iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** VII **Month of publication:** July 2020

DOI: http://doi.org/10.22214/ijraset.2020.7047

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VII July 2020- Available at www.ijraset.com

Comparison between Wishbone and APB Protocol

Sumiksha Shetty[1], Kripa A[2], Rachana[3], Rayesunnisha[4], Ankitha[5]
[1]*Assistant Professor,* [2, 3, 4]*Student, Department of Electronics & Communication Engineering, Sahyadri College of Engineering and Management, India*

Abstract: ***The efficiency of bus architecture is an important factor. The performance of an on-chip interconnection architecture, which is used for communication between the functional blocks, depends on the efficiency. The low cost, efficient bus architecture is determined by various factors such as improved width of the bus, more data transfer cycle, a faster clock speed of the bus, and throughput. This paper presents a survey on APB bus architecture and Wishbone bus architecture and a comparison between them. It starts with the introduction and features of the APB bus architecture followed by the introduction to Wishbone bus Architecture and its features and concludes with a comparison between them. The Wishbone Bus Architecture is developed by the Silicore Corporation appears to have more special performance parameters such as additional data transfer cycle (Read-Modify-Write cycle) and the use of a flexible arbitration scheme. It also has an advantage that its IP cores do not require registration, agreement or license and are available free for use.***
Keywords: APB bus, Wishbone Bus, Wishbone Interface, SoC integration, AMBA

I. INTRODUCTION

The increasing design sizes with the rapid advancement in the field of technology and shrinking process technologies led to the advancement of multimillion-gate chips technology, a highly complex billion-transistor within a chip called the integrated circuits (ICs). This leads to a reduction in the size of the system with an increase in system performance and increasing the complexity. As stated in Moore's law, the integration of the transistor in a single chip doubled every two years, as a result, if the given chip area is constant, the integrated chip complexities also increases. The components in a heterogeneous system-on-chip (SoC) might include programmable components such as digital signal processor cores, general-purpose processor cores or application-specific intellectual property (IP) cores and it may also include an analog front end, on-chip memory, I/O devices, and other application-specific circuits [2] which combined perform the single application. Due to the rapidly increasing operation frequency in the chip and growing chip size, the On-chip bus organized CA (computer architecture) is the most complex in SoC technology in addition to testing the performance of topologies like area, delay, and power dissipation. To avoid the problem of imbalance in the computation, communication, and performance, standards of on-chip bus architectures where developed so that it will be convenient for the present and future re-use. The performance on the system is dependent on both efficient bus architecture and CPU speed [3]. With the help of arbitration, the performance can be increased by reducing contention. This paper presents the survey on APB bus architecture, a member of AMBA and WISHBONE bus architecture, and the comparison between them. The paper endeavors to survey and review the features like interface, arbitration techniques, signals and bus cycles in these SoC organized bus architectures.

II. APB

A. AMBA

AMBA, abbreviated as Advanced Microcontroller Bus Architecture is a bus standard which supports On-chip communication protocol which is devised by ARM, The AMBA is one of the leading On-chip bus system which is used in high performance System on Chip (SoC) design. The AMBA is divided into two bus segments namely, the System bus and the Peripheral bus which is connected via a bridge, which serves to buffer the data and perform operation between them.

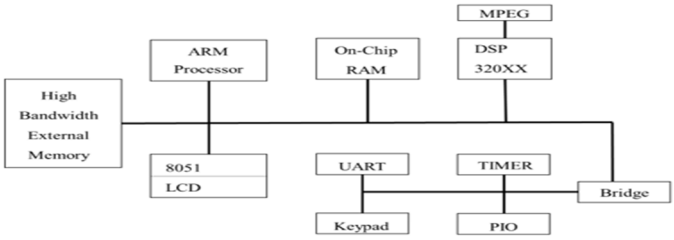


Fig. 1: AMBA based System Architecture

B. Introduction to APB

Advanced Peripheral Bus (APB) is part of the AMBA protocol family and it is used to connect low-speed and low-power peripheral devices. The bridge is assigned as the bus master whereas the other components such as UART, Timer, Keypad etc. are used as slaves. In order to achieve easy interfacing, the APB provides a simple method for addressing by the use of control signals and latch addressing. The APB protocol is not pipelined, thus it can be used to connect low-bandwidth peripherals which does not require the high performance of AXI protocol [7].

