A Review on Architecture of High Speed data Communication for USB 2.0 Device using FPGA

Zeesha Mishra
ME Research Scholar SSTC,Bhilai

Anil Kumar Sahu
Assistant Professor, SSTC,Bhilai

ABSTRACT
In this paper a review on USB2.0 using this innovative approach presented the authors own concept of USB receiver/transmitter architecture. The various concept was implemented in hardware description language to provide model for simulations. Simultaneously the code is synthesizable, and may be physically implemented in programmable logic devices. Selected details of construction and functionality of USB protocol were reported. Described The Universal Serial Bus (USB) Transceiver Macro cell Interface (UTMI) is a two wire, bi- directional serial bus interface between USB devices through D+ and D- lines. There are three functional blocks present in USB controller; those are Serial Interface Engine (SIE), UTMI and Device Specific Logic (DSL).When the data is received on the serial bus, it is decoded, bit unstuffed and is sent to receive shift register. Designed a High speed easy to use peripheral interfaces like USB 2.0 .Implemented hardware on a Spartan-3 FPGA is easily capable of handling data throughputs as high as what USB 2.0 demands. Using dedicated resources such as multipliers it is also possible to do signal processing tasks on these data.

Keywords
USB.2.0, UTMI, Transmitter, Receiver.

1. INTRODUCTION
The trend in electronics is to make devices as small as possible and to get them to market quickly. This trend has caused designers to focus on FPGAs rather than the traditional PCBs and ASIC's. This trend to make devices smaller and cheaper combined with the rising costs of labor to solder components onto PCBs is resulting in developments of "Systems on a Chip" on FPGAs. The USB standard was developed to overcome the shortcomings of older interfaces to peripheral devices for PCs. The standard makes interfacing to the PC extremely easy for the end user and life more complicated for the peripheral designer. The USB 2.0 Transceiver Macro cell [8] understands the USB protocol and is capable of carrying out transactions on behalf of the device. Verilog is a hardware description language widely used in the VLSI industry. A field-programmable gate array (FPGA) is a programmable logic device that supports implementation of relatively large logic circuits [7].

1.1 PROBLEM AREA
The work complies with the USB 2.0 Transceiver Macro cell and has been developed in Verilog .The USB 2.0 Transceiver Macro cell made in this project is capable of carrying out high speed USB transactions which is 12 Mbits/sec.

1.2 SOLUTION
In order to implement the USB 2.0 Transceiver Macro cell [8], the system can be implemented by three major components. These components are the receiver, the transmitter and the controller.

The receiver and transmitter form the data path. The controller coordinates the data path. The receiver's main task is to detect and receive token and data packets from the host. It is also the receiver's responsibility to inform the device specific logic that it should accept a byte of data. The receiver informs the device if it is the target of the transaction and not some other device on the USB topology. The transmitter is responsible for emitting data and handshake packets to the host. The transmitter is responsible for sending data from the device to the PC.

2. LITERATURE REVIEW
2.1 Papers Reviewed
A number of research papers of various journals and conferences were studied and survey of existing literatures in the proposed area is reported below:

Pandey et al. (2013), designed USB2.0 using this innovative approach (validation using U-Boot framework) root-caused several notorious issues which were hard to narrow down using legacy approach. This possessed both the legacy capability of low level programming (JTAG) as well as of application level (High Level) programming (Linux). Limitation of JTAG based approach: JTAG based approach is suitable for low level programming. If entire validation is done using this approach the amount of code to be written for each feature will grow exponentially. Though state of system can be checked in intermediate stages, this is very slow. Limitation of Linux based approach: The Linux code is bulky, run too fast and enables several modules in background. The number of retries are more. Because of this several issues remains unnoticed or are overwritten. The state of system cannot be checked in intermediate stage which is needed for debugging[1].

Przemyslaw et al. (2012) presented the authors own concept of USB receiver/transmitter architecture. The concept was implemented in hardware description language to provide model for simulations. Simultaneously the code is synthesizable, and may be physically implemented in programmable logic devices.

