International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Design and Implementation of UART Protocol Using Verilog

Mr. Vinayak Sinha[1], Ms. Puja Priya[2], Mr. Nikhil Singh Negi[3], Ms. Tanuja Kumari[4], Ms. Shreyanshi Jaiswal[5]

[1,3,4,5]Student, Electronics and Communication Engineering, Gl Bajaj Institute of Technology and Management
[2]Supervisor, Electronics and Communication Engineering, Gl Bajaj Institute of Technology and Management

Abstract
The given paper introduces a novel approach which involves the design and implementation of a UART (Universal Asynchronous Receiver/Transmitter) module with FIFO (First-In-First-Out) buffering, developed using Verilog HDL and verified on a Basys 3 FPGA board through Xilinx Vivado 2023.2.) UART is a standard serial communication interface used in embedded systems for reliable data exchange. FIFO integration enhances serial data flow by reducing CPU over- head and improving full-duplex performance. In addition to functional implementation, this paper identifies the research gap between existing FIFO-based UART designs, particularly focusing on earlier works that lacked hardware buffering or efficient timing strategies. The study identifies which FPGA platforms are suitable for low-power, high-performance UART applications, and highlights improvements over earlier FPGA generations. The results demonstrate that the system meets its performance targets in terms of timing, power, and resource utilization, while the simulation waveforms validate the correct functional operation of both transmitter and receiver modules.

Keywords: UART · Verilog HDL · FIFO · Oversampling · Baud rate generator · FPGA · Serial communication

1 Introduction

In FPGA-based systems, UART (Universal Asynchronous Receiver/Transmitter) is commonly used to transmit data by converting parallel data into a serial stream. However, dur-ing high-speed data transmission, issues such as improper data transfer, data loss, and power overload can occur. These problems negatively affect system reliability and contribute to inefficient power usage in FPGA designs [1]. Asynchro-nous FIFO (First In, First Out) buffers offer an effective solution to address these challenges. FIFO structures help manage data flow efficiently particularly between differ- ent clock domains by ensuring synchronization, reducing CPU load, and lowering power consumption in high-speed

1 Department of Electronics and Communication Engineering, ITS Engineering College, Greater Noida, Uttar Pradesh, India
2 Department of Electrical and Electronics Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Bengaluru, India
3 Center for Renewable Energy and Microgrids, Huanjiang Laboratory, Zhejiang University, Zheijang, China
4 Dept. of Computer Science and Applications, Sharda University, Greater Noida, Uttar Pradesh, India
5 Department of Computer Science and Engineering, Manav Rachna International Institute of Research and Studies (MRIIRS), Faridabad, Haryana, India

systems [2]. The main difference between ordinary mem-ory and FIFO is that FIFO does not use external read/write address lines. Instead, it relies on internal read and write pointers that automatically increment to manage data flow [3]. Asynchronous FIFOs are widely used in various appli-cations such as network data transmission (ensuring data integrity), operating system scheduling (preserving task order), and cache controllers (for temporary data buffering). Moreover, asynchronous FIFOs can synchronize cross-domain data exchange more efficiently and support faster data transfer compared to traditional handshake protocols [4]. The given research introduces UART to ensure reli-able, efficient, and accurate asynchronous communication. It involves comparing various FPGA boards to identify the ones best suited for low-power, high-performance appli-cations. This analysis includes evaluating parameters like speed, area, and power consumption, along with the com-patibility of each board for integrating FIFO-based designs. A few development tools are also explored such as Xilinx ISE, Xilinx Vivado, Intel Quartus Prime, and ModelSim to determine which tools provide the most robust support for advanced integration and simulation [5]. This research aims to address gaps in areas of power optimization, data reliabil-ity, and scalability of FIFO implementations across different FPGA platforms. The given paper is novel in the following aspects:

- The given paper introduces a novel approach which involves the design and implementation of the UART system to ensure reliable, efficient, and accurate asyn-chronous communication.
- The system is synthesized, implemented, and tested using the Xilinx Vivado Design Suite on the Basys 3 FPGA board. It offers a balanced trade-off between per-formance and power consumption, supports dynamic power management, and is fully compatible with Viva-do. This makes it an ideal platform for FIFO integration and modern UART design implementations.
- The designed UART system is verified through synthesis, implementation, and simulation using the Xilinx Vivado Design Suite. The results demonstrate that the system meets its performance targets in terms of timing, power, and resource utilization, while the simulation waveforms validate the correct functional operation of both transmitter and receiver modules.

