ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(9), DOI: 10.17485/ijst/2016/v9i9/84500, March 2016* ISSN (Online) : 0974-5645

On Efficient Minimization Techniques of Logical Constituents and Sequential Data Transmission for Digital IC

Sahadev Roy[1*], Rajesh Saha[1] and Chandan Tilak Bhunia[2]

[1]Department of Electronics and Communication Engineering, NIT Arunachal Pradesh, Yupia - 791112, Arunachal Pradesh, India; sahadevroy@gmail.com, rajeshsaha29@gmail.com
[2]Department of Computer Science and Engineering, NIT Arunachal Pradesh, Yupia - 791112, Arunachal Pradesh, India; ctbhunia@yahoo.com

Abstract

Objectives: In this paper, a systematic analysis is done on existing logical architecture of on-chip data transmission for a digital integrated circuit like System-on-a-Chip (SoC), Network on Chip (NoC), etc. for fast data communications. **Analysis:** For systematic comparisons, buses are categorized in different topology based on logical architectures and in the different protocol depending on their communication mechanism. The paper illustrates the operation of the arbiter mechanism. The paper present a systematic analysis of the topology and protocols based on sequential data transmission methods, the priority of accession mode, flexibility and compatibility and performance analysis for the cases of different logic structures and clock frequencies. **Finding:** Multiple buses are suitable to connect maximum number blocks to avoid the overloading effect on a single bus. The Matrix bus seems to be the best solution for attaining high-speed block data communication, but not energy efficient where as share bus is the most energy efficient bus but not suitable for high-speed operation. Several existing On-Chip Communication Architecture (OCBCA) in the literature and compare them according to their structure, advantages, limitations and application criteria to achieve full performance by minimized logical constituent and the different functional blocks using different communication channels and interconnections. **Novelty:** Comparison among the all existing on-chip interconnection topology, protocols and minimization techniques of logical constituents.

Keywords: Arbiter, Embedded Cores-Based System-on-a-Chip, High-Speed Data Transmission, Logical Architecture, On-Chip Bus Communication Architecture, On-Chip Bus Protocols, The Sequential Operation of the On-Chip Bus

1. Introduction

The size of the System-on-a-Chip (SoC) and Network-on-a-Chip (NoC) gets reduced rapidly with an evolution of digital Integration Circuits (IC) recently[1,2]. Design costs of ICs are reduced gradually using low power design technique[3]. In SoC, different types of input- output peripherals, IP cores, special purpose processor, memory and many other dedicated systems are connected through On-Chip Bus[4]. Interconnection complexity[5-7] among the devices increase with the size of SoC gets reduced. An on-chip bus is a fundamental tool, which is designed to interconnect functional blocks or devices of a system in a systematic manner by a set of interconnection to transport and exchange information among them. A bus connects several components with a single channel to carry data efficiently from source to the desired destination using different communication protocols. The transport medium physically implemented as the set of separate connections which make up a parallel bus[8]. A parallel bus is the typical implementation choice in almost all widely used On-Chip Bus design based on communication architectures. The components in a SoC design invariably require communicating with each other during application. A typical processing system fetches instructions from memory, otherwise writes to external memories by sending data to an on-chip memory controller[9]. On-Chip Bus, communication architecture plays important roles on the performance of SoC. On-Chip Communication Architecture (OCBCA) must

* *Author for correspondence*

ensure that the multiple coexisting data streams correctly and reliably routed from the source components to proper destinations. Bus-based communication design consists of different logic components such as decoders, arbiters, bridges and communication path, etc. The performance of bus including valid data bandwidth, Dynamic Energy (DE), Dynamic Energy Efficiency (DEE)[10–13], data Transfer Time (TC)[14] and also Wire Efficiency (WE)[15]. There are many On-Chip Bus standard ARM Microcontroller bus standard (AMBA) by ARM[16,17], Core - Connect by IBM[18], ST bus by STMicroelectronics[19], Wishbone by Open cores[20] and Avalon by Altera[21] have been proposed since 1990 for their special purpose IC, which has worked based on their on-chip bus communication architecture.

2. Logical Constituents Minimization Techniques

Different digital ICs, SoC builds with different functional blocks and for data transmission between them, interconnection is required. Every On-Chip Buses followed some particular communication topology and protocol for data transmission[22–25]. Different minimize logic architecture available to optimize data transmission process. Logic structure or topologies play an important to design any digital IC. Some traditional logical constituents' minimization topologies discussed here.

