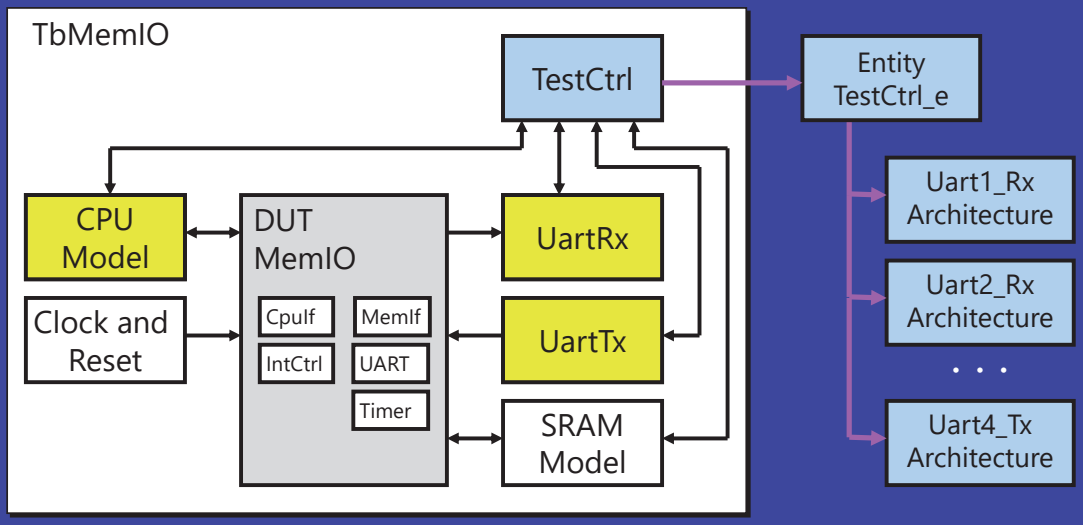
Co-Simulation
Run Software in Hardware Simulator
Write tests in C++

Test Reports
HTML Test Suite, Test Case, Log Files
JUnit XML for CI/CD tools

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

OSVVM's 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 ;
```

OSVVM is **not** OO based
- It plugs together just like RTL
- Any VHDL engineer can do this

DUT

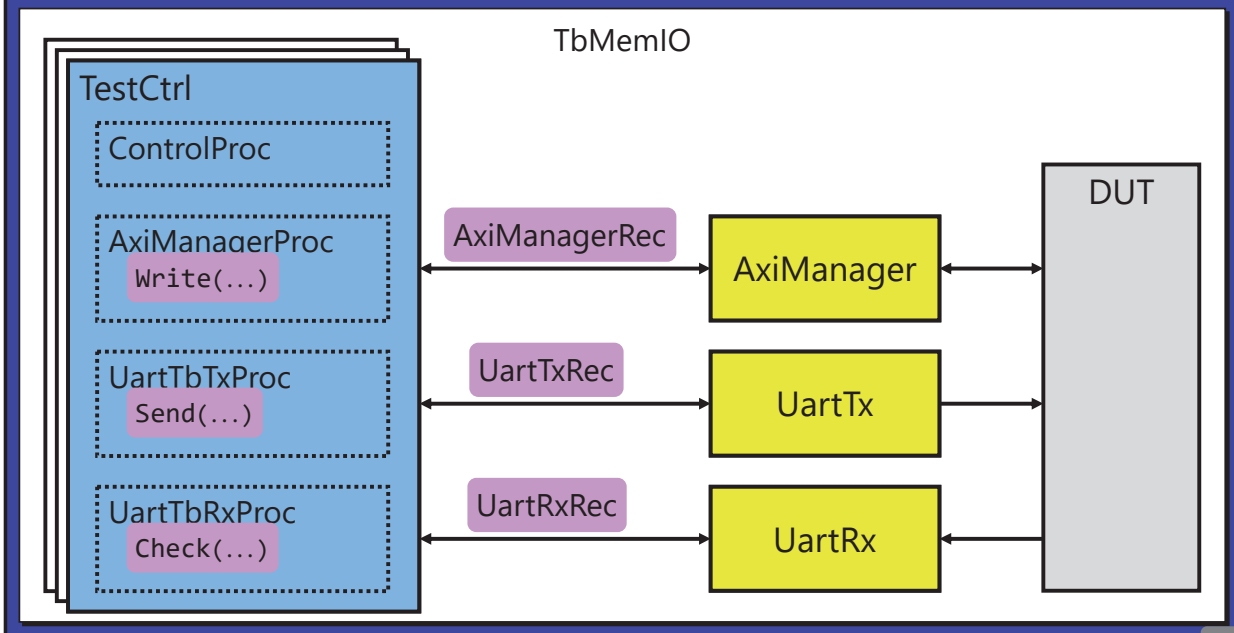
Verification Components

Test Sequencer

SynthWorks

Verification Framework - Elements

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



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
- Only requires a mode view declaration for the record

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.DataFromMod
end procedure Read ;
```

