International Journal of Thesis Projects and Dissertations (IJTPD)
Vol. 2, Issue 2, PP: (9-14), Month: April - June 2014, Available At: www.researchpublish.com

IMPLEMENTATION OF HIGH-SPEED SERIAL I/O OVER DUAL INDEPENDENT AURORA LANES/CHANNELS OF SAME GTX DUAL TILE USING AURORA PROTOCOL

Jeevan Reddy Mandali[1], Sudeshna Roy[2], G.Chandana[3]

[1,2]Master of Technology, Dept. of Electronics & Instrumentation Engineering, GITAM University, Visakhapatnam, India
[3]Master of Technology, Dept. of ECE, JNTU Hyderabad, Hyderabad, India

Abstract: **For over years, parallel I/O schemes ruled the chip to chip, board to board or backplane communication. Parallel I/O had to experience performance issues like crosstalk, signal integrity and increased skew after passing certain I/O clock-frequency rates [1]. Also parallel I/O methods increased the complexity of the hardware (high pin count, more wires) and made bandwidth sharing inevitable. In many new communication protocols serial data transmission has become very common due to low pin count (reduced cost). Serial transmission methods can also transmit at much higher clock rates per bit transmitted, thus outweighing the parallel transmission method. This paper contains the implementation of high-speed serial transmission on multi-gigabit serial I/O regulated by the Aurora protocol without channel bonding, supporting results and future scope of multi-gigabit designs.**

Keyword: Serial I/O, Aurora protocol, multi-gigabit serial I/O

I. INTRODUCTION

High-speed serial I/O standards require only a half or less pin and wire count for the transmission unlike the parallel transmission schemes. Surprisingly serial transmission methods, which are being widely used today in many new communication protocols, are able to transfer data at rates higher than 1 Gb/s. In high-speed serial transmission, clock and data are combined in a single stream thus reducing the problem of bit-to-bit skew. High-speed serial I/O standards span in more than one communication scenario like, Fibre channel, InfiBand and Gigabit & 10-Gigabit Ethernet etc. However, a rate of 3.125 Gb/s, achieved using multi-gigabit serial I/O and Aurora protocol, carried out on a Virtex-5 FPGA is my work of concern. Two lanes of a same dual tile act independently as I did not use the concept of channel bonding.

II. GTX_DUAL TILE INSIDE THE VIRTEX-5 XC5VFX100T FPGA

- GTX transceiver is used as an alias to multi-gigabit serial I/O in this paper.
- There are sixteen GTX transceivers on the Virtex-5 XCVFX100T placed in GTX_DUAL tiles, each containing two transceivers [2].

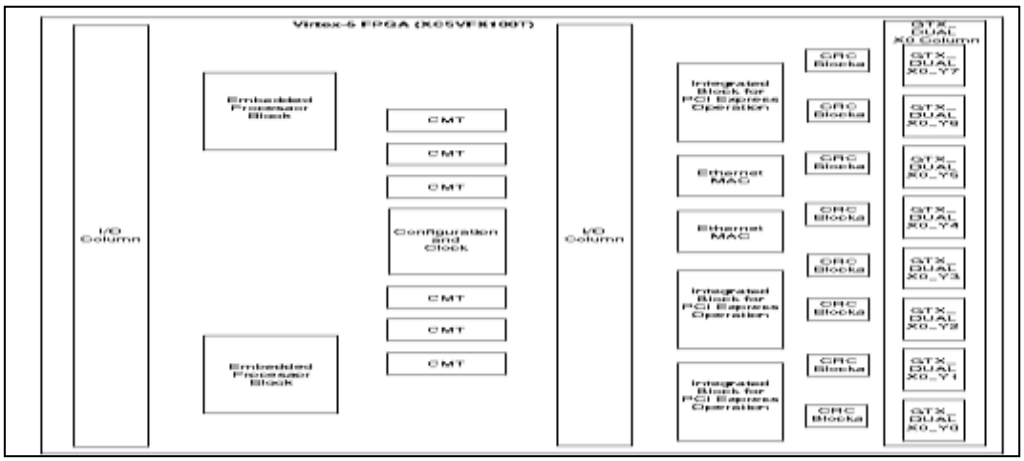

Fig. 1 GTX_DUAL Tile Inside the Virtex-5 Xc5vfx100t FPGA

III. BLOCK DIAGRAM OF A MULTI-GIGABIT TRANSCEIVER

Each GTX transceiver in the GTX_DUAL tile includes an independent transmitter, which consists of a PCS and a PMA.

