Design and Simulation of VHDL Based UART Using FSM

[1]Dr. (Mrs.) S. W. Varade, [2]Ms. Dipali Pujari, [3]Ms. Ashwini Raman,
[4]Ms. Rajashri Mankar and [5]Mr. Pratik Kalmengh

[1]Dr. (Mrs.) S. W. Varade, Electronics and Telecommunication Engg. Priyadarshini College of Engineering, Nagpur, Maharashtra, India
[1]swvarade@gmail.com
[2]Ms. Dipali Pujari, Electronics and Telecommunication Engg. Priyadarshini College of Engineering, Nagpur, Maharashtra, India
[2]dipalipujari17@gmail.com
[3]Ms. Ashwini Raman, Electronics and Telecommunication Engg. Priyadarshini College of Engineering, Nagpur, Maharashtra, India
[3]ashwini.raman65@gmail.com
[4]Ms. Rajashri Mankar, Electronics and Telecommunication Engg. Priyadarshini College of Engineering, Nagpur, Maharashtra, India
[4]rajashrimankar312@gmail.com
[5]Mr. Pratik Kalmengh, Electronics and Telecommunication Engg. Priyadarshini College of Engineering, Nagpur, Maharashtra, India
[5]pratikkalmengh2595@gmail.com

ABSTRACT

The proposed work discusses the designing and simulation of VHDL based UART where the control path is provided by FSM. In this research work basic UART modules are implemented via VHDL to achieve the error free and reliable data transmission. Top-down design methodology of VHDL is implemented to specify and verify high speed applications. In the result and simulation part, this project will concentrate on checking the received bits of data to ensure whether it is error free. The proposed research work aims at application design on UART. All modules are designed using VHDL on Xilinx software 14.5.

Keywords- UART, VHDL, FSM, FPGA, RTL, TDC, SOC.

1. INTRODUCTION

UART (Universal Asynchronous Receiver/Transmitter) is basically a serial communication protocol generally used for short distance, low speed and inexpensive data transfer between computer and peripherals. In UART data path and control path governing transmission and reception of data. During high speed transmission & reception of data UART stores the data temporarily in a buffer. This paper presents the accomplishment of high speed and efficient UART using FSM. The proposed paper is a software oriented project which is designed and simulated using Xilinx ISE 14.5.Basically UART is a full duplex communication, to achieve so it requires two signal lines namely; txd, rxd. The transmitter side i.e. the output side is represented by txd and the receiver side i.e. the input side id rxd. The advantage of asynchronous serial communication is that it requires less transmission lines, provides high reliability and supports long transmission, thereby, widely used.

2. RELATED WORK

A Low Power UART Design Based on Asynchronous Techniques:

Universal Asynchronous Receiver Transmitter (UART) is used for serial communication between the computer and its peripherals. This work check out whether the UART, which is a serial communication protocol, can be advantageously implemented using asynchronous design techniques. A full duplex clocked and asynchronous UART are implemented and examine in contrast. The implementation of UART using asynchronous techniques results in average power of about one fourth to that of the clocked design under standard operating modes.

Design of High Speed UART for Programming FPGA:

FPGA (Field Programmable Gate Array) devices are one of the latest technologies that are changing the electronics industry. The Time triggered communication in the FPGA is further improved with the help of UART. It converts between the serial and parallel bits of data. The second UART can be used to receive the information. In this paper a high speed UART is designed using VHDL which performs all the tasks, timing, parity checking, etc. needed for the communication. [2]

Design and Simulation of UART Serial Communication Protocol Based On VHDL:

In this Paper UART includes three important modules which are the baud rate generator, the receiver and the transmitter. In this paper the UART is design for the communication between FPGA and TDC (Treatment Delivery Controller). The UART frame consists of one start bit, 8 data bit and one stop bit. A UART is a communication device which is mainly used for communication between computer and peripheral devices. This project proposes the hardware implementation of VHDL based UART that have been modified to enhance its performance. This project was implemented in VHDL and the synthesis is done using Xilinx software and Spartan library. This design is highly integrated and has more flexibility. [3]

