International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-2 Issue-1, March 2013

A Novel Architecture for Super Speed Data Communication for USB 3.0 Device Using FPGA

Amit Kumar Amod, Ansuman DiptiSankar Das, Tapas Sahu, Sekhar Sahani, Deepak Kumar Panda

Abstract - The need for SuperSpeed data communication leads to the use of USB 3.0. USB 3.0 utilizes dual bus architecture which provides both SuperSpeed and non-SuperSpeed connectivity. This can be possible by mixing the advantage of parallel and serial data transfer. This paper provides a novel architecture for communication between USB 3.0 device and USB 3.0 host controller at a data rate of maximum up to 5.0 Gbps using Altera's Stratix IV FPGA. To maintain synchronization between GPIF II and PCIe hard IP, FIFO is used. PLL is used to provide clock signal at different frequencies.

Keywords – FIFO, FPGA, GPIF, Hard IP, PLL, USB 3.0.

I. INTRODUCTION

The motivation for the Universal serial Bus (USB) came from several considerations like Ease-of-use, Port expansion etc. Initially, USB provided two speeds, 12 Mb/s and 1.5 Mb/s, for peripherals. However, as PCs processed more data, users needed a powerful protocol. To meet that need, the USB 2.0 specification was published in 2000 to provide a third transfer rate of 480 Mb/s while retaining backward compatibility. Now, new kinds of devices, media formats, and large, inexpensive storage devices are converging. These applications have prompted USB 3.0 [1], in 2008, which transfers data at 5 Gb/s and is compatible with past protocols.

USB is the most powerful PC peripheral interconnect and has a great use in mobile and CE segments. The USB On-The-Go technique provides a way for two devices, capable of dual role, to be connected and decides which one is the "host". As new technologies emerges, new kind of devices, large inexpensive storage and new media format are converging.

This leads to the requirement of more bus bandwidth to maintain proper interaction. The USB is still the answer to connectivity for consumer electronics, PC and mobile architectures. It is a bidirectional, fast, dynamically attachable and low cost interface which fulfils the requirement of interconnection. USB 3.0 is a cable bus supporting data exchange between a host computer and a wide range of simultaneously accessible peripherals. USB 3.0 utilizes dual bus architecture which provides its compatibility with USB 2.0. It provides both SuperSpeed and non-SuperSpeed connectivity.

Revised Manuscript Received on 30 March 2013.
* Correspondence Author
Amit Kumar Amod[1], Dept. Of ECE, NIT Rourkela, India
Ansuman DiptiSankar Das[2], Dept. Of ECE, NIT Rourkela, India
Tapas Sahu[3], School of Electronics, KIIT University, India
Sekhar Sahani[4], School of Electronics, KIIT University, India
Deepak Kumar Panda[5], School of Electronics, KIIT University, India

The example of device controller is CYUSB3014, which is a Cypress's device. It provides highly integrated and flexible features which supports USB 3.0 functionality so that it can be added to any system. It has a fully parallel, configurable, General Programmable Interface known as GPIF II, which can be connected to any processor, ASIC, or FPGA [2]. The example of host controller is TUSB7320, which is a Texas Instrument's device. It is a USB 3.0 compatible device, which supports PCIe x1 Gen2 Interface [3] - [5] with two downstream ports. Each downstream port may be independently enabled or disabled. With 40-nm Stratix IV FPGAs, our design can reach new levels of system-on-a-chip (SoC) integration [6]. Stratix IV FPGAs deliver a high-density, feature-rich and high-performance core fabric. Combined with flexible I/Os, high-bandwidth transceivers, and memory interfaces, Stratix IV FPGAs meet the requirements for high-end digital systems in wireless, wireline, military, broadcast, and other market segments.

The paper is structured as follows. In section II the migration of USB 3.0 from USB 2.0 has been described. The proposed architecture is illustrated in Section III. In Section IV, the results and discussion are debated. At last, the topic concludes with section V.

