Faster than "Lite" Verification Component Development with OSVVM

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

1

Faster than "Lite" VC Dev with OSVVM

Copyright © 2020 - 2023 by SynthWorks Design Inc.
Distributing a pdf version of this document in whole for individual usage is permitted.
Printing a paper version of this document in whole for individual usage is permitted.
All other rights reserved.

In particular, without express written permission of SynthWorks Design Inc,
You may not distribute powerpoint versions of this work,
You may not alter, transform, or build upon this work,
You may not use any material from this guide in a group presentation,
tutorial, training class, or classroom,
You must include this page in any printed copy of this document.

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

Copyright © 2022 - 2023 SynthWorks Design Inc.

2

SynthWorks

Faster than "Lite" VC Dev with OSVVM

- Agenda
  - Why VHDL?
  - Why OSVVM?
  - What is OSVVM?
  - OSVVM Verification Framework
  - Walk-through of OSVVM's Faster than "Lite" VC Approach
  - Architecture of a non-blocking VC

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

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

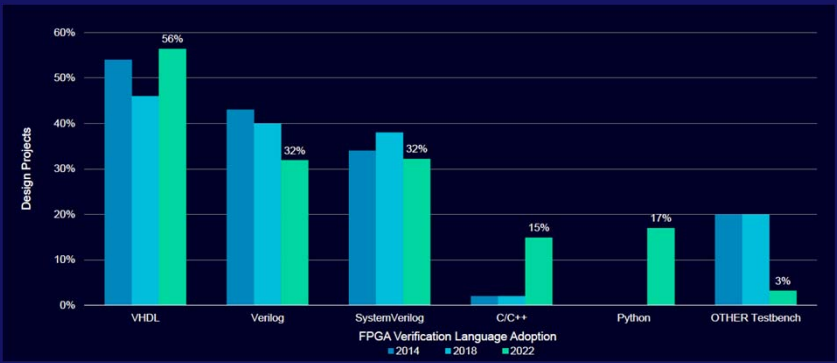
OSVVM is developed by the same VHDL experts who have helped develop VHDL standards.

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

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



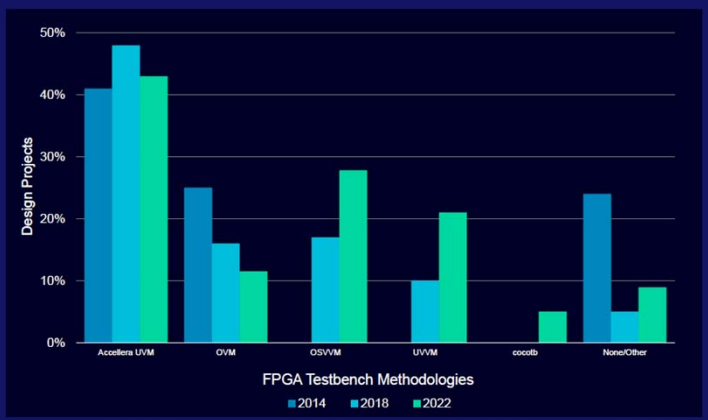
- © Siemens 2022 https://blogs.sw.siemens.com/verificationhorizons/2022/11/21/part-6-the-2022-wilson-research-group-functional-verification-study/

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Why OSVVM?

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

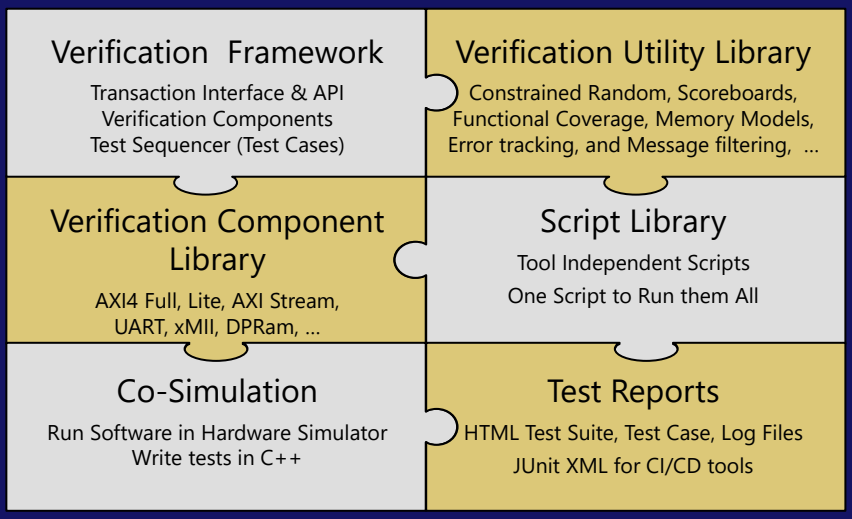


- © Siemens 2022 https://blogs.sw.siemens.com/verificationhorizons/2022/11/21/part-6-the-2022-wilson-research-group-functional-verification-study/

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

What is OSVVM?



