Design and Verification of WISHBONE Bus Interface for System-on-Chip Integration

Ayas Kanta Swain[1], KamalaKanta Mahapatra[2]

Department of Electronics and Communication Engineering,
National Institute of Technology, Rourkela, India-769008.
[1]swain.ayas@gmail.com,[2]kmaha2@rediffmail.com

***Abstract*—System-on-Chip (SOC) design is an integration of multi million transistors in a single chip for alleviating time to market and reducing the cost of the design. Design reuse – the use of pre-designed and pre-verified cores is now the cornerstone of SOC design. It uses reusable Intellectual property (IP) blocks that supports plug and play integration and in turn allows huge chips to be designed at an acceptable cost, and quality. Hence to increase the productivity with reduction in design time a standard interface bus protocol is required to perform the plug and play integration. Open core SOC design methodology utilizes WISHBONE bus interface to foster design reuse by alleviating system-on-chip integration problems. In this paper we present the various features of WISHBONE bus interface. Two types of systems have been designed which utilizes DMA master cores and memory slave cores using WISHBONE point-to-point and shared bus interconnection schemes and the final implementations have been done in XILINX FPGA platform. The functionality of the system is verified using Xilinx simulation results as well as board level ChipScope Pro results.**

***Keywords*— SOC, WISHBONE, Point-to-Point, Shared Bus interconnection, Xilinx, FPGA.**

I. INTRODUCTION

Recent advancement in technology allows integration of logic functions of multimillion transistors into a single chip. In order to keep pace with the levels of integration, design engineers have developed new methodologies and techniques to manage the increased complexity in these large chips [1]. System-on-Chip (SOC) design is proposed as an extended methodology to this problem where intellectual property (IP) cores of embedded processors, memory blocks, interface blocks, and analog blocks are combined on a single chip targeting a specific application. [2].

Generally, the IP cores are developed independently from each other and are tied together and tested by a third party system integrator [3]. This required the creation of custom glue logic to connect each of the cores together. By adopting a standard interconnection scheme, the cores can be integrated more quickly and easily by the end user.

The WISHBONE [3] System-on-Chip (SOC) Interconnection is a method for connecting IP cores together to form integrated circuits. Open core [4] SOC design methodology utilizes WISHBONE bus interface to foster design reuse by alleviating system-on-chip integration problems. With use of this standardize bus interface it is much easier to connect the cores, and therefore much easier to create a custom System-on-Chip. The objective behind WISHBONE is to create a portable interface that supports both FPGA and ASIC which is independent of the semiconductor technology and can be written using any hardware description language such as VHDL and VERILOG® [3].

This paper describes the various issues related to system design using WISHBONE bus interface, its implementation in FPGA. It also evaluates the performance of the designed system in terms of minimum size and maximum speed of the system.

The rest of this paper is compiled as follows. A brief background of WISHBONE interface basics is discussed in section II. Proposed system architectures are presented in section III. System integration issues are discussed in section IV. Verification results are presented in section V. FPGA implementation of the system is demonstrated in section VI. Board level verification of the design using Xilinx ChipScope Pro tool is presented in section VII. Finally a conclusion is drawn in section VIII.

II. WISHBONE BASIC

This section presents a brief background of WISHBONE bus interface and its specifications. WISHBONE utilizes "Master" and "Slave" architectures which are connected to each other through an interface called "Intercon". Master is an IP core that initiates the data transaction to the Slave IP core. Master starts transaction by providing an address and control signal to Slave. Slave in turn responds to the data transaction with the Master with the specified address range. The Intercon is the medium consists of wires and logics which help in data transfer between Master and Slave. The interconnection can be described using hardware description languages like VHDL and Verilog®, and the system integrator can modify the interconnection according to the requirement of the design. This makes WISHBONE interface different from traditional microcomputer buses. WISHBONE interface supports variable interconnection. Master and Slave interface may use four types of interconnections such as, *point to point, dataflow, shared bus and crossbar switch interconnection* [3].