1) *Operating States of APB:* The IDLE is the standard state in the APB protocol. It stays in the same state when no transfer is observed. The bus jumps into the SETUP state when a transfer is required, and where the suitable select signal signal, PSELx, is asserted. Only for a single clock cycle the bus waits in the SETUP state and then it moves into the ACCESS state on the next rising edge of the clock. The ACCESS will enable the PENABLE signal which is asserted in the ACCESS state. The write, select, write data signals and address have to remain stable when transition takes place from SETUP state to ACCESS state. ACCESS state controls when to exit the slave through the PREADY signal. These are the conditions, one is if the slave holds the PREADY at LOW then the bus remains in ACCESS state and another is if the slave holds the PREADY to HIGH then the bus exists the access state and if no more transfer are required the bus will return to IDLE state and it will start the same cycle [8][9].

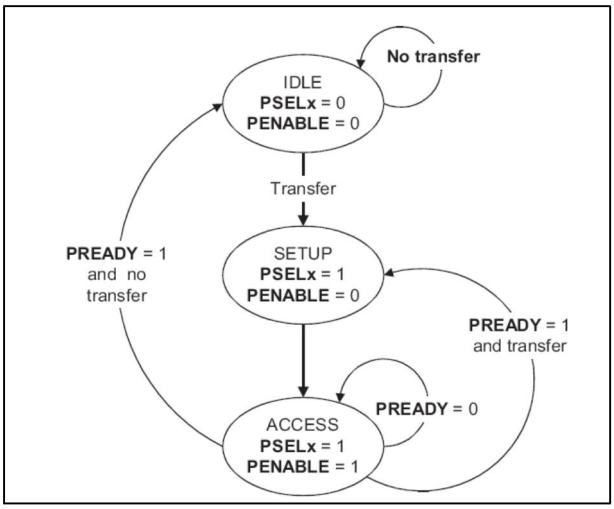


Fig. 2: State Diagram [8][9]

C. APB signals

The signals involved in the APB is listed and the description for each signal is detailed in Table 1.

TABLE I
LIST OF APB SIGNALS [8][9]

Signal	Signal Description
PCLK	clock. The rising edge of PCLK times all transfers on the APB
PRESET	system bus equivalent Reset. The APB reset signal is active LOW.
PADDER	32-bit address bus
PSEL	the slave device is selected and that a data transfer is required.
PENABLE	enable. This signal indicates the second and subsequent cycles of an APB transfer.
PWRITE	access when HIGH
PWDATA	32 bits. Write data. PWRITE is HIGH.
PREADY	ready. To extend an APB transfer
PRDATA	32 bits. Read data. PWRITE is LOW.
PSLAVERR	slave error. This signal indicates a transfer failure.

D. APB bus cycles

1) *Write cycle:* At time T, a write transfer begins with PWRITE, PADDR, PSEL and PWDATA, being registered at PCLK's rising edge. This is called as the SETUP cycle. The next rising edge of clock T2 is called ACCESS cycle where PREADY and PENABLE are registered. When they are asserted, the PENABLE will indicate the initiation of Access phase of the transfer, and the PREADY will indicate that at next rising edge of PCLK the slave will complete the transfer. Until the transfer is completed at T3, the end of Access phase, the PWDATA, PADDR and control signal will remain valid. At the end of the transfer the PENABLE is disabled. Unless the transfer is to be followed by another transfer to the same peripheral, the select signal PSEL will also be disabled [8][9].

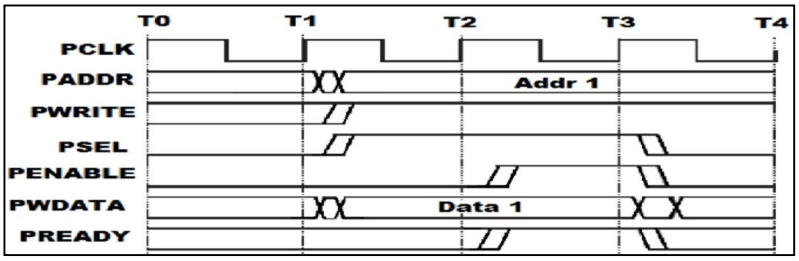


Fig. 3: Write cycle

2) *Read cycle:* At T1, during the read operation the signals PSEL, PENABLE, PWRITE, PADDR are asserted, this is the SETUP cycle. At the clock edge T2 i.e. the access cycle, the PREADY and PENABLE is asserted and PRDATA is read during this phase. Before the read transfer ends the slave must provide the data [8][9].