Topology refers to how the various components and functional blocks of SoC are mapped to the buses and also how the buses interlinked. CPU, DSPs DMA controllers or other Master components are initiating the communication transaction[26] and slave, components like USB, UART and on-chip memory to give respond of transactions which are initiated by master[27]. It required a trade-off among the cost, complexity, power and performance profiles of the communication architecture. Many topologies exist, ranging from single shared bus to more complex structures such as bus token ring, hierarchies, crossbar and partial crossbar etc[28,29]. Different buses may have different logic structures and clock frequencies hence for interconnections among them and the bridge is playing an important role. Bridge circuits operations are very complex in nature and it handle an inter-bus transaction, data buffering, frequency conversion, etc. The bridge samples the address and data signals from the master and holds these values for the write transfer on the other bus. Bridge act as a switch and controls the data flow. Few modern OCBCA topologies are explained briefly below.

2.1 Shared Bus Topology

These topologies basically for system bus[26,30], all the components connected to a single communication channel (Figure 1). The main advantage of the shared bus is that it reduced the number of interconnection efficiently and used as one of the most power efficient buses. Here all the functional blocks are interconnected with single buses. So it can carry only one signal at a time because of sharing a single bus. Share bus is a typical example of a clock driven sequential circuit and need clock speed much slower than the CPU. As per example, MSBUS is an ideal example of shared bus topology.

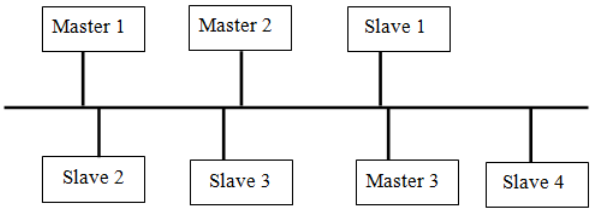

Figure 1. Illustration of share bus topology

2.2 Hierarchical Bus Topology

In this topology, the components are connected to multiple buses that interface with each other using a bridge component[16,30]. Concurrent data transfer is feasible on a different functional block by the bus as shown in Figure 2. Here multiple buses can access at a time, so performance increases but interconnection becomes complicated and consume more power. Hence, a trade-off is required. This topology used by AMBA standards of ARM for interconnection.

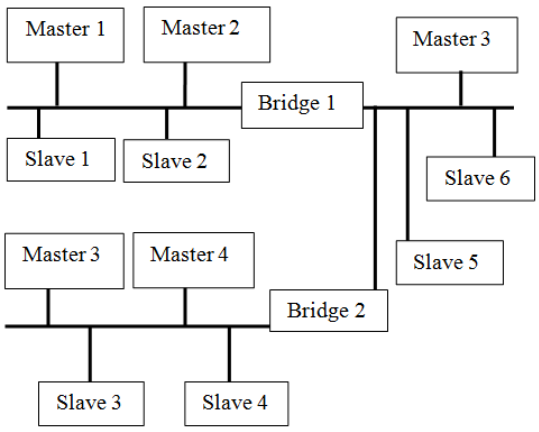

Figure 2. Illustration of hierarchical bus topology.

2.3 Crossbar Matrix Topology

In this topology every master connected to the every slave in the system through separate buses, which would be considering the point to point interconnection like network connection[30] as shown in Figure 3.

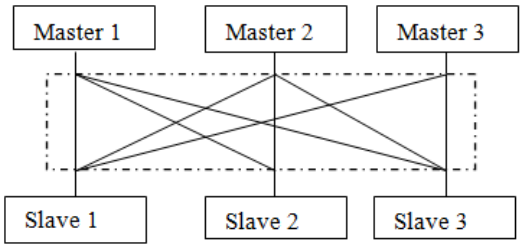

Figure 3. Illustration of crossbar matrix bus topology.

The main advantage of this topology is high throughput, but it has required independent arbiter for each bus. Due to the use of multiple bus placement required a large area and also consumed more power, to overcome this problem an alternate hybrid topology popularly known as partial crossbar topology proposed. The hybrid topologies are the combination of shared bus topology and point to point interconnection as presented in Figure 4. Partial crossbar reduces the performance due to sharing the same channel. AXI protocols of AMBA bus used this topology for interconnection.