The point-to-point interconnection is the simplest one that allows a single Master interface to connect to a slave interface. The dataflow interconnection is needed for sequential data processing. In the shared bus interconnection two or more Masters can be connected with one or more Slaves. An arbiter is used to allow the master to gain access to the shared bus. Crossbar switch interconnection allows two or more WISHBONE masters to access two or more slaves at the

same time. More than one master can use the interconnection as long as two masters don't access the same slave at the same time. WISHBONE supports all the popular data transfer bus protocols such as single Read/Write, block Read/Write and Read- Modify-Write (RMW). Big Endian and Little Endian data ordering are also supported by WISHBONE [3].

Other bus protocols available in market are AMBA [5], OPB [6], and Core Connect [7]. WISHONE offers almost free royalty, hence reducing the overall cost of the system design. WISHBONE Intercon can be designed to operate over an infinite frequency range. This is called as *variable time specification* [3]. The speed of the operation is only limited by the technology of the integrated circuits.

III. Proposed System Architecture

A. Point-to-Point Interconnection

Fig. 1 shows the architecture of system design using point-to-point interconnection that includes SYSCON, DMA and MEMORY Cores. DMA transfers data to and from the memory using block transfer cycles. These cores are available in the WISHBONE public domain library for VHDL [7].

The **DMA** core is a simple 32-bit DMA unit with a WISHBONE master interface. Two methods of data transfers are supported such as, single Read/Write cycles and block Read/Write cycles. The type of cycle is selected by a non-WISHBONE signal input DMODE. If DMODE input is negated, the DMA generates single read/write cycles, if it is asserted the DMA generates block read/write cycles. In block read/write mode, the DMA initiates eight phases of block write cycle, and then DMA generates a similar kind of block read cycle.

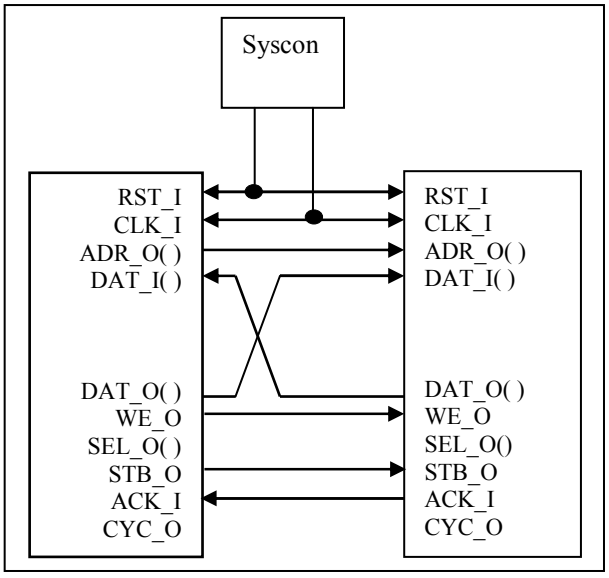

Fig. 1 Proposed Point-to-Point System Architecture

The **Memory** is a simple, 8x32-bit size memory module with WISHBONE Slave interface designed for Xilinx [20] FPGA. It consists of a wrapper that interfaces the Xilinx ram to a WISHBONE Slave interface. Xilinx Core Generator [9] tool is used to create the ram element. Xilinx Core Generator is a parametric core generator that generates optimized core for Xilinx FPGA. It supports single Read/Write, block Read/Write and RMW cycles.

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

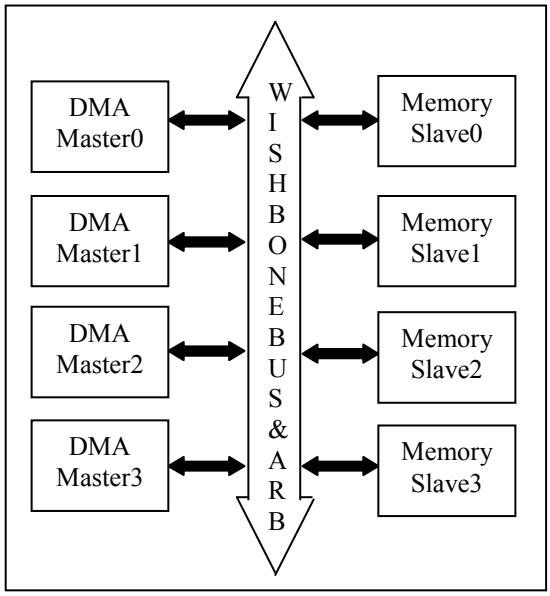

Fig. 2 Proposed Shared Bus System Architecture

B. Shared-bus Interconnection

Fig. 2 shows the architecture of system design using WISHBONE shared bus interconnection scheme. It consists of four DMA Masters, four Memory Slaves, and SYSCON cores as described above. The cores are connected to each other through WISHBONE interface with a shared bus interconnection scheme. An arbiter core is used to grant the access of the bus to a master.