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

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

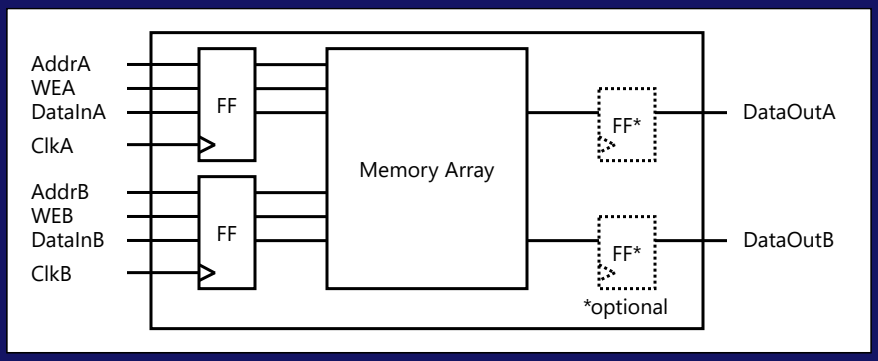
OSVVM History

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 AXI VC with delay randomization for xValid and xReady

SynthWorks

DUT: DPRAM

- DPRAM is similar to an FPGA BRAM



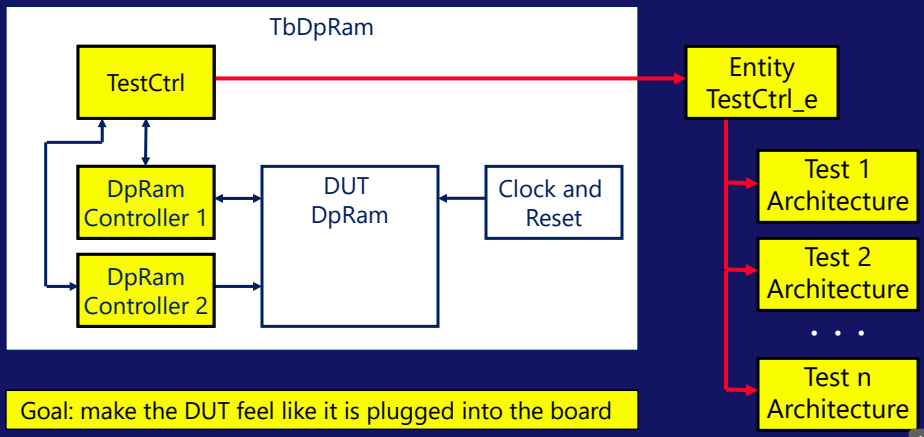
- DPRAM has two identical interfaces to access the memory
  - DpRamController VC supports a single interface
  - Testbench uses 2 DpRamController VC to support testing

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Verification Framework

- OSVVM 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



Copyright © 2022 - 2023 SynthWorks Design Inc.

Framework Implementation

SynthWorks

```
library osvvm, osvvm_Axi4 ;
  context osvvm.OsvvmContext ;
...
entity  TbDpRam  is
end entity TbDpRam ;
architecture TestHarness of TbDpRam is
  . . .
  signal Controller1Rec, Controller2Rec : AddressBusRecType (
      Address      (Address'length-1 downto 0),
      DataToModel  (iData'length-1 downto 0),
      DataFromModel(oData'length-1 downto 0)
    ) ;
begin
  osvvm.TbUtilPkg.CreateClock(Clk, tperiod_Clk) ;
  osvvm.TbUtilPkg.CreateReset(nReset, . . .) ;

  DpRam_1: DpRam ( . . . ) ;

  DpRamCtrl_1 : DpRamController (Controller1Rec, . . . ) ;
  DpRamCtrl_2 : DpRamController (Controller2Rec, . . . ) ;

  TestCtrl_1 : TestCtrl (Controller1Rec, Controller2Rec, nR
end Test1 ;
```

Structural Code
It is simple - just like RTL

DUT

Verification Components

Test Sequencer

Copyright © 2022 - 2023 SynthWorks Design Inc.

TestCtrl = Test Sequencer

SynthWorks

```
entity TestCtrl is
port (
  ControllerRec1 : InOut AddressBusRecType;
  ControllerRec2 : InOut AddressBusRecType;

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

Ports =
Transaction Interface
(Record Types)

Copyright © 2022 - 2023 SynthWorks Design Inc.

```
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

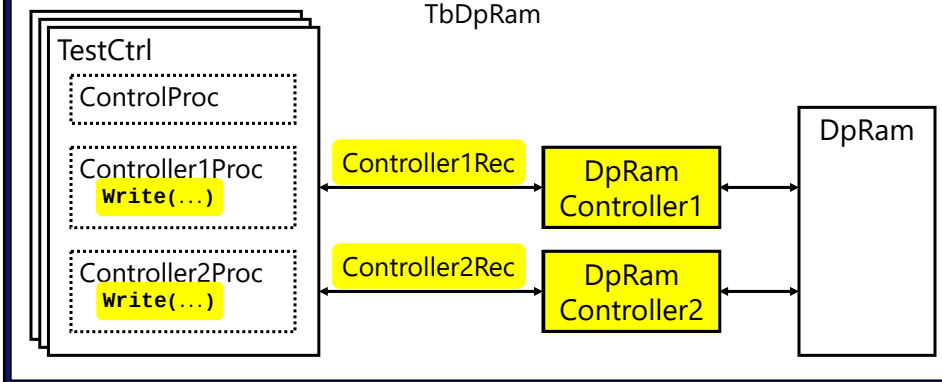
13

13

SynthWorks

3 Steps to Verification Component Development

- Step 1: Transaction Interface (Controller1Rec, Controller2Rec)
- Step 2: Transaction API (Write, Read, …)
- Step 3: Verification components



Copyright © 2022 - 2023 SynthWorks Design Inc.

