ISSN 2347 - 3983
Volume 11. No.8, August 2023
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter021182023.pdf
https://doi.org/10.30534/ijeter/2023/021182023

# Implementation of High-Speed Serial I/O using Xilinx Tools in FPGA

**Munaganooru Shanthi[1], Returi Nehata Sreeya[2], Potukuchi Ajita[3]**
[1] Assistant Professor, shanthim@gnits.ac.in
[2]Student, sreeyareturi@gmail.com
[3]Student, ajita913@gmail.com
[1,2,3]Department of Communications and Electronic Engineering, G. Narayanamma Institute of Technology and Science, Hyderabad, India.

Received Date: June 30, 2023 Accepted Date: July 18, 2023 Published Date : August 07, 2023

**ABSTRACT**
The I/O (Input Output) module conveys the information between I/O device and processor. I/O devices are majorly of two types: Parallel I/O and Serial I/O. Parallel I/O performs multiple I/O operations simultaneously. Due to this speed and higher bandwidths are achieved, but the usage of parallel I/O devices is decreasing as time progresses because it involves complex design due to the usage of multiple wires for the transmission hence only limited to usage in shorter distances.It also uses a greater number of pins compared to serial I/O forthe same number of data bits which makes its usage problematic in higher level devices. Serial I/O transmits individual data bits sequentially. It uses lesser number of lines for data transmission thereby reducing the design complexity. Since, the data transmission is sequential the signal delay increases. Thus, this project aims to develop a protocol which achieves High Speed Serial I/O which helps to increase the data rate from Mbps to Gbps, decrease the design complexity, to design hardware using fewer number of pins on PCB and reduce signal delay to maximum extent possible.

**Key words :** AURORA Protocol, Delay, Speed.

## 1. INTRODUCTION

Serial I/O uses fewer number of pins decreasing the hardware design issues, but increases delay due to bit by- bit data transmission. Though Parallel I/O provides higher speed of data transmission, the complexity of hardware design increases reducing its effectiveness for usage. Thus, this project aims at the implementation of High-Speed Serial I/O which gives effective communication with minimum delay and reduces the design complexity, as referred from [1].

## 2. GENERATION OF AURORA 8B/10B CORE

Start the CORE Generator software, choose a new file and give a file name and click generate by assigning the required values to the parameters. The Aurora 8B/10B core provides a quick way to simulate and observe the behavior of the core using the provided example design, as referred from [3] and [4].

### 2.1 Simulation of AURORA Protocol

Simulation of protocol is done (as shown in Figure1) and the following bugs were encountered, and they were rectified through back tracing the source code module.

**Channel Initialization Error:** After simulation, we found that the lane_up signal(It is asserted upon successful lane initialization) was undefined and thus the channel_up signal(It is set to high when Aurora channel is in line to send data) was also found to be undefined.Since the channel_up is deasserted, data transfer could not take place. Through back-tracing we asserted the input signal of lane_up. Thus the data transfer was successful.

**D flipflop error:** UNISIM is the library that is included in the source code. Inclusion of this library provided the correct results eradicating the D- Flipflop problem which arised due to the FDR component in D-Flip Flop code module.

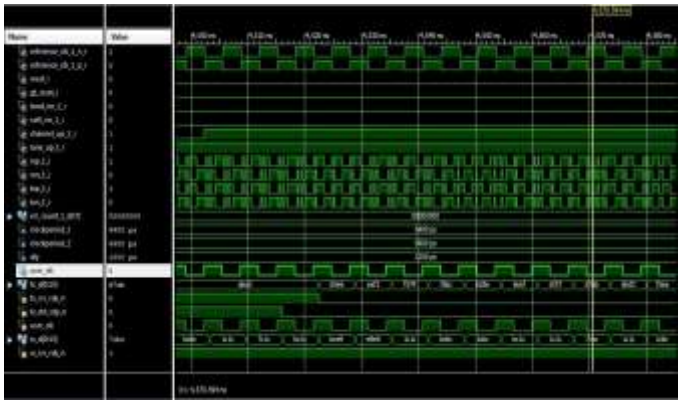

**Figure 1:** Simulation of Aurora protocol

## 3. IMPLEMENTATION, INSERTION OF GLUE LOGIC AND TESTING

After the simulation of the AURORA protocol, Up-Counter logic is inserted for transmission of data instead of previous logic that transmitted random seed data. As referred from [5] and [7]. After replacement of transmission logic, the module is simulated again to verify the results. With the up-counter logic the data transmission took place with a delay of 40 clock cycles, as observed in the Figure 2.

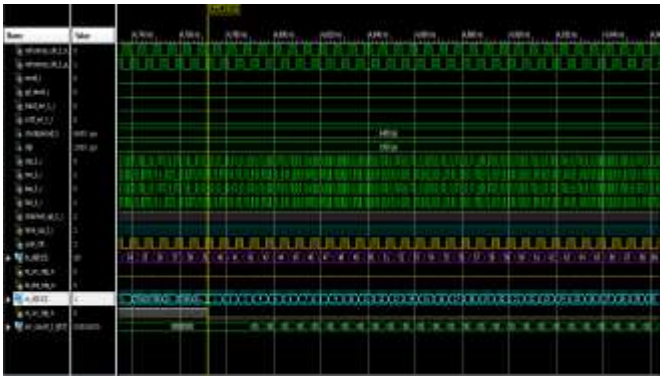

**Figure 2:** Data transmission window

## 4. PORTING CODE ONTO FPGA

The FPGA design flow for porting code on FPGA is as follows, as referred from [2]:

- Simulation, Implementation, Synthesize
- Synthesize: .vhd file to .ngc file
- Check syntax, Implement Design
- Translate: .ngc file to .ngd file
- Map: .ngd file to .pcf file
- Place & Routes: Ensure that Setup and Holdparameters should be equal to 0.
- Generating Programming File: Ensure that there are no software errors in the code and hardware device isdetected.
- Boundary Scan: To check whether the chip I/O pins are working properly or not. It will test the functionalityof pins by giving some random test pattern.

