Verification of an AXI DMA Master Using UVM

Electrical Engineering Department

Chung, I-Da(Yida) (MS ElectricalEngineering)

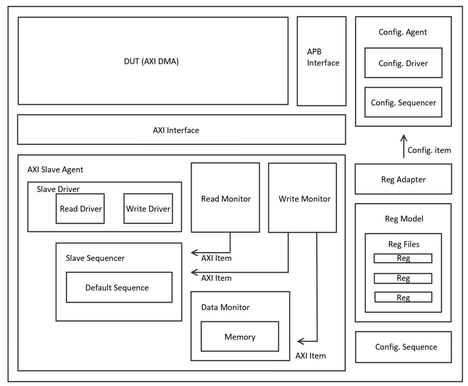
Project Advisor: Lili He

Introduction

The project presents an Advanced eXtensible Interface (AXI) functional verification environment for a configurable AXI Direct Memory Access (DMA) master using Universal Verification Methodology (UVM). AXI is one of the most common micro controller bus first introduced from ARM AMBA in 1996. The AXI interface is widely used in many SoC products and intellectual property (IP) designs as an important industry standard. The design under test of the project, AXI DMA master, is an 32/64-bit AXI master used for memory copy and peripheral control. The verification environment contains two agents – an configuration agent, and an AXI transaction agent. The configuration agent sends out the APB data to set up the command and static registers, while the AXI agent provides randomization responses to the AXI DMA master. The AXI slave agent is a reusable verification block which can be easily modified and configured targeting various user-defined AXI master transactions. Simulation results and more details are provided in the following sections.

Methodology

High-level Overview of the UVM Agents



Both the configuration agent and the AXI slave agent communicate with the DUT through interfaces. The configuration agent connects to the APB interface, which includes the APB ports, interrupt bit and other peripheral control signals. The AXI slave agent connects to the AXI interface with port read(*R*), read address(*AR*), write(*AW*), write address(*W*), and response(*B*)

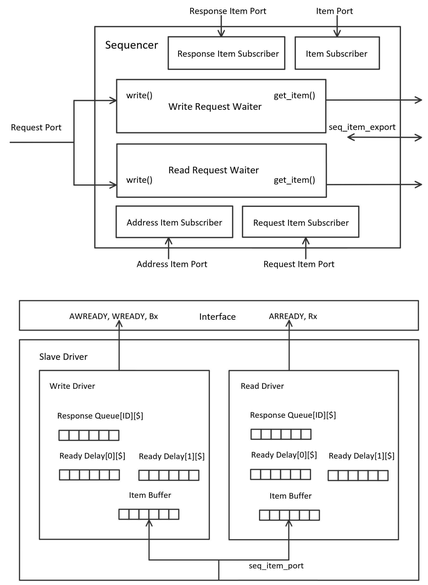
Configuration Agent

The configuration agent works with UVM Register Abstraction Layer as a middle stage of register configuration between the user and the design itself. The RAL model implements an adapter, a register files model and configuration sequences designed by the user.

Methodology

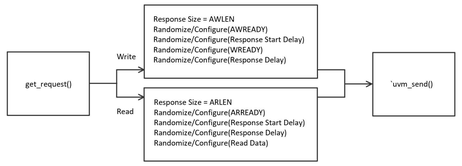
AXI Slave Agent

The AXI agent contains a slave driver, a slave sequencer, a read monitor, a write monitor and a data monitor. The slave driver has two sub-drivers, a read driver and a write driver. The read driver drives read (*Rx*) and read address (*ARx*) signals as an AXI slave. The write driver drives the write (*Wx*), write address (*AWx*), and response (*Bx*) signals. The read monitor and write monitor collect all the AXI port's signals and pack them up as AXI items. The data monitor is used as a memory implemented with UVM memory library to collect read and write data.



AXI Item, AXI Configuration, General Sequence

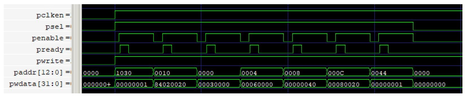
An AXI item class defines the *uvm_object* transferred in the UVM test. The item contains the data properties, delay amount and the event time. The AXI configuration sets up the basic structure, which is inherited by other AXI UVM sequences, including the interface of the AXI transaction and several property constraints and delay. The sequence class implements the general responses of AXI request received from the sequencer. The read/write monitor records the address request and sends the data to the sequencer. The sequence class gets the address item, and then randomizes the data. After the response is set up and ready to be transmitted, the sequence class starts the sequence transaction.