14

14

SynthWorks

Step 1: 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

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Step 2: 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
end procedure Read ;
```

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

Copyright © 2022 - 2023 SynthWorks Design Inc.

Step 3: Verification Components

SynthWorks

```
entity DpRamController is
generic (
  MODEL_ID_NAME    :  string := "" ;
  DEFAULT_DELAY    :  time   := 1 ns ;
  tpd_Clk_Address  :  time   := DEFAULT_DELAY ;
  tpd_Clk_Write    :  time   := DEFAULT_DELAY ;
  tpd_Clk_oData    :  time   := DEFAULT_DELAY
) ;
port (
  Clk         : In    std_logic ;
  nReset      : In    std_logic ;

  -- DpRam Functional Interface
  Address     : Out   std_logic_vector ;
  Write       : Out   std_logic ;
  oData       : Out   std_logic_vector ;
  iData       : In    std_logic_vector ;

  -- Address Bus Transaction Interface
  TransRec    : InOut AddressBusRecType
) ;
. . .
end entity DpRamController ;
```

DUT Interface

Transaction Interface

17

17

Step 3: Verification Components

SynthWorks

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

  -- Decode and execute the transaction
  case TransRec.Operation is
    when WRITE_OP =>
      DpcWrite(TransRec.Address, TransRec.Data, …);
    when READ_OP =>
      DpcRead (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 an RTL designer

18

18

SynthWorks

Simplifying VC Development

- 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]) ;
. . .
```

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Benefits of OSVVM MIT

- Accelerates VC Development, Test Writing, and Documentation
  - Verification Component Developers
    - Re-use the MIT defined transaction interface and API
    - Focus on writing VC behavior
  - Test Writers
    - Already know most of the transaction interface
    - Can reuse sequences (subprograms) written for other VC
  - Co-Simulation Interface
    - Supports all MIT based VC - including ones you write
  - Documentation
    - A VC only needs to identify which transactions it supports
- More Information is in:
  - Address_Bus_Model_Independent_Transactions_user_guide.pdf
  - Stream_Model_Independent_Transactions_user_guide.pdf

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

"Faster than Lite" VC Development

- Approach
  - Write a simple blocking VC
  - Write functional design tests
  - Write basic interface tests
  - May need to upgrade the VC to support complex interface tests
- Blocking vs. Non-blocking
  - A blocking API call waits until the transaction completes and then returns
  - A non-blocking API call queues the transaction and returns
  - MIT supports both
- Non-blocking is required for interfaces that support simultaneous activity
- Benefits
  - One Verification Framework that runs all tests
  - Start simple, deliver initial VC sooner
  - Start writing test cases sooner
  - An upgraded VC will still run the test cases written for the blocking VC

Copyright © 2022 - 2023 SynthWorks Design Inc.

21

SynthWorks

"Faster than Lite" VC Development

- Steps
  - Identify Transactions
  - Look up Transactions in MIT
  - Package References
  - Create AlertLogIDs
  - Implement the TransactionHandler with code for transactions

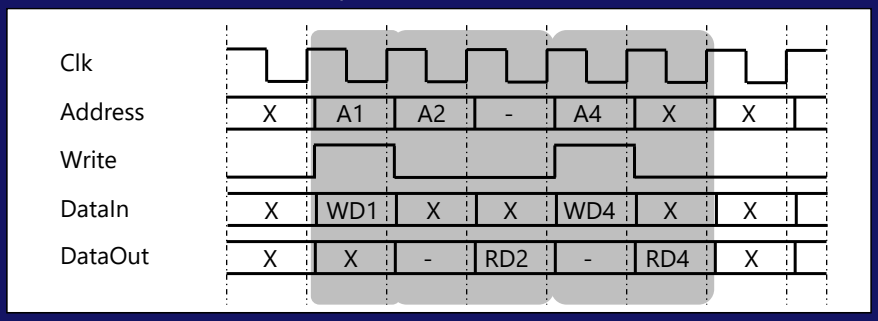
Copyright © 2022 - 2023 SynthWorks Design Inc.

22

SynthWorks

Identify Transactions

- DPRAM Interface characteristics
  - Write cycles complete in a single cycle
  - Read cycles complete in two (to three) cycles
  - Supports Write and Read simultaneously
    - Read data is from a previous write.



Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Look Up Transactions in MIT

- Identify the Interface Transaction API calls

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

- Look up Operation value in MIT User Guide

9.2 **Basic Interface Independent Transactions**

AddressBusOperationType Value	Manager Transaction Name	Subordinate Transaction Name
WRITE_OP	Write	GetWrite
READ_OP	Read	SendRead
READ_CHECK	ReadCheck	
WRITE_AND_READ	WriteAndRead	

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Look Up Transactions in MIT

- Identify Directive Transactions

```
WaitForClock       (TransRec, iNumberOfClocks) ;
GetAlertLogID      (TransRec, oAlertLogID) ;
GetTransactionCount(TransRec, oCount) ;
```

- Look up Operation value in MIT User Guide

9.4.1 **Common Directive Transactions**

AddressBusOperationType Value	Manager Transaction Name	Subordinate Transaction Name
WAIT_FOR_CLOCK	WaitForClock	WaitForClock
GET_TRANSACTION_COUNT	GetTransactionCount	GetTransactionCount
GET_ALERTLOG_ID	GetAlertLogID	GetAlertLogID

Copyright © 2022 - 2023 SynthWorks Design Inc.

25

SynthWorks

Verification Component Architecture

Architecture Model of DpRamController is

Create AlertLogIDs	• Associates messages and errors with the VC
Preprocess Inputs	• Remove H and L from inputs
Transaction Handler / Transaction Dispatcher	• Decode transactions • Handlers also execute transactions
[Interface Handler]	• Execute* transactions
Protocol Checker	• Detect interface signaling errors
Timing Checker	• Check timing
Log Results	• Print interface values

Copyright © 2022 - 2023 SynthWorks Design Inc.

26

Package References for VC

SynthWorks

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

library osvvm ;
  context osvvm.OsvvmContext ;

library osvvm_common ;
  context osvvm_common.OsvvmCommonContext ;
```

OSVVM Utility Library

MIT Library

Copyright © 2022 - 2023 SynthWorks Design Inc.

27

Verification Component Entity

SynthWorks

```
entity DpRamController is
generic (
  MODEL_ID_NAME    :  string := "" ;
  . . .
) ;
port (
  Clk         : In    std_logic ;
  nReset      : In    std_logic ;

  -- DpRam Functional Interface
  Address     : Out   std_logic_vector ;
  Write       : Out   std_logic ;
  oData       : Out   std_logic_vector ;
  iData       : In    std_logic_vector ;

  -- Address Bus Transaction Interface
  TransRec    : InOut AddressBusRecType
) ;

constant RESOLVED_MODEL_ID_NAME : string :=
  IfElse(MODEL_ID_NAME'length > 0, MODEL_ID_NAME,
    to_lower(PathTail(DpRamController'PATH_NAME)));

end entity DpRamController ;
```

Model Name (Optional)

DUT Interface

Transaction Interface

Resolved Model Name

28

SynthWorks

Create AlertLogIDs

- AlertLogIDs associate messages, error counts, and control with a VC

```
signal ModelID, ErrorID, CheckerID : AlertLogIDType ;
. . .
Initialize : process
  variable ID : AlertLogIDType ;
begin
  ID         := NewID(RESOLVED_MODEL_ID_NAME );
  ModelID    <= ID ;
  DataCheckID     <= NewID("Data", ID) ;
  ProtocolCheckID <= NewID("Protocol", ID) ;
  wait ;
end process Initialize ;
```

Create ID

Create Child ID

  - An ID references storage in the AlertLog data structure (singleton)
  - To change settings, the test sequencer looks up the ID of a VC by name

- The name associated with an ID prints with messages

```
%% Log   INFO   In DpRamController_1, Read Address: A0004  Operation#
5  Data: Received: 0004  at 2150 ns
```

Copyright © 2022 - 2023 SynthWorks Design Inc.

29

29

SynthWorks

Handling Interface Initializations

```
TransactionHandler : process
begin
  -- Initialize Outputs
  Address     <= (Address'range  => 'X') ;
  Write       <= 'X' ;
  oData       <= (oData'range => 'X') ;

  loop
    WaitForTransaction (. . .) ;

    -- Decode and execute the transaction
    case TransRec.Operation is
      when WRITE_OP =>          . . .
      when READ_OP =>           . . .
      when WRITE_READ_OP =>     . . .
      when . . . =>     -- Other Transactions
    end case ;
  end loop ;
end process TransactionHandler ;
```

Initialize Outputs

Handler now inside loop

Copyright © 2022 - 2023 SynthWorks Design Inc.

30

30

Wait For Transaction

SynthWorks

```
TransactionHandler : process
begin
  -- Initialize Outputs
  . . .
  loop
    WaitForTransaction (
      Clk => Clk,
      Rdy => TransRec.Rdy,
      Ack => TransRec.Ack
    ) ;
    -- Decode and execute the transaction
    case TransRec.Operation is
      when WRITE_OP =>          . . .
      when READ_OP =>           . . .
      when . . . =>    -- Other Transactions
    end case ;
  end loop ;
end process TransactionHandler ;
```

Wait for Transaction
- Finds a transaction
- Aligns to clock if not aligned

Copyright © 2022 - 2023 SynthWorks Design Inc.

Read And Check Operation

SynthWorks

```
case TransRec.Operation is
  when READ_OP | READ_CHECK =>
    Address <= SafeResize(TransRec.Address,
                          Address'length) after tpd_Clk_Address ;
    Write   <= '0' after tpd_Clk_Write ;
    WaitForClock(Clk) ;

