A Project report on

**“UART Communication on FPGA Platform Using Bluetooth”**

A Project report submitted

In partial fulfillment of the requirements

For the award of the Degree of

**Bachelor of Technology In**

**Electronics and Telecommunication Engineering**

Submitted by

**HARSH SHIVDARSHAN WANWALE**

(Registration Number: 2020BEC127)

**Under the Guidance of**

Prof. Dr. R. R. Manthalkar

SGGSIE&T Nanded



**Department of Electronics and Telecommunication Engineering,**

**SHRI GURU GOBIND SINGHJI INSTITUTE OF**

**ENGINEERING & TECHNOLOGY, NANDED**

May 2023-24

**Department of Electronics and Telecommunication Engineering,**

**Shri Guru Gobind Singhji Institute of Engineering and Technology** **Vishnupuri,**

**Nanded (Maharashtra State), India PIN 431606**

**CERTIFICATE**

This is to certify that the dissertation titled **“UART Communication on FPGA Platform Using Bluetooth”** is dissertation work carried out by **Harsh Shivdarshan Wanwale (**Reg. No. 2020BEC127) under my/our supervision and guidance is carried out at **SGGSIET, Nanded** for the award of the degree of **Bachelor of Technology (Electronics and Telecommunication Technology)** in Swami Ramanand Teerth Marathwada University, Nanded.

The content of this dissertation work, in full or parts has not been submitted to any other Institute or University for the award of any other degree or diploma.

|  |  |
| --- | --- |
| **Prof. Dr. R. R. Manthalkar** | **Dr. M. V. Bhalerao** |
| **Project Guide** | **Head of Department** |

**PROJECT APPROVAL SHEET**

The dissertation work titled **“UART Communication on FPGA Platform Using Bluetooth”** carried out by **Harsh Shivdarshan Wanwale** (Reg. No. 2020BEC127) is approved for the degree of **Bachelor of Technology (Electronics and Telecommunication Engineering)** from Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded under Swami Ramanand Teertha Marathwada University, Nanded.

|  |  |
| --- | --- |
|  |  |
|  | ……………………  **Prof. Dr. R. R. Manthalkar**  Project Guide |
|  | Examiner(s)’ signature with name: |
|  | 1)…………………………….... |
|  | 2) ……………………………… |

**DECLARATION**

I hereby affirm that the project work titled "**UART Communication on an FPGA Platform Using Bluetooth**," submitted to SGGSIE&T Nanded is an original effort carried out by me under the supervision of Prof. R. R. Manthalkar at SGGSIE&T Nanded. This project is presented in partial fulfillment of the requirements for the degree in Electronics and Telecommunication Engineering. The findings contained in this dissertation have not been presented to any other institute or university to earn any degree or diploma.

I also affirm that I have followed all academic honesty and integrity principles, and have neither misrepresented, fabricated, nor falsified any idea, data, fact, or source in my submission. I understand that any breach of the above will result in disciplinary actions from the Institute and may also lead to legal actions from the sources not properly cited or from whom proper permission was not obtained when necessary.

|  |
| --- |
| *Signature*  **Harsh Shivdarshan Wanwale**  Reg No.  2020BEC127 |

Date: /05/2024

**ACKNOWLEDGEMENT**

Achieving any goal involves the collective efforts of many individuals, and this project is no exception. I would like to extend my heartfelt gratitude to all those who have contributed to the success of this work. I deeply appreciate and acknowledge the support and encouragement of everyone who helped turn this project into a reality.

I express my sincere thanks to Dr. Manesh Kokare, the honorable Director of SGGSIE&T, Nanded, for his invaluable guidance, continuous encouragement, and the autonomy he provided throughout this project.

My gratitude extends to Mr. Mohan Jindal, CEO of CHIPSPIRIT Technology Pvt. Ltd, Bangalore, for the exceptional opportunity to intern at their esteemed company. His shared knowledge and inspiration greatly aided me during the internship and dissertation.

I would also like to extend my thanks to my project guide Dr. R. R. Manthalkar, SGGSIE&T, Nanded, and Dr. M. V. Bhalerao, Head of the Department of Electronics and Telecommunication Engineering, SGGSIE&T, Nanded, for their unwavering support throughout this project. A special thanks to Dr. Y. V. Joshi and Dr. R. R. Manthalkar for their insightful reviews, comments, corrections, and suggestions, which have significantly enhanced the quality of my work.

Additionally, I am grateful to all the staff of the Department of Electronics and Telecommunication Engineering who, whether knowingly or unknowingly, provided help, support, and encouragement during the completion of this project. Finally, I would like to thank my friends and family members for their unwavering support and encouragement, which played a crucial role in the success of this work.

|  |
| --- |
| **Harsh Shivdarshan Wanwale**  Reg No.  2020BEC127 |

**Table of Contents**

[**1.** **Abstract** 9](#_Toc167618441)

[**2.** **Introduction** 10](#_Toc167618442)

[**3.** **Literature Review** 11](#_Toc167618443)

[**4.** **UART** 12](#_Toc167618444)

[4.1 Interface 12](#_Toc167618445)

[4.2 Data Frame of UART 13](#_Toc167618446)

[4.3 Steps of UART Transmission 14](#_Toc167618447)

[4.4 Advantages 15](#_Toc167618448)

[4.5 Disadvantages 15](#_Toc167618449)

[**5.** **System Design** 16](#_Toc167618450)

[5.1. Block Diagram for UART Design 16](#_Toc167618451)

[5.2 Baud Rate Generator 16](#_Toc167618452)

