[SoC Basics] ARM AXI Interface

Florian Hirner

October 26, 2021

Presentation Outline

- Introduction
- Thread IDs
- Handshake Mechanism
- Channels
- Transactions

Introduction

Background

- A **bus** is a communication system that **transfers data** between components like:
  $\rightarrow$ CPU, RAM, Storage Devices, GPU, etc.
- **Bus protocols** are needed to ensure order during transmitting
- Otherwise it would lead to unwanted interruption
- Solution for SoCs$\rightarrow$ AXI Protocol

AXI Introduction - Overview

- Advanced eXtensible Interface $\rightarrow$ **AXI**
- AXI is a Burst-based Protocol
- Latest revision in 2010 (AXI4, AXI4-lite)
- Freely available on ARM
- Adoped by Xilinx and other ventors as communication bus
- Used on Zybo boards (which are used in the praticals)

AXI Introduction - Interface

There are two different types of AXI Interfaces, namely the AXI memory mapping and the AXI4-Stream.

- **AXI Memory Mapping:**
  $\rightarrow$ **AXI4:** Capable of doing memory map burst transaction up to 256 data transfer cycles per address phase.
  $\rightarrow$**AXI4-Lite:** Utilized for the single bit memory map transaction. No Burst support.
- **AXI-Stream:** There is no address channel and it allows an unlimited burst transaction between the master and slave.

AXI Introduction - Features

- separate address, control and data phases
- support of unaligned data accesses
- burst-transfers with different modes
- separated and independent read and write channels
- out-of-order execution for transactions with different thread-ids on the same master port
- in-order execution on different master ports

AXI Introduction - Interconnect

- The AXI Interconnect IP connects a AXI Master devices to a Slave devices
- allows N:M connect. btw. masters and slaves
  $\rightarrow$ multi-master
  $\rightarrow$ multi-slave
- AXI Master is connect. to Interconnect Slave
- AXI Slave is connect. to Interconnect Master

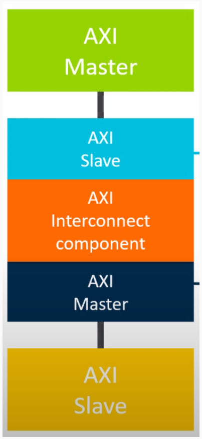


Figure 1: simplified Comm. Illustration

AXI Thread IDs

AXI Thread IDs (TIDs)

- TIDs allow a single master port to support multiple threads
- Each Thread has **in-order** access to the AXI Address Space
- Different Threads may execute their transaction **out-of-order**
  $\rightarrow$ for threads that use the same master port
- Solves problem of waiting for slow peripherals

AXI Thread IDs (TIDs) - Example

A example of an out-of-order execution on a master port with two threads which execute their transactions in-order

Padding

Thread 1	Thread 2		Master Port
in-order	in-order		out-of-order
read1	write1	$\rightarrow$	T1 read1
write1	write2		T2 write1
read2	read1		T2 write2
			...

AXI Handshake Mechanism

AXI Handshake Mechanism - Signals

A handshake uses specified signals to transfer data via the AMBA AXI protocol.

- xVALID $\rightarrow$ indicates that data is ready asserted by sender
- xREADY $\rightarrow$ indicates that receiver is ready for data

AXI Handshake Mechanism - Constraints / Rules

Their are specific rules/constraints for signals:

- A source (slave) must not wait for a high xREADY to assert a xVAILD
- Once a xVAILD is asserted by a source (slave) it must wait for the occurring of a handshake ($\rightarrow$ must wait until xREADY is asserted by destination (master))
- PAYLOAD is considered transferred when both are high
- PAYLOAD only valid for one cycle when both are high

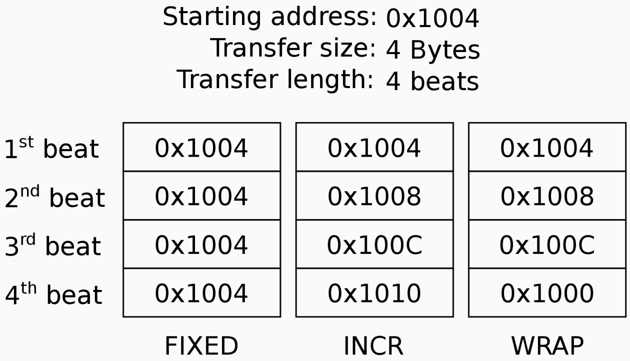
AXI Handshake Mechanism - Bursts

Burst enable a transfer of multiple data in one transaction per address phase.

The AXI interface supports three different types of bursts:

- FIXES (writes data from address multiple times)
- INCR (writes data from address to address+len)
- WRAP (writes data from address to address+len with a wrap-bound)

AXI Handshake Mechanism - Bursts



Starting address:	0x1004
Transfer size:	4 Bytes
Transfer length:	4 beats

	FIXED	INCR	WRAP
1st beat	0x1004	0x1004	0x1004
2nd beat	0x1004	0x1008	0x1008
3rd beat	0x1004	0x100C	0x100C
4th beat	0x1004	0x1010	0x1000

Figure 2: Illustration of different Burst types

AXI Handshake Mechanism - Bursts transfer examples

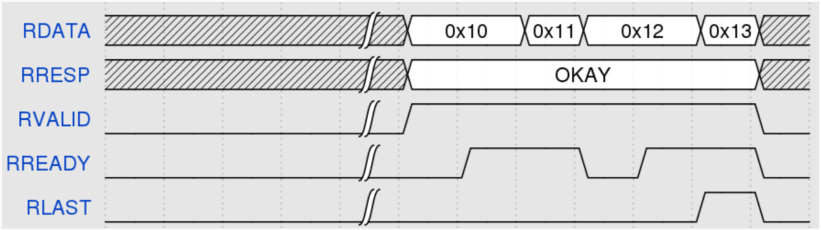


Figure 3: interrupted Burst transaction

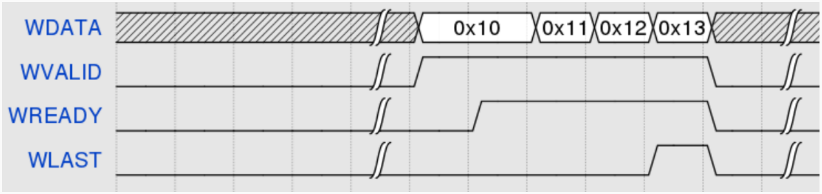


Figure 4: uninterrupted Burst transaction

AXI Channels

AXI Channels

The AXI Interface specifies 5 Channels, where each one is independent (expect of some basic ordering rules) and has its one signals for a handshake.

- Read Address channel (AR)
- Read Data channel (R)
- Write Address channel (AW)
- Write Data channel (W)
- Write Response channel (B)

AXI Read Channels Overview

Padding

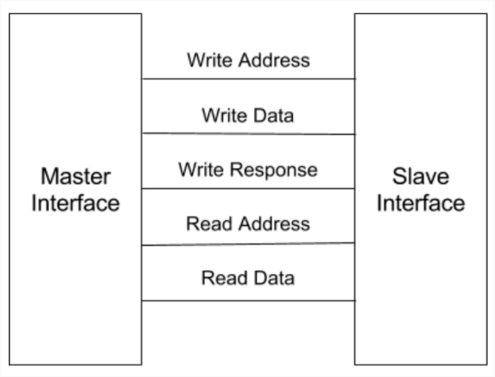


Figure 5: Channel connections between master and slave interface

AXI Channels - Read Phases

To read data these 2 phases have to be completed:

- A master has to send a read-request over the ”read-address channel” to the slave to read data
- The slave will then transmit the loaded data via the ”read-data channel” to the master

AXI Read Channels Overview

Padding

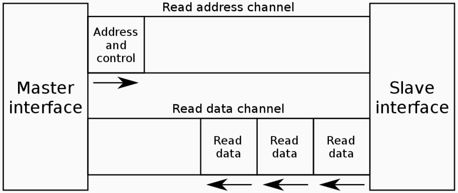


Figure 6: AXI Read Address and Read Data channels.

AXI Channels - Write Overwiew

To write data these 3 phases have to be completed:

- A master has to send a write-request over the "write-address channel" to the slave to write data.
- Then the master must provide the data on the "write data channel" to the slave.
- After the transfer the slave will led the master know if the transfer was successful via the "write response channel" to the master.

AXI Channels - Write Channels Overview

Padding

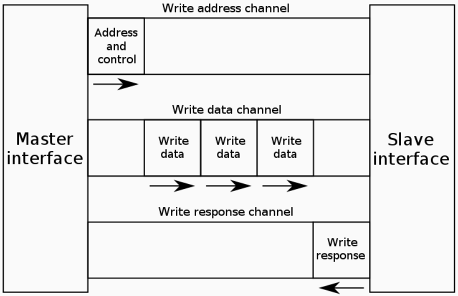


Figure 7: AXI Write Address, Write Data and Write Response channels.

AXI Transactions

AXI Transaction - Read Transaction Example

A master requests 4 beats of data from the slave. The slaves returns the requested 4 beats of data to the master.

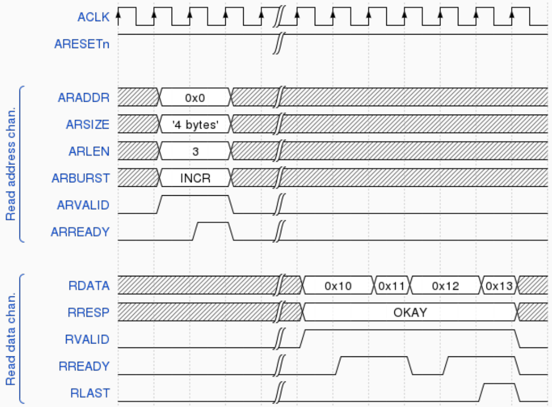


Figure 8: AXI Read Transaction

AXI Transaction - Write Transaction Example

A master requests to write 4 beats to the slace. The slave reads the 4 beats data and sends "OKAY" to the master if everything was successfully.

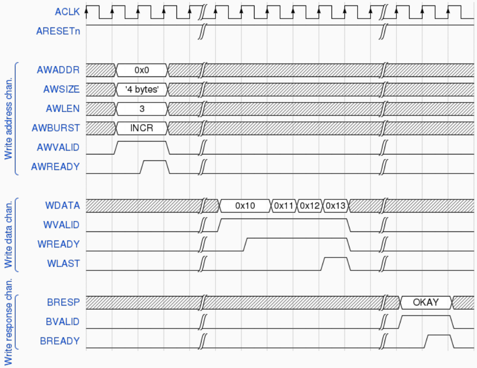


Figure 9: AXI Write Transaction

References

References i

[1] Xilinx. *"AXI Reference .Guide"* https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/v13_4/ug761_axi_reference_guide.pdf p 46. January 18, 2012

[2] Arm Holdings. *"AMBA AXI and ACE Protocol Specification"*. developer.arm.com. pp. 45–47. 5 July 2019.

[3] Arm Holdings. *"AMBA AXI and ACE Protocol Specification"*. developer.arm.com. pp. 22-23. 5 July 2019.

References ii

[4] Xilinx. *"AXI Reference .Guide"* https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/v13_4/ug761_axi_reference_guide.pdf p 5. January 18, 2012

[5] Aldec Interns, FAE Intern Program. *"Introduction to AXI Protocol - Understanding the AXI interface"* https://www.aldec.com/en/company/blog/122--introduction-to-axi-protocol

[6] Farhad Fallahlalehzari. *"Demystifying AXI Interconnection for Zynq SoC FPGA"* https://www.aldec.com/en/company/blog/145--demystifying-axi-interconnection-for-zynq-soc-fpga