A. Transmitter Overview[2]

The FPGA TX interface is the FPGA's gateway to the TX data path of the GTX transceiver. Applications transmit data through the GTX transceiver by writing data to the TXDATA port. The width of the port can be configured to two bytes wide. The TX port is a differential port, which means two outputs with opposite polarity will be available on TX^+ and TX^-. 8B/10B is used for improved performance. The GTX TX data path has two internal parallel clock domains used in the PCS: the PMA parallel clock domain (XCLK) and the TXUSRCLK domain. The phase differences between XCLK and TXUSRCLK are reduced by the TX buffer & TX Phase alignment system in TX-PCS.

As differential signaling method is used, TX polarity provides an option to invert the outgoing data from PCS before serialization to avoid hardware fixes for swapped TXP/TXN. TX Gearbox supports 64B/66B and 64B/67B encoding if needed. Pseudo-random sequences are generated to test the signal integrity, which is used in conjunction with loopback test.

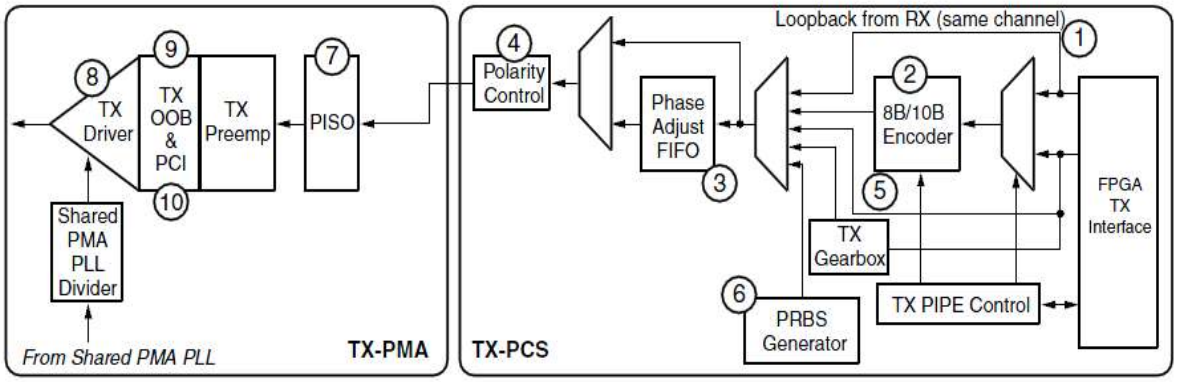

Fig. 2 GTX TX Block Diagram

PISO serializes the parallel data from PCS using a high-speed clock from shared PLL. A high-speed current-mode differential output buffer named TX Driver is there for maximizing signal integrity.

B. Receiver Overview[2]

The first stage of the RX data path in each GTX transceiver is the RX current mode logic (CML) receiver. The CML receiver includes circuits to allow the termination of the channel to be optimized for the best possible signal integrity.

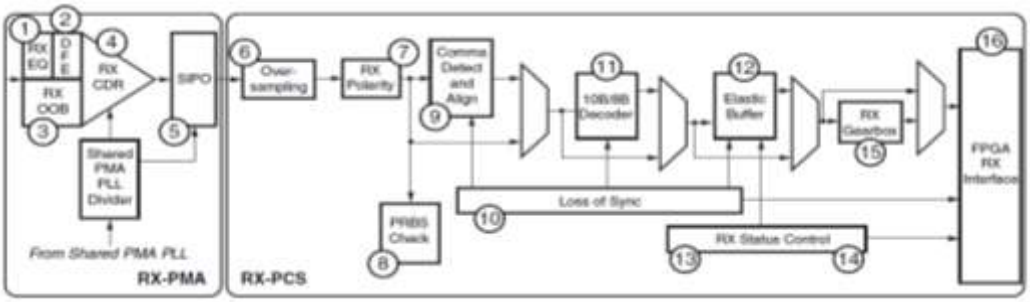

Fig. 3 GTX RX Block Diagram

The receiver also features an RX equalization circuit that allows it to compensate for high frequency losses in the channel, improving the quality of the received signal. The RX Clock Data Recovery (CDR) circuit in each GTX transceiver extracts a recovered clock from incoming data.