Design & Simulation of UART Serial Communication Module Based on VHDL:

In the process of manufacturing UART, sometimes we do not need the full functionality of UART instead of that just have to combine its main modules. UART has three main modules which are the baud rate generator, the receiver and the transmitter. The UART modules which are accomplished using VHDL language can be combined into the FPGA chip to achieve the reliable and stable transmission of data. Specific interface chip causes the waste of resources and increases the cost of the system hence to deal with this problem, now a day's SOC technology is extensively used. The result and simulation obtained in this paper is using Quartus II software. This design is of great importance in designing SOC. [4]

3. PROPOSED SYSTEM

Testing difficulties which are occurring due to rapid changes in the technology field has cause design and test engineers to take over the new responsibilities that had been performed by groups of technicians in the previous years. The below given fig. is the generic block diagram of UART. The UART is bifurcated as three sub modules: the baud rate generator, the receiver module and the transmitter module. UART receiver module receives serial data at rxd, and converts them into parallel data. The UART transmitter module accepts parallel data and converts them into the serial bits and transmits those bits through txd. This is shown in Fig. 1:

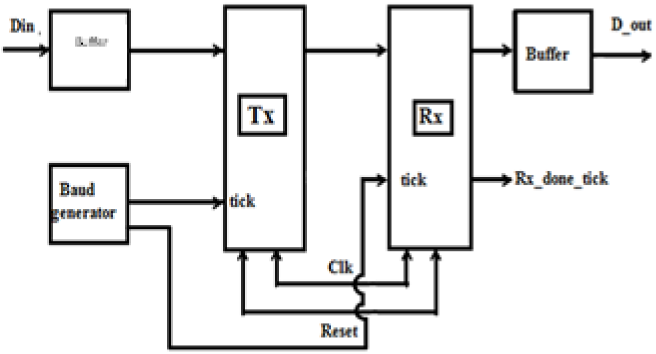


Fig. 1: Block Diagram of UART

Serial port is a universal parts of a computer through which we transmit serial. In UART serial port acts as a connector where serial line is attached in order to connect peripheral devices such as mouse, modem, printer, etc. The serial port is connected to, an integrated circuit which handles the conversion between parallel data and serial data.

4. UART PROTOCOL

UART frame format consist of start bit, data bit, parity bit, stop bit and idle state as shown in Fig. 2. The idle state or no data state is high-voltage, or powered. The first character sent here as a logic low start bit (or a logic high), which is accompanied by a configurable number of data bits, followed by an optional parity bit and a stop bit which is logic high (one or more).

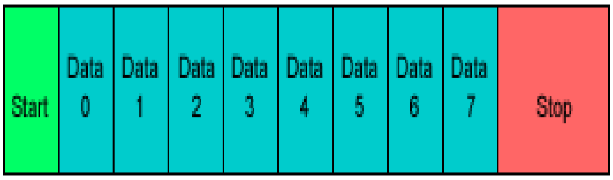


Fig. 2: UART Frame Format

The start bit signals the receiver about the arrival of a new character. Once the receiver is ready, the data bits (5 to 8 bits) containing the information is transmitted. After the information bits the following bit may be parity bit. The next bits are stop bits (logic high i.e. “1”).

5. THE UART MODULES

5.1 Transmitter:

The below given Fig. 3 is the UART Transmitter Module:

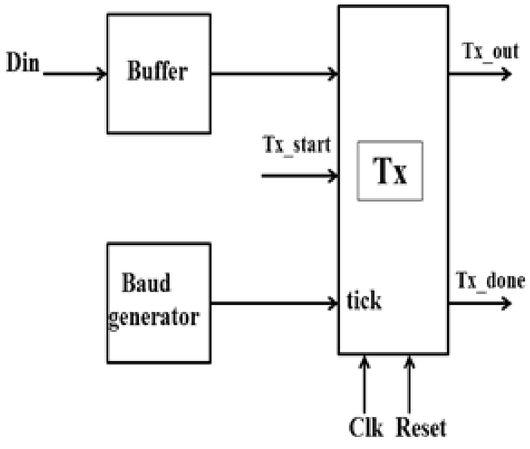


Fig. 3: UART Transmitter Module

UART Transmitter module inserts the start bit in the beginning of data, after data bits it adds an optional parity bit and a stop bit (1 or 2 bits) which indicate the end of the data frame. The transmitter basically converts the 8-bit parallel data into serial data. The below given is the RTL (Register Transfer Level) schematic of the transmitter module shown in Fig. 4:

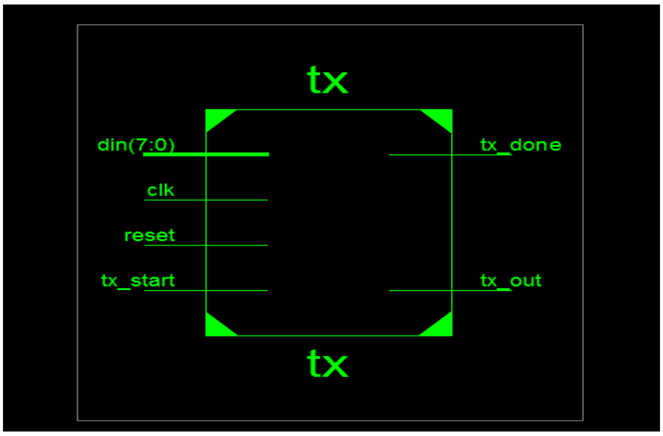


Fig. 4: RTL Schematic

In the starting the UART transmitter module is in idle state after getting the logic high on Tx line it starts accepting the data. If the parity bit is used which is basically of two types odd and even parity. It helps in analysing whether all the information bits are correctly received at the UART reception, it is determined by either number of logic 0 or logic 1 in the data i.e. logic 1 for 8-bit data (even parity) and logic zero for odd parity data. In the end of the data frame we have the, stop bit indicated by logic 1.

