OpenCores

SoC Bus Review

Author: Rudolf Usselmann

Rev. 0.6
January 2, 2001

Preliminary

Revision History

Rev.	Date	Author	Description
0.5	29/12/00	Rudolf Usselmann	Initial Release
0.6	2/1/01	Rudolf Usselmann	Added a lot more technical information about the busses.

1. Introduction

I have read three SoC interconnect specification: IBM CoreConnect, ARM AMBA and Silicore Corp. Wishbone.

All of these address the same basic goal: connecting IP cores. They all provide basic handshaking and variable data bus sizes. None specifies a clock frequency, which could be a problem when connecting cores from different vendors.

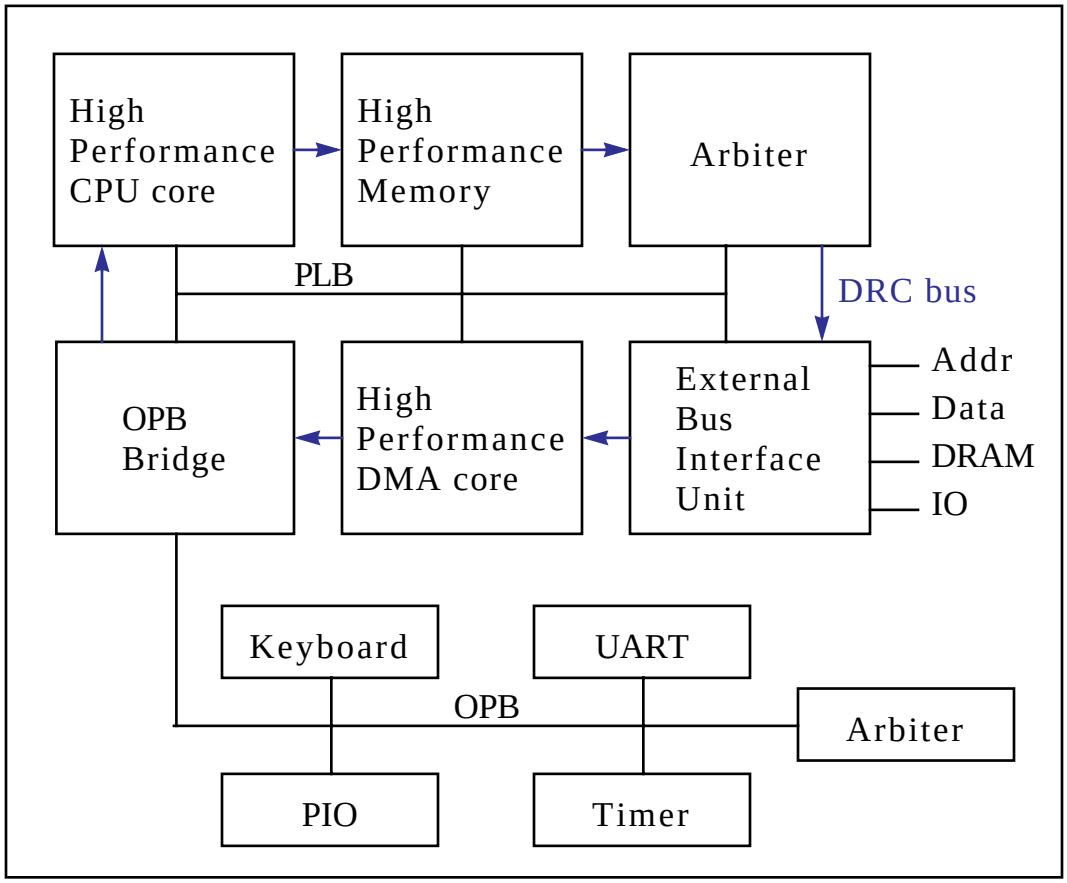
The purpose of this review is to choose a SoC bus for OpenCores, that we would adopt and use in any core development. Standardizing on a common SoC will help us as a community to produce cores that can be easily integrated. Bridges to other SoC standards could be developed and would allow for our cores to be used with other SoC standards as well.

2. CoreConnect

It appeared to be the most complete set of documentation and technically very well though through. IBM has provided specs for each possible building block PLB, OPB, DCR[1], Arbiter and 64+ bit extensions. IBM also provides a testsuite (could not find any information if they actually charge for it or not).

2.1. Logical Bus Structure

Below diagram illustrates the structure of CoreConnect bus.



CoreConnect defines a clear structure for all system components and how they connect. The DRC bus wraps in a daisy chain configuration through all components attached to the PLB.

1. PLB: Processor Local Bus; OPB: On-Chip peripheral Bus; DCR: Device Control Register Bus

2.2. Technical Details

Below is a summary of main features of each CoreConnect bus.

