A Review of WISHBONE Bus Architecture and Comparison with on-Chip Bus Architectures

Bhankhar Dhvani[1] Samir Shroff [2]

[1]P.GStudent [2]Director

[1,2]Department Of Electronics Engineering

[1]Gujarat Technological University, Ahmedabad, Gujarat, India [2]Pronesis Technology, Ahmedabad, Gujarat, India.

Abstract— WISHBONE is an on-chip interconnection architecture used for communication between IP cores. WISHBONE bus used to interface variety of devices due to its open structural design and many a free IP core with a WISHBONE interface supplied by Open Core Association. Open Core SOC design methodology utilizes WISHBONE bus interface to foster design reuse by alleviating system-on-chip integration problems. In this paper an overview of WISHBONE bus architecture and a comparison with two other on-chip bus architectures (AMBA, CoreConnect) is presented. WISHBONE appears to be gaining more usage than other bus because of its performance parameters with flexibility, portability, reliability and addition data transfer cycle (RMW). WISHBONE is widely uses because its IP cores are freely available requiring neither any license agreement nor any registration.

Key words: WISHBONE, WISHBONE bus, WISHBONE interface, SoC buses

I. INTRODUCTION

Generally, the IP cores are developed independently from each other and tied together and tested by a third party system integrator [1]. This required the creation of custom glue logic to connect each of cores together. By adopting standard interconnection scheme, the cores can be integrated more quickly and easily by the end user.

WISHBONE specification defines the WISHBONE bus as a system-on-chip (SoC) architecture which is a handy edge for use with semiconductor IP cores. It is intended to be used as an internal bus for SoC applications with the aim of alleviating SoC integration problems by nurturing design reuse.

The object behind WISHBONE is to create a portable interface that supports both ASIC and FPGA that is independent of the semiconductor technology and WISHBONE interface should be independent of logic signaling levels. Important reason is to create a flexible interconnection scheme that is independent of type of IP core delivery method (Hard, Soft IP) and can be written using any hardware description language such as a VHDL, VERILOG$^{@}$.

The WISHBONE design was strongly influenced by three factors. First, there was a requirement for a good, reliable SoC integration solution. Second, there was a need for a common interface specification to facilitate structured design methodologies on large project items. Third, they were impressed by traditional system integration solution afforded by microcomputer buses. Another reason to use Wishbone is that it is an Open interconnection architecture, which means that it is free to use for any developer without having to pay a fee.

The rest of this paper is complied as follow. A brief background of WISHBONE interface basic is discussed in section II. Features of WISHBONE are discussed in section III. After that interface specification and types are discussed in section IV and V. WISHBONE bus cycles are discussed in section VI. Termination of cycle types is discussed in section VII. At end comparison of WISHBONE with other on-chip architectures is discussed in section VIII.

II. WISHBONE BASIC

The WISHBONE architects have a attempted to create a specification that is robust enough to insure complete compatibility between IP cores. WISHBONE utilizes "master" and "Slave" architectures which are connected to each other through an interface called "Intercon". A Master is an IP core, for example processor core that can initiate a read or write operation between IP cores, by providing an address and control signals. A Slave is an IP core that responds to read or write operations within a given address-range. The Intercon is all wires and logic between the masters and slaves that lets the data transfers take place. The Intercon requires a "SYSCON" module which generates WISHBONE reset and clock signal for prepare functioning of system.

For an overview of how WISHBONE system may look like at Figure.1.