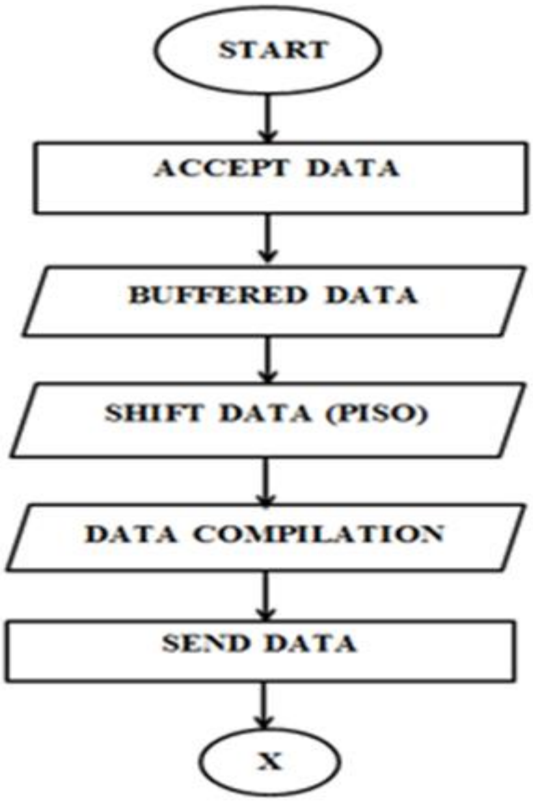


Fig. 5: Flowchart

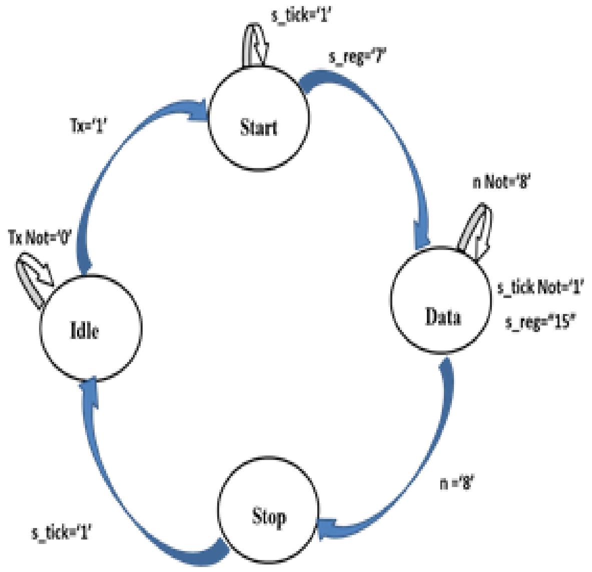


Fig. 6: State Diagram

The above depicted state diagram in FIG 10 states that the state machine has four states; IDLE, START, DATA and STOP.

IDLE State: When the UART reset the state machine will be in this state. If tx_start=1 then state machine transfer to START state and if not then state machine remains in IDEAL state.

START State: In this state, when s_reg =7 and s_tick=1then the state machine will move to data state and if not it remains in the same state.

DATA State: In this state, when s_tick is not equal to"1" and s_reg is equal to "15" the data will shift one by one till all the 8 bits are transmitted and stored in b_next register i.e. b_next<= {tx, b_reg[7:1]}. If all data bits are transmitted and s_tick=1 then the state machine will transfer to STOP state.

STOP State: In this state, if tx_done=1 then state machine transfers to IDLE state again.

5.2 Receiver:

The below given Fig. 7 is the UART Receiver Module. On each clock pulse, the state of incoming signal is tested i.e. start bit. Once the required data bits have received, the receiving system is allowed to access the contents of shift register parallel. It is very important to consider the aspect of determining the start bit of the frame correctly, as during UART reception the serial data and the receiving clock are asynchronous.

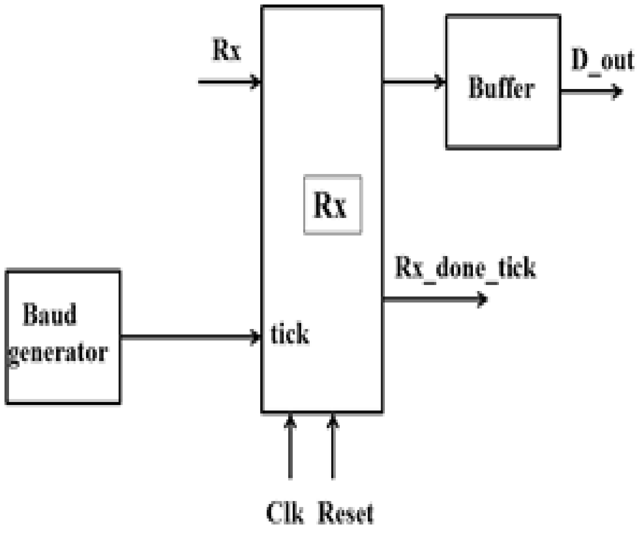


Fig. 7: UART Receiver Module

The receiver module receives data on rx pin. Thus we can regard start of data when rxd jumps into logic 0 from logic 1. In the process of waiting for the rxd level to jump, the UART receiver module gets reset. When rxd level jumps from logic high to logic low then the start bit is detected. The below given is the RTL (Register Transfer Level) schematic of receiver module shown in Fig. 8:

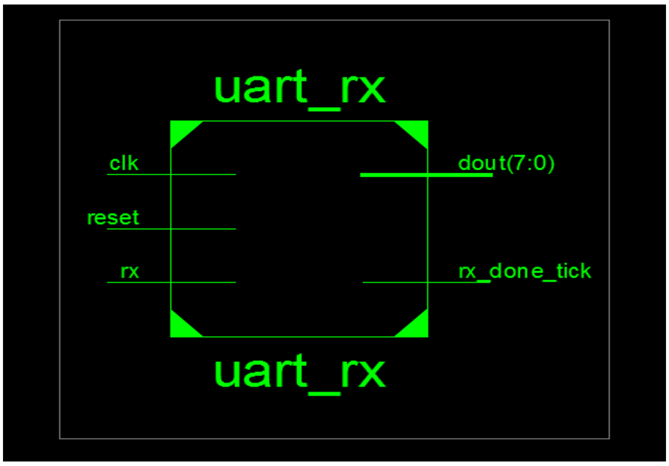


Fig. 8: RTL Schematic

We may state that the clock frequency (F) is 16 times the baud rate, when the start bit arrives rxd pin is low for8 consecutive receiving clock cycles. After the start bit has been identified, starting from the first data bit to the last data bit mid points of all data bits are checked in order to ensure that the tick is high which occurs for 16 clock cycles periodically. Once this process is completed i.e. all data bits are received then serial bit stream is converted into parallel.

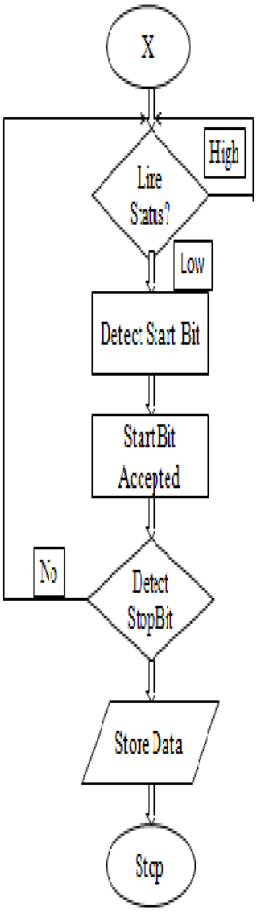


Fig. 9: Flow Chart

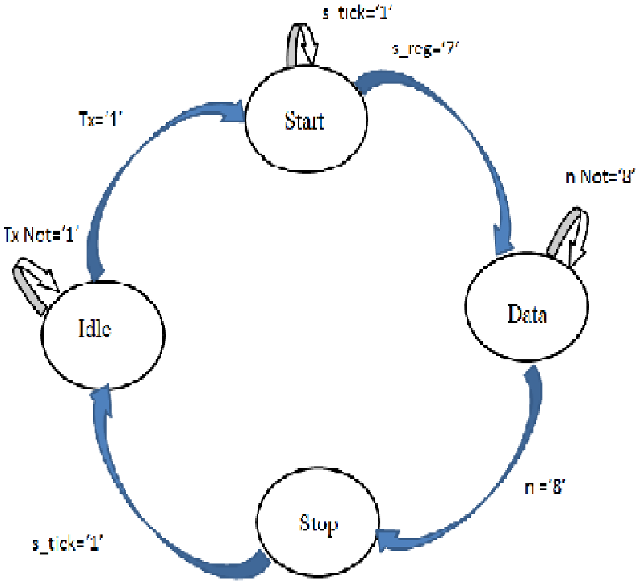


Fig. 10: State Diagram

The above depicted state diagram in FIG 6 states that the state machine has four states; IDLE, START, DATA and STOP.

IDLE State: When the UART reset the state machine will be in this state. If rx=0 then state machine transfer to START state and if not then state machine remains in IDEAL state.

START State: In this state, when s_reg =7 and s_tick=1 then the state machine will move to data state and if not it remains in the same state.

DATA State: In this state, when s_tick is not equal to "1" and s_reg is equal to "15" the data will shift one by one till all the 8 bits are received and stored in b_next register i.e. b_next<= {rx, b_reg[7:1]}. If all data bits are received and s_tick=1 then the state machine will transfer to STOP state.

