JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

INTEGRATION AND VERIFICATION OF DMA CONTROLLER FOR OPEN POWER PROCESSOR CORE BASED FABLESS SYSTEM ON CHIP(SOC)

[1] Vema Akshaya, [2]Dr.Gannera Mamatha

[1]MTech Scholar, [2]Associate Professor
[1, 2,] MTech in VLSI-SD,
[1, 2,] JNTUA College of Engineering Ananthapuramu, Ananthapuramu, India

Abstract : This project propose to integrate and verification of the DMA controller via AXI4 interface for fabless SoC based on A2O cores for open power processors. The DMA controller provides an interface to user logic and supports eight channels with 64-bit data transfer for read and write operations involving data transmission between memory and memory, memory to peripheral, and peripheral to memory. Priority determines the channel assignment. Direct Memory Access, also known as DMA, provides direct interaction with the memory and devices while the CPU is temporarily deactivated or busy executing other orders in parallel. DMA gets control of the buses to transfer the data directly to the I/O devices. DMA completes the data transfer to all peripherals without the interference of the processor. The methodology used for designing is: develop system Verilog HDL code The verification of DMA controller can be created by using Verilog, System Verilog, and universal methodology (UVM). Software tools such as Xilinx Vivado® and Mentor Graphics (Questa®) can be used for the verification.

IndexTerms: DMA, A2O core, Open POWER, verification, integration, fabless, soc, system Verilog.

1.INTRODUCTION:

A specialized piece of hardware called Fast data transfers among memory and peripherals are made possible by a Direct Memory Having access (DMA) Controller, which eliminates the need for the CPU. improving system performance and efficiency. It plays a vital role in embedded systems and SoC architectures where fast data movement is crucial. The DMA controller operates by responding to a request from a peripheral device (via a DREQ signal), acquiring control of the system bus, and performing the transfer using designated address lines without CPU intervention. Once the data transfer is complete, it can generate an interrupt (INT) to inform the CPU. The main modes of DMA operation include Burst Mode, where large blocks of data are transferred continuously; Cycle Stealing Mode, where the DMA accesses the bus intermittently to allow CPU and DMA to share the bus; Transparent Mode, where DMA transfers occur only when the CPU is idle; and Block Transfer Mode, where transfers include auto-incrementing address pointers for efficient memory movement. DMA controllers typically have multiple channels, each of which can be independently configured to handle data transfers for different peripherals.

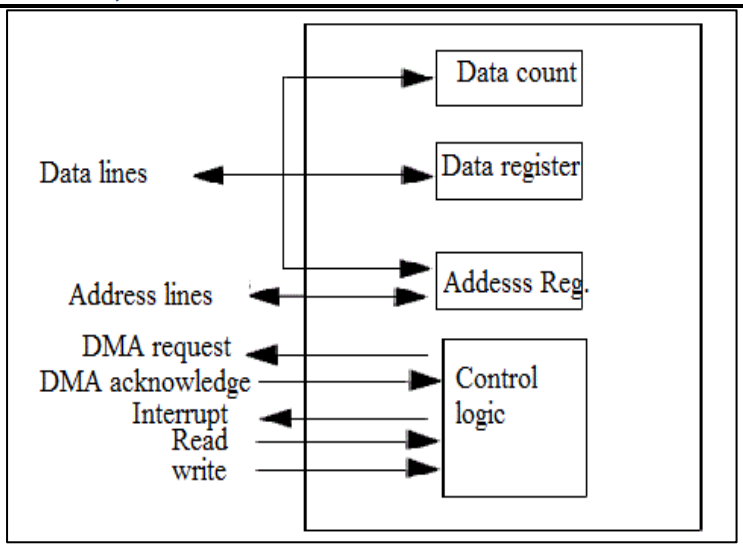

Fig: General block diagram of DMA

Each channel can have a specific priority and work either in parallel or sequentially, depending on the system design. The common pins found in DMA controllers include DREQ (request), DACK (acknowledge), RD, WR, address lines (A0, A1, etc.), CS (chip select), INT (interrupt), CLK (clock), and RESET. By offloading memory operations from the CPU, DMA significantly boosts system throughput, lowers CPU overhead, and supports real-time data processing in high-speed applications.

