International Journal of Engineering In Advanced Research Science and Technology ISSN: 2278-256

November 2016
VOLUME -2
ISSUE-4
Page:7932-37

RELIABLE AND EFFICIENT ON CHIP BUS PROTOCOL USING AHB

1. VATSAVAI LALITHASUVARNA, 2. V. RAMOJI, 3.B.V.RAMANA
1. PG Scholar,Dept of ECE, BonamVenkataChalamayya institute of technology & science, amalapuram
2. Assistant professor, Dept of ECE, BonamVenkataChalamayya institute of technology & science, amalapuram
3. Professor, Dept of ECE, BonamVenkataChalamayya institute of technology & science, amalapuram

ABSTRACT:
Designing an on-chip interconnection bus with more efficiency using modified AHB is main criteria in this project. A bridge between AHB master and slave with supportive application of MC is also proposed and the resultant efficiency in respect of area overhead and speed is provided. In the subject process, the design and implementation details of AMBA high-performance bus (AHB) master and slave with memory controller (MC) interface are discussed. Scalable encryption Algorithm is an enhanced algorithm for providing an efficient security for data. SEA provides area and power efficient implementation. It yields atmost secured bus interfacing data as enhancement for this project.
Keywords: Soc, O.C.P, AMBA, AHB, SEA

I. Introduction: As more and more IP cores are integrated into an SOC design, the communication flow between IP cores has increased drastically and the efficiency of the on-chip bus has become a dominant factor for the performance of a system. An SOC chip usually contains a large number of IP cores that communicate with each other through on-chip buses. As the VLSI process technology continuously advances, the frequency and the amount of the data communication between cores increase substantially. As a result, the ability of on-chip buses to deal with the large amount of data traffic becomes a dominant factor for the overall performance. The design of on-chip buses can be divided into two parts: **bus interface** and **bus architecture**. The bus interface involves a set of interface signals and their corresponding timing relationship, while the bus architecture refers to the internal components of buses and the interconnections among the IP cores. The widely accepted on-chip bus, AMBA AHB [1], defines a set of bus interface to facilitate basic (single) and burst read/write transactions. AHB also defines the internal bus architecture, which is mainly a shared bus composed of multiplexors. The multiplexer-based bus architecture works well for a design with a small number of IP cores. When the number of integrated IP cores increases, the communication between IP cores also increase and it becomes quite frequent that two or more master IPs would request data from different slaves at the same time. The shared bus architecture often cannot provide efficient communication since only one bus transaction can be supported at a time.

To solve this problem, two bus protocols have been proposed recently. One is the Advanced extensible Interface protocol (AXI) proposed by the ARM company. AXI defines five independent channels (write address, write data, write response, read address, and read data channels). Each channel involves a set of signals. AXI does not restrict the internal bus architecture and leaves it to designers. Thus designers are allowed to integrate two IP cores with AXI by either connecting the wires directly or invoking an in-house bus between them. The other bus interface protocol is proposed by a non-profitable organization, the Open Core Protocol – AHB [2]. AHB is an interface (or socket) aiming to standardize and thus simplify the system integration problems. It facilitates system integration by defining a set of concrete interface (I/O signals and the handshaking protocol) which is independent of the bus architecture. Based on this interface IP core designers can concentrate on designing the internal functionality of IP cores, bus designers can emphasize on the internal bus architecture, and system integrators can focus on the system issues such as the requirement of the bandwidth and the whole system architecture. In this way, system integration becomes much more efficient. Most of the bus functionalities defined in AXI and AHB are quite similar. The most conspicuous difference between them is that AXI divides the address channel into independent write address channel and read address channel such that read and write transactions can be processed simultaneously. However, the additional area of the separated address channels is the penalty. Some previous work has investigated on-chip buses from various aspects. The work presented in [3] and [4] develops high-level AMBA bus models with fast simulation speed and high timing accuracy. The authors in [7] propose an automatic approach to generate high-level bus models from a formal channel model of AHB. In both of the above work, the authors concentrate on fast and accurate simulation models at high-level but did not provide real hardware implementation details. In [9], the authors implement the AXI interface on shared bus architecture. Even though it costs less in area, the benefit of AXI in the communication efficiency may be limited by the shared-bus architecture. In this paper we propose a high-performance on-chip bus design with