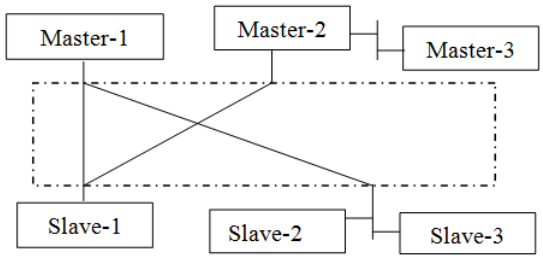

Figure 4. Partial crossbar matrix bus topology.

2.4 Token Ring

In this topology, components are connected to one or more parallel buses as Figure 5, where each element communicated by using a ring interface[30]. This topology is preferable than bus matrix on comparison area. The advantage of token ring bus topology is it support higher clock rate due to a point-to-point connection hence speed of operation increases. IBM core connect used this topology for interconnection.

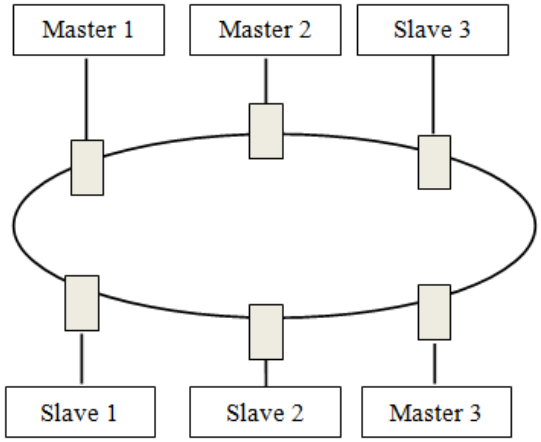

Figure 5. Illustration of ring bus topology.

3. For Efficient Sequential Data Trasmission Protocol

OCBCA protocols are used for communication transaction in an exact manner. OCBCA protocols are also required to manage access of different buses present in the SoC. These are implemented in centralized or distributed bus arbiters, bus widths, decoder and bus clock frequencies buffer sizes and burst transfer sizes.

3.1 Arbitration Scheme

A bus requesting process handling unit, controls the access of bus by the different controller through arbitration mechanism. The arbitration mechanism is a sequential process by which the current bus master used the bus and after that left the control of the bus, after that another master able to access it. It's a component on the bus that uses several criteria to determine which master gains control when more than one master request access to it simultaneously. The criteria used to determine which master gains access to the bus is called arbitration scheme.

Logical constitution of an arbiter mechanism is shown in Figure 6. Arbiter is a significant component of the whole interconnection for handle the multiple requests arises from the various masters based on certain criteria. This mechanism is called arbitration scheme. Based on these criteria, arbiter grants masters to access the bus to handle the slave as per requirement, like read or write data. Decoder is used to select slave based on address

signal of the master through a multiplexer. For arbitration mechanism, three methods are used Daisy Chain method, Independent Bus Requests and Grant method and Polling method[31]. Few suitable arbitration mechanisms like Round Robin (RR) access, priority-based selection, Time Division Multiplexed Access (TDMA), etc.

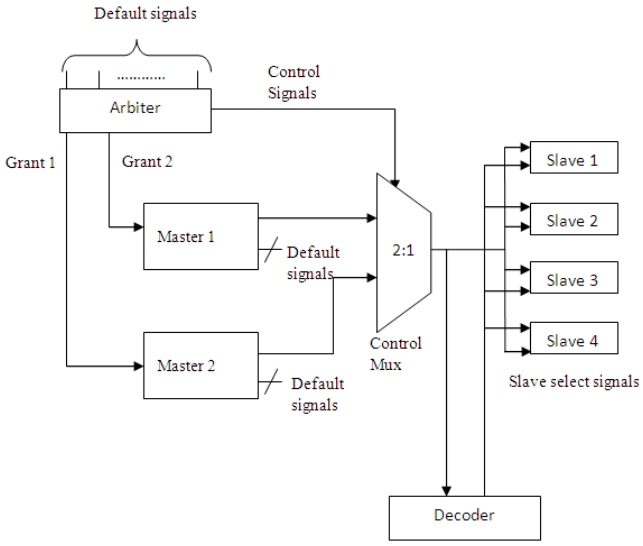

Figure 6. Logical constitution of simple arbiter mechanism.