### 4.1 Defining Pins in User Constraint File

Specific pin numbers should be assigned to the defined signals on FPGA as follows as referred from [6]

- INIT_CLK is assigned AH7 pin.
- GTXD0_P and GTXD0_N are assigned H4, H3 locations.
- Reset is assigned at location E9 and it is given as aPULL_UP
- GT_RESET_IN is assigned location AK7.
- SFP_DISABLE which is used for hardware control isassigned K24 location.
- CHANNEL_UP and LANE_UP are assigned AE24 and AD24 locations respectively and they are depicted asLED's.
- RXP, RXN, TXP, TXN are assigned G1, H1, F2, G2locations respectively on FPGA board.

## 5. TESTING HARDWARE (CHIP SCOPE PRO ENVIRONMENT)

To observe the date transmission clearly in form of data plots and list representations ChipScope Pro environment is used. A "Test file" is created to give all the required specifications, such as number of bits, number of channels, the clock signal, trigger ports, trigger width, number of data samples (2048). After performing all the above steps, the simulated results of source code can be observed in list representation as shown in Figure 3.

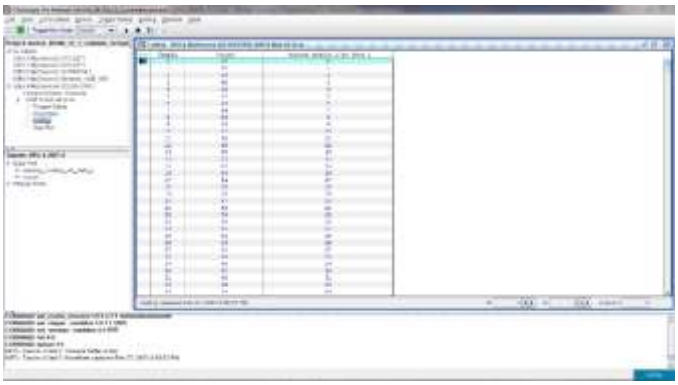

**Figure 3**: List Representation of Result

The "sample" column indicates the transmitted data in up-counter fashion The "count" column represents the received data. The delay can be observed and analyzed as, when the input data(sample) is 0, the output data(count) is 37. The delay is thus (37-0) 37 clock cycles.

## 6. CONCLUSION

In the Simulation of code in software there were 40 clock cycles of delay but after porting onto hardware the delay decreased to 37 clock cycles. This shows decreased latency by using FPGA hardware and optical fiber cable for higher speed in transmission of serial data. In this project, frequency of operation was chosen as 146.25 MHz. Hence, the delay introduced is 1/146.25 = 6.4ns for 1 clock cycle. Therefore, for 37 clock cycles the throughput delay observed is 6.4ns x 37 clock cycles = **236.8ns** only. Parallel I/O provides faster communication in comparison to Serial I/O, but involves design issues and uses more number of pins on the PCB which also makes the devices costlier. Whereas, Serial I/O uses lesser number of pins but takes time to transmit data bits which

increases the delay. Hence this project helps in generating a protocol which implements High-Speed Serial I/O which reduces the delay, reduces the design issues, is easier and convenient to use. It provides reliable and effective communication.

**ACKNOWLDEGEMENTS**

We take this opportunity to thank our project coordinator Mrs.M.Shanthi, Assistant Professor of Electronics and Communication Engineering, GNITS, for helping us in the project with her valuable suggestions.

We are grateful to our guide Sri M. Krishna Prasad, Sc-'E' and Sponsor Sri T.V.Bhaskar, Sc-'F', DLRL for their guidance and supervision during the course of our training.

We would like to acknowledge Dr. B. Venkateshulu, Professor and HOD, Electronics and Communication Engineering, GNITS, for being supportive and leading us towards the completion of the major project.

We also render our thanks to Dr.K.Ramesh Reddy, Principal, GNITS for providing us with the scope to accomplish our major project.

**REFERENCES**

[1] Abhijit Athavale and Carl Christensen. **High-SpeedSerial I/O Made Simple**, 1st ed, ch.3, pp. 20-52.

[2] Clive Maxfield. **Design Warrior's Guide to FPGAs**, Burlington, USA, 2004, ch.2, pp. 9-22.

[3] User Guide, **LogiCORETM IP Aurora 8B/10B ug_353** v5.3, pp. 35-104, January 2012.

[4] User Guide, **LogiCORETM IP Aurora 8B/10B ds_637,** v5.3, pp. 2-17, January 2012.

[5] Samir Palnitkar. **Verilog HDL**, California, USA, pp. 344-440, February 2003

[6] Padmanabhan and B. Bala Tripura Sundari. **Design through Verilog HDL,** pp. 159-268.

[7] Rajeev Madhavan. **Quick reference for Verilog HDL,** San Jose, CA, pp. 4-12.