II. MIGRATION OF USB 3.0 FROM USB 2.0

USB 2.0, also referred to as high-speed USB, is an external bus that supports data rates up to 480 Mbps. USB 2.0 is the extension of USB 1.1 with additional features. USB 2.0 uses the same connectors and cables and is fully compatible with USB 1.1. USB 3.0 is a physical SuperSpeed bus combined in parallel with a physical USB 2.0 bus. It has similar architectural component as USB 2.0, namely USB 3.0 interconnect, USB 3.0 devices and USB 3.0 host. The USB 3.0 interconnect defines the manner in which USB 2.0 and USB 3.0 devices connect to and communicate with the USB 3.0 host. USB 3.0 consists of a star topology with a single host. The USB 3.0 connection model accommodates backwards and forward compatibility for connecting USB 2.0 or USB 3.0 devices into a USB 3.0 bus. In a similar fashion USB 3.0 devices can be attached to a USB 2.0 bus.

The physical interface of USB 3.0 is compromise of USB 2.0 electrical, mechanical and SuperSpeed physical specifications for the buses. USB 3.0 cables have eight primary conductors, three twisted signal pairs for USB data paths, and a power pair. In addition to the twisted signal pair for USB 2.0 data path, two extra twisted signal pairs are used to provide the SuperSpeed data path, one for the receive path and the other for transmit path.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

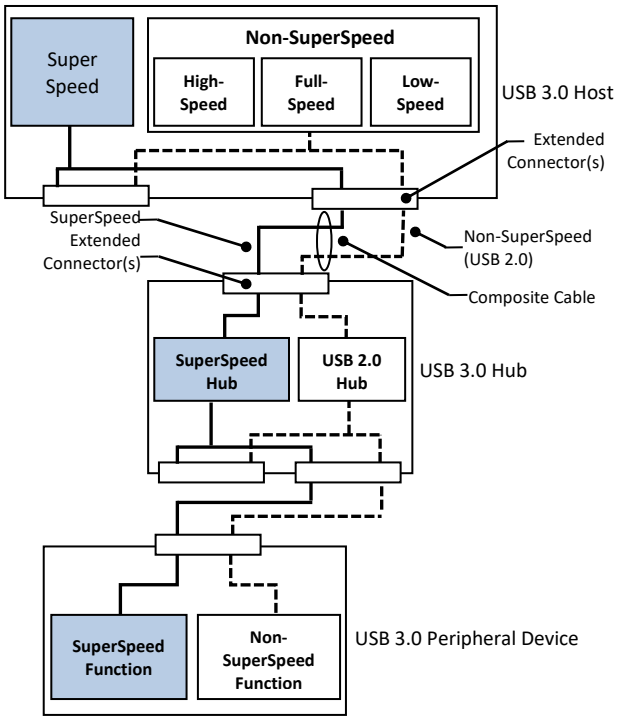

Fig. 1. Dual bus architecture of USB 3.0

III. PROPOSED ARCHITECTURE

The main objective of this project is to use the SuperSpeed data rate in applications like imaging and video devices, scanners, and printers etc. For that, the implementation is done on FPGA board. In this paper, bridging in FPGA between GPIF and PCI Express (Peripheral Component Interconnect Express) has to be done. The PCI Express Hard IP is used for the implementation. And hence, according to the requirement we need to select the proper devices. The whole system is shown in figure 2 and the project's work has been outlined.

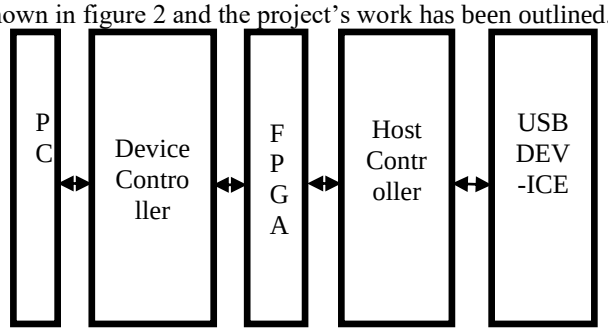

Fig. 2. Overall system showing interconnection between PC and USB devices

The basic Blocks used in the proposed architecture and their significances are described below.

