Design of an Efficient AXI-4 Protocol for High Speed SOC Applications on FPGA Platform

Archana H. R[1], C. R. Byrareddy[2], and Narendra C. P[2]
[1]*Department of ECE, BMS College of Engineering, Bangalore, India*
[2]*Department of ECE, Bangalore Institute of Technology, Bangalore, India*
archanahr.ece@bmsce.ac.in

***Abstract*—The system-on-chip(SoC) design process encounters various challenges of communication between one to another module. Thus, the Bus interconnection plays a significant role in improving the system performance on a single chip. The traditional bus interconnections cease its applicability to meet the requirements of future generation SoC. This paper proposes an efficient design of the AXI-4 protocol to achieve high-speed data transfer in the SoC application. The proposed AXI-4 Interface protocol includes the Master and Slave module, which are designed using state flow and state diagrams. Both the Master and Slave module operations support burst based transactions and perform the five different channel transactions that include "write address," "write data," "write a response," "read address," "read data" along with "read response." The simulation results of the AXI-4 interface and its FPGA realization on Artix-7 illustrate lower resource utilization, and the performance benchmarking between proposed AXI-4 with traditional AHB and Wishbone bus modules illustrates an average minimization in area and increase in frequency by 40% and 41% respectively.**

***Index Terms*—AHB; AMBA; ASB; AXI-4; Burst; FPGA; Master; Slave; SOC, Transaction.**

I. INTRODUCTION

In recent years, the development of SoC and multicore processors are gaining popularity because of their advantages. The suitably designed bus interconnection between the devices plays a vital role in improvising the system performance. The high throughput and low latency bus architectures are an essential part of the system communication and interconnection. Many of the bus interconnection system designs include Hyper transport, PCI, and Quick Path Interconnect to provide high performance in SoC, but it still faces congestion issues in system performance [1].

In VLSI Field, the SoC is the fastest growing design platform to achieve low cost, low power, and high-speed constraints on a single chip rather than conventional board-level design. In recent years, compared to ASIC design, FPGA (Field-programmable gate array) has been widely used due to its advantageous capacity of reconfigurability and its easiness to realize and modify the design at the device level. The programmable SoC architecture FPGA is the Zynq-7000 series, which supports the ARM- Cortex-A9 processing system and 28nm Xilinx programmable logic on a single chip. The Zynq SoC supports secured data transmission with high performance and low latency embedded systems along with shared memory access [2][3].

The performance and functionality aspects of the SoC in the context of associated verification is a challenging task within the constraints of the Master and Slaves configuration along with dynamic topology, a total number of transaction types, and different interface protocols. The verification challenges are corrected by using new tools and technologies with functional correctness, verification completeness, protocol and conversion compliance, stress verification, security, and power management [4]. The SoC bus interconnections are designed based on the topologies and protocol deployments in buses. The bus topologies include single-level, multi-level shared structure, and Multi-bus interconnection structure. The AMBA protocol used as a bus module in the AXI (Advanced Extensible Interface) provides communications to high-performance devices, and APB (Advanced Peripheral Bus) provides transmission among low-speed devices and peripherals [5].

The Bridge module is an essential component found in SoC to establish communication between protocols [6]. For multiple Master and multiple Slaves, the communication interface plays an essential role in improving system performance. The serial communications can transfer any data, although it could not meet the complex system requirements. The parallel systems transfer data from a particular source to a destination using the AXI interface protocol [7].

The proposed AXI-4 Interface protocol offers high-speed interconnection to the SoC systems. The proposed design follows the AMBA AXI Protocol specifications [8], which supports bus-based transactions suitable for low latency and high bandwidth designs to improve the system performance. The proposed design overcomes the drawbacks of the previous bus-based architectures like wishbone and AHB interfaces. The AXI-4 Design uses five-channel transactions, which includes write address, write data, write a response, read address, and read data with read response. The AXI-4 Master and Slave interface modules are designed using state flow and state diagrams. The data width is considered for 8-bits wide. Further, these interface protocols are used in image processing applications.

This paper is organized in six sections. After the introduction, presented in Section I, Section II discusses the existing works of interface protocols, AXI, and its related technologies and research gaps findings. The methodology adopted for the proposed work is discussed in section III. Section IV explains the proposed system with detailed descriptions. The results and analysis of the work are elaborated in section V. Finally, section VI concludes the overall system with improvements and future work.

II. Related Work

The review of the existing work on AXI protocol with different functional architectures and for different applications is described below.