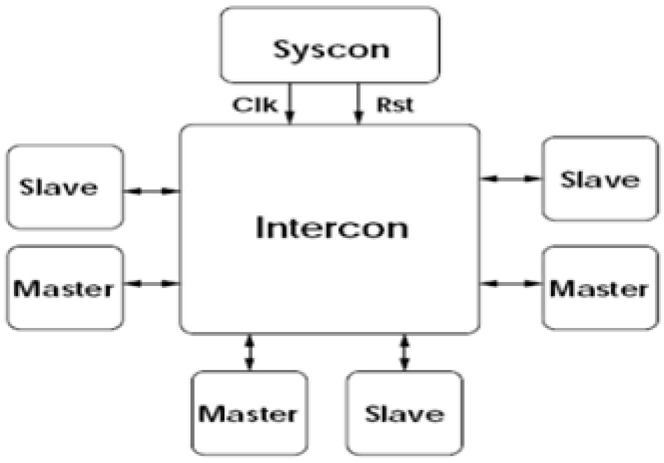

Fig.1: Schematic of WISHBONE Intercon system

Fig.1 shows the WISHBONE Intercon system which contains Masters, Slaves and SYSCON module. WISHBONE Intercon can be designed to operate over an infinite frequency range. This is called a *variable time specification.* The speed of operation is only limited by the technology of the integration circuits.

A. Structured Design Methodology:

WISHBONE is excellent tool for use with structured design methodologies. *Structured design* is popular way to manage complex and large project. When this design practices are used, individual team members build and test small pieces of design. Each of these pieces are designed to interface to a known specifications. When all of the sub-assemblies have been designed and tested, then the full system is integrated and tested.

WISHBONE supports structured design by providing a common interface for all of pieces of the system. Each team member designs their part of the system to WISHBONE standard. These can then be integrated together very easily. Also, the WISHBONE interface is quite simple and does not add significantly to the overall gate count of the finished device at the sub-assembly level. In some cases, it does not add any extra gates in design.

III. WISHBONE FEATURES

By creating standard data exchange protocol, WISHBONE interconnection makes SoC and design reuse easy. Features of WISHBONE [1] include:

– Simple, compact, logical IP core hardware interfaces that require very few logic gates.
– Supports structured design methodologies used by large project teams.
– Modular data bus widths and operand sizes.
– Supports both BIG ENDIAN and LITTLE ENDIAN data ordering.
– Full set of popular data transfer bus protocols including:
    - READ/WRITE cycle
    - BLOCK transfer cycle
    - RMW cycle
– Variable core interconnection methods support point-to-point, shared bus, crossbar switch, and switched fabric interconnections.
– Supports single clock data transfers.
– Supports normal cycle termination, retry termination and termination due to error.
– MASTER / SLAVE architecture for very flexible system designs.
– User-defined tags. These are useful for applying information to an address bus, a data bus or a bus cycle.
– Very simple, variable timing specification.
– Documentation standards simplify IP core reference manuals.
– Independent of FPGA and ASIC test methodologies.
– Independent of hardware technology (FPGA, ASIC, etc.).
– Independent of delivery method (soft, firm or hard core).
– Independent of synthesis tool, router and layout tool technology.

IV. WISHBONE INTERFACE SPECIFICATIONS

Specifications describe signaling between the Master interface, Slave interface, and SYSCON module. It also specifies the way to create a proper documentation for WISHBONE compatible IP cores that becomes main driving factor for design reuse.

The interconnection can be described using hardware description language such as a VHDL, VERILOG$^{@}$, and system integrator can be modify the interconnection according to requirement of the design. Hence WISHBONE interface is different from traditional microprocessor buses such as PCI, VME bus and ISA bus.

A. Required Documentation For IP Cores:

Each IP core must be supplied with a WISHBONE datasheet, which describes the interface of the cores. Datasheet is needed to understand the operation between the IP cores and how to connect it to other cores. After that the user can be reused the cores for integration to produce SoC. Datasheet also includes the information about signal naming and logic levels.

B. WISHBONE Interface Signals:

WISHBONE signals are design in reusable manner. Because of that the Master and Slave interface can be connected together using several interconnection methods. These signals are classified into four categories, SYSCON module signals, Master signals, Slave signals, signals common to both Master and Slave.

V. WISHBONE ARCHITECTURE AND CONNECTION TYPE

The WISHBONE architecture solves the problem of integration of IP core by defined how to connect circuit functions together with flexible, simple and compatible way. WISHBOE does not put any specific constraints on design of Intercon. The user can be designed freely as rules of specification and bus cycles are followed. Intercon can support four type of interconnection. These are point-to-point, data flow, shared bus, crossbar switch interconnections.

