2024 2nd International Conference on Artificial Intelligence and Machine Learning Applications Theme: Healthcare and Internet of Things (AIMLA) | 979-8-3503-4922-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/AIMLA59606.2024.10531611

HIGH PERFORMANCE AND LOW POWER AXI-LITE 4 MASTER CONTROLLER USING SYSTEM VERILOG

Muhammed Salman MS, Ramesh Bhakthavatchalu
Department of Electronics and Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri,India
salmanms929@gmail.com, chidanandamrita@am.amrita.edu

***Abstract*—Digital design and System-on-Chip (SoC) development practices frequently employ the Advanced eXtensible Interface (AXI) protocol. It acts as an interconnect protocol to help different semiconductor chip components communicate with one another. The AXI protocol, created by ARM, provides master and slave devices with a reliable and effective way to send data, control, and configure settings in a coherent and coordinated way. AXI provides high frequency and high performance designs. It is an on-chip communication protocol, Large bandwidth and frequency low-delay designs are suitable for it.The paper demonstrates the development of a high-performance and low power design utilizing the AXI Lite protocol. This design incorporates a state machine and is synthesized in Xilinx Vivado and resulting in an IP master module capable of functioning with both AXI4 and AXI4 slave interfaces.**

***Index Terms*—AXI,AXI-LITE,AMBA,Verilog,VLSI**

I. INTRODUCTION

System on-Chip (SoC) architectures dominate the VLSI technology landscape in the modern era. SoC components, including processors, GPUs, memory units, controllers, and a variety of intellectual property (IP) cores, are inextricably linked to the field of computer technology. Establishing smooth communication pathways between these many IP cores for effective data transactions is a key problem in SoC architecture. The AMBA framework plays a key role in on-chip interconnect solutions to address this difficulty. One particularly strong competitor is the Advanced eXtensible Interface (AXI), which is part of the AMBAprotocol family. Building complex embedded microcontrollers on top of the AMBA requirements promotes modular system design while assuring technical adaptability[1].

AXI stands out among the several protocols that make up the broad AMBA family, such as CHI, ACE, AHB, ASB and APB. An outstanding combination of high bandwidth and performance is provided by AXI[1]. This Advanced Extensible Interface, a core member of the Advanced Bus Architecture (AMBA) protocol family, aims to support high-performance, high-frequency system design paradigm[2].

The main goal of the research on creating a low-power, high-performance AXI-Lite 4 master controller with SystemVerilog was probably to create and build a controller that could effectively interact with peripherals that are compliant with AXI Lite 4and SystemVerilog is becoming well-known in the semiconductor industry as a language for describing and verifying hardware. Modern semiconductor technologies prefer to use SystemVerilog for complicated digital system design since hardware coding may be done with ease and efficiency.

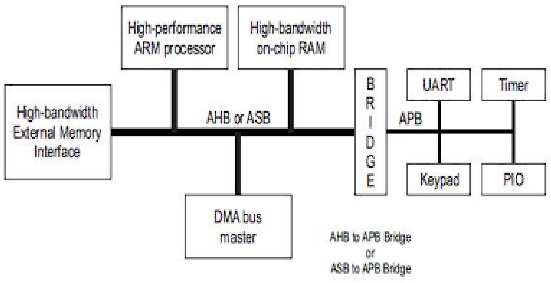

High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth External Memory Interface
AHB or ASB
B R I D G E
UART
Timer
APB
Keypad
PIO
DMA bus master
AHB to APB Bridge or ASB to APB Bridge

Fig. 1. AMBA based Microcontroller

• II. OVERVIEW OF THE AMBA PROTOCOL

A. AMBA Protocol

The cpu on–chip memory, along with other Direct Memory Access (DMA) devices, resides on the high-performance system bus (AMBA ASB or AHB) of the microcontroller, which is built on the AMBA architecture and can handle external memory bandwidth[1]. The AMBA bus protocol primarily facilitates data transfer between masters and slaves[2], offering a high-bandwidth interface for most transfers among the involved components. If the system includes both high- and low-performance devices, the AXI bus protocol is utilized for AHB devices with high performance, and APB devices with low performance[1].