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

Model Independent Transaction Library

SynthWorks

- MIT defines the transaction Interface and API for . . .
  - … Address Bus Interfaces (AXI4, Wishbone, …) do read and write (subset)

```
type AddressBusRecType is record  . . . ;
Write(AddrRec, iAddr, iData) ;
Read (AddrRec, iAddr, oData) ;
ReadCheck(AddrRec, iAddr, iData) ;
. . .
```

  - … Streaming Interfaces (AxiStream, UART, …) do send and get (subset)

```
type StreamRecType is record  . . . ;
Send (TxRec, iData [, iParam]) ;
Get  (RxRec, oData [, oParam]) ;
Check(RxRec, iData [, iParam]) ;
. . .
```

All OSVVM VC use MIT.
Simplifies writing both VC and test cases. Supports Co-Sim.

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

Benefits: Writing VC is as easy as writing a procedure - plus a VC is concurrent = more capability than a procedure

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 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 add and mix in
  - Directed Tests
  - Constrained Random
  - Scoreboards
  - Functional Coverage
- Synchronization
- Error Reporting & Messaging

Test Initialization

SynthWorks

```
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 <= NewReqID("Req1") ;

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

  WaitForBarrier(TestDone, 5 ms) ;

  . . . -- Test Finalization
```

IDs for data structures

Set Test Name

Open Transcript File
+ Write to Console

Construct Data Structures
AlertLog, Requirements, and
Scoreboard

Enable Logs/Message Filters

Stop until Test Done or 5 ms
passes = WatchDog timer

A Test Case

SynthWorks

- 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");
  Get(RxRec, RxD) ;
  AffirmIfEqual(TBID, RxD, 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: Self-checking tests are easy. Readable. OSVVM tracks errors

SynthWorks

Scoreboards

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

```
TxProc : process
begin
  wait until Reset = '1';
  Push(SB, X"10") ;
  Send(TRec, X"10") ;
  . . .
  Push(SB, X"12") ;
  TestActive <= FALSE ... ;
  Send(TRec, X"12") ;
  . . .
  WaitForBarrier(TestDone) ;
end process TxProc ;
```

```
RxProc : process
  variable RcvD :
    std_logic_vector(7 downto 0);
begin

  while TestActive loop
    Get(RRec, RcvD) ;
    Check(SB, RcvD) ;
    -- Wait ??
  end loop ;
  WaitForBarrier(TestDone) ;
end process RxProc ;
```

Benefit:

- TxProc can change from directed to random without changing RxProc

SynthWorks

Functional Coverage

```
RxProc : process
  variable RxCov : CoverageIDType ;
  . . .
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, to_integer(RxOp));
  end loop ;
```

Coverage Object

Construct the Data Structure

Define coverage model

Collect Coverage

Functional Coverage

- Tracks that design requirements are met
- Is as simple and concise as language syntax

Sharing OSVVM Data Structures

SynthWorks

- All OSVVM data structures support a search/look up capability

Construct Data Structure in one entity

```
entity Axi4Memory is
. . .
Initialize : process
  variable MemID : MemoryIDType;
begin
  -- construct
  MemID := NewID(
    Name        => MEM_NAME,
    AddrWidth   => ADDR_WIDTH,
    DataWidth   => DATA_WIDTH,
    Search      => NAME   ) ;
  . . .
```

Look Up the Data Structure in another

```
architecture Test1 of TestCtrl is
. . .
MemTestProc : process
  variable MemID : MemoryIDType;
begin
  -- look up
  MemID := NewID(
    Name        =>  MEM_NAME,
    Addrwidth   =>  ADDR_WIDTH,
    DataWidth   =>  DATA_WIDTH,
    Search      =>  NAME ) ;

  MemRead (MemID, A1, D1) ;
  MemWrite(MemID, A1, D1+5) ;
  . . .
```

- Simplify SECDED Memory tests.
- Put scoreboards in different VC. . . .