The **ARB** arbiter is a four level, round-robin arbiter. An arbiter is used in shared bus interconnection to grant the access of the bus to a Master. Round-robin grants the access to bus on a rotating basis like a rotary switch.

IV. System Integration Issues

Point-to-point interconnection design is a direct connection of the master core with a slave core and is simple to design. But the system design using shared bus interconnection scheme is much more complex and imposes design complexities to the system integrator during SOC integration. The important factor in designing a system is to how to move the data around the system. The data may be a binary address value or a simple data value. Hence buses are used to move the data around a system. Use of multiplexor based bus reduces the number of pins on a chip, but it requires two clock pulses to move the data and address information in a system, and thus reduces the performance of the system. To

implement logic interconnections multiplexor logic interconnection is mostly used, as these are easier to route in FPGA and ASIC devices than that of three-state logic interconnection. Round-robin arbiter is used to grant the access of the bus to the masters. All the Masters in round-robin arbiter are granted the bus on an equal basis. Every Slave in an interconnection is defined by a specific binary address. In order to access the Slave the Master has to decode the corresponding address. Two types of methods are used to decode the address in a system. These are, *full address decoding, and partial address decoding.* In partial decoding a range of the address is assigned as a decoding address of the Slave module. In full address decoding full address of the bus is utilized to access a Slave. As part of an address is in use, the size of the decoder is minimized and it allows high speed decoders and speeds up the interface. As WISHBONE is using partial decoding technique, the integrator has flexibility in defining the address map of the system.

TABLE I

ADDRESS MAP OF INTERCONNECTION TOPOLOGY

DMA Master	Memory SLAVE	Address
Master0	Slave0	0x00 – 0x07
Master1	Slave1	0x08 – 0x0F
Master2	Slave2	0x10 – 0x17
Master3	Slave3	0x18 – 0x1F

Considering these issues VHDL top module files are created for both the interconnections. For implementing shared bus interconnection topology a multiplexor interconnections and non-multiplexed address and data buses are chosen. Table 1 shows the address map for designing the shared bus interconnection.

V. VERIFICATION RESULTS

The verification of the proposed system architectures were done using Xilinx ISE simulator. The point-to-point interconnection result is shown in Fig. 3 demonstrates that DMA is reading and writing 01234567 data continuously to and from the memory. Fig. 4 shows that initially arbiter grants request to Master1. As write signal 'ewe' is high, DMA starts writing data from the address 5'h08. It continues 8 phases of write operation and write 32'hA5A5A5A1 in the data write output bus '**edwr**'. Then it makes its cycle output '**ecyc**' low indicating arbiter to re-arbitrate. The arbiter grants the access of the bus to the next Master in the round-robin i.e., Master2. Master2 performs 8 phases of write signal and write 32'hA5A5A5A2 to the '**edwr**' bus. After Master2 leaves the bus Master3 gains the access of bus and performs a block write operation as defined in the design and writes 32'hA5A5A5A3 to the output bus. Finally, Master0 is in the loop to get the access of the bus and to write data value 32'hA5A5A5A0 to '**edwr**' bus. The simulation results show that the system interconnection functions accurately. It is also clearly visible that all the slaves are accessed by their corresponding address values defined in the address map.