Sarojini et al. [9] presented the high optimized throughput Memory Interface design, using AXI protocol as a Hardware approach with separate read/write channels. The data is passed through FIFO and accessed by AXI Master, while AXI Core interconnect provides the interface between Master and Slave. The slave DDR memory controller receives the data and connects to the external world. The design is speedup in terms of Mbps based on the clock frequency. The design limits the Input-Output speed, which depends upon the FPGA Hardware capabilities. Tidala [10] describes the FPGA (Hardware approach) based on high-speed on-chip communication using the AXI-4 protocol. The design overcomes the many challenges concerning quality of service, resolving the complexity issues in Network routing interface, hence providing better data transmission. The Network communication happens between Programmable Logic device and Memory via AXI interconnect on FPGA. For each burst data transaction, the Bandwidth is observed and tabulated. Erfan et al. [11] reported that the AXI-4 is based on interconnect as a software approach, and it is used to improve the performance in terms of low latency, high bandwidth, less traffic, and better execution timing for Smart Memory Cube (SMC) Module.

Makni et al. [12] described ABMA based AXI4 bus protocol as hardware/software (H/S) approach for Wireless Sensor Network (WSN). SoC has three similar type interconnect bus protocols namely stream, lite and burst type. These interconnect bus protocols are configured and optimized by High-Level Synthesis (HLS) techniques. Compared with the benchmarked designs with improvements, the optimization techniques include loop pipelining, dataflow, and array partitioning. The functional verifications of the digital system are done by Bus functional Modelling (BFM) and Transaction-level modeling (TLM) techniques. The transaction-based SOC system is designed using the AXI-4 bus, interconnected with the help of VHDL language to provide cost-effective solutions on hardware [13].

Sebastian et al. [14] presented a Verification of Serial gigabit media independent interface (SGMII) IP core using Universal Verification Methodology – (UVM-VC) Verification component with AXI to Wishbone Bus (WB) Bridge as a software approach. By using AXI Bus, the coverage of SGMIII is improved along with the creation of a a reusable, reliable verification environment.

The verification Intellectual Property (VIP) for AXI4 as a software approach is designed in Prasad et al. [15], which provides the verification and IP core based flow for SOC designs. The functionality of five channel transactions and the verification of out-of-order and multiple outstanding transaction scenarios with the Questa tool. Sharma et al. [16] presented the design of conventional AMBA AXI-3 bus protocol as a software approach, which includes the write and read burst based transactions with verification analysis with the identification of code coverage findings. Panjkov et al. [17] addressed the Bridging of the well-known protocols, like AXI and OCP (Open core protocol), as a software approach which supports the multiple transactions. The Bridge mainly contains AXI Master, AXI downsizer, OCP Slave, and OCP to AXI Kernel. The AXI Master is designed using the write and read FSM to handle the handshake mechanism between an AXI Interface and bridge. The AXI bus interface module with the Master, Slave, and interconnect module is designed as a hardware approach by Ramesh et al. [18]. The interconnect module includes Master controller, read-write arbiter and decoder, and Slave controller. The AXI Bus interface supports 2 master and 4-slave communications. Archana et al. [19] explained the imaging chip concept using the AXI protocol for medical applications on a hardware platform. Fabio et al. [20] presented a reliable communication analysis between embedded processing systems and programmable logic as a Hybrid (hardware and software) approach, using AXI ports on ZYNQ-7000 Multicore ARM processor.

Research Gap: It has been noticed from the review works, and existing approaches, the significant work carried on the Interface-bus protocol designs are based either on software or hardware approaches. The research gaps are explained from the identification of existing approaches as follows:

1) Most of the interface modules, like bus-based architectures, shared bus connections are used in many SOC applications for communication purposes, which are facing scalability and reliability problems on the hardware platform.
2) Very few hardware-based designs work towards AMBA based protocols like AHB, APB, and ASB. Even with the AXI protocol, there has been very limited work carried out and they have a lot of constraints issues.
3) Many vendors designed soft-core AXI Protocol as an IP core. Although it is used in most of the applications, which includes MPSOC fast communications, they are not customized to other hardware.
4) Most of the AXI-based interface protocol design on hardware-based approaches are facing cost-effective solutions over SOC platform.
5) The Complete AXI-4 interface protocol with high-speed architecture is yet to come with optimized constraints to improve the performance of the SOC applications.

Hence an efficient, cost-effective solution with better performance is required to fulfill the above research gaps. The overview of the research methodology for the proposed design to address the research gaps is described in the next section.

III. Research Methodology

An efficient AXI-4 protocol is a high speed interface bus protocol, which corrects the drawbacks of existing bus interface protocols with better performance. The schematic flow of the proposed High-speed AXI-4 Interface is represented in Figure 1. The model includes AXI-4 Master Module, interconnect module, and AXI4 Slave module. These modules are interconnected, each based on AMBA-AXI-4 Input-outputs with its specifications, which are: Define the AXI-4 Master and slave Input-output (IO) as per ARM –AXI-4 Specifications [8] and Assign the Input-output connections as per five channels. The Master and Slave modules are designed using FSM (Finite state machine) along with five transactions which are: write address channel (WAC), write data channel(WDC), write response channel

(WRC), read address channel (RAC) and read data channel (RDC) with the response.