3.2 Static Priority Scheme

In this scheme, Master allows to access the bus based on its priority[26]. This plan is not fair because low priority master sometimes does not get a chance to access the slave for a long time. Shared bus topology commonly used by this protocol to control the access of bus and different peripheral. Based on this protocols OCBCA supports a burst mode of data transfer where masters negotiated with the arbiter for sending or receiving multiple words of data over the same bus without incurring the overhead of handshaking for each word.

3.3 Round Robin Scheme

In Round Robin (RR) scheme access to the bus is granted in a circular manner so that every master has the equal chance to access the bus[32,33]. In general context switches, which leads to the wastage of time, memory and leads to scheduler overhead is a serious drawback of this scheme[34]. Larger waiting time and response time and low throughput of this scheme are not suitable for the designing of a real-time system.

3.4 TDMA Scheme

In this scheme, components are allowed to access the bus in an interleaved manner using two leveled arbitration[26,32]. The first level uses a timing wheel where each master reserved with a unique slot. If the master associated with the current slot has a pending request, a single word transfer is granted and the wheel is rotated one slot. To overcome the wasted slots, the second level of arbitration identifies, which is known as hybrid arbitration TDMA/RR. TDMA /RR arbitration[30] defines a slot which assigned master does not have a pending communication request and issues a grant to next requesting master as RR manner.

4. Standard Protocols for Efficient Inter IC Data Transmission

There is several On-Chip Bus protocols of many standards have been published to handle the communication needs of emerging SoC designs. Design style varies from manufacturer to manufacturer as per system requirement. Few of popular standards of On-Chip Bus protocols are AMBA by ARM and core connects by IBM, STBus by ST Microelectronics, Wish bone by open core and many more. X. Yang proposed MSBUS[10] and Lahiri suggested lottery bus[26] and many more. Some standard bus architecture discussed in this section.

4.1 Advanced Microcontroller Bus Architecture

AMBA is Advanced Microcontroller Bus Architecture introduced in 1996 by ARM Ltd[35,36]. It is widely used on-chip communication standards which aim to support efficient on-chip communication among the components. Many high-end SoC designed using AMBA bus. The first version AMBA designed with the Advanced System Bus (ASB) and Advanced Peripheral Bus (APB) protocol and second version AMBA designed with Advanced High-Performance Bus (AHB) and APB bus protocol[37]. It defines three standards version and which are discussed below:

4.1.1 AHB

It's used to connect high-speed devices like processors such as ARM, AVR, DSP, off-chip memory interfaces,

DMA controllers, high-speed on-chip memory[9] and high-performance hardware accelerators such as LCD, ASICs MPEG, etc. It supports multiple bus master operation as well as different data transfer mode as a pipeline, burst and split, etc. It supports shared, hierarchy bus topology.

4.1.2 ASB

It's a first generation bus. It supports multiple bus master operation as well as different data transfer mode as a pipeline, burst and split, etc. It supports shared hierarchy bus topology. It's cost effective, but inefficient compared to AHB in a case of supporting complex arbitration protocols which are used to design low-cost SoC.

4.1.3 APB

It's low bandwidth bus which is used to connect a low-speed device like UART, USB and Ethernet etc. The primary objective of design APB bus to optimize the power consumption, interfacing complexity and it's connected with AHB via bridge where the bridge is used to convert high-frequencies signals from AHB into a low-frequencies signal to APB bus. The advantage of APB reduces interfacing complexity efficiently.

