VERIFICATION OF AMBA AXI BUS PROTOCOL IMPLEMENTING INCR AND WRAP BURST USING SYSTEM VERILOG

Harsha Garua¹, Keshav Sharma², Chusen Duari³
¹Manipal University Jaipur
²Manipal University Jaipur
³Assistant Professor, Manipal University Jaipur

Abstract
This paper describes the development of verification environment for AMBA AXI (Advanced Extensible Interface) protocol using System Verilog. AXI supports high performance, high-frequency system designs. It is an On-Chip communication protocol. It is suitable for high-bandwidth and high frequency designs with minimal delays. It provides flexibility in the implementation of interconnect architectures and avoid use of complex bridges. It is backward-compatible with existing AHB and APB interfaces. The key features of the AXI protocol are that it consists of separate address, control and data phases. It support unaligned data transfers using byte strobes. It requires only start address to be issued in a burst-based transaction. It has separate read and write data channels that provide low-cost Direct Memory Access (DMA). This paper is aimed at the verification of various burst type transaction (INCR and WRAP) of the AXI bus protocol and the Verification Environment is built using System Verilog coding[1].

Keywords: AMBA AXI, INCR, Wrap Burst, System Verilog

I. INTRODUCTION
AMBA stands for Advanced Microcontroller Bus Architecture (AMBA). It is an open-standard, on-chip interconnect specification for the connection and management of intellectual property (IP) blocks in system-on-a-chip (SoC) designs. It supports development of designs with large numbers of controllers and peripherals/IP blocks. Since its inception, the scope of AMBA has, despite its name, gone far beyond micro controller devices.

The AXI protocol support burst-based transaction which signifies transfer of multiple data grouped in packets. It consists of independent transaction channels: Read Address Channel, Read Data Channel, Write Address Channel, Write Data Channel and the Write Response Channel. An address channel carries control information that describes the size of bursts, number of bursts and length of the data to be transferred. The data is transferred between master and slave using either:

- Write Data Channel: in write transaction data is transferred from the master to the slave using this channel.
- Read Data Channel: in read transaction data transferred from the slave to the master using this channel.

FIXED burst:
In a fixed burst-type, single address is used for storing data that is frequently used by the slave similar to a FIFO for every transfer in the burst. This type of burst enables repeated accesses to the same location.

INCR burst:
In an incrementing burst, the address is incremented for each transfer in the burst from the previous address. The size of burst determines the increment value for the address.
Case I:
Address ADDR AxBURST AxSIZE Data
0x9 0x9 INCR 8-bit 1 data transfer
0xA 0xA INCR 8-bit 1 data transfer
0xB 0xB INCR 8-bit 1 data transfer
0xC 0xC INCR 8-bit 1 data transfer

Case II:
Address ADDR AxBURST AxSIZE Data
0x0 0x00 INCR4 4-byte 1 data transfer
0x4 0x04 INCR4 4-byte 1 data transfer
0x8 0x08 INCR4 4-byte 1 data transfer
0xC 0xC INCR4 4-byte 1 data transfer

Wrapping burst:
Similar to INCR a wrapping burst is also consists of incrementing the address for each transfer of the previous transfer address. In a wrapping burst when the boundary is reached the address wraps around to a lower address. The wrap boundary is determined by the size of each transfer in the burst multiplied by the total number of transfers in the burst. Two restrictions apply to wrapping bursts:
- WRAP can work only for the start address aligned to the size of the transfer.
- And the burst length must be (2, 4, 8, 16).

Case I: If we do 4 beat burst on 32 bit AXI with AWLEN=16 and starting address 0x00000004 address

3. DUT
The verification environment is organized in a hierarchical layered structure which helps to verification. The following subsections explain the functionality of four primary components in the verification environment.

4. Virtual Interface
Virtual interfaces provide a mechanism for separating abstract models and test programs from the actual signals that make up the design. It allows the same subprogram to operate on different portions of design, and to dynamically control the set of signals associated with the subprogram. Instead of referring to the actual set of signals directly, users are able to manipulate a set of virtual signals.