Selected details of construction and functionality of USB protocol were presented. The design was partitioned to three modules forming a system of bi-directional, sequential data flow. Each module, responsible for specific stage of data processing is controlled by its own Finite State Machine. The other imaginable solutions are splitting the architecture in accordance with direction of data transmission, to the two parts – receive and transmit, controlled by e.g. Two Finite State Machines or implementation of all the functionality in a single automaton. Selected approach brings some risk of deadlock caused by cross-dependencies of the three automata. Its key advantage however is preservation of natural concurrency of operation of the three modules. Functionality of USB interface

quite often requires simultaneous processing of various parts of transmitted data [2].

Babulu et al. (2008), described The Universal Serial Bus (USB) Transceiver Macro cell Interface (UTMI) is a two wire, bi- directional serial bus interface between USB devices through D+ and D- lines. There are three functional blocks present in USB controller; those are Serial Interface Engine (SIE), UTMI and Device Specific Logic (DSL).When the data is received on the serial bus, it is decoded, bit unstuffed and is sent to receive shift register. After the shift register is full, the data is sent to receive hold register [3].

Jolfaei et al. (2009) designed a High speed easy to use peripheral interfaces like USB 2.0.

Implemented hardware on a Spartan-3 FPGA is easily capable of handling data throughputs as high as what USB 2.0 demands. Using dedicated resources such as multipliers it is also possible to do signal processing tasks on these data. In addition, the large amount of available logic resources on FPGA allows integration of a complete system on a single chip. This makes Spartan-3 FPGA a suitable choice for use in USB peripheral designs [4].

3. PROPOSED METHOLOGY

High volume USB 2.0 devices will be designed using ASIC technology with embedded USB 2.0 support. For full-speed USB devices the operating frequency was low enough to allow data recovery to be handled in a vendors VHDL code, with the ASIC vendor providing only a simple level translator to meet the USB signaling requirements. Today's gate arrays operate comfortably between 30 and 60 MHz with USB 2.0 signaling running at hundreds of MHz, the existing design methodology must change. As operating frequencies go up it becomes more difficult to compile VHDL code without modification. This report defines the USB 2.0 Transceiver Macro cell Interface (UTMI) and many operational aspects of the USB 2.0 Transceiver Macro cell (UTM). The intent of the UTMI is to accelerate USB 2.0 peripheral development. This document defines an interface to which ASIC and peripheral vendors can develop. ASIC vendors and foundries will implement the UTM and add it to their device libraries. Peripheral and IP vendors will be able to develop their designs, insulated from the high-speed and analog circuitry issues associated with the USB 2.0 interface, thus minimizing the time and risk of their development cycles.

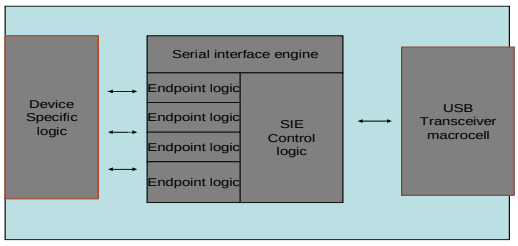

Figure 3.1: USB 2.0 Transceiver Macro cell (UTM)[7]

There are assumed to be three major functional blocks in a USB 2.0 peripheral ASIC design: the USB 2.0 Transceiver Macro cell, the Serial Interface Engine (SIE), and the device specific logic.

3.2 USB 2.0 Transceiver Macro cell:

This block handles the low level USB protocol and signaling. This includes features such as; data serialization and deserialization, bit stuffing and clock recovery and synchronization. The primary focus of this block is to shift the clock domain of the data from the USB 2.0 rate to one that is compatible with the general logic in the ASIC.

