Interrupt Handler User Guide

Release 2022.11

By

Jim Lewis

SynthWorks VHDL Training

Jim@SynthWorks.com

http://www.SynthWorks.com

Table of Contents

1 Overview 3
2 Testbench Architecture 3
3 The Test Sequencer (TestCtrl) 4
4 The Interrupt Handler Verification Component (VC) 4
5 Component Declaration is Provided in InterruptHandlerComponentPkg 5
6 Interrupt Handler VC Operation 5
7 Restrictions When Using the Interrupt Handler Verification Component (VC) 6
8 Running the Examples 6
9 Summary 6
10 About the OSVVM 7
11 About the Author - Jim Lewis 7
12 References 7

1 Overview

When software is running and an interrupt is received, information about the current running code is pushed onto the stack, then the interrupt handler is called. When the interrupt handler finishes, the information is popped off the stack and code execution resumes where it left off before the interrupt was received.

VHDL does not have an event handler that will help us do this, so OSVVM created an Interrupt Handler verification component that implements this capability.

To follow along with the demo example in this document, you may find it helpful to read the following first.

- OSVVM's Structured Testbench Framework User Guide
- OSVVM's Test Writers User Guide
- AXI4 Verification Components (VCs) User Guide

These can be found in OsvvmLibraries/Documentation or in the Documentation repository on GitHub.

2 Testbench Architecture

When using the interrupt handler, the testbench architecture (also called test harness) includes the test sequencer (TestCtrl), verification components (Axi4Manager and Axi4Memory), the InterruptHandler verification component, and DUT as shown in Figure 1.

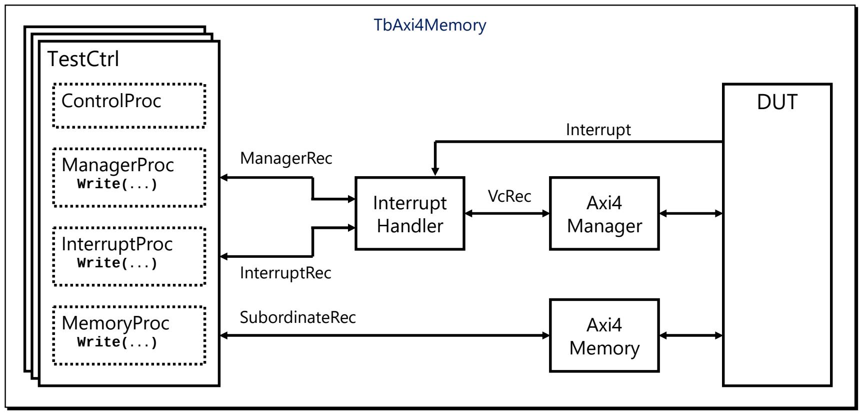

Figure 1. OSVVM Testbench Framework

The demo example can be found in OsvvmLibraries/AXI4/Axi4/testbench_interrupt. Note that in the demo test harness (TbAxi4Memory.vhd), the DUT was removed to simplify the example and instead the Axi4Manager is connected directly to the Axi4Memory. This also means that the Interrupt source comes from TestCtrl rather than the DUT (as it would in a real system).

3 The Test Sequencer (TestCtrl)

The test sequencer (TestCtrl) in general includes a control process and a separate process for each verification component (ManagerProc for Axi4Manager and SubordinateProc for Axi4Memory). When working with an interrupt handler an interrupt process (InterruptProc) is added. This also adds the record InterruptRec which contains the transaction source for the Interrupt Handler.

InterruptProc does the same function that a software interrupt handler would do. It reads the interrupt status and other registers as needed to service and clear the interrupt source. When it has finished servicing the interrupt, it calls InterruptReturn transaction. InterruptReturn must be the last transaction call in the process. For every interrupt event, the InterruptProc is run once in its entirety.

The test cases, such as TbAxi4_Interrupt1.vhd, have a different objective than a normal interrupt handler. Its objective is to verify the InterruptHandler by reading and writing the Axi4Memory while the ManagerProc is doing the same and show that all reads and writes complete properly.

See the next section, Interrupt Handler for more details on how control is switched from ManagerProc to InterruptProc and back.

4 The Interrupt Handler Verification Component (VC)

The Interrupt handler verification component (InterruptHandler.vhd) is a general-purpose solution that works for any VC that uses OSVVM's Address Bus Model Independent Transaction Interface. It can be found in the directory OsvvmLibraries/Common/src. It is compiled into the OSVVM_Common library.

The interface for the Interrupt Handler VC is shown in Figure 2.

```
entity InterruptHandler is
generic (
  MODEL_ID_NAME    : string := ""
) ;
port (
  -- Interrupt Input
  IntReq          : in   std_logic ;

  -- From TestCtrl
  TransRec        : inout AddressBusRecType ;
  InterruptRec    : inout AddressBusRecType ;

  -- To Verification Component
  VcRec           : inout AddressBusRecType
) ;
```

Figure 2. Interrupt Handler VC Interface

IntReq port is the interrupt input and is generally connected to the DUT. Here IntReq port is connected to TestCtrl only because there is no DUT. TransRec port connects to ManagerRec (driven by ManagerProc) . InterruptRec connects to InterruptRec (driven by InterruptProc). VcRec connects to the Axi4Manager TransRec port.

5 Component Declaration is Provided in InterruptHandlerComponentPkg

As with all OSVVM VC, you can use a simple component declaration when instaning the InterruptHandler. No component declaration is needed in the test harness as a component declaration is provided in InterruptHandlerComponentPkg.vhd. InterruptHandlerComponentPkg is compiled into the osvvm_common library by OSVVM scripts and referenced by the OsvvmCommonContext.

OsvvmCommonContext is included in Axi4Context. As a result, you only need the reference shown in Figure 3 to include both InterruptHandlerComponentPkg, AddressBusRecType (Address Bus Interface), and the component declarations for AX4 VC.

```
library OSVVM_AXI4 ;
  context OSVVM_AXI4.Axi4Context ;
```

Figure 3. Context Reference to use InterruptHandler and Axi4 VC

6 Interrupt Handler VC Operation

At the start of the simulation, both ManagerProc and InterruptProc start running and eventually call a transaction.

When no interrupt is pending, the Interrupt Handler VC passes transactions from TransRec (ManagerRec) to the Axi4Manager (via VcRec). Results received via VcRec are sent back to TransRec (ManagerRec).

If an interrupt is pending and the InterruptRec has a transaction, the Interrupt Handler VC allows the current TransRec (ManagerRec) transaction to complete and then switches to Interrupt Handling Mode. When in Interrupt Handling Mode, it passes transactions from InterruptRec to the Axi4Manager (via VcRec) and results received via VcRec are sent back to InterruptRec.

The Interrupt Handler VC remains in Interrupt Handling Mode until an InterruptReturn transaction is received on InterruptRec. The InterruptReturn must be last transaction in the InterruptProc process. Following that, the InterruptProc loops back to the top of the process and calls the first transaction, however, it stops because the InterruptHandler VC is no longer in Interrupt Handling mode and is now handling transactions from TransRec (ManagerRec).

In the InterruptHandler VC, interrupt pending is defined to be IntReq is asserted (currently a 1) and there is a transaction pending on InterruptRec (ie: a transaction was called in InterruptProc). Hence, if there is not InterruptProc to calls transactions using InterruptRec, the InterruptHandler will not go into Interrupt Handling Mode.

7 Restrictions When Using the Interrupt Handler Verification Component (VC)

When using the InterruptHandler VC, do not use split transactions in the ManagerProc. If you were to call WriteAddressAsync and then WriteDataAsync, an interrupt could interrupt after WriteAddressAsync and before WriteDataAsync – which would leave the Axi4Manager write interface in a bad state. Same applies for ReadAddressAsync and ReadData (or ReadCheckData, TryReadData, TryReadCheckData).

Note that the split transactions are intended to be used to fully test the operation of split transaction interfaces, such as AXI4. Hence, split transactions are a low level test capability that is not needed when testing higher level functionality such as interrupts.

8 Running the Examples

See the steps in the OSVVM Overview guide for running the OSVVM demos. Do the steps shown in Figure 4 in your simulator. StartUp.tcl needs to be sourced each time you start the simulator. See the Script User Guide for additional details for Aldec's ActiveHDL, GHDL, Synopsys VCS, and Cadence Xcelium.

```
cd sim
source ../OsvvmLibraries/Scripts/StartUp.tcl
build  ../OsvvmLibraries/OsvvmLibraries.pro
build  ../OsvvmLibraries/AXI4/Axi4/testbench_interrupt
```

Figure 4. Compiling and Running DPRAM tests

9 Summary

At the end of the day, the InterruptHandler is just a specialized multiplexer/demultiplexer (responses) that chooses between connecting TransRec to VcRec or InterruptRec to VcRec depending on the state of the Interrupt Handling Mode.

The InterruptHandler works with any OSVVM verification component that uses the Address Bus Model Independent Transaction interfaces.

It is simple to deploy since all you must do is

- Add an additional AddressBusRecType port to TestCtrl Entity
- Instance InterruptHandler to your test harness – don't forget we provide the component declaration in a package to further simplify this.
- Add an InterruptProc to TestCtrl with the interrupt handler

10 About the OSVVM

The OSVVM utility and verification component libraries were developed and are maintained by Jim Lewis of SynthWorks VHDL Training. These libraries evolved from methodology and packages developed for SynthWorks' VHDL Testbenches and verification class.

Please support OSVVM by purchasing your VHDL training from SynthWorks.

11 About the Author - Jim Lewis

Jim Lewis, the founder of SynthWorks, has thirty plus years of design, teaching, and problem solving experience. In addition to working as a Principal Trainer for SynthWorks, Mr Lewis has done ASIC and FPGA design, custom model development, and consulting.

Mr. Lewis is chair of the IEEE 1076 VHDL Working Group (VASG) and is the primary developer of the Open Source VHDL Verification Methodology (OSVVM.org) packages. Neither of these activities generate revenue. Please support our volunteer efforts by buying your VHDL training from SynthWorks.

If you find bugs these packages or would like to request enhancements, you can reach me at jim@synthworks.com.

12 References

[1] Jim Lewis, Advanced VHDL Testbenches and Verification, student manual for SynthWorks' class.