International Journal of Engineering In Advanced Research Science and Technology ISSN: 2278-256

November 2016
VOLUME -2
ISSUE-4
Page:7932-37

AHB as the bus interface. We choose AHB because it is open to the public and AHB has provided some free tools to verify this protocol. Our proposed bus architecture features crossbar/partial -crossbar based interconnect and realizes most transactions defined in AHB, including 1) single transactions,2) burst transactions, 3) lock transactions, 4) pipelined transactions, and 5) out-of-order transactions. In addition, the proposed bus is flexible such that one can adjust the bus architecture according to the system requirement.

The remainder of this paper is organized as follows. In section 2The various advanced functionalities of on-chip buses are described. Section 3 describes functioning of the AHB Block Diagram. Section 4 gives the Implementation of AHB master/slave FSM. Section 5 is experimental results which show the efficiency on both simulation speed and data communication. Conclusions are then drawn in Section 6.

II. AHB bus functionalities

The various bus functionalities includes
1) Burst
2) Lock
3) Pipelined and
4) Out-of-Order transactions.

- Burst transactions

The burst transactions allow the grouping of multiple transactions that have a certain address relationship, and can be classified into multi-request burst and single-request burst according to how many times the addresses are issued. FIGURE 1 shows the two types of burst read transactions. The multi-request burst as defined in AHB is illustrated in FIGURE 1(a) where the address information must be issued for each command of a burst transaction (e.g., A11, A12, A13 and A14). This may cause some unnecessary overhead. In the more advanced bus architecture, the single-request burst transaction is supported. As shown in FIGURE 1(b), which is the burst type defined in AHB, the address information is issued only once for each burst transaction. In our proposed bus design we support both burst transactions such that IP cores with various burst types can use the proposed on-chip bus without changing their original burst behaviour.

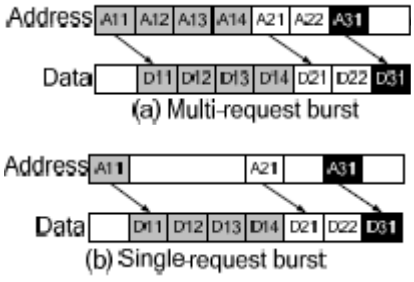


FIGURE 1. Burst transactions

- Lock transactions

Lock is a protection mechanism for masters that have low bus priorities. Without this mechanism the read/write transactions of masters with lower priority would be interrupted whenever a higher-priority master issues a request. Lock transactions prevent an arbiter from performing arbitration and assure that the low priority masters can complete its granted transaction without being interrupted.

- Pipelined transactions

Figure 2(a) and 2(b) show the difference between non-pipelined and pipelined read transactions. In FIGURE 2(a), for a non-pipelined transaction a read data must be returned after its corresponding address is issued plus a period of latency. For example, D21 is sent right after A21 is issued plus *t*. For a pipelined transaction as shown in FIGURE 2(b), this hard link is not required. Thus A21 can be issued right after A11 is issued without waiting for the return of data requested by A11 (i.e., D11-D14).

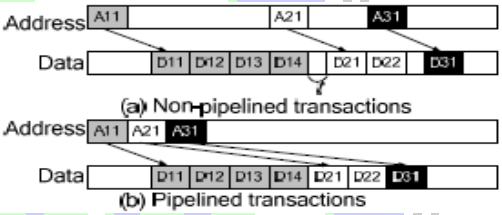


FIGURE 2. Pipeline transactions

- *Out-of-order transactions*

The out-of-order transactions allow the return order of responses to be different from the order of their requests. These transactions can significantly improve the communication efficiency of an SOC system containing IP cores with various access latencies as illustrated in FIGURE 3.In FIGURE 3(a) which does not allow out-of-order transactions, the corresponding responses of A21 and A31must be returned after the response of A11. With the support of out-of-order transactions as shown in FIGURE 3(b), the response with shorter access latency (D21, D22 and D31) can be returned before those with longer latency (D11-D14) and thus the transactions can be completed in much less cycles.