The Serial In to Parallel Out (SIPO) block Deserializes serial data from the GTX receiver PMA and presents it as parallel data to the PCS. It uses both edges of a high-speed clock to deserialize incoming data and present it to the PCS.

The GTX transceiver includes an 8B/10B decoder to decode RX data without consuming FPGA resources. The decoder includes status signals to indicate errors and incoming control sequences. If decoding is not needed, the block can be disabled to minimize latency.

IV. AURORA CORE

The Aurora core is a high-speed serial solution based on the Aurora protocol. The Aurora 8B/10B is protocol independent and can be used to transport industry standard protocols, such as Ethernet and TCP/IP, or proprietary protocols.

A. Functional Overview of Aurora core[2]:

Aurora 8B/10B cores automatically initialize a channel when they are connected to an Aurora channel partner. After initialization, applications can pass data freely across the channel as frames or streams of data [3].

Aurora frames can be any size, and can be interrupted at any time. Gaps between valid data bytes are automatically filled with idle sequences to maintain lock and prevent excessive electromagnetic interference.

User application can be anything like a counter or some RADAR, giving the data in time. This will be fed to the Aurora core for the further modification of data according to the protocol and then sent serially at a high rate to other applications. User can also check the validity of data by loop backing the data to the same transmitter device by forcing the LOOPBACK [2:0] port to a value from the Table 1.

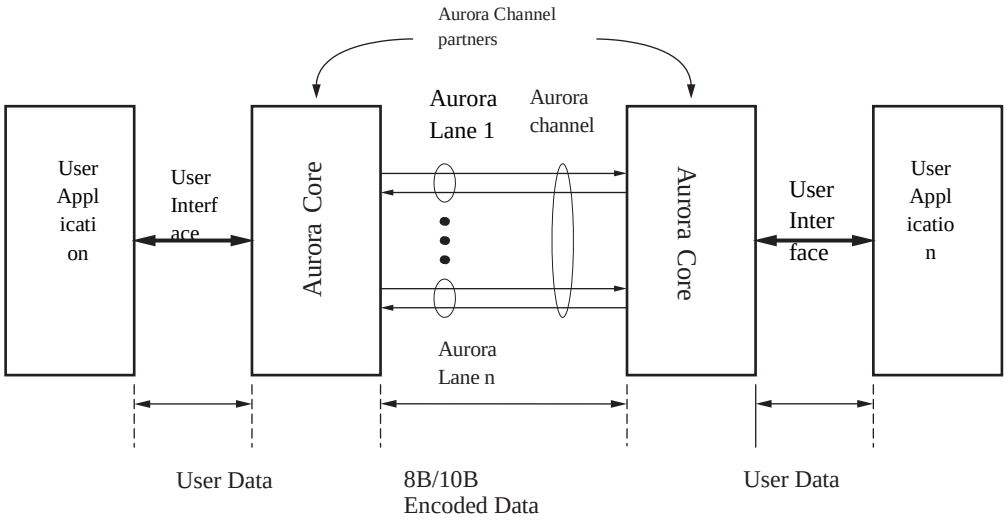

Fig. 4 Functional Overview of Aurora core

Table 1. Loopback Ports

Port(LOOPBACK[2:0])	Description
000	Normal Operation
001	Near-End PCS
010	Near-End PMA
011	Reserved
100	Far-End PMA
101	Reserved
110	Far-End PCS
111	Reserved

a. Near-End PCS Loopback[2]:

The test data is generated and checked by user logic and then looped back in the PCS. The difference compared to the Near-End PMA Loopback mode is that the PMA section is not involved.

b. Near-End PMA Loopback[2]:

This mode uses the Near-End source for generating and checking the test data. The loopback occurs in the serial section of the PMA before the line drivers. The test data can be generated and checked by the built-in PRBS block inside the PCS or by user logic.

c. Far-End PMA Loopback[2]:

This mode uses the Near-End data source for generating and checking test data. The loopback occurs after passing the serial-to-parallel converter of the PMA. This mode tests the complete PMA section including the serial-to-parallel and parallel-to-serial conversion. Almost the entire PCS section is bypassed because the RX serial inputs are looped back to the TX serial outputs.

d. Far-End PCS Loopback[2]:

This mode uses an external source to generate and check the test data. The loopback occurs in the PCS section and the received data is presented to the user logic. The transmit data of the user logic is not transmitted when this mode is activated.

