DESIGN AND IMPLEMENTATION OF UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER

Anusha Sunkaraboina  
Electronics and Communication Engineering  
National Institute of Technology Warangal India  
[sa21ecb0b59@student.nitw.ac.in](mailto:sa21ecb0b59@student.nitw.ac.in)

Tanneru Bhavya Electronics and Communication Engineering  
National Institute of Technology Warangal India  
[tb21ecb0b60@student.nitw.ac.in](mailto:tb21ecb0b60@student.nitw.ac.in)

Somaraju Sai Rohith  
Electronics and Communication Engineering  
National Institute of Technology Warangal India  
[ss21ecb0b58@student.nitw.ac.in](mailto:ss21ecb0b58@student.nitw.ac.in)

*Abstract*— Universal Asynchronous Receiver Transmitter (UART) is a communication protocol commonly used for serial data communication. Our project presents the design and implementation method of a Universal Asynchronous Receiver Transmitter (UART) using Verilog HDL. It is a semiconductor chip that governs a computer's functionality to something like a serial computer connected to that though. Throughout opposed versus parallel communication, serial communication is substantially more cost-effective although the system's complexity increases. For something like the design of either a UART that is performed in Verilog HDL, it may be quickly racially segregated upon an FPGA to achieve the highest level of data reliability as well as blunder data.

Keywords—UART

# *Introduction*

UART, or universal asynchronous receiver-transmitter, is one of the most used device-to-device communication protocols. This article shows how to use UART as a hardware communication protocol by following the standard procedure. When properly configured, UART can work with many different types of serial protocols that involve transmitting and receiving serial data. In serial communication, data is transferred bit by bit using a single line or wire. In two-way communication, we use two wires for successful serial data transfer. Depending on the application and system requirements, serial communications needs less circuitry and wires, which reduces the cost of implementation.

# *LITERATURE REVIEW*

## A. SERIAL COMMUNICATION

Communication protocol plays a big role in organizing communication between devices. It is designed in different ways based on system requirements, and these protocols have a specific rule agreed upon between devices to achieve successful communication.

Embedded systems, microcontrollers, and computers mostly use UART as a form of device-to-device hardware communication protocol. Among the available communication protocols, UART uses only two wires for its transmitting and receiving ends. Despite being a widely used method of hardware communication protocol, it is not fully optimized all the time. Proper implementation of frame protocol is commonly disregarded when using the UART module inside the microcontroller. By definition, UART is a hardware communication protocol that uses asynchronous serial communication with configurable speed. Asynchronousmeans there is no clock signal to synchronize the output bits from the transmitting device going to the receiving end.

*B. Interface*

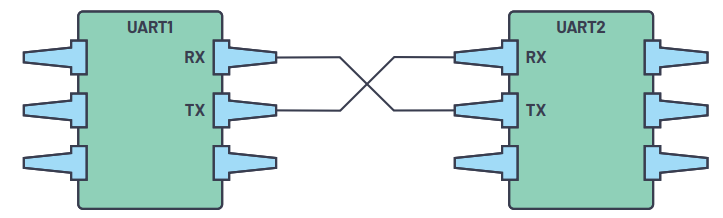


Figure1.Two UARTs directly communicate with each other

The two signals of each UART device are named:

* Transmitter (Tx)
* Receiver (Rx)

The main purpose of a transmitter and receiver line for each device is to transmit and receive serial data intended for serial communication.

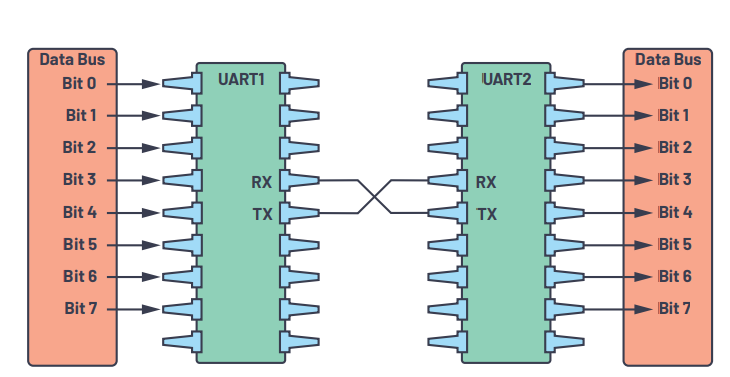
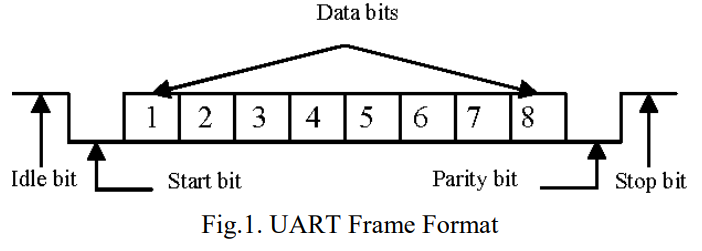