A. Point-To-Point Interconnection:

A point-to-point interconnection means that a single Master has a direct connection to a single Slave. This is simplest way to connect two WISHBONE cores together. The traffic of interconnection is handled by handshaking protocol. For example, Master interface could be on microprocessor and Slave interface may be on a serial I/O port.

Bellow Figure.2 shows the architecture of system design by using point-to-point interconnection. This includes SYSCON, and WISHBONE cores.

The SYSCON is known as *system controller.* It is used to generate reset and clock signal for the system. The clock output directly fed from the external clock signal. The reset generator produces a single signal reset RST in accordance with the WISHBONE reset timing.

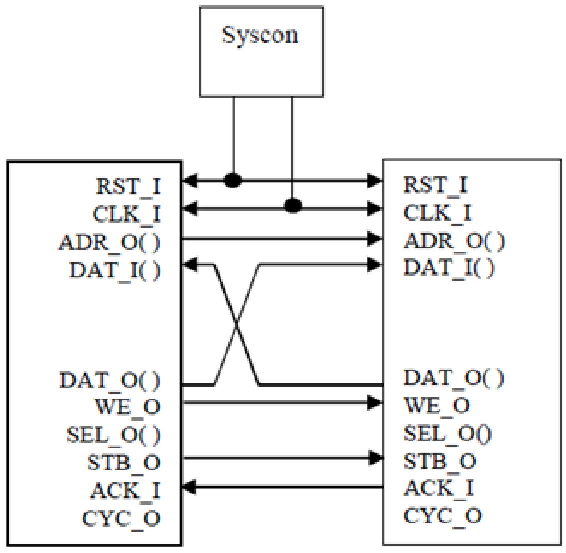

Fig. 2: Point-to-point System Architecture

B. *Data Flow Interconnection:*

The data flow interconnection processes data in sequential manner. Each IP core contains both as Master and as Slave interface. An IP core acts as a Master to next IP core in the sequential chain and as a Slave to IP core prior to it. Here traffic is controlled by Handshaking method. Because of it uses concept of parallelism, speed of this type interconnection is faster. Figure 3 (b) shows data flow interconnection diagram.

C. *Shared Bus Interconnection:*

A shared bus interconnection has a bus that that many Masters and Slaves share with each others. Master starts the bus cycle to Slave, in turn Slave participates in transaction with Master. Only one Master at a time can use the shared bus. The other Masters have to wait for their turn for transfer data. Here, an arbiter is required to decide which master may use the bus at a time and for how much long it can use the bus. So, An arbiter controls bus. Figure.3 (c) shows shared bus interconnection diagram.

D. *Crossbar Switch Interconnection:*

In crossbar switch interconnection there are several ways for data to be transfer between Masters and Slaves. So, two or more Masters can communicate with Slave at a same time, but Slave isn't same. Because of two or more data transfers are made at a same time, it leads to higher data transfer rate then the shared bus interconnection. Here, an arbiter is also required to decide which masters may communicate with which Slave. Figure 3 (d) shows crossbar switch interconnection.