Some key features of the USB 2.0 Transceiver are:

1. Eliminates high speed USB 2.0 logic design for peripheral developers.

2. Standard Transceiver interface enables multiple IP sources for USB 2.0 SIE VHDL.
3. Supports 480 Mbit/s "High Speed" (HS)/ 12 Mbit/s “Full Speed” (FS), FS Only and "Low Speed" (LS) Only 1.5 Mbit/s serial data transmission rates.
4. Utilizes 8-bit parallel interface to transmit and receive USB 2.0 cable data.
5. SYNC/EOP generation and checking.
6. Allows integration of high speed components in to a single functional block as seen by the peripheral designer.
7. High Speed and Full Speed operation to support the development of "Dual Mode" devices.
8. Data and clock recovery from serial stream on the USB.
9. Bit-stuffing/unsnuffing; bit stuff error detection.
10. Holding registers to stage transmit and receive data.
11. Logic to facilitate Resume signaling.
12. Logic to facilitate Wake Up and suspend detection.
13. Supports USB 2.0 Test Modes.
14. Ability to switch between FS and HS terminations/signaling.
15. Single parallel data clock output with on-chip PLL to generate higher speed serial data clocks.

A HS/FS implementation of the transceiver can operate at either a 480 Mb/s or a 12 Mb/s rate. Two modes of operation are required to properly emulate High-speed device connection and suspend/resume features of USB 2.0, as well as Full-speed connections if implementing a Dual-Mode device. FS Only and LS Only UTM implementations do not require the speed selection signals since there is no alternate speed to switch to.

The USB 2.0 Transceiver can be placed in a low-power mode with the SuspendM signal.

3.3 Transmitter

Transmitter must undertake several tasks. It converts data from parallel to serial. This data will then have to be bit stuffed. The USB protocol requires a zero to be stuffed after every six consecutive ones. This is required so that there is a transition at least every six bits of the transmitter output. The stuffed binary data is eventually converted to NRZI encoding before being sent through the transceiver to the USB host. The transmitter is also responsible for forming data and handshake packets before sending them to the host. The transmitter is responsible for calculating the 16-bit CRC included in data packets sent to the host.

3.4 Block level Descriptions

This section provides descriptions of each of the blocks shown in Figure 3.2 above. These blocks represent high level functionality that is required to exist in the Macro -cell.

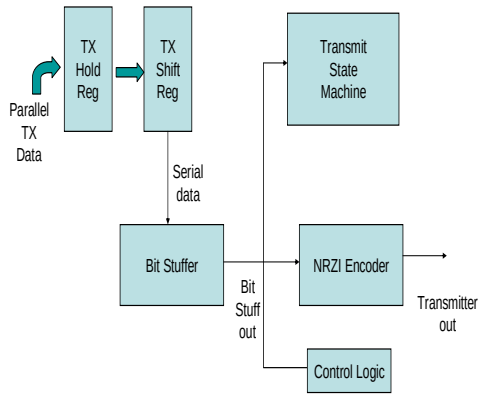

Figure 3.2: USB 2.0 Transceiver Macro cell (Transmitter)[7]

3.4 Receiver:

The receiver's function is the inverse of the transmitter. The receiver is more complicated than the transmitter and requires further functionality. The receiver has capabilities to calculate both 5-bit CRC and 16-bit CRC. This is necessary since the receiver receives token packets, which contain a 5-bit CRC and data packets, which contain a 16-bit CRC. The transmitter compares the value it calculated with that received from the packets. If there is any inconsistency, retransmission is requested from the host.

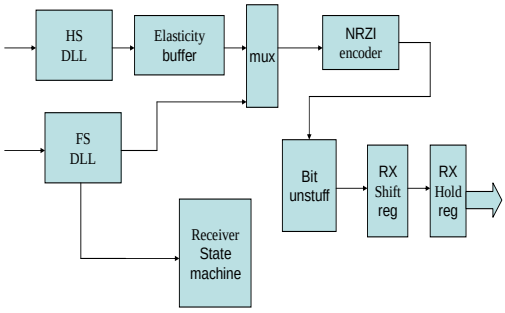

Figure 3.3: USB 2.0