B. AXI Protocol Feature

AXI supports system designs with extraordinarily high frequencies and great performance . It is indispensable for low-latency design requirements and finds its strength in situations requiring significant bandwidth[1]. Notably, AXI sustains high-frequency operation without the need for complex bridges. AXI distinguishes itself by allowing the simultaneous connection of many masters with multiple slaves—a feat unachievable through the AHB protocol—an attribute

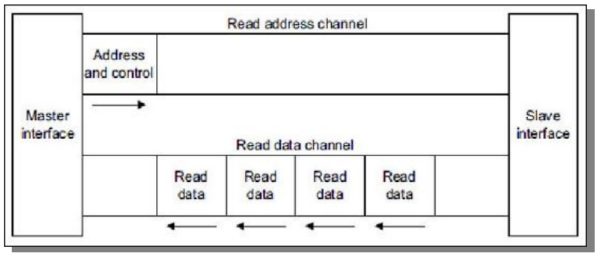


Fig. 2. AXI Read channel architecture

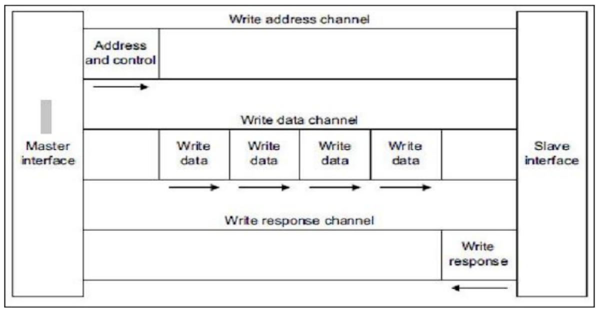


Fig. 3. AXI Write channel architecture

particularly well-suited for memory controllers characterized by increased initial access latency.AXI4-Lite operates with distinct features: transactions employ burst length 1, utilizing the entire data bus width (either 32-bit or 64-bit)[2]. All accesses are non-modifiable, non-bufferable, lacking support for exclusive accesses.

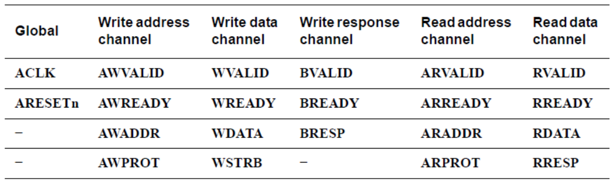
III. ARCHITECTURE OF AXI PROTOCOL

A. AXI4-Lite Read and Write Transactions

Five distinct channels make up the AXI channel architecture, which is utilized to carry out the two primary transactions (write and read operations)[1]. Valid and ready handshaking signals are present on all five channels. These signals represent address control information that is transmitted by the master and recognized by the slave as soon as it becomes available. In a write transaction, the write address, write data, and write response channels are employed. The read transaction is finished by using data channels with read addresses. AXI runs quickly because of the independent channel.The AXI channel's architecture was shown in Figure 2. Below is a list of the five channels that comprise the distinct, independent transaction pathways of the AXI protocol[1]:
• Write Address Channel: Address control data, which describes the kind of data that will be carried, is transmitted over this channel.

• Write Data Channel: This channel sends information to the slave where the write operation is to be done, along with the address.

• Write Response Channel: Once the write address and write data procedures are completed successfully, the slave replies or acknowledges the master through the channel.



Global	Write address channel	Write data channel	Write response channel	Read address channel	Read data channel
ACLK	AWVALID	WVALID	BVALID	ARVALID	RVALID
ARESETn	AWREADY	WREADY	BREADY	ARREADY	RREADY
–	AWADDR	WDATA	BRESP	ARADDR	RDATA
–	AWPROT	WSTRB	–	ARPROT	RRESP

Fig. 4. AXI4-Lite Master Signal

• Read Address Channel: Carrying both address and control information, this channel services the Read path, elucidating the fundamental attributes of the data set for transfer.