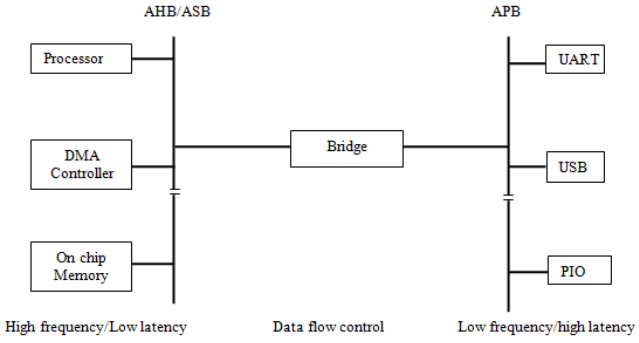

Figure 7. Typical architecture of AMBA2.0

The third version AMBA introduced in 2003 which has the advanced architecture version of AMBA 2.0 and it includes Advanced eXtensible Interface (AXI) bus protocol[17]. AXI bus which extends the versions of AHB bus interface with new features to support the next generation of Multiprocessor System on Chip (MPSoC) designs[10]. The primary objective of the AXI bus is to support high-frequency operation without using the complex bridges and its meet high flexibility in interface and performance requirements of a different set of components and compatibility with AMBA 2.0 APB and AHB interface. There are many differences between AHB of AMBA 2.0 and AXI of AMBA 3.0, among them most significant difference in their way of address handling in burst transfer mode. In every burst data transfer, AHB needs an address to transmit on address bus whereas AXI requires only the first address of the data in burst mode transmit. A schematic diagram of AMBA bus is shown in Figure 7.

4.2 Core Connect

Core Connect bus architecture developed by IBM[18] and it has provided three bus protocols and these are Processor Local Bus (PLB), On-Chip Peripheral Bus (OPB) and Device Control Register (DCR) Bus[38]. The Core Connect architecture can be used to interconnect macros in a PowerPC 440 based SoC. High performance and high bandwidth blocks or cores like CPU, PCI bridge and RAM controller work through the PLB while the OPB handled the low-performance peripherals. DCR bus gives low-speed data paths to pass the status and configuration between the CPU and other on-chip cores. Interconnection among all these protocols is shown in Figure 8.

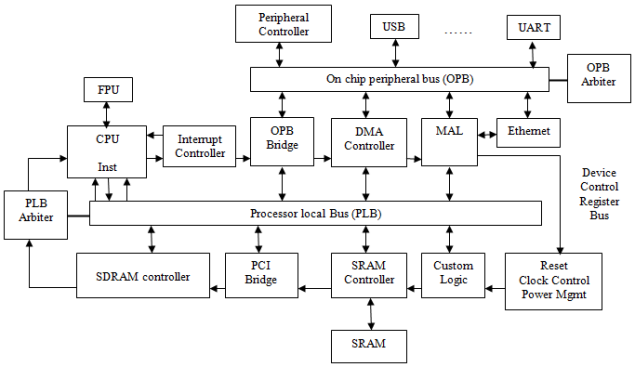

Figure 8. Illustration of core connect bus.

4.2.1 Processor Local Bus

It's a synchronous and high-performance bus protocol which is used to the high-speed component as like AHB of AMBA. It has 32-bit address bus and 16, 32, 64 and128-bit data bus width for read-write operation. It uses pipelined or overlapped arbitration, address transfers, read data transfers and write data transfers with burst transfers capability to perform a maximum of two data transfers per clock cycle.

4.2.2 On-Chip Peripheral Bus

For various on-chip peripherals, OPB provides general design points and its acts independently in synchronous ways at a separate hierarchy level. It is used to connect low-speed component like UART, USB and Ethernet, etc. to the processor through PLB to OPB Bridge.

4.2.3 Device Control Register

Device Control Register (DCR) bus used for read and write operation in lower performance status and configuration registers of any processor. The DCR provides a maximum throughput of one read or write transfer in every two cycles and is an entirely synchronous bus typically implemented as a distributed multiplexer. Relatively slow DCR bus utilizes a ring-type data bus which required minimizing silicon usage.

4.3 STBus

STBus bus architecture developed by ST Microelectronics[19]. It's an evolutionary standard designed for microcontroller consumer application like set top boxes, Digital cameras, ATM networks. Based on performance and complexity three types of bus protocol are present, Type-1, Type-2 and Type-3. All of these are synchronous protocols. Type-1 is a low-speed and simplest bus protocol to interconnect components used for peripheral. Simple controllers that require medium data rate communication with the rest of the system like GPIO, UART. Both the operation read (LOAD) and write (STORE) supported by it with address bus size of 32 bits and a data size support 8, 16, 32 and 64 bit. Type-2 bus protocol supports all the Type-1functionality with additionally provides support for pipelined, SPLIT, compound operations, source labeling and some priority based operations. Type-3 address bus size is 32 bits and it supports read and write operation with data bus size 8, 16, 32, 64, 128 and 256. It also supports compound work and which include READMODWRITE, SWAP, FLUSH, PURGE and USER. The Type-3 protocol supports all Type-1 and Type-2 protocol. It has two essential features as packet shaping which enhance the bandwidth so that throughput gets increased and another one is out of order transfer mode which reduced the latency of data transmission. Principal components of STBus are data path, control logic, master interface and slave interface where control logic unit consists of arbiter scheme, decoder and other logic, etc. Data path has the different topology of communication architecture like shared bus, full crossbar and partial crossbar etc. Another advantage it able to convert both size, type and also able to provide a buffer as per the requirements. The type converters allow communication between two different STBus protocols and a buffer used as a retiming stage between two IPs following Type-3 or Type-2 protocols.