2 Literature review

Several studies have explored the implementation of UART (Universal Asynchronous Receiver/Transmitter) in FPGA systems as shown in Table 1. In Table 1, bold is done to show more clarity to the readers.

3 Proposed approach

The design and implementation of the UART system fol-lows a structured, step-by-step methodology to ensure reli-able, efficient, and accurate asynchronous communication. The complete process is illustrated in Fig. 1, which outlines the key phases from requirement analysis to final testing and result evaluation.

The design begins with requirement analysis, defining UART communication parameters such as baud rate, data length, stop bits, and parity.

System-level constraints are also considered, including the target FPGA platform (Basys 3 – Artix-7 family), power efficiency goals, and toolchain compatibility (e.g., Xilinx Vivado).

In the system design stage, the complete UART archi-tecture is formulated. Major components include the trans-mitter, receiver, FIFO buffers, baud rate generator, and Finite State Machines (FSMs) for

control [15]. Signal flow between modules is conceptualized using block diagrams for clarity and planning.

Figure 2 shows the UART Receiver FSM flowchart, and same figure show the Transmitter flowchart which governs the state transitions for receiving data based on sampled clock ticks and data bit counting. The states include IDLE, START, DATA, and STOP, progressing according to tick sampling and logic conditions.

Table 1 Existing studies overview

Cited works	Key Features	Technology Used	Identified Gaps
[6]	Programmable transmission and reception- Baud rate generation- FIFO storage- Error detection	Verilog HDL- Xilinx ISE	limited focus on power optimiza-tion - Implementation on older FPGA technology
[7]	Multi-channel UART controller- Asynchronous FIFO using gray code counter- Configurable baud rates	VHD- Xilinx ISE 9.2i	Limited scalability- Lack of support for high-speed data transmission
[8]	UART design and verification- UVM-based testbench - Achieved 100% functional coverage and ~ 76.4% code coverage	SystemVerilog- UVM-Cadence Xcelium	Moderate code coverage; potential for further optimization
[9]	Full-duplex communication Configurable parity and baud rate-Built-in CDC synchronizer at receiver line	Verilog HDL- Artix-7 FPGA	Limited baud rate tolerance; poten-tial issues with metastability

	Unsatisfied Issue	Why it's important	Why Often Skipped	Research Focus	Problem Being Solved	Approach	Future Solution to Unsolved Issues
[10]	Multi-Clock Domain Synchronizatio n	Ensures proper data synchroniza-tion when TX and RX have different clocks.	Adds com-plexity and difficulty in synchroniz a-tion across different clock domains.	UART Design & Implementatio n	Ensuring reliable communicatio n when TX and RX are on sepa-rate clocks.	Introduced Clock Domain Crossing (CDC) logic to handle multi-clock synchronizatio n.	**Future**: Develop advanced CDC logic to handle more complex clock domains
	Power-Aware Design	Critical for battery-powered devices	Energy optimi-zation is rarely addressed, especially in simulation-only research.	UART Design	Minimizing power consump-tion in idle or low-activity states.	Designed a low-power UART with power-saving modes during inactivity.	**Future**: Imple-ment dynamic power scaling based on data activity.
	Security	Protects	Security	UART Design	Securing	Introduced	**Future**: Focus

	Against Line Interception	against unauthorized access or spoof-ing in UART communicatio n.	rarely discussed in UART, assumed to be secure in closed systems.		UART data transmis-sion from mali-cious attacks.	potential security features like encryption for UART systems.	on integrating advanced security protocols like AES for UART systems.
[11]	Adaptive Baud Rate Adjustment	Dynamic adapta-tion to changing baud rates at runtime.	Fixed baud rate assump-tion for simplicity in design.	Energy-Efficient UART Design	Reducing power consumption while ensuring adaptability to different baud rates.	Focused on low-power UART designs without consideration for variable baud rates.	**Future**: Introduce real-time baud rate detection and automatic adaptation.
	Multi-Clock Domain Synchronizatio n	Critical for ensur-ing data integrity across multiple clock domains in real systems.	Difficult to implement, often skipped in basic UART designs.	Energy-Efficient UART Design	Ensuring proper synchronizatio n across multiple clocks.	Integrated efficient clock synchroniza-tion techniques for multi-clock UART systems.	**Future**: Create more robust clock synchronizatio n methods for multi-clock systems.
[12]	Parity & Frame Error Recovery	Real-time error detection and correction for cor-rupted frames.	Many systems only flag errors but lack error recovery mechanism s.	Full-Duplex UART Design	Handling error recovery and preventing data loss due to cor-rupted frames.	Implemented error correction and retransmis-sion strategies for robustness.	**Future**: Expand error recovery mechanisms to handle multiple error types beyond frame corruption.
	Hardware Flow Control (RTS/CTS)	Prevents data loss due to buffer overflows in high-speed or multi-device systems.	Extra hand-shaking lines and FSM over-head make it more complex.	Full-Duplex UART Design	Preventing buf-fer overflows and ensuring smooth data flow between multiple devices.	Implemented RTS/CTS hard-ware flow control to manage buffer overflow.	**Future**: Develop more efficient flow control mechanisms to support ultra-high-speed data rates.
	Security	Prevents	Security	Full-Duplex	Ensuring	Proposed	**Future**:

	Against Line Interception	unauthorized interception or spoofing of serial communication.	rarely discussed in low-level UART designs.	UART Design	secure communication by protecting against malicious interference.	encryption and security methods for secure UART communication.	Incorporate robust encryption algorithms and anti-spoofing mechanisms.
[13]	Hardware Testability & Fault Injection	Ensures UART design resilience against physical faults like stuck-at faults.	Simulation-only testing doesn't cover real-world fault conditions.	UART Design & Verification	Validating UART's reliability under fault conditions.	Introduced built-in self-test (BIST) and fault injection techniques.	**Future**: Implement more comprehensive BIST techniques with real-world fault injection.
	ECC Integration (Error Correction Codes)	Enhances error detection and correction for noisy channels.	noisy channels. Parity checking is simpler but less effective.	UART Design & Verification	Improving error resilience in UART communication.	Integrated ECC for better error correction over simple parity.	**Future**: Extend ECC to handle multi-bit errors and improve error correction.

Table (continued)

	Unsatisfied Issue	Why it's important	Why Often Skipped	Research Focus	Problem Being Solved	Approach	Future Solution to Unsolved Issues
[14]	Multi-Channel UART on One FPGA	Enables multiple peripheral communication on a single FPGA.	Simplified designs with single-channel UARTs are easier to verify.	Full-Duplex UART Design	Supporting simultaneous communication with multiple peripherals.	Designed multi-channel UART to enable parallel communication with multiple devices.	**Future**: Develop scalable multi-channel UARTs with automated channel management.
	Parity & Frame Error Recovery	Real-time error correction for corrupted data frames.	Error correction often skipped in favor of	Full-Duplex UART Design	Enhancing robustness in UART systems by ensuring error recovery.	Implemented error recovery and retransmission mechanisms in	**Future**: Include more advanced error correction

			simpler error flags.			high-speed UART.	methods like multi-level parity.
	Hardware Flow Control (RTS/CTS)	Prevents data loss due to buffer overflows in high-speed or multi-device systems.	Extra hand-shaking lines and FSM over-head make it more complex.	Full-Duplex UART Design	Preventing buf-fer overflows and ensuring smooth data flow between multiple devices.	Implemented RTS/CTS hard-ware flow control to manage buffer overflow.	**Future**: Develop more efficient flow control mechanisms to support ultra-high-speed data rates.

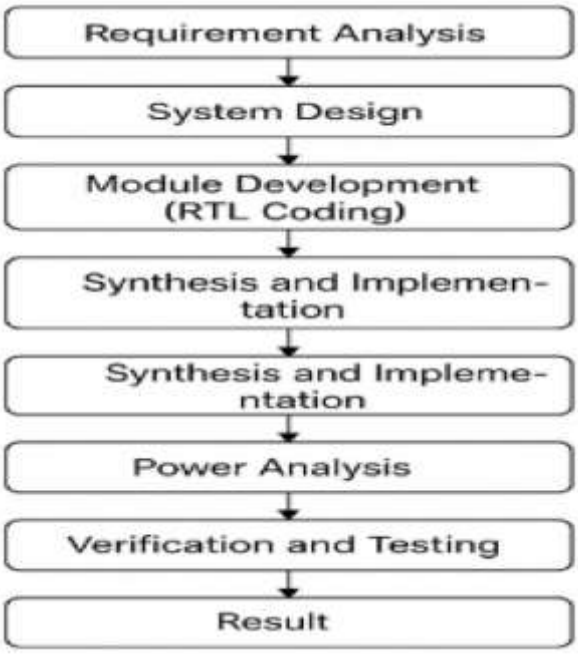

Fig. 1 Proposed workflow

Figure 3 illustrates the top-level block diagram of the complete UART system, highlighting the signal flow and interactions between modules. The FIFOs ensure smooth data buffering and synchronization between transmission and reception, while the baud generator regulates timing based on the configured divisor.

Following this, module development involves imple-menting each component in Verilog HDL. These modules are then integrated into a cohesive top-level design. In the functional simulation phase, simulation tools like Xilinx

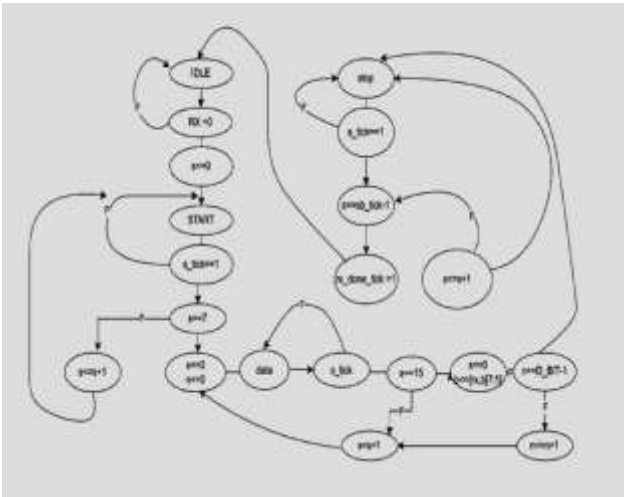

Fig. 2 UART Receiver FSM Flowchart showing state transitions based on sampled ticks, bit counters, and line status

Vivado are used to validate module behavior and full system operation. Waveform analysis is conducted

to ensure correct timing, FSM transitions, and data integrity.

Once verified, the design moves into synthesis and implementation, where it is synthesized and mapped onto the Basys 3 FPGA. Timing (e.g., Worst Negative Slack), logic utilization (LUTs, FFs, BRAM), and physical place-ment are evaluated. The power analysis phase then assesses both static and dynamic power consumption using Vivado's power estimation tools under typical conditions (e.g., 100 MHz system clock, 115200 bps baud rate).

After synthesis, verification and testing are conducted on actual hardware using loopback setups or external UART modules to validate transmission and reception of known data patterns (such as 0xA5). In the comparative analysis

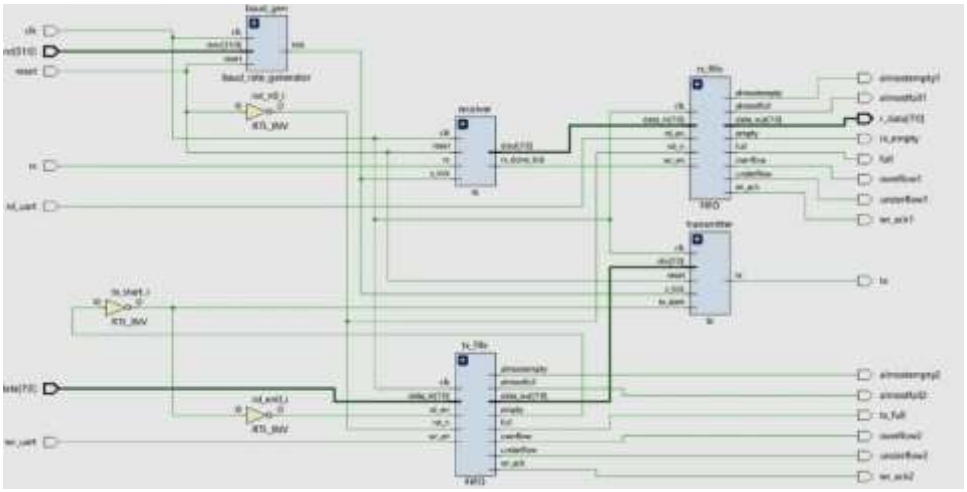

Fig. 3 UART System Block Diagram showing intercon-nected modules (baud generator, FIFOs, transmitter, receiver) and signal flow for full-duplex serial communication

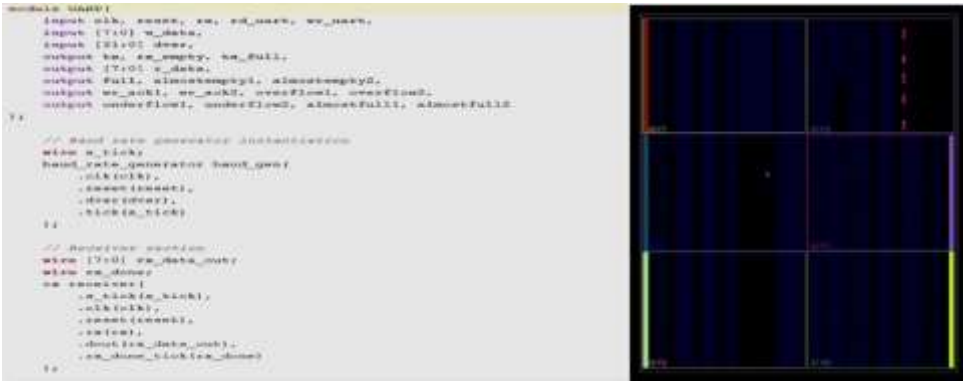

Fig. 4 Floorplan View and RTL Code Display in Vivado

phase, performance, power consumption, and accuracy are compared against previous UART implementations, high-lighting improvements like FIFO buffering and oversam-pling techniques [7]. Finally, in the result phase, all test data, timing reports, simulation waveforms, and resource usage statistics are compiled.

4 Results and simulation

The designed UART system is verified through synthesis, implementation, and simulation using the Xilinx Vivado Design Suite. The results demonstrate that the system meets its performance targets in terms of timing, power, and resource utilization, while the simulation waveforms vali-date the correct functional operation of both transmitter and receiver modules.

4.1 Floorplan, Timing, and resource utilization

Floorplan and Timing Summary.Figure 4 shows the Vivado environment during the implementation of the UART system. On the left side, the Verilog RTL code is displayed, showing the mod-ule structure and design logic used for synthesis. On the right side, the floorplan view reveals how the de-sign components—such as the transmitter, receiver, and baud rate generator—are physically placed on the Basys 3 board using the Artix-7 FPGA.

The floorplan confirms that all major modules were placed efficiently, with no routing congestion or criti-cal placement issues. The design met all timing re-quirements, achieving a Worst Negative Slack (WNS) of 4.059 ns at a 100 MHz clock frequency, indicating a stable and well-optimized implementation Fig.5.

4.2 Simulation waveforms

The functional simulation was carried out in Vivado, and the following waveforms provide visual evidence of the UART's correct operation:

Waveform 1 – Receiving Data (10100101).

Waveform 2 – Transmitting Data (10100101).

- Key Observations:

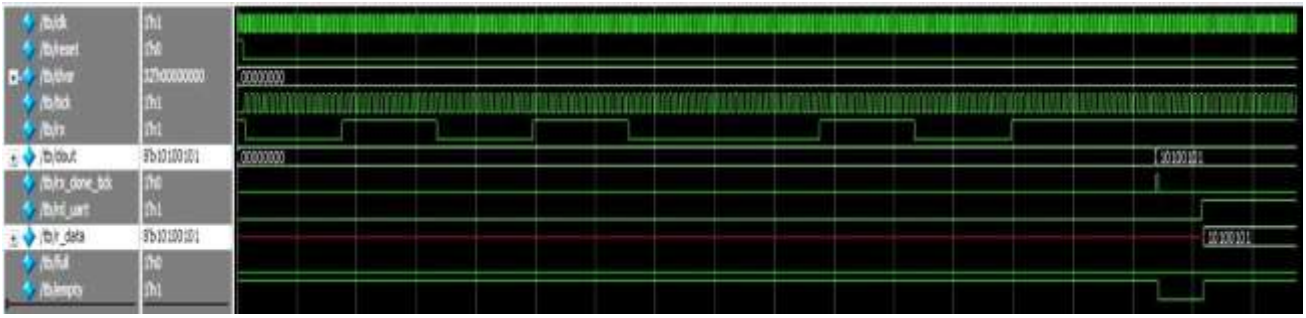

Fig. 5 Receiver capturing an 8-bit data frames with the pattern 10,100,101

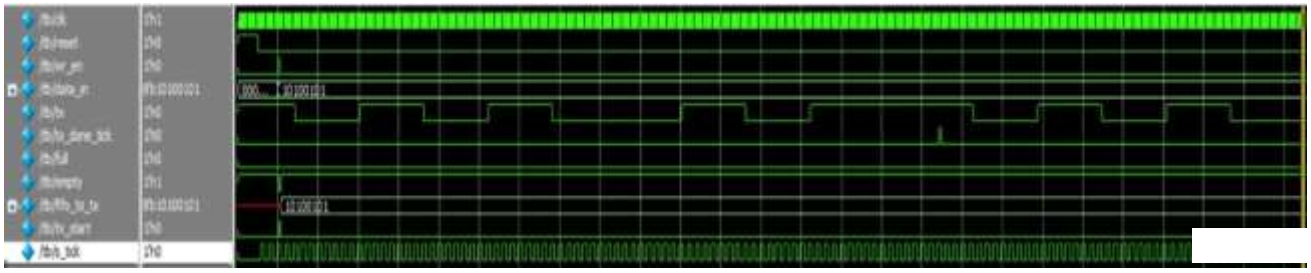

Fig. 6 shows the transmitter sending the same 8-bit pattern, 10,100,101

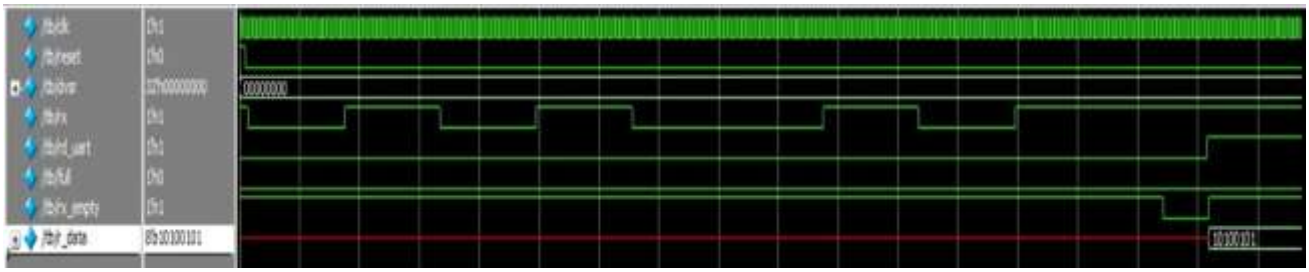

Fig. 7 confirms the consistent operation of the receiver through a secondary test, capturing the same data sequence

1 The transmitter initiates transmission with a start bit (logic LOW).
2 Data bits are shifted out sequentially, beginning with the least significant bit.
3 A stop bit (logic HIGH) concludes the transmission, returning the line to idle.

Waveform 3 – Repeated Reception Confirmation.

- **Key Observations:**

1 The oversampling mechanism stabilizes the data capture process.
2 State transitions within the receiver's finite state machine (FSM) are observed from IDLE through RECEIVING and back, ensuring accurate data recovery.

The UART design is synthesized and validated using the Vivado Design Suite. Timing and power analysis confirmed that the system meets real-time operational requirements Fig.6. The Worst Negative Slack (WNS) is measured at
4.059 ns, ensuring reliable operation at 100 MHz. The esti-mated power consumption is 45.67 mW, suitable for low-power embedded applications. FPGA resource utilization is minimal, making the design scalable for multi-module communication Fig.7. These results demonstrate that the proposed UART architecture is both timing-accurate and resource-efficient.

Validation of the proposed approach In this sub-section, Table 2 shows validation of the proposed work by compar-ing it with recent studies [6–14].

Table 2 Validation of the proposed approach

Studies	Baud Rate	FIFO Integration	Simulation Tool	Timing Accuracy/WNS	Power Usage	
[6–8]	9600 bps	Yes (Basic FIFO)	Xilinx ISE	~ 2.1 ns (Reported Max Delay)	~ 70 mW (estimated)	No advanced buffering, older FPGA platform
[9–11]	Configuration	Yes (Gray code FIFO)	Xilinx ISE 9.2i	~ 3.4 ns (Post-Place Delay)	~ 82 mW (estimated)	Multi-channel, but lacked high-speed testing
[12–14]	9600 bps	No	Cadence Xcelium	1.97 ns (Hold Slack)	~ 65 mW	Verified with UVM, limited code coverage
Proposed	115,200 bps	Yes (Tx & Rx FIFO)	Vivado	4. 059**ns**	45.67 mW	High-speed, oversampled UART

						with stable FSM

5 Conclusion and future scope

This research introduces a novel approach which details the development and verification of a UART communication module using Verilog HDL, aimed at supporting reliable and efficient asynchronous serial data exchange. The design incorporates several key features—such as FIFO buffering, finite state machine (FSM) control logic, and a baud rate generator set to 115,200 bps—to enhance system performance. A major contribution of this work is the use of 16x oversampling, which significantly improves bit-level sam-pling precision and minimizes timing jitter—an approach rarely emphasized in earlier UART implementations. The system is synthesized, implemented, and tested using the Xilinx Vivado Design Suite on the Basys 3 FPGA board. Functional simulations confirmed successful data transmission and reception with correct framing for 8-bit data (e.g., binary 10100101 or hexadecimal 0xA5). Timing analysis reported a Worst Negative Slack (WNS) of 4.059 ns, vali-dating the system's timing accuracy. The power consump-tion, estimated at around 45.67 mW, shows that the design is well-optimized for low-power applications, a parameter often neglected in prior studies.