2.DMA ARCHITECTURE:

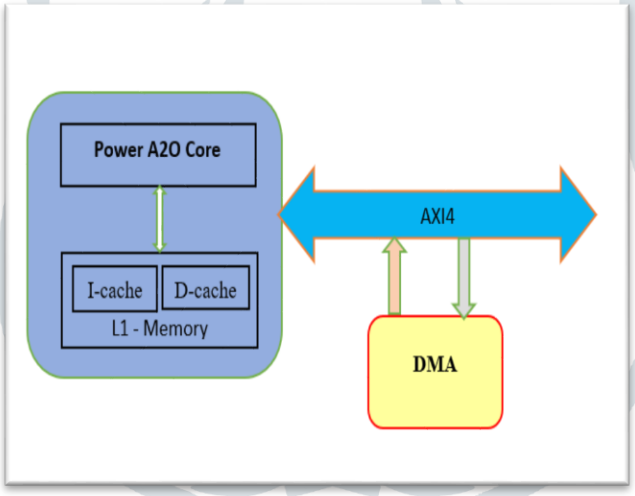

Fig: DMA with AXI4 Architecture

Burst Mode: In this mode, the DMA controller charges the system bus and only releases it once the data transfer is finished. Till then the CPU has to wait for the system buses.

Cycle Stealing Mode: In this mode, the DMA controller forces the CPU to stop its operation and relinquish the bus under the short-term DMA controller's control. The DMA controller requests the system bus again after releasing the bus after every byte has been transferred. The clock cycle used to transport each byte is thereby stolen by the DMA controller.Transparent Mode: In this mode, the system bus is only managed by the DMA controller if the processor does not need it.

3.INTERFACE WITH AXI:

The integration of a Direct Memory Access (DMA) controller with the A2O OpenPOWER processor through the AXI (Advanced eXtensible Interface) bus protocol is a critical aspect of system-on-chip (SoC) design that ensures high-speed, efficient data movement with minimal processor intervention. In this setup, the DMA controller acts as an AXI master, while the memory components or peripherals connected to the A2O processor function as AXI slaves. Five separate channels are supported by the AXI procedure, which is a component that makes up the ARM AMBA specification. These include the write destination channel, writer data channel, write response channel, read address channel, and read data channel—each designed to manage different aspects of

read and write transactions. During a write operation, the DMA sends the address and data over the respective write address and write data channels, and awaits acknowledgment through the write response channel.

Fig: AXI Interfacing

For read operations, the address is sent via the read address channel, and data is returned through the read data channel. The separation of these channels allows simultaneous operations, pipelining, and out-of-order processing, greatly enhancing throughput and latency characteristics. An AXI monitor is typically integrated into the system to capture, analyze, and verify all AXI transactions, ensuring adherence to protocol specifications and aiding in debugging during simulation and verification stages. A wrapper module is developed to bridge the native DMA interface with the AXI protocol, handling necessary signal translation and control logic. Once the interface is verified for correct functionality and protocol compliance, the DMA controller is integrated with the A2O core. This AXI-based integration enables the A2O processor to offload data movement tasks to the DMA, optimizing CPU utilization and improving overall system performance, which is particularly beneficial in data-intensive embedded and computing applications.

4.INTEGRATION WITH AXI4:

The AXI Interconnect is responsible for the arbitration and routing of AXI transactions between multiple masters and slaves. It connects the DMA master interface to one or more AXI slave peripherals or memory modules on the right side of the diagram. Additionally, the interconnect also links to slave ports (likely for control and status operations), which could be part of CPU or configuration logic. This architecture allows the DMA to autonomously read from or write to memory or peripherals, supporting high-speed data transfers within the SoC while reducing CPU load. The dense wiring reflects multiple AXI channels: address, write, read, control, and handshake signals, each critical to proper AXI4 protocol operation.