4.4 Wishbone

Wishbone is the Open Cores[20] introduces open source standard bus architecture. It proposes a single high-speed synchronous bus specification to connect all the components in a SoC design and itself used for all application like low-speed to the high-speed module. It supports read and write operation with a 64-bit address bus and data bus 8 to 64 (expendables). It has a significant signal that is Read Modify Write (RMW) which support for semaphore type operation. Because of lack of data transfer mode (out of order, SPLIT transactions) and power management its scope is limited to small and mid-range embedded systems. Topology supported by it is shared, point to point, data flow and bus matrix. Because of simplicity and flexibility and its application is also limited in embedded system.

4.5 MSBUS

MSBUS protocol is a Master-Slave bus which is newly developed protocol in 2014[10]. It has MBUS (control bus) and SBUS (data bus) where MBUS tends to master bus and SBUS tends to slave bus. It has communication architecture of shared bus topology with dedicated MBUS and SBUS. It supports block data transfer mode to improve the performance of the bus. There is being linear transfer mode other than block transfer mode in which data transfer is sequential or one by one like handshaking, pipeline, burst, split and out of order. In block transfer mode bus supports data transfer by block or matrix and which indicated by the Most Significant Bit (MSB) of data size signal, data [10:0]. If data [10] is logic 1, the current transfer is a block transfer and then data [9:6] indicate the width of the block and data [5:0] denotes the height of the block. In this mode, data can be transfer to byte, word, half word, etc.

If SADDR is the initial address of block MWD, which denotes the address gap between data of the vertical neighbors and the term DS indicates the bus transfer size.

Table 1. Comparative tables of different bus protocols

SOC Bus protocols	Transfer Mode	Bus Width		Bus Topology
		Data bus (bit)	Address bus(bit)	
AHB	Linear and block transfer	16/32/ 64/128/256	32	Hierarchy
AXI	Linear and block transfer	16/3/64/128 256/51/1024 bit	32	Bus matrix
MSBUS	Linear and block	32	32	Shared
PLB	Linear	16/3/64/128	32	Hierarchy/ data flow/ring
OPB	Linear	32/64	64	Hierarchy/data flow/ring
STbus Type1	linear	8/16/32/64	32	Shared/bus matrix/ partial bus matrix
STbus Type2	linear	8/16/32/64	32	Shared/bus matrix/ partial bus matrix
STbus Typ3	linear	8/16/32/64/128 bit	32	Shared/bus matrix/ partial bus matrix
Wishbone	Linear	8/16/32/64 bit	Maximum 64	Shared/point to point/ data flow

General formulae to find the line address[10], LinNaddr = [SADDR + N * MWD) << DS] >> DS.

Where the shift operators "<<" and ">>" perform left and right shifts.

5. Comparison and Analysis

In general, any system buses are shared between the controllers and the IO processor for sequential data transmission for digital IC. If multiple controllers want to access any bus, only one of them can allow to use it through the proper requesting process. From the analytical model of DMA, we found that Transfer Latency (TC) is minimum in MSBUS in comparison with AHB and AXI protocols. Moreover, Wire Efficiency (WE) of MSBUS is more than 1.5 times in linear data-transfer mode and two times in block transfer mode compare with other protocols. Hence, it may be used for high-efficiency interconnection. BW of MSBUS and AXI bus protocols is same where MSBUS to AXI ratio of VDB is more than 1 in linear case and 1.5 times in the block transfer mode. DE of MSBUS is half of AHB or AXI and DEE is close to the double of AHB and AXI in block transfer mode. Finally, conclude that MSBUS can transfer twice times of data transfer as in AHB and AXI with less power consumption. Comparison of different bus protocols is shown in Table 1.