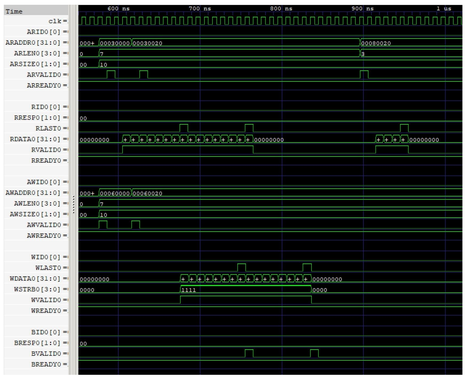


Analysis and Results

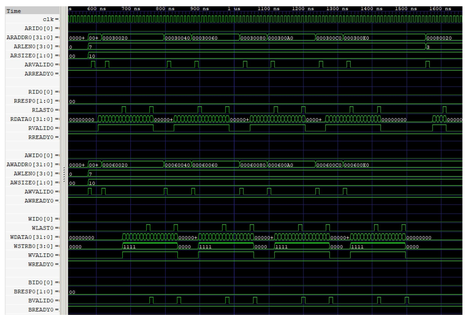
Standard Transaction

	Value	Reg. Offset
READ Buffer max	32-bit	#0010
READ Start Address	0x00030000	#0000
WRITE Start Address	0x00060000	#0004
Buffer Size	64-bit	#0008
Next Command Address	0x00080020	#000C





	Value	Reg. Offset
Buffer Size	256	#0008

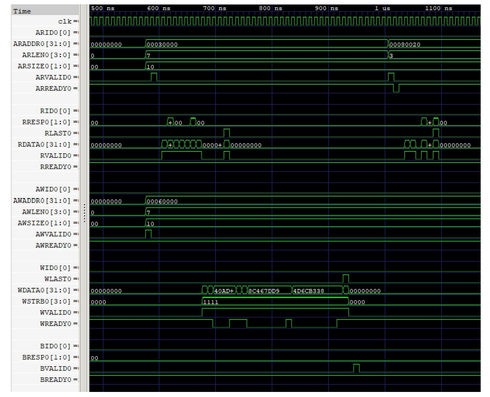


Delay Transaction

Delay: AWREADY, WREADY, ARREADY, Rx, Bx

Min Delay	Max Delay	Weight (0)	Weight (Shrt)	Weight (Long)
0	10	6	3	1

	Value	Reg. Offset
READ Buffer max	32-bit	#0010
READ Start Address	0x00030000	#0000
WRITE Start Address	0x00060000	#0004
Buffer Size	32-bit	#0008
Next Command Address	0x00080020	#000C



Summary/Conclusions

The project introduces a thorough verification environment for both the zero-delay and delayed AXI transactions testing the AXI DMA master. The UVM test could be configured targeting different size of buffers depending on the needs of the design. Further, the AXI agent provides the developers a way to validate the AXI transaction, including the read and write burst of the AXI channels, and the handshake mechanism between a master and a slave. The agents can be further extended or reused by other IP blocks implementing the AXI master transaction.

Key References

[1] "Verification Methodology Cookbooks | Coverage, UVM and OVM.", https://verificationacademy.com/cookbook. Accessed: 1 Sep. 2020.

[2] ARM. AMBA AXI and ACE Protocol Specification. ver. H, March 2020.

[3] Gayathri M, R. Sebastian, S. R. Mary and A. Thomas, "A SV-UVM framework for Verification of SGMII IP core with reusable AXI to WB Bridge UVC," 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, 2016, pp. 1-4, doi: 10.1109/ICACCS.2016.7586315.

[4] Ishitani, Taichi (2020) tvip-axi (Version 5e33a13) [Source code]. https://github.com/taichi-ishitani/tvip-axi.

[5] Provartec. PR200 Configurable Dual-Core High Performance AXI DMA Reference Guide. Ver. 1.6, Dec. 2010

[6] S.Guruprasad, K.Sudharshan "Design and Analysis of Master module for AMBA AXI-4" in (NCECS-2011), Siliguri Institute of Technology, Sept. 2011

Acknowledgements

I would like to thank Professors Lili He, David Parent, Morris Jones and San Jose State University for my academic support during the hard time of COVID-19 pandemic. The code I reference is licensed under the Apache-2.0 license.