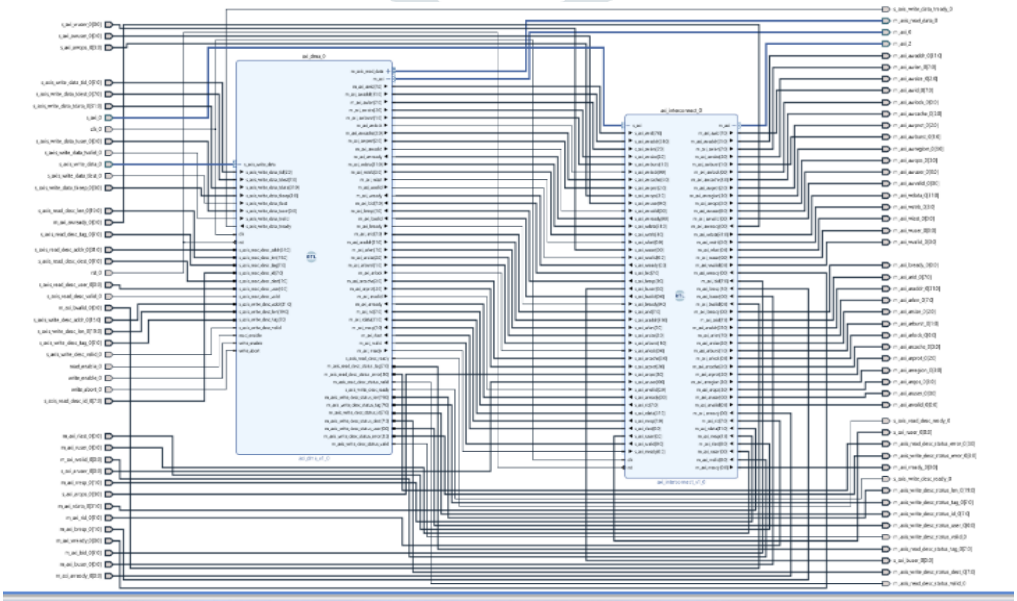

Fig: DMA with AXI Interconnect

5.INTEGRATION WITH A2O CORE:

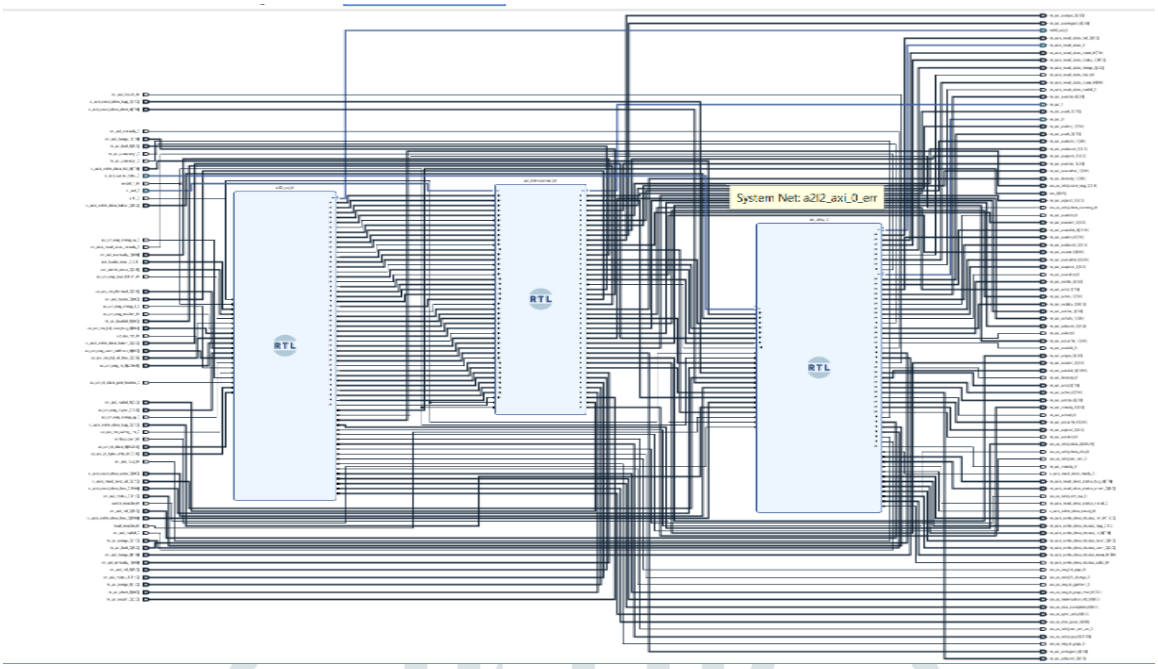

Fig: DMA with A2Ocore

Integrating a DMA controller with the A2O core enhances system efficiency by offloading memory transfer tasks from the processor. In this setup, the DMA's AXI slave interface is connected to the A2O core's AXI master port, allowing the processor to configure the DMA by writing to its control registers. Once set up, the DMA uses its AXI master interface to autonomously perform data transfers between memory and peripherals via the AXI interconnect. This architecture enables concurrent CPU processing and data movement, improving overall throughput. The A2O core can initiate DMA operations by specifying source/destination addresses, transfer length, and mode. The AXI interconnect routes the AXI transactions between DMA, memory, and other peripherals. Proper handshake and clock synchronization are maintained across components for reliable data flow. Verification includes validating control transactions from A2O and ensuring successful data transfer by the DMA. This setup is ideal for high-speed, low-latency SoC designs where CPU efficiency is critical.