Figure 2. UART with data bus

The transmitting UART is connected to a controlling data bus that sends data in a parallel form. From this, the data will now be transmitted on the transmission line (wire) serially, bit by bit, to the receiving UART. This, in turn, will convert the serial data into parallel for the receiving device. The UART lines serve as the communication medium to transmit and receive one data to another. Take note that a UART device has a transmit and receive pin dedicated for either transmitting or receiving. For UART and most serial communications, the baud rate needs to be set the same on both the transmitting and receiving device. The baud rate is the rate at which information is transferred to a communication channel. In the serial port context, the set baud rate will serve as the maximum number of bits per second to be transferred.

# *Working function and operational states process of the UART*



*A.IDLE*

The equipment enters the above state because when reset is asserted. It's also the default condition. Two activities can take place throughout this state. The information of transmitter data register being imported in and out of transmitter shift register, which seems to have ten bits; the begin bit is LSB, the end bit is MSB, and the middle eight bits are data bits. State transition from state idle to state waiting.

## B.Data Transmission

In UART, the mode of transmission is in the form of a packet. The piece that connects the transmitter and receiver includes the creation of serial packets and controls those physical hardware lines. A packet consists of a start bit, data frame, a parity bit, and stop bits.

*C.* *Start Bit*

The UART data transmission line is normally held at a high voltage level when it’s not transmitting data. To start the transfer of data, the transmitting UART pulls the transmission line from high to low for one clock cycle. When the receiving UART detects the high to low voltage transition, it begins reading the bits in the data frame at the frequency of the baud rate*.*

**

*D. Data Frame*

The data frame contains the actual data being transferred. It can be five bits up to eight bits long if a parity bit is used. If no parity bit is used, the data frame can be nine bits long. In most cases, the data is sent with the least significant bit first.

**

*E. Parity Bit*

Parity describes the evenness or oddness of a number. The parity bit is a way for the receiving UART to tell if any data has changed during transmission. Bits can be changed by electromagnetic radiation, mismatched baud rates, or long distance data transfers.

After the receiving UART reads the data frame, it counts the number of bits with a value of 1 and checks if the total is an even or odd number. If the parity bit is a 0 (even parity), the 1 or logic-high bit in the data frame should total to an even number. If the parity bit is a 1 (odd parity), the 1 bit or logic highs in the data frame should total to an odd number.

When the parity bit matches the data, the UART knows that the transmission was free of errors. But if the parity bit is a 0, and the total is odd, or the parity bit is a 1, and the total is even, the UART knows that bits in the data frame have changed.