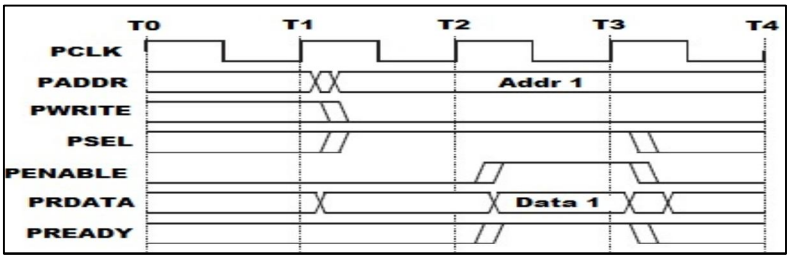


Fig. 4: Read cycle

III. WISHBONE

A. Introduction to WISHBONE

WISHBONE bus architecture is defined as System-on-Chip (SoC) a portable interface for use with semiconductor IP cores and the aim is to reduce SoC integration problems by reusing design. More quickly and easily integrated by end users, all objectives are accomplished at one platform. WISHBONE is combination of both MASTER and SLAVE architecture.

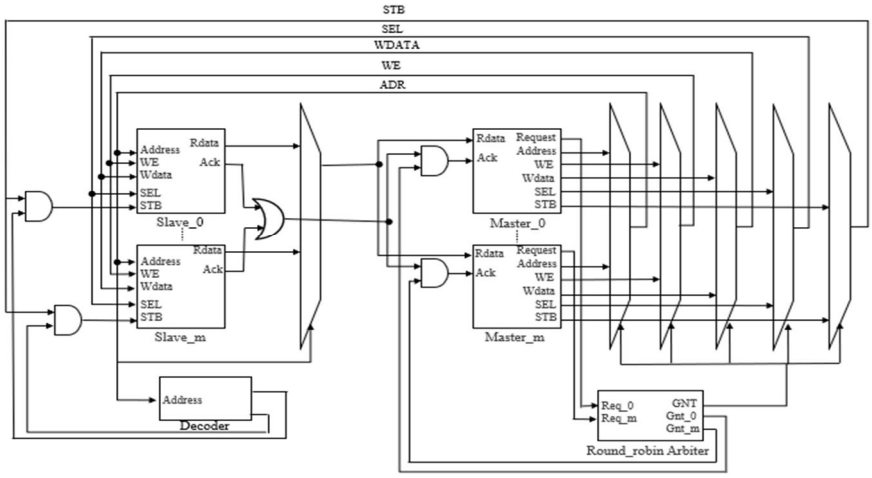


Fig. 5: Wishbone bus Architecture [5]

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VII July 2020- Available at www.ijraset.com

Fig.1 illustrates the overall Wishbone bus architecture. MASTER and SLAVE are the two interfaces defined under the WBA interface. There are four ways of different interface interconnection to connect between MASTER and SLAVE interface called as INTERCON. MASTER and SLAVE interface communicate with Point-to-point interconnection, Dataflow interconnection, Shared bus interconnection, Crossbar switch interconnections. Wishbone bus architecture includes Round Robin, TDMA, CDMA, Static Priority, Token passing etc.

B. WISHBONE Interface

WISHBONE bus interface is responsible for driving or receiving data or power from a bus with the help of electronic circuit involved in it. Interface interconnections classified into four different types they are Point-to-point interconnection, Dataflow interconnection, Shared bus interconnection, Crossbar switch interconnection [7].

1) *Point-to-point Interconnection:* It is the simplest way to connect a single MASTER interface to single SLAVE interface with the help of IP cores. By handshaking signal is used to transmit data between two interfaces, a single MASTER is directly connected to a single SLAVE with the help of INTERCON interface. It does not suitable for SoC interconnection because it supports only single Master and single Slave.

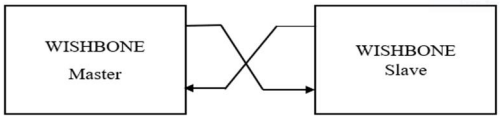


Fig. 6: Point-to-point interconnection

2) *Dataflow Interconnection*: Data is processed in sequential manner, it consisting both a MASTRE and a SLAVE interface. The process is called as pipelining because of data flow from core-to-core and flow of data is controlled by handshaking signal. Data flow interconnection execution time is faster as it uses the concept called parallelism.