6.RESULTS:

The design's Xilinx Vivado testing results are attached below: The integration of the DMA controller with the A2O processor through the AXI interface was successfully verified for both operations to read and write. In the process of writing, the DMA correctly initiated AXI write bursts, and the transaction completed with proper handshaking of AWVALID, WVALID, BVALID, and their corresponding ready signals.

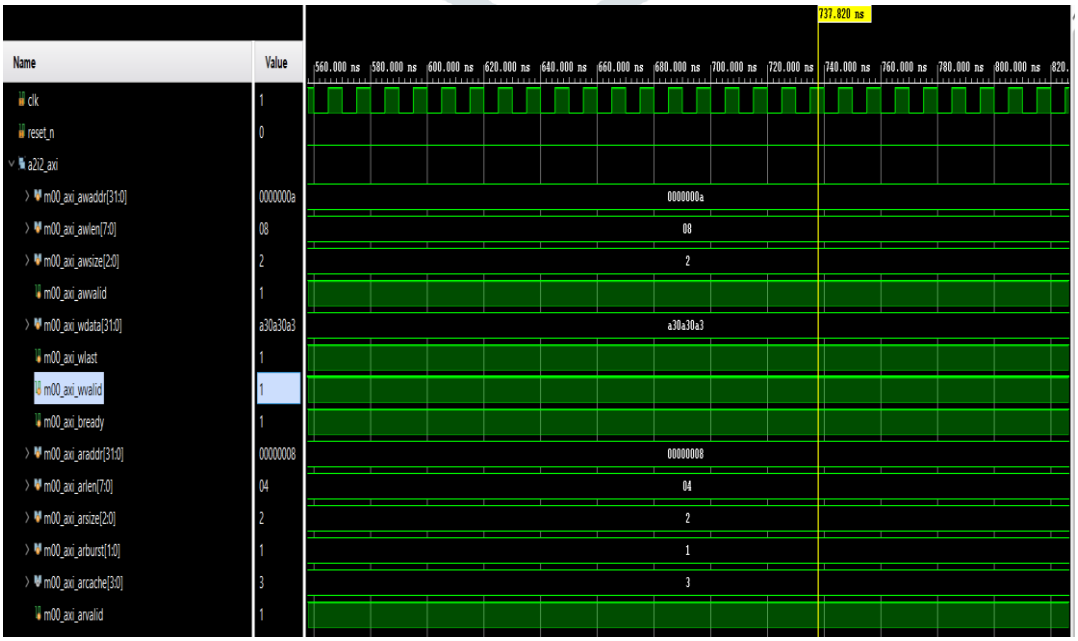

Fig: READ and WRITE data of A2O

The written data was accurately transferred to the destination memory, and verification confirmed data integrity. Similarly, for the read operation, the DMA successfully generated AXI read transactions, with correct timing of ARVALID, RVALID, and RLAST signals.