The proposed module provides a high speed interface between Master and slave modules via interconnection. The AXI-4 Interconnect provides a comman interface between Master and slave using a state machine. The AXI-4 Slave Module performs different transcations, which include write address ready signal generation, write address latching with different burst transactions, write ready and response signal generation, read address ready signal with burst transactions, memory-mapped register select, read logic signal generation and design the Block RAM (BRAM) to access the master data.

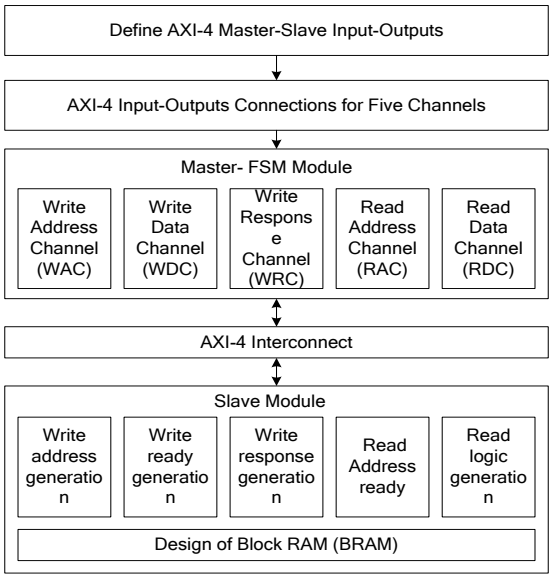

Figure 1: Schematic flow of proposed AXI-4 interface module

The study outcome offers high throughput, reduced resource consumption, and better performance in a single chip for low-cost SOC applications. It is also anticipated that the proposed scheme overcomes the drawbacks of traditional interface protocols with the constraints and performance improvements. The next section describes the proposed AXI-4 Master-slave design with FSM models with different Input-output functionality for high-speed SOC applications.

IV. PROPOSED SYSTEM

This section discusses the proposed AXI-4 Master-Slave Modules using FSM and state flow with AXI-4 Input-output functionality. The AXI protocol offers the latest key features for the next-generation technology in high-speed interconnection. The Features include the AXI-4 interface protocol, which suits low latency and high bandwidth designs, memory controller designs with low latency, interconnect architecture flexibility, backward compatible with previous ASP, AHB, and APR interface protocols.

The AXI-4 protocol supports write response requirement updation, up to 256 beats burst length support, Updated cache signal details, QoS signaling, and provides ordering requirements information. The AXI-4 protocol provides a burst-based transaction. In the address channel, every transaction has control and address information that deals with the type of data to be transferred. The data is transformed from the Master to the Slave module using a write data channel. The write response channel provides the completion of the transaction details in Slave to module using response and acknowledges signals. Data is transferred from slave to master along with the response to the Slave using read data channel. The detailed design description about master and slave is explained below.

A. AXI-4 Master Module Operation

The AXI-4 Master module is designed using five signal transactions. First, define the AXI-4 Master input-output port signals with many transactions and data width for write address channel, write data channel, write response channel, read address channel, and read data channel with a response. Next, the AXI-4 internal temporary signals for channel transactions are defined using AXI-4 Master input-outputs. The local parameter is set for a targeted Slave with the base address to 8'h80, and the Master module waits for start-counter, which is set to 16 clock cycles before initiating the write transaction. In the Master module, the burst size and burst length are allotted based on the total number of burst transfers. The write and read burst counters are used to find the number of burst transfers based on burst length.

The AXI-4 Master Input-output connections are set for five channel transactions. The master AXI address (AWADDR) is the concatenation of the targeted salve base address and active offset range (axi_awaddr). The write Burst length (AWLEN) is defined as based on the number of transactions minus one. The burst size (AWSIZE) is set to 3, which is 2^3 and equal to 8-bit data. The 2-bit increment burst type (AWBURST) is selected and set 2'b01. The master cache type (AWCACHE) is set 4'b0011, which indicates to cacheable and bufferable type. The write address valid (AWVALID) is set to 1, if valid address and control information is present, otherwise 0.

The master AXI Write data (WDATA) is 8-bit wide write data. Write strobe (WSTRB) is used to indicate which byte is used to update the memory. The Write Last (WLAST) is the last data transaction in a write burst. The write valid (WVALID) is set to 1, if data is valid, otherwise 0. The write response (BREADY) provides the response information, if it is 1= Master is ready, 0= Master is not ready. The AXI-4 Master Read address (ARADDR), the read burst length (ARLEN), the read burst size (ARSIZE), read increment burst type (ARBURST), Read cache type (ARCACHE), the read address valid (ARVALID) and read ready (ARREADY) is set the same as the write transactions.

B. Write Address Channel (WAC)

The address and control information is requested for all the transactions and processes for the write operation as early as possible. In the initial process, the valid address signal (awvalid) is reset, the initial next transaction is set, if the previous address is not valid. In the next clock cycle, awvalid=1. Once valid is set, wait for the AWREADY signal to accept the transactions. Once the AWREADY indicates the previous address is accepted, the next address is set based on the burst size and burst length.

