Design of IP Core for AMBA AXI4 Bus-Based System-On-Chip Communication Protocol

Mr. Mohit N. Kandiya  
P.G. Student, EC Department  
Darshan Insti. of Engg. & Tech, Rajkot  
mohit.kandiya@gmail.com

Mr. Manish K. Harniya  
P.G. Student, EC Department  
Darshan Insti. of Engg. & Tech, Rajkot  
manishharniya3@gmail.com

Prof. Kishan K. Govani  
Guide, M.Tech(Embedded & VLSI System)  
Darshan Insti. of Engg. & Tech, Rajkot  
govanikishan@gmail.com

Abstract: Typically, an embedded design have a general purpose processor, cache, DMA port and other Bridge port to I/O bus. Advanced Microcontroller Bus Architecture (AMBA) is a I/O bus for interfacing embedded system. Increasing amount of logic that can be placed onto a single silicon die is the development of highly integrated SoC designs. System-on-chip (SoC) designs use bus protocols for high performance data transfer among the Intellectual Property Core. These protocols have advanced features such as unaligned data transfer, pipelining, burst and split transfers. The goal is to design intellectual property core for AMBA AXI4 bus based System-On-Chip Communication Protocol. AMBA AXI4 is the bus that performs best in terms of throughput, latency and utilization for single or multiple channels. Paper starts with a brief introduction AMBA AHB protocol, AMBA AXI, and simulation result for performance analysis of AMBA AXI (Advanced Extensible Interface) bus protocol.

Keywords: SoC, AMBA, Bus Protocol

I. INTRODUCTION

System-level design and intellectual property cores modeling is the right way to fast SoC innovation with the capability of different design alternatives. Early in the design process in order to confirm the best possible architecture, HW/SW partition and performance parameters like power consumption. For quickly innovation, system-level design provides a high level of abstraction, very fast simulation speed and allows a high degree of IP reuse. Modern computer system more and more on highly complex on-chip communication protocol to exchange data. The increasing amount of logic that can be developed onto a silicon die is driving SoC designs. Such high computing power must be matched by interconnect fabrics with adequate bandwidth and efficiency. [1] The paper includes, The first section contain the description of AHB protocol Second section describes AMBA AXI bus. Third section shows the comparisons of AHB to AXI based on different parameters.

A) AMBA

The Advanced Microcontroller Bus Architecture (AMBA) was registered trademark of ARM Ltd in 1996. It is widely used for the on-chip bus in SoC designs. The 1st AMBA buses were Advanced System Bus (ASB) and Advanced Peripheral Bus (APB). The 2nd AMBA 2, ARM added AMBA Advanced High-performance Bus (AHB) that is a single clock-edge protocol. In 2003, ARM introduced the 3rd generation, AMBA 3 with the Advanced Trace Bus (ATB) as part of the Core Sight on-chip debug and trace solution. These protocols are today the de-facto standard for 32-bit embedded processors. [5] In 2010, ARM introduced the 4th generation, AMBA 4, including AMBA 4 AXI4, AXI4 Lite, and AXI4 Stream Protocol.

Traditional SoC is AMBA. The goal of AMBA is given bellow:[1]

- **Partitioning modular design**: Its methodology for embedded processor design encourages both a modular and first-time right system design which supporting module reuse.
- **Clocking and reset**: AMBA specifies a flexible, low-overhead bus interface and clocking structure to the core. This simplifies system design because interface protocols are clearly defined to have one or more bus masters and slaves
- **Supporting for low-power designs**: By partitioning high and Low bandwidth devices inside the system, AMBA ensures power efficient designs.
- **On-chip test**: AMBA integrates an minimum on-chip test access methodology that reuses the basic bus infrastructure. This helps make the external test of CPU and peripheral macrocells more efficiently.
- **Support of multiple development platforms**: This allows cycle-accurate benchmarking and hardware prototyping. It also features a set of generic reference peripherals that make it easier to port real-time kernel software.

II. AMBA 2

The Advanced Microcontroller Bus Architecture (AMBA) specification is an on-chip communications standard for designing of high-performance embedded system. [2] AMBA based microcontroller shown in Fig.1
Three different buses are defined within the AMBA specification:

- AHB:- Advanced High-performance Bus
- ASB:- Advanced System Bus
- APB:- Advanced Peripheral Bus

III. AMBA AXI4

The AMBA AXI4 protocol is designed for high-performance and high-frequency system designs, which has many of features that make AXI4 suitable for a high-speed, submicron interconnect. The AMBA AXI protocol objectives: The AMBA AXI specification was created with the same objectives. [3]

- Suitability for high-bandwidth and low-latency designs [3]
- To enable high-frequency operation without using complex bridges [3]

B) AMBA AXI4 architecture

AMBA AXI4 supports data transfers up to 256 beats. In AMBA AXI4 system 16 masters and 16 slaves are interfaced. The system consists of five channels namely write address channel, write data channel, read data channel, read address channel, and write response channel. The AXI4 protocol supports the following mechanisms; [9]

- Unaligned data transfers and up-dated write response requirements.
- Variable-length bursts, from 1 to 16 data transfers per burst.
- A burst with a transfer size of 8, 16, 32, 64, 128, 256,512 or 1024 bits wide is supported.
- Updated AWCACHE and ARCACHE signaling details.

Each transaction is burst-based which has control information on the address channel that describes the nature of the data to be transferred. The data is transferred between master and slave using a write data channel to the slave or a read data channel to the master.