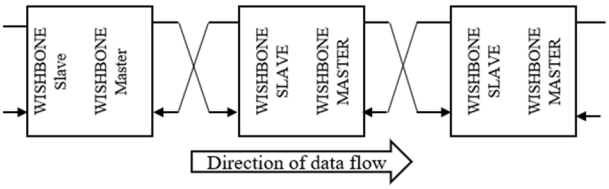


Fig. 7: Dataflow interconnection

3) *Shared Bus Interconnection:* The Shared bus interconnection supports many MASTER and SLAVE architecture. For the first execution time only one MASTER can use the bus, rest MASTER has to wait for their turn and with the help of channel at a time only one MASTER can initiate a bus cycle to a target SLAVE. In Shared bus control has taken by arbiter to which MASTER can use the bus at that moment.

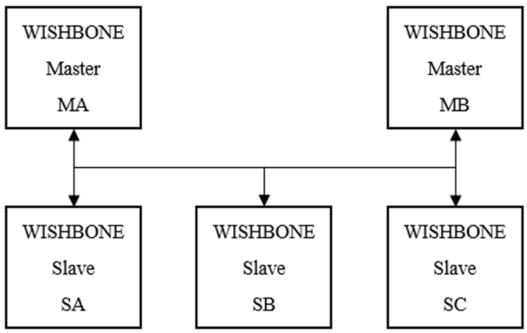


Fig.8: Shared bus interconnection

4) *Crossbar Switch Interconnection:* In Crossbar switch interconnection we can see the multiple MASTERs and multiple SLAVEs, at a time one MASTER can access multiple SLAVEs because of this reason it can be used in multicore SoCs. It has many methods for data transmission between MASTERs and SLAVEs, so data rate in Crossbar switch interconnection is high as compared to Shared bus interconnection. To identify which MASTER is communicating with which SLAVE and to control bus arbiter is used.

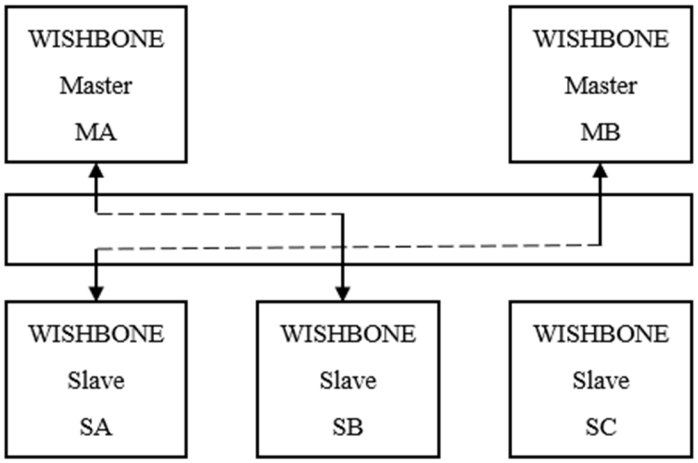


Fig.9: Crossbar switch interconnection

C. Arbitration Techniques

Arbitration techniques are necessary for the fast and efficient allocation of the resources to the appropriate devices. For any on-chip communication protocol in order to acquire access to different interconnections and share resources among different devices, certain mechanisms or set of methods should be followed and these mechanisms are generally known as arbitration techniques [1]. The survey of various arbitration techniques and its use in existing communication protocols are briefly explained below:

1) *Static-Priority:* It adopts arbitration technique where centralized arbiter grants access to the master having highest priority, after inspecting several master requests. Hence it is used in shared bus interconnection where at a time only one master can use the bus. Arbiter is nothing but an authority to use the shared bus among different devices and the mechanism is called arbitration. This arbitration technique is simpler and requires less implementation and area cost. Since it is application specific it is used in both Wishbone and AMBA bus protocol except in the case of Advanced Peripheral Bus. Avalon bus protocol makes use of slave side or distributed kind arbitration.
2) *Time Division Multiple Access (TDMA):* In this arbitration mechanism each master possess a reservation of a fixed time slot. It supports timely fault detection. Efficiency of Round Robin is greater compared to TDMA [10] since there is no wasted time slot in Round Robin where as TDMA sometimes allocates time slots that are not required by the master. TDMA technique is used in industrial applications.
3) *Round Robin:* Most of the crossbar switch interconnections are based on Round Robin technique because it allows equal time slots to access resources among devices in a circular way. As a result more than one master can simultaneously access each slave device. Round Robin is an enhanced version of TDMA. However Round Robin technique involves two level arbitration protocol. Advantages of Round Robin techniques includes easy implementation and equal access of shared resources.
4) *Lottery:* Both Round Robin and TDMA sometimes results in latencies like starvation and wasting of bus cycles which can be resolved by lottery technique. Dynamic lottery bus technique is more efficient and complex compared to static bus technique [11]. In this technique, a lottery is assigned to each master in the case of multiple master design and based on the lottery winner i.e. if the lottery number generated by the lottery manager matches with any of the lottery ticket of the master that are assigned previously, then that master can access the bus. Lottery arbitration technique works based on probability arbitration algorithm [12]. This method resolves competition among the masters for same shared sources like bus, memory etc.
5) *Token Passing:* Ring based architectures uses token passing protocol. When any of the interface receives the token, then it starts the transaction and transfers the same token to the next interface as soon as the interface completes the transaction. This method is advantageous since it allows transaction to be completed within a stipulated time interval and even priority can be assigned to the data that are needs to be transmitted.