A. GPIF II

The GPIF II provides glue-less connectivity to popular interfaces, such as an asynchronous SRAM or an asynchronous and synchronous address data multiplexed interface. One popular implementation of GPIF II is the synchronous Slave FIFO interface. This interface is used for applications in which the external device connected to EZ-USB FX3 accesses the EZ-USB FX3 FIFOs, reading from or writing data to them. Direct register accesses are not performed over the Slave FIFO interface. The GPIF II is a programmable state machine that enables the flexibility of implementing an industry standard or proprietary interface. Both parallel and serial interfaces may be implemented with GPIF II. The GPIFII can function either as master or slave. The GPIF II can Functions as master or slave, which offers 256firmware programmable states and supports 8-bit, 16-bit, and 32-bit parallel data bus. GPIF II enables interface frequencies up to 100 MHz and supports 14 configurable control pins when 32-bit data bus is used; all control pins can be either input/output or bidirectional and supports 16 configurable control pins when 16- or 8-bit data bus is used; all control pins can be either input/output or bidirectional. GPIF II state transitions occur based on control input signals. Control output signals are driven by GPIF II state transitions.

The behaviour of the state machine is defined by a descriptor, which is designed to meet the required interface specifications. The GPIF II descriptor is essentially a set of programmable register configurations. In the EZ-USB FX3 register space 8 Kb is dedicated as GPIF II waveform memory, where the GPIF II descriptor is stored.

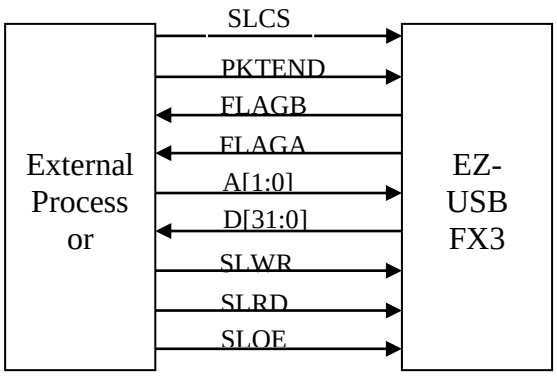

Fig. 3. Synchronous slave FIFO interface

B. PCI Express Hard IP

PCI (Peripheral Component Interconnect) Express is a high-performance interconnect protocol for use in a variety of applications including network adapters, storage area networks, embedded controllers, graphic accelerator boards, and audio-video products. The PCI Express protocol is software backwards-compatible with the earlier PCI and PCI-X protocols, but is significantly different from its predecessors. It is a packet-based, serial, point-to-point interconnect between two devices. The performance is scalable based on the number of lanes and the generation that is implemented. The PCI Express hard intellectual property (IP) [7] – [8] block embeds the PCI Express protocol stack into the Altera FPGA. The hard IP block includes the transceiver modules, physical layer, data link layer, and transaction layer. Altera offers both endpoints and root ports that are compliant with PCI Express Base Specification 1.0a or 1.1 for Gen1 and PCI Express Base Specification 2.0 for Gen1 or Gen2. Both endpoints and root ports can be implemented as a configurable hard IP block rather than programmable logic, saving significant FPGA resources. The IP Compiler for PCI Express is available in ×1, ×2, ×4, and ×8 configurations. The protocol specifies 2.5 GT/s for Gen1 and 5 GT/s for Gen2.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