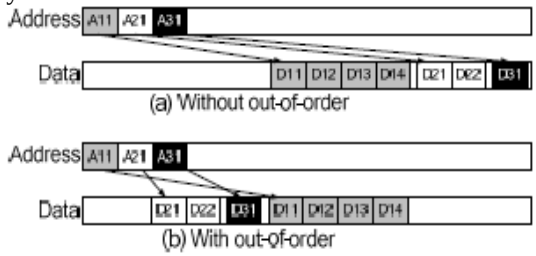


FIGURE 3. Out-of-order transactions

International Journal of Engineering In Advanced Research Science and Technology ISSN: 2278-256

November 2016
VOLUME -2
ISSUE-4
Page:7932-37

III. Functional blocks of the AHB bus

The architecture of the proposed on-chip bus is illustrated in FIGURE 4, where we show an example with four masters and four slaves. A crossbar architecture is employed such that more than one master can communicate with more than one slave simultaneously. If not all masters require the accessing paths to all slaves, partial crossbar architecture is also allowed. The main blocks of the proposed bus architecture are described next.

- Arbiter

In traditional shared bus architecture, resource contention happens whenever more than one master requests the bus at the same time. For a crossbar or partial crossbar architecture, resource contention occurs when more than one master is to access the same slave simultaneously. In the proposed design each slave IP is associated with an arbiter that determines which master can access the slave. The arbiter sends grant signal to the requested master.

- *Decoder*

Since more than one slave exists in the system, the decoder decodes the address and decides which slave return response to the target master. In addition, the proposed decoder also checks whether the transaction address is illegal or nonexistent and responses with an error message if necessary.

- *Multiplexer*

A multiplexer is used to solve the problem of resource contention when more than one slave returns the responses to the same master. It selects the corresponding slave/master according to the given selection lines. There are multiplexers one is address and control mux and the two are for data read and write.

- *Master/Slave*

Master and Slave are the entities consisting of different Address and Data Locations. Only the master can send the commands and is the controlling entity. The Slave responds to commands presented to it, either by accepting data from the master, or representing data to the master.

- *FSM-M & FSM-S*

FSM-M acts as a master and generates the AHB signals of a master, while FSM-S acts as a slave and generates those of a slave.

IV. Implementation of AHB

The request issued by system is given to slave by MCmd signal. Similarly, in Write operation, the input address and data provided by the system will be given to slave through the signal MAddr and MData and when those information's are accepted, slave will give SCmd Accept signal which ensures that the system can issue next request. During Read operation, system issues the request