D. Wishbone Signals

The below table indicates the signals that are required for the Wishbone interconnect. Most of the Wishbone interface signal use active high logic.

TABLE II
Wishbone Signals

Signal	Name	Description
CLK_I	Clock In	Within the Wishbone interface all the necessary functions of the internal logic are synchronized by the clock input
RST_I	Reset	Restarts the Wishbone interface
ADR_O/I()	Address Out/In	Used to pass binary address array
DAT_I/O	Data In/Out	Used to pass binary data
WE_I/O	Write Enable In/Out	Identifies the READ or WRITE cycle of the local bus cycle
ACK_O	Acknowledged Out	When asserted, it indicates the end of a normal bus cycle
ERR_I/O	Error In/Out	Indicates abnormal cycle termination
STB_I/O	Strobe In	Indicates a valid data transfer cycle
TGC_O()	Cycle Tag Type Out	Contains bus cycle information
TGA_O()	Address Tag Out	Contains address line information
CYC_I/O	Cycle In/Out	When asserted, it indicates the valid bus cycle
SEL_O()	Select Output	Indicates the valid data information on the signal array during READ or WRITE cycle

E. Wishbone Bus Cycles

Wishbone interface supports three types of Wishbone bus cycle. They are Single Read/Write cycle, Block Read/Write cycle and Read Modify Write (RMW) cycle [13]. These cycles use handshaking protocol in order to achieve data transfer between master and slave interface in Wishbone interconnect.

1) *Single Read/Write Cycle:* As the name indicates only one data transfer occur between master and slave interface in Single Read/Write cycle. If a master wants to perform a single read operation initially it offers a valid address on its ADR_O port and negates the WE_O signal in order to indicate the read operation. The master presents SEL_O() signal to identify where it expects the data and then asserts CYC_O and STB_O signal to indicate the start of the transfer. Once the slave identifies the assertion of these two signals it will assert the Acknowledgement signal in order to indicate the presence of valid data and transfer the data to the DAT_I() signal. Then the master reads the data transmitted from the slave in the next clock pulse and negate the STB_O and CYC_O signal. After this process negation of the acknowledgement signal takes place in order to indicate the end of the transfer. In the same way single write cycle occur except that the master transfers the data (DAT_O) to the slave by enabling its write signal (WE_O). The transfer terminates once the slave asserts its acknowledgement signal.