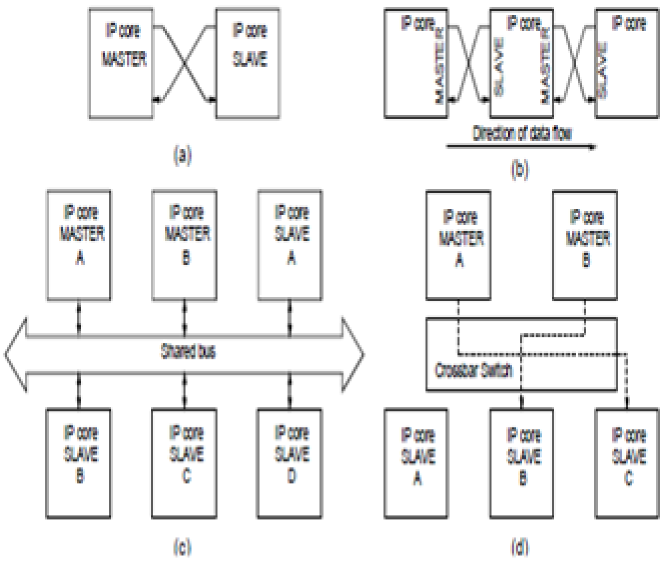

Fig. 3: (a) point-to-point Interconnection, (b) Data Flow Interconnection, (c) Shared Bus Interconnection, (d) Crossbar Switch Interconnection

VI. WISHBONE BUS CYCLES

WISHBONE specification defines three types of bus cycles. These are Single Read/Write Cycle, Block Read/Write Cycle, and Read Modify Write cycle.

A. *Single Read/Write Cycle:*

A Single Read/Write Cycle means that is only one data transfer done at a time. As example, Master wants to perform read operation by present valid address on its address out port. To show that read operation is to be done, then it negates the write enable signal. For indication to Slave that data transfer is ready to start, it asserts cyclic out and strobe out signals. Assertion of cyclic out and strobe out signals are noticed by Slave, and then it places right data on data out port. After that Slave inserts Acknowledged out signal. At next clock edge, data will be read by Master and Master pull down its cyclic out and strobe out signals. So Slave negates its Acknowledged out signal. There by transfer is completed.

B. *Block Read/Write Cycle:*

Blocked Read/Write Cycle is used when transfer need to done in similar way to Single Read/Write Cycle, but Master wants to read or write multiple data arrays. The main difference is negation of cyclic signal from master does not occur until all data is transferred. Flow of data arrays between Master and Slave is control by Strobe and Acknowledge signals. To put wait state, Master pulls down its strobe signal and Slave puts down its Acknowledge signal.

C. *Read Modify Write Cycle (RMW):*

When system having multiple processors that shared memory, then there is possibility to occur the situation in which two or more Masters gaining access of same Slave. So it becomes important to avoid this situation. To prevent such happening of this type situation, a Slave use under must be blocked. And this can be done by asserting semaphore bit. When one Master uses bus, then it asserts semaphore. So other Master reads that semaphore bit is asserted and Slave is accessed by other Master. A Master has to use RMW in this situation. To start the transaction, master first reads the semaphore bit and if it is cleared then Master will assert it. If single read and write operations are done, both Masters could try to get access of Slave at same time. So RMW gives opportunity to Master does a read and write operation before other Master may use the bus. There by it prevent a system crashed.

VII. WISHBONE CYCLES TERMINATION

There exits two different way to terminate the transfer cycle. One is Asynchronous cycle termination and other is synchronous cycle termination.

A. Asynchronous Cycle Termination:

In Asynchronous cycle termination, timing factor will depend upon the Master to Slave though Intercon and back. Slave responds by asserting its Acknowledgement signal when it gets a request. When Master gets respond from Slave, it will terminate the transfer cycle at next clock edge. Figure 4 shows the Asynchronous cycle termination path.

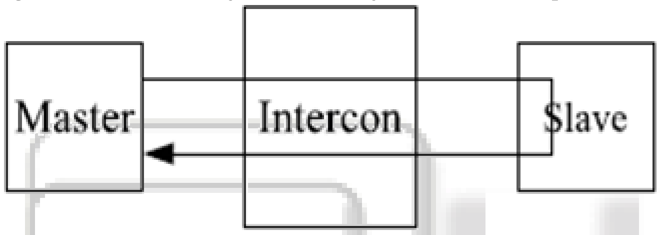

Fig. 4: Asynchronous cycle termination path

B. Synchronous Cycle Termination:

To avoid this long delay, there is simplest method by cutting loop into two parts. This can be done by using Synchronous cycle termination. Master sets it signals and at nest clock edge, Slave responds to them. At following clock edge, Master gets respond from Slave and it negates its signals. In this method, cycle takes two WISHBONE cycles to be complete and uses only half of bandwidth. So clock period can be reduced by shorter loop. Figure 5 shows the Synchronous cycle termination path.