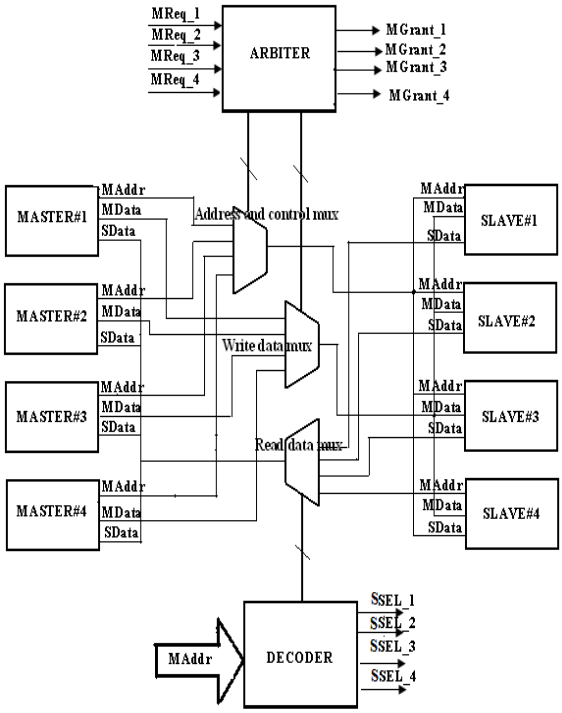


FIGURE 4.AHB Block Diagram

and address to slave which will set SResp and fetch the corresponding data that is given to output through Sdata.The design of the Open Core Protocol starts with the initial study based on which the development of FSM(Finite State Machine) for the various supporting operation after which the development of VHDL for the FSM.

FSM for AHB master: The Finite State Machine (FSM) is developed for the simple write and read operation of AHB Master. The simple write and read operation indicates that the control goes to IDLE state after every operation. Basically, the operation in the AHB will be held in two phases.

- Request Phase
- Response Phase

Initially the control will be in IDLE state (Control = "000") at which all the outputs such as MCmd, MAddr and MData are set to "don't care". The system will issue the request to the master such write request which leads to the WRITE state (Control = "001"). In this state, the address and the data will be given to the slave that is to be written and hence the process will get over only when the SCmd Accept is asserted to high. If SCmd Accept is not set, this represents that the write operation still in process and the control will be in the WRITE state itself. Once the

International Journal of Engineering In Advanced Research Science and Technology ISSN: 2278-256

November 2016
VOLUME -2
ISSUE-4
Page:7932-37

write operation is over the control will go to the IDLE state and then it will check for the next request.

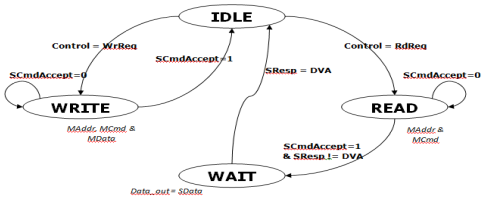


figure 5: fsm forAHB master

When the read request is made, the control will go to the READ state (Control = "010") and the address is send to the slave which in turn gives the SCmdAccept signal that ends the request phase. Once the SCmdAccept is set and SResp is not Data Valid (DVA), the control will go the WAIT state and will be waiting for the SResp signal. When the read operation is over which represents that the SResp is set to DVA and the data for the corresponding address is taken. Hence the SResp signal ends the response phase and the control will go the IDLE state, then checks for the next request.

FSM for AHB slave: The FSM for the AHB Slave which has the simple write and read operation is developed and is shown in the Figure 6

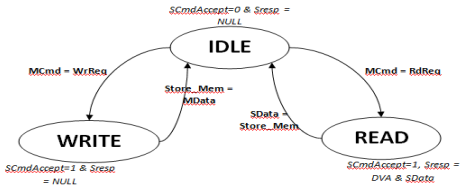


FIGURE 6. fsm for AHB slave

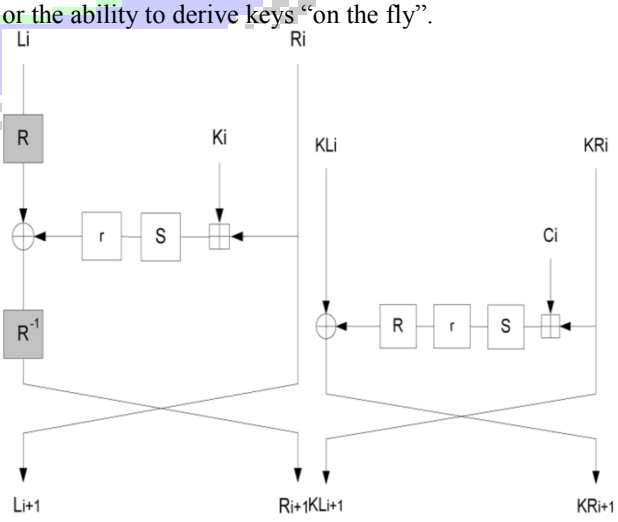
The slave will be set to the respective state based on the MCmd issued by the master and the output of this slave is that the SCmdAccept and SResp. Initially control will be in the IDLE state and when the master issues the command as write request, and then the control will go the WRITE state in which the data will be written to the corresponding memory address location which is sent by the masters. Once the write operation is finished, the SCmdAccept signal is set to high and is given to the master. When MCmd is given as read request, then the control will move to the READ state in which the data will read from the particular memory address location that is given by the master. Hence the SCmdAccept is set to high and the SResp is set to the DVA which represents that the read operation over and control goes to the IDLE state.

Depending on whether a transaction is a read or a write operation, the request and response processes are different. So for burst, burst count is specified that is burstsize both for read and write.

Scalable Encryption Algorithm (SEA):

Most present symmetric encryption algorithms result from a trade-off between implementation cost and resulting performances. In addition, they generally aim to be implemented efficiently on a large variety of platforms. In this paper, we take an opposite approach and consider a context where we have very limited processing resources and throughput requirements. For this purpose, we propose low-cost encryption routines (i.e. with small code size and memory) targeted for processors with a limited instruction set (i.e. AND, OR, XOR gates, word rotation and modular addition). The proposed design is parametric in the text, key and processor size, allows efficient combination of encryption/decryption, "on-the-fly" key derivation and its security against a number of recent cryptanalytic techniques is discussed. Target applications for such routines include any context requiring low-cost encryption and/or authentication.

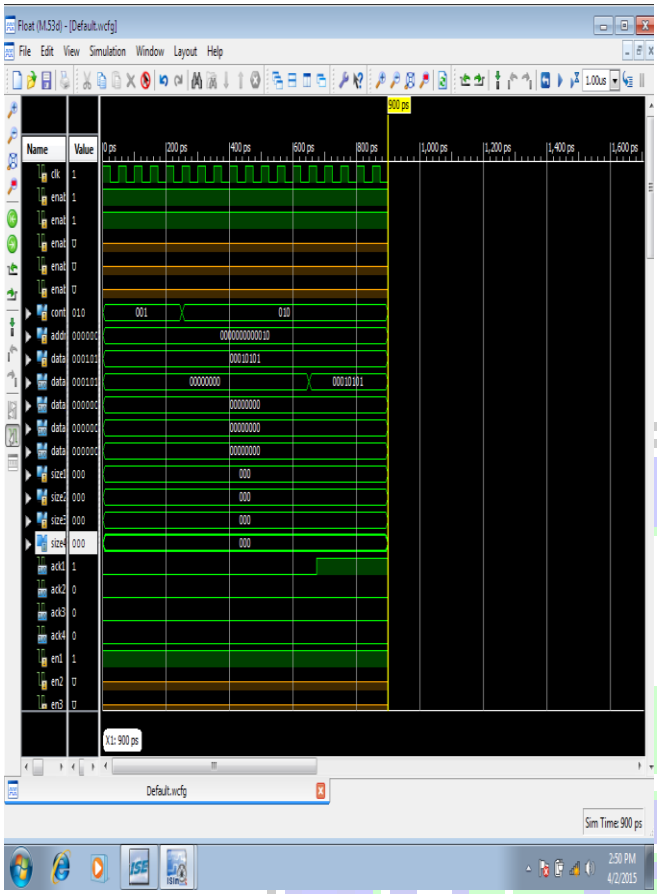
In this paper, we consequently consider a general context where we have very limited processing resources (e.g. a small processor) and throughput requirements. It yields design criteria such as: low memory requirements, small code size, limited instruction set. In addition, we propose the flexibility as another unusual design principle. SEAn,b is parametric in the text, key and processor size. Such an approach was motivated by the fact that many algorithms behave differently on different platforms (e.g.8-bit or 32-bit processors). In opposition, SEAn,b allows to obtain a small encryption routine targeted to any given processor, the security of the cipher being adapted in function of its key size. Beyond these general guidelines, alternative features were wanted, including the efficient combination of encryption and decryption or the ability to derive keys "on the fly".



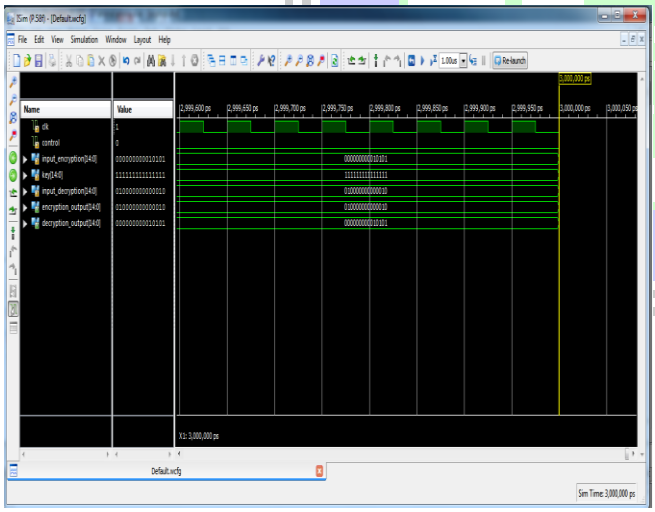
International Journal of Engineering In Advanced Research Science and Technology ISSN: 2278-256

November 2016
VOLUME -2
ISSUE-4
Page:7932-37

V. RESULTS



SEA RESULTS



VI. Conclusion

An AHB-based MC is designed ,realized and tested in the present concept. The implementation was carried out using Vhdl HDL for SOC solutions. The design has taken care of balance between area overhead and speed.. A bridge between AHB master and slave with supportive application of MC along with SEA is also proposed and the resultant efficiency in respect of area overhead, speed and power optimization is provided in this concept.

REFERENCES

. [1] Advanced Microcontroller Bus Architecture (AMBA) Specification Rev 2.0 & 3.0, ttp://www.arm.com.

[2] Open Core Protocol (OCP) Specification, http://www.ocpip.org/home.

[3] Y.-T. Kim, T. Kim, Y. Kim, C. Shin, E.-Y. Chung, K.-M. Choi, J.-T. Kong, S.-K. Eo, "Fast and Accurate Transaction Level Modeling of an Extended AMBA2.0 Bus Architecture,"
Design, Automation, and Test in Europe, pages 138-139, 2005.

[4] James Aldis, "Use of OCP in OMAP 2420", http://www.ocpip.org/, 2005.

[5] G. Schirner and R. Domer, "Quantitative Analysis of Transaction Level Models for the AMBA Bus," *Design, Automation, and Test in Europe*, 6 pages, 2006.

[6] David C.-W. Chang, I.-T. Liao, J.-K. Lee, W.-F. Chen, S.-Y. Tseng and C.-W. Jen, "PAC DSP Core and Application Processors," *International Conference on Multimedia and*
Expo, pages 289-292, 2006.

[7] Partha Pratim Pande, Cristian Grecu, Michael Jones, Andr´e vanov, and Resve Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures", IEEE Trans. Computers, vol. 54, no. 8, pp. 1025–1040, Aug. 2005.

[8] IBM Corporation, "Prioritization of Out-of-Order Data Transfers on Shared Data Bus," *US Patent No. 7,392,353*,
2008.

[9] N.Y.-C. Chang, Y.-Z. Liao and T.-S. Chang, "Analysis of Shared-link AXI," *IET Computers & Digital Techniques*, Volume 3, Issue 4, pages 373-383, 2009.

[10] C.-K. Lo and R.-S. Tsay, "Automatic Generation of Cycle Accurate and Cycle Count Accurate Transaction Level Bus Models from a Formal Model," *Asia and South Pacific Design Automation Conference*, pages 558-563, 2009.