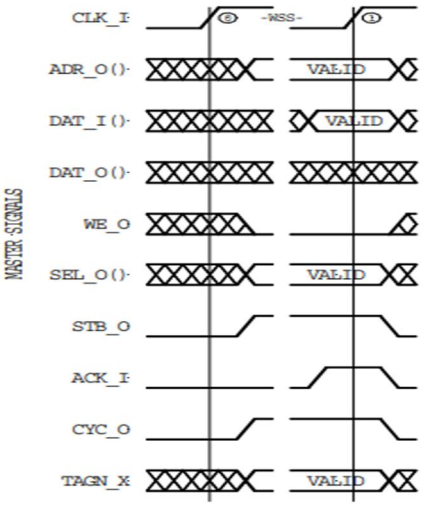


Fig. 10: Single Read Cycle

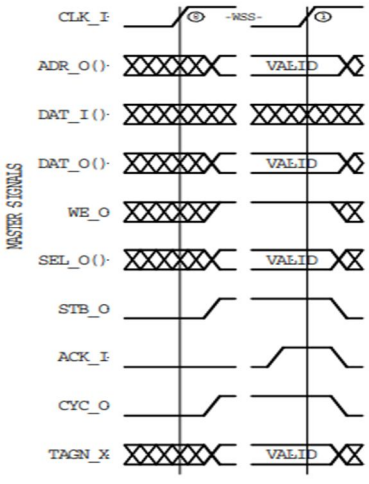


Fig. 11: Single Write Cycle

2) *Block Read/Write Cycle:* Only single data transfer occurs in Single Read/Write cycle but when a master or slave wants to transfer multiple data arrays then we have to switch into Block Read/Write cycle. This cycle works in a similar way when compared to Single Read/Write cycle but the only difference is that until all the data transfer completes between the master and slave, the clock signal (CYC_O) will not be disabled. The flow of data arrays between the master and slave is controlled by using strobe and acknowledgement signal. Through these signals a master can undergo a wait state by negating strobe signal and the negation of acknowledgement results in the wait state of slave signal. Once the data transfer completes then the assertion of clock signal takes place to indicate the end of transfer. When a design involves multiple masters, this type of cycle will be more useful.

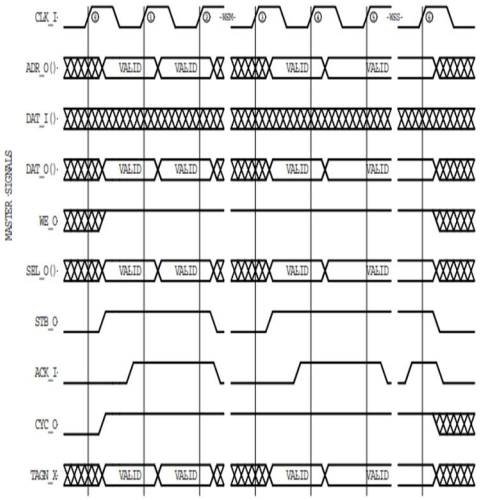


Fig. 12: Block Read Cycle

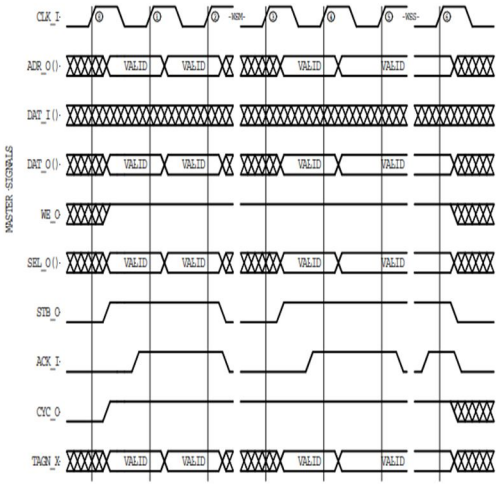


Fig. 13: Block Write Cycle

3) *Read Modify Write (RMW) Cycle:* Instructions like test-and-set (TAS) and compare-and-set (CAS) are the two types of instructions used by the processors during data transmission in RMW cycle. Sometimes multiple masters try to gain access to the same slave. In a system consisting of multiprocessors, there may be possibility that same memory may be accessed by the two different processors at the same time which leads to system crash. So this problem could be resolved by blocking the system under use and thus a special bit called semaphore bit is used which will be asserted when the system is already under use. Only if the asserted state of the bit is changed or clear then the other masters gets access to that particular slave. In this cycle both read and write operations are completed in a single bus cycle. If we use Single Read/Write cycle then another master will try to gain access to the slave in between read and write operation. Serial port, memory and disc controllers use RMW cycle operation.

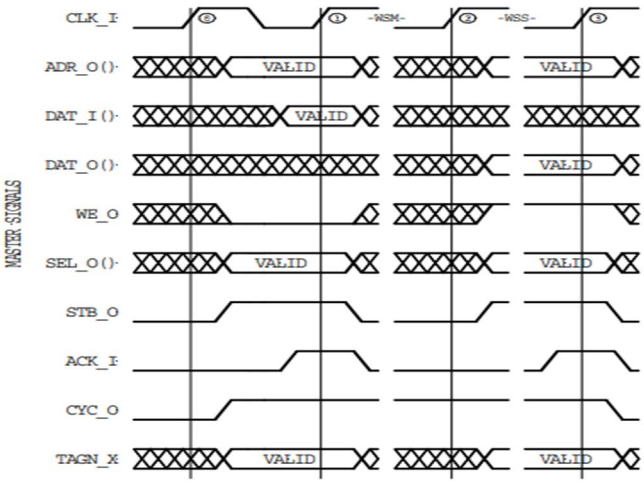


Fig. 14: RWM Cycle

IV. COMPARISON

The owner of the WISHBONE bus protocol is Open Cores whereas the owner of the APB bus protocol is ARM. Both the bus architectures have Open architecture. APB belongs to the family of ARM and requires free registration to be done before using it whereas WISHBONE protocol does not have any registration steps required before using the interface. No license/royalty is required for designing and production of integrated circuits.