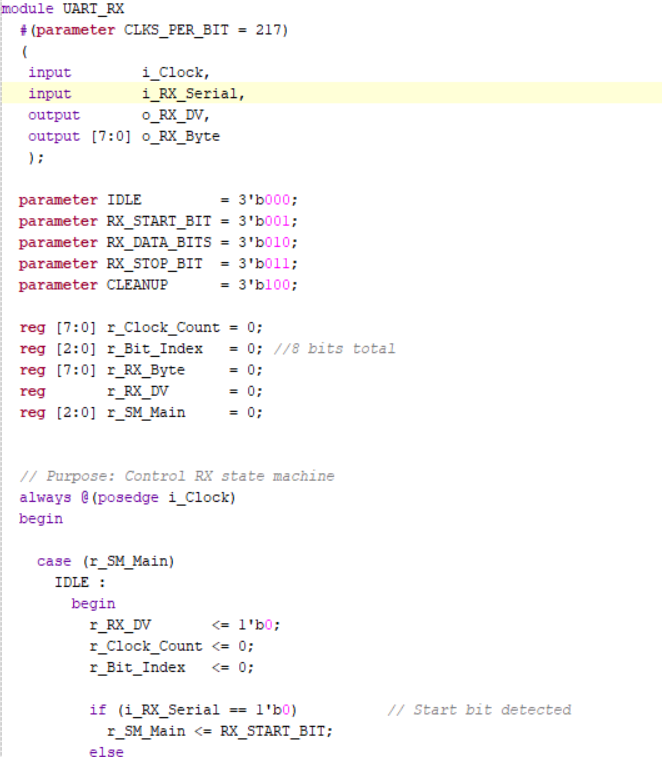
*F. Stop Bit*

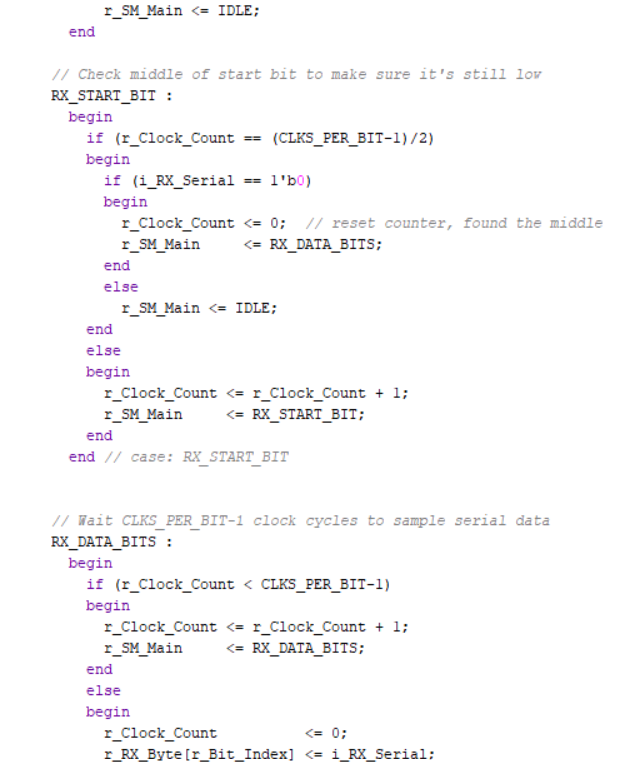
To signal the end of the data packet, the sending UART drives the data transmission line from a low voltage to a high voltage for one to two bit(s) duration.

