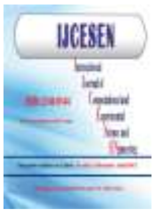
International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.3 (2025) pp. 5878-5886
http://www.ijcesen.com

ISSN: 2149-9144



Research Article

Digital System Design of FPGA – Based UART Protocol Using Verilog HDL

V Venkata Sai Raghava[1], M Ravi Kumar[2]

[1]Student, Koneru Lakshmaiah Educational Foundation, Green Fields, Vaddeswaram, Vijayawada, A.P
* **Corresponding Author Email:** vidadalaraghava2206@gmail.com - **ORCID:** 0000-0002-5247-6850

[2]Professor, Koneru Lakshmaiah Educational Foundation, Green Fields, Vaddeswaram, Vijayawada, A.P
Email: kuma2r@gmail.com- **ORCID:** 0000-0002-5247-5850

Article Info:

DOI: 10.22399/ijcesen.3306
Received : 14 January 2025
Accepted : 21 March 2025

Keywords

Baud rate,
Asynchronous Transmitter/Receiver,
Finite State Machine (FSM),
Secure communication.

Abstract:

The paper centres on the design and implementation of a Universal Asynchronous Receiver/Transmitter (UART) communication protocol to efficiently send and receive 128-bit data with the aid of Verilog HDL, with verification carried out on a Field-Programmable Gate Array (ARTIX - 7 FPGA) platform. UART is a commonly used asynchronous serial communication protocol that facilitates data exchange between two devices via a straightforward two-wire interface (TX and RX) without necessitating a clock signal for synchronisation. The design comprises two primary elements: a transmitter unit that converts input data from a parallel structure to a serial configuration, including start and stop bits for synchronisation, and a receiver unit that captures the serial data, checks its accuracy, and transforms it back into a parallel format. A baud rate generator ensures reliable data transmission by producing consistent clock pulses, thus preventing data transmission rate discrepancies.
A thorough testbench is used to validate the system, simulating a range of transmission scenarios with distinct data values, start/stop bit configurations, and error states to verify proper operation. The system is synthesized and implemented on a field-programmable gate array (ARTIX - 7 FPGA), specifically the Xilinx Artix-7, to illustrate real-time functionality. Enhancing efficiency and scalability, the UART design increases the reliability of data transmission, thereby making it a versatile choice for various embedded communication systems.

1. Introduction

In modern digital systems, communication between different components or devices is crucial for efficient data exchange. The Universal Asynchronous Receiver/Transmitter (UART) protocol is widely used for asynchronous serial communication, eliminating the need for a separate clock signal. This project focuses on designing and implementing a UART communication system using Verilog Hardware Description Language (HDL), with real-time validation on an ARTIX - 7 FPGA platform. The system comprises a transmitter module that converts parallel data into a serial stream with start and stop bits for synchronization and a receiver module that captures, verifies, and converts serial data back into parallel form. The baud rate generator ensures proper synchronization between the transmitter and receiver, maintaining stable data transmission.

The project highlights the significance of accurate timing and synchronization in serial communication and provides a foundation for exploring advanced protocols like SPI, I2C, and CAN. The UART receiver module facilitates secure communication with a processor, allowing authentication key exchange through a terminal interface like PuTTY. These authentication keys undergo validation by the processor's security mechanisms before granting access to core functionalities.
The UART interface integrates error detection mechanisms such as parity bits and a custom frame protocol to ensure communication integrity. The Verilog-based ARTIX - 7 FPGA implementation maintains minimal resource overhead while enhancing security without compromising system performance. Additionally, a digital clock displayed on a seven-segment display is implemented using ARTIX - 7 FPGA counter modules to track hours, minutes, and seconds. The PuTTY terminal

interface enables remote control and troubleshooting, making it a valuable tool for testing the system's functionality.

Figure 1 depicts the UART frame format for both serial transmission and reception scenarios. The upper part of the figure illustrates the transmission process, whereas the lower part depicts the reception process. In serial transmission, data is transmitted one bit at a time, initiated by a start bit, followed by the eight data bits (ranging from the least significant bit to the most significant bit), and concluded by a stop bit. The figure demonstrates how each bit is driven onto the transmission line at precise intervals, facilitating data transmission. On the receiving end, the process starts with the detection of the start bit, which is then followed by the sampling of each data bit at predetermined intervals to reconstruct the transmitted information. The receiver takes a sample of the data precisely at the midpoint of each bit period for the purpose of achieving high accuracy. The frame comprises a start bit, 8 data bits, and a stop bit, like a transmission. This UART frame format ensures reliable asynchronous communication by defining a clear protocol for data transmission and reception without requiring a clock signal.