In terms of architecture, the WISHBONE architecture does not support hierarchical structure whereas the APB architecture is a part of the hierarchical structure from AMBA [4]. That is, there are two levels of hierarchy in AMBA from ARM. These are Advanced High-Performance Bus (AHB) and the Advanced Peripheral Bus (APB). In the view of architecture, WISHBONE only supports structured design methodologies [6]. WISHBONE protocol does not support pipelined architecture. APB protocol, on the other hand, supports pipelined architecture. Both WISHBONE bus and APB bus are (Multi) MASTER/ (Multi) SLAVE but APB bus does not require any arbitration whereas the arbitration logic in WISHBONE bus is user-defined.

The way SoC components are connected is referred to as a topology. There are different forms namely single shared architecture and more complex architectures include hierarchical buses, token ring, or crossbars. AMBA makes use of the hierarchical bus topology. Except for the crossbar switching, the APB bus protocol does not support any topology. The WISHBONE topology is open-ended and it makes use of point to point, a ring, a shared bus or a cross-bar interconnection network. Multiprocessing feature is also available in WISHBONE due to the capability of multi-master.

Both the APB bus and WISHBONE bus uses a handshaking protocol for communication. To ensure the successful data transfer, this protocol uses request signal and the acknowledgment signals. APB also uses Pipelined, Split, and Burst transfer for the communication purpose whereas WISHBONE deploys Handshaking and Burst transfer only for the specific purpose of data transfer. The data bus width for the APB bus is 8, 16, and 32 bits, and the address bus width is 32 bits. In the case of the WISHBONE bus, the data bus width is 8, 16, 32, or 64 bits and the width of the address bus is 1 to 64 bits. Both APB and WISHBONE has user-defined operating frequency.

V. CONCLUSION

In this paper, the survey is done on APB bus architecture, a family of AMBA from ARM and WISHBONE bus architecture by SILICORE Corporation and is compared considering various factors such as architecture, topology, arbitration, data transfer, bus width and it reveals when it compared to the performance parameters, the WISHBONE bus would be a wise choice because it is simple, flexible and portable. WISHBONE makes it easier for custom SoC and it is easier to connect the cores. The WISHBONE Bus differs from the APB bus over the registration issues, bus architecture, and transfer cycle. The WISHBONE bus architecture offers a special transfer called the Read-Modify-Write (RMW) transfer which is not offered by any other bus architecture. It is easily adopted by the other interfaces when needed.