Requirements Local vs. Tracked

SynthWorks

- OSVVM uses Affirmations and Functional Coverage for requirements
- Local Requirement: Create with NewID - impacts test case pass/fail

```
TbId   <= NewID("TB") ;
. . .
Cov1Id <= NewID("Cov1", TbId) ;
```

- Tracked Requirement: Create with NewReqID

```
ReqId    <= NewReqID("TB") ;
. . .
CovReqId <= NewReqID("Cov1") ;
```

- Tracked requirements are reported automatically in HTML and csv
  - Failed affirmations are a requirements failure
  - Affirmations that do not reach their requirement goal fail
  - Functional coverage that does not reach its goal is a requirements failure

Test Finalization

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

  TranscriptClose ;
  AffirmIfTranscriptsMatch("ValidatedResults") ;

  AffirmIf(TbID, osvvm.Scoreboard_slv.AllScoreboardsEmpty,
    "Scoreboards not empty") ;
  . . .  -- repeat for other scoreboard package instances used

  AffirmIf(TbID, AllCovered, "Coverage %" & to_string(GetCov)) ;

  EndOfTestReports (TimeOut => (Now >= 5 ms) ) ;
  std.env.stop ;
  wait ;
end process ControlProc ;
```

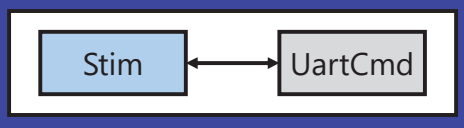
Stop until Test Done or 5 ms has passed

Create Reports

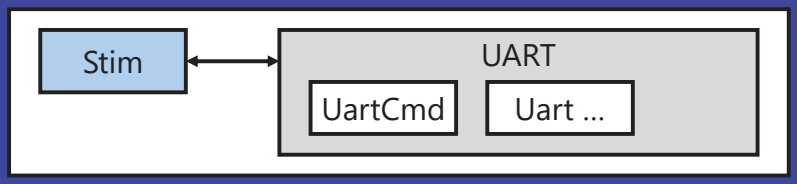
Testbench Architecture + Maximizing Reuse

- Historically a separate testbench is written for different levels of testing

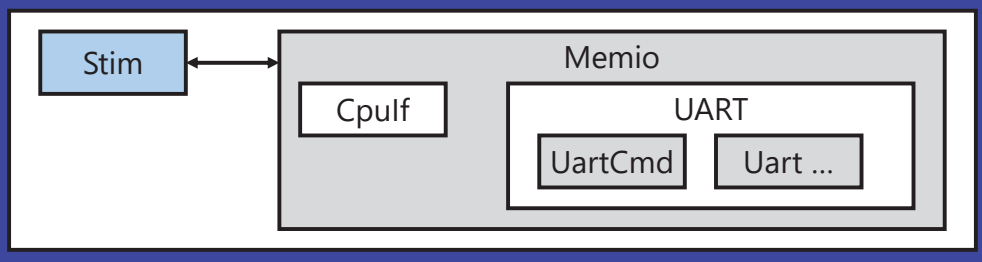
RTL



Core



Chip

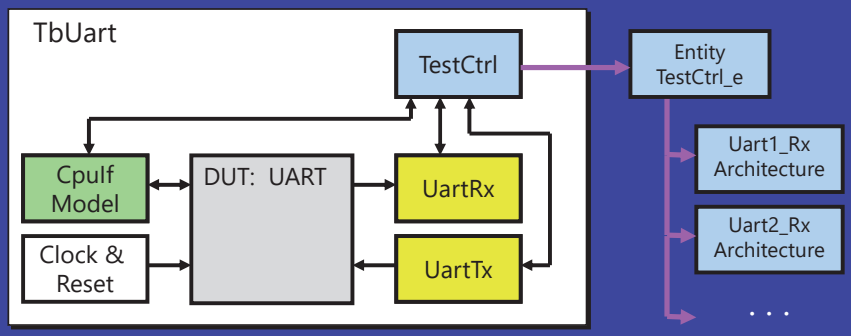


Issue: Each larger and more sophisticated with lots of duplicated effort

SynthWorks

Testbench Architecture + Maximizing Reuse

- To maximize reuse, use the same architecture for RTL, Core and Chip tests



- Step 1: Identify VC needed for Chip level first
- Step 2: For Core level tests,
  - Plan to Reuse/Pre-use Chip level VC as much as possible
  - Plan to Reuse test cases as much as possible
    - Transactions are the same
- Step 3: For RTL level tests, repeat step 2 reusing Core and Chip VC and tests

Benefit: Using the same testbench structure maximizes reuse

SynthWorks

Test Automation / Scripting

```
library  osvvm_TbUart

analyze  TestCtrl_e.vhd
analyze  TbUart.vhd

analyze  TbUart_SendGet1.vhd
simulate TbUart_SendGet1

RunTest  TbUart_SendGet2.vhd
. . .
```

Activate Library

Compile

Simulate & Run TbUart_SendGet1, TbUart_SendGet2

- Procedure based API that runs on top of TCL
- Benefits
  - Simple, just like a list of source files …
    - … Except we also get the power of TCL
- One Simulator Script that runs
  - GHDL, NVC, Aldec, Siemens, Synopsys VCS, Cadence Xcelium, XSIM

Benefit: Same scripts run other simulators - such as Open Source

SynthWorks

Co-Simulation

- Run C++ code in the hardware simulation
- For each MIT transaction there is a corresponding call in C

```
cosim.transRead(addr, &rdata32);
cosim.transWrite(addr, wdata32);
cosim.transReadCheck(addr, data32);

cosim.transBurstWrite(addr, 128);
cosim.transBurstRead(addr, 128);

cosim.transBurstPushData(wbuf, 128);
cosim.transBurstPopData(rbuf, 128);
```

- There is also an ISS that supports RISC-V
- OSVVM's CoSim capability was created by Simon Southwell and is based on his Virtual Processor (VProc) project.

SynthWorks

OSVVM Creates Unmatched Test Reports

- Test Completion Message - Text
- Build Summary
  - Mini-Report - Text (in simulator)
  - Detailed report - HTML
  - CI report - JUnit XML - for continuous integration tools
- Requirements Summary – HTML and CSV
- Test Case Reports
  - Alerts, Functional Coverage, and FIFOs/Scoreboards - HTML
  - Test Case Transcript - output of TranscriptOpen - text
- Simulation Transcript (aka log of simulator output) - HTML and text
- For more see: https://osvvm.github.io/Overview/Osvvm3Reports.html

Reports are generated by EndOfTestReports (in VHDL) + OSVVM Scripting

SynthWorks

Test Completion Message

- 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
```


SynthWorks

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

sim_RunDemoTestsWithCov Build Report

Build	sim_RunDemoTestsWithCov
Status	PASSED
PASSED	33
FAILED	0
SKIPPED	0
Analyze Failures	0
Simulate Failures	0
Start Time	2024-06-05 - 05:54:57 (PDT)
Finish Time	2024-06-05 - 05:57:24 (PDT)
Elapsed Time (h:m:s)	0:02:26
Simulator (Version)	RivieraPRO (2023.10.106.9105)
OSVVM Version	2024.05
Simulation Transcript	sim_RunDemoTestsWithCov.log
HTML Simulation Transcript	sim_RunDemoTestsWithCov_log.html
Requirements Summary	sim_RunDemoTestsWithCov_req.html
Code Coverage	Code Coverage Results

Created by Scripts + EndOfTestReports

Build Status

- Summarizes all tests run
- Simulator & Revision
- Link to Simulation Transcript
  - Both text and html
- Link to Requirements Report
- Link to code coverage

▼ **Test Suite Summary**

TestSuites	Status	Test Cases PASSED	Test Cases FAILED	Test Cases SKIPPED	Requirements passed / goal	Disabled Alerts	Elapsed Time
Axi4Full	PASSED	4	0	0	——	0	14.661
AxiStream	PASSED	4	0	0	——	0	14.473
Uart	PASSED	1	0	0	——	0	4.538
Ethernet	PASSED	1	0	0	——	0	4.965
AlertLogPkg	PASSED	14	0	0	9 / 9	0	42.832
RandomPkg	PASSED	2	0	0	——	0	6.441
CoveragePkg	PASSED	1	0	0	——	0	3.261
MemoryPkg	PASSED	3	0	0	——	0	11.057
ScoreboardPkg	PASSED	3	0	0	——	0	15.416

Test Suite Summaries

Test Suite = multiple test cases
Summarizes Pass/Fail+

▼ **Axi4Full Test Case Summary**

Test Case	Status	Checks Total	Checks Passed	Checks Failed	Requirements Goal	Requirements Passed	Functional Coverage	Disabled Alerts	Elapsed Time
TbAxi4_DemoMemoryReadWrite1	PASSED	334	334	0	——	——	43.75	0	2.806
TbAxi4_ManagerRandomTiming1	PASSED	512	512	0	——	——	100.00	0	1.383
TbAxi4_ManagerMemoryRandomTiming1	PASSED	512	512	0	——	——	100.00	0	1.678
TbAxi4_MemoryBurstPattern1	PASSED	114	114	0	——	——	100.00	0	1.480

▼ **AxiStream Test Case Summary**

Test Case	Status	Checks Total	Checks Passed	Checks Failed	Requirements Goal	Requirements Passed	Functional Coverage	Disabled Alerts	Elapsed Time
TbStream_SendGetDemo1	PASSED	341	341	0	——	——	66.67	0	1.539
TbStream_SendGetPacketBurst1	PASSED	133	133	0	——	——	50.00	0	1.508
TbStream_SendGetRandom1	PASSED	1616	1616	0	——	——	100.00	0	1.592
TbStream_SendGetRandom2	PASSED	6042	6042	0	——	——	100.00	0	1.301

Test Case Summaries

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

Build Summary Report with Errors

sim_OsvvmDemo Build Report

Build	sim_OsvvmDemo
Status	FAILED
PASSED	157
FAILED	3
SKIPPED	0
Analyze Failures	0
Simulate Failures	0
Start Time	2024-05-17 - 19:45:41 (PDT)
Finish Time	2024-05-17 - 19:54:48 (PDT)
Elapsed Time (h:m:s)	0:09:07
Simulator (Version)	RivieraPRO (2023.10.106.9105)
OSVVM Version	2024.0X-Dev
Simulation Transcript	sim_OsvvmDemo.log
HTML Simulation Transcript	sim_OsvvmDemo_log.html
Code Coverage	Code Coverage Results

▼ **Test Suite Summary**

TestSuites	Status	Test Cases PASSED	Test Cases FAILED	Test Cases SKIPPED	Requirements passed / goal	Disabled Alerts	Elapsed Time
Axi4Lite	PASSED	17	0	0	——	0	55.762
Axi4Full	FAILED	59	3	0	——	0	206.268
AxiStream	PASSED	65	0	0	——	0	206.805
Uart	PASSED	9	0	0	——	0	29.519
DpRam	PASSED	1	0	0	——	0	2.945
Ethernet	PASSED	6	0	0	——	0	20.324

▶ **Axi4Lite Test Case Summary**

▼ **Axi4Full Test Case Summary**

Test Case	Status	Checks Total	Checks Passed	Checks Failed	Requirements Goal	Requirements Passed	Functional Coverage	Disabled Alerts	Elapsed Time
TbAxi4_DemoMemoryReadWrite1	PASSED	334	334	0	——	——	43.75	0	1.383
TbAxi4_DemoErrorMemoryReadWrite1	FAILED	302	300	2	——	——	100.00	0	1.180
TbAxi4_BasicReadWrite	PASSED	60	60	0	——	——	100.00	0	1.094
TbAxi4_RandomReadWrite	PASSED	3000	3000	0	——	——	100.00	0	1.594
TbAxi4_RandomReadWriteByte1	PASSED	3000	3000	0	——	——	100.00	0	1.341
TbAxi4_SubordinateReadWrite1	PASSED	60	60	0	——	——	100.00	0	1.159
TbAxi4_SubordinateReadWrite2	PASSED	60	60	0	——	——	100.00	0	1.190
TbAxi4_SubordinateReadWrite3	PASSED	60	60	0	——	——	100.00	0	1.289

SynthWorks

Requirements Report

sim_RunAllTests Requirement Results

OSVVM

Requirement	TestName	Status	Requirements		Checks			Alert Counts			Disabled Alert Counts		
			Goal	Passed	Total	Passed	Failed	Failures	Errors	Warnings	Failures	Errors	Warnings
1.1	TbAlertLog_t18_requirements	PASSED	1	1	1	1	0	0	0	0	0	0	0
1.2	TbAlertLog_t18_requirements	PASSED	2	2	2	2	0	0	0	0	0	0	0
1.3	TbAlertLog_t18_requirements	PASSED	3	3	3	3	0	0	0	0	0	0	0
1.4	TbAlertLog_t18_requirements	PASSED	4	4	4	4	0	0	0	0	0	0	0
R1.1	Merged	PASSED	1	5	5	5	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
R1.2	Merged	PASSED	2	10	10	10	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
R1.3	Merged	FAILED	3	15	17	15	2	0	2	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
R1.4	Merged	FAILED	4	20	22	20	2	0	2	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
R2.1	TbAlertLog_t23_read_requirements	PASSED	2	2	2	2	0	0	0	0	0	0	0
R2.2	TbAlertLog_t23_read_requirements	PASSED	2	2	2	2	0	0	0	0	0	0	0
R2.3	TbAlertLog_t23_read_requirements	PASSED	2	2	2	2	0	0	0	0	0	0	0
R2.4	TbAlertLog_t23_read_requirements	PASSED	2	2	2	2	0	0	0	0	0	0	0
R3.1	TbAlertLog_t23_read_requirements	PASSED	1	1	1	1	0	0	0	0	0	0	0
R3.2	TbAlertLog_t23_read_requirements	PASSED	1	1	1	1	0	0	0	0	0	0	0
R3.3	TbAlertLog_t23_read_requirements	PASSED	1	1	1	1	0	0	0	0	0	0	0
R3.4	TbAlertLog_t23_read_requirements	PASSED	1	1	1	1	0	0	0	0	0	0	0
R4.1	TbAlertLog_t23_read_requirements	PASSED	1	1	1	1	0	0	0	0	0	0	0
R4.2	TbAlertLog_t23_read_requirements	FAILED	2	2	4	2	2	0	2	0	0	0	0

SynthWorks

Test Case Report

TbAxi4_DemoMemoryReadWrite1 Test Case Report

Available Reports

- Alert Report
- Functional Coverage Report(s)
- ScoreboardPkg_slv Report(s)
- Link to Simulation Results
- TbAxi4_DemoMemoryReadWrite1.txt
- sim_OsvvmDemo Build Summary

OSVVM

TbAxi4_DemoMemoryReadWrite1 Alert Report

▶ TbAxi4_DemoMemoryReadWrite1 Alert Settings

▶ TbAxi4_DemoMemoryReadWrite1 Alert Results

TbAxi4_DemoMemoryReadWrite1 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

TbAxi4_DemoMemoryReadWrite1 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

Links

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

Alert Report

Settings (hidden)
Results (hidden)

Functional Coverage Report

Report for each FC model in testbench (each hidden)

Scoreboard Report

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

SynthWorks

Test Case Report: Alert Report

TbAxi4_DemoMemoryReadWrite1 Alert Report

▼ **TbAxi4_DemoMemoryReadWrite1 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

▼ **TbAxi4_DemoMemoryReadWrite1 Alert Results**

Name	Status	Checks			Requirements		Alert Counts			Disabled Alert Counts		
		Total	Passed	Failed	Goal	Passed	Failures	Errors	Warnings	Failures	Errors	Warnings
TbAxi4_DemoMemoryReadWrite1	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
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
memory_1:memory	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
WriteBurstFifo	PASSED	0	0	0	0	0	0	0	0	0	0	0
ReadBurstFifo	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
Protocol Error	PASSED	0	0	0	0	0	0	0	0	0	0	0
Data Check	PASSED	16	16	0	0	0	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

Alert Report

SynthWorks

Test Case Report: Alert Report with Errors

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			Requirements		Alert Counts			Disabled Alert Counts		
		Total	Passed	Failed	Goal	Passed	Failures	Errors	Warnings	Failures	Errors	Warnings
TbAxi4_DemoErrorMemoryReadWrite1	FAILED	302	300	2	0	0	0	2	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
memory_1	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
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
WriteBurstFifo	PASSED	0	0	0	0	0	0	0	0	0	0	0
ReadBurstFifo	PASSED	0	0	0	0	0	0	0	0	0	0	0
manager_1	FAILED	0	0	2	0	0	0	2	0	0	0	0
Protocol Error	PASSED	0	0	0	0	0	0	0	0	0	0	0
Data Check	FAILED	16	15	1	0	0	0	1	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	92	91	1	0	0	0	1	0	0	0	0
Testbench	PASSED	20	20	0	0	0	0	0	0	0	0	0

Test Case Report: Functional Coverage

Uart7_Random_part3 Coverage Report

Total Coverage: 100.00

▼ **UART_RX_STIM_COV Coverage Model** **Coverage: 100.0**

▼ **UART_RX_STIM_COV Coverage Settings**

Settings	Value
CovWeight	1
Goal	100.0
WeightMode	REMAIN
Seeds	1881697261, 1078732634
CountMode	COUNT_FIRST
IllegalMode	ILLEGAL_ON
Threshold	45.0
ThresholdEnable	FALSE
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	0 to 255	0	63	63	100.0
NORMAL	COUNT	1	0 to 255	1 to 15	7	7	100.0
PARITY	COUNT	3	0 to 255	2 to 15	11	11	100.0
STOP	COUNT	5	1 to 255	2 to 15	11	11	100.0
PARITY_STOP	COUNT	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**

Name	Type	Mode	Count	AtLeast	Percent Coverage

HTML'ized Simulation Log File 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
```

Simulation Transcript

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

HTML'ized Simulation Log – Test Case Information

```
▼ RunTest TbAxi4_DemoErrorMemoryReadWrite1.vhd

TestName TbAxi4_DemoErrorMemoryReadWrite1
analyze TbAxi4_DemoErrorMemoryReadWrite1.vhd
vcom -2008 -relax -work osvvm_TbAxi4 C:/OsvvmLibraries/AXI4/Axi4/TestCases/TbAxi4_DemoErrorMemoryReadWrite1.vhd
ACOM: Compile Architecture "DemoErrorMemoryReadWrite1" of Entity "TestCtrl"
ACOM: Compile Configuration "TbAxi4_DemoErrorMemoryReadWrite1"
ACOM: Compile success 0 Errors 0 Warnings  Analysis time :  0.5 [s]
ACDB: Closing Code Coverage session.
VSIM: Simulation has finished.
Simulation Start time 18:25:07
simulate TbAxi4_DemoErrorMemoryReadWrite1
vsim -interceptcoutput -t ps -lib osvvm_TbAxi4 TbAxi4_DemoErrorMemoryReadWrite1 -acdb_cov sbm -cc_all
ELBREAD: Elaboration process.
ELBREAD: Elaboration time 0.0 [s].
Main thread initiated.
Kernel process initialization phase.
ELAB2: Elaboration final pass...
PLI/VHPI kernel's engine initialization done.
PLI: Loading library 'C:\tools\Aldec\Riviera-PRO-2023.10-x64\bin\systf.dll'
VHPI: Loading library 'systf.dll'
ELAB2: Create instances ...
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 process initialization done.
Allocation: Simulator allocated 13246 kB (elbread=427 elab2=12300 kernel=518 sdf=0)
ASDB file was created in location C:/tools/sim/riviera/dataset.asdb
%%          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 Data.  WData: 00000001  WStrb: 1111  Operation# 1

. . .

%%         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    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 Data.  WData: 00002001  WStrb: 1111  Operation# 33
%%         1070 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 00002000  AWProt: 000  Operation# 33
%%         1080 ns    Log     INFO      in manager_1,                        Write Data.  WData: 00002002  WStrb: 1111  Operation# 34
%%         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    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 Data.  WData: 0001UUUU  WStrb: 1100  Operation# 49
%%         1400 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 00003000  AWProt: 000  Operation# 34
%%         1410 ns    Log     INFO      in manager_1,                        Write Data.  WData: 00000003  WStrb: 1111  Operation# 50
```

Simulation Transcript

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

VHDL Transcript File

```
%%          110 ns    Log     PASSED    in Default,                          Default BurstMode is ADDRESS_
%%          110 ns    Log     PASSED    in Default,                          BurstMode Received : 0
%%          110 ns    Log     ALWAYS    in Default,                          Write and Read. Addr = 0000.
%%          110 ns    Log     INFO      in manager_1,                        Write Data.  WData: 00000001
%%          110 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 0000
%%          120 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 00000010  AWProt: 000  WData: 00000001  WStrb: 1111  Operation# 0
%%          120 ns    Log     INFO      in manager_1,                        Write Data.  WData: 00000002  WStrb: 1111  Operation# 2
%%          120 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 00000020  AWProt: 000  Operation# 2
%%          130 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 1
%%          130 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 00000020  AWProt: 000  WData: 00000002  WStrb: 1111  Operation# 1
%%          130 ns    Log     INFO      in manager_1,                        Write Data.  WData: 00000003  WStrb: 1111  Operation# 3
%%          130 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 00000030  AWProt: 000  Operation# 3
%%          140 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 2
%%          140 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 00000030  AWProt: 000  WData: 00000003  WStrb: 1111  Operation# 2
%%          140 ns    Log     INFO      in manager_1,                        Write Data.  WData: 00000004  WStrb: 1111  Operation# 4
%%          140 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 00000040  AWProt: 000  Operation# 4
%%          150 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 3
%%          150 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 00000040  AWProt: 000  WData: 00000004  WStrb: 1111  Operation# 3
%%          150 ns    Log     INFO      in manager_1,                        Write Data.  WData: 00000005  WStrb: 1111  Operation# 5
%%          150 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 00000050  AWProt: 000  Operation# 5
%%          160 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 4
%%          160 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 00000050  AWProt: 000  WData: 00000005  WStrb: 1111  Operation# 4
%%          160 ns    Log     INFO      in manager_1,                        Write Data.  WData: 00000006  WStrb: 1111  Operation# 6
%%          160 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 00000060  AWProt: 000  Operation# 6
%%          170 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 5
%%          170 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 00000060  AWProt: 000  WData: 00000006  WStrb: 1111  Operation# 5
%%          170 ns    Log     INFO      in manager_1,                        Write Data.  WData: 00000007  WStrb: 1111  Operation# 7
%%          170 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 00000070  AWProt: 000  Operation# 7
%%          180 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 6
%%          180 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 00000070  AWProt: 000  WData: 00000007  WStrb: 1111  Operation# 6
%%          180 ns    Log     INFO      in manager_1,                        Write Data.  WData: 00000008  WStrb: 1111  Operation# 8
%%          180 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 00000080  AWProt: 000  Operation# 8
%%          190 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 7
%%          190 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 00000080  AWProt: 000  WData: 00000008  WStrb: 1111  Operation# 7
%%          190 ns    Log     INFO      in manager_1,                        Write Data.  WData: 00000009  WStrb: 1111  Operation# 9
%%          190 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 00000090  AWProt: 000  Operation# 9
%%          200 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 8
%%          200 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 00000090  AWProt: 000  WData: 00000009  WStrb: 1111  Operation# 8
%%          200 ns    Log     INFO      in manager_1,                        Write Data.  WData: 0000000A  WStrb: 1111  Operation# 10
%%          200 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 000000A0  AWProt: 000  Operation# 10
%%          210 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 9
%%          210 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 000000A0  AWProt: 000  WData: 0000000A  WStrb: 1111  Operation# 9
%%          210 ns    Log     INFO      in manager_1,                        Write Data.  WData: 0000000B  WStrb: 1111  Operation# 11
%%          210 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 000000B0  AWProt: 000  Operation# 11
%%          220 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 10
%%          220 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 000000B0  AWProt: 000  WData: 0000000B  WStrb: 1111  Operation# 10
%%          220 ns    Log     INFO      in manager_1,                        Write Data.  WData: 0000000C  WStrb: 1111  Operation# 12
%%          220 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 000000C0  AWProt: 000  Operation# 12
%%          230 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 11
%%          230 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 000000C0  AWProt: 000  WData: 0000000C  WStrb: 1111  Operation# 11
%%          230 ns    Log     INFO      in manager_1,                        Write Data.  WData: 0000000D  WStrb: 1111  Operation# 13
%%          230 ns    Log     INFO      in manager_1,                        Write Address.  AWAddr: 000000D0  AWProt: 000  Operation# 13
%%          240 ns    Log     PASSED    in manager_1: WriteResponse Scoreboard,  Received: 0   Item Number: 12
%%          240 ns    Log     INFO      in memory_1,                         Memory Write.  AWAddr: 000000D0  AWProt: 000  WData: 0000000D  WStrb: 1111  Operation# 1
%%          240 ns    Log     INFO      in manager_1,                        Write Data.  WData: 0000000E  WStrb: 1111  Operation# 14
```

Created by TranscriptOpen
Test Case Report links to this file

Getting OSVVM

- Get the sources from GitHub:

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

  - Use the recursive flag since OsvvmLibraries has submodules

- Get the sources as a *.zip from osvvm.org