Figure 2 shows the PuTTY Configuration Menu, which is used to establish serial communication between a computer and an external device via a COM port. PuTTY is a versatile terminal emulator that supports multiple connection protocols, including SSH, Telnet, and Serial communication. In this configuration, the serial line is set to COM1, which is a common default serial port on Windows-based systems, and the baud rate is set to 9600, a standard rate for UART communication. The Serial connection type is selected, indicating that the communication will be established using a direct UART interface rather than SSH or Telnet. The menu also includes options to load, save, or delete a stored session under "Saved Sessions," allowing users to retain frequently used settings for quick access. Additionally, session management features enable users to define whether the PuTTY window should close automatically upon session termination. To initiate the connection, the user clicks the "Open" button, launching the terminal interface for real-time communication with the connected device. This setup is essential for debugging embedded systems, microcontrollers, and UART-based communication protocols.

2. Literature survey

[1] Tianjun Zhan (2025) presented a Verilog HDL-based implementation of UART design, focusing on the development and optimization of UART communication systems using Verilog HDL, ensuring efficient data transmission and robust hardware implementation. By leveraging Verilog's capabilities, the research explores improved design methodologies that enhance the reliability and performance of UART interfaces. The findings contribute to ongoing advancements in digital communication, reinforcing the importance of hardware description languages in modern electronic system design. [2] Gupta et al. (2020) focused on the design and deployment of a high-speed UART, emphasizing its importance in contemporary communication systems. [3] Plugariu et al. (2019) expanded on FPGA-based high-performance computing by developing a Hadoop ZedBoard cluster with FPGA-based GZIP compression acceleration, demonstrating its efficiency in data processing applications. [4] Jeevan and Sivani (2018) explored various logic design styles to enhance high-performance VLSI decoders, which are critical in digital signal processing. [5] Koren (2018) examined advanced computer arithmetic algorithms, forming the foundation for efficient computation in digital circuits, a topic further developed by [6] Brent and Zimmermann (2010) through their exploration of key computational techniques used in modern digital systems.

[7] Gani et al. (2017) contributed to the biomedical engineering field by investigating EEG data acquisition systems utilizing FPGA ZedBoard, showcasing FPGA’s versatility beyond communication systems. [8] Nanda and Pattnaik (2016) made a significant contribution to UART research by analyzing its implementation and optimization, which has appeared in multiple publications. [9] Gopal et al. (2015) proposed a Built-In Self-Test (BIST) method for on-chip UARTs, improving fault detection in digital circuits, which is essential for increasing the reliability of integrated systems. [10] Jeevan et al. (2014) extended the application of FPGA beyond communication by implementing a secure image compression system using 2D Discrete Cosine Transform and Verilog HDL, demonstrating FPGA's role in data security and efficient image processing.

[11] Wakhle et al. (2012) provided a structured approach to hardware design by integrating UART using VHDL codes, offering a systematic methodology for designing reliable communication interfaces. [12] Fang and Chen (2011) focused on UART serial communication, designed and simulated using VHDL to ensure robust data transmission. [6] Brent and Zimmermann (2010) presented key computational techniques that contribute to modern digital systems, particularly in

arithmetic circuit design. [13] Minns and Elliott (2008) investigated the application of FSM-based digital design within Verilog HDL, providing valuable insights into structured methods for hardware development.
[14] Yu et al. (2007) addressed the challenges of managing multiple communication channels by proposing a multi-channel UART controller that integrates FIFO and FPGA, ensuring efficient data handling. [15] Norhuzaimin and Maimun (2005) tackled the complexities of high-speed UART system development, focusing on efficient data handling and maintaining signal integrity, which remains a crucial aspect of digital communication. [16] Gordon (1995) laid an early foundation in the field by examining the semantic intricacies of Verilog HDL, identifying challenges in hardware description languages and influencing subsequent developments in digital design methodologies.

3. Existing method

The existing method involves developing a communication system, incorporating both a transmitter and a receiver, and utilising Verilog for its implementation on a Field-Programmable Gate Array (FPGA) platform. The transmitter module is accountable for transmitting data via a communication channel, relying on a baud rate generator to regulate a transmission speed of 19200 baud. A baud rate generator produces precise timing signals to ensure consistent data transmission. Data is transmitted one bit at a time using a shift register or serializer. At the receiving end, the receiver module receives the incoming data, synchronizes its reception, and reassembles the original data stream. Accurate timing and bit alignment are crucial for ensuring reliable communication.
The system utilises FPGA's capacity for parallel processing, resulting in faster speeds and lower latency when compared to software-based systems traditionally employed. The FPGA-based design provides hardware reliability and scalability, making it well-suited for high-frequency communication. A terminal emulator, such as PuTTY, is employed for verifying and tracking data exchange between the sending and receiving units. PuTTY enables serial communication between the FPGA and a PC, thereby enabling real-time data transmission monitoring and interaction with the system. The Verilog implementation includes a transmitter module, which is combined with a baud rate generator that guarantees a transmission speed of exactly 19200 baud. Data transmission is managed by a state machine, and the receiver module employs a comparable process to synchronize and capture data. Verilog modules are connected to FPGA I/O pins to facilitate signal transmission, with the necessary constraints in place to guarantee co rrect operation. This approach delivers a robust solution for serial data communication, guaranteeing dependable synchronization and timing within FPGA-based communication systems. Characteristics of UART (Universal Asynchronous Receiver/Transmitter) Communication

1. This communication method does not use a synchronized clock signal between the sender and the recipient.
2. Data transmission and reception occur simultaneously in Full Duplex systems.
3. Transmission speed for Baud Rate Control can be customised to specific values such as 9600 or 19200.
4. Data transmission is initiated and terminated by the Start and Stop Bits.
5. The Parity Bit (optional) serves as an error-checking feature.
6. A Data Frame usually consists of 8-bit data which can be tailored to suit specific application requirements.

This project provides practical experience in designing digital communication systems and implementing them using FPGAs. A UART gives a basic understanding of serial communication, acting as a stepping stone for mastering more intricate communication protocols. This further illustrates the hands-on implementation of Verilog HDL in addressing real-world communication difficulties, thereby strengthening core concepts in digital design and FPGA-based system development.

4. Proposed method

We Proposed an upgraded UART-based communication system featuring adjustable parameters that enable the transmission of a variable numbers of bits (n) between a primary device and a secondary device. The system operates at a predetermined baud rate, enabling seamless synchronization of data incorporation of crucial parameters, which in turn facilitates the transmission of both small and large data packages as required by applications.
The system's flexibility allows for scalability, making it adaptable to a range of communication requirements, such as rapid and large-scale data transfers. Parameterization streamlines the overall system architecture, maximizing resource efficiency as data widths increase. The proposed method facilitates the rapid transfer of 128-bit data between the master and slave, thereby enhancing

both speed and dependability. This advanced UART system delivers a flexible and effective solution for contemporary communication applications through its support of multiple configurations and rigorous data integrity measures.

V. Block Diagram & Methodology

Figure 3 illustrates the block diagram of the UART (Universal Asynchronous Receiver-Transmitter) protocol, which facilitates a synchronous serial communication system between a master (transmitter) and a slave (receiver), operating based on a predefined baud rate. The BAUD_RATE block generates a clock signal that ensures both the master and slave operate at the same transmission speed, preventing data loss or misinterpretation. The master is responsible for sending data using a Parallel-In Serial-Out (PISO) shift register, which converts parallel data into a serial stream. This serial data is transmitted via the DATA_TX line to the slave, which then converts it back into parallel data using a Serial - In Parallel - Out (SIPO) shift register. The RESET signal ensures proper initialization, resetting both master and slave to a known state before communication begins.

The shift registers (PISO and SIPO) facilitate seamless serial – to - parallel and parallel – to - serial conversions, enabling effective communication between digital systems. The master initiates and controls the data flow, while the slave passively receives and processes the transmitted information. This method of communication is widely implemented in embedded systems, microcontroller-based designs, and real-time data transmission applications, ensuring reliable and synchronized data exchange between interconnected devices.

Figure 4 represents the flowchart of receiving data through the UART (Universal Asynchronous Receiver-Transmitter) data reception process, ensuring proper synchronization, baud rate generation, and data validation. The process begins with the Start step, which initializes the UART reception system. Before processing, the incoming serial data is stored in the Synchronization Register (rs232_rx) to ensure proper alignment with the system clock. This synchronization is crucial for accurately detecting and decoding the transmitted bits. The next step involves monitoring the input signal to detect the start bit. This is done using a negative edge trigger (negedge), as UART communication begins with a low (0) bit. Detecting this falling edge signals the receiver to prepare for incoming data. This mechanism ensures that the system correctly identifies the beginning of a valid data frame.

Once the start bit is detected, the UART state is updated (uart_start) to indicate that the system has transitioned into the data reception phase. At this point, the baud rate clock (bps_clk) is generated, ensuring that data is received at precisely defined intervals based on the communication speed. This baud rate synchronization is essential for properly sampling each bit at the correct moment. As the transmission progresses, the system detects and processes an entire data frame, including all transmitted bits. Upon successful reception, the system identifies that a complete frame has been detected, marking the transition to the data verification phase. To ensure integrity, the received data, including any parity or check bit, is then read and stored (r_data_bit) for further validation.

To maintain proper synchronization, the baud rate indexer is updated, ensuring that each subsequent data bit aligns with the system's clock and timing requirements. The receiver then sets the correct baud rate and baud rate frequency division (Bps_out and Bps_DR), allowing accurate data sampling and ensuring that communication remains stable throughout the reception process. Once the received data has been validated and processed, the system accepts and outputs the correct data. If additional frames are expected, the process loops back to detect the next start bit, allowing continuous reception. Otherwise, if no further data is received, the process terminates, completing the reception cycle.

This flow ensures reliable, error-free communication in UART-based systems by properly handling synchronization, clock generation, and data validation. The methodology is widely used in embedded systems, microcontrollers, serial communication interfaces, and industrial automation where UART-based data transfer is a fundamental requirement. The image in Fig5, the waveform provides a detailed view of the UART signals and their values over time, which can be useful for debugging and analyzing the UART design. The image features a timeline at the top, which displays a time scale measured in microseconds, covering a period of 0 to 40 microseconds. The diagram appears to depict a block representation of a UART design, illustrating the connections between the transmitter, receiver, and RS-232 interface signals, encompassing input/output data, control flags, and timing signals. Fig7 appears to be a power estimation summary for a UART (Universal Asynchronous Receiver-Transmitter) design, showing the total on-chip power of 1.899W, with a breakdown of the power consumption by different components such as dynamic, signals, logic, and I/O. The design also includes information on junction temperature, thermal margin, and power delivered to off-chip components. Fig8 shows the timing analysis for the unconstrained paths of a UART (Universal Asynchronous Receiver-Transmitter) design. The statement offers information on name, slack, levels, routes, high fanout, source and destination registers, total delay, and logic delay for numerous paths. Fig9 shows the hierarchy and resource utilization of the UART (Universal Asynchronous Receiver-Transmitter) design. The key area-related information is, the design utilizes 436 Slice LUTs, 352 Slice Registers, 9 F7 Muxes, 260 Bonded IOBs, and 1 BUFGCTRL. The Rx (Rs232_Rx_nbytes) module uses 319 Slice LUTs and 295 Slice Registers, while the tx (Rs232_Tx_nbytes) module uses 117 Slice LUTs and 57 Slice Registers.

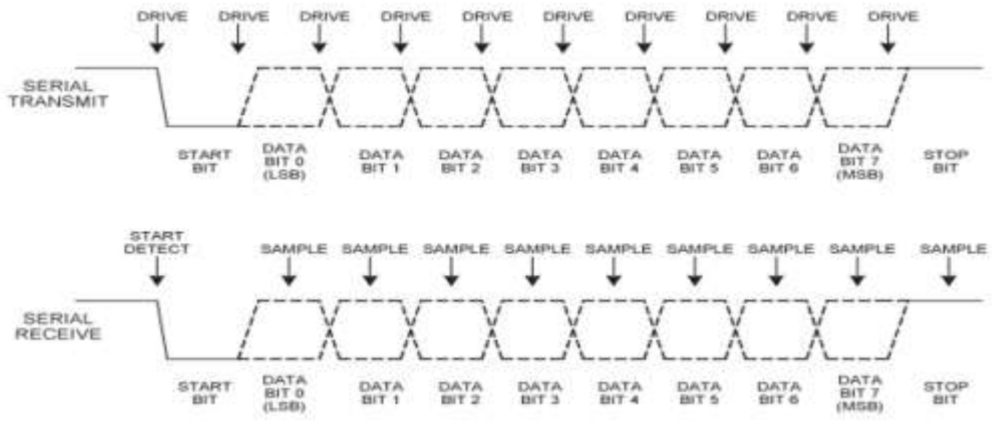
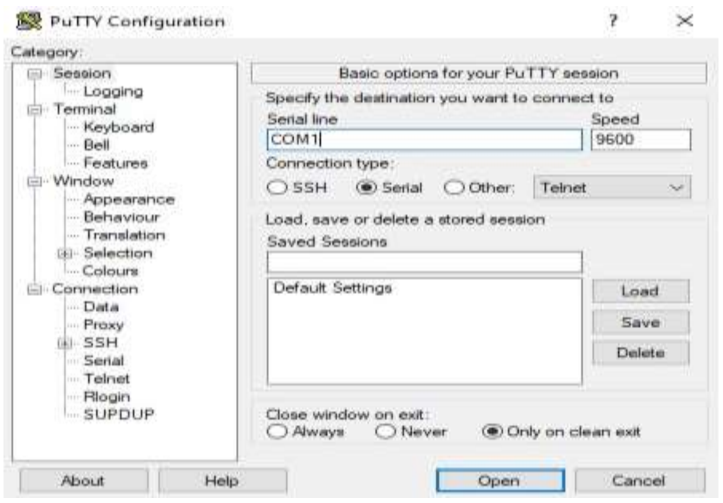


Figure 1: *UART frame format for both transmitter and receiver*



Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM1

Speed: 9600

Connection type: SSH [✓ Serial] Other: Telnet

Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

Figure 2: *Putty Configuration Menu*

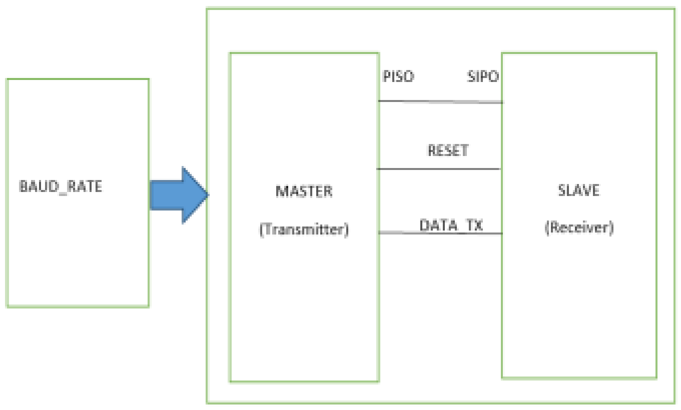


Figure 3: *- Block Diagram of UART Protocol*

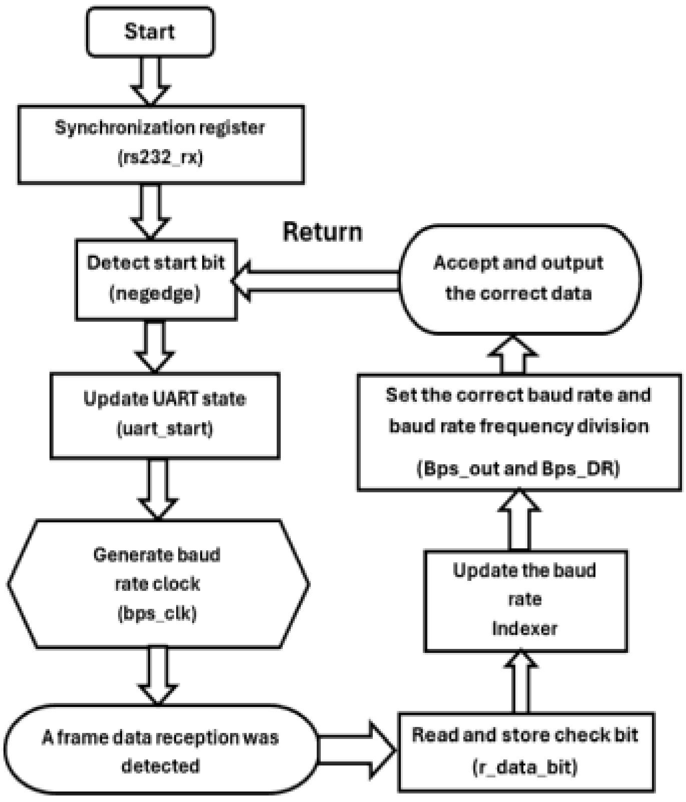
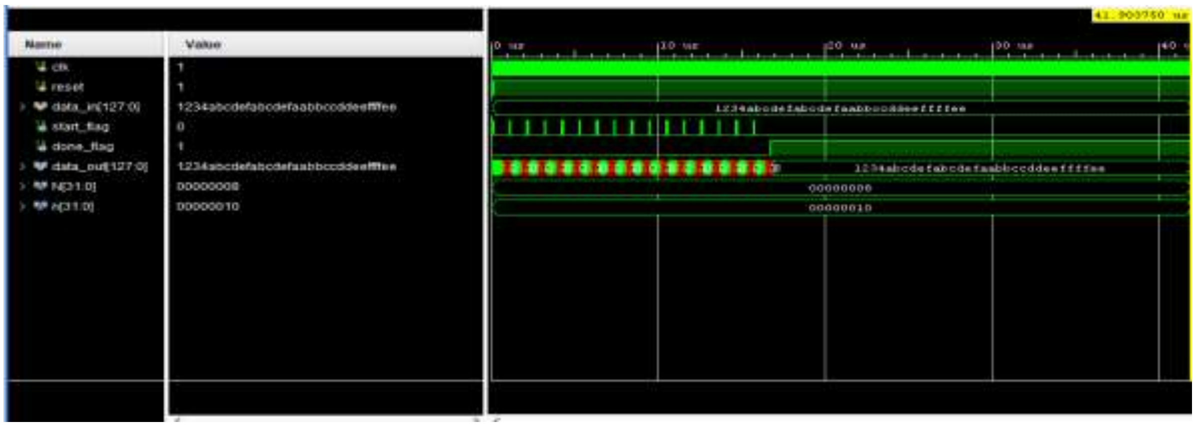

Start
Synchronization register (rs232_rx)
Detect start bit (negedge)
Update UART state (uart_start)
Generate baud rate clock (bps_clk)
A frame data reception was detected
Read and store check bit (r_data_bit)
Update the baud rate Indexer
Set the correct baud rate and baud rate frequency division (Bps_out and Bps_DR)
Accept and output the correct data
Return

Figure 4: *- Flowchart for Receiving data through a UART*



Name	Value
clk	1
reset	1
data_in[127:0]	1234abcdefabcdefaabbccddeeffffee
start_flag	0
done_flag	1
data_out[127:0]	1234abcdefabcdefaabbccddeeffffee
N[31:0]	00000008
n[31:0]	00000010

Figure 5: *UART - Simulation Waveform for Extension Paper*

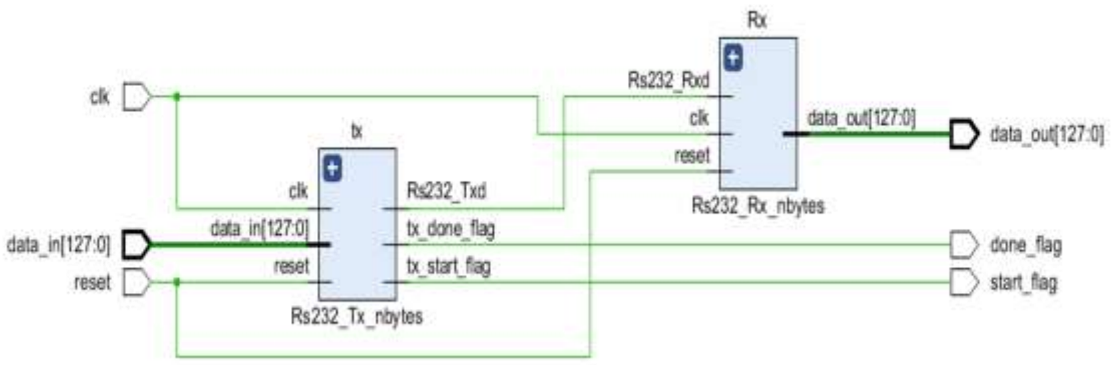
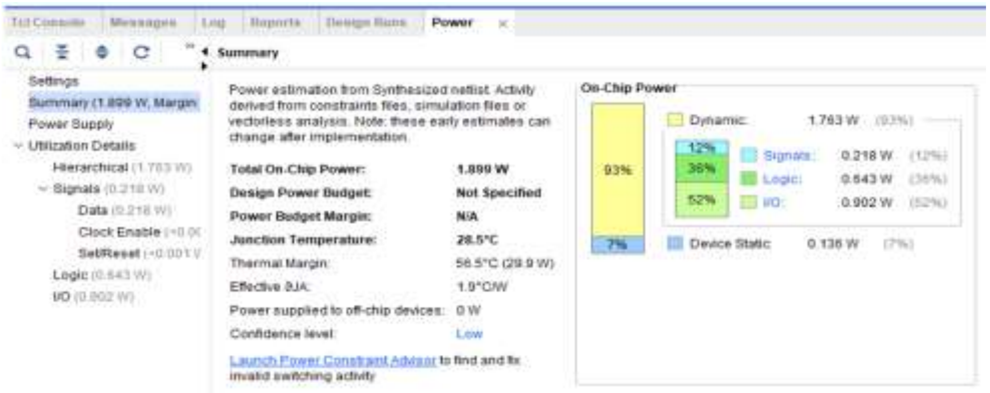


Figure 6: *Schematic of UART for Extension Paper*



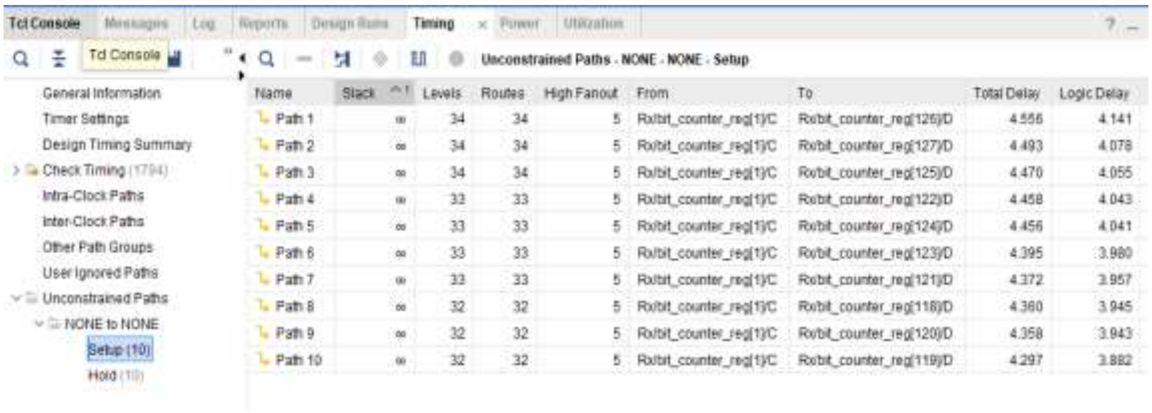
Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	1.899 W
Design Power Budget:	Not Specified
Power Budget Margin:	N/A
Junction Temperature:	28.5°C
Thermal Margin:	56.5°C (29.9 W)
Effective ϑJA:	1.9°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

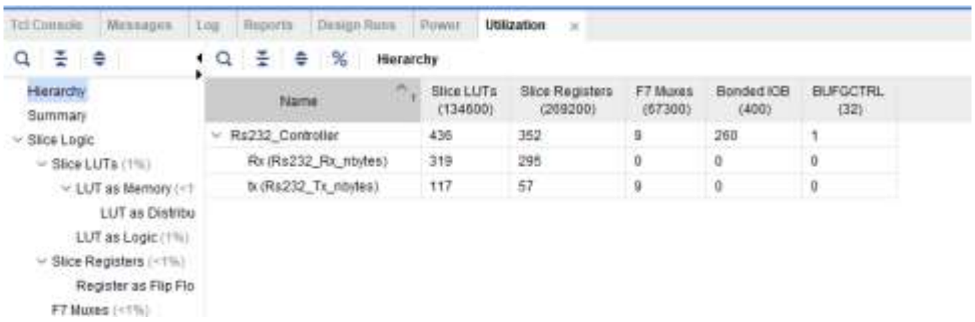
Dynamic:	1.763 W	(93%)
Signals:	0.218 W	(12%)
Logic:	0.643 W	(36%)
I/O:	0.902 W	(52%)
Device Static:	0.136 W	(7%)

Figure 7: Power of UART



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay
Path 1	∞	34	34	5	Rx/bit_counter_reg[1]/C	Rx/bit_counter_reg[126]/D	4.556	4.141
Path 2	∞	34	34	5	Rx/bit_counter_reg[1]/C	Rx/bit_counter_reg[127]/D	4.493	4.078
Path 3	∞	34	34	5	Rx/bit_counter_reg[1]/C	Rx/bit_counter_reg[125]/D	4.470	4.055
Path 4	∞	33	33	5	Rx/bit_counter_reg[1]/C	Rx/bit_counter_reg[122]/D	4.458	4.043
Path 5	∞	33	33	5	Rx/bit_counter_reg[1]/C	Rx/bit_counter_reg[124]/D	4.456	4.041
Path 6	∞	33	33	5	Rx/bit_counter_reg[1]/C	Rx/bit_counter_reg[123]/D	4.395	3.980
Path 7	∞	33	33	5	Rx/bit_counter_reg[1]/C	Rx/bit_counter_reg[121]/D	4.372	3.957
Path 8	∞	32	32	5	Rx/bit_counter_reg[1]/C	Rx/bit_counter_reg[118]/D	4.360	3.945
Path 9	∞	32	32	5	Rx/bit_counter_reg[1]/C	Rx/bit_counter_reg[120]/D	4.358	3.943
Path 10	∞	32	32	5	Rx/bit_counter_reg[1]/C	Rx/bit_counter_reg[119]/D	4.297	3.882

Figure 8: Delay of UART



Name	Slice LUTs (134600)	Slice Registers (269200)	F7 Muxes (67300)	Bonded IOB (400)	BUFGCTRL (32)
Rs232_Controller	436	352	9	260	1
Rx (Rs232_Rx_nbytes)	319	295	0	0	0
tx (Rs232_Tx_nbytes)	117	57	9	0	0

Figure 9: Area of UART

4. Conclusions

In this section conclusions of work should be given A Universal Asynchronous Receiver/Transmitter (UART) communication protocol was successfully designed and implemented using Verilog HDL, with real-time validation carried out on an FPGA platform. The system was composed of two key modules: a transmitter for converting parallel data into serial form with proper framing, and a receiver for capturing the serial data, verifying its integrity, and converting it back to parallel form. A baud rate generator operating at a predefined speed baud ensured accurate timing and synchronization between the transmitter and receiver, enabling reliable data transmission. The implementation demonstrated the feasibility of UART communication for efficient data transfer between devices, successfully transferring 128 bits of data from the master to the slave. With the designed system's successful validation on FPGA hardware, this UART communication protocol proves to be a robust and adaptable solution for a variety of real-time applications. The system offers flexibility for both simple and complex data transmission needs, providing a foundation for future enhancements such as higher data rates and multi-bit configurations, further improving the scalability and performance of the UART communication system.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1]T. Zhan, "Verilog HDL-based implementation of UART design," International Conference on Physics, Photonics, and Optical Engineering (ICPPOE 2024), 7 March 2025.

[2]A. K. A. R. N. K. a. R. R. Gupta, "”Design and implementation of high-speed universal asynchronous re- ceiver and transmitter (UART).”," 7TH International Conference on Signal Processing and Integrated Networks (SPIN), vol. IEEE, 2020.

[3]O. L. P. R. P. a. R. H. Plugariu, "Hadoop ZedBoard cluster with GZIP compression FPGAacceleration," 11th International Conference on Electronics, Computers and ArtificialIntelligence(ECAI), vol. IEEE, pp. pp.1- 5, 2019.

[4]B. a. K. S. Jeevan, "A Review on different Logic Styles to design High Performance VLSI Decoders.," International Conference on Networking, Embedded and Wireless Systems (ICNEWS), vol. IEEE, pp. pp. 1-6, 2018.

[5]I. C. a. a. A. P. P. 2. Koren, "Computer arithmetic algorithms," vol. AK Peters/CRC Press, 2018.

[6]R. P. a. P. Z. Brent, "Modern computerarithmetic," no. Cambridge University Press, p. Vol. 18, 2010.

[7]H. S. S. K. W. a. Z. T. L. O. H. Gani, "Development of EEG Data Acquisition System Based on FPGA Zedboard," In 2017 5th International Conference on Instru- mentation, Communications, Information Technology, and Biomedical Engineering(ICICI-BME), vol. IEEE, pp. pp.1-5, 2017.

[8]U. a. S. K. P. Nanda, "Universal asyn- chronous receiver and transmitter (uart)," 3rd international con- ference on advanced computing and communication systems (ICACCS), vol. IEEE, pp. vol.1,pp.1-5, 2016.

[9]P. B. K. H. K. a. B. P. K. Gopal, "An FPGA Implementation of On Chip UART Testing with BIST Techniques.," International Journal of Applied Engineering Research, Vols. ISSN (2015): 0973-4562, 2015.

[10]B. C. N. B. C. V. K. a. K. S. Jeevan, "FPGA im- plementation of secure image compression with 2DDCT using Verilog HDL," 2nd International Conference on Devices, Circuits and Systems(ICDCS), vol. IEEE, pp. pp.1- 4, 2014.

