VISVESVARAYA TECHNOLOGICAL UNIVERSITY,

JNANASANGAMA, BELAGAVI- 590018

BLDEA's V.P. Dr. P.G. HALAKATTI COLLEGE OF ENGINEERING AND TECHNOLOGY, VIJAYAPUR

DEPARTMENT OF
ELECTRONICS AND COMMUNICATION ENGINEERING

A Mini Project report on

"UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER"

Submitted in partial fulfillment for the award of degree of Bachelor of Engineering in Electronics and Communication Engineering

Submitted by

AMAR HONAWAD	**2BL20EC009**
ANIL HALAGUNAKI	**2BL20EC010**
RAHUL BASETTI	**2BL20EC057**
RAHUL PATIL	**2BL20EC058**

Under the Guidance of

Prof. **PRAVEENKUMAR HAVALAGI**

2022-23

VISVESVARAYA TECHNOLOGICAL UNIVERSITY,

BELGAUM

B.L.D.E. Association's

V.P Dr. P.G HALAKATTI COLLEGE OF ENGINEERING AND TECHNOLOGY, VIJAYAPUR

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is Certified that the Mini project work entitled **"Universal Asynchronous Receiver Transmitter"** carried out by **Amar Honawad, Anil Halagunaki, Rahul Basetti, Rahul Patil,** bonafide students of **VP Dr P.G Halakatti College of Engineering and Technology, Vijayapura** in partial fulfillment for the award of **Bachelor of Engineering** in **Electronics and Communication Engineering** of the **Visvesvaraya Technological University, Belgaum** during the year 2022-2023. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The Mini project report has been approved as it satisfies the academic requirement in respect of Mini project work prescribed for the said degree.

GUIDE	H.O.D	PRINCIPAL
Prof. PRAVEENKUMAR H	**Dr. UMESH DIXIT**	**Dr. V. G. SANGAM**

B.L.D.E. Association's
VACHANA PITAMAHA Dr. P.G. HALAKATTI COLLEGE OF ENGINEERING & TECHNOLOGY, VIJAYAPURA

DEPARTMENT OF ELECTRONICS & COMMINUCATION ENGINEERING

DECLARATION

We, students of Sixth semester B.E, at the department of Electronics & Communication Engineering, hereby declare that, the Mini Project entitled " **Universal Asynchronous Receiver Transmitter**",embodies the report of our mini project work, carried out by us under the guidance of **Prof. Praveenkumar Havalagi,** We also declare that, to the best of our knowledge and belief, the work reported here in does not form part of any other report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this by any student.

Place:-Vijayapur

Date:- 08/07/2023

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose consistent guidance and encouragement crowned our efforts with success. We consider it as our privilege to express the gratitude to all those who guided in the completion of our Mini Project.

First and foremost, We wish to express our profound gratitude to our respected Principal **Dr. V.G. Sangam, B.L.D.E. Association's VACHANA PITAMAHA Dr. P.G. HALAKATTI COLLEGE OF ENGINEERING & TECHNOLOGY, Vijayapura** , for providing us with a congenial environment to work in.

We would like to express our sincere thanks to **Dr. U D Dixit** , the HOD of **Electronics and Communication Engineering**, **B.L.D.E. Association's VACHANA PITAMAHA Dr. P.G. HALAKATTI COLLEGE OF ENGINEERING & TECHNOLOGY, Vijayapura**, for his continuous support and encouragement.

We are greatly indebted to our guide **Prof. Praveenkumar Havalagi**, Department of **Electronics and Communication Engineering**, **B.L.D.E. Association's VACHANA PITAMAHA Dr. P.G. HALAKATTI COLLEGE OF ENGINEERING & TECHNOLOGY, Vijayapura**, who took great interest in our work. He motivated us and guided us throughout the accomplishment of this goal. We express our profound thanks for his meticulous guidance.

ABSTRACT

In parallel communication, due to simultaneous transmission of data bits the cost and complexity increases. Serial communication removes this drawback and shows effective results for long distance communication. UART (Universal Asynchronous Receiver Transmitter) is referred as serial communication protocol. This report contains implementation of UART with different baud rates. UART comprises three main modules baud rate generator, transmitter, receiver. Further, it is tested on hardware boards i.e. FPGA. We used Virtex 5 board for testing of UART as it holds the requirement needed for our project. The language used to design UART is Verilog which is a hardware descriptive language. And tool used is Xilinx ISE design suit.The serial communication interface, which receives and transmits the serial data is called a UART (Universal Asynchronous Receiver-Transmitter).

RxD is the received serial data signal and TxD is transmitted data signal. In this project UART is implemented in virtex II pro FPGA chip due to low cost, high speed, reprogram ability and fast time to market.

CONTENTS

Abbreviations, Notations and Nomenclature:

UART	Universal Asynchronous Receiver Transmitter
FPGA	Field Programmable Gate Array
RTL	Register Transfer Logic
THR	Transmitter Hold Register
TSR	Transmitter Shift Register
RHR	Receiver Hold Register
RSR	Receiver Shift Register
FSM	Finite State Machine
LUT	Look Up Table
HDL	Hardware Description Language
PAR	Place and Route
LSB	Least Significant Bit
MSB	Most Significant Bit
FIFO	First In First Out

CHAPTER 1
INTRODUCTION

The FPGA consist of I/O processor (IOP) core and Front- End Subsystem. IOP is used to handle data communication between PC and Design Under Test (DUT) in the FPGA board. The IOP in UART module is utilized for serial communication.

UART stands for Universal Asynchronous Receiver Transmitter. UART is the combination of hardware used for serial communication between computer and any other peripheral devices. As we know, the processing of data in personal computers or any other peripheral devices is in parallel form as it ensures speed. But the data communication in these systems is in serial form, so first the data in parallel form is to be converted to serial form. To do this process, UART is responsible for breaking parallel data into serial form and again convert it to parallel form at the receiver end.

UART is called ‘Universal’ because the data frame and transmission speed can be configured as per the requirement. And ‘Asynchronous’ as the transmitter and receiver are not synchronized by a clock signal. UART needs two signals rx_data and tx_data. Tx_data is output of the UART and rx_data is input to UART. In UART, it consists of two extra bits start and stop bit. When data is sent ‘start bit’ is inserted at the beginning of the data frame and stop bit is inserted at the end of the data frame. The start bit sends a signal to the receiver telling that the data bits are to be sent and force the clock to synchronize with the transmitter clock.

After adding the start bit, the data bits are sent one by one with first sending LSB and then remaining bits. In receiver, the data bits are counted bit by bit using the counter. After every data bit is received it is converted into parallel form and transmitted to other peripheral devices

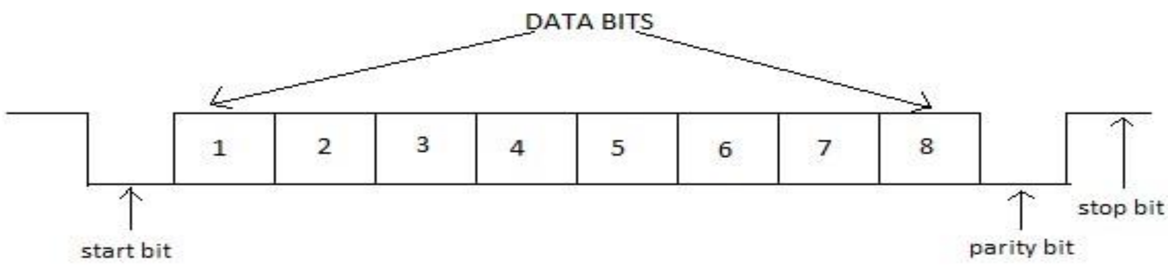


Fig. 1.1 Basic data frame of UART

The clock synchronization is absent between transmitter and receiver. To synchronize transmitter and receiver start bit and stop bit is used. UART design for FPGA can act as an interface for FPGA based embedded system which used soft core processors. This reduces external routing problems and cost as FPGA contains huge number of logic gates unused. Using FPGA can be good solutions for reconfiguration of the hardware devices.

UART also provides the basic operations as:

- Converts the bytes it gets from the computer along parallel circuits to a single serial bit stream for outbound transmission.
- For inbound transmission, converts the serial bit stream to the bytes that the system handles.
- Adds a parity bit after selection in outbound transmissions, checks the parity of incoming bytes (if selected) and rejects the parity bit.
- Adds start and stop delineators for outbound and helps to strip them from inbound transmissions.
- Handles interrupts from keyboard and mouse (which are serial devices with special ports).

CHAPTER 2
LITERATURE SURVEY

An electronic system requires a medium for communicating with the external world. It can transfer data to another device, sending or receiving commands or just for debugging purpose. The most common interface used is UART. As UART works on serial transmission concept. In early days RS232 interface where used to connect serial port. But now many PC you will come across has no serial port, so it is commonly implemented using USB, however transmission concept is same. In UART main tasks are firstly to convert parallel data coming from data bus in serial form. Then transmitting the serial data, finally after receiving data converting it again into parallel form.

UART is used for serial communication. The UART has 3 main blocks transmitter, receiver and baud rate generator. The first UART was designed by Western Digital in 1971. The first UART is designed by Gordon Bell for PDP series of computers. The main innovation was this UART was using sampling to convert signal to digital domain, providing more steady timing than previous telegraphic circuits. To reduce cost of wiring and other components, DEC shortened the line unit design into a single chip, this was first UART.

In 1980s National Semiconductor 8250 UART were used. 8250 UART has an onchip programmable bit rate generator which allowed to transmit data at any frequency. These UART were used in IBM PC5150. To avoid problem such as interrupts and to increase efficiency, Intel designed a UART 82510 which has two independent 4-byte FIFOs. They introduced this UART in 1993. It allowed operation up to 288 Kbit/s. After Intel's UART many companies designed various UART with different capacities of FIFO buffers, and different speed of transmission. Further, NXP Semiconductor introduced dual UART (DUART), comprising of two channels for communication, control registers and counter/timer. The channels are independently programmable and supports independent transmission and reception. Some dual UART model are SCC2692, SC26C92, SC28L92, etc. By mounting the dual UART in common package NXP also designed Quad UART and Octal UART.

The authors created an energy-efficient instruction register for integrating green communication on Virtex 4, Virtex 5, and Virtex 6 FPGAs [9]. As a result, while much work has been done to incorporate the ideas of green communication and energy-/power-efficient devices for future generations on CU with various FPGAs, no work has been done to implement the CU circuit on Kintex-7 ultrascale FPGA, so in this work, the CU circuit is being designed on Kintex-7 ultrascale FPGA to promote green communication techniques.To provide a high-performance

FIFO for the CPU while reducing power usage, Saxena et al. [10] employed voltages and frequency scaling techniques to create FPGA-based FIFO architecture. They altered frequency from 20 MHz to 250 MHz while keeping the voltage constant at 2.3 volt, while for the other experiments, they maintained the frequency constant while varying voltages from 1 volt to 2.3 volt. They concluded that the voltage scaling reduces power usage by 95.79 percent, whereas frequency scaling reduces power consumption by 4.38 percent.

Kumar et al. [1] proposed a Spartan-3 and Spartan-6 fieldprogrammable gate arrays that are used to create a low-power transceiver (FPGA). As a transceiver, a universal asynchronous receiver transmitter (UART) device is employed. The power analysis findings are aimed on Spartan-3 and Spartan6 FPGAs, and the implementation of UART is achievable with EDA tools named Xilinx 14.1. By altering the voltage supply, the change of different power of chips built on FPGA is noticed, for example, input/output (I/O) power consumption, leakage power absorption, signal power utilization, logic power utilization, and the use of complete power. This study examines how voltage changes affect the power consumption of the UART on Spartan-3 and Spartan-6 FPGA devices. Spartan-6 is proven to be more power efficient as the voltage supply is increased.

In the current international situation, the global energy crisis is a very serious concern. The energy crisis in India, as well as a scarcity of natural resources such as crude oil, coal, and other minerals, has an impact on the country’s economy

2.1 Specification Extraction

Following are the specification extracted and implemented

1. Mode: Full Duplex.
2. Data Frame Sequence: 8 bits
3. Primary Frequency: 50MHz
4. Maximum Operating Frequency: 100MHz
5. 1 start bit, 1 stop bit.
6. No. of flags : 2.

CHAPTER 3

PRE-REQUIREMENTS OF PROJECT

3.1 Concept of Project

Before starting the project, one should look upon the pre-requirements needed for their project. In this project, pre-requirements for UART Implementation using FPGA includes the study of serial communication, how data is transferred from peripheral devices to computer and vise-versa. UART transmitter starts the transmission by taking a data frame in parallel form and advising the UART to transmit the data frame in a serial form. Similarly, the receiver must detect transmission and receive the data in serial form, remove the start bit and stop bit, and store the data frame in parallel form. As the project states, UART is Universal Asynchronous Receiver Transmitter, we should know the difference between synchronous and asynchronous system. There are some main terms that we should study before starting the project work and the main terms includes UART, baud rate, data frame, etc. We should study the main block of UART separately - transmitter, receiver and baud rate generator. As this will have registers, flip flops, latches, counters and others one should know Area optimization technique so that less amount of area must be used on the board. Now, we will study some concepts that we need in this project.

3.2 What is UART?

UART stands for Universal Asynchronous Receiver Transmitter which is a serial communication interface protocol. UART sends and receives data in the form of chunks or packets which are known as “transmission characters”. There are two types of communications in UART, half duplex and full duplex communications. In full duplex, both ends can communicate with each other simultaneously. For example, Telephone; users at both ends can speak and listen to each other simultaneously. With reference to UART, in full duplex communication one end can transmit as well as receive the bits. In half duplex system, both ends communicate each other but one user at a time. For example, walkie-talkie; in which one user speaks and other only listen and vice-versa. Half duplex is used to conserve bandwidth. In UART at a time one end can be active either transmitting or receiving end

.

3.3 Synchronous and Asynchronous transmission

In synchronous transmission, data is transmitted in form of packets or frames. Synchronous transmission follows full duplex type of communication. There is synchronization between transmitter and the receiver. In synchronous transmission, there is no gap between two data frames. It is more efficient and trustworthy to send large data..

In asynchronous transmission, data is transmitted in form of byte or characters. Asynchronous transmission follows half duplex type of communication. Synchronization is not required. In this transmission, start bit and stop is added to the data frame. Therefore, there is some gap between two data frames. The transmission is coordinated with no clock signal, asynchronous transmission does not depend on any clock signal.

Table 3.1 synchronous and asynchronous transmission

Synchronous transmission	Asynchronous transmission
1. Data is transmitted in the form of packets or frames	1. Data is transmitted in the form of byte or character.
2. Synchronous transmission is much faster than asynchronous	2. Asynchronous transmission is slower than synchronous.
3. Synchronous transmission is not economical.	3. Asynchronous transmission is economical.
4. There is constant time interval in transmission.	4. There is no constant time interval in transmission.
5. There is no gap between two data frames	5. There is gap between data frames as start bit and stop bit is added to the frame.

3.4 Serial Communication

In serial communication, the data is sent one bit at a time, sequential over the communication channel. It is opposite to parallel communication, where all the data bits are sent all at a time. In serial communication, data is in binary pulse form. Binary '1' represents high logic and binary '0' represent low logic. Serial communication can be of many types depending on the type of transmission mode. It has 3 transmission modes, simplex mode, half duplex, full duplex.

a) Simplex mode:

The simplex mode is one-way communication technique. Only one user (either the transmitter or receiver is active at a time). If transmitter transmits, the receiver can also accept. For example, Radio and Television.

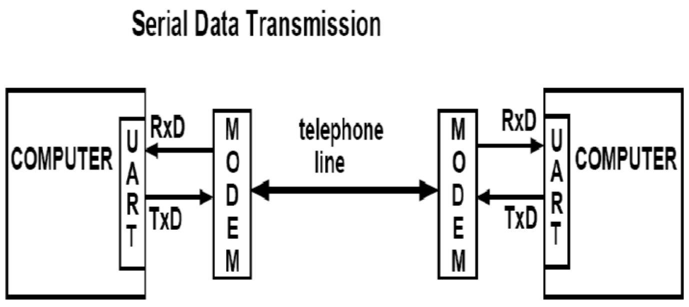


Fig. 3.1. Serial data transmission.

b) Half Duplex:

Both transmitter and receiver in half duplex mode are active but one at a time that means if transmitter transmits, the receiver can accept the data but cannot send and viceversa. For example, internet. If a user sends a request for any website, the web server processes the application and sends back the information

c) Full Duplex:

Full duplex mode is widely used communication mode in the world. In this mode, transmitter and receiver can transmit and receive simultaneously. For example, smartphone.

3.5 Tools used in project

Another important aspect to be known before is the language we are using, in this we are using Verilog. Some may use VHDL or any other language. We must have brief knowledge of Verilog. Software used here is Xilinx ISE design suite 14.7, we can also use Vivado which is another tool for writing Verilog codes. Tool should be selected by looking the specifications or the package we will be requiring or the FPGA board we will be using, it can be of Virtex 5, Spartan 3E, Spartan 3A, Spartan 6, etc. These boards have their specification, we should choose board matching our requirements. For pin description and other details about FPGA boards refer the data sheet.

CHAPTER 4

PROPOSED WORK

4.1 Top level diagram

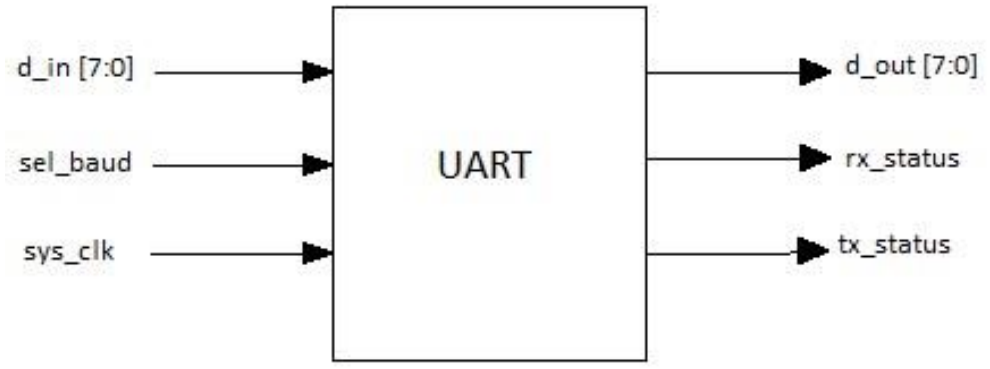


Fig. 4.1 Top level diagram of UART

Table 4.1 Pin description for top level diagram of UART.

Pins	Width	I/O	Description
d_in	8 bits	Input	Input data from data bus.
sel_baud	2 bits	Input	Used to select different baud set for different values of pin.
sys_clk	1 bit	Input	Input clock to UART from FPGA board on which it runs.
d_out	8 bits	Output	Output data sent to data bus.
rx_status	1 bit	Output	Shows status of receiver.
tx_status	1 bit	Output	Shows status of transmitter.

4.2 Architecture

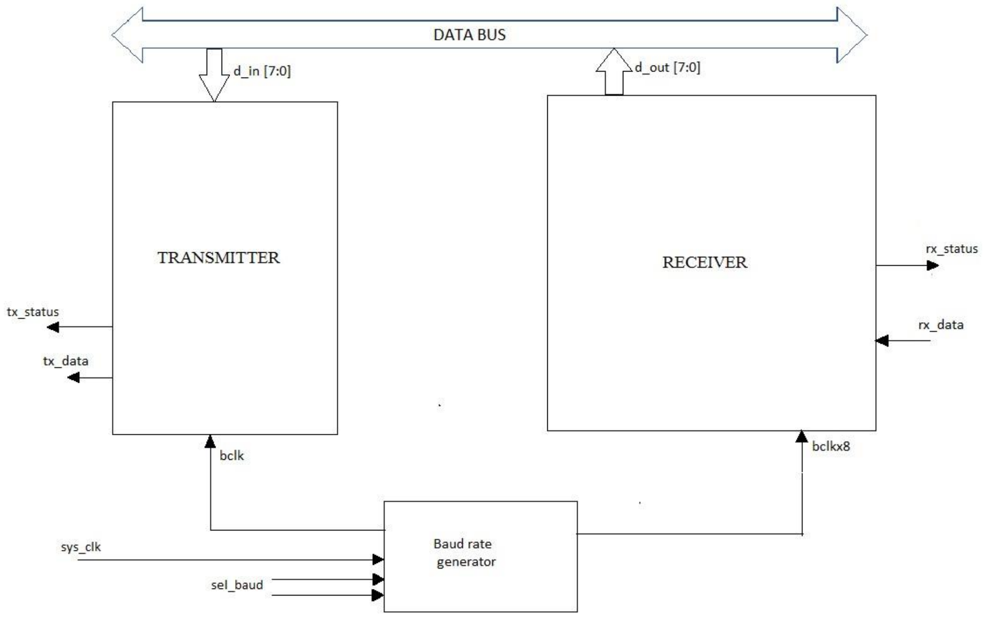


Fig. 4.2 Architecture of UART

UART architecture consists of three main blocks Transmitter, Receiver and Baud rate generator. The data is taken from the computer or the peripheral devices. Below are the block level micro-architecture and working of the modules.

4.2.1 Transmitter Module

Block level micro-architecture of transmitter module is given below,

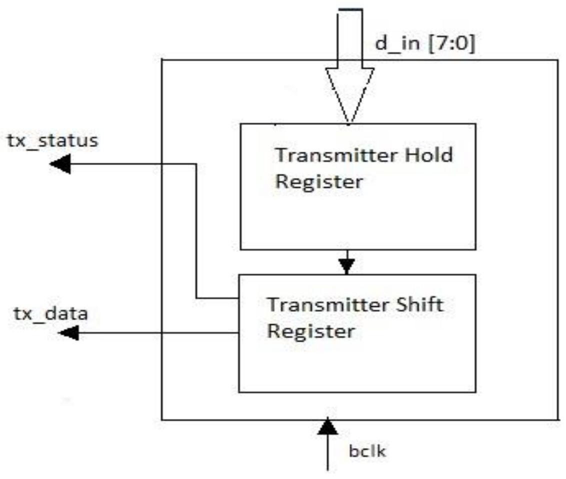


Fig. 4.3

4.2 Micro-architecture of transmitter module
Table 4.2

Pin description of transmitter module.

Pins	Width	I/O	Description
d_in	8 bits	Input	Input to the transmitter form data bus.
tx_status	1 bit	Output	Shows the status of the transmitter module.
tx_data	1 bit	Output	Sends data to receiver bit by bit.
bclk	1 bit	Input	Input clock to transmitter module from baud rate generator.

All the operations in transmitter are with respect to clock which runs on multiple baud rates. The transmitter converts the 8-bit parallel data into serial data and adds start bit at start of the data frame and stop bit at the end of the data frame.

In transmitter, the 8-bit data from data bus is loaded to transmitter hold register (THR), it is an 8-bit register to store the data parallelly through d_in pin. Then after all the data is loaded in THR then data is loaded in Transmitter shift Register (TSR) parallelly. In TSR, start bit and stop bit is added to the data bits, transmitter shift register is of 10-bit size. TSR convert parallel data into serial form and then data is transmitted serially bit by bit through tx_out. While transmitting data LSB of the data bit is transmitted first and then remaining data is transmitted and last stop bit is transmitted to stop the transmission. The clock given to transmitter block is the baud clock (bclk) on which the baud rate is set. Counter is used in TSR to count the data bits. Until and unless all the data is transmitted no other data can be stored in hold register.
In this project, transmitter is designed based of FSM used as given below,

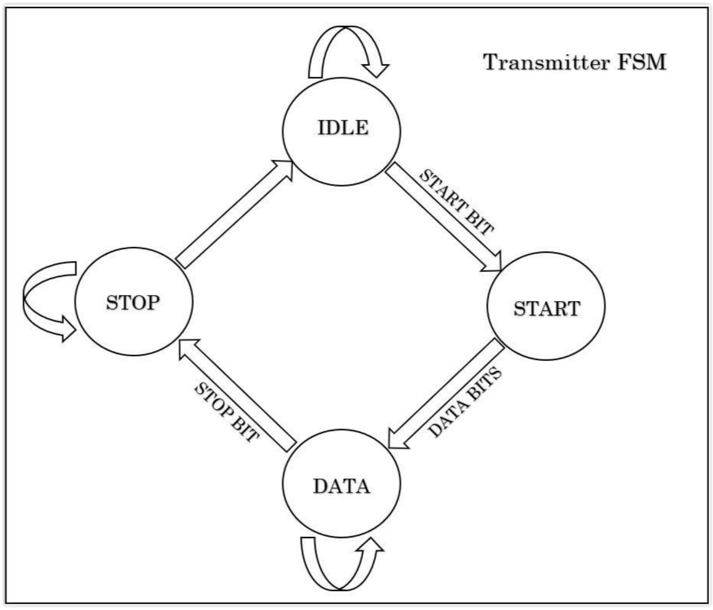


Fig. 4.4 FSM for transmitter module.

Transmitter FSM have four states they are named as follows:

- Idle State.
- Start State. • Data State.
- Stop State.

In transmitter FSM first state is idle state in which transmitter remains in idle condition means nothing will going to happen in this state. But when desired input is given to this state then there is change of state takes place. Here when valid start bit (logic ‘0’) is given as input then state changes from idle to start state means in idle state there is a detection of valid start bit. When invalid input is given in idle state then transmitter remains in same state i.e. idle state. Output of this state is logic ‘0’ for valid input (logic ‘0’).

When transmitter is in start state input is logic ‘0’ which is valid so there is change of state from the start state to data state. If input is not logic ‘0’ then transmitter goes in idle state. Start state will not change to data state until valid start bit is detected. In data state transmitter sends frame sequence given to the input of transmitter and when complete frame sequence is transmitted then state changes to stop state otherwise transmitter will remain in same state until the complete transmission of frame sequence.

Stop state of transmitter will detect stop bit (logic ‘1’). If there will be any error in stop bit, then transmitter remains in stop state only otherwise it will again go to idle state to detect another start bit. The above process continues till required number of bits will be transmitted.

4.2.2 Receiver Module

Block level micro-architecture of receiver module is given below,

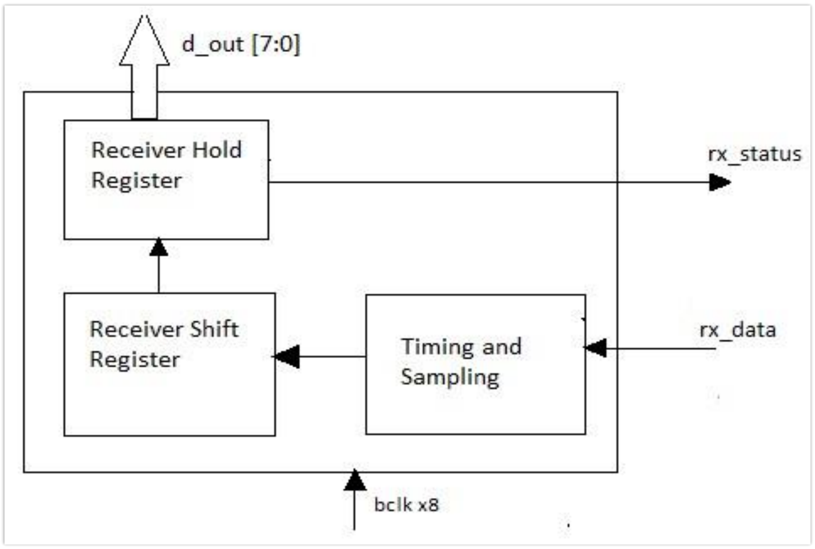


Fig. 4.5 Micro-architecture of receiver module.

Table 4.3 Pin description for receiver module.

Pin	Width	I/O	Description
rx_data	1 bit	Input	Input data from transmitter.
bclkx8	1 bit	Input	Input clock to receiver from baud rate generator. Receiver clock is 8 times faster than transmitter clock.
rx_status	1 bit	Output	Shows status of receiver block.
d_out	8 bit	Output	Sends output data to PC or peripheral devices.

The receiver module is complex than the transmitter module. It runs 4 X baud rate faster than transmitter. The serial data is received from transmitter through rx_data pin 1-bit at a time serially. Then, rx_data pin jumps into logic 0 from logic 1 indicating beginning of the data frame. The start bit is identified by change in level from high to low level and stop bit by change in the level from low to high. After identifying start bit, all the data bits are sampled and counted using the bit counter generated in the receiver. The counter counts the positive edge of the clock at every riding edge of the clock 1 bit is counted. When count equals to 8 then it says all bits are received and stored in an 8 bit Receiver Shift Register (RSR) and the data bits are converted into parallel data. Then it sends the data bits to Receiver Hold Register (RHR) which is also an 8 bit register. RSR send only data bits to RHR, start bit and stop bit is not sent. Then the parallel data is sent to the data bus or peripheral device.

In this project, Receiver is designed based on FSM used as given below,

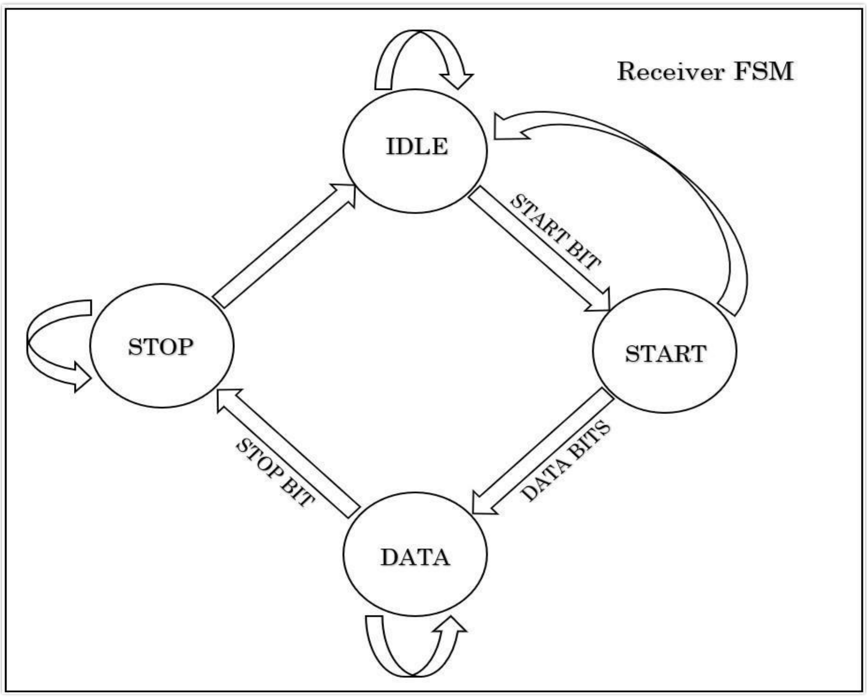


Fig. 4.6 FSM for receiver module

Receiver FSM also have same four states like transmitter they are named as follows:

- Idle State.
- Start State. • Data State.
- Stop State.

In receiver FSM first state is idle state in which receiver remains in idle condition means nothing will going to happen in this state. But when desired input is given to this state then there is change of state takes place. Here when valid start bit (logic ‘0’) detected idle state changes from idle to start state. When wrong bit is received in idle state then receiver remains in idle state. Output of this state is logic ‘0’ for valid input (logic ‘0’).

When receiver is in start state then it will not detect any bit because valid bit is already detected in idle state, so receiver simply changes it’s states from start to data state. In data state, if valid frame sequence received then there will be change in state from data to stop state otherwise receiver will remain in same state until the arrival of valid data.

Stop state of receiver will detect stop bit (logic ‘1’). If there will be any error in stop bit, then receiver remains in stop state only otherwise it will again go to idle state to detect another start bit. The above process continues till required number of bits will be received.

Both transmitter and receiver works on mealy machine means output depends on state as well as input. There is change in output when any one of them changes (state or input).

CHAPTER 5
EXPERIMENTAL RESULTS

5.1Transmitter code

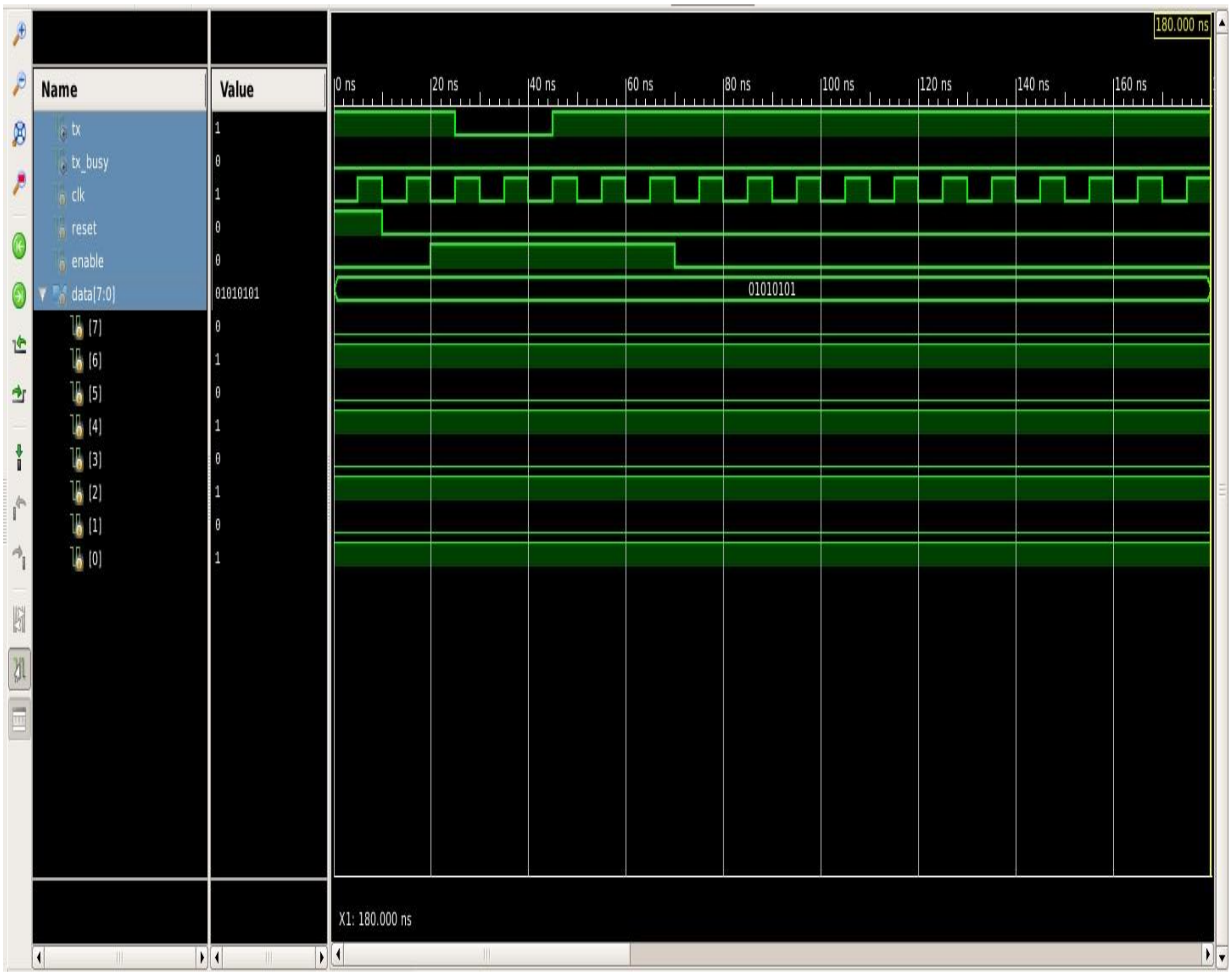
```
// UART Transmitter code
module uart_transmitter (
 input clk, // System clock
 input reset, // Reset signal
 input enable, // Enable signal
 input [7:0] data, // Data input
 output reg tx, // Transmitter output
 output reg tx_busy // Transmitter busy flag
);

 reg [3:0] bit_count;
 reg [10:0] shift_reg;
 reg start_bit;
 reg stop_bit;

 always @(posedge clk or posedge reset) begin
  if (reset) begin
   tx <= 1'b1; // Idle state
   tx_busy <= 1'b0;
   bit_count <= 4'b0;
   shift_reg <= 11'b0;
   start_bit <= 1'b0;
   stop_bit <= 1'b0;
  end
```

```
else if (enable) begin
 if (bit_count == 4'd0) begin
  // Start bit
  tx <= 1'b0;
  start_bit <= 1'b1;
  bit_count <= bit_count + 1;
 end
 else if (bit_count >= 4'd1 && bit_count <= 4'd8) begin
  // Data bits
  tx <= shift_reg[0];
  shift_reg <= {shift_reg[9:0], data};
  bit_count <= bit_count + 1;
 end
 else if (bit_count == 4'd9) begin
  // Stop bit
  tx <= 1'b1;
  stop_bit <= 1'b1;
  bit_count <= bit_count + 1;
 end
 else if (bit_count == 4'd10) begin
  // End of transmission
  tx_busy <= 1'b0;
  bit_count <= 4'd0;
 end
end
else begin
 tx_busy <= 1'b0;
     end
      end
        endmodule
```

5.1.1 Test-bench Transmitter

```
// Testbench for UART Transmitter
module uart_transmitter_tb;

  // Inputs
  reg clk;
  reg reset;
  reg enable;
  reg [7:0] data;

  // Outputs
  wire tx;
  wire tx_busy;

  // Instantiate the UART Transmitter
  uart_transmitter dut (
    .clk(clk),
    .reset(reset),
    .enable(enable),
    .data(data),
    .tx(tx),
    .tx_busy(tx_busy)
  );

  // Clock generation
  always begin
    #5 clk = ~clk;
  end
 // Testcase
 initial begin
  // Initialize inputs
  clk = 0;
  reset = 1;
  enable = 0;
  data = 8'b01010101;

  // Reset
  #10 reset = 0;

  // Transmit data
  #10 enable = 1;
  #50 enable = 0;

  // Wait for transmission completion
  #100;

  // End simulation
  #10 $finish;
 end

endmodule
```

5.1.2 Waveform -Transmitter



Name	Value
tx	1
tx_busy	0
clk	1
reset	0
enable	0
data[7:0]	01010101
[7]	0
[6]	1
[5]	0
[4]	1
[3]	0
[2]	1
[1]	0
[0]	1

5.2 Receiver code

```
// UART Receiver
module uart_receiver (
 input clk, // System clock
 input reset, // Reset signal
 input rx, // Receiver input
 output reg enable, // Enable signal
 output reg [7:0] data, // Received data
 output reg rx_valid // Receiver valid flag
);

 reg [3:0] bit_count;
 reg [10:0] shift_reg;
 reg start_bit;
 reg stop_bit;

 always @(posedge clk or posedge reset) begin
  if (reset) begin
   enable <= 1'b0;
   data <= 8'b0;
   rx_valid <= 1'b0;
   bit_count <= 4'b0;
   shift_reg <= 11'b0;
   start_bit <= 1'b0;
   stop_bit <= 1'b0;
  end
  else begin
   if (rx == 1'b0 && !start_bit && !stop_bit) begin
    // Start bit detected
    enable <= 1'b1;
    bit_count <= 4'd1;
    start_bit <= 1'b1;
    shift_reg <= {shift_reg[9:0], rx};
   end
   else if (enable && bit_count >= 4'd1 && bit_count <= 4'd8) begin
    // Data bits
    shift_reg <= {shift_reg[9:0], rx};
    bit_count <= bit_count + 1;
   end
   else if (enable && bit_count == 4'd9) begin
    // Stop bit detected
    enable <= 1'b0;
    rx_valid <= 1'b1;
    stop_bit <= 1'b1;
    data <= shift_reg[7:0];
    bit_count <= bit_count + 1;
   end
   else if (rx_valid && bit_count == 4'd10) begin
    // End of reception
```

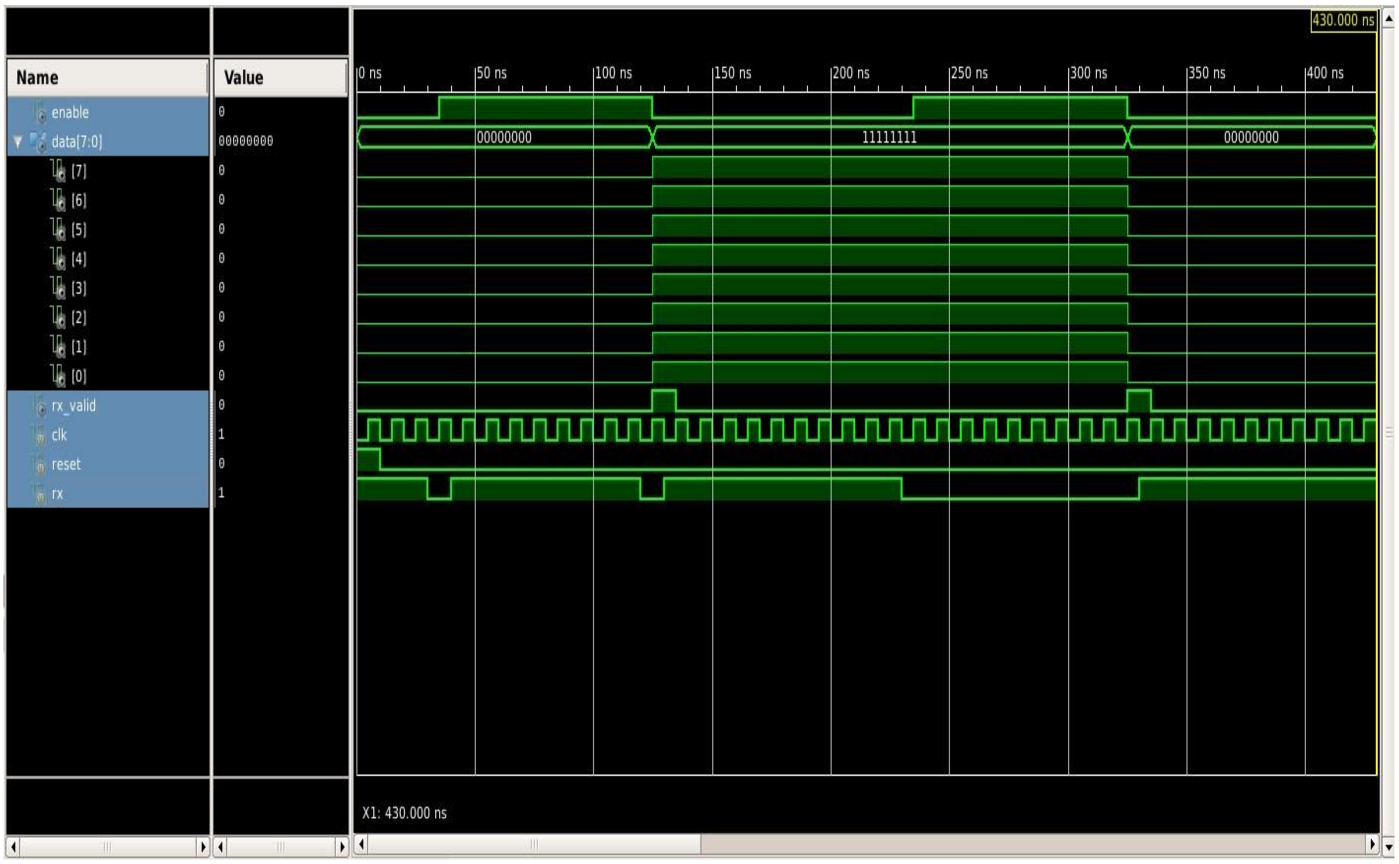
```
        rx_valid <= 1'b0;
        bit_count <= 4'd0;
      end
      else begin
        // Idle state
        enable <= 1'b0;
        start_bit <= 1'b0;
        stop_bit <= 1'b0;
      end
    end
  end

endmodule
```