```
https://osvvm.org/downloads
```

  - You will need to register first and then login

- Initialize the simulator – see Documentation/Scripts_user_guide.pdf

```
source <PathToOsvvm>/OsvvmLibraries/Scripts/StartUp.tcl
```

- Build all OSVVM and Run All VC Tests

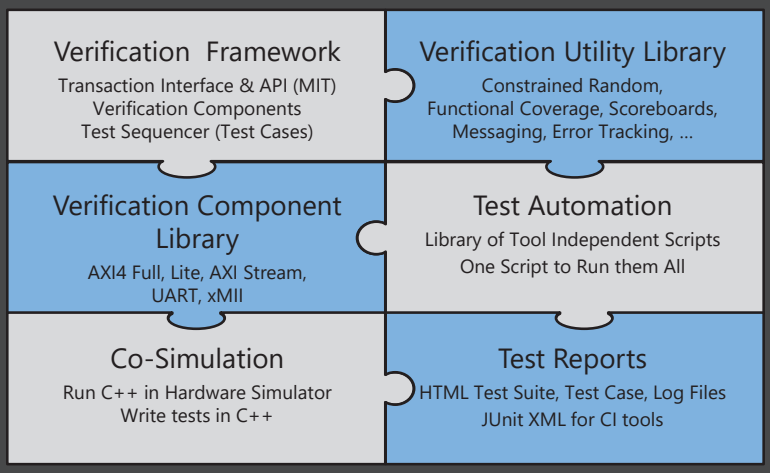
```
build $OsvvmLibraries/OsvvmLibraries.pro
build $OsvvmLibraries/RunDemoTests.pro
```

  - Each VC has a RunDemoTests and RunAllTests

More About OSVVM

- OSVVM uses the Apache License. You may …
- OSVVM Documentation
  - PDF: OsvvmLibraries/Documentation - part of release files
  - HTML: https://osvvm.github.io/Overview/Osvvm1About.html
- For OSVVM questions ask at:
  - https://osvvm.org/forums/forum/os-vvm
- For OSVVM enhancement requests see:
  - https://github.com/OSVVM/OsvvmLibraries/issues
  - Each submodule in OSVVM has its own issue page.
- Jump start your VHDL verification effort with training
  - Advanced VHDL Testbenches and Verification – OSVVM Boot Camp
  - https://synthworks.com/vhdl_testbench_verification.htm

All you need is … OSVVM



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