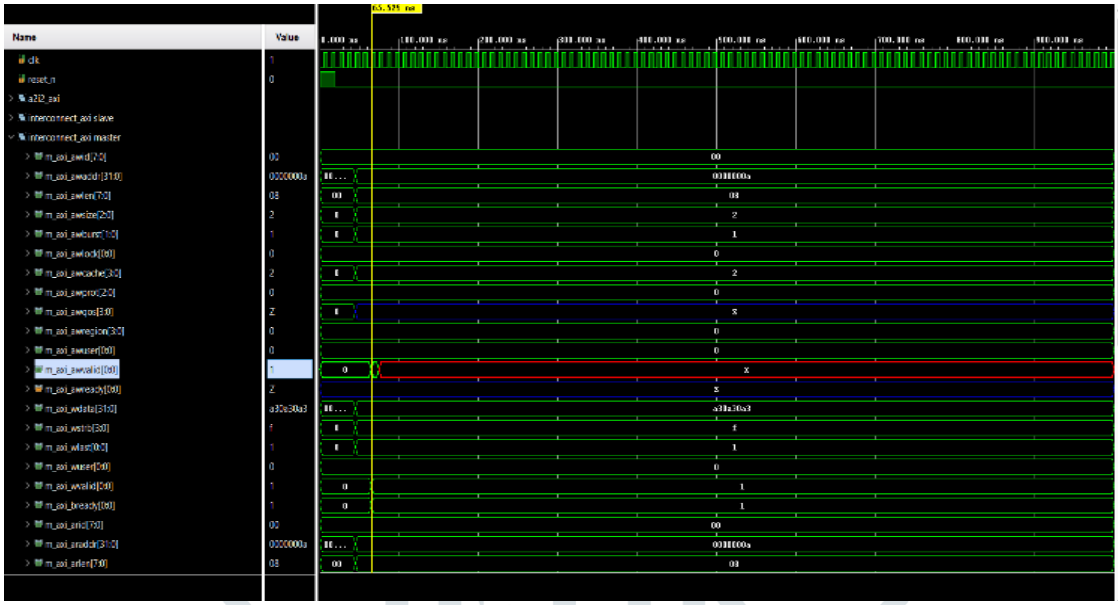

Fig: READ and WRITE data from AXI Inter Connect

The read data matched the expected source content, demonstrating reliable performance of the DMA module. These results validate the correctness and efficiency of the AXI-based DMA integration with the A2O core, making it suitable for fast data transmission in SoC environments.

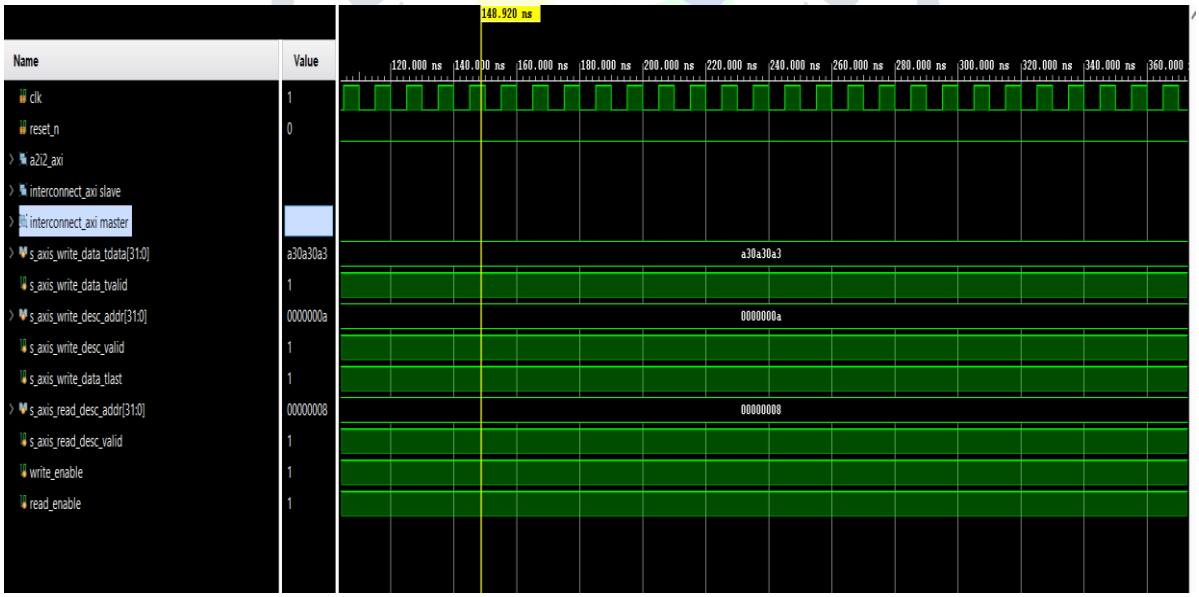

Fig: READ and WRITE data from DMA.

7.CONCLUSION:

The AXI4 DMA Controller was created, simulated, and synthesized in this work and used in fabless systems based on A2O processors. Cyclic buffer mode and other optional features are also accomplished via the controller. This suggested architecture could use AXI4 to enhance performance of data transfer. This layout is elaborated and interfaced, simulated with A2O in Xilinx Vivado using slave-based function unit testing with the write and read operations. This controller supports the slow communication devices with the help of the clock divider and the several cores.

8.REFERENCES:

[1] Open POWER Foundation. A2O Processor Core Documentation, 2020.

[2] Xilinx Inc. Vivado Design Suite User Guide: High-Level Synthesis, UG902, v2022.1.

[3] David A. Patterson and John L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 5thEdition, Morgan Kaufmann, 2013.

[4] R. D. Williams, "Design and Verification of AXI Compliant DMA Controller," International Journal of Engineering Research and Applications, vol. 6, issue 6, pp. 45–49, 2016.

[5] K. Chatha, A. Raghunathan, "Design of Efficient and Reliable DMA Engines for SoC Communication Architectures," Proceedings of DAC 2005, pp. 914–919.

[6] Kuncham, P., & Reddy, T. P. (2022). *IP Verification of DMA Controller for OpenPOWER Processor Core Based Fabless System on Chip (SoC)*.