Integration Of I2C Communication Protocol In Open Power Processor-Based Fabless Soc

K. Charitha[1] , Dr. M. Madhu Babu[2] , B.C. Vengamuni[3]

[1]M.Tech Student, Department of ECE, JNTUACEA, Anantapuram, Andhra Pradesh, India.
[2]Assistant Professor(A) Department of ECE, JNTUACEA, Anantapuram, Andhra Pradesh, India
3Assistant Professor(A) Department of ECE, JNTUACEA, Anantapuram, Andhra Pradesh, India.

Index Terms— I2C, Integration, Verilog, Open Power Processor.

Abstract—The Inter-Integrated Circuit (I2C) protocol is a widely used two-wire, half-duplex communication interface that enables short-distance data transfer between a master device and one or more slave devices. I2C is commonly employed for interfacing low-speed peripherals such as sensors and real- time clocks in embedded systems. This paper focuses on the integration of the I2C protocol with open power A2O core- based fabless SOC through AXI4 interface. The integration aims to facilitate efficient data exchange between the A2O processor and various peripherals with minimal processor intervention. A hardware-software co-design methodology combines Verilog HDL for hardware development and C programming for software control. The physical implementation ensures reliable commu- nication over the SCL (Serial Clock) and SDA (Serial Data) lines. The Integration and verifying an I2C controller can be created by using Verilog. Simulation and functional validation are conducted using industry-standard tools, including Mentor Graphics Questa® and Xilinx Vivado.

I. INTRODUCTION

The Inter-Integrated Circuit (I2C) protocol is a widely adopted serial communication standard developed to enable efficient data transfer between components on the same circuit board. It operates using only two bidirectional lines: SCL (Serial Clock Line), which synchronizes communication, and SDA (Serial Data Line), which is used for both transmitting and receiving data. I2C supports a multi-master, multi-slave configuration, allowing multiple devices to share the same communication bus without conflict [1][2]. Each device con- nected to the I2C bus is assigned a unique address, enabling selective and organized communication between devices.

To meet the needs of different types of applications, the I2C protocol defines multiple speed modes, which include:

a. Standard Mode: It operates at speeds up to 100 kbps and is widely used in applications requiring simple, low-speed communication, such as with RTCs and EEPROMs [3].

b. Fast Mode: Supports speeds up to 400 kbps, commonly used with LCDs and mid-speed peripherals. An enhancement to this, Fast Mode Plus, allows for data rates up to 1 Mbps, en- abling higher throughput while maintaining compatibility with I2C principles [4]. This is particularly useful for performance- critical applications in embedded systems.

c. High-Speed Mode: This mode offers speeds up to 3.4 Mbps and allows rapid bidirectional data transfer. It requires a compatible master to initiate and manage this mode, which is the fastest among the modes [5].

d. Ultra-Fast Mode: It supports speeds up to 5 Mbps but is designed for unidirectional write-only communication and does not support features like clock stretching or multi- master operation, trading flexibility for raw speed [6]. These varying modes make I2C highly versatile for a wide range of embedded and system-on-chip (SoC) applications [7].

These operating modes make I2C a versatile communication protocol suitable for various embedded systems and System-on-Chip (SoC) applications. Furthermore, the I2C architecture facilitates efficient, low-intervention communication between the processor and peripheral devices. When integrated with the AXI4 interface, it combines the simplicity of I2C with the modular, high-performance interconnect capability of AXI, making it ideal for scalable embedded applications [8].

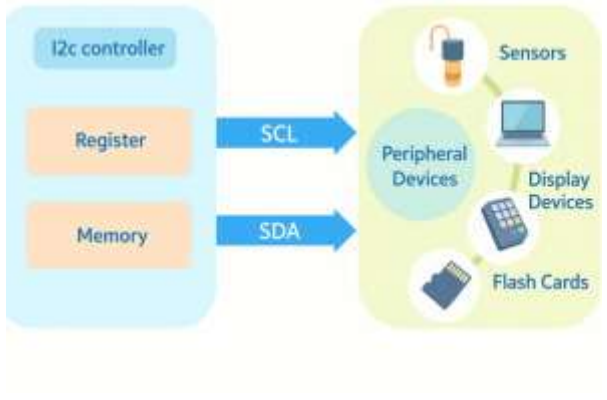

Fig. 1. Block diagram of I2C controller interfacing with the Peripheral

II. I2C ARCHITECTURE

The above diagram illustrates the integration of the I2C communication protocol with a Power A2O core-based System-on-Chip (SoC) using the AXI4 interconnect. The Power A2O core, an open-source 64-bit processor based on the POWER ISA, includes instruction cache (I-cache), data cache (D-cache), and level-1 (L1) memory to handle to maintain a balance between processing speed and commu- nication latency.

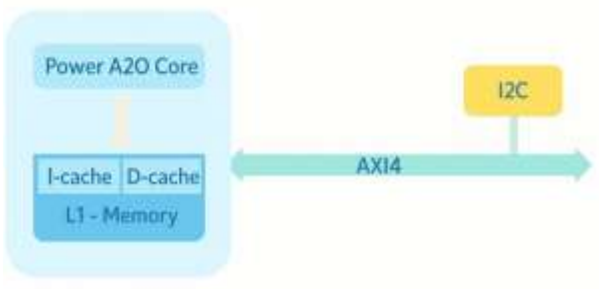

Fig. 2. I2C with open power processor through AXI4

data and instruction flow efficiently. Communication between the processor and peripheral devices is enabled through the AXI4 interface, a high-performance memory-mapped bus that supports scalable and modular interconnects in SoC designs. The I2C module is connected externally through the AXI4 interface, facilitating low-speed serial communication with peripheral devices such as sensors, EEPROMs, and RTCs using only two lines: SDA for data and SCL for clock. This architecture enables the A2O processor to manage I2C operations with minimal intervention, improving overall sys- tem efficiency and supporting hardware-software co-design in embedded applications.

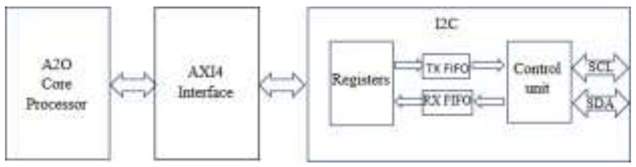

Fig. 3. Data Communication of I2C to Processor

During the data transmission process, the processor initiates communication by writing control commands or data to the memory-mapped registers associated with the I2C controller. These transactions are carried out through the AXI4 bus, which ensures efficient routing of the data from the processor to the I2C module. Once the data reaches the controller, it interprets the instructions and converts them into the appropriate I2C signaling patterns. The I2C controller then generates the required START condition, transmits the target slave address, and manages the data transfer through the SDA and SCL lines in accordance with the I2C protocol. Depending on the operation, it either reads data from or writes data to the peripheral device. The connected slave device acknowledges the transaction, and the response, whether it's the data read or an acknowledgment signal, is sent back to the processor. This sequence ensures synchronized communication between the processor and external devices while allowing the system

III. Design Flow

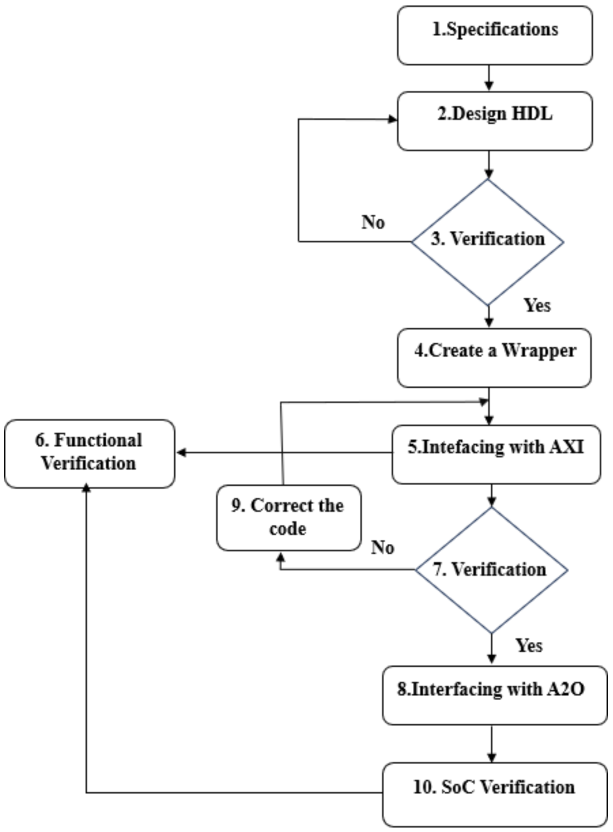

Fig. 4. Design Flow

I2C controller—into a system-on-chip (SoC) architecture, specifically targeting the A2O core using the AXI inter- face. The process begins with defining the system specifica- tions, which outline the functional requirements and design constraints. Based on these specifications, the hardware is implemented using Hardware Description Language (HDL) such as Verilog. Once the initial design is completed, it is evaluated to determine if it meets all predefined requirements. If the design does not satisfy the criteria, modifications are made, and the process loops back to the design stage. Upon meeting the requirements, the next step involves creating a wrapper, which encapsulates the HDL module to facilitate smooth interfacing with other components, particularly the AXI (Advanced eXtensible Interface) protocol. After the AXI interface is implemented, the design is checked again to ensure proper integration. If any issues are found, the code is corrected and retested. Once the AXI interfacing is verified, functional verification is performed to validate the design behaviour under expected operating conditions. Finally, after successful verification, the design is integrated with the A2O processor core, completing the hardware integration process.

This structured approach ensures a modular, verifiable, and reliable design suitable for complex SoC environments.

IV. RESULTS

A. Interconnect of I2C with AXI

The AXI Interconnect is responsible for managing the routing and arbitration of AXI transactions between multiple master and slave components. When integrating the I2C con- troller with the AXI4 interface, the interconnect connects the I2C master interface to one or more AXI slave peripherals or memory blocks within the system. It also facilitates commu- nication with control and status registers, which may reside in the CPU or configuration logic. This setup allows the I2C controller to perform data transfers efficiently, either sending or receiving information from memory or peripheral devices, with minimal processor intervention. The complex wiring in the system includes several AXI4 channels—such as address, write, read, control, and handshake lines—each playing a vital role in ensuring correct communication according to AXI4 protocol standards. This architecture supports reliable and high-speed serial communication between the I2C module and other system components in the SoC design.

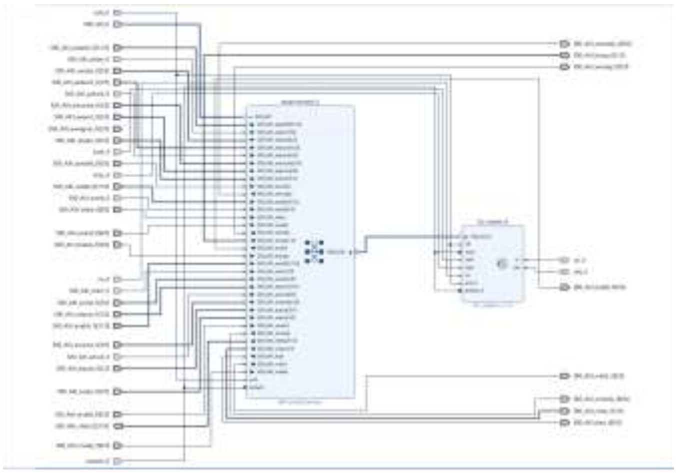

Fig. 5. Interconnecting AXI to I2C

B. Interconnect of I2C with Processor

Integrating an I2C controller with the A2O core improves system functionality by enabling efficient communication with external peripheral devices using a low-pin, serial interface. In this architecture, the I2C controller's AXI slave interface is connected to the A2O core's AXI master port, allowing the processor to configure the I2C controller by writing to its internal control and status registers. Once configured, the I2C controller operates as a master on the I2C bus, handling data transfers between the processor and connected peripherals through the two-wire interface (SCL and SDA). The AXI interconnect plays a key role by routing AXI transactions between the A2O core and the I2C controller, ensuring seam- less communication. The processor can initiate I2C operations by specifying the target device address, read/write mode, and data length. Synchronization, clock stretching, and ac- knowledgment mechanisms are maintained to comply with the I2C protocol. Verification involves testing register-level access from the A2O core and validating accurate data exchange with peripheral devices. This setup is well-suited for embedded SoC applications that require reliable, low-power, and low-latency peripheral interfacing.