C. Proposed Design

The data communication between the host controller and PCIe hard IP is a serial communication, whereas the output from PCIe hard IP is a 64-bit parallel data. The device controller supports maximum data width of 32-bit. Hence to communicate between device controller and PCIe hard IP, we need to convert 64-bit data to 32-bit and vice versa for duplex communication. The host controller and device controller can operate at difference frequencies. For example, CYUSB3014 (device controller) and TUSB7320 (host controller), works at different frequency. Hence to maintain synchronisation between both the devices, we need to use two FIFO, one for transmission of data and another for reception of data. The FIFO used is a double clock mixed width FIFO. It converts 64-bit data to 32-bit data and vice versa at transmitter and receiver side respectively. PLL is used to provide clock signal at multiple frequencies. An external clock signal is given to PLL to produce output clock signal of different frequencies. Altera's STRATX IV FPGA is used to implement the architecture. It consists of PCIe hard IP block and transceiver blocks. Altera's ModelSim is used to simulate the code.

A differential pair of input is given to the hard IP from the host controller in terms of PCIE_TXP and PCIE_TXN. The hard IP will check first whether the FIFO controller is ready or not via rx_st_ready pin. If the FIFO controller is ready, then it will start transmitting data to FIFO_RX until the end of packet goes high. In this process, the start of packet will go high for one clock cycle and the valid pins goes high. The FIFO_RX converts the 64-bit parallel data to 32-bit parallel data. The output data of FIFO_RX is given to tri state buffer. The wrusedw_rx tells whether the FIFO_RX is full or not.

A bidirectional data bus is used in the device controller so that data can be read or written depending on the usage. The device controller gives an interrupt signal to the GPIF controller. The GPIF controller checks whether there is data present in FIFO_RX or not. Depending on the content of FIFO_RX, the rdempty_rx signal changes its state. If there is data in FIFO_RX, then GPIF controller will select the device controller for write operation through a chip select signal. The FLAGA and FLAGB show the state of FIFO present in the device controller. FLAGA represents the empty state and FLAGB represents the full state of the FIFO present in the device controller. If the FIFO in the device controller is empty, then GPIF controller starts sending the data from FIFO_RX via tri state buffer to the device controller till packet end occurs. If the FIFO in the device controller is full, then the GPIF controller reads data from the device controller and gives it to FIFO_TX.

If the FIFO_TX is full, then the write operation stops. This is determined by the status of wrused_tx signal. If data is available in FIFO_TX, then FIFO controller sends read request signal to FIFO_TX and data transmission starts only when hard IP is ready to accept data till end of packet. A differential pair of output in terms of PCIE_RXP and PCIE_RXN is given to the device controller. The blocks present in the proposed architecture works at different clock frequencies. As there is no in build clock source present in the FPGA, so to provide clock signals at different frequencies, Phase Locked Loop (PLL) is used. We have used the hard PLL here, which takes a clock signal at a particular frequency as input and provides clock signals at different frequencies as its output. The input frequency to the PLL is 100 MHz and the output frequencies are 100 MHz, 125 MHz, and 200 MHz.

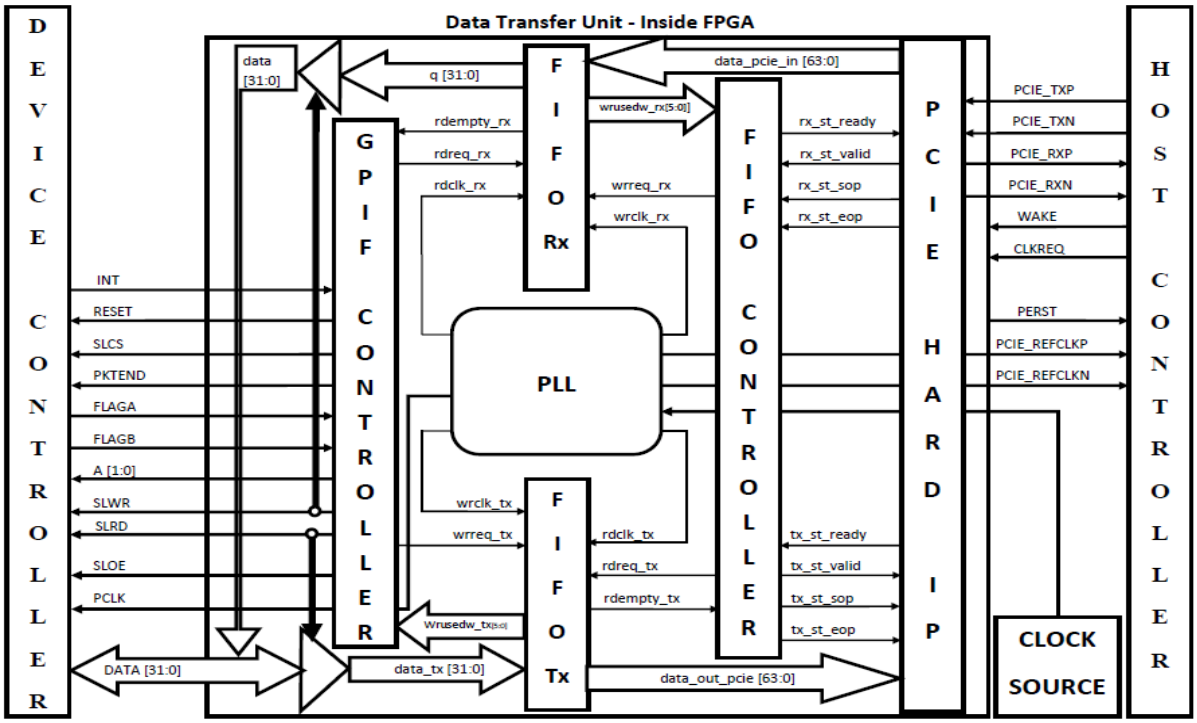

Fig. 4. Proposed architecture

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

IV. RESULTS AND DISCUSSION

The functionality of the different blocks used in the proposed architecture has been tested by writing code in ModelSim-Altera 6.6d. The FPGA used here is Stratix-IV. Different clock signals are used here for FIFO read and write so that it can support different port widths. The simulation result for FIFO_RX and FIFO_TX are shown in figure 5 and 6 respectively. From figure 5, it is clear that FIFO_RX takes 64-bit parallel input and gives 32-bit parallel outputs in two clock cycles. Similarly the FIFO_TX takes two 32-bits inputs and gives an output of 64 bits.

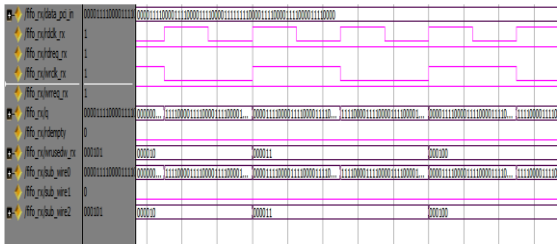

Fig. 5. Simulation result of FIFO_RX

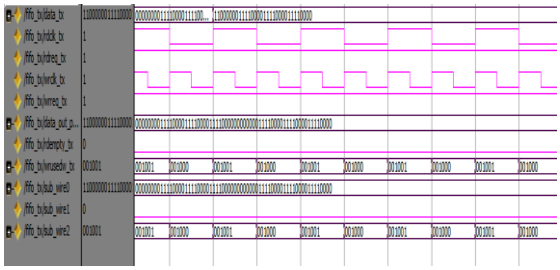

Fig. 6. Simulation result of FIFO_TX

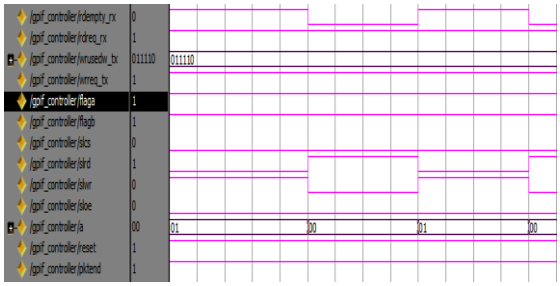

Fig. 7. Simulation result of GPIF controller

The simulation result for GPIF controller is shown in figure 7. The results are verified with the proposed functionality of GPIF controller as discussed in the proposed design. The output of PLL is shown in figure 8. A delay of 10.7 ns is experienced while obtaining the output. For the above mentioned delay period, the output remains undefined.