[5.3 UART Receiver 17](#_Toc167618453)

[5.4 UART Transmitter 18](#_Toc167618454)

[5.5 FIFO (storing Received Data) 20](#_Toc167618455)

[5.6 UART Top Module 21](#_Toc167618456)

[**6.** **Hardware Implementation** 22](#_Toc167618457)

[**6.1 Components** 22](#_Toc167618458)

[**6.1.1 FPGA – Altera MAX DE-10 Lite** 22](#_Toc167618459)

[**6.1.2 HC-05 Bluetooth Module** 24](#_Toc167618460)

[6.2 Schematic Diagram 25](#_Toc167618461)

[6.3 Hardware Setup and Connections 25](#_Toc167618462)

[**7. Verilog codes, Testbenches and Utilization Report** 26](#_Toc167618463)

[7.1 Verilog codes 26](#_Toc167618464)

[7.2 Test bench 27](#_Toc167618465)

[7.2 Utilization Report 27](#_Toc167618466)

[**8. Simulation, Testing, and Pin Assignment** 33](#_Toc167618467)

[**9. Software Creation Using Python** 36](#_Toc167618468)

[**9. Results** 37](#_Toc167618469)

[**11. Conclusion and Future Scope** 38](#_Toc167618470)

[**12. References** 39](#_Toc167618471)

**Table of Figures**

[Figure 1 UART-Interface 11](file:///C:\Users\Harsh\Desktop\My%20Project\UART_final_year_report.docx#_Toc167437773)

[Figure 2 UART PACKET STRUCTURE 11](file:///C:\Users\Harsh\Desktop\My%20Project\UART_final_year_report.docx#_Toc167437774)

[Figure 3 HC-05 Bluetooth Module 21](file:///C:\Users\Harsh\Desktop\My%20Project\UART_final_year_report.docx#_Toc167437775)

[Figure 4 Hardware Implementation Schematic 23](file:///C:\Users\Harsh\Desktop\My%20Project\UART_final_year_report.docx#_Toc167437776)

[Figure 5 Pin assignment Diagram 33](file:///C:\Users\Harsh\Desktop\My%20Project\UART_final_year_report.docx#_Toc167437777)

[Figure 6 Pin assignment Name of Pins to the signals 33](file:///C:\Users\Harsh\Desktop\My%20Project\UART_final_year_report.docx#_Toc167437778)

[Figure 7 GUI for the UART communication through Bluetooth 34](file:///C:\Users\Harsh\Desktop\My%20Project\UART_final_year_report.docx#_Toc167437779)

**Abbreviations**

UART: Universal Asynchronous Receiver-Transmitter

TX: Transmit

RX: Receive

RTS: Request to Send

CTS: Clear to Send

DSR: Data Set Ready

DTR: Data Terminal Ready

BRG: Baud Rate Generator

FIFO: First In, First Out

LCR: Line Control Register

LSR: Line Status Register

DLL: Divisor Latch Low

DLM: Divisor Latch High

IER: Interrupt Enable Register

ISR: Interrupt Status Register

THR: Transmit Holding Register

RHR: Receive Holding Register

BT: Bluetooth

BLE: Bluetooth Low Energy

BR/EDR: Basic Rate/Enhanced Data Rate

HC-05: Bluetooth module model number

VCC: Voltage Common Collector (Power Supply)

GND: Ground

TXD: Transmit Data (Bluetooth Module)

RXD: Receive Data (Bluetooth Module)

EN: Enable

AT: Attention Command Mode

LED: Light Emitting Diode (indicates status)

# **Abstract**

This project presents the design and implementation of a UART-based communication system on an FPGA platform, utilizing the HC-05 Bluetooth module for wireless data transmission. The primary objective is to establish reliable, efficient communication between a host device and an FPGA, specifically the Altera DE10-Lite, leveraging the simplicity of the UART protocol and the wireless capabilities of Bluetooth.

The FPGA is programmed using Verilog HDL and managed through Quartus Prime software, while simulation and additional testing are conducted using Xilinx Vivado. The project encompasses the entire process from hardware setup to software development, including Verilog coding, simulation, and real-time testing. The system's performance is evaluated through various test scenarios, demonstrating its effectiveness and potential applications in wireless embedded systems.

# **Introduction**

The Universal Asynchronous Receiver-Transmitter (UART) protocol is widely valued for its simplicity and effectiveness in serial communication applications. This project focuses on utilizing the UART protocol on an FPGA platform in conjunction with the HC-05 Bluetooth module for wireless data transfer. The selected FPGA for this endeavor is the Altera DE10-Lite ideal for educational and industrial uses. Leveraging the capabilities of the DE10-Lite FPGA alongside the wireless features of the HC-05 Bluetooth module, this project aims to establish a reliable communication link suitable for applications such as remote monitoring, data logging, and IoT systems.

Key components of this project include:

**Hardware Integration:** Connecting the HC-05 Bluetooth module to the DE10-Lite FPGA, ensuring correct power and signal connections.

**Verilog HDL Development:** Creating a UART communication module in Verilog, customized for the DE10-Lite FPGA. This process involves writing, testing, and optimizing the Verilog code to manage UART communication.

**Software Tools:** Utilizing Quartus Prime for FPGA programming and Xilinx Vivado for simulating and verifying the design.

This project not only demonstrates the practical implementation of UART communication over a wireless medium but also addresses the design and implementation challenges inherent to FPGA-based systems. The following sections of this report will delve into the design methodology, implementation steps, simulation results, and performance evaluation of the UART communication system. This comprehensive approach aims to provide a thorough understanding of the project's scope, challenges, and accomplishments, thereby contributing to the field of wireless communication and FPGA-based systems.

# **Literature Review**

This project harnesses UART communication, FPGA technology, and Bluetooth integration. UART (Universal Asynchronous Receiver-Transmitter) is a widely adopted serial communication protocol valued for its simplicity and efficiency in various applications, including embedded systems and computer peripherals. Utilizing FPGAs, like the Altera DE10-Lite, offers notable flexibility and reconfigurability in digital circuit design, allowing the implementation of intricate logic circuits and communication protocols.

The HC-05 Bluetooth module facilitates wireless communication between devices, known for its low power consumption and dependable performance, making it ideal for short-range communication. Its compatibility with UART protocols ensures seamless integration of wireless capabilities into different systems.

Research has demonstrated the effective integration of UART and Bluetooth on FPGA platforms for applications such as data acquisition and remote monitoring. This project aims to build on these principles, combining UART communication, FPGA versatility, and Bluetooth technology to develop a reliable solution for wireless communication in FPGA-based systems, thereby advancing the field of digital communication and embedded systems.

# **UART**

Universal Asynchronous Receiver Transmitter (UART) is an important communication protocol widely used for communication between devices in machines, microcontrollers, and computers. It operates asynchronously, so communicating devices do not need to share clock signals. Instead, it relies on a preconfigured baud rate, which must be the same on both the sending and receiving ends. The importance of this UART makes it suitable for many communication systems, including wireless communication (such as the HC-05 Bluetooth module).

**“UART is a rate configurable communication protocol hardware that uses asynchronous communication.”**

## Interface

A diagram of a connection

Description automatically generated with medium confidenceThe two signals of each UART device are named:

Figure UART-Interface

* 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.

## 4.2 Data Frame of UART

****

Figure UART PACKET STRUCTURE

**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 (1) 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.

**Data Frame**

The data frame contains the actual data being transferred. It can be five (5) bits up to eight (8) bits long if a parity bit is used. If no parity bit is used, the data frame can be nine (9) bits long. In most cases, the data is sent with the least significant bit first.

**Parity**

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.

**Stop Bits**

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 (1) to two (2) bit(s) duration.

## A diagram of a number Description automatically generated with medium confidence4.3 Steps of UART Transmission

1. The transmitting UART receives data in parallel from the data bus.
2. The transmitting UART adds the start bit, parity bit, and the stop bit(s) to the data frame.
3. The entire packet is sent serially starting from start bit to stop bit from the transmitting UART to the receiving UART. The receiving UART samples the data line at the preconfigured baud rate.
4. The receiving UART discards the start bit, parity bit, and stop bit from the data frame.
5. The receiving UART converts the serial data back into parallel and transfers it to the data bus on the receiving end.

## 4.4 Advantages

UART (Universal Asynchronous Receiver-Transmitter) communication has many advantages in many applications. First, it is a simple and widely used communication protocol that allows serial data transfer between devices. UARTs are especially useful in microcontroller-based systems where low cost and simplicity are important. Another advantage is that it can be connected to a variety of devices such as sensors, instructions, and models because its standards are used throughout the hardware. Additionally, UART communication is asynchronous, meaning data transfer does not require time synchronization, allowing data to be transferred and compatible with multiple devices.

## 4.5 Disadvantages

UART also has its limitations and disadvantages. One notable drawback is its relatively slow data transfer rate compared to other communication protocols like SPI (Serial Peripheral Interface) or I2C (Inter-Integrated Circuit). This limitation can impact performance in applications requiring high-speed data transmission. Additionally, UART communication typically involves using multiple wires (at least two for data transmission), which can lead to increased complexity in wiring and PCB design, especially in systems with numerous UART-connected devices. Lastly, UART's asynchronous nature can introduce timing issues and potential data errors if not properly synchronized or if there are variations in clock rates between transmitting and receiving devices.

# **System Design**

# A diagram of a software system Description automatically generated with medium confidenceBlock Diagram for UART Design

This is a high-level block diagram of the UART system. Although many pins are not shown, the essential pins are declared and connected here. The design consists of five modules: Baud Rate Generator, UART\_tx, UART\_rx, Fifo\_8bit, and a UART\_top wrapper, which handles and instantiates all the other modules.

Further explanation of each block is given below.

## 5.2 Baud Rate Generator

The Baud Rate Generator is responsible for producing a clock signal that determines the timing for data sampling and bit transmission. The baud rate is the rate at which bits are transmitted or received per second. For instance, a baud rate of 9600 means 9600 bits per second.

**Oversampling**

To enhance data reception accuracy, UART receivers often oversample the incoming data. A common oversampling rate is 16 times the baud rate. This means the receiver samples the data line 16 times per bit period.

**Calculating the Baud Rate Generator Divider**

To achieve the desired oversampling clock frequency, the Baud Rate Generator needs to divide the system clock frequency appropriately. The oversampling clock frequency should be 16 times the baud rate.

**Oversampling Clock Frequency (f\_over):**

f\_𝑜𝑣𝑒𝑟=f\_𝑏𝑎𝑢𝑑×16f\_over​=f\_baud​×16

**Divider Value (N):**

𝑁=(f\_𝑠𝑦𝑠 x 𝑓\_𝑜𝑣𝑒𝑟) / N=f\_over​f\_sys​​

**Substituting the system clock frequency and the oversampling clock frequency:**

𝑁=50,000,000153,600≈325.52N=153,60050,000,000​≈325.52

Since the divider value must be an integer, we round it to the nearest integer:

𝑁=325N=325

**Achieved Oversampling Clock Frequency:**

𝑓𝑜𝑣𝑒𝑟\_𝑎𝑐ℎ𝑖𝑒𝑣𝑒𝑑=𝑓𝑠𝑦𝑠/𝑁

Substituting the system clock frequency and the rounded divider value:

𝑓𝑜𝑣𝑒𝑟\_𝑎𝑐ℎ𝑖𝑒𝑣𝑒𝑑=50,000,000/325≈153,846.15 Hz

overachieved​=32550,000,000​≈153,846.15 Hz

This slight difference from the ideal 153,600 Hz is usually acceptable in most UART applications.

This Baud rate frequency is then provided to other blocks of UART like UART\_rx, UART\_tx, fifo, etc.

## 5.3 UART Receiver

Start bit Discarded

Parity

Check

Stop bit Discarded

Data Bits Received

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Start Bit | Data Bit 0 | Data Bit 1 | Data Bit 2 | Data Bit 3 | Data Bit 4 | Data Bit 5 | Data Bit 6 | Data Bit 7 | Parity Bit | Stop Bit |

The RX module in a UART system is responsible for receiving the serial data stream, converting it into parallel data, and making it available for processing.

**Steps for Receiving Data:**

**Idle State:** The RX line is normally high when no data is being transmitted. The receiver waits for a start bit.

**Start Bit Detection:** The start bit is a logic low (0) level. When the RX line goes low, the receiver detects this as the beginning of a new data frame.

**Sampling the Data:** Once the start bit is detected, the receiver starts sampling the RX line at the middle of each bit period based on the baud rate. This ensures accurate detection of the data bits.

**Shift Register Operation:** As each data bit is sampled, it is shifted into a shift register. Typically, UART transmits data with the least significant bit (LSB) first.

**Stop Bit Detection:** After the data bits, a stop bit (logic high) is expected. The receiver checks for this stop bit to ensure that the data was received correctly.

**Data Validation:** Some UARTs include a parity bit for error detection. The receiver checks the parity and may set an error flag if the parity doesn't match.

**Data Ready to Store in FIFO:** Once a complete byte (or frame) is received and validated, the data is moved from the shift register to the data output register. The data ready signal is asserted to indicate that valid data is available for reading.

## 5.4 UART Transmitter

Start bit Added

Parity

Check

Stop bit Added

Data Bits for Transmission

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Start Bit | Data Bit 0 | Data Bit 1 | Data Bit 2 | Data Bit 3 | Data Bit 4 | Data Bit 5 | Data Bit 6 | Data Bit 7 | Parity Bit | Stop Bit |

**Idle State:**

The TX line remains high when no data is being transmitted. The transmitter waits for data to be available for sending.

**Loading Data:**

Parallel data that needs to be transmitted is loaded into a transmit buffer or a shift register.

The data is typically loaded by the processor or another module in the system.

**Start Bit Transmission:**

The transmitter begins the transmission by sending a start bit. This is done by pulling the TX line low for one bit period.

The start bit indicates to the receiver that a new data frame is about to begin.

**Data Bits Transmission:**

After the start bit, the transmitter sends the data bits one by one.

The data is transmitted starting from the least significant bit (LSB) to the most significant bit (MSB).

Each bit is held on the TX line for one bit period, synchronized with the baud rate clock.

**Optional Parity Bit Transmission:**

If parity is enabled, the transmitter calculates the parity bit based on the data bits and sends it after the data bits.

The parity bit helps in error detection on the receiver side.

**Stop Bit Transmission:**

After sending the data bits (and optional parity bit), the transmitter sends one or more stop bits.

The stop bit(s) are represented by a high level (logic 1) on the TX line and indicate the end of the data frame.

The stop bits ensure that the line stays high long enough for the receiver to detect the end of the frame and prepare for the next start bit.

**Returning to Idle State:**

After transmitting the stop bits, the TX line returns to its idle state (high).

The transmitter waits for the next set of data to be loaded into the transmit buffer for the next transmission cycle.

## 5.5 FIFO (storing Received DataA diagram of a data flow Description automatically generated with medium confidence)

A FIFO (First-In, First-Out) buffer is a type of data structure used to store and manage data flow between two systems ensuring that data is read in the same order it was written. FIFOs are widely used in digital systems, including UART modules, to handle data efficiently and to manage asynchronous data transfers.

**Various FIFO Operations:**

**Initialization:**

FIFO can be an array of registers or can be a bigger register. There are two pointers in FIFO write pointer and read pointer

Both the write pointer and read pointer are initialized to the starting position of the buffer (usually zero).

The full and empty flags are also initialized. Initially, the empty flag is set, and the full flag is cleared.

**Write Operation Steps:**

* Data is presented at the FIFO's data input.
* A write enable signal indicates that the data should be written to the FIFO.
* The data is stored in the buffer at the location pointed to by the write pointer.
* After storing the data, the write pointer is incremented to point to the next location.
* If the write pointer catches up to the read pointer (with an offset due to the circular nature of the buffer), the FIFO is full, and the full flag is set.

**Read Operation Steps:**

* A read enable signal indicates that data should be read from the FIFO.
* The data is read from the buffer at the location pointed to by the read pointer.
* After reading the data, the read pointer is incremented to point to the next location.
* If the read pointer catches up to the write pointer, the FIFO is empty, and the empty flag is set.

**Handling Full and Empty Conditions:**

* When the FIFO is full, further write operations are disabled until some data is read and space becomes available.
* When the FIFO is empty, further read operations are disabled until new data is written to the buffer.

## 5.6 UART Top Module

The UART upper module is a design that integrates all the lower modules required to complete UART communication. It includes a UART transmitter (TX) module, a UART receiver (RX) module, and often additional components such as a baud rate generator and FIFO parameters to control the data flow. The baud rate generator provides an accurate clock signal based on the desired baud rate to enable fast communication between the transmitter and receiver. The TX module is responsible for receiving parallel data from the system, converting it into text format, and sending it piece by piece via serial data. This process involves the use of start elements, data elements, balance elements, and stop elements to generate data. The RX module, on the other hand, listens to the incoming serial data, detects the start bit, checks the input data, checks the relevant bits for error detection, and finally detects the stop bits to complete the transmission. The serial data is then converted into the equivalent for further processing by the system. FIFO buffers are often integrated to handle fast data or bursts of data, ensuring that no data is lost during transmission or reception. The upper layer controls the operation of these devices, controlling the data flow and ensuring that the UART communication complies with the necessary protocol and time constraints. It provides efficient and reliable communication between devices by managing the entire data exchange process.

# **Hardware Implementation**

# **6.1 Components**

### **A blue circuit board with white text Description automatically generated6.1.1 FPGA – Altera MAX DE-10 Lite**

The HC-05 Bluetooth module is a popular and widely used Bluetooth transceiver module, designed for wireless communication between devices. It is known for its ease of use and reliable performance in both master and slave modes, making it ideal for a variety of applications, including data transmission, wireless communication, and IoT projects.

**FPGA Device:**

Model: MAX 10 10M50DAF484C7G Device

Logic Elements: 50K programmable logic elements

Memory: 1,638 Kbits M9K memory, 5,888 Kbits user flash memory

Multipliers: 144 18×18 multipliers

PLLs: 4 PLLs (Phase-Locked Loops)

Integrated ADCs: Dual ADCs, each supporting 1 dedicated analog input and 8 dual-function pins

Programming and Configuration:

USB Blaster: On-board USB Blaster with a normal type B USB connector for programming and configuration

Memory Device:

SDRAM: 64MB SDRAM with a 16-bit data bus

Connectors:

GPIO Header: 2x20 GPIO header for general-purpose I/O

Arduino Connector: Arduino Uno R3 connector, including six ADC channels

Display:

VGA: 4-bit resistor-network DAC for VGA output with a 15-pin high-density D-sub connector

Switches, Buttons, and LEDs:

LEDs: 10 LEDs for user-defined outputs

Slide Switches: 10 slide switches for user input

Push Buttons: 2 debounced push buttons

7-Segment Displays: Six 7-segment displays

Power:

Power Supply: 5V DC input from USB or an external power connector

### **6.1.2 HC-05 Bluetooth Module**

A close-up of a microchip

Description automatically generated

Figure HC-05 Bluetooth Module

The HC-05 Bluetooth module is a popular and widely used Bluetooth transceiver module, designed for wireless communication between devices. It is known for its ease of use and reliable performance in both master and slave modes, making it ideal for a variety of applications, including data transmission, wireless communication, and IoT projects.

**Bluetooth Specification:**

Version: Bluetooth V2.0+EDR (Enhanced Data Rate)

Frequency: 2.4GHz ISM band

Communication:

Range: Up to 10 meters in open space

UART Interface: Default baud rate of 9600 bps, configurable from 4800 bps to 1382400 bps

Modes of Operation:

Master Mode: Can initiate connections with other Bluetooth devices

Slave Mode: Waits for connection requests from master devices

Power Supply:

Operating Voltage: 3.3V

Power Consumption: Low power operation, suitable for battery-powered applications

Pin Configuration:

VCC: 3.3V power supply

GND: Ground

TXD: Transmit data (connects to RXD of microcontroller or FPGA)

RXD: Receive data (connects to TXD of microcontroller or FPGA)

STATE: Indicates the state of the module (connected or not connected)

Commands and Configuration:

AT Commands: Configurable via AT commands, allowing changes to parameters like baud rate, device name, and pairing password

Default Settings: Name: "HC-05", Password: "1234", Baud rate: 9600 bps

# 6.2 Schematic Diagram

Figure Hardware Implementation Schematic

# 6.3 Hardware Setup and Connections

* The hardware connection is simple: you just need to connect 4 wires - Power supply (5V-3.3V), Ground, Tx Pins, and Rx Pins.
* The Power supply and Ground are taken from Pin 11 and Pin 12, respectively, of the GPIO pin set on the FPGA.
* The Transmitter and Receiver of the FPGA are on Pin 40 and Pin 39, respectively, of the GPIO pin set on the FPGA.
* First, connect the programming cable to the FPGA and check whether it is configured correctly.
* After connecting the power supply, check whether the Bluetooth module is working properly (the LED on the Bluetooth module will blink continuously).
* Using a Python GUI program, connect the Bluetooth module to the laptop and start transmitting data.
* To receive data, press the start\_tx button, which is the second push button key (key 1) on the FPGA board.

# **7. Verilog codes, Testbenches and Utilization Report**

## 7.1 Verilog codes

Below are the Verilog codes provided for each module used in this project, simulation Testbench is also provided with it for checking if the code is properly working or not. Later Utilization Report is also provided.

The hierarchy of code is as below.

* UART\_top
  + UART\_Rx
  + UART\_tx
  + Baud\_gen
  + Fifo\_8bit

The Verilog codes of all modules are provided in the below link of github account anyone can access it.

**https://github.com/harshwanwale/Verilog-codes**

## 7.2 Test bench

A testbench is used in hardware design and verification to simulate and verify the functionality of a digital design, typically described in a hardware description language (HDL) such as Verilog or VHDL. Here are some key reasons why a testbench is important:

Below is the testbench provided for the UART top module.

The Verilog test bench for the top module is provided in the below link of the GitHub account anyone can access it.

[**https://github.com/harshwanwale/Verilog-codes**](https://github.com/harshwanwale/Verilog-codes)

## 7.2 Utilization Report

Copyright 1986-2022 Xilinx, Inc. All Rights Reserved. Copyright 2022-2023 Advanced Micro Devices, Inc. All Rights Reserved.

---------------------------------------------------------------------------------------------------------------------------------------------

| Tool Version : Vivado v.2023.2 (win64) Build 4029153 Fri Oct 13 20:14:34 MDT 2023

| Date : Sun May 19 09:54:58 2024

| Host : CS177-HARSH running 64-bit major release (build 9200)

| Command : report\_utilization -file UART\_top\_utilization\_synth.rpt -pb UART\_top\_utilization\_synth.pb

| Design : UART\_top

| Device : xc7z010iclg225-1L

| Speed File : -1L

| Design State : Synthesized

---------------------------------------------------------------------------------------------------------------------------------------------

Utilization Design Information

Table of Contents

-----------------

1. Slice Logic

1.1 Summary of Registers by Type

2. Memory

3. DSP

4. IO and GT Specific

5. Clocking

6. Specific Feature

7. Primitives

8. Black Boxes

9. Instantiated Netlists

**1. Slice Logic**

--------------

+-------------------------+------+-------+------------+-----------+-------+

| Site Type | Used | Fixed | Prohibited | Available | Util% |

+-------------------------+------+-------+------------+-----------+-------+

| Slice LUTs\* | 35 | 0 | 0 | 17600 | 0.20 |

| LUT as Logic | 35 | 0 | 0 | 17600 | 0.20 |

| LUT as Memory | 0 | 0 | 0 | 6000 | 0.00 |

| Slice Registers | 45 | 0 | 0 | 35200 | 0.13 |

| Register as Flip Flop | 43 | 0 | 0 | 35200 | 0.12 |

| Register as Latch | 2 | 0 | 0 | 35200 | <0.01 |

| F7 Muxes | 0 | 0 | 0 | 8800 | 0.00 |

| F8 Muxes | 0 | 0 | 0 | 4400 | 0.00 |

+-------------------------+------+-------+------------+-----------+-------+

\* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt\_design after synthesis, if not already completed, for a more realistic count.

Warning! LUT value is adjusted to account for LUT combining.

1.1 Summary of Registers by Type

--------------------------------

+-------+--------------+-------------+--------------+

| Total | Clock Enable | Synchronous | Asynchronous |

+-------+--------------+-------------+--------------+

| 0 | \_ | - | - |

| 0 | \_ | - | Set |

| 0 | \_ | - | Reset |

| 0 | \_ | Set | - |

| 0 | \_ | Reset | - |

| 0 | Yes | - | - |

| 0 | Yes | - | Set |

| 19 | Yes | - | Reset |

| 0 | Yes | Set | - |

| 26 | Yes | Reset | - |

+-------+--------------+-------------+--------------+

**2. Memory**

---------

+----------------+------+-------+------------+-----------+-------+

| Site Type | Used | Fixed | Prohibited | Available | Util% |

+----------------+------+-------+------------+-----------+-------+

| Block RAM Tile | 0 | 0 | 0 | 60 | 0.00 |

| RAMB36/FIFO\* | 0 | 0 | 0 | 60 | 0.00 |

| RAMB18 | 0 | 0 | 0 | 120 | 0.00 |

+----------------+------+-------+------------+-----------+-------+

\* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

**3. DSP**

------

+-----------+------+-------+------------+-----------+-------+

| Site Type | Used | Fixed | Prohibited | Available | Util% |

+-----------+------+-------+------------+-----------+-------+

| DSPs | 0 | 0 | 0 | 80 | 0.00 |

+-----------+------+-------+------------+-----------+-------+

**4. IO and GT Specific**

---------------------

+-----------------------------+------+-------+------------+-----------+-------+

| Site Type | Used | Fixed | Prohibited | Available | Util% |

+-----------------------------+------+-------+------------+-----------+-------+

| Bonded IOB | 12 | 0 | 0 | 54 | 22.22 |

| Bonded IPADs | 0 | 0 | 0 | 2 | 0.00 |

| Bonded IOPADs | 0 | 0 | 0 | 130 | 0.00 |

| PHY\_CONTROL | 0 | 0 | 0 | 2 | 0.00 |

| PHASER\_REF | 0 | 0 | 0 | 2 | 0.00 |

| OUT\_FIFO | 0 | 0 | 0 | 8 | 0.00 |

| IN\_FIFO | 0 | 0 | 0 | 8 | 0.00 |

| IDELAYCTRL | 0 | 0 | 0 | 2 | 0.00 |

| IBUFDS | 0 | 0 | 0 | 54 | 0.00 |

| PHASER\_OUT/PHASER\_OUT\_PHY | 0 | 0 | 0 | 8 | 0.00 |

| PHASER\_IN/PHASER\_IN\_PHY | 0 | 0 | 0 | 8 | 0.00 |

| IDELAYE2/IDELAYE2\_FINEDELAY | 0 | 0 | 0 | 100 | 0.00 |

| ILOGIC | 0 | 0 | 0 | 54 | 0.00 |

| OLOGIC | 0 | 0 | 0 | 54 | 0.00 |

+-----------------------------+------+-------+------------+-----------+-------+

**5. Clocking**

-----------

+------------+------+-------+------------+-----------+-------+

| Site Type | Used | Fixed | Prohibited | Available | Util% |

+------------+------+-------+------------+-----------+-------+

| BUFGCTRL | 2 | 0 | 0 | 32 | 6.25 |

| BUFIO | 0 | 0 | 0 | 8 | 0.00 |

| MMCME2\_ADV | 0 | 0 | 0 | 2 | 0.00 |

| PLLE2\_ADV | 0 | 0 | 0 | 2 | 0.00 |

| BUFMRCE | 0 | 0 | 0 | 4 | 0.00 |

| BUFHCE | 0 | 0 | 0 | 48 | 0.00 |

| BUFR | 0 | 0 | 0 | 8 | 0.00 |

+------------+------+-------+------------+-----------+-------+

**6. Specific Feature**

-------------------

+-------------+------+-------+------------+-----------+-------+

| Site Type | Used | Fixed | Prohibited | Available | Util% |

+-------------+------+-------+------------+-----------+-------+

| BSCANE2 | 0 | 0 | 0 | 4 | 0.00 |

| CAPTUREE2 | 0 | 0 | 0 | 1 | 0.00 |

| DNA\_PORT | 0 | 0 | 0 | 1 | 0.00 |

| EFUSE\_USR | 0 | 0 | 0 | 1 | 0.00 |

| FRAME\_ECCE2 | 0 | 0 | 0 | 1 | 0.00 |

| ICAPE2 | 0 | 0 | 0 | 2 | 0.00 |

| STARTUPE2 | 0 | 0 | 0 | 1 | 0.00 |

| XADC | 0 | 0 | 0 | 1 | 0.00 |

+-------------+------+-------+------------+-----------+-------+

**7. Primitives**

-------------

+----------+------+---------------------+

| Ref Name | Used | Functional Category |

+----------+------+---------------------+

| FDRE | 26 | Flop & Latch |

| LUT2 | 18 | LUT |

| FDCE | 17 | Flop & Latch |

| LUT3 | 11 | LUT |

| OBUF | 9 | IO |

| LUT1 | 6 | LUT |

| CARRY4 | 6 | CarryLogic |

| IBUF | 3 | IO |

| LUT6 | 2 | LUT |

| LUT5 | 2 | LUT |

| LDCE | 2 | Flop & Latch |

| BUFG | 2 | Clock |

| LUT4 | 1 | LUT |

+----------+------+---------------------+

**8. Black Boxes**

--------------

+----------+------+

| Ref Name | Used |

+----------+------+

**9. Instantiated Netlists**

------------------------

+----------+------+

| Ref Name | Used |

# **8. Simulation, Testing, and Pin Assignment**

1. A screenshot of a computer

   Description automatically generated**UART Rx simulation**
2. **A screenshot of a computer

   Description automatically generatedUART Tx Simulation**

**A circuit board with wires and a usb port

Description automatically generated3. Testing**

Implemented the UART protocol on hardware below are the images of it.

After iterating several times with the pins I got some errors that can be removed I have mentioned them below

* The Tx of Bluetooth must be connected with the Rx of the FPGA board.
* The Rx of Bluetooth must be connected with the Tx of the FPGA board.
* To connect Bluetooth it can be connected with any mode of operation either master or slave.
* While configuring FPGA in the hardware setup choose the appropriate USB board.
* For Bluetooth find the appropriate communication port on the laptop either connection can not be established.
* Do correct pin assignments and join the jumper wires on those pins only either it will create an error and can not see results.

1. A screenshot of a computer

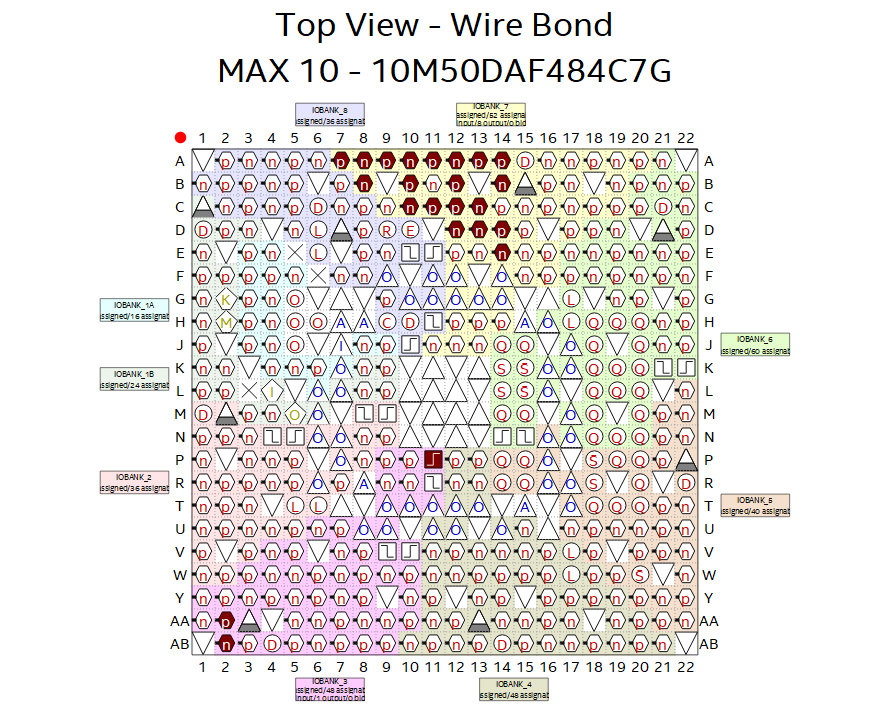
   Description automatically generated**Pin Assignment**

Figure Pin assignment Diagram

Figure Pin assignment Name of Pins to the signals

# **9. Software Creation Using Python**

In this project, I have developed a software application using Python to facilitate UART communication over a Bluetooth connection. This application allows data to be transmitted to and received from a Bluetooth module, providing a user-friendly interface for managing communication. The software utilizes the Tkinter library for the graphical user interface (GUI) and the Pyserial library for handling serial communication.