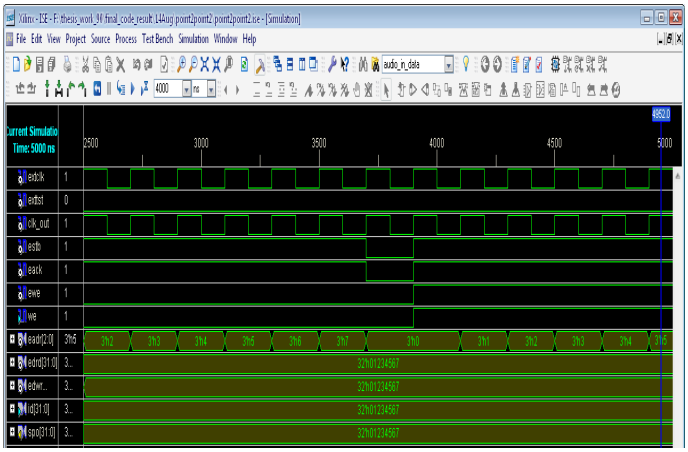

Fig. 3: Simulation Results of Point-to-Point Architecture using ISE Simulator

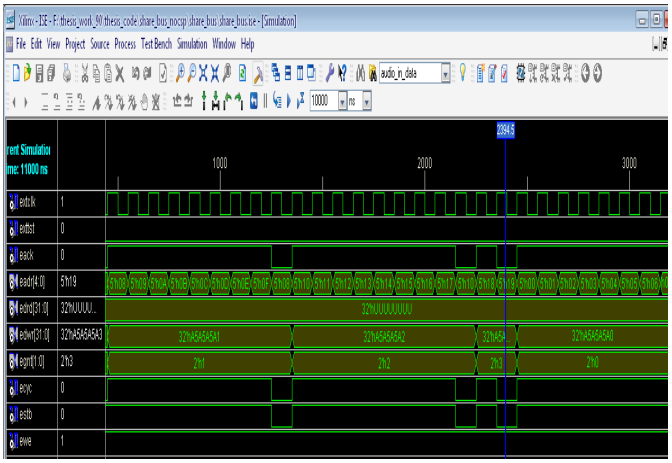

Fig. 4: Simulation Results of Shared Bus Architecture using ISE Simulator

VI. SYSTEM IMPLEMENTATION IN FPGA

The system implementations of two types of architectures were done in two types of XILINX FPGA: Spartan3e and Virtex-II Pro. For memory implementation Xilinx distributed RAMs are chosen. Table 2 and Table 3 show the point-to-point and shared bus system implementation results in two types of FPGAs. Xilinx ISE 9.1 software is used as implementation environment. The results show the point-to-point system design utilizes 40 slices and shared bus system utilizes 292 slices in Spartan-3E and Virtex-II Pro FPGAs. The maximum speeds of the implemented systems are listed in the tables.

TABLE II

POINT-TO-POINT SYSTEM IMPLEMENTATION RESULTS

Device Type	xc3s500e-4fg320 (Spartan3E)	xc2vp30-7ff896 (Virtex-II Pro)
No. of Slices	40	40
No. of Slice Flip Flops	69	69
No. of 4 input LUTs	15	15
Max Speed	248.675 MHz	450.113 MHz

TABLE III

SHARED BUS SYSTEM IMPLEMENTATION RESULTS

Device Type	xc3s500e-4fg320 (Spartan3E)	xc2vp30-7ff896 (Virtex-II Pro)
No. of Slices	292	295
No. of Slice Flip Flops	416	416
No. of 4 input LUTs	459	472
Max Speed	118.312 MHz	219.896 MHz

VII. BOARD LEVEL VERIFICATION USING CHIPSCOPE PRO

The board level verification of the system architectures have been done using XILINX ChipScope Pro [9] tool and the results are shown in the Fig.5 and Fig. 6. Fig. 5 shows in the point-to-point system DMA is continuously write and read a data of value 32'0123456 to and from the memory.

Fig. 6 shows Master0 initially requests for the bus by asserting ECYC_OBUF signal. Arbiter first grants the access of the bus to Master0. Master0 generates eight phases of block write and read signals. Master0 writes a data of 32'hA5A5A5A0 to the memory and read the same data again from memory as shown in EDWR and EDRD bus signals. The arbiter grant signals, WISHBONE acknowledge, cycle, strobe and write signals are shown in the Fig. 6**.** The data read and write by Master1, Master2 and Master3 are 32'hA5A5A5A1, 32'hA5A5A5A2, and 32' hA5A5A5A3 respectively.

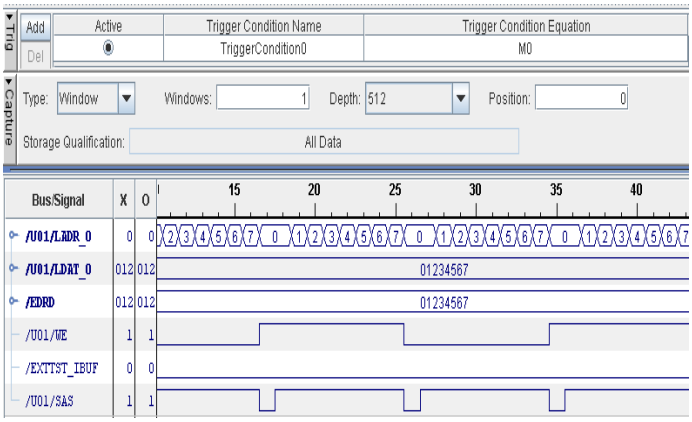

Fig. 5: Board Level Verification Results of Point-to-Point Architecture

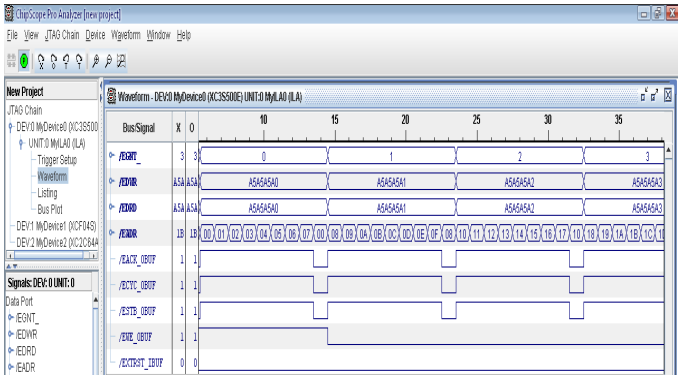

Fig. 6: Board Level Verification Results of Shared bus Architecture

VIII. SUMMARY AND CONCLUSIONS

A 32-bit point-to-point and shared bus interconnection systems are designed and related issues were discussed. The verification of the design is done using XILINX ISE simulator. Finally, by using ChipScope Pro provided by Xilinx the proper functionality of the designed systems are observed. The following conclusions are made from the above discussions: The minimum size requires for implementing point-to-point interconnection system is 40 slices and shared bus interconnection system is 292 slices. WISHBONE interface requires a very little logic overhead to implement the entire interface and gives rise to a highly portable system design that works with standard logic primitives available in most of the FPGA and ASIC devices. Both the interconnections support an operating frequency of more than 100 MHz.

It is also observed that the maximum operating frequency of the design depends on the target device technology. For high speed FPGA like Virtex-II Pro the frequency is higher than the low speed FPGA Spartan3e. Hence, it supports variable timing specification. Low cost, portable and time to market SOC can be designed successfully using WISHBONE bus interface.

ACKNOWLEDGMENT

This work was supported by Ministry of Communication and Information Technology (MCIT), Government of India. Also, CAD tools and boards used in this work are supported by MCIT.

REFERENCES

[1] Resve Saleh and Steve Wilton, "System-on-chip: Reuse and integration," *Proceedings of the IEEE,* vol. 94, No.6, pp. 1050-1069, June 2006.

[2] A. K. Swain and K. K. Mahapatra, "Low Cost System on Chip design for Audio Processing, " in *Proceedings of International MultiConference of Engineers and Computer Scientists -DATICS-IMECS,10*, vol. II, pp. 1380-1385, Hong Kong, 17-19 March, 2010.

[3] Wishbone Specification site, [Online]. Available: www.opencores.org/downloads/wbspec_b3.pdf

[4] OpenCores project site, [Online]. Available: http://www.opencores.org

[5] www.arm.com

[6] www.xilinx.com/products/ipcenter/OPB_Bus_Structure.htm

[7] https://www-01.ibm.com/chips/techlib/techlib.nsf/.../crcon_pb.pdf

[8] http://www.pldworld.com/_hdl/2/_ip/-silicore.net/wishfaq.htm

[9] Xilinx user manual, [Online]. Available: www.xilinx.com

[10] Mohamed A. Salem and Jamil Khatib, "An introduction to open-source hardware development", [Online]. Available: http://www.eetimes.com/