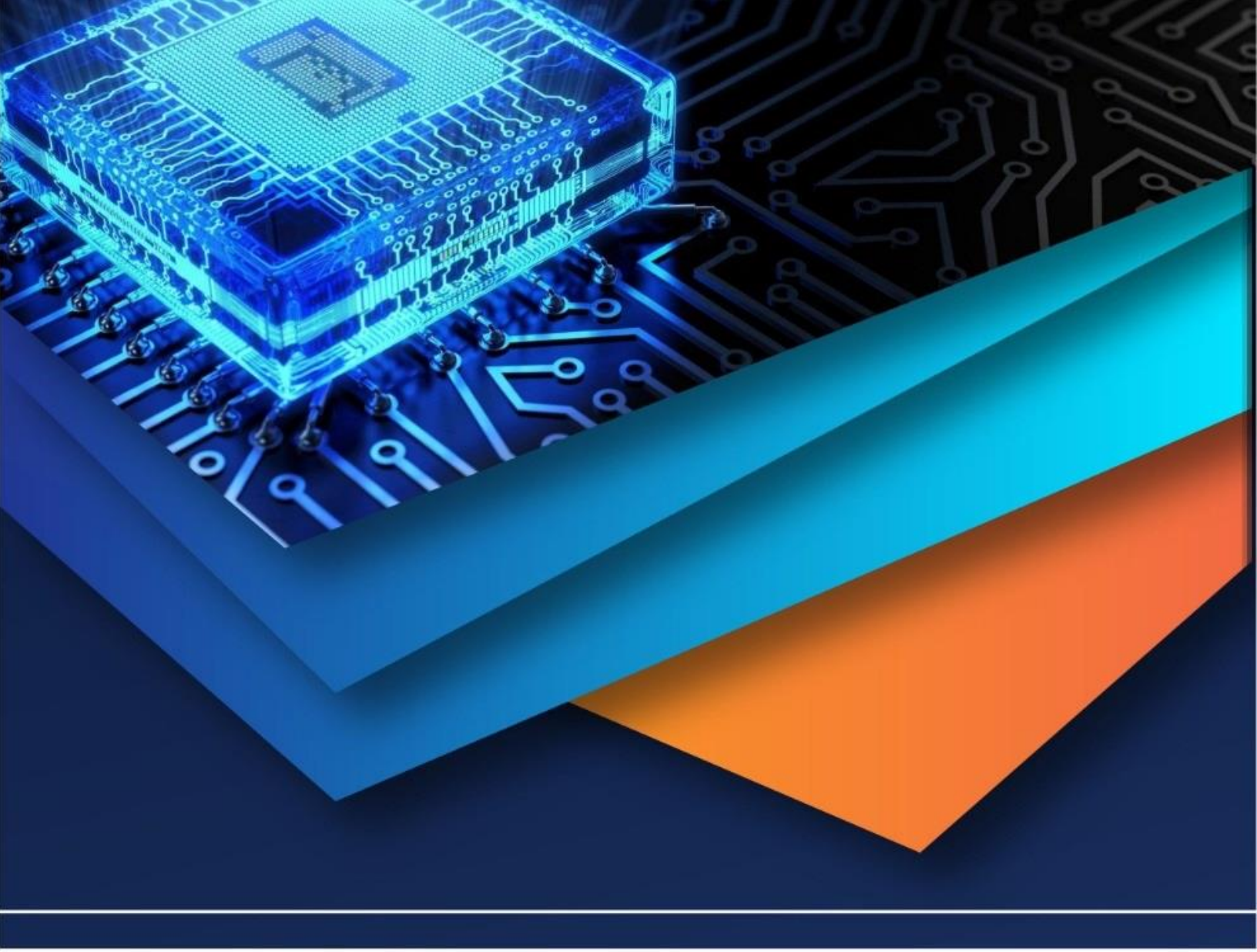
VI. ACKNOWLEDGMENT

The authors would like to thank Ms. Sumiksha Shetty, Assistant Professor for her guidance, Department of Electronics and Communication Engineering, Sahyadri College of Engineering and Management for its support and opportunity.

REFERENCES

[1] J. Gupta and N. Goel, "Efficient bus arbitration protocol for SoC design," 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Chennai, 2015, pp. 396- 400, doi: 10.1109/ICSTM.2015.7225449.
[2] Bennini L., DeMicheli G., Networks on Chips: A New SoC Paradigm, IEEE Computer, Vol. 35, No. 1, January 2002, pp. 70- 78.
[3] Kyeong Keol Ryu, Shin, and Vincent J. Mooney, "A Comparison of Five Different Multi processor SoC Bus Architectures," Digital Systems Design: Proceedings. Euro micro Symposium, Warsaw, Poland, p.209, Sept. 2001.
[4] P. J. Aldworth, "System-on-a-Chip Bus Architecture for Embedded Applications," (ICCD'99) International Conference on Computer Design, pp. 297 -298, 1999.
[5] Wishbone bus architecture available at https://www.researchgate.net/publication/44024600
[6] Richard Herveille, "WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores," Specification Rev. B3, September 7, 2002.
[7] Heli Shah, P. "Chinmay modi P, Bhargav Tarpara P "Design & implementation of advance peripheral bus protocol"." *International journal of scientific engineering and applied science (ijseas) volume-1, issue-3* (2015).
[8] ARM, "AMBA Specification Overview", available at http://www.arm.com.
[9] ARM, "AMBA Specification (Rev 2.0)", available at http://www.arm.com.
[10] M. Ben Slimane, I. Ben Hafaiedh and R. Robbana, "Formal-Based Design and Verification of SoC Arbitration Protocols: A Comparative Analysis of TDMA and Round-Robin," in IEEE Design & Test, vol. 34, no. 5, pp. 54-62, Oct. 2017, doi:10.1109/MDAT.2017.2713352.

[11] Tiwari, B., Chandraker, R., & Goel, N. (2016). Comparative analysis of different lottery bus arbitration techniques for SoC communication. 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT). doi:10.1109/icctict.2016.7514631.
[12] Chang Hee Pyoun, Chi Ho Lin, Hi Seok Kim and Jong Wha Chong, "The efficient bus arbitration scheme in SoC environment," The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications, 2003. Proceedings.,Calgary, Alberta, Canada, 2003, pp. 311-315, doi: 10.1109/IWSOC.2003.1213054.
[13] WISHBONE SoC Architecture Specification, Revision B. 3. Stewardship.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY