Power System Protection and Control
ISSN:1674-3415

AXI4-LITE UART FIFO DESIGN ON FPGA

Dr. S Jagadeesha[1], Dr Prathibha Kiran[2], S. Meghana[3], Prof Madhukar C S[4], Dr H N Suresh[5], Dr. Shivakumaraswamy G M[6]

Professor, Dept of ECE, AMC Engineering College, Jagadeesh.sd69@gmail.com
Associate Professor, Dept of ECE, AMC Engineering College, prathibhakiran1504@gmailk.com
M.Tech Student ,AMCEC, somarameghanabose@gmail.com
Associate Professor, Dept of E &T, JNNCE, csmadhukar@jnnce.ac.in
Professor, Dept of ECE, ,AMC Engineering College, suresh.narayana@amceducation,in
Associate Professor, Dept of ECE, BIET, Davanagere, gms20mar@gmail.com

Received: 22/03/2026 **Revised**: 22/04/2026 **Accepted**: 27/05/2026

***ABSTRACT*:**

The design presents the FPGA implementation of a UART data buffering system based on a synchronous FIFO with Axi4-lite interface. The design is for the Cyclone II FPGA development board EP2C5T144 device and was created using Intel Quartus Prime for synthesis and ModelSim for functional simulation, with all RTL written in synthesizable Verilog. A 50 MHz onboard oscillator powers the system, and a baud rate divider CLK_DIV = 434 allows UART operation at 115200 baud. Two 16-deep, 8-bit synchronous FIFOs buffer data between the UART receiver and transmitter, addressing the speed mismatch of serial connection 86.8 µs per byte at 115200 baud and providing data transfer. Functional simulation confirms proper UART framing, FIFO read/write operations, and timing behavior, whereas post-synthesis results demonstrate efficient resource utilization with fewer than 300 logic elements (LEs) and positive temporal slack at 50 MHz. Hardware validation using a USB-to-UART interface confirms reliable real-time data transmission and system stability.

***Keywords**: UART, Synchronous FIFO, FPGA, Cyclone II, Verilog, Quartus Prime, ModelSim,Serial Communication, RTL Design*

INTRODUCTION

Modern embedded systems demand efficient and dependable communication between high-speed processor units and low-speed peripheral devices. In such systems, standardised bus protocols such as AXI4-Lite provide a lightweight, fully handshaked interface for register-based communication, allowing for organised data flow between components. However, because of its simplicity and low hardware requirements, UART is still one of the most popular serial communication protocols for external communication and debugging.

A major challenge in integrating these systems is the data-rate mismatch between internal digital logic and serial communication interfaces. FPGA-based logic often operates at nanosecond-scale speeds using system clocks in the range of tens of megahertz, while UART transmission at 115200 baud takes around 86.8 µs to convey a single byte. If not correctly controlled, the large variation in operating speeds might result in data loss, buffer overflow, or inefficient use of system resources.

To overcome this issue, a synchronous FIFO buffer is introduced to decouple the data transfer between modules operating at different speeds. The FIFO temporarily stores incoming data from the faster domain and allows controlled, sequential transmission to the slower UART interface. This buffering mechanism ensures smooth data flow, prevents overflow conditions, and enables continuous communication without requiring tight synchronization between producer and consumer modules. The UART connection in this design adheres to the conventional 8N1 frame structure, which consists of one start bit, eight data bits transferred in least significant bit (LSB) first order, and one stop bit. Because UART does not have a shared clock, a baud rate generator is used to ensure accurate timing. To achieve a baud rate of 115200 with a 50 MHz system clock, use a clock divider value of CLK_DIV 434. This ensures accurate bit timing and reliable data transfer.

The synchronous FIFO architecture employs (ADDR_W+1)-bit read and write pointers to reliably detect full and empty circumstances, removing ambiguity during pointer wrap-around. In this paper, a 16-byte FIFO buffer is used to efficiently manage data transfer between the UART receiver and transmitter modules. This architecture

DOI: 10.46121/pspc.54.2.40

not only increases data integrity but also boosts system performance by permitting burst data handling and lowering reliance on immediate processing, making it ideal for FPGA-based embedded communication systems.

LITERATURE SURVEY

To evaluate system reliability and robustness under radiation-induced errors. Overall, these works [1] highlight the importance of efficient AXI-based communication in improving data transfer performance and system efficiency. However, existing approaches [2] mainly focus on bridging techniques between AXI and FIFO for data transfer without addressing complete system-level integration and optimization. Similarly, the work in [3] emphasizes UART design with AXI interfaces, concentrating on communication functionality, but lacks deeper enhancements in buffering strategies and efficient data handling mechanisms. The asynchronous FIFO techniques discussed in [4] effectively solve clock domain crossing issues using synchronization methods, yet they are not integrated with higher-level communication protocols such as AXI and UART. Therefore, there is a clear need to combine these individual concepts into a unified and optimized architecture[5]. The proposed work [6] builds upon these limitations by developing an AXI-FIFO-UART system that ensures efficient, reliable, and scalable communication suitable for modern embedded and FPGA-based applications.

The work proposed in [7] uses Chisel-based modular and parameterized design techniques to enhance scalability, flexibility, and performance in FPGA-based hardware accelerators, making it easier to design reusable and high-performance systems. Finally, the work in [8] introduces an FPGA-based platform for SEU characterization in memory devices, incorporating fault injection and monitoring techniques to analyse system reliability, thereby contributing to the development of more robust and fault-tolerant FPGA-based designs.

SYSTEM ARCHITECTURE

Methodology:
The system is designed, with UART communication, synchronous FIFO buffering, and an AXI4-Lite interface on FPGA. The UART receiver uses a baud rate generator to transform serial data into parallel form, whilst the FIFO stores data momentarily and regulates flow using full and empty signals respectively. The AXI4-Lite interface provides controlled data access via memory-mapped registers. All modules are combined into a top-level design and implemented in Verilog, which is then simulated and hardware verified.

Block diagram:
The block diagram shows the overall architecture of the FPGA-based UART data buffering system. It enables communication between a host PC and the FPGA using a USB-to-UART module, while internally using FIFO buffers and an AXI4-Lite interface for efficient data handling.

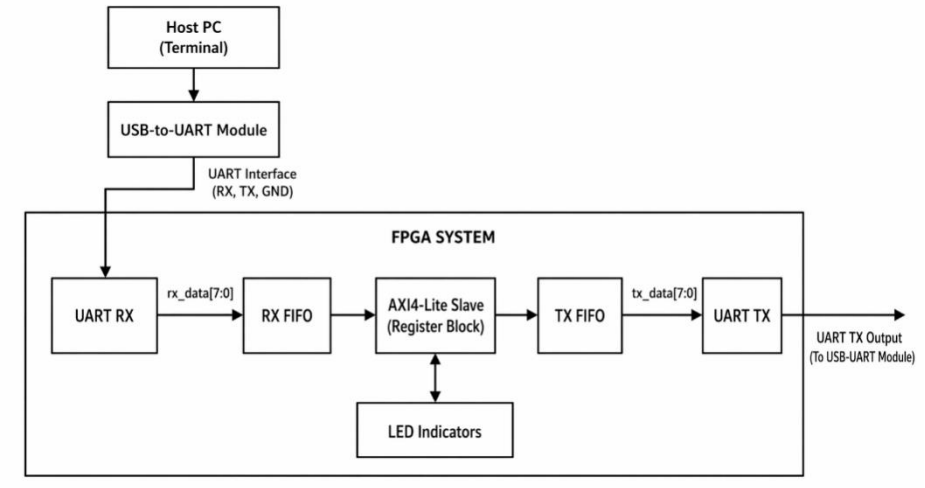

Fig 3.1. Block diagram of AXI4-Lite based UART FIFO system implemented on FPGA

DOI: 10.46121/pspc.54.2.40

As shown in the figure 3.1, the host PC communicates with the FPGA through a USB-to-UART module, which converts USB signals into UART-compatible serial signals. The UART interface consists of RX, TX, and GND connections, enabling bidirectional communication between the external system and the FPGA.

The host PC communicates with the FPGA through a USB-to-UART module. Inside the FPGA, the UART RX receives serial data and converts it into parallel data rx_data [7:0], which is stored in the RX FIFO. The AXI4-Lite slave interface controls data transfer between the RX FIFO and TX FIFO using register-based access.

The TX FIFO stores outgoing data, which is then transmitted through the UART TX back to the PC. LED indicators provide status information such as FIFO conditions and data activity. This design ensures reliable data transfer by handling the speed mismatch between UART communication and FPGA logic.

Design Flow: The design flow of the proposed FPGA-based UART FIFO system follows a structured sequence to ensure correct implementation and verification. It includes stages from initial specification to final hardware validation, covering both design and testing aspects.

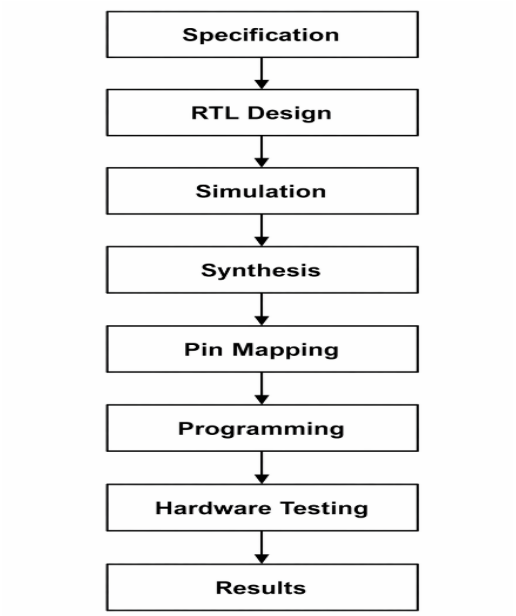

Fig 3.2. Design flow of the proposed AXI4-Lite based UART FIFO system on FPGA

As shown in the figure 3.2, the process begins with defining system specifications, followed by RTL design using Verilog. The design is then verified through simulation and synthesized to generate hardware logic. Pin mapping assigns FPGA resources, after which the design is programmed onto the board. Finally, hardware testing is performed to validate system functionality and analyze results.

IMPLEMENTATION

Hardware Implemetation

The hardware setup shown in the figure 4.1 illustrates the practical implementation of the proposed system using an FPGA development board interfaced with a host PC through a USB-to-UART module. The host PC runs a serial terminal application to transmit and receive data over a USB connection. The USB-to-UART module (FT232/CP2102) converts USB signals into UART-compatible serial signals.

DOI: 10.46121/pspc.54.2.40

Power System Protection and Control
ISSN:1674-3415

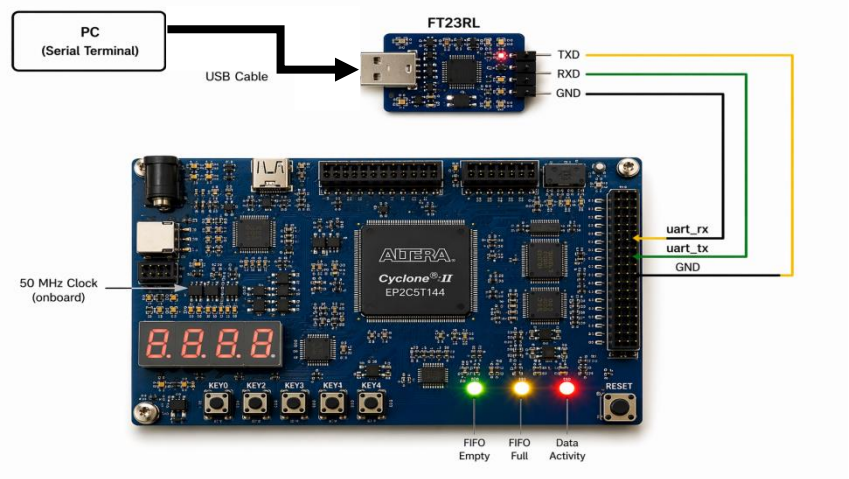

Fig. 4.1. Hardware implementation of UART FIFO system using USB-to-UART interface and FPGA board

The transmitter (TXD) of the USB-to-UART module is connected to the UART receiver (uart_rx) pin of the FPGA, while the receiver (RXD) is connected to the UART transmitter (uart_tx) pin, ensuring proper cross-connection for bidirectional serial communication. A common ground (GND) connection is maintained between the module and the FPGA to provide a stable voltage reference and ensure reliable signal transmission. The serial communication is configured at a standard baud rate of 115200, allowing synchronized data exchange between the host PC and the FPGA.

The FPGA operates using an onboard 50 MHz clock, which serves as the primary timing source for UART operation, FIFO control, and internal logic. A reset mechanism is provided to initialize the system and ensure proper startup conditions. Internally, the UART modules handle serial-to-parallel and parallel-to-serial data conversion, while the synchronous FIFO buffers incoming and outgoing data to manage the speed mismatch between the UART interface and high-speed FPGA logic. Control logic coordinates data movement and ensures correct sequencing. LED indicators are used to represent system status signals such as FIFO empty, FIFO full, and data activity, providing visual feedback during operation. This hardware setup enables real-time testing, confirms correct data buffering and transmission, and demonstrates reliable communication between the host system and the FPGA.

Software Implemetation

The RTL implementation is divided into five functional modules: a UART transmitter, a UART receiver, a synchronous FIFO, an AXI4-Lite slave interface, and a top module that connects these blocks and handles overall data flow and control.

The proposed system is implemented using a modular RTL design in Verilog HDL, integrating UART communication, synchronous FIFO buffering, and an AXI4-Lite interface within a unified top-level architecture. The UART receiver operates using a finite state machine (FSM) that transitions through idle, start-bit detection, data reception, and stop-bit validation states, with sampling controlled by a baud rate generator derived from the system clock (CLK_DIV = 434 for 115200 baud at 50 MHz). The received serial bits are shifted into a register and presented as parallel data (rx_data[7:0]) with a corresponding rx_valid signal. This data is written into the receive FIFO, which uses synchronous read/write operations and pointer-based addressing to maintain data order. The FIFO employs (ADDR_W+1)-bit pointers to distinguish full and empty conditions, using logic such as assign empty = (wr_ptr == rd_ptr); to ensure safe data access and avoid ambiguity during pointer wrap-around. The AXI4-Lite interface manages data transfer using independent read and write channels with VALID/READY handshaking, enabling memory-mapped communication and proper synchronization between modules. On the transmission side, the UART transmitter is controlled by an FSM that sequences start, data, and stop bits, with timing governed by the baud rate generator to ensure correct bit intervals. Data is fetched from the transmit FIFO using control logic such as assign tx_fifo_rd_en = tx_ready && !tx_fifo_empty;, allowing continuous data transmission without underflow. The integration of these modules within the top-level design ensures efficient

DOI: 10.46121/pspc.54.2.40

Power System Protection and Control
ISSN:1674-3415

handling of the speed mismatch between high-frequency FPGA logic and low-speed UART communication, providing reliable and synchronized data transfer across the system.

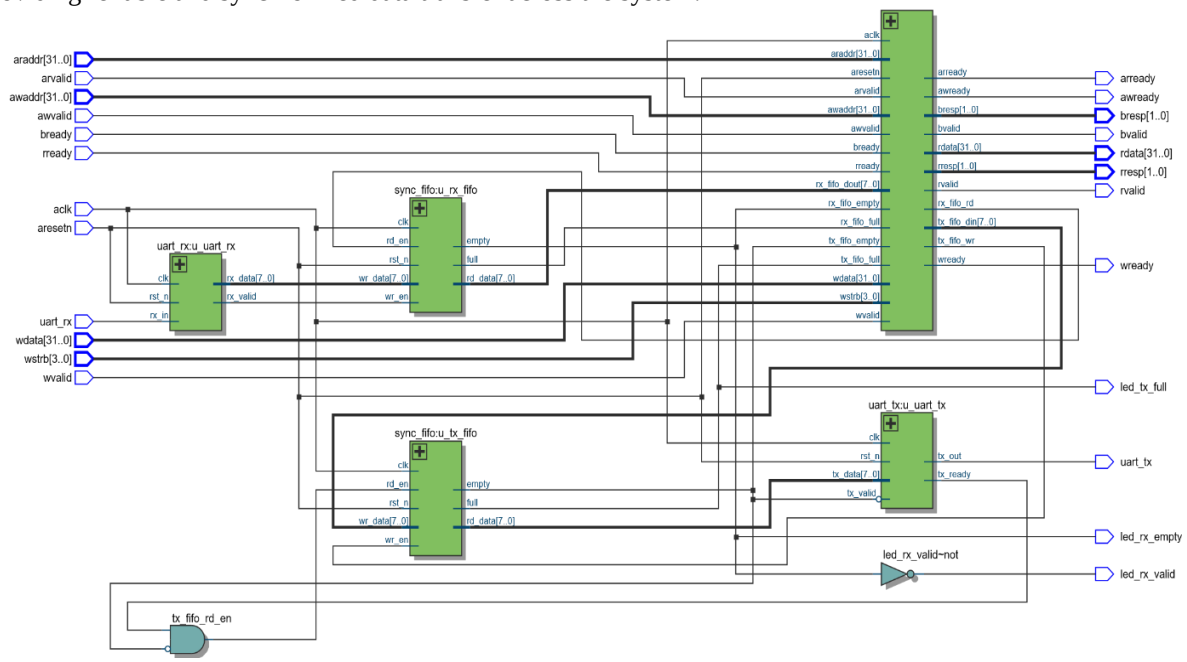

Fig.4.2. Top-level RTL schematic of AXI4-Lite based UART FIFO system on FPGA

The figure 4.2 represents the top-level RTL implementation of the proposed system, where all submodules are interconnected to achieve efficient and synchronized data communication. The UART receiver samples the incoming serial signal (uart_rx) using a baud rate generator and converts it into parallel data (rx_data[7:0]), asserting a rx_valid signal upon successful reception of a byte. This data is written into the receive FIFO using wr_en, while the FIFO generates status signals such as rx_fifo_empty and rx_fifo_full to control data flow. The AXI4-Lite slave interface performs memory-mapped communication by decoding addresses (awaddr, araddr) and handling read/write transactions using VALID/READY handshake signals (awvalid, wvalid, arvalid, rvalid, bvalid). It controls FIFO operations through signals like rx_fifo_rd and tx_fifo_wr, enabling data transfer between the receive and transmit FIFOs. On the transmission side, the transmit FIFO provides data (tx_data[7:0]) to the UART transmitter, which initiates serial transmission when tx_valid is asserted and indicates readiness through tx_ready. The transmit FIFO read operation is governed by logic such as tx_fifo_rd_en = tx_ready && !tx_fifo_empty, ensuring continuous data flow without underflow. Overall, the use of FIFO status flags, handshake signals, and control logic ensures proper synchronization, prevents data loss, and enables reliable communication between high-speed FPGA logic and the low-speed UART interface.

RESULTS AND DISSCUSION

Simulation Results

Figure 5.1 illustrates the UART transmission process for the data byte 0xA5 written through the AXI interface. The data is first stored in the transmit FIFO, as indicated by the FIFO control signals. When the UART transmitter becomes ready (tx_ready = 1), the system asserts the tx_valid signal for one clock cycle, enabling the transfer of data from the FIFO to the UART module.

DOI: 10.46121/pspc.54.2.40

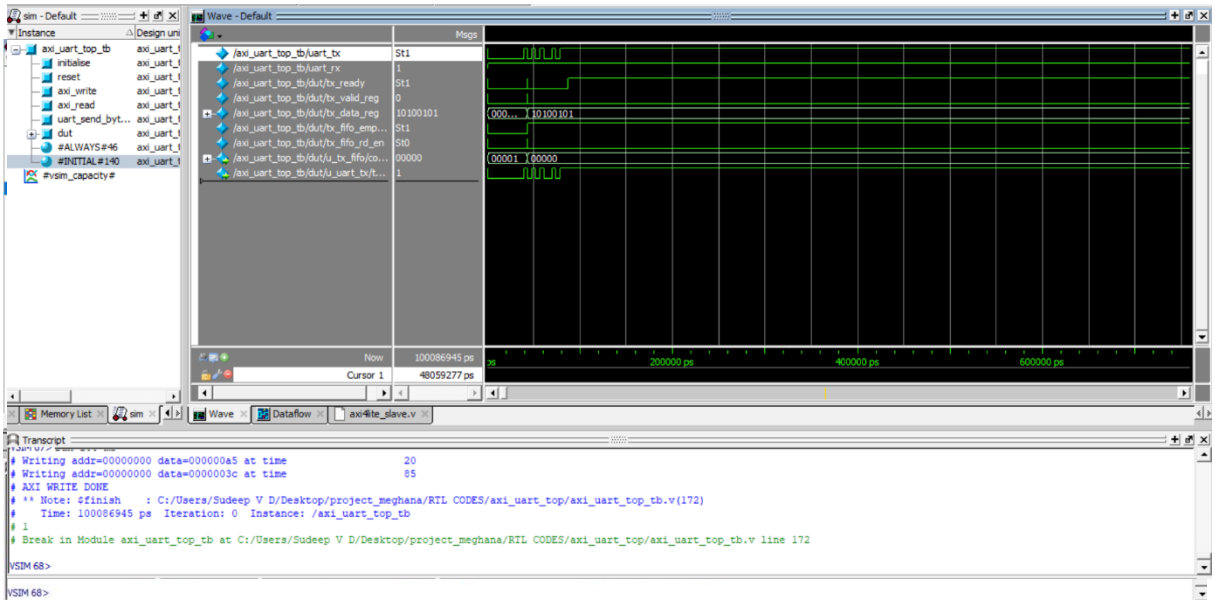

Fig.5.1 AXI to UART Data Transmission Waveform through FIFO Buffer

The signal tx_data_reg holds the transmitted value (10100101), confirming correct data propagation. The tx_fifo_rd_en signal indicates that data is being read from the FIFO. Following this, the UART transmitter serializes the data and produces the corresponding bitstream on the uart_tx line. The waveform clearly shows the transition of bits corresponding to the transmitted data.

Figure 5.2 presents a detailed view of the UART transmission. The serial output on uart_tx follows the standard UART frame structure, beginning with a start bit (logic low), which indicates the start of data transmission. This is followed by eight data bits (10100101), transmitted in a sequential manner, and finally a stop bit (logic high), marking the end of the frame.

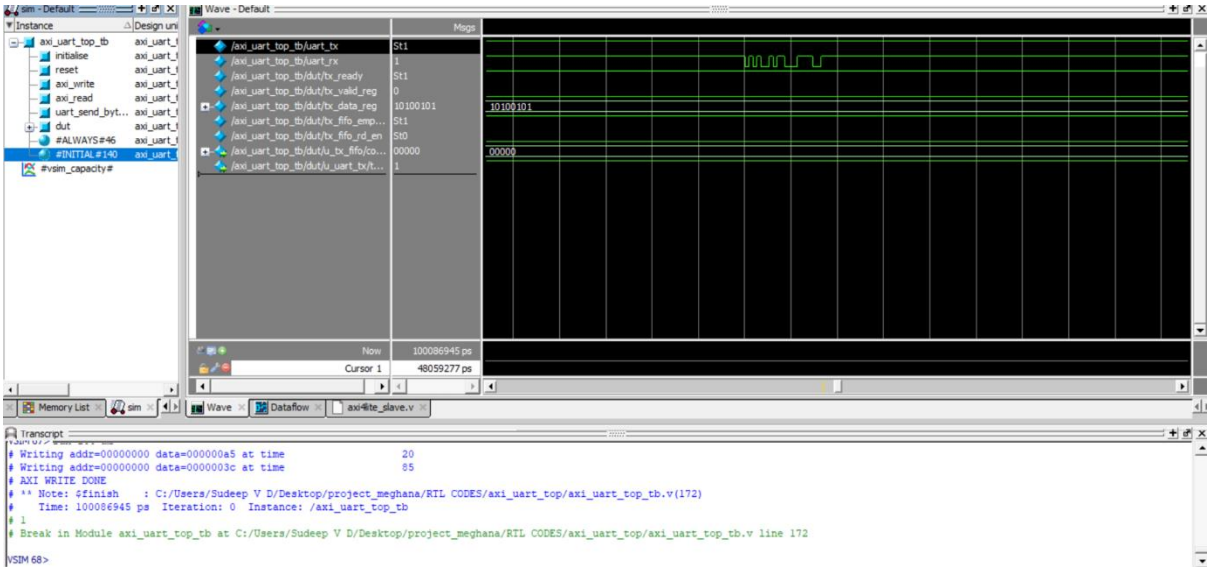

Fig.5.2 Detailed UART Bitstream Showing Start Bit, Data Bits, and Stop Bit

Each bit is transmitted over a fixed time interval, determined by the configured baud rate, resulting in uniform spacing between transitions in the waveform. This consistent timing confirms that the baud rate generator within the UART module is functioning correctly.

DOI: 10.46121/pspc.54.2.40

The observed waveform validates that the UART transmitter successfully converts parallel input data into a serialized bitstream while maintaining proper synchronization and timing accuracy. This ensures reliable communication between the transmitting and receiving ends.

Hardware

The hardware implementation of the proposed AXI-based UART system was verified on an FPGA platform. The AXI interface successfully writes input data into the transmit FIFO, which acts as an intermediate buffer between the processor interface and the UART module. Upon detection of valid data and UART readiness, the control logic enables data transfer from the FIFO to the UART transmitter.

The system ensures proper synchronization using ready-valid handshaking, allowing reliable data movement without loss. The FIFO prevents data overflow and maintains continuous data flow, even when there is a mismatch between AXI and UART operating speeds.

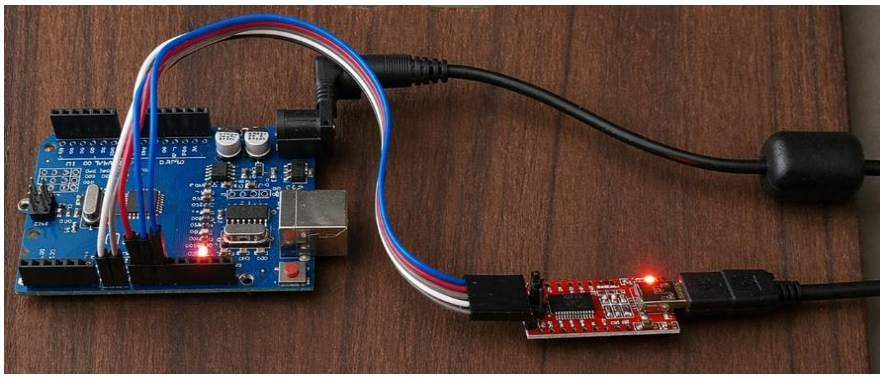

Fig.5.3 Hardware Setup for AXI-Based UART Communication System

As shown in Figure 5.3, the transmitted data is successfully serialized on the *uart_tx* line. The UART output follows the standard communication protocol, consisting of a start bit, eight data bits, and a stop bit. The observed bit pattern corresponds to the transmitted data (0xA5), confirming correct functionality of the system.

The uniform timing between signal transitions indicates accurate baud rate generation and stable clock operation. The results demonstrate that the designed system correctly integrates AXI communication, FIFO buffering, and UART transmission, achieving reliable hardware-level data transfer.

FPGA Resource Utilization

Table: 5.1 FPGA Resource Utilization of the Proposed AXI-FIFO-UART System

Resource	Used	Available (10M50)	Utilization
ALMs (Adaptive Logic Modules)	~275	49,760	0.55%
Flip-Flops / Registers	~260	99,520	0.26%
M9K Block RAM bits	0	1,677,312	0% (distributed)
I/O Pins	6	360	1.7%
PLLs	0	4	0%

Table 5.1 represents the FPGA resource utilization of the proposed system implemented on the Intel 10M50 device. The design occupies a very small fraction of available resources, utilizing only 0.55% of ALMs and 0.26% of registers. The FIFO is implemented using distributed logic instead of dedicated block memory, resulting in zero M9K usage. The low resource consumption demonstrates that the proposed design is lightweight and suitable for integration into larger FPGA-based systems.

Timing and Performance Analysis

Table: 5.2 Timing and Performance Characteristics of the Proposed System

Parameter	**Value**
Clock Frequency (Target)	50 MHz (20 ns period)
Fmax (Post-Fit)	> 120 MHz

DOI: 10.46121/pspc.54.2.40

CLK_DIV (115200 baud)	434 clock cycles
UART Bit Period	8.68 µs
UART Byte Latency	86.8 µs
AXI Write Latency	60–100 ns
AXI Read Latency	60–80 ns
Maximum TX Throughput	11,520 bytes/sec
FIFO Burst Transfer (16 bytes)	~320 ns load, 1.39 ms drain

Table 5.2 summarizes the timing and performance characteristics of the proposed design. The system operates at a target clock frequency of 50 MHz, while achieving a post-fit maximum frequency greater than 120 MHz, indicating robust timing performance. The UART operates at a baud rate of 115200, corresponding to a bit period of 8.68 µs and a byte transmission latency of approximately 86.8 µs.

The AXI interface demonstrates low latency, with write and read operations completing within a few clock cycles. The system achieves a maximum throughput of 11,520 bytes per second. Additionally, the FIFO enables burst data handling, allowing multiple bytes to be loaded rapidly and transmitted sequentially.

CONCLUSION

This work presented the complete design and FPGA implementation of an AXI4-Lite based UART data buffer using a synchronous FIFO on the Intel DE10-Lite platform. The system employs AXI communication, FIFO buffering, and UART transmission to ensure efficient and dependable data transfer between a CPU interface and a serial communication channel. The design assures accurate timing and reliable operation by using a 50 MHz clock and a calibrated CLK_DIV = 434 for 115200 baud. Functional simulation confirmed proper AXI handshaking, FIFO operations, and UART serialisation, whilst synthesis results revealed very low resource utilisation and positive timing slack. The parameterised architecture provides for simple scalability in terms of FIFO depth and baud rate, making the solution adaptable and portable across many FPGA platforms.

The proposed system is suitable for a variety of real-world applications, such as embedded system communication, FPGA-based data logging, and serial debugging interfaces. It has applications in industrial automation for real-time data interchange, as well as IoT and edge devices for transferring sensor data to external systems. Furthermore, the design can be integrated with soft processors such as Nios II or other AXI-based systems, allowing for use in custom SoC designs. Overall, the provided approach offers a lightweight, scalable, and efficient communication interface for FPGA-based applications.

FUTERSCOPE

The AXI-FIFO-UART system can be further improved by adding an interrupt-based mechanism using an rx_irq signal, which avoids continuous polling of the status register and reduces processor load. Additionally, making the CLK_DIV parameter configurable through an AXI register would allow dynamic adjustment of baud rates, improving flexibility for different communication requirements.

REFERENCES

1. Chaithanya, Sainath, Sameera Sulthana, and Ch Haritha. "Design of AMBA AXI4-lite for Effective Read/Write Transactions with a Customized Memory." *International Journal on Emerging Technologies* 11, no. 1 (2020): 396–402.
2. Efil, Mahmut. "Design and Verification of AXI4-Stream to FIFO Bridge Communication Protocol." 2022.
3. Basavaiah, Jagadeesh, Abhishek R. Bharadwaj, and T. K. Raj. "A Review on Design and Verification of Programmable UART with AXI." *International Journal for Multidisciplinary Research (IJFMR)* 6, no. 3 (2024).
4. Cao, Yuxiang. "AXI4 Interfaces on a Deep Neural Network Accelerator." 2024.
5. Kandeh, Stephen. "FPGA Based Data Acquisition System for Cryogenic Device Verification." PhD diss., Massachusetts Institute of Technology, 2025.