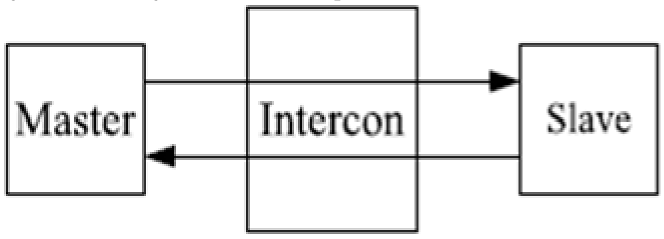

Fig. 5: Synchronous cycle termination path

VIII. WISHBONE COMPARISON

In this section, comparison between WISHBONE, AMBA, and CoreConnect is presented. AMBA and CoreConnect both are well known and well used SoC bus architecture. So comparison is limited with those two buses.

All of these address same basic goal to connect IP cores. They all provide handshaking protocol and variable data bus sizes.

A. AMBA Bus:

AMBA (Advanced Microcontroller Bus Architecture) is basic SoC bus which standard specification has been devised by ARM [5]. AMBA is divided into three different sub buses. These are AHB (Advanced High performance Bus), ASB (Advanced System Bus), and APB (Advanced Peripheral Bus). AMBA is organized hierarchically into two bus segments: System- bus and Peripheral-bus, which are mutually connected through a bridge that serves to buffer data and operations between these segments. 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.

1) Advanced High Performance Bus (AHB):

AHB is intended to address requirement for high performance and high clock frequency system module. It is used as backbone bus with high performance. It supports sufficient connection for processor, on-chip memories and off-chip external memories. It provides high bandwidth by supporting multiple bus Masters.

2) Advanced System Bus (ASB):

ASB is first generation of AMBA bus. It supports one or many Masters. For example, processor and test interface. Interface would be common for Direct Memory Access (DMA) or Digital Signal processor (DSP) which can be included as bus Masters. The external memory and also internal memory have most commonly ASB Slaves.

3) Advanced Peripheral Bus (APB):

APB is used for low power devices. It can be optimized for reducing power consumption and interface complexity. APB appears as a local secondary bus which can be encapsulated as AHB or ASB Slave.

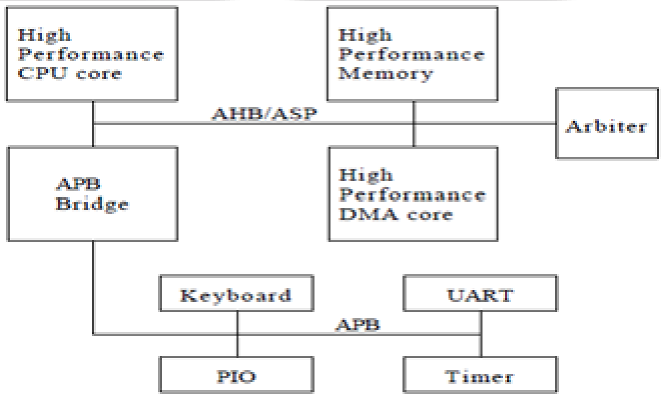

Fig. 6: AMBA Logical Bus Structure

In Figure 6, High performance bus can be either AHB or APB. The main goal is to connect high performance system integration in SoC design.

B. Core connect Bus:

CoreConnect is on-chip bus developed by IBM [4]. It allows the reuse of processor, sub-system and peripheral core, supplied from different sources, and helps these by integrating into a single VLSI design. The architecture of this bus is hierarchical organization. IBM provides specs for each possible building block PLB, OPB, DCR, Arbiter and 64+ bit extensions. It is the most complete set of documentation and technically very well defined.