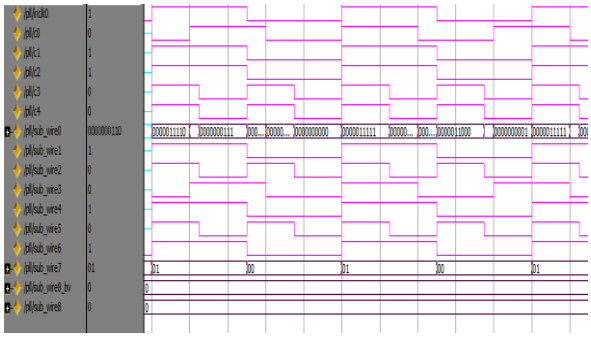

Fig. 8. Simulation result of PLL

For the verification of data transfer between host and device, a 64-bit data has been given to FIFO_RX. As output, we got a 32-bit data. The data transfer is being controlled by GPIF controller and is also passed through a buffer. The 32-bit data has been successfully transferred from the device to FIFO_TX. The above operation is controlled by GPIF controller. The simulated outputs are shown in figure 9 and 10 respectively.

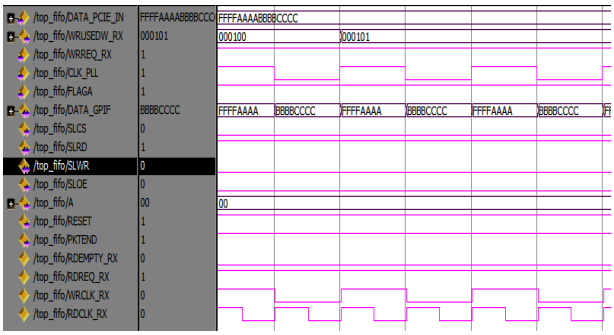

Fig. 9. Simulation result showing data transfer between host controller and device controller

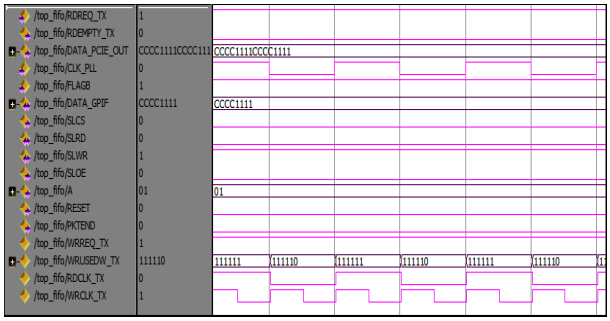

Fig. 10. Simulation result showing data transfer between device controller and host controller

V. CONCLUSION

The individual blocks in the proposed architecture has been tested and verified successfully on ModelSim - Altera. The simulation results have been shown. We have used here the hard IP of Stratix-IV FPGA. Phase locked loop, present in Stratix-IV FPGA has been used to provide clock signal at different frequencies. Individual blocks have been given clock signal at different frequencies for maintaining synchronization. FIFO has been used to convert 64-bit output of PCIe hard-IP into 32-bit to make it compatible with GPIF controller. Finally, data has been successfully transferred from the host controller to the device controller.