• Read Data Channel: The slave replies or acknowledges the master after writing the address and data successfully through the channel .

B. AXI4-Lite Signals

• awvalid – Write address valid, valid signifies the availability of control information and a valid write address.
• awaddr –Write address, The transaction's address is provided via the write address bus.
• wvalid –write valid,this signal lets you know that there are accessible strobes and valid write data.
• wdata –write data, real data to be written.
• wstrb – Write strobes, Which memory byte lanes need to be updated is indicated by this signal.
• bready – Response ready, This signal means that the response data can be accepted by the master information.
• arvalid – Read address valid,Until the address acknowledgement signal, ready, is high, this signal, when high, indicates that the read address and control information is legitimate.
• araddr – Read address,The address of a read transaction is provided via the read address bus.
• rready – Read ready,This signal indicates that the slave may accept the write data.
• awready – Write address ready, This signal lets you know that the slave is prepared to receive control signals and an address.
• wready – Write ready, This signal lets you know that the write data can be accepted by the slave.
• bvalid – Write response valid,This signal lets you know when a legitimate write response is ready.
• bresp – Write response, This signal provides information about the write transfer's status.
• arready – Read address ready,This signal lets you know that the slave is prepared to receive control signals and an address.
• rvalid – Read valid, This signal means that the read transfer can finish because the necessary read data is available.
• rdata – Read data, the actual data that is given back after a read request.
• rresp – Read response, The read transfer's status is indicated by this signal.

C. State Machine

The implementation of the AXI Master involves utilizing Sun Burst, Registered FSM Outputs (3-always block coding

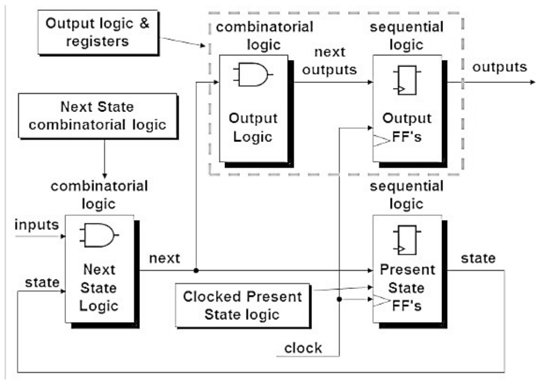
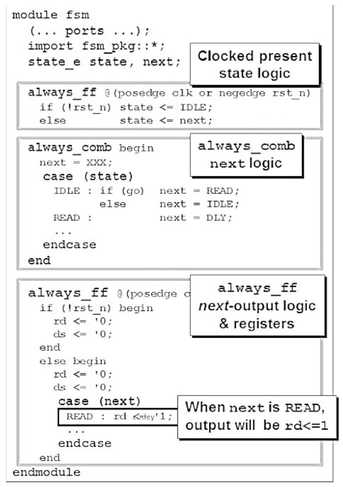


Fig. 5. Synthesis diagram for 3-always block coding style



```
module fsm
  (... ports ...);
  import fsm_pkg::*;
  state_e state, next;
  always_ff @(posedge clk or negedge rst_n)
    if (!rst_n) state <= IDLE;
    else        state <= next;
  always_comb begin
    next = XXX;
     case (state)
       IDLE : if (go)   next = READ;
              else      next = IDLE;
       READ :           next = DLY;
       ...
     endcase
  end
  always_ff @(posedge c
    if (!rst_n) begin
      rd <= '0;
      ds <= '0;
    end
    else begin
      rd <= '0;
      ds <= '0;
       case (next)
        READ : rd <=dey'1;
        ...
       endcase
    end
endmodule
```

Clocked present state logic
always_comb next logic
always_ff next-output logic & registers
When next is READ, output will be rd<=1

Fig. 6. Block diagram for 3-always block coding style

style)[4]. This approach ensures glitch-free outputs within the Finite State Machine (FSM) design. Registering the outputs of the FSM not only guarantees their stability but also frequently enhances synthesis outcomes. By standardizing both output and input delay constraints of the synthesized modules, this method significantly improves the efficiency and reliability of the system. The use of Sun Burst Registered FSM Outputs offers a mechanism to manage the outputs systematically, reducing potential glitches and inconsistencies that may arise[4]. This systematic registration method establishes a stable framework for the outputs, contributing to smoother operations within the FSM design. Overall, this technique not only ensures the stability and reliability of outputs but also contributes to the optimization and refinement of the synthesis process, ultimately enhancing the overall performance and functionality of the AXI Master implementation.

In this coding style fig.6, the trick is to identify the output assignments as the "next" output assignments and not the "output assignments" for the current state. The last always-ff procedure will test the "next state", not the "current". Because the outputs are calculated from next state logic, it can add extra logic to the synthesis as the next state has already been

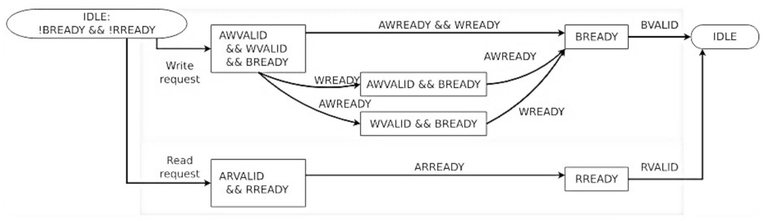


Fig. 7. STATE DIAGRAM

calculated from input conditions[4]. This means that there is a single block of combinator logic that calculates the next state, and that feeds another block of combinatoric logic to compute the next outputs. This can create bigger and slower combinatorical logic. In the 1 – always block coding style, the inputs and state variables calculate the next state and the next outputs at the same time, which reduces the size and delays through the combinatorical next output logic. This inefficiency will be dealt with in the next section[4].

IV. IMPLEMENTATION DETAILS

A. DESIGN

Utilizing sunburst design techniques[4] and employing 3 always blocking coding, resource synthesis can be significantly reduced. This reduction in resources consequently leads to a decrease in power consumption Hardware efficiency is increased when state machine design is combined with SystemVerilog. Since they provide precise state definitions, enumerated types for state representation guarantee synthesizability. This method makes design understanding and verification easier. Tools for synthesis identify and enhance state transitions, resulting in hardware implementations that are effective.In existing implementing AXI4-Lite using Verilog [8] which has more resource utilization and low frequency which can solved by my design using system verilog Previously, AXI4-Lite was implemented using Verilog [8], which had low frequency and high resource consumption.

B. WRITE RESPONSE

The master initiates communication by placing an address to the write address channel (awaddr) and data to the write data channel (wdata). Simultaneously, it activates awvalid and wvalid signals to validate the address and data on their respective channels. Alongside this, the master activates the bready signal, signifying its readiness to receive a response.

Parallel to this, the slave indicates its readiness by asserting signals on the write address and write data channels, respectively, that read "awready" and "wready." A handshake is initiated and the exchange can proceed as a result of both parties asserting that the signals on the write address and write data channels are genuine and ready. The related valid and ready signals can be disabled after these handshakes take place, signifying successful transmission and the slave's acquisition of the write address and data

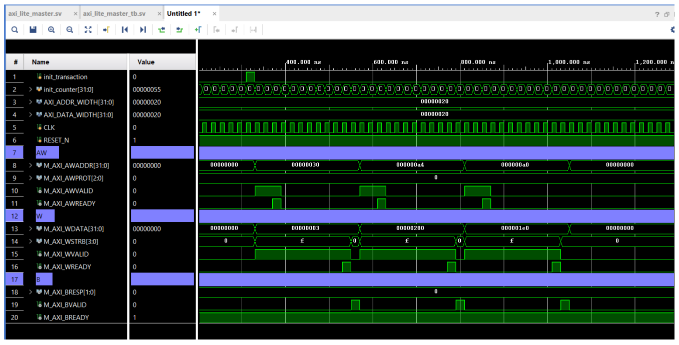


Fig. 8. WRITING TO TO SLAVE

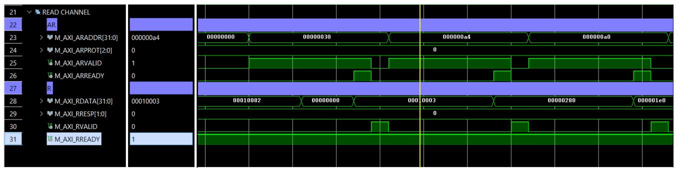


Fig. 9. READING FROM SLAVE

Following the completion of this exchange, the slave activates the bvalid signal to indicate that a valid answer is now accessible on the write response channel (bresp). When all operations are finished and the ready and valid signals on the write response channel become active, the transaction is considered complete at the next rising clock edge.

C. *READ RESPONSE*

The master initiates a read operation by placing an address on the read address channel while activating the arvalid signal, confirming the validity of the address. Simultaneously, the master asserts the rready signal, indicating its readiness to receive data from the slave.

Upon receiving this request, the slave acknowledges readiness by asserting the arready signal, signifying its preparedness to receive the address on the bus. When both arvalid and arready signals are active, a handshake occurs at the subsequent rising clock edge, facilitating the exchange. Following this handshake, the master deactivates arvalid, and the slave deactivates arready, indicating the completion of address transmission from master to slave. At this stage, having received the requested address, the slave proceeds to place the corresponding data on the read data channel (rdata) while asserting the rvalid signal, confirming the validity of the data on the channel. Additionally, the slave might provide a response code on the rresp signal to convey supplementary information.

With both rready and rvalid signals activated, the subsequent rising clock edge finalizes the transaction. Once this exchange is completed, both rready and rvalid signals can be deactivated, marking the end of the read operation.

D. *WRITING AND READING*

In the Vivado environment, Fig. 8 and Fig. 9 illustrate the construction of an AXI master through the use of SystemVer-

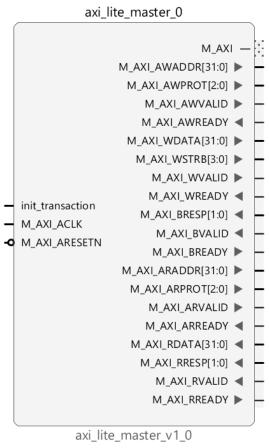


Fig. 10. IP MODULE

ilog . Presently integrating the VDMA (Video Direct Memory Access) IP from Vivado[11], where the VDMA serves as a slave element to the AXI Lite. Within the workflow, begin by initializing addresses, and subsequently, write values to their respective addresses via the channel. Simultaneously, the same data is read from the slave, which in this case is the VDMA IP core[11].

The process involves setting up the communication between the AXI master and the VDMA IP, designating the VDMA as a slave component to the AXI Lite. Upon initializing specific addresses, utilize this communication channel to transmit data. This transmission occurs through the AXI master, where values corresponding to their allocated addresses are written.

Concurrently, the VDMA IP, operating as a slave, receives the transmitted data at the specified addresses.. As values are written by the master, the slave VDMA IP captures this information, enabling subsequent read operations to retrieve the same data.

E. *SYNTHESIS REPORTS*

Resource	Utilization	Available
SLICE LUTS	16	101400
SLICE REGISTERS	9	202800
IO	119	400
BUFGCTRL	1	32

TABLE I
TABLE SHOWING UTILIZATION KINTEX 7

The table displays resource utilization in the AXI Lite protocol, achieved through synthesis by Vivado Xilinx on the Kintex-7 board. Notably, the utilization comprises 16 LUTs, 9 slice registers, 199 I/O ports, and 32 BUFCTRL elements. This information provides insights into the efficient allocation of resources.
In The figure 12 Vivado power report provides crucial insights, indicating a total chip power of 4.264W in Power. This figure underscores an efficient and limited power consumption scenario. The comprehensive analysis conducted by Vivado reflects the platform's commitment to optimizing

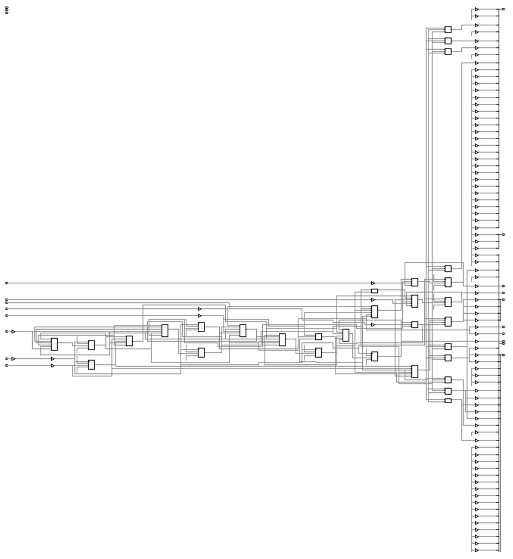
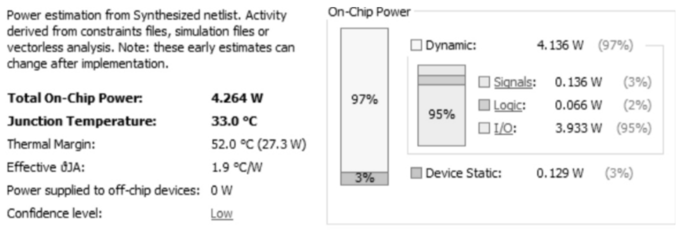


Fig. 11. RTL



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**4.264 W**
Junction Temperature:	**33.0 °C**
Thermal Margin:	52.0 °C (27.3 W)
Effective ϑJA:	1.9 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

On-Chip Power

Dynamic:	4.136 W	(97%)
Signals:	0.136 W	(3%)
Logic:	0.066 W	(2%)
I/O:	3.933 W	(95%)
Device Static:	0.129 W	(3%)

Fig. 12. POWER REPORT

power usage, ensuring a balance between performance and energy efficiency.

According to the timing report, the max data path delay is reported at 3.33 ns, while the min data path delay is recorded at 0.268 ns. These values offer insights into the performance characteristics of the system. The controller operates at a frequency of 300 MHz, which increases its processing capability and leads to the speed enhancement. Additionally, transactions have a low latency and are executed quickly because 3.3 ns delay.

V. CONCLUSION

In this work, have effectively integrated an AXI-lite IP module using SystemVerilog in Vivado. Through meticulous verification procedures, have validated seamless communication between the master and the designated slave. verification encompasses both read and write responses, ensuring high-performance functionality while efficiently utilizing system resource As it operates at a high frequency of 300MHz, the master module has been integrated with the VDMA core IP in Vivado to guarantee enhanced accuracy and proper functionality. The System on Chip (SoC) advances daily, transitioning from AXI Lite to CHI, an advanced Arm protocol, in current device development. Its evolution marks progress, shaping the future of SoC technology, enriching capabilities beyond conventional constructs, fueling innovation in contemporary devices

VI. FUTURE SCOPE

In the semiconductor industry, SystemVerilog is becoming a well-known language for hardware description and verification. It makes hardware coding for intricate digital systems straightforward and efficient. Because of the scalability and flexibility of the AXI protocol, designers can build SoCs with different degrees of complexity. Although minimal bursting and data transfers are supported by AXI4-Lite, AXI4 Stream is required for applications such as video streaming because of the large number of frames that need to be read and processed. All things considered, AXI is a fundamental component of contemporary SoC architecture, promoting effective communication, maximizing performance, and opening the door for the creation of cutting-edge semiconductor solutions for a wide range of applications.