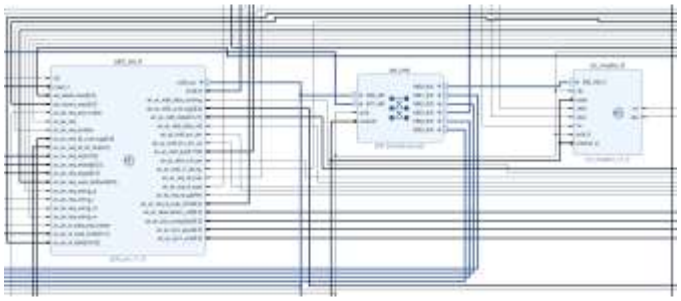

Fig. 6. Interconnecting Processor to I2C

V. SIMULATION RESULTS

Testing of the I2C controller integration was carried out using Xilinx Vivado, and the results are summarized below.

Fig. 7. Read and write data from A2O

The integration of the I2C controller with the A2O processor via the AXI interface was successfully verified for both data transmission (write) and reception (read) operations. During write operations, the A2O core configured the I2C controller through AXI transactions by asserting appropriate signals such as AWVALID, WVALID, and BVALID, along with their respective ready signals. The transmitted data was correctly passed to the I2C bus and delivered to the addressed slave device, with verification confirming the accuracy and integrity of the transfer. Similarly, in read operations, the I2C controller initiated data reception from the peripheral device. The A2O core successfully accessed this data via AXI read transactions, with correct timing of ARVALID, RVALID, and RLAST signals.

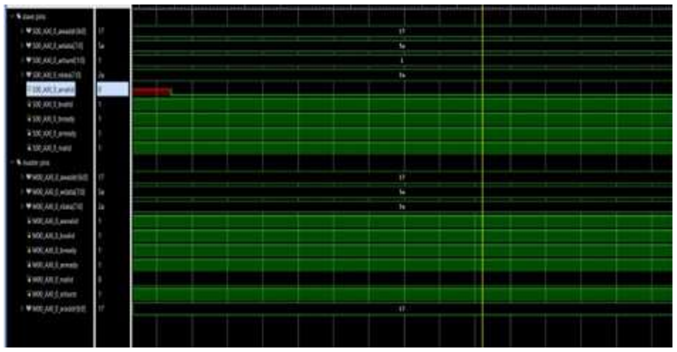

Fig. 8. Read and write data from AXI

The received data matched the expected values, demonstrat- ing proper functioning of the I2C controller in both communi- cation directions. These test results confirm that the AXI-based I2C integration with the A2O core functions reliably, making it a suitable solution for low-power, serial communication in embedded SoC designs.