[11]G. B. I. A. a. S. G. Wakhle, "Syn- thesis and Implementation of UART using VHDL Codes," International Symposium on Computer, Consumer and Control, vol. IEEE, pp. pp. 1-3, 2012.

[12]Y.-y. a. X.-j. C. Fang, "Design and simulation of UART se- rial communication module based on VHDL," 3rd International Workshop on Intelligent Systems and Applications, no. IEEE, pp. pp. 1-4, 2011.

[13]P. D. a. I. E. L. Minns, "FSM-based digital design using Verilog HD," 2008.

[14]S. L. Y. W. C. a. Z. W. Yu, "Implementation of a Multi-channel UART Controller Based on FIFO Technique and FPGA," 2nd IEEE Conference on Industrial Electronics and Applications, vol. IEEE, pp. pp. 2633-2638, 2007.

[15]J. a. H. H. M. Norhuzaimin, "The design of high speed UART," Asia-Pacific Conference on Applied Electromagnetics, vol. IEEE, pp. 5-pp, 2005.

[16]M. Gordon, "The semantic challenge of Verilog HDL.," In Proceed- ings of tenth annual IEEE symposium on logic in computer science, vol. IEEE, pp. pp. 136-145, 1995.

[17]A. (Intel), " UART Megafunction User Guide," Altera Corporation, 2011.

[18]R. &. Y. R. Ali, "Low Power UART Design Based on FPGA Implementation," 2010.

[19]W. e. a. Al Khateeb, "Design and Implementation of UART Serial Communication on FPGA using VHDL," 2014.

[20]V. &. M. M. Amarnath, "Design of Low Power UART with Adaptive Baud Rate Generator," International Journal of Computer Applications, 2016.

[21]M. &. Y. A. Ayub, "Design and FPGA implementation of UART with automatic baud rate detection," 2015.

[22]S. Banerjee, " Use Verilog to Create Practical Digital Designs," IEEE, 2020.

[23]J. Bhaskar, The Verilog HDL Primer (3rd ed.), BS Publications, 2005.

[24]G. &. R. D. B. Gajjar, "Design and Implementation of UART on FPGA," 2011.

[25]E. &. A. K. M. Habib, "Design and Implementation of Parameterized UART Transmitter for FPGA Applications," 2013.

[26]X. Inc., "Vivado Design Suite User Guide: Synthesis (UG901)," Xilinx Inc., 2020.

[27]S. &. B. P. Kaur, "A review paper on the UART protocol," 2014.

[28]A. &. S. S. Kumari, "Design and Simulation of UART Protocol Based on Verilog HDL," 2014.

[29]Z. &. H. J. Liu, Semantics of Programming Languages: Structures and Techniques, Springer, 2009.

[30]H. L. Y. &. C. L. Zhang, "Design of a UART Communication Module Based on FPGA," 2014.
[31]A. &. S. P. Maiti, "Improving the Robustness of Ring Oscillator Based True Random Number Generators," 2010.
[32]M. M. &. C. M. D. Mano, Digital Design: An Introduction to Verilog HDL (5th ed.), Pearson, 2013.
[33]S. Palnitkar, Verilog HDL: A Guide to Digital Design and Synthesis (2nd ed.), Pearson Education, 2003.
[34]P. Pong, Digital System Design with FPGA: Implementation Using Verilog and VHDL, Springer, 2021.
[35]J. M. C. A. &. N. B. Rabaey, Digital Integrated Circuits: A Design Perspective (2nd ed.), Pearson Education, 2003.
[36]S. R. &. S. S. Sahu, "Design and Implementation of UART Protocol Based on FPGA," International Journal of Advanced Research in Electrical, Electronics, and Instrumentation Engineering, 2016.
[37]Y. &. Z. M. Tang, "Low Power Design Techniques for FPGA Based UART Controllers," 2014.
[38]D. E. &. M. P. R. Thomas, Verilog Hardware Description Language (5th ed.), Springer Science and Business Media, 2002.
[39]S. M. Trimberger, "Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology," 2015.
[40]F. &. G. A. Vahid, "Embedded System Design: A Unified Hardware/Software Introduction," 2001.
[41]J. F. Wakerly, Digital Design: Principles and Practices (5th ed.), Pearson Education, 2018.
[42]G. B. A. I. &. G. S. Wakhle, "Synthesis and Implementation of UART using VHDL Codes," 2012.
[43]F. &. J. Z. Wang, "FPGA Based UART Design and Simulation," 2010.
[44]W. Wolf, "FPGA Based System Design," 2004.
[45]K. C. &. L. C. N. Young, " Implementation of Reconfigurable UART Controller Using FPGA," International Journal of Engineering Research and Technology (IJERT), 2013.
[46]S. Y. L. C. W. &. W. Z. Yu, "Implementation of a Multi-channel UART Controller Based on FIFO Technique and FPGA," 2007.
[47]L. &. Y. G. Zhang, "FPGA based UART Implementation and Verification," 2012.