C. Write Data Channel (WDC)

The write data is continuously forwarding to the slave via interface signals. If the valid (WVALID) and ready (WREADY) signals are high, the next transaction (wnext) will be started. Reset the write valid signal (wvalid) initially, start the next transaction, if it is not valid previously. In next clock cycle, wvalid=1. When many write transactions are in

the process (wnext), and till last burst (wlast), the WVALID signal must wait to complete the write process. The write counter is used to synchronize the last write data when it is full. The write last (wlast) is active only, write counter is equal to burst length along with the next transactions. Burst counter is used with an additional counter to perform the next transactions to avoid the decoding logic in AXI-4 interconnection. The write data (wdata) is generated based on the wnext logic, with incrementing till the burst count.

D. Write Response Channel (WRC)

The write response channel assures that all the write transactions are ready to store in slave memory. The BREADY is response ready. If it is 1, the master is ready to accept the slave response information. If it is 0, the master is not ready to receive the slave response information. Reset the write response ready (bready) initially; when the Write response valid (BVAILD) is valid from the slave and master is not ready to respond, Then, the next clock cycle, the master is ready to respond (axi_bready=1). If the Master is not ready, it retains its previous(axi_bready) value, as presented in Figure 2.

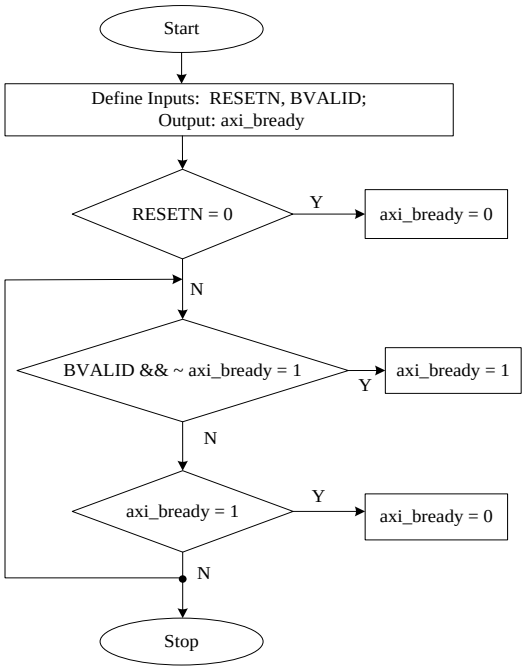

Figure 2: AXI-4 master write response state flow

E. Read Address Channel (RAC) and Read Data Channel (RDC) with Response

The read address channel works similarly to the write address channel. Instead of write address signals, read address signals are the process to complete all the read transactions. The Read data is continuously forwarding to the master via interface signals. If the valid (RVALID) and ready (RREADY) signals are high, the next transaction (rnext) will be started. The read last (rlast) is active only; the read counter is equal to burst length along with its next read transactions. When the read valid (RVAILD) is valid from the master, the slave is not ready to respond. Then, the next clock cycle, the slave is ready to respond (axi_rready=1). If the Slave is not ready, it retains its previous (axi_rready) value. When RVALID indicates that the required read is available and read transaction can complete according to the response and master ready.

Any data mismatch during the write and read transactions leads to read or write interface errors. If any read mismatch happens or error in write and read response, these error data are stored in the error register. The write and read burst counter are used to track the total number of burst transactions, which is initiated against the individual number of burst transactions for master or salve to initiate.

F. Master Interface Finite State Machine

The Master interface Finite state machine (FSM) is used to compare and validate the write and read transactions. The FSM mainly has five states, which include the IDLE, COUNTER, WRITE, READ, and FINAL states, as presented in Figure 3.

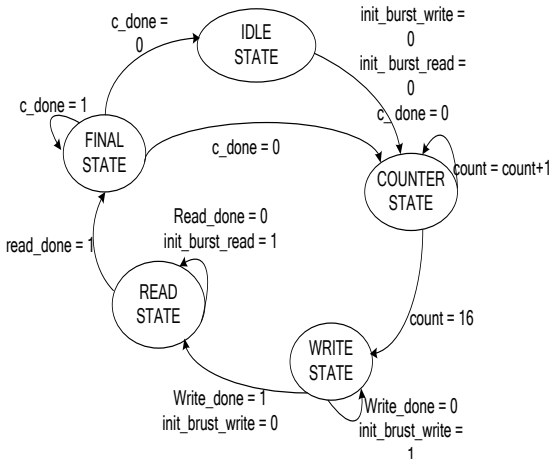

Figure 3: AXI-4 master interface finite state machine

The IDLE state is used to reset to initial values under reset condition and assign the next transition to COUNTER state. The COUNTER state initializes the counter and waits for start counter, which is set to 16 clock cycles, before initiating the write transaction, and once counting is done, the next transition to WRITE State is assigned. The Write state initiates the write transactions. It remains active till burst write signal (init_burst_write) is asserted, If the burst write signal is zero, the write transactions will stop and write done signal will be high and assign next transition to READ state. The Read state initiates the read transactions. It will remain active till burst read signal (init_burst_read) is asserted. If the burst read signal is zero, the read transactions will stop, and the read done signal will be high and the next transition to the FINAL state is assigned.