2.2.1. PLB

- High Performance bus (Processor Local Bus)
- Overlapped read and write (up to two transfers per cycle)
- Split transfer support
- Address pipelining (reduces latency)
- Separate read and write data
- 32-64+ bit data bus with
- 32 bit address space
- support for 16-64 byte bursts
- supports byte enabling (unaligned and 3 byte transfers)
- Arbitration support, REQ, GNT and LOCK
- Late and hidden arbitration (reduces latency)
- 4 levels of arbitration priority
- Special DMA modes, such a flyby and memory to memory
- Address and data phase throttling
- Latency timer (ensures latency is kept to a desired level)

2.2.2. OPB

- On-Chip Peripheral Bus
- multiple masters
- 32 bit address space
- separate read and write data bus
- 8-32 bit data bus
- dynamic bus sizing
- retry support
- Burst support
- DAM support
- devices may be memory mapped (DMA support)
- bus time out function (in arbiter)
- Arbitration support, REQ, GNT and LOCK
- Bus parking support

2.2.3. DCR

The device control register (DCR) bus is designed to transfer data between the CPU's general purpose registers (GPRs) and the DCR slave logic's device control registers (DCRs). The DCR bus removes configuration registers from the memory address map, reduces loading and improves bandwidth of the processor local bus.

- 10 bit address
- 32 bit data
- Synchronous and asynchronous transfers
- Distributed architecture

2.3. Applications

I can see CoreConnect to be an important part of a true high performance system, like a workstation. My feeling is that CoreConnect might be to complicated and offer to many features that will be unused in simple embedded applications.

2.4. Cost & License

Free! No Fee, No Royalty! Licensing required.

IBM currently lists 36 companies on their web site who have licensed and might be using CoreConnect.

2.5. Summary

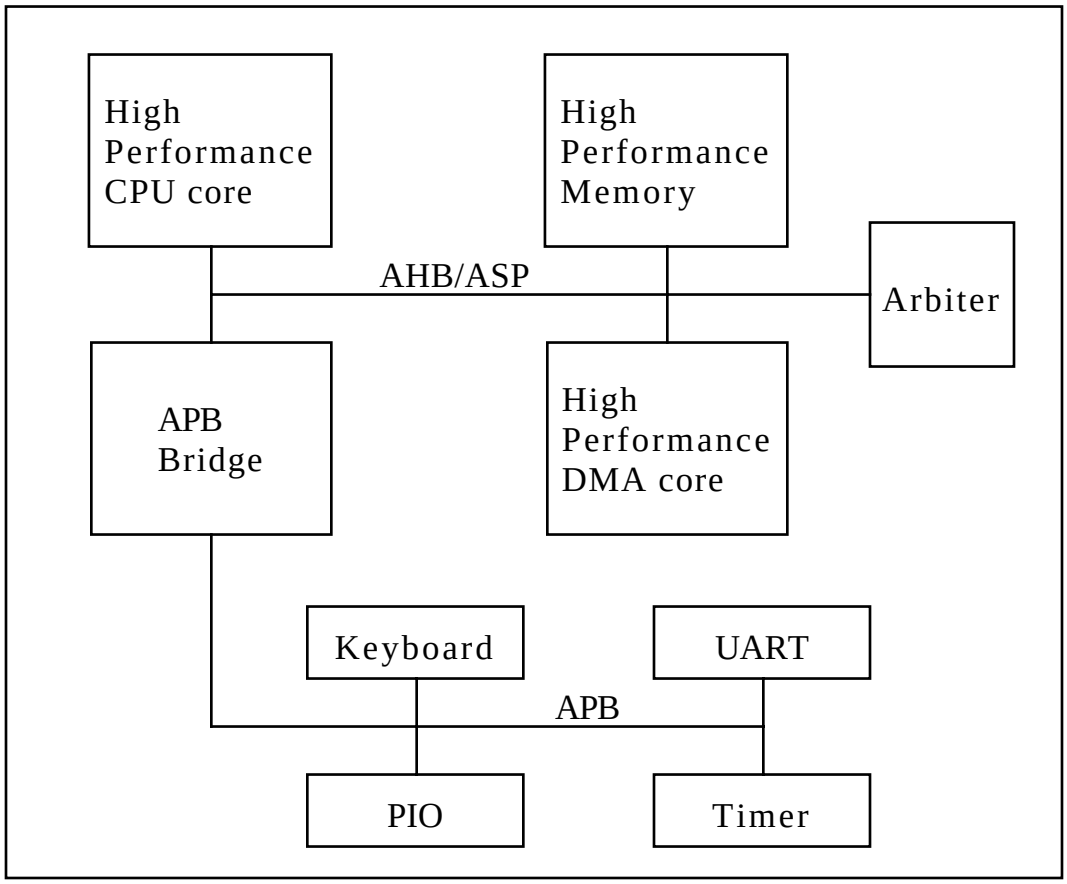
CoreConnect is a complete and versatile solution. It is well thought through and architected. IBM didn’t leave out anything to wish for.
On the other hand, however, CoreConnect might be an overkill for simple applications.