5.2.1 Test-bench Receiver

```
//UART RECEIVER TESTBENCH
`timescale 1ns/1ps

module uart_receiver_tb;

 reg clk;
 reg reset;
 reg rx;
 wire enable;
 wire [7:0] data;
 wire rx_valid;

 // Instantiate the UART receiver module
 uart_receiver uart_inst (
  .clk(clk),
  .reset(reset),
  .rx(rx),
  .enable(enable),
  .data(data),
  .rx_valid(rx_valid)
 );

 // Clock generation
 always #5 clk = ~clk;

 // Testbench stimuli
 initial begin
  clk = 0;
  reset = 1;
  rx = 1;
  #10 reset = 0;

// Wait for a few cycles for initialization
#20;

// Test 1: Receive byte 0x37
$display("Test 1: Receive byte 0x37");
rx = 0; // Start bit
#10;
rx = 1; // Data bits: 00110111 (LSB first)
#80;
rx = 0; // Stop bit
#10;
rx = 1; // Idle state

// Wait for reception to complete
#100;

// Verify the received data
if (enable && rx_valid && data === 8'h37)
```

```
      $display("Test 1: PASSED - Correct byte received");
    else
      $display("Test 1: FAILED - Incorrect byte received");

    // Test 2: Receive byte 0xAA
    $display("Test 2: Receive byte 0xAA");
    rx = 0; // Start bit
    #10;
    rx = 0; // Data bits: 10101010 (LSB first)
    #80;
    rx = 0; // Stop bit
    #10;
    rx = 1; // Idle state

    // Wait for reception to complete
    #100;

    // Verify the received data
    if (enable && rx_valid && data === 8'hAA)
      $display("Test 2: PASSED - Correct byte received");
    else
      $display("Test 2: FAILED - Incorrect byte received");

    // End the simulation
    $finish;
  end

endmodule
```

5.1.2 Waveform -Receiver



CHAPTER 6

Conclusion and Future Scope

6.1 Conclusion

In this project, UART is designed with configurable baud rates. The language used in this project is Verilog descriptive language to achieve reliable serial data communication. The Transmitter and Receiver module of the UART are designed using finite state machine concept and successfully synthesized using Xilinx ISE 14.7. The UART with different baud rates is successfully simulated and has being verified on Xilinx ISE 14.7. The design is compatible for high speed due different baud rates.

The architecture is successfully implemented in FPGA. Fast time to market, low cost for small production volume and reprogram ability make FPGA devices an ideal solution for military and university research. The high speed capability and register rich architecture of FPGA an ideal implementation of UART architecture. The design implementation entailed the employment of Xilinx ISE8.2 software tool. Implementing the design on a virtex-II and hardware testing and verification of the UART could be done. Finally simulation and synthesis with Xilinx xst tool and RTL schematic of filter chip was obtained

6.2 Future Scope

Thinking about improving the UART system, we can use status registers and error logic for modifications. Considering the individual blocks of UART after making the modifications.

UART (Universal Asynchronous Receiver-Transmitter) has a promising future scope due to its simplicity, versatility, and widespread adoption. It is expected to play a significant role in IoT connectivity, embedded systems, legacy device communication, automotive systems, wireless communication, and real-time data transfer. While newer communication interfaces exist, UART's simplicity, low cost, and compatibility ensure its continued relevance in various applications.

7. REFERENCES

[1] Ms.Neha R. Laddha, Prof. A. P. Thakare, “A Review on Serial Communication by UART”,Volume 3, Issue 1, January 2013 International Journal of Advanced Research in Computer Science and Software Engineering.

[2] Kavyashree S, “Design and Implementation of UART using Verilog”, International Journal
Of Engineering And Computer Science ISSN:2319-7242 Volume – 4 Issue - 12 December, 2015 Page No. 15240-15245

[3] FANG Yi-yuan, CHEN Xue-jun, “Design and Simulation of UART Serial Communication Module Based on VHDL”, College of Electrical and Electronic Engineering Shanghai University of Engineering Science Shanghai, China

[4] KeyStone Architecture Universal Asynchronous Receiver/Transmitter (UART) User Guide by Texas Instruments.

[5] Hazim Kamal Ansari, Asad Suhail Farooqi, “ Design Of High Speed Uart For Programming Fpga”, International Journal Of Engineering And Computer Science Volume1 Issue 1 Oct 2012 Page No. 28-36

[6] Virtex 5 FPGA Starter Kit Board User Guide

[7] https://www.xilinx.com › products › silicon-devices › fpga

[8] Umakanta Nanda, Sushant Kumar Pattnaik, “Universal Asynchronous Receiver and Transmitter(UART)”, 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS -2016), Jan. 22 – 23, 2016, Coimbatore, INDIA

[9] Abhay Malviya, Vijay Kumar Sharma, “An Improved Approach of UART Implementation in VHDL using Status Register”, Volume 4, Issue 10, May 2016 International Journal of Digital Application & Contemporary Research.

[10] T. Das, B. Pandey, M. A. Rahman, and T. Kumar, “SSTL based green image ALU design on different FPGA,” in *2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, pp. 146–150, Chennai, India, 2013.