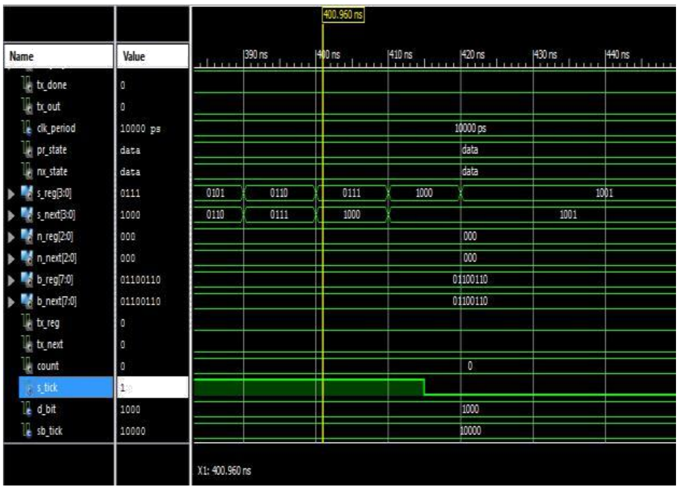
STOP State: In this state, if rx_done=1 then state machine transfers to IDLE state again.

6. SIMULATION RESULTS

The simulation software is XILINX 14.5 and selected device is SPARTAN 3.

6.1 Transmitter Module Simulation:

As the transmitter converts parallel data into serial data, according to the simulation report the timing diagram depicted contains a start bit, 8-bit input data, and a stop bit, in total transmitter contains 10 bit of data. Here the input data is 01001011. The transmission starts when the start bit is logic low. In order to terminate the transmission active high signal is passed on stop bit. The below given Fig. 10 is the simulation result of UART transmitter.



Name	Value
tx_done	0
tx_out	0
clk_period	10000 ps
pr_state	data
nx_state	data
s_reg[3:0]	0111
s_next[3:0]	1000
n_reg[2:0]	000
n_next[2:0]	000
b_reg[7:0]	01100110
b_next[7:0]	01100110
tx_reg	0
tx_next	0
count	0
s_tick	1
d_bit	1000
sb_tick	10000

Fig. 11: Simulation Result

6.2 Receiver Module Simulation:

Receiver converts parallel data into serial it takes the data from transmitter and compares whether the transmitted data is correct or not.

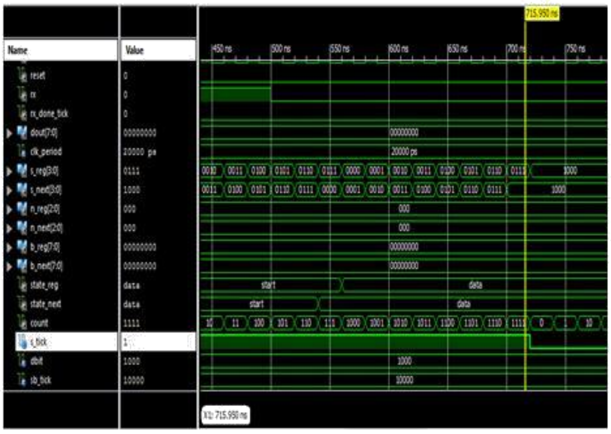
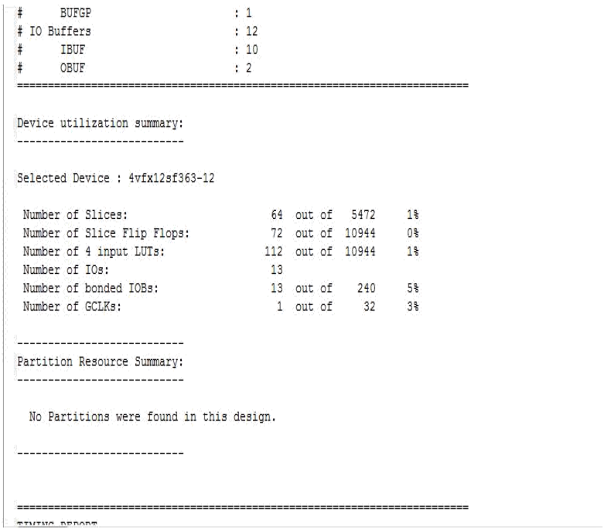


Fig. 11: Simulation Results

7. DEVICE UTILIZATION SUMMARY

The device utilization summary is generated by synthesis tool. It basically, gives an overview about the synthesis report. In that report several things like number of slice register, number of occupied slices, number of DSP slices, number of slice LTUs, etc. are mentioned. A description of what synthesis tool understood from the proposed designs is given in device utilization summary. Below given is the device utilization summary:

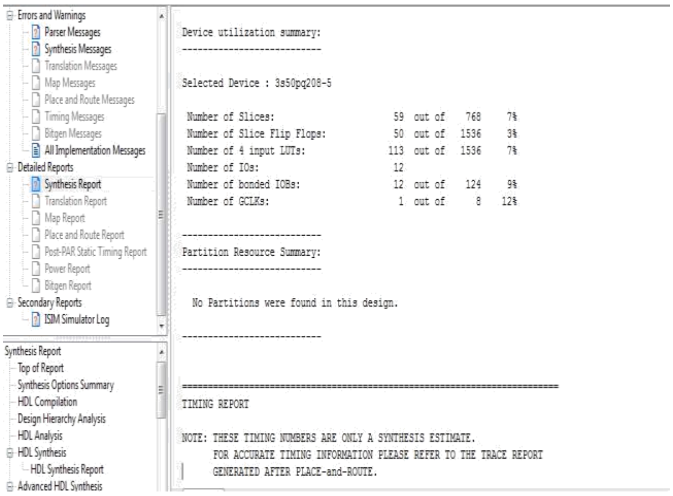
7.1 Transmitter Device Utilization Summary:



```
#      BUFGP                     : 1
# IO Buffers                     : 12
#      IBUF                      : 10
#      OBUF                      : 2
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 4vfx12sf363-12

 Number of Slices:                      64  out of   5472     1%
 Number of Slice Flip Flops:            72  out of  10944     0%
 Number of 4 input LUTs:               112  out of  10944     1%
 Number of IOs:                         13
 Number of bonded IOBs:                 13  out of    240     5%
 Number of GCLKs:                        1  out of     32     3%

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

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


=========================================================================
```

Fig. 12

7.2 Receiver Device Utilization Summary:



```
Device utilization summary:
---------------------------

Selected Device : 3s50pq208-5

 Number of Slices:                      59  out of    768     7%
 Number of Slice Flip Flops:            50  out of   1536     3%
 Number of 4 input LUTs:               113  out of   1536     7%
 Number of IOs:                         12
 Number of bonded IOBs:                 12  out of    124     9%
 Number of GCLKs:                        1  out of      8    12%

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

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


=========================================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.
```

Fig. 13

8. CONCLUSION

In this project VHDL is used as a designing language for the implementation of different UART modules. Using Xilinx 14.5 software the simulation results are obtained and are quite stable and reliable. The simulation results presented in this paper have proven the reliability of the VHDL to describe the characteristics and the architecture of UART.

REFERENCES

[1] Dipanjan Bhadra, Vikas S.Vij, Kenneth, S. Stevens “A Low Power UART Design Based on Asynchronous Techniques”, 2013 IEEE.

[2] Hazim Kamal Ansari, Asad Suhail Farooqi, “Design of High Speed UART for Programming FPGA”, International Journal of Engg. & Computer Science Vol. 1, Issue I, October 2012.

[3] Sumedha Laxmi, K. Hari Babu, “Design and Simulation of UART Serial Communication Protocol Based on VHDL”, International Journal of Advance Research and Innovation – Vol.7, Issue.III.

[4] Fang Yi-Yaun, Chen Xue-jun, “Design and Simulation of UART Serial Communication Module Based on VHDL”, Shanghai University of Engineering and Science, 2011.