This work uses Basys 3 (Artix-7) as a superior platform due to its energy efficiency, compatibility with Vivado, and suitability for educational and industrial applications. Vivado itself, being more advanced and beginner-friendly than ISE, offers robust support for IP-based design, timing analysis, and hardware debugging.

By combining advanced sampling techniques, modern toolchain usage, and practical power analysis, this study provides a comprehensive and resource-efficient UART solution. Future enhancements may include implementing error detection and correction, supporting variable baud rates, and expanding the design to handle multi-channel communication for more complex embedded systems.

References

1. Salem SG, Hosseny ME (2024) Signal processing implementa-tion of low-cost target speed detection of CW radar using FPGA. Int J Inf Tecnol 16:4565–4572. https://doi.org/10.1007/s41870-0 24-02033-3
2. Chakraborty R, Mondal UK, Debnath A et al (2023) Lightweight micro-architecture for IoT & FPGA security. Int J Inf Tecnol 15:3899–3905. https://doi.org/10.1007/s41870-023-01460-y
3. Bhoyar P, Sahare P, Hashmi MF et al (2024) Lightweight archi-tecture for fault detection in Simeck cryptographic algorithms on FPGA. Int J Inf Tecnol 16:337–343. https://doi.org/10.1007/s418 70-023-01593-0
4. Abirami B, Vasudevan V (2025) Temporal convolutional network (TCN) model for optimizing round robin scheduling in FPGA-based systems. Int J Inf Tecnol 17:3913–3920. https://doi.org/10.1007/s41870-025-02621-x
5. Singh AK, Mishra SP (2023) Enabling power attack on traces of FPGA based block ciphers not acquired with start of encryption. Int J Inf Tecnol 15:1367–1373. https://doi.org/10.1007/s41870-023-01178-x
6. Tayal V, Jhamb M (2023) Partial product based improved recon-figurable FIR filter with control logic for automated guided vehi-cles on virtex-7 FPGA. Int J Inf Tecnol 15:2077–2088. https://doi.org/10.1007/s41870-023-01266-y
7. Yamini R, Ramya MV (2020) Design and verification of full-duplex UART using systemverilog. Int J Eng Technol 8(3):455–460
8. Srinath M, Hiremath S (2022) UVM-based verification of UART for high reliability in

communication systems. J VLSI Des Tools Technol 12(2):99–105
9. Krushna K, Chavan et al (2024) A UVM-Verified UART design emphasizing metastability avoidance and Baud rate configura-tion. Int J Adv Comput Sci Appl 15(1):88–94
10. Rane HS, Mishra DS (2020) UART with UVM for Reliable Com-munication, International Conference on Advances in Comput-ing, Communications and Informatics (ICACCI)
11. Bhadra D, Vij VS, Stevens KS (2013) *A Low Power UART Design Based on Asynchronous Techniques*, Proc. IEEE Int. Mid-west Symp. on Circuits and Systems (MWSCAS)
12. Fang Y-Y, Chen X-J (2011) *Design and Simulation of UART Serial Communication Module Based on VHDL*, Proc. IEEE 3rd Int. Workshop on Intelligent Systems and Applications (ISA)
13. Varade SW et al (2017) *Design and Simulation of VHDL Based UART Using FSM*, Int. J. Emerging Science and Engineering (IJESE), vol. 4, no. 3, Mar
14. Wakhle GB, Aggarwal I, Gaba S (2012) *Synthesis and Implemen-tation of UART Using VHDL Codes*, Proc. IEEE Int. Symp. on Consumer Electronics, Communications and Networks (IS3C)
15. Jusoh NF, Ibrahim A, Haron MA, Sulaiman F (2019) *An FPGA Implementation of Shift Converter Block Technique on FIFO for UART*, Proc. IEEE RF and Microwave Conf. (RFM)