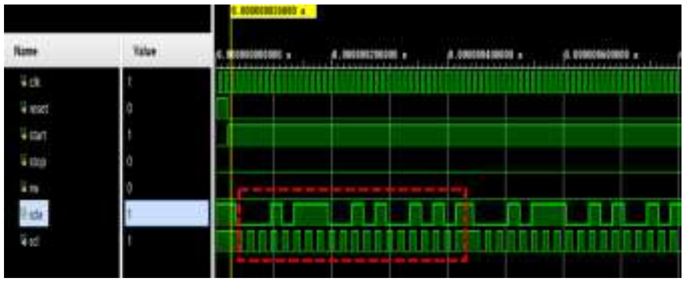

Fig. 9. write data in I2C

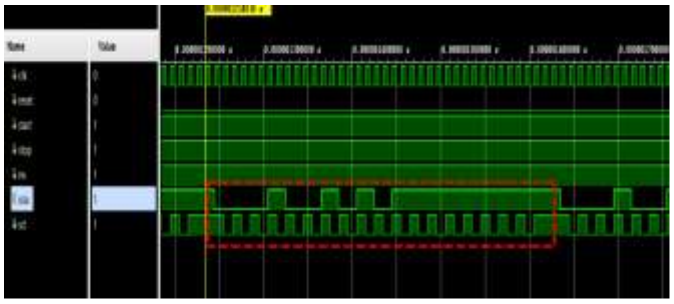

Fig. 10. Read data

VI. CONCLUSION

This work involved developing, simulating, and synthesiz- ing an AXI4-based I2C controller, integrated into an A2O processor-based fabless SOC. The I2C controller supports key features such as programmable clock division and multi-device addressing, enabling communication with various low-speed peripheral devices. The proposed architecture enhances data exchange efficiency between the processor and external com- ponents using the standardized AXI4 interface. The design was implemented and simulated in Xilinx Vivado, with functional testing carried out through AXI slave-based read and write operations. The controller is well-suited for interfacing with slow communication devices by utilizing a clock divider and flexible control logic, ensuring compatibility and reliability in embedded SoC applications.

REFERENCES

[1] Philips Semiconductors, ”The I2C-bus and how to use it (including specifications),” Application Note AN10216-01, 2000.
[2] Ko, H. et al., ”Design of an I2C protocol verification platform for SoC,” IEEE International SoC Design Conference, 2005.
[3] NXP Semiconductors, ”UM10204 I2C-bus specification and user man- ual,” Rev. 6, 2014.
[4] Lee, H. et al., ”Fast Mode Plus I2C Design and Implementation in Mixed-Signal Systems,” IEEE Transactions on Circuits and Systems, 2011.
[5] Xilinx Inc., ”AXI IIC Bus Interface v2.0 Product Guide,” PG090, 2022.
[6] Yoon, J. and Park, S., ”High-Speed I2C-Compatible Interface for High- Density Sensor Networks,” Sensors, MDPI, 2016.
[7] Goyal, P. et al., ”Implementation of I2C Protocol for Low Power Embedded Applications,” *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*, vol. 3, no. 4, pp. 1234–1238, 2014.
[8] Rahman, A. et al., ”AXI4-based I2C Controller Design for FPGA Integration,” International Conference on Electronics, Communication and Aerospace Technology, 2019.
[9] Leens, F., ”An introduction to I2C and SPI protocols,” *IEEE Instru- mentation Measurement Magazine*, vol. 12, no. 1, pp. 8–13, Feb. 2009.
[10] NXP Semiconductors, ”I2C-bus specification and user manual,” 2012.
[11] Ibrahim, D., ”16-bit micro programmable microcomputer with writable control store,” *IEEE Transactions on Computers*, vol. 39, no. 11, pp. 1385–1390, Nov. 2011.
[12] Shanavas, I.H. and Gnanamurthy, R.K., “Wavelength minimization in partitioning and floor planning using evolutionary algorithms,” *VLSI Design*, vol. 2011, Article ID 896241, 9 pages, 2011.
[13] Mulani, P., Patoliya, J., Patel, H., Chauhan, D., “Verification of I2C DUT using System Verilog,” *International Journal of Advanced Engineering Technology*, vol. 1, no. 3, pp. 130–134, Oct.–Dec. 2010.
[14] Accellera Organization, ”Universal Verification Methodology (UVM) 1.1 Class Reference,” June 2011.
[15] Glasser, M., ”UVM: The Next Generation in Verification Methodology,” *Methodology Architect*, Mentor Graphics Corporation, February 4, 2011.
[16] Yun, Y.-N., Kim, J.-B., Kim, N.-D., Min, B., ”Beyond UVM for practical SoC Verification,” *IEEE*, pp. 158–162, 2011. doi: 10.1109/SoC.2011.6119284
[17] Bhargav, P. and Reddy, M.J., ”Design and Implementation of I2C Master Controller on FPGA Using Verilog,” *International Journal of Engineering Trends and Technology (IJETT)*, vol. 45, pp. 123–127, 2017.
[18] Akhtar, N. et al., ”Implementation of I2C Master Bus Controller using Verilog HDL,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 5, 2013.
[19] Chen, W., ”A High-Speed I2C-Compatible Interface Design for Embed- ded Systems,” Proceedings of the International Symposium on Com- puter, Consumer and Control, 2015.