3. AMBA

AMBA is very similar to CoreConnect. ARM includes specifications for AHB, ASP, and APB[1]. The “A” in the abbreviations stands for “Advanced”. However, I could not see anything advanced about this buses at all! Descriptions of arbitration and 64+ bit extensions are integrated in to the main specifications.
On the Silicore Web site (Wishbone), I found a note that AMBA might protected under at least one patent (US 5,525,971).

3.1. Logical Bus Structure

Below diagram illustrates the structure of AMBA bus.



The high performance bus can be either AHB or ASP. Both service the same goal: High Performance System Interconnect. A bridge from AHB or ASP is required to interface to APB.

3.1.1. APB Bridge

It is not clear what the advantage of the APB and the associated bridge to the high performance busses is. According to the AMBA specification, the only function the

1. AHP: Advanced High Speed Bus; ASP: Advanced System Bus; APB: Advanced Peripheral Bus

bridge provides is a simpler interface. Any latencies presented by low performance peripherals are reflected by the bridge to the high performance (AHB/ASP) busses. The bridge itself appears to be a simple APB bus master that addresses the attached slaves and controls them through a subset of the control signals available on the high performance busses.

3.2. Technical Details

Below is a summary of main features of each AMBA bus.

3.2.1. AHB

The AHB is the advanced system bus. It's main purpose if for interconnecting high performance, high through put devices, such as CPU, DMA and DSP. It's main features are:

- High Performance Bus (New Generation Bus)
- Multi Master
- Split transfers
- Single cycle bus master handover
- Non tristate implementation
- 32 - 128+ bit bus width
- Includes a access protection mechanism, to distinguish between such access as privileged and non privileged modes, instruction and data fetch, etc.
- Bursts limited to 16 'beats' max
- Address space limited to 32 bits
- Throttling of data for slower devices provided
- Arbitration support, REQ, GNT and LOCK
- Supports transfers of bytes, half-word and word

3.2.2. ASP

The ASP is the general purpose system bus. It is a high performance interconnect for micro controller and system peripherals. The main features are:

- First Generation System Bus
- Multiple Masters
- Burst Transfers
- Pipeline Transfers
- 32 - 128+ bit bus width
- Includes a access protection mechanism, to distinguish between such access as privileged and non privileged modes, instruction and data fetch, etc.
- Bidirectional data bus
- Address space limited to 32 bits
- Throttling of data for slower devices provided
- Arbitration support, REQ, GNT and LOCK
- Supports transfers of bytes, half-word and word

3.2.3. APB

The APB is the peripheral interconnect bus. Focus here was minimal power consumption and ease of use. The main features include:

- Low performance, low power peripheral bus
- Single Master
- Very Simple, only 4 control signals (plus clock and reset)
- 32 bit address space
- up to 32 bit data bus
- separate read and write data bus

3.3. Applications

Because of the obvious uncertainties of AMBA, my guess is that it would mostly benefit to complete solution providers rather than the general public. Technically I can see it as sufficient architecture for small embedded systems, that are not necessarily performance driven.

3.4. Cost & License

Unknown, waiting for reply from ARM. Presumably free of charge.