REFERENCES

1. Dr. Andreas C. Wolf , Dr. Wolf Wireless GmbH, Dr. Christoph Scheytt , “15 Gbps Communication over an USB3.0 Cable and Even More,” 2012 9th International Multi-Conference on Systems, Signals and Devices (SSD), Mar. 2012, pp. 1 – 3.
2. Roberto Ammendolaab, Andrea Biagionic, Giacomo Chiodic, etal, “High Speed Data Transfer with FPGAs and QSFP+ Modules,” 2010 IEEE Nuclear Science Symposium Conference Record (NSS/MIC), Oct. 30 2010-Nov. 6 2010, pp. 1323 – 1325.
3. Hu Li, Yuan’an Liu, Dongming Yuan, Hefei Hu, “A Wrapper of PCI Express with FIFO Interfaces based on FPGA,” 2012 International Conference on Industrial Control and Electronics Engineering, Aug. 2012, pp. 525 – 529.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

4. Hu Li, Yuan'an Liu, Dongming Yuan, Hefei Hu, "A Wrapper of PCI Express with FIFO Interfaces based on FPGA," 2012 International Conference on Industrial Control and Electronics Engineering, Aug. 2012, pp. 525 – 529.
5. Hossein Kavianipour, Steffen Muschter and Christian Bohm, " High Performance FPGA-based DMA Interface for PCIe," 2012 18th IEEE-NPSS Real Time Conference (RT), June 2012, pp. 1 – 3.
6. Ranianand Venkata, Wilson Won, etal, "Architecture and Methodology of a SoPC with 3.25Gbps CDR based Serdes and Gbps Dynamic Phase Alignment," IEEE 2003 CUSTOM INTEGRATED CIRCUITS CONFERENCE, Sept. 2003, pp. 659 662.
7. Shao-Hang Hung, Chih-Feng Chao, Yu-Chun Yan, etal., "Independent Component Analysis Hard-IP integration System on Programmable Chip (SOPC) Platform," TENCON 2010 - 2010 IEEE Region 10 Conference, Nov. 2010, pp. 1705 – 1709.
8. Edin Kadric, Naraig Manjikian, Zeljko Zilic, "AN FPGA IMPLEMENTATION FOR A HIGH- SPEED OPTICAL LINK WITH A PCIE INTERFACE," 2012 IEEE International SOC Conference (SOCC), 12-14 Sept. 2012, pp. 83 – 87.

AUTHOR PROFILE

Amit Kumar Amod was born in Uren, India, in 1986. He received his B. Tech degree in electronics and communication engineering form Anna University, Chennai, India, in 2009. He is currently pursuing his M. Tech in VLSI Design and Embedded Systems at National Institute of Technology Rourkela, India. His current areas of interest are VLSI architectures for FPGA.

Ansuman DiptiSankar Das was born in Balasore, India, in 1986. He received his B. Tech degree in electronics and telecommunication engineering form BPUT, Odisha, in 2007. He's currently pursuing his M. Tech in VLSI Design and Embedded Systems at National Institute of Technology Rourkela, India. His current areas of interest are VLSI architectures for digital signal processing and design of real-time embedded systems.

Tapas sahu was born in Balasore, India in 1985. He received his B.Tech degree in Electronics and Telecommunication engineering from BPUT, Odisha, in 2007. He is currently pursuing his M.Tech in Communication Systems Engineering at KIIT University, Bhubaneswar, Odisha, India. His current areas of interests are wireless Communication.

Sekhar Sahani was born in Jajpur, India, in 1989. He received his B.Tech in Electronics and Telecommunication Engineering from BPUT, Odisha, in 2010. He's currently pursuing his M.Tech in Communication Systems at KIIT UNIVERSITY, Bhubaneswar, India. His current areas of interest are Green Wireless Communications and Digital signal processing.

Deepak Kumar Panda was born in Berhampur, India, in 1989. He received his B.Tech degree in Electronics and Communication Engineering from BPUT, Orissa, in 2010. He is currently pursuing M.Tech in VLSI Design and Embedded System at KIIT University, Odisha, India. His current areas of interests are Low Power and high speed VLSI circuit designs, embedded systems.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)