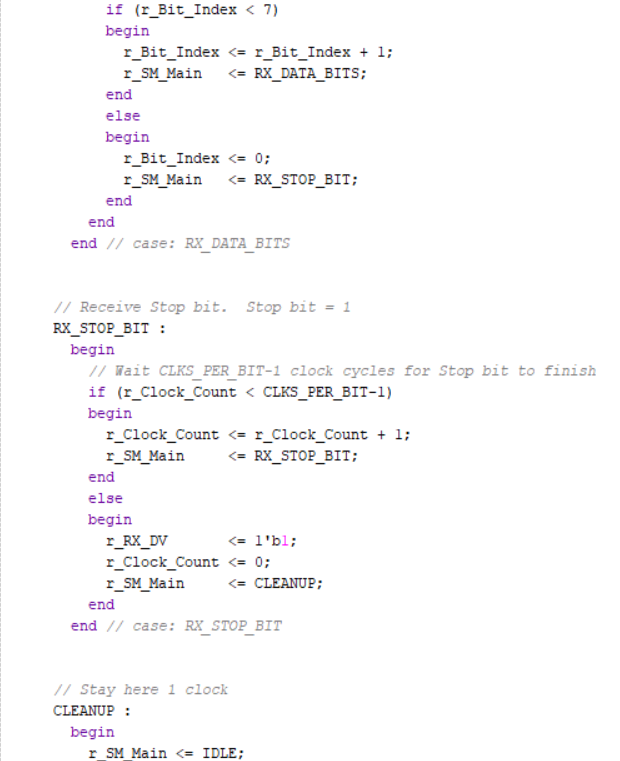


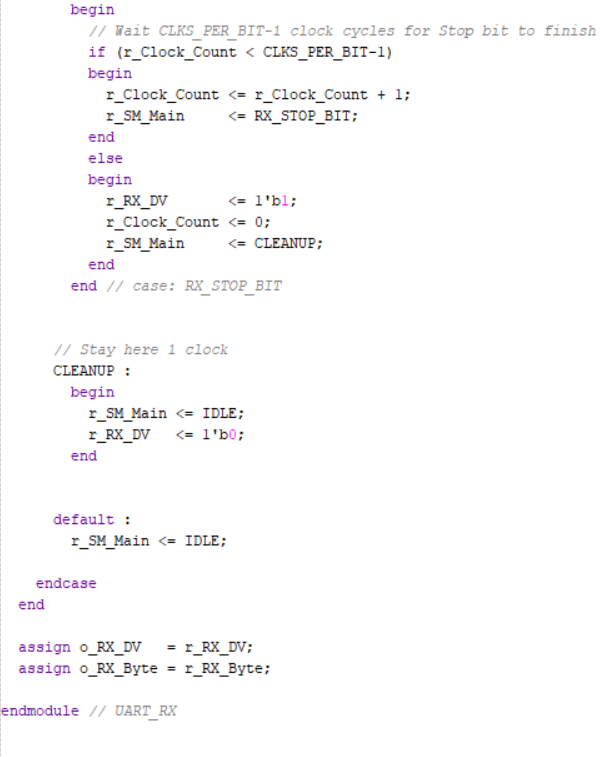
#### 4.Code

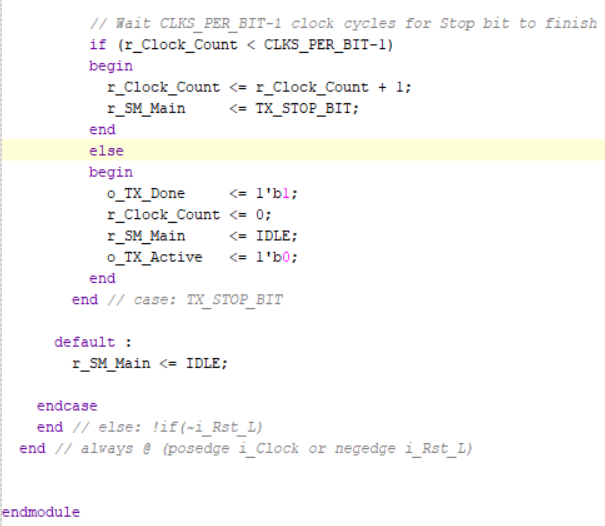
Receiver module



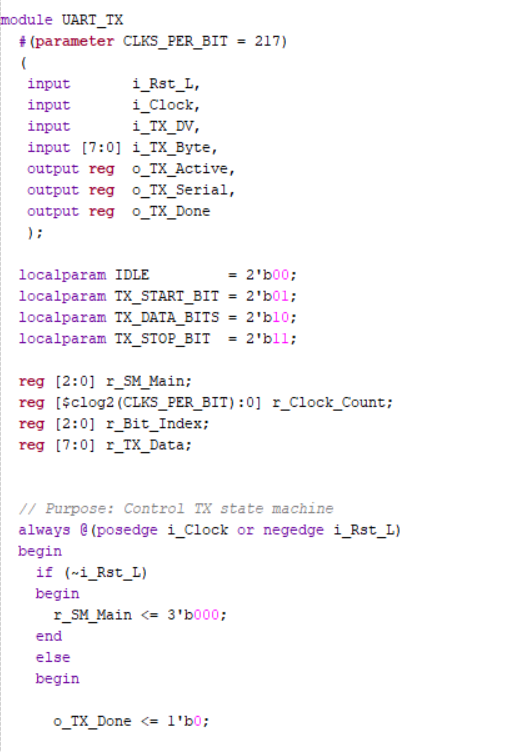


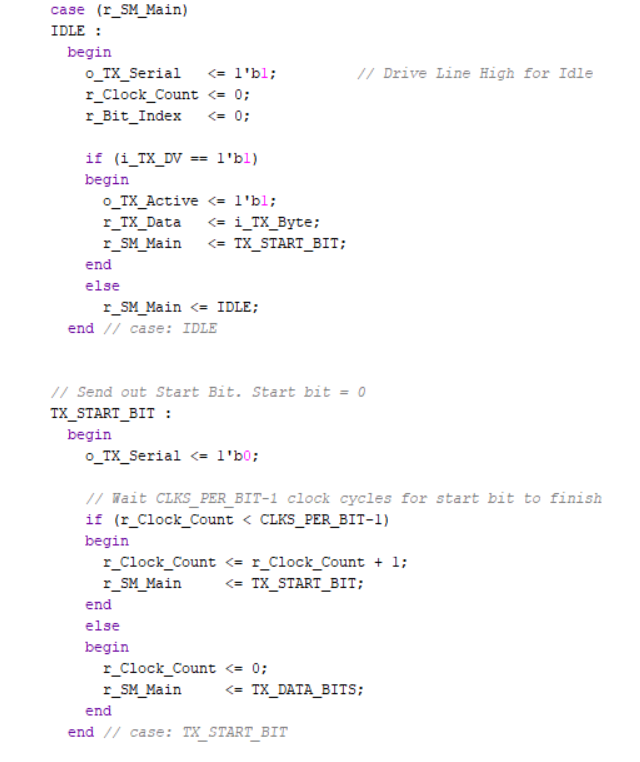


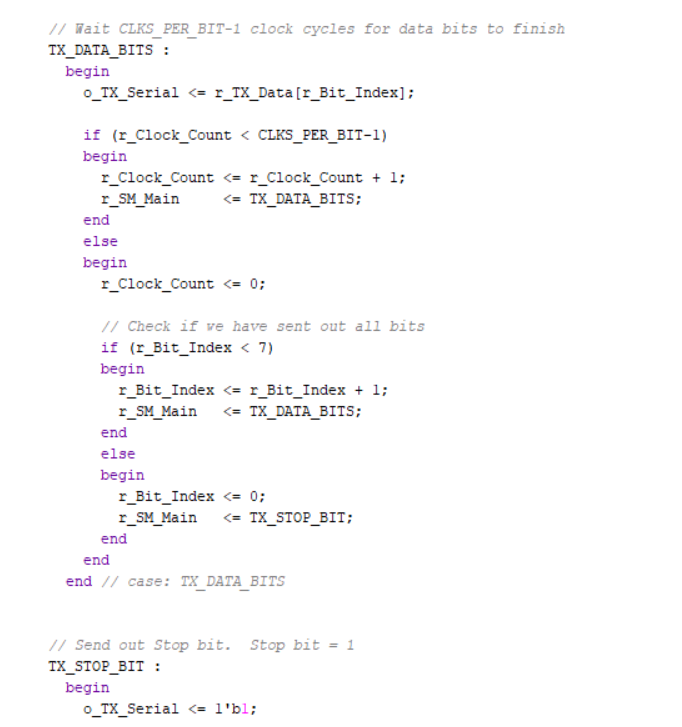




Transmitter Module

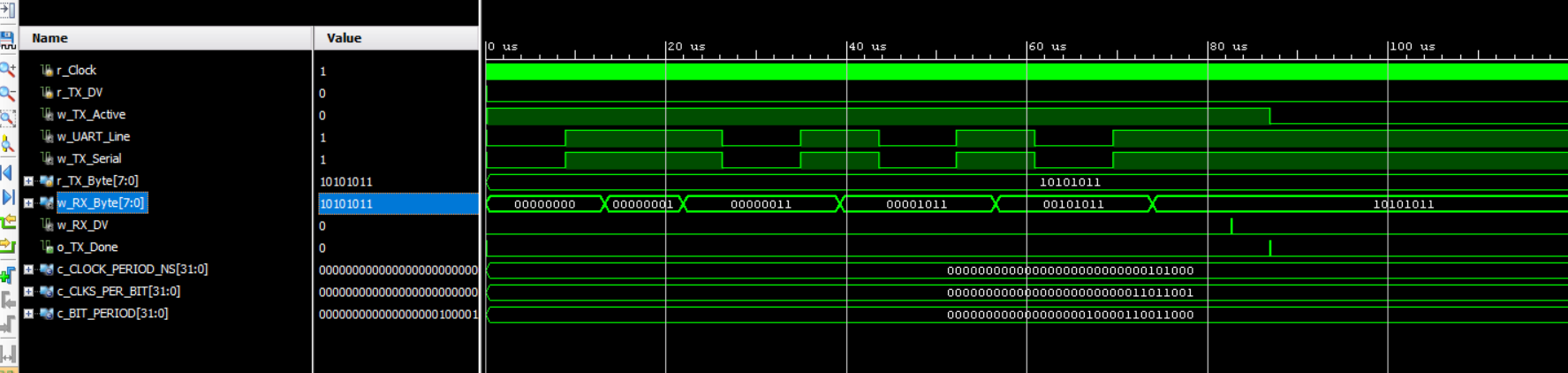




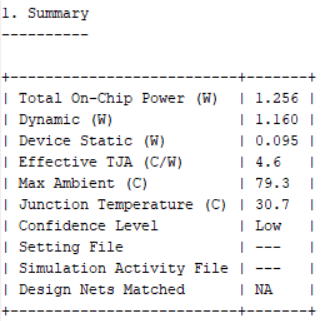


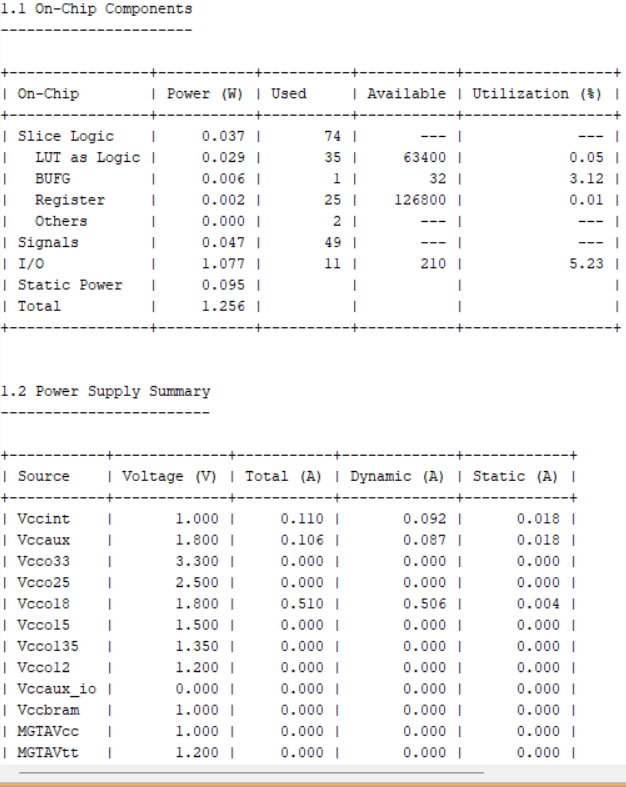
In the code here, parity bit is not added. You can add it for good design and implementation.

*5. Simulation Results*

**

*6. Design Summary Results*

**



#### 7.Applications

* Debugging: Early detection of system bugs is important during development. Adding UART can help in this scenario by capturing messages from the system.
* X Manufacturing function-level tracing: Logs are very important in manufacturing. They determine functionalities by alerting operators to what is happening on the manufacturing line.
* Customer or client updates: Software updates are highly important. Having complete, dynamic hardware with update-capable software is important to having a complete system.
* Testing/verification: Verifying products before they leave the manufacturing process helps deliver the best quality products possible to customers

*CONCLUSION*

We designed and Implemented Universal Asynchronous Receiver Transmitter(UART). In future we would like to implement it in large system and have high flexibility in FPGA based design. UART controller can be designed using FIFO (First In First Out) buffer to avoid loss of data. It can be Simulated and synthesized using Xilinx ISE 13.1. The design can be successfully downloaded and verified on Spartan-3E FPGA board. The data that will be sent can generate output LEDs on Spartan-3E.

##### Acknowledgment

##### We would like to thank the course staff (Chayan Bhar Sir) and Lab Incharge (Narendar Sir) for Introduction to Digital Design and Digital Integrated Circuits and also our lab assistants who helped us to improve Verilog Language.

##### References

1. Irfansyah, Sardi. “DESIGN AND IMPLEMENTATION OF UART WITH FIFO BUFFER USING VHDL ON FPGA.” (2019). J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.