    Address <= not Address after tpd_Clk_Address ;
    WaitForClock(Clk) ;
    if (USE_OUTPUT_READ_REGISTER) then
      WaitForClock(Clk) ;
    end if ;
    TransRec.DataFromModel <= SafeResize(iData,
                          TransRec.DataFromModel'length) ;

    if IsReadCheck(Operation) then
      -- Checking and Logging on Following Slides
      . . .
```

Read Cycle

Wait for read data

Copyright © 2022 - 2023 SynthWorks Design Inc.

Read And Check Operation

SynthWorks

```
if IsReadCheck(Operation) then
  . . .
else  -- Read Operation
  Log( ModelID,
    "Read, Address: " & to_hxstring(Address) &
    "  Data: " & to_hxstring(iData) &
    "  Operation# " & to_string (TransRec.Rdy),
    INFO, TransRec.StatusMsgOn
  ) ;
end if ;
```

Log = Conditional Printing

- Logs print when enabled and are disabled by default
  - Control is either global or for just this ModelID
  - Logs have levels ALWAYS, INFO, DEBUG, PASSED, and FINAL
  - If either log level INFO is enabled or TransRec.StatusMsgOn is TRUE then

```
%% Log   INFO   In DpRamController_1, Read Address: A0004  Operation#
5  Data: Received: 0004  at 2150 ns
```

Copyright © 2022 - 2023 SynthWorks Design Inc.

Read And Check Operation

SynthWorks

```
if IsReadCheck(Operation) then
  ExpectedData  := SafeResize(TransRec.DataToModel,
                              ExpectedData'length) ;
  AffirmIfEqual(DataCheckID, iData, ExpectedData,
    "Read Address: " & to_hxstring(Address) &
    "  Operation# " & to_string (TransRec.Rdy) &
    "  Data: ",
    TransRec.StatusMsgOn or IsLogEnabled(ModelID, INFO)
  ) ;
else
```

Self-Check Data

- AffirmIfEqual checks the data values

```
%% Log   PASSED In DpRamController_1, Read Address: A0004  Operation#
5  Data: Received: 0004  at 2150 ns
```

```
%% Alert ERROR  In DpRamController_1, Read Address: A0004  Operation#
5  Data: Received: 0008 /= Expected: 0004 at 2150 ns
```

Copyright © 2022 - 2023 SynthWorks Design Inc.

Write Operation

SynthWorks

```
case TransRec.Operation is
  when WRITE_OP =>
    Address <= SafeResize(TransRec.Address,
                 Address'length) after tpd_Clk_Address ;
    oData   <= SafeResize(TransRec.DataToModel,
                 oData'length)  after tpd_Clk_oData ;
    Write   <= '1' after tpd_Clk_Write ;
    WaitForClock(Clk) ;

    Log( ModelID,
      "Write Operation, Address: " & to_hxstring(Address) &
      "  Data: " & to_hxstring(oData) &
      "  Operation# " & to_string (TransRec.Rdy),
      INFO, TransRec.StatusMsgOn
    ) ;

    Address <= not Address after tpd_Clk_Address ;
    oData   <= not oData    after tpd_Clk_oData ;
    Write   <= '0' after tpd_Clk_Write ;
```

Write Cycle

Log

Idle Cycle

As simple as a procedure call – yet more capable.

Copyright © 2022 - 2023 SynthWorks Design Inc.

35

35

Directive Transactions

SynthWorks

- Directive transactions interact with the VC and not the DUT

```
case TransRec.Operation is
  when WAIT_FOR_CLOCK =>
    WaitForClock(Clk, TransRec.IntToModel) ;

  when GET_ALERTLOG_ID =>
    TransRec.IntFromModel <= integer(ModelID) ;

  when GET_TRANSACTION_COUNT =>
    TransRec.IntFromModel <= TransRec.Rdy ;
```

Copyright © 2022 - 2023 SynthWorks Design Inc.

36

36

SynthWorks

Detect Multiple Drivers

- MULTIPLE_DRIVER_DETECT is generated by the resolution function

```
case TransRec.Operation is
  when . . . =>      -- Other transactions

  when MULTIPLE_DRIVER_DETECT =>
    Alert(ModelID,
      "Multiple Drivers on Transaction Record." &
      "  Transaction # " & to_string(TransRec.Rdy),
      FAILURE) ;
```

Alerts Handle Errors

- Alerts Signal (print) and Count Errors
  - Control is either global or for just this ModelID
  - Alerts have levels FAILURE, ERROR, and WARNING
  - Alerts are enabled by default and rarely disabled

```
%% Alert FAILURE In DpRamController_1, Multiple Driver on Transaction
Record.  Transaction # 15  at 20100 ns
```

Copyright © 2022 - 2023 SynthWorks Design Inc.

37

37

SynthWorks

Detect Unused Transactions

- Generate an error message for any transaction not supported by the VC

```
case TransRec.Operation is
  when . . . =>      -- Other transactions

  when others =>
    Alert(ModelID,
      "Unimplemented Transaction: " & to_string(Operation) &
      "  Transaction # " & to_string(TransRec.Rdy),
      FAILURE) ;
end case ;
```

Copyright © 2022 - 2023 SynthWorks Design Inc.

38

38

SynthWorks

Handling Settings

- Settings are handled with SetModelOptions

```
SetModelOptions(TransRec, ModelOptions, iOptVal) ;
GetModelOptions(TransRec, ModelOptions, oOptVal) ;
```

  - ModelOptions is an integer value
  - OptVal can be integer, boolean, or std_logic_vector
- Options are commonly directives but can also be transactions
- If the setting is static during a test, use a GENERIC instead
  - USE_OUTPUT_READ_REGISTER in DpRam
- Look up Operation value in MIT User Guide

9.6 **Set and Get Model Options**

AddressBusOperationType Value	Manager Transaction Name	Subordinate Transaction Name
SET_MODEL_OPTIONS	SetModelOptions	SetModelOptions
GET_MODEL_OPTIONS	GetModelOptions	GetModelOptions

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Handling Settings

- Define constants for ModelOptions in a package:

```
constant RECEIVE_READY_BEFORE_VALID : integer := 1 ;
constant RECEIVE_READY_DELAY_CYCLES : integer := 2 ;
```

- Define a signal in the architecture to hold the setting

```
signal ReceiveReadyBeforeValid : boolean := TRUE ;
```

- In Transaction Dispatcher store the setting when it selected:

```
case TransRec.Operation is
  . . .
  when SET_MODEL_OPTIONS =>
    case TransRec.Options is
      when RECEIVE_READY_BEFORE_VALID =>
        ReceiveReadyBeforeValid <= TransRec.BoolToModel ;
```

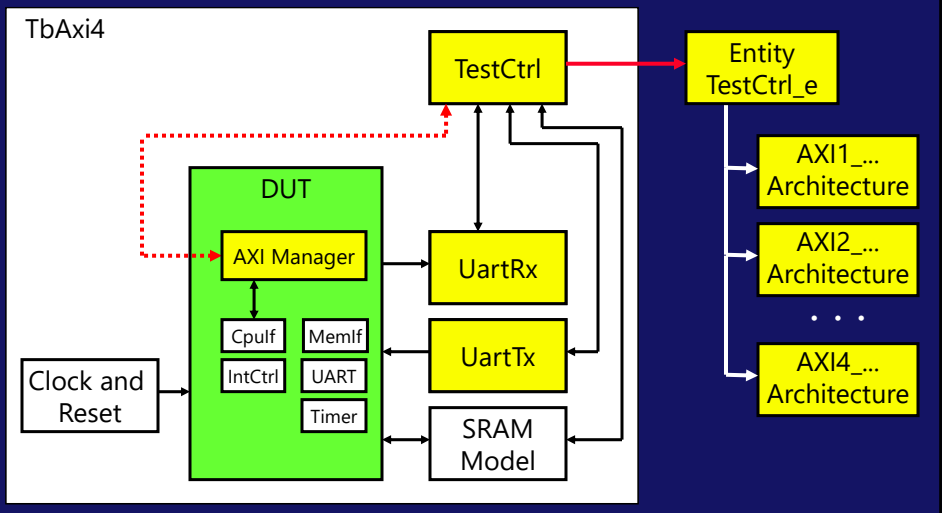
- OSVVM AxiStream VCs are similar to this except they use enumerations

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Virtual Transaction Interfaces

- Virtual Transaction Interfaces are for VC internal to the DUT.



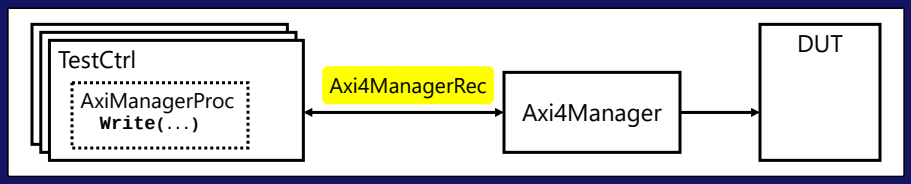
Copyright © 2022 - 2023 SynthWorks Design Inc.

41

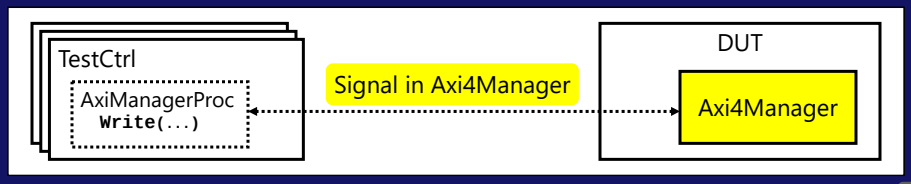
SynthWorks

Virtual Transaction Interface

- OSVVM's Connected Transaction Interface (CTI)
  - Interface = Record Port.
  - Connect to it just like RTL



- OSVVM's Virtual Transaction Interface (VTI)
  - Interface = Record Signal inside the VC
  - Connect using an external name



Copyright © 2022 - 2023 SynthWorks Design Inc.

42

Virtual Transaction Interfaces

SynthWorks

```
entity DpRamControllerVti is
generic ( . . . ) ;
port (
  Clk          : In    std_logic ;
  nReset       : In    std_logic ;

  -- DpRam Functional Interface
  Address      : Out   std_logic_vector ;
  Write        : Out   std_logic ;
  oData        : Out   std_logic_vector ;
  iData        : In    std_logic_vector
) ;

  -- Address Bus Transaction Interface
  signal TransRec : AddressBusRecType (
           Address      (Address'length-1 downto 0),
           DataToModel  (iData'length-1 downto 0),
           DataFromModel(oData'length-1 downto 0)
         ) ;

  constant RESOLVED_MODEL_ID_NAME : string := . . . ;
end entity DpRamControllerVti ;
```

DUT Interface

Transaction Interface is a signal

Architectures are identical – only entity is different

Copyright © 2022 - 2023 SynthWorks Design Inc.

VTI Test Harness

SynthWorks

```
library osvvm, osvvm_Axi4 ;
  context osvvm.OsvvmContext ;
. . .
entity  TbDpRam  is
end entity TbDpRam ;
architecture TestHarness of TbDpRam is
  . . .
  signal Controller1Rec, Controller2Rec : AddressBusRecType (. . .) ;
begin

  osvvm.TbUtilPkg.CreateClock(Clk, tperiod_Clk) ;
  osvvm.TbUtilPkg.CreateReset(nReset, . . .) ;

  DpRam_1: DpRam ( . . . ) ;

  DpRamController_1 : DpRamController (Controller1Rec, . . . ) ;
  DpRamController_2 : DpRamController (Controller2Rec, . . . ) ;

  TestCtrl_1    : TestCtrl (
    <<signal .TbDpRam.DpRamController_1.TransRec : AddressBusRecType>>,
    <<signal .TbDpRam.DpRamController_2.TransRec : AddressBusRecType>>,
    nReset) ;
end Test1 ;
```

External names connected to test sequencer ports

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Alternately in TestCtrl

- Alternately add external names in TestCtrl

```
entity TestCtrl is
port (
  nReset       : In    std_logic
) ;
  alias Controller1Rec is
    <<signal ^.DpRamController_1.TransRec : AddressBusRecType>> ;

  alias Controller2Rec is
    <<signal ^.DpRamController_2.TransRec : AddressBusRecType>> ;
end TestCtrl ;
```

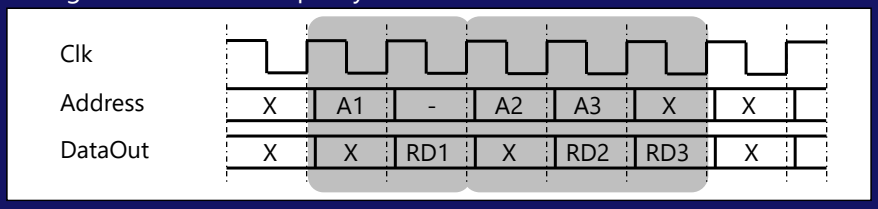
- Recommendation: Make the VTI external name match the CTI port name
- To simplify testing for VC that support CTI and VTI versions
  - Create separate test harnesses (1 CTI, 1 VTI) that have the same name
  - Compile these into separate VHDL libraries
  - The same test cases can be used to test both versions

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Implementing a Non-blocking VC

- A non-blocking API call queues the transaction and returns
- This increases VC complexity, so why do it?
- A single read takes multiple cycles for data to return



- To do consecutive read cycles requires non-blocking dispatch

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Implementing a Non-blocking VC

- Non-blocking Transactions for DpRam_Controller

```
WriteAsync       (TransRec, iAddr, iData) ;

ReadAddressAsync (TransRec, iAddr) ;
TryReadData      (TransRec, oData, oAvailable) ;
TryReadCheckData (TransRec, iData, oAvailable) ;
ReadData         (TransRec, oData) ; -- * blocking
ReadCheckData    (TransRec, iData) ; -- * blocking

WriteAndReadAsync(TransRec, iAddr, iData) ;
```

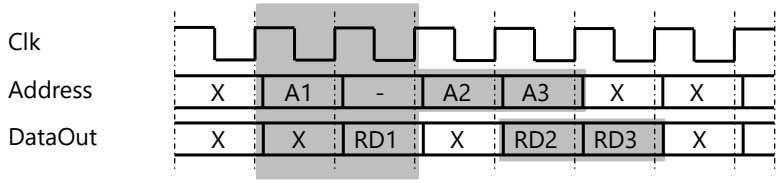
  - WriteAndReadAsync does not return read data
  - ReadData and ReadCheckData support ReadAddressAsync

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Implementing a Non-blocking VC

- Desired Transaction Sequence



- Transaction API calls to implement the sequence

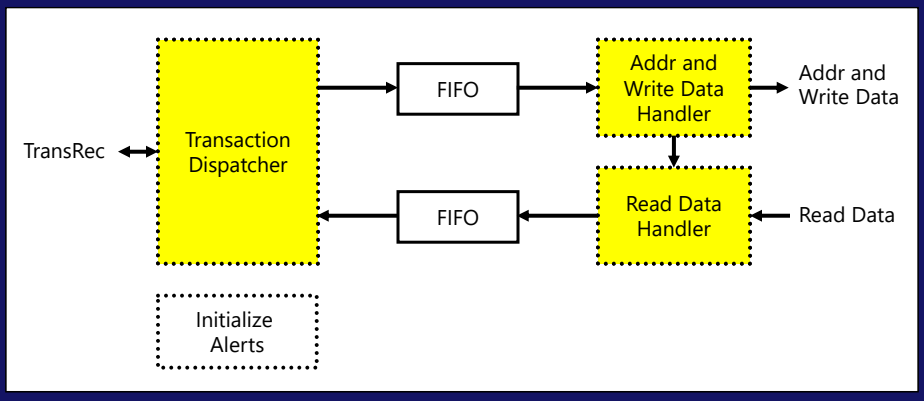
```
ReadCheck        (DpCtrlRec, A1, RD1) ;

ReadAddressAsync (DpCtrlRec, A2) ;
ReadAddressAsync (DpCtrlRec, A3) ;

ReadCheckData    (TransRec, RD2) ;
ReadCheckData    (TransRec, RD3) ;
```

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

DpRamController: Non-Blocking Transactions

- Each independent aspect of the VC is supported by a separate process
  - Transaction Dispatcher receives transactions
  - Addr and Write Data Handler interacts with the DUT
  - Read Data Handler interacts with the DUT

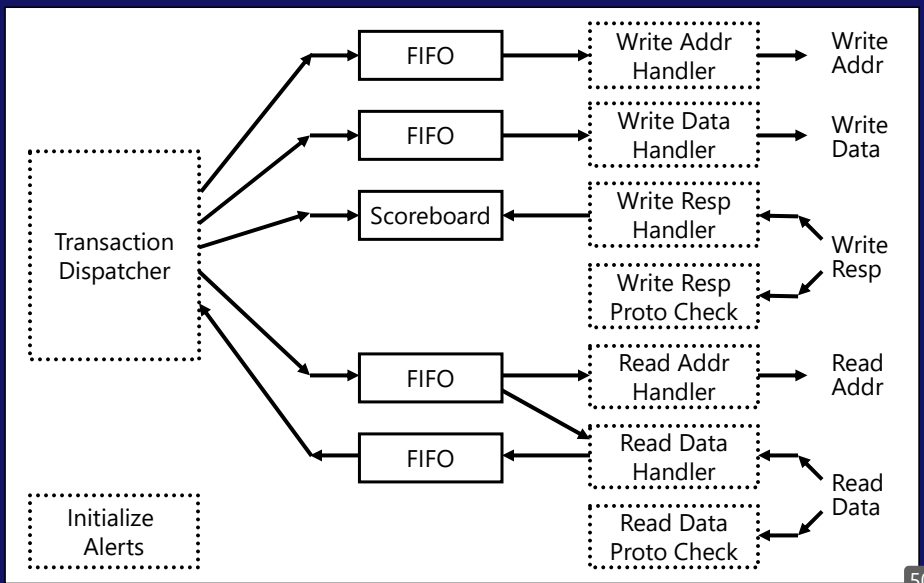


Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Axi4Manager: Non-Blocking Transactions

- Architecture of OSVVM's AXI4 Manager Full Verification Component



Copyright © 2022 - 2023 SynthWorks Design Inc.

Getting OSVVM & Running Scripts

SynthWorks

- Get the sources:

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

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

- Initialize the simulator – see Documentation/Scripts_user_guide.pdf

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

- Build all OSVVM and Run All VC Tests

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

  - Each VC has a RunAllTests and RunDemoTests

Copyright © 2022 - 2023 SynthWorks Design Inc.

OSVVM Resources

SynthWorks

- Documentation
  - HTML: https://osvvm.github.io/Overview/Osvvm1About.html
  - PDF: OsvvmLibraries/Documentation - in OSVVM release
- Recorded Webinars
  - OSVVM: Leading Edge Verification for the VHDL Community
    - https://www.youtube.com/watch?v=KVmGDy_PHNI
  - 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

Copyright © 2022 - 2023 SynthWorks Design Inc.

SynthWorks

Summary

- An ordinary "Lite" approach tends to be simple, but also less capable
  - Such as use procedures to drive stimulus rather than a VC
- OSVVM's "Faster than Lite" approach
  - Write a simple blocking VC
  - Write functional design tests and basic interface tests
  - Later, upgrade to non-blocking VC if needed for testing
- OSVVM Simplifies Verification Component Development
  - Use the Transaction Interface and API from the MIT library
  - Just write the Verification Component Behavior
  - Use either behavioral or RTL like coding
  - As simple as writing a procedure – except it is more capable
  - VC (and tests) can be written by any VHDL Engineer

Copyright © 2022 - 2023 SynthWorks Design Inc.

53

53

All you need is ... OSVVM

Verification Framework Transaction Interface & API Verification Components Test Sequencer (Test Cases)	**Verification Utility Library** Constrained Random, Scoreboards, Functional Coverage, Memory Models, Error tracking, and Message filtering, ...
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

54

54