5. Scoreboard
The scoreboard collects address and control information on Write Address Channel and the data on Write Data Channel at the output of AXI slave and compares it with the address, data and control information at the output of DUV. It collects address and control information on Read Address Channel at the AXI slave input side and compares it with the address and control information at the output of the DUV. It collects the read response and data at the DUV side and compares it with that on the input side[4].

II. AXI PROTOCOL SPECIFICATION
A typical system-on-chip design consists of a number of master and slave devices connected together using the interconnecting protocol [2]. The AXI protocol provides a single interface definition, for the interfaces:
- Between a master and the interconnect
- Between a slave and the interconnect
- Between a master and a slave.

The key features of the AXI protocol are:
- Consists of separate address phase and data phase.
- support non-sequential data transfers using byte strobes
- data is transferred in form of packets i.e. bursts with only start address issued
- separate channels for read and write transactions to enable low-cost Direct Memory Access (DMA)

AXI stands for Advanced Extensible Interface. It is a part of the Advanced Microcontroller Bus Architecture (AMBA) developed by ARM (Advanced RISC Machines) company. It is an On-Chip communication protocol. The AMBA AXI protocol supports high-performance, high-frequency system designs. The key features of the AXI protocol are that it has separate address phase and data phases. It uses byte strobes to support unaligned data transfers. It utilizes burst-based transactions with only the start address issued. It has separate read and write data channels that provide support for Direct Memory Access (DMA) at a lower cost. It support for out-of-order transaction completion. It permits easy addition of register stages to provide timing closure.

Objective of paper:
- To study AMBA AXI Specification
- To implement INCR and WRAP burst type transaction in AXI design, design coding in system verilog
- To prepare testcases for various burst size and burst length.
- To achieve code and functional coverage of the verification environment developed
issue multiple outstanding addresses
support out-of-order transactions
supports addition of new register stages to provide timing closure.
allows address information to be issued before the actual data transfer.

III. Burst Operation in AXI

The AXI protocol support is burst-based transaction. Every transaction consists of address and control information issued on the address channel which describes various parameters of the data to be transferred. A write data channel is used to transfer data between master and slave. The AXI protocol consists of an additional write response channel in which slave sends response to the master as an acknowledgement for the write transaction [5]. The AXI protocol enables:

- address information to be issued ahead of the actual data transfer.
- support for multiple outstanding transactions.
- support for out-of-order completion of transactions.

The AXI protocol defines three transaction bursts types described in:

- Fixed burst
- Incrementing burst
- Wrapping burst

<table>
<thead>
<tr>
<th>Data</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>First: 0x00000004</td>
<td>0x00000004</td>
</tr>
<tr>
<td>Second: 0x00000008</td>
<td>0x00000008</td>
</tr>
<tr>
<td>Third: 0x0000000C</td>
<td>0x0000000C</td>
</tr>
<tr>
<td>Fourth: 0x00000010</td>
<td>0x00000000 (Wrapped)</td>
</tr>
</tbody>
</table>

The WRAP operation address is wrapped to transfer size.

Case II: If we do 4 beat burst on 16 bit AXI with AWLEN=4 and starting address 0x00000004 address

<table>
<thead>
<tr>
<th>Data</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>First: 0x00000002</td>
<td>0x00000004</td>
</tr>
<tr>
<td>Second: 0x00000004</td>
<td>0x00000006</td>
</tr>
<tr>
<td>Third: 0x00000006</td>
<td>0x00000008</td>
</tr>
<tr>
<td>Fourth: 0x00000008</td>
<td>0x00000002 (Wrapped)</td>
</tr>
</tbody>
</table>

The WRAP operation address is wrapped to transfer size.

IV. SYSTEM VERILOG BASED VERIFICATION

2. Test Cases
Test cases are written for different scenarios, which cover the functionality and corner cases. Basic read-write transactions and various burst types are verified. Parameterization will be randomized for different test cases and are written to check all the possible scenarios. These test cases are run in regression with multiple seeds.