B. Customizing the Aurora 8B/10B Core using IP customizer:

Fig. 5 Parameter customization

V. APPLICATION DESIGN

The following figure explains the design and its procedure to implement the high-speed serial data transmission technique. The 192 bit counter data is given to an Asynchronous FIFO, which then outputs 16 bit data to the Aurora core. Then the GTX transmitter section transmits the data serially at a rate of 3.125 G/ps. The transmitted data is then loop backed and collected by the GTX receiver block on the RX port, in order to test the mechanism.

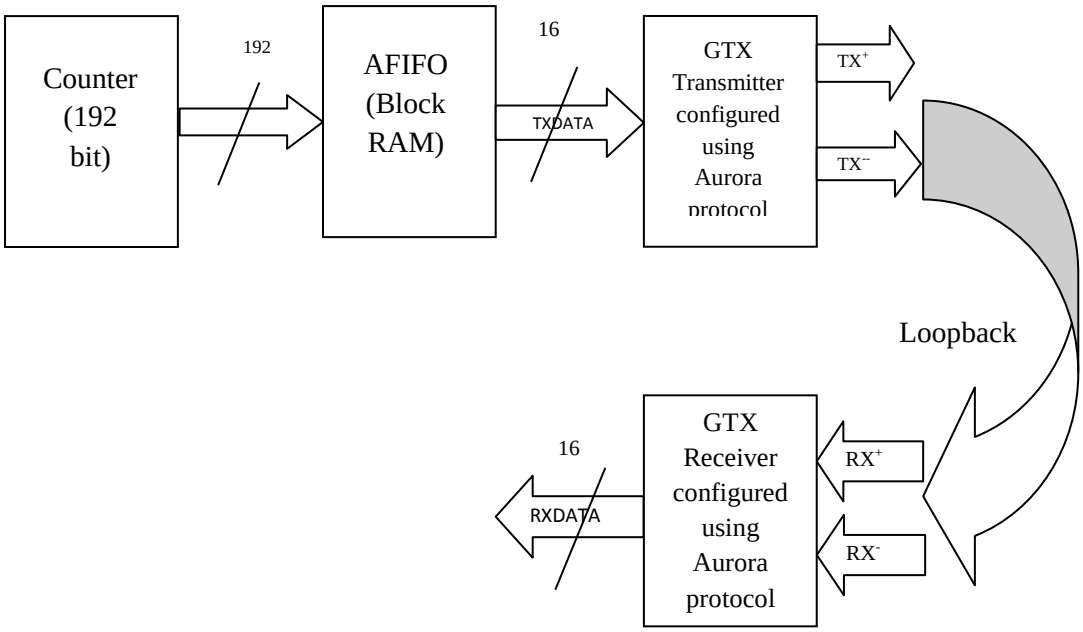

Fig. 6 Application design

VI. SOFTWARE & HARDWARE REQUIREMENTS

A. Software resources:

1. Xilinx ISE Design Suite 12.4
2. Xilinx CORE Generator
3. Xilinx ChipScope Pro Analyzer

The ChipScope Pro tools integrate key logic analyzer and other test and measurement hardware components with the target design inside the supported Xilinx FPGA devices [5]. ChipScope Pro is software-based logic analyzer. It allows monitoring the status of the selected signals in a design in order to detect possible design errors. It provides several cores that can be added to a design by generating the cores with the CORE Generator tool, instantiating them into the source code, and connecting the cores to the design before the synthesis process. Alternatively, it is possible to customize the cores and insert them into the design netlist using the ChipScope Pro Core Inserter tool after the synthesis process. The design is then implemented into the FPGA device using the implementation tools of the Xilinx.

B. Hardware resources:

1. Virtex-5 RocketIO development board (custom)

The Xilinx FPGA device used is Virtex-5 FX series (XCV5FX100T-1FF1136) with speed grade -1 and package ff1136. XCV5FX100T-1FF1136 platform is used as it provides high-performance embedded system with advanced serial connectivity.

2. JTAG

JTAG is a tool used for In-system programming. JTAG Programmer software uses sequences of JTAG instructions to perform the following programming and verification operations.

a) **Program**: Downloads the contents of the JEDEC or BIT file to the device programming registers.
b) **Verify**: Reads back the contents of the device programming registers and compares them with the JEDEC or BIT file.
c) **Erase**: Clears device configuration information.
d) **Bypass**: Ignores this device when addressing devices in the JTAG boundary scan chain. This option is only available through chain operations.