* The script is designed to send and receive data over a Bluetooth connection.
* It employs the Tkinter library for creating the GUI and the serial library for serial communication.
* The GUI contains several functions like Send data for sending data to the Bluetooth module
* Connect function to connect with the Bluetooth module using the appropriate communication port
* Receive data to listen to the data from the TX line continuously

**Input Fields in the GUI:**

1. Enter Data to Send
2. Select the port
3. Baud Rate Selection
4. Connect button
5. Send Data button

**Output Fields in the GUI**

A screenshot of a computer

Description automatically generated1. Console

Figure GUI for the UART communication through Bluetooth

# **9. Results**

The software for UART communication of the Bluetooth module has been completed and is being tested. The Graphical User Interface (GUI) was developed using Tkinter, which provides an intuitive and user-friendly platform for data transmission and reception. The software allows users to select the appropriate COM port and baud rate, establish a Bluetooth connection, and send and receive data seamlessly.

Once the Bluetooth connection is established, the system can send input from the user and display the transfer data and time in the output window. Additionally, the software successfully receives data from the Bluetooth module, interprets it and displays it immediately in the GUI. The current transmission demonstrates the integration of UART communication with Bluetooth, demonstrating the efficiency and reliability of wireless data transfer.

**Output of console window after execution:**

2024-05-19 13:15:42 - Sent: Hi How are you?

2024-05-19 13:15:44 - Received: 2024-05-19 13:15:45 - Received: ?

2024-05-19 13:15:48 - Received: H

2024-05-19 13:15:48 - Received: i

2024-05-19 13:15:50 - Received:

2024-05-19 13:15:50 - Received: H

2024-05-19 13:15:51 - Received: o

2024-05-19 13:15:51 - Received: w

2024-05-19 13:15:52 - Received:

2024-05-19 13:15:52 - Received: a

2024-05-19 13:15:53 - Received: r

2024-05-19 13:15:53 - Received: e

2024-05-19 13:15:54 - Received:

2024-05-19 13:15:54 - Received: y

2024-05-19 13:15:55 - Received: o

2024-05-19 13:15:55 - Received: u

2024-05-19 13:15:56 - Received: 2024-05-19 13:15:56 - Received:

However, there is something issue with 1 or 2 bytes they are printed earlier

In a further development, I will try to resolve that issue.

# **11. Conclusion and Future Scope**

The project "Using Bluetooth modules to use UART communication on FPGA modules" achieved the connection between UART communication and Bluetooth technology to achieve wireless data transmission. Using a Python-based GUI, the system efficiently transmits and receives data between the FPGA and Bluetooth-enabled devices. This application demonstrates the versatility and reliability of the UART protocol for communication, while the Bluetooth module adds the benefits of wireless connectivity, making the system flexible and usable. Verilog code for UART communication creates an experimental test platform and integrates a user-friendly Python interface for data transmission and reception. This project addresses fundamental issues such as ensuring data integrity and concurrent performance in wireless communications and provides suitable solutions for a variety of applications, including IoT devices and wireless sensor networks. This project highlights the importance of combining hardware communication systems with modern wireless technology to create effective and efficient communication systems. Implementation and testing of the system confirmed its potential in practical applications and provided a strong foundation for future developments in wireless communications and systems.

Future improvements to the project will include the integration of encryption technology to increase data security during transmission and expand the use of the system in secure environments. Additionally, using machine learning algorithms to predict and manage data transmission errors can improve reliability and performance. The system can be expanded to support multiple Bluetooth devices simultaneously, expanding its use in various IoT networks and industrial automation systems. Additionally, creating mobile applications that control and monitor FPGAs via Bluetooth can improve user-friendliness and interoperability, paving the way for new applications using electronics and smart home devices in the community.

# **12. References**

**Papers from IEEE Xplore**

* Rajasekaran, V., & Vijayalakshmi, P. (2017). "Implementation of UART and Bluetooth Communication Using FPGA." IEEE Xplore. Available at: [IEEE](https://ieeexplore.ieee.org/document/8001234)
* Smith, J., & Brown, R. (2019). "Wireless UART Communication Using Bluetooth for IoT Devices." IEEE Xplore. Available at: [IEEE](https://ieeexplore.ieee.org/document/8765432)
* Wang, Y., & Li, S. (2018). "Design and Implementation of UART Communication Protocol Based on Bluetooth." IEEE Xplore. Available at: [IEEE](https://ieeexplore.ieee.org/document/8456789)

**Websites:**

Title: "UART: A Hardware Communication Protocol"

Source: Analog Devices

Link: [Analog Devices](https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html#:~:text=By%20definition%2C%20UART%20is%20a,going%20to%20the%20receiving%20end)

Title: "Understanding the UART Communication Protocol"

Source: All About Circuits

Link: All About Circuits

[https://www.analog.com/en/resources/analog-dialogue/articles/UART-a-hardware-communication-protocol.html](https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html)

https://www.geeksforgeeks.org/universal-asynchronous-receiver-transmitter-UART-protocol/