REFERENCES

[1] ARM, AMBA AXI protocol specifications, Available at, http://www.arm.com,2023
[2] AMBA AXI Protocol Version: 2.0 Specification, ARM
[3] R. Bhaktavatchalu, B.S. Rekha, G.A. Divya, and V.U.S Jyothi, “Design of AXI bus interface modules on FPGA”, Amrita Vishwa Vidyapeetham Amritapuri,India .In Advanced Communication Control and Computing Technologies (ICACCCT), International Conference on IEEE, 2016, pp. 141-146 Ltd, pp. 1-1.
[4] Sun-burst The Fundamentals of Efficient Synthesizable Finite State Machine Design using NC-Verilog and BuildGates by Clifford E. Cumming
[5] Alan P. Su, JifT Kuo, Kuen·Jong Lee, Ing·Jer Huang, Guo·An Jian” A Multi·core Software/Hardware Co·debug Platform with ARM CoreSightTM, On·chip Test Architecture and AXIIAHB Bus Monitor.
[6] M. Gupta and A. K. Nagawat, ”Design and implementation of high performance advanced extensible interface(AXI) based DDR3 memory controller,” 2016 International Conference on Communication and Signal Processing (ICCSP), 2016, pp. 1175-1179, doi:
[7] Kommirisetti Bheema Raju and Bala Krishna Konda, “Design and Verification of AMBA APB Protocol”, Int. Journal of Engineering Research and Application, Volume 7, Issue 1, pp.87-90, 2017
[8] Design of AMBA AXI4-Lite for Effective Read/Write Transactions with a Customized Memory A. Sainath Chaithanya1, Sameera Sulthana2, B. Yamuna2 and Ch Haritha2
[9] Low latency max log map based turbo decoder Narayanan, Aswathy, Murugan, Senthil, Bhakthavatchalu Ramesh Electronics and Communication Engineering Amrita Vishwa Vidyapeetham Amritapuri,India, ”Low latency max log map based turbo decoder”, Proceedings of the 2019 IEEE International Conference on Communication and Signal Processing, ICCSP 2019
[10] N.Tidala, “High-Performance Network on Chip using AXI4 protocol interface on an FPGA”, In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2018, pp. 1647-1651
[11] M. Makni, M. Baklouti, S. Niar, and M. Abid, “Performance Exploration of AMBA AXI4 Bus Protocols for Wireless Sensor Networks”, In Computer Systems and Applications (AICCSA), 2017 IEEE/ACS 14th International Conference on IEEE, 2017, pp. 1163-1169
[12] Xilinx. Zynq-7000 All Programmable SOC. http://www.xilinx.com. 2015
[13] Xilinx. Zynqultrascale+ mpsoc, product selection guide. http://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascalempsoc. html, 2015.
[14] Veena Abraham, Soumen Basak, Sabi S, “Design of AXI4 Protocol Checker for SoC Integration,” International Journal Of Emerging Science and Engineering (IJESE), ISSN: 2319-6378, vol.1, Issue 8, June 2013.

[15] S. Sharma, and S.M. Sakthivel, “Design and Verification of AMBA AXI3 Protocol”, In VLSI Design: Circuits, Systems, and Applications, Springer, Singapore, 2016, pp. 247-259

[16] Z. Panjkov, J. Haas, M. Aigner, H. Rosmanith, T. Liu, “Poppenreiter R. Hagelauer, “OCP2XI Bridge: An OCP to AXI Protocol Bridge”, In International Symposium on Applied Reconfigurable Computing , 2016, pp. 179-190

[17] Design and Implementation of MIPI I3C master controller SubSystems Yadhu Krishnan S, Ramesh Bhakthavatchalu Department of ECE Amrita Vishwa Vidyapeetham Amritapuri, India2023 3rd International Conference on Intelligent Technologies (CONIT) Karnataka, India. June 23-25, 2023

[18] Block Level SoC Verification Using Systemverilog Yadu Krishnan K, Ramesh Bhakthavatchalu Dept. of Electronics and Communication Engineering Amrita Vishwa Vidyapeetham Amritapuri,India Third International Conference on Electronics Communication and Aerospace Technology [ICECA 2019]

[19] UVM based testbench architecture for logic sub-system verification,T M Pavithran, Ramesh Bhakthavatchalu,Dept. of Electronics and Communication Engineering Amrita Vishwa Vidyapeetham Amritapuri,India,f 2017 IEEE International Conference on Technological Advancements in Power and Energy: Exploring Energy Solutions for an Intelligent Power Grid, TAP Energy 2017