I. AXI Master
The AXI master mainly consists of two classes, axi_m_driver and axi_m_env. The AXI master is implemented by the axi_m_driver class and the axi_m_env consists of the tasks from master transaction and generator class. The axi_m_driver class consists of a main task named run_m_driver() which drives packets from Interface to the DUV.

II. AXI Slave
The AXI slave consists of axi_s_driver class and axi_s_env class. The AXI slave is implemented by the axi_s_driver class and the axi_s_env runs the tasks of slave transaction and generator class. The axi_s_driver class has a run_s_driver() task which drives the generated signals from interface to the DUV in loops using mailbox element.

III. AXI Assertion
System Verilog assertion (SVA) has been used here to check the various properties of AXI interconnection for an expected behavior. In this paper, assertions are applied to detect a successful communication between master and slave.

IV. AXI Scoreboard
The AXI scoreboard comprises of the comparison between the address and data send by an AXI master in a write/read transaction in a burst and the address and data in the slave. The scoreboard consists of the output values from the Golden Model and the DUV and sends it to the Checker block for analysis. Every transaction in AXI consists of a unique AWID or ARID that is issued according to the AXI protocol and the transactions can arrive in any order with different IDs.
V. WAVEFORM

Figure 4: AXI Basic Transaction

Figure 5: FIXED Burst-type Transaction

Figure 6: INCR Burst-type Transaction
VI. SYSTEM VERILOG VERIFICATION REPORT

The results of verification components such as Interface, Driver, Generator, Transaction, Scoreboard, Assertions and Monitor are developed using System Verilog. According to the designed verification environment, the test cases are verified for various burst types implementing AXI Protocol interconnection between a master and a slave. The Test Cases are written in the Generator block by assigning test-case number. The Driver drives the generated packets to the DUV. Various burst-type transactions have been performed in DUV. The DUV output values are compared with the output values from the Golden Model in the checker block. If the DUV output value matches the Golden Model output value then we conclude that the verification of the design is completed successfully. By using the simulator tool Questa various test cases are generated and simulated which resulted in generation of waveforms shown in Figures 4 to 8.

VIII. CONCLUSION

Verification of AXI Bus Protocol for Single Master-Single Slave implementing FIXED, INCR and WRAP has been achieved by developing the Verification IP using System Verilog. Test has been passed for various burst size and burst length has been used for:

- INCR: single address followed by data in sequence has been passed unlike FIXED where address and data was transferred in pipeline.
- WRAP: when boundary was reached data was wrapped to one lower address unlike INCR and FIXED.
An analysis of the verification plan has been made according to the working of AXI protocol for single master and single slave for various burst-type transactions. The study of a successful verification environment resulted in a high coverage verification.

IX. ACKNOWLEDGEMENT

We the authors of this paper would like to acknowledge with thanks to Mr. Chusen Duari, Assistant Professor, Department of Electronics and Communication Engineering, Manipal University Jaipur for constant encouragement and support for enabling us to submit this paper.

X. REFERENCES

Journal / Conference Papers

Web:

IX. BIOGRAPHIES

- Mr. Chusen Duari, Assistant Professor, Department of Electronics & Communication Engineering, Manipal University Jaipur.

Appendix

AMBA - Advanced Microcontroller Based Architecture.
AXI- Advanced Extensible Interface
AHB – Advanced High performance Bus.
APB – Advanced Peripheral Bus.
SOC - System On Chip.
IP - Intellectual Property
DUV- Design Under Verification
INCR- Increment burst
WRAP- Wrapping burst
FIFO- First-In-First-Out

<table>
<thead>
<tr>
<th>Burst-</th>
<th>Burst-</th>
<th>Burst-</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Length</td>
<td>Size</td>
<td>Address</td>
<td>Address</td>
</tr>
<tr>
<td>FIXED</td>
<td>4</td>
<td>1-Byte</td>
<td>0x00</td>
<td>-</td>
</tr>
<tr>
<td>INCR</td>
<td>4</td>
<td>1-Byte</td>
<td>0x02</td>
<td>0x05</td>
</tr>
<tr>
<td>WRAP</td>
<td>4</td>
<td>1-byte</td>
<td>0x02</td>
<td>0x01</td>
</tr>
</tbody>
</table>