3.5. Summary

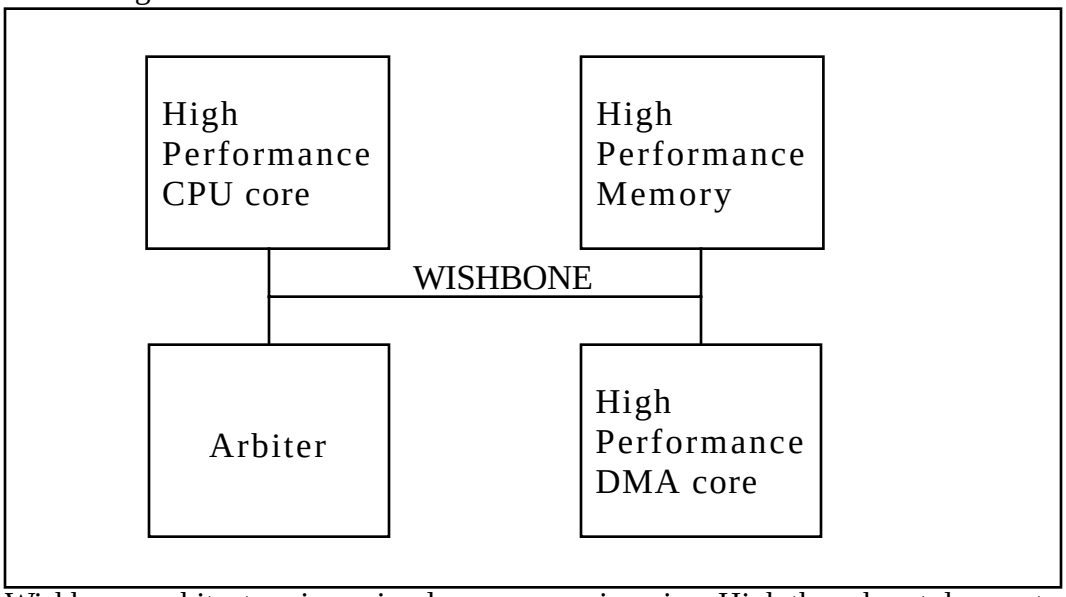
AMBA is a basic SoC bus divided into three different sub busses. Depending on requirements, the system designer has to chose which of the three busses he will interface to. A high performance device has the choice of the AHB and ASP bus, which makes it very difficult for core integrators since both busses try to address the same type of devices. There is no clear path of integrating devices with AHB and ASP busses.
The bridge appears not to provide any intelligence, and might throttle the attached high performance bus to a crawl. Kind of defeating the purpose of having a bridge and different performance busses.
All three busses consist of a address and one or multiple data phases.

4. Wishbone

The Wishbone spec takes some getting used to. It is comprised of RULES, SUGGESTIONS, PERMISSIONS, OBSERVATION etc. (I wish the PDF version of the specification would come with a proper table of contest and ‘clickable’ cross references!) Not having implemented a interface bus, it is hard to say how complete it might be and whether the bus will cover all needs or not. This is especially true for wishbone.

4.1. Logical Bus Structure

Below diagram illustrates the structure of wishbone bus.



Wishbone architecture is as simple as one can imagine. High though put does not always have to be complex. A single bus, addressing almost every need. A system with many components, might want to include two wishbone interfaces: one for high performance blocks; and one for low performance peripherals.

4.2. Technical Details

- One Bus Architecture for all applications
- Simple, compact architecture
- Multi master support
- 64 bit address space
- 8 - 64 bit data bus (expandable)
- Single read and write cycles
- RMW cycles
- Event cycles
- Supports retry
- Supports memory mapped, FIFO and crossbar interface
- Throttling of data for slower devices provided
- User defined TAGs for identifying data transfer types
- Arbitration defined by the end user

4.3. Applications

Because of the simplicity and flexibility of wishbone, it’s application areas are truly limitless. It can be easily utilized in simple embedded controllers and high performance systems. It might leave a few things to wish for, when implementing high performance systems, but it will definitely not stop one from implementing them.

4.4. Cost & License

Wishbone is absolutely free!

4.5. Summary

It appears the simplest of the three busses I have reviewed. It defines only one bus - a high speed bus. I don't see a problem with that, as in a system that needs both a high speed and a slower low performance peripheral bus, two wishbone interface could be provided, which should be simpler, than designing two different bus interfaces.
Users of wishbone might have to create their own substandard of wishbone, specifying data ordering (Little/Big endian) and the meaning of TAGs. Additional features and functionality might also has to be added.

5. The Last Word

As much as I wished to leave political motivation out of this report, I feel it is impossible in today world of patients and lawsuits. Both AMBA and CoreConnect are controlled by large, major corporations (ARM and IBM). Even though they claim that customer comments and suggestions are considered, the final decision of evolution lies with those corporations.

Wishbone appears to be in a state in-between: The authors, have currently full control of wishbone, but are promising to transfer it to a “standards organization” (no time frame is provided). I presume they are talking about IEEE, which will then control the spec. This is not necessarily an advantage for us. I have personally worked on several IEEE standards a voting member, and know that this process is very lengthy, expensive and might produce an useless interface, due to all the changes they could make.

All three busses are fully Synchrounous, using the rising edge of the clock to drive and sample all signals. There is almost no difference in basic operations between the busses. The most differences are in the feature set provided and completeness/ relaxation of the specification.

Both CoreConnect and AMBA, offer a choice of system busses to the designer. A integrator, might face a problem, when he tries to connect a devices designed for the different portions of those interconnects. Bridges might be required to build a complete system. With wishbone, all cores connect to the same standard interface. A system designer may choose to implement two wishbone interfaces in a micro controller core, one for high speed low latency devices and one for low speed, low performance devices.

At the end I feel it would be a wise choice to adopt wishbone as a primary interface to our cores. It's signaling appears to be very intuitive and should be easily adopted to the other interfaces when needed.