Buses can be categories based on the latency and bandwidth utilization. It is better to eliminate the buses that do not meet the bandwidth and latency requirement of the system. For the high-frequency processor, the high-performance system bus is suitable. Separation is essential among the lower bandwidth cores into a peripheral bus. To select the system bus and peripheral bus bridge is an important component. For shared bus, only one communication channel is available then one master allows to access it and others will wait for next transactions. Reduction of SoC size increases the interconnection complexity among the different functional blocks. Power consumption and cost also increase with the number of bus present in the SoC. Though the performance gets reduced the complexity of the interconnection becomes easy hence trade-off required independent arbiter for each bus increases the speed of operation. Due to the use of multiple buses, placement required a large area and also consumes more power.

6. Conclusion

From the above analytical study on logical constituents and sequential data transmission techniques for digital IC, we can conclude that it's possible to design different standards for SoC to enhance the performance of the system based as per requirements or modifying their specification to improve the system performance. Lower bandwidth components are better to connect by using the peripheral bus. Performance of any bus is evaluated based on its latency and data bandwidth. If any single type bus does not meet the required of latency and bandwidth for any particular application are unsuitable for used in that. Multiple buses are suitable to connect maximum number blocks to avoid the overloading effect on a single bus. In such a case, multiple bus is appropriate. The dedicated communication channels or shared bus for present embedded core SoCs offered the best communication performance, but the complexity of the system increases and design become harder. For a low bandwidth bus, bandwidth may be improved by making a combined bus

or increasing the channel, but a bus with a bad latency of any bus is not an easy task to improve the latency. Some case packet loss may occur due to poor latency to complete the task before its deadline over. Both topology and protocol play a significant role to ensure reliable data transmission. Larger waiting time and response time and low throughput buses are unsuitable for the designing of a real-time system.