The FINAL State provides the final comparison of written data with read data. If any data mismatch are found, the error flag will set and assign the error data to error register and assign the next transition to IDEAL or COUNTER state. If the comparison is carried out (c_done) with no error, it indicates the transaction is completed.

The write and read done signals are activated based on the last write and read transactions, and which are dependent on write and read the response, ready and valid signals.

G. AXI-4 Slave Module Operation

The AXI-4 slave module supports burst based transactions and it is implemented with Block Random Access Memory (B-RAM). The slave module receives the master output signals as input signals through interconnects. The slave module write and read the data based on the five signal

transactions. The slave address and data width are fixed to 8, and the slave module receives the master write data and issue the read data. First, define the AXI-4 slave input-output port signals of write address, write data, write a response, read address, and read data channel with the response. Also, define the AXI-4 internal temporary signals for channels transaction using AXI-4 slave input-outputs.

The AXI-4 slave Input-output connections are set for write-read transactions. The master Output signals are input to the slave interface module. The slave output signals include thewrite address ready (AWREADY), which receives the address and control information, when it is set to 1 slave is ready to accept, and 0 slave is not to accept the information. The write ready (WREADY), which receives the data information, when set to 1 slave is ready to accept, and 0 slave is not to accept the write data information. The 2-bit write response (BRESP), which provides write transaction status information with a response like OKAY, EXOKAY, etc. The write response valid (BVALID) signal provides a valid write response if it is set to 1, otherwise set to 0 when it is not available. The write response ID (BID) is a write response identification tag if it matches with AWID, then the slave will respond to write transactions.

The read address ready (ARREADY), which receives the read address and control information, when set to 1 slave is ready to accept, and 0 slave is not to accept the information. The 2-bit Read response (RRESP) which provides read transaction status information with the response. The read response valid (RVALID) signal provides a valid read response if it is set to 1 otherwise 0. The Read Last (RLAST) is the last data transaction in a read burst. The Read ID (RID) is read identification tag if it is matched with ARID, then the slave will respond to read transactions. The slave AXI read data (RDATA) is 8-bit wide read data, which receives the block memory data as final data output.

The proper valid and response transactions are supported from the write flag (axi_flagw) and read flag (axi_flagr) registers. The write flags are reset to generate write address ready (axi_awaddr). If the AWVALID and previous control signals like write and read flags are set, then AWREADY will be active along with the write flag. If the WLAST is set high, the Slave is ready to accept the next address transactions after completion of the present write transaction.

H. Write Address Generation

The write address offset is defined based on the write address size and length. If both the AWVALID and WVALID signals are valid, the process of write address latching is started. Reset the write address (axi_awaddr) and burst length counter (axi_len_cnt). If the AWVALID and previous write flag (axi_flagw) is valid, then set the write address (axi_awaddr) using AWADDR [7:0] and burst length counter using AWLEN. The 2-bit burst type transaction is performing, which includes fixed burst (00), incremental burst (01) based on the burst sizes—wrapping burst (10) based on the wrap boundary.

I. Write Ready Generation

The write address (AWVALID) and write valid (WVALID) are valid when the write ready (axi_wready) is ready to accept them for the number of burst transactions (wdata). The write ready (axi_wready) is generated, when the WVALID and previous control signals like write flags are set, then axi_wready will be active. If the WLAST is set high, the slave is ready to accept the next address transactions after the completion of the present write transaction.

J. Write Response Logic Generation

The Slave module declares the valid response and write response signals when the WVALID and axi_wready are set, results in the response type is (00- OKAY) and valid response (bvalid). The master module's Ready response (BREADY) and slave's valid response (bvalid) are active, which results in the write transaction is accepted from the slave module.

K. Read Address Generation

The read address generation is similar to write address generation except, use the read address signals than write address signals. It also supports the read burst type.

L. Read Logic Generation

Reset the previous read response (rresp) and valid response (rvalid). Set the Read flag (axi_flagr), to access the slave module read data (axi_rdata) when rvalid =1 and read response (rresp=00) is OKAY. The master accepts the valid response when RREADY=1 is represented in Figure 4.

M. Block RAM Module

The block RAM is designed, and it supports up to 256 memory locations, each allocates 8-bit at a time. Based on the write and read flag set, the memory location is allocated.

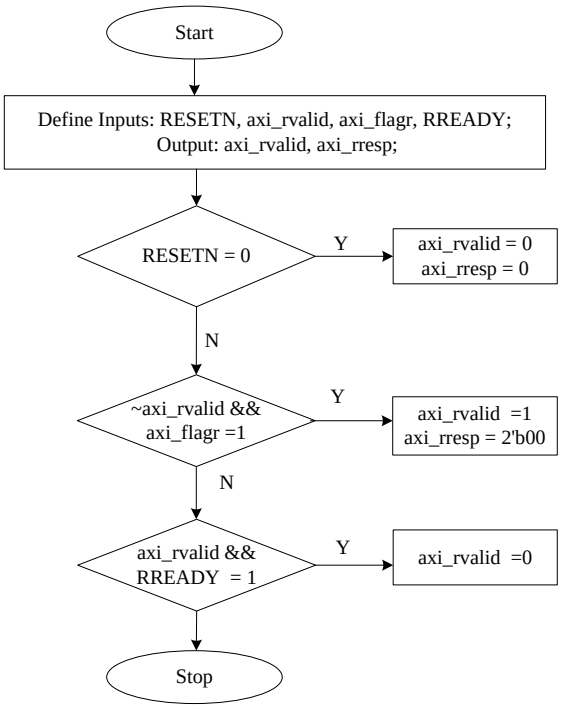

Figure 4: AXI-4 slave read response state flow

V. RESULTS AND ANALYSIS

The proposed AXI-4 Master-slave Protocol results are described in detail in the below section. The Complete AXI-4 Master-slave Protocol is designed using Verilog HDL over the Xilinx ISE Platform and simulated on Modelsim simulator and Hardware prototyped on low-cost Artix-7 FPGA.

The AXI-4 Master-slave Protocol simulation results are represented in Figure 5. The global clock (ACLK) is

activated with toggling with a positive edge. The Global asynchronous Reset Negedge (ARESETN) signal is initially set low, then keep it high to start the AXI-4 master-slave operations. The AXI-4 Master Module receives the incoming signals after the AXI-4 Slave response and, based on Master Module operation, generates the Master output signals, which are input signals to the AXI-4 Slave module through AX-4 interconnect signals. After the slave process, the AXI-4 Slave output signals will be generated after the response, and again slave output signals are input to the AXI-4 Master Module. The same process repeats till the data and address transaction happens.

The transactions are performed based on AXI- 4 Master-slave specifications, which includes write address (WA), Write Data (WD), Write Response (WR), Read Address (RA) and Read data (RD). These transactions are explained below as per simulation results.

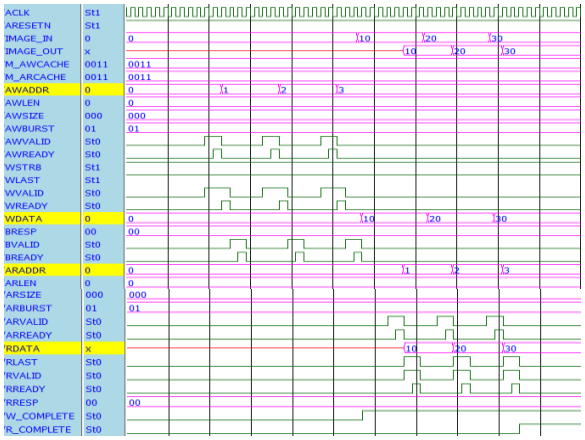

Figure 5: Simulation results of AXI-4 master-slave module

A. Master-Slave Write Address Transaction

The Write address valid output signal (AWVALID) will be high, followed by write address ready slave input signal (AWREADY=1) from the slave module to initialize the AXI-4 process. When both AWVALID and AWREADY are high, the master write address (AWADDR) initiates the address for each transaction and set to 1, 2 and 3 which is followed by master output signals like 8-bit Burst Length (AWLEN=0), 3-bit Burst size (AWSIZE = 000), 2-bit increment address Burst type (AWBURST=01) and 4-bit cache type (AWCACHE=0011) which is bufferable and cacheable are allocated.

B. Master-Slave Write Data Transaction

Write data transaction will perform parallelly with write address, the master output valid (WVALID=1) signal followed by slave ready (WREADY =1), when the write data (WDATA) transact the data for each transaction with an address. The 8-bit WDATA is 8-bit image data (IMAGE_IN) is set to 10, 20, and 30 to the corresponding 1, 2, and 3 address.

C. Master-Slave Write Response Transaction

For each data transactions, the master input or slave output write response valid (BVALID=1) signal followed by response ready (BREADY=1) signal will be high for one clock, both the signals will be low, till the new transaction happens and it will be followed till the last address and data transactions. The 2-bit master write a response (BRESP=00), which is an OKAY response about the transaction.