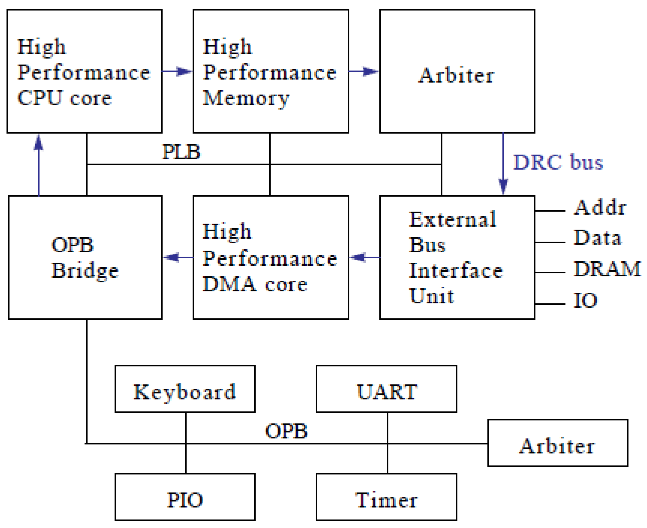

Fig. 7: Core Connect Logical Bus Structure

It defines a clear structure for all system components and also defines how they connect. The DRC bus wraps in a daisy chain configuration through all components. PLB used to attach all components.

There is no difference between buses in basic operations. But the most differences are in their features set provided and specifications [3].

	Features	WISHBONE	AMBA	CoreConnect
1.	Open Architecture	Yes	Yes	Yes
2.	Registration	No	Yes	Yes
3.	Maximum data bus width	64 - bit	1024-bit	128-bit
4.	Maximum address bus width	64 -bit	32-bit	Unknown
5.	Split transaction	No	Yes	Yes
6.	RMW transfers	Yes	No	No
7.	Switched bus	Yes	Yes	Yes
8.	Multiplexed	Yes	Yes	Yes

TABLE I: Comparison between Bus Architecture

IX. Conclusion

Silicore Corporation introduces WISHBONE bus interface which is solution for standard interface required in SoC design. Open cores utilizes WISHBONE for design reuse. WISHBONE interconnection architecture for portable IP cores is a flexible design methodology which can be used with semiconductor IP cores. WISHBONE helps plug-and-play integration which leads low cost, reliable, time-to-market SoC design. WISHBONE signals implements both Master and Slave interface which reduces amount of required interconnection signals. The features of WISHBONE are discussed.

Both AMBA and CoreConnect offer a choice of system busses to the designer. When one tries to connect devices designed for the different portions of those interconnect, an integrator might face a problem. Bridges might be required to build a complete system. All cores connect to the same standard interface with wishbone. With WISHBONE, A system designer may choose to implement two interfaces in a micro controller core, one for low speed, low performance devices and one for high speed low latency devices. So, two level bus is linked with IP cores which improves "design-reuse concept and reduced the power consumption. WISHBONE is differing from other bus in offering support and development tools. WISHBONE offers RMW cycle which none of other bus supports. Plug-and-play utilization can possible with WISHBONE.

At the end, I feel that it's wise to adopt WISHBONE interface for interface and integration of cores because signaling of WISHBONE should be easily adopted for other interface when needed.

References

[1] Opencore organization, "Specification for the: WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores", Revision: B.3, Released: September 7, 2002.

[2] Ayas Kanta Swain, Kamala Kanta Mahapatra,"Design and Verification of WISHBONE Bus Interface for System-on- Chip Integration", Annual IEEE India Conference (INDICON), 2010.

[3] Mohandeep Sharma and Dilip Kumar, "WISHBONE BUS ARCHITECTURE – A SURVEY AND COMPARISON", International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, No.2, April 2012.

[4] IBM Corp., USA," *CoreConnect bus Architecture*," 1999.[Online]at: www.ibm.com/chips/products/coreconnect/

[5] ARM. "AMBA Specification (Rev. 2.0), 1999. Available [Online] at: http://www.arm.com/