7. References

1. Roy S, Bhunia CT. On synthesis of combinational logic circuits. International J of Computer Applications. 2015 Oct; 127(1):21–6. Doi no.: 10.5120/ijca2015906311.
2. Rowen C. Engineering the complex SOC: Fast, flexible design with configurable processors.1st edn. Pearson Education Ltd Publication; 2008.
3. Verma G, Kumar M, Khare V. Low power techniques for digital system design. Indian Journal of Science and Technology. 2015 Aug; 8(17):1–6.
4. Xilinx ISE Synthesis and Verification Design Guide. 2015. Available from: www.xilinx.com
5. Karim F, Nguyen A, Dey S. An interconnect architecture for networking system on chips. IEEE Micro. 2002 Sep-Oct; 22(5):36–45.
6. Furbe S. Future trends in SoC interconnect. Proceedings of IEEE International Symposium on VLSI Design, Automation and Test; 2005 Apr 27-29. p. 295–8.
7. Ho WH, Pinkston TM. A design methodology for efficient application-specific on-chip interconnects. IEEE Transaction on Parallel Distributed Syst. 2006 Feb; 17(2):174–90.
8. Damrudi M, Aval KJ. Generalized approach to SoCD sorting on centralized diamond architecture. Indian Journal of Science and Technology. 2012 Sep; 5(9):3288–91. Doi no,: 10.17485/ijst/2012/v5i9/30673.
9. Vennelakanti S, Saravanan S. Design and analysis of low power memory built in self test architecture for SoC based design. Indian Journal of Science and Technology. 2015 Jul; 8(14):1–5. Doi no: 10.17485/ijst/2015/v8i14/62716.
10. Yang X, Andrian JH. A High-Performance On-Chip Bus (MSBUS) design and verification. IEEE Transactions on VLSI. 2015 Jul; 23(7):1350–4.
11. Lakshminarayana A, Ahuja S, Shukla S. High level power estimation models for FPGA. IEEE Computer SoC Annual Symposium VLSI; Chennai. 2011 Jul 4-6. p. 7–12.
12. Selvaraj G, Kashwan KR. Reconfigurable adaptive routing buffer design for scalable power efficient network on chip. Indian Journal of Science and Technology. 2015 Jun; 8(12):1–9. Doi no: 10.17485/ijst/2015/v8i12/59145.
13. Gopal PR, Saravanan S. Low power estimation on test compression technique for SoC based design. Indian Journal of Science and Technology. 2015 Jul; 8(14):1–5. Doi no: 10.17485/ijst/2015/v8i14/61848.
14. Cho YS, Choi EJ, Cho KR. Modeling and analysis of the system bus latency on the SoC platform. Proceedings of International Workshop SLIP; Germany. 2006. p. 67–74.
15. Lee J, Lee HJ. Wire optimization for multimedia SoC and SiP designs. IEEE Transaction on Circuits Syst. 2008 Sept; 55(8):2202–15.
16. AMBA Specification. 2015. Available from: http://www.arm.com/products/solutions/AMBA_Spec.html
17. AMBA AXI Protocol Specification. 2015. Available from: http://www.arm.com/products/system-ip/amba-specifications.php
18. Goel A, William RL. Formal verification of an IBM Core Connect processor local bus arbiter core. Proceedings of ACM 37th Annual Design Automation; USA. 2000. p. 196–200.
19. STBus Communication System: Concepts and Definitions. 2015. Available from: http://freedatasheets.com/downloads/User%20Manual.pdf
20. Wishbone Specification. 2015. Available from: http://www.opencores.org/ open cores
21. Altera Avalon Interface Specification. 2015. Available from: http://www.altera.com/
22. Mitic M, Stojcev M. An overview of on chip bus. Electrical Energy. 2006 Dec; 19(3): 405–28.
23. Sekar K, Lahiri K, Raghunathan A, Dey S. Dynamically configurable bus topologies for high-performance on-chip communication. IEEE Transaction on VLSI Systems. 2008 Oct; 16(10): 1413–26.
24. Lahiri K, Raghunathan A, Lakshminarayaba G, Dey S. Design of high-performance system-on-chips using communication architecture tuners. IEEE Transaction on CAD. 2004 May; 23(5):620–36.
25. Pasricha S, Dutt N, Romdhane BM. Fast exploration of bus-based on-chip communication architectures. Proceedings of 2nd International Symposium of HW/SW Co-design; USA. 2004 Sep 8-10. p. 242–7.
26. Lahiri K, Raghunathan A, Lakshminarayana G. LOTTERYBUS: New communication architecture for high-performance system-on-chip designs. Proceedings of 38th Design and Automation Conference; USA. 2001. p. 15–20.
27. Cordan B. An efficient bus architecture for system-on-a-chip design. Proceeding of 12th Custom Integrated Circuits Conference; San Diego, CA, USA. 1999. p. 623–6.
28. Kim S, Ha S. Efficient exploration of bus-based system-on-chip architectures. IEEE Transaction on Very Large Scale Integration (VLSI) System. 2006 Jul; 14(7): 681–92.
29. Hsiu PC, Hsieh CK, Lee DN, Kuo TW. Multilayer bus optimization for real-time embedded systems. IEEE Transaction on Computing. 2012 Nov; 61(11):1638–50.
30. Pasricha S, Dutt N. On-chip communication architectures: System on chip interconnect. Morgan Kaufmann; 2008.
31. Bystrov A, Kinniment DJ, Yakovlev A. Priority arbiters. Proceedings of IEEE 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems; Eilat. 2000. p. 128–37.
32. Chang HP, Chi HL, HI SK, Jong WC. The efficient bus arbitration scheme in SoC environment. Proceedings of 3rd

IEEE international workshop on SoC; Canada. 2003 Jun 30-Jul 2.p. 311–5.
33. Saha R, Nandi V, Roy S, Bhunia CT. Design and verifications of efficient arbiter of SoC's on-chip bus. Proceeding of IEEE 3rd International Conference on Electronics and Communication Systems; India. 2016. p. 989–92.
34. Shin ES, Mooney VJ, Riley GF. Round-robin arbiter design and generation. Proceedings of the 15th International Symposium on System Synthesis; Kyoto, Japan. 2002 Oct 2-4. p. 243–8.
35. Flynn D. AMBA: Enabling reusable on-chip designs. IEEE Micro. 1997 Jul; 17(4):20–7.
36. ARM. AMBA Multi-Layer AHB Overview. 2015. Available from: http://www.arm.com
37. Hwang S, Kang D, Park H, Jhang K. Implementation of a self-motivated arbitration scheme for the multilayer AHB busmatrix. IEEE Transaction on Very Large Scale Integral (VLSI) Systems. 2010 May; 18(5): 818–30.
38. Halfhill TR. PowerPC 405GP has CoreConnect Bus. Microprocessor Report. 1999; 13(9):8–9.