D. Master-Slave Read Address Transaction

When the write transactions are over, the read address transaction will start. The read address master valid output signal (ARVALID=1) will be high, followed by write address ready slave output (master input) signal (ARREADY=1) to initialize the AXI-4 read process. When both ARVALID and ARREADY are high, the master read address (ARADDR) initiates the address for each transaction and set to 1, 2, and 3, which is followed by master output signals like ARLEN=0, ARSIZE = 000, 2-bit ARBURST=01 and AWCACHE=0011 which are allocated.

E. Master-Slave Read Data Transaction

Read data transaction will perform parallelly with read address, the slave output valid (RVALID) signal followed by master ready (RWREADY) will be high, when the 8-bit read data (RDATA) receives 10, 20 and 30 data for each read address transaction completion. The Last read (RLAST) signal will be high for each read data transaction, and The 2-bit Read response (RRESP=00) indicates the OKAY response to receives the correct data.

The performance analysis of the proposed AXI-4 Master-Slave module and its sub-modules resource utilization in terms of Area (Slices) are presented in Table 1.

Table 1
Resource Utilization of AX-4 Master-Slave Module on Atrix-7

Logic Utilization	AXI-4 Master	AXI-4 Slave	AXI-4 Master-Slave
Slice Registers	49	24	63
Slice LUTs	61	28	77
LUT-FF pairs	48	17	56

The AXI-4 Master, Slave, and Complete AXI-4 Master-slave models area utilization after PAR (Place and Route) is obtained on the Atrix-7 Device environment. The AXI-4 Master-slave Utilizes 63 Slice registers, 77 Slice LUT's, and 56 LUT-FF pairs. The Complete AXI-4 works at a high frequency of 561.07 MHz. The AXI4 Design fits low-cost Artix-7 FPGA device at high speed. The total power consumption of the AXI-4 Master-slave Module from the X-Power Analyzer tool is 0.085W with a clock frequency of 100 MHz. The AXI-4 Full design utilizes less Area, power, and process at high speed on any low-cost FPGA Devices.

The comparative analysis in terms of Area utilization of the proposed AXI-4 Master-Slave module with AHB Protocol [21] on the same FPGA Spartan-3E device is tabulated in Table 2. The proposed AXI-4 improved in terms of Slice Registers around 6.80%, Slice LUT 40%, and LUT-FF Pairs 33.33% overhead over AHB Protocol. The AXI-4 works at 234.36, whereas AHB Protocol frequency is not mentioned by [21].

In a similar passion, the proposed AXI-4 module is compared with Wishbone Shared Bus Protocol [22] in terms of Area and frequency on the same FPGA device. The proposed AXI-4 improves in terms of Slice Registers Slice LUT, and LUT-FF Pairs is around 84 % over Wishbone Shared Bus Protocol [22]. The maximum frequency utilization of AXI-4 was improved around 41 % over Wishbone Shared Bus.

Table 2
Area Comparison of Proposed AX-4 with AHB-Bus [21]

Logic Utilization	AHB- Bus [21]	AXI-4 Proposed
FPGA Device	Spartan 3E-1600	Spartan 3E-1600
Slice Registers	44	41
Slice LUTs	95	57
LUT-FF pairs	96	64
Frequency(MHz)	-	234.36

Table 3
Resource Comparison of Proposed AXI-4 with Wishbone-Shared-Bus [22]

Logic Utilization	Wishbone-shared Bus [22]	AXI-4 Proposed
FPGA Device	Spartan 3E-500	Spartan 3E-500
Slice Registers	292	41
Slice LUTs	416	57
LUT-FF pairs	459	64
Frequency (MHz)	118.312	201.572

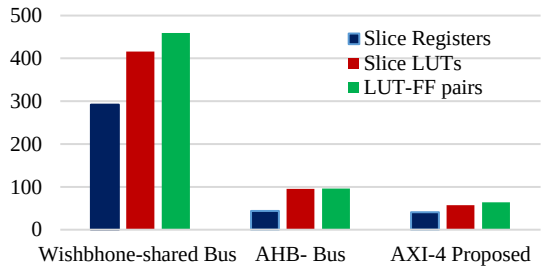

Figure 6: Comparative analysis of AXI-4 with other bus protocols

The AXI-4 protocol is compared with the traditional AHB and Wishbone bus protocols with improvements in hardware design constraints. Table 2 and Table 3 clearly show that the AXI-4 bus protocol is more efficient than the AHB and Wishbone bus protocols.

For communication and interconnection to multiple devices on SoC, the platform, the AXI-4, is highly commendable because of its ow cost, flexible less area and power utilization. It also works at a higher speed. The overview of comparative analysis is presented in Figure 6. The proposed AXI-4 utilizes less area overhead than the other two similar functionally working bus protocols, namely the AHB and Wishbone Bus protocol.