C. BIT Files:

Bit files are Xilinx FPGA configuration files generated by the Xilinx FPGA design software [2]. They are proprietary format binary files containing configuration information.

VII. EXPERIMENTAL RESULTS

The result observed using ChipScope Pro analyser.

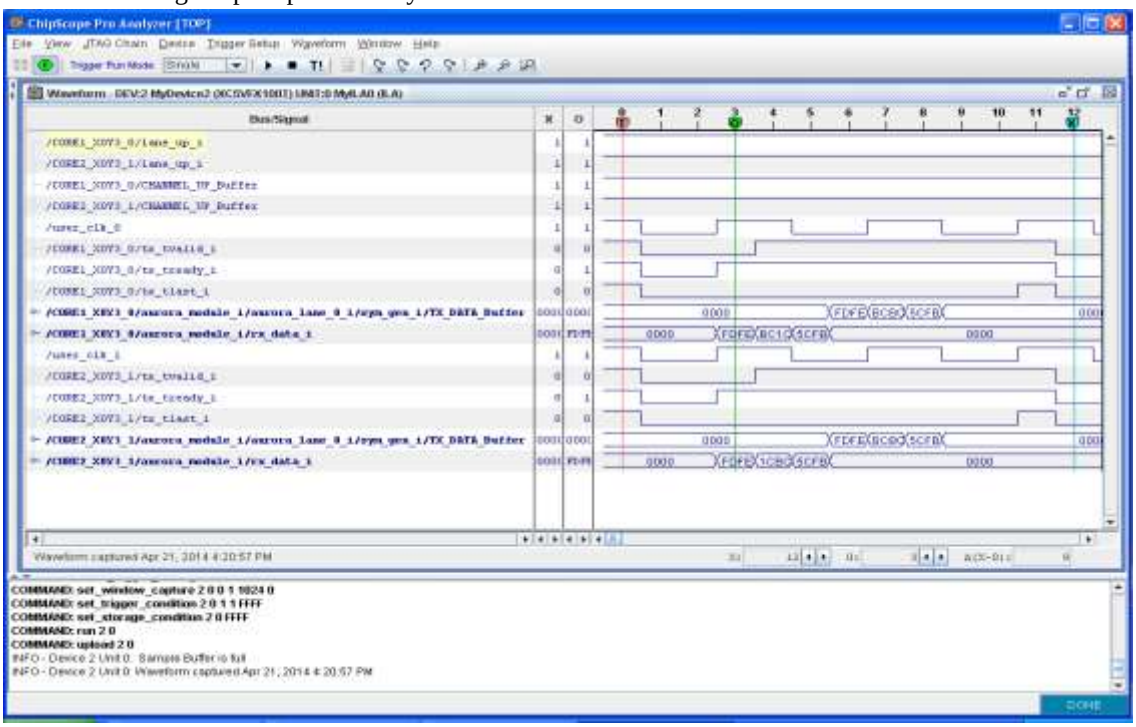

Fig. 7 ChipScope Pro result

VIII. CONCLUSION AND FUTURE SCOPE

A. Conclusion:

The high-speed serial data transmission at a rate of 3.125 G/ps using multi-gigabit serial I/O is successful. The data transmitted is compared to the data received using the loopback technique. The received data is same as the transmitted data and hence the data correctness is verified.

B. Future scope:

Serial designers must contend with signal integrity, smaller time bases, and possibly the need for extra gates and additional CPU cycles. However, multi-gigabit advantages in box-to-box and chip-to-chip communication far outweigh the perceived shortcomings. For example, high speed, fewer pins, lower EMI, and lower cost make it the ideal choice in many communication designs. These advantages will ensure its continued use in communication applications far into the future and speed can be increased to higher level.

REFERENCES

[1]. Applications drive trend to serial I/O, Khanh Le, http://www.eetimes.com/document.asp?doc_id=1206164, 10/31/2002.
[2]. Virtex-5 FPGA RocketIO GTX Transceiver User Guide, UG198 (v2.1) November 17, 2008.
[3]. LogiCORE IP Aurora 8B/10B v6.1, DS797 [v1.0] September 21, 2010.
[4]. JTAG Programmer Guide, http://support.xilinx.com
[5]. ChipScope Pro Software and Cores User Guide, UG029 (v13.4) January 18, 2012.