DOI: 10.46121/pspc.54.2.40

6. Cummings, Clifford E. "Simulation and Synthesis Techniques for Asynchronous FIFO Design." In *SNUG 2002 (Synopsys Users Group Conference, San Jose, CA, 2002) User Papers*, vol. 281, 2002.
7. Gay, Robin, and Tarek Ould-Bachir. "An Open Chisel-Based Framework for Hardware Acceleration on High-Performance FPGA Cards." 2025.
8. Molina Montes, Mariana. "Design and Experimental Validation of a Versatile FPGA-Based Platform for SEU Characterization of Memory Devices." PhD diss., Jyväskylän yliopisto, 2025.
9. Alsalmani, Abdullah. "Design Modular Command and Data Handling Subsystem Hardware Architectures." 2023.
10. Lu, Ci-Chian. "Hardware Implementation and Analysis of Memory Interfaces to Integrate a Vector Accelerator into a Manycore Network-On-Chip." Master's thesis, University of California, Santa Barbara, 2023.
11. Roostaie, Vahid. "Design and Analysis of a Coherent Memory Sub-System for FPGA-Based Embedded Systems." 2011.
12. Basha, K. Hassen. "Localization Using IMU/GPS Sensors for Mobility Assistant for Visually Impaired System (MAVI)." PhD diss., Indian Institute of Technology Delhi, 2016.
13. Magnaye, Gilbert, Josh Poh, Myo Tun Aung, and Tom Sun. "SMPTE2022-5/6 High Bit Rate Media Transport over IP Networks with Forward Error Correction on Kintex-7 FPGAs." Xilinx Application Note, 2013.
14. Bi, Jinrui, Hongyu Zhang, Lihua Sun, and Qingchao Jiang. "Design and Realization of Dynamically Adjustable Multi-Pulse Real-Time Coherent Integration System." *Electronics* 15, no. 2 (2026): 397.
15. Velarte, Antonio, Antonio Castel, Aranzazu Otin, Aida Olivan-Viguera, and Esther Pueyo. "A Modular Soft Core-Based System for Affordable Acquisition and Processing of Electrophysiological Recordings." *Results in Engineering* (2026): 109764.
16. Malavika, M. U., and Karthikeyan Murukesan. "CDMA in Embedded Systems to Transfer Data Efficiently from SPI Interface Memory Access to BRAM Memory." In *2025 5th Asian Conference on Innovation in Technology (ASIANCON)*, 1–6. IEEE, 2025.
17. Niraula, Amrit. *Zero Trust Architecture for Peer-to-Peer Communication Using Blockchain and Hardware-Oriented Security.* The University of Toledo, 2021.
18. Ferreira, André Tiago Pereira Almendra. "Development of a Spectrum Analyzer Based on FPGA." Master's thesis, Universidade do Minho, 2023.
19. Wang, Rui, Jianyu Zhang, Chunhua Jiang, and Hui Liu. "High-Speed ADC Interface Design Based on JESD204B Protocol." In *Fourth International Conference on Signal Processing and Computer Science (SPCS 2023)*, vol. 12970, 205–210. SPIE, 2023.
20. Papaloukas, Emmanouil. "HW/SW Co-Design and Preprocessing for Accelerating Star Trackers on SoC FPGA." 2022.
Obaid, Mohammad Tasneem. "Efficient Implementation of Sobel Edge Detection with ZYNQ-7000." Master's thesis, Purdue University, 2020.
21. Saini, Lalit Mohan. "Overview of AMBA AHB2APB Bridge for SOC Interconnects." In *2024 International Conference on System, Computation, Automation and Networking (ICSCAN)*, 1–6. IEEE, 2024.
22. Gaikwad, Nikhil, and Vijay N. Patil. "Verification of AMBA AXI on-chip communication protocol." In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1-5. IEEE, 2018.
23. Mahesh, Golla, and S. M. Sakthivel. "Verification IP for an AMBA-AXI protocol using system Verilog." *International Journal of Engineering Research and General Science* 3, no. 1 (2015).
24. Yamini, R., and M. V. Ramya. "Design and Verification of UART using system

DOI: 10.46121/pspc.54.2.40

verilog." *International Journal of Engineering and Advanced Technology (IJEAT)* 9, no. 5 (2020): 1208-1211.

25. Samanth, Rashmi, and Subramanya G. Nayak. "Design and sv based verification of AMBA AXI protocol for SoC integration." *International Journal of Recent Technology and Engineering* 8, no. 2 (2019): 1465-1469.

DOI: 10.46121/pspc.54.2.40