VI. CONCLUSION

In this paper, an efficient AMBA-AXI-4 interface protocol which offers high-speed communication between the processing elements has been designed,. The limitations of the conventional bus-based communications are overcome with the inclusion of Optimized AXI-4 interface protocol for high-speed SOC usage. The proposed AMBA-AXI-4 protocol module contains a Master, Interconnect model, and slave module. These modules are designed using efficient state machines. The simulation results of complete AXI-4 Master-slave are presented with a detailed description. The proposed model consumes very less area utilization and is tabulated after place and route on a low-cost FPGA device. The model work at high speed with an operating frequency of 561.07 MHz and utilize a small amount of power 0.085W. The proposed AXI-4 protocol provides a notable improvement in slice LUT's around 40% over the AHB bus protocol. When compared to wishbone bus architecture, a huge margin in area improvement and around 41% overhead in frequency. In the future, these models can be incorporated to image processing applications to process and monitor the images at high speed as an imaging chip.

REFERENCES

[1] W. Su, J. Wang, H, Wang, and L. Wang, "An Optimized Solution for Cross-Domain System Bus Transaction Processing", *In Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 14th ACIS International Conference on IEEE, 2013, pp. 165-170*

[2] Z. Li, J. Li, Y. Zhao, C. Rong, & J. Ma, "A SoC Design and Implementation of H. 264 Video Encoding System Based on FPGA", *In Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2014 Sixth International Conference on IEEE,* Vol. 2,2014, pp.321-324

[3] S. Ramagond, S. Yellampalli and C. Kanagasabapathi, "A review and analysis of communication logic between PL and PS in ZYNQ AP SoC", *In 2017 International Conference on Smart Technologies for Smart Nation (Smart Tech-Con),* 2017, pp. 946-951

[4] A. B. Mehta, "SoC Interconnect Verification", *In ASIC/SoC Functional Design Verification, Springer, Cham,* 2018, pp. 273-284

[5] D.C. Liang, "Hard real-time bus architecture and arbitration algorithm based on AMBA", 2015, pp. 1-7

[6] G. Mahesh and S.M. Sakthivel, "Functional verification of the Axi2OCP Bridge using system verilog and effective bus utilization calculation for AMBA AXI 3.0 protocol", *In Innovations in Information, Embedded and Communication Systems (ICIIECS), International Conference* on IEEE, 2015, pp. 1-5

[7] P.R. Ronak and S. Jagtap, "Design and verification of flexible interface for multicore system using PCIe IO virtualization", *In Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE International Conference on IEEE,* 2016, pp. 623-627

[8] AMBA, "AXI-Protocol Specification V2", *0. ARM Holdings plc. Std,* 2010

[9] C. Sarojini and J. Thangaraj "Implementation and Optimization of Throughput in High Speed Memory Interface Using AXI Protocol", *In 2018 9th International Conference on Computing, Communication, and Networking Technologies (ICCCNT),* 2018, pp. 1-5

[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] E. Azarkhish, D. Rossi, I. Loi, and L. Benini,"High-performance AXI-4.0 based interconnect for extensible smart memory cubes", *In Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition,* 2015, pp. 1317-1322

[12] 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

[13] D.C. Kho and K. Munusamy, "Transaction-based SoC design techniques for AMBA AXI4 bus interconnects using VHDL", In Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2014 11th International Conference on IEEE, 2014, pp. 1-6

[14] M. Gayathri, R. Sebastian, S.R. Mary, and A. Thomas, "A SV-UVM framework for verification of SGMII IP core with reusable AXI to WB Bridge UVC", *In Advanced Computing and Communication Systems (ICACCS), 3rd International Conference on IEEE,* 2016, pp. 1-4

[15] R.H. Prasad and C.S. Rani, "Development of VIP for AMBA AXI-4.0 Protocol", Indian Journal of Science and Technology, Vol.9, 2016, pp. 48

[16] 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

[17] 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

[18] R. Bhaktavatchalu, B.S. Rekha, G.A. Divya, and V.U.S Jyothi, "Design of AXI bus interface modules on FPGA", In Advanced Communication Control and Computing Technologies (ICACCCT), *International Conference on IEEE*, 2016, pp. 141-146

[19] H.R. Archana, and K.V. Patel, "A Novel Design and Implementation of Imaging Chip Using AXI Protocol for MPSOC on FPGA", *In Proceedings of the Computational Methods in Systems and Software Springer, Cham, 2018*, pp. 44-57

[20] F. Benevenuti and F.L. Kastensmidt, "Reliability evaluation on interfacing with AXI and AXI-S on Xilinx Zynq-7000 AP-SoC", *In Test Symposium (LATS), IEEE 19th Latin-American, 2018*, pp. 1-6

[21] A. Gaur, P. Sharma, and S.P. Pandey, "HDL and timing analysis of AMBA AHB on FPGA platform", *In Control, Automation & Power Engineering (RDCAPE), Recent Developments in* IEEE, 2017, pp. 22-27

[22] A.K. Swain, and K. Mahapatra, "Design and verification of WISHBONE bus interface for System-on-Chip integration", *In India Conference (INDICON), Annual IEEE*, 2010, pp. 1-4