C) AMBA AXI4 Interconnect

The interconnect block consists of arbiter, decoder and multiplexer. When two masters initiate a transaction simultaneously, the arbiter gives priority to access the bus by arbitration algorithm. The decoder decodes the address sent by master and the control for transaction goes to one slave out of 16. The AMBA AXI4 interface decoder is centralized digital block. The decoder decodes the address sent by master and goes to one slave out of 16. Range-1 locations are meant for slave-1, next Range 2 addressable locations are meant for slave-2, and so on till slave-16. The AXI4 master and slave modules consists of their own common read/write buffer which stores the read/write address and data as shown in Fig. 2.

D) AMBA AXI4 IP CORE Data Transaction Operation

The read address register stores the remaining read addresses to be sent; pending write address register which stores the remaining write addresses to be sent and pending write data register which stores the remaining write data to be sent. The read/write state machines receive internal inputs from the read/write buffer.

The signals that are used for AXI 4 is shown in Table 1. These signals include Signal for read and write channel with VALID and READY signal. “AW” indicates address write, “AR” indicates address read, “R” indicates read data, “W” indicates write data. For write operation write address and write data are used, so for write address AWVALID, AWADDR, AWREADY, for data WVALID, WREADY, WDATA are available. [3]

The AXI4 test bench initiates the read or write transaction and the output from the AXI4 slave are standard read/write channel signals. The AXI4 slave receives the write data in the same order as address. Signals used to design master module is shown in Fig. 4. The finite state machine for IP Core is shown in Fig. 3, which include the three state read, write and idle state.
Table 1  AMBA AXI4 Signals\(^{(3)}\)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Source: master/slave</th>
<th>Input/Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aclk</td>
<td>Global</td>
<td>Input</td>
<td>Global clock signal.</td>
</tr>
<tr>
<td>AResetn</td>
<td>Global</td>
<td>Input</td>
<td>Global reset signal.</td>
</tr>
<tr>
<td>AWADDR[31:0]</td>
<td>Master</td>
<td>Input</td>
<td>Write address.</td>
</tr>
<tr>
<td>AWBURST[1:0]</td>
<td>Master</td>
<td>Input</td>
<td>Write burst type.</td>
</tr>
<tr>
<td>WDATA[31:0]</td>
<td>Master</td>
<td>Input</td>
<td>Write data.</td>
</tr>
<tr>
<td>ARADDR[31:0]</td>
<td>Master</td>
<td>Input</td>
<td>Read address.</td>
</tr>
<tr>
<td>RDATA[31:0]</td>
<td>Master</td>
<td>Input</td>
<td>Read data.</td>
</tr>
<tr>
<td>WLAST</td>
<td>Master</td>
<td>Input</td>
<td>Write last.</td>
</tr>
<tr>
<td>RLast</td>
<td>Slave</td>
<td>Output</td>
<td>Read last.</td>
</tr>
<tr>
<td>AWVALID</td>
<td>Master</td>
<td>Output</td>
<td>Write address valid.</td>
</tr>
<tr>
<td>AWREADY</td>
<td>Slave</td>
<td>Output</td>
<td>Write address ready.</td>
</tr>
<tr>
<td>WVALID</td>
<td>Master</td>
<td>Output</td>
<td>Write valid.</td>
</tr>
<tr>
<td>RVALID</td>
<td>Slave</td>
<td>Output</td>
<td>Read valid.</td>
</tr>
<tr>
<td>BREADY</td>
<td>Master</td>
<td>Output</td>
<td>Response ready.</td>
</tr>
<tr>
<td>RVALID</td>
<td>Slave</td>
<td>Output</td>
<td>Read valid.</td>
</tr>
</tbody>
</table>

In write operation, when the process starts, then master sends control and an address information on the write address channel as shown in Fig. 4. The master sends write data on the write data channel. When the master sends the last data item, the WLAST signal goes HIGH, which indicate that the write transaction is finished. This conditional signal indicates the “ok” status of the write transaction; But WLAST and RLAST are used in Burst based transaction.

Simulation result for master write is shown in Fig. 4, which includes RESET, IDLE and WRITES state. In the reset state, when arst is low then grant, address and data are undefined and all conditions are active low. In the Ideal state arst, aenable are high control is “000”. Address is zero. Data are undefined. Xreq request to arbiter for select the master and arbiter respond by Xgrant signal. When contro is in “001” condition then WRITE state is selected. When VALID from the master and READY are high then address and data are transfer to the channel.

Fig. 3  AXI4 IP Block FSM

Fig. 4  AXI Master Write Operation

Fig. 5  AXI Master Read Operation

When Control is “010”, Aenable and Xgrant are high then control goes to the Read state. When then idle state goes to
read state. When ARVALID and ARREADY are high then address is asserted in address channel. RVALID and RREADY are high then data signals are asserted in data lines. Simulation result for AMBA AXI4 Master Read is shown in Fig. 5.

IV. CONCLUSION

AMBA AXI4 bus specification and a technology independent methodology for designing of IP Core read and Write operation is satisfied for testing customized high-integration embedded interfaces. In this work data transactions were carried out using AMBA AXI4 protocol modeled in VHDL and simulation results for read and write operation of data and address are shown in Xilinx tool.

FUTURE WORK

The AMBA AXI4 has burst Mode data transfer. The burst must not cross the 4k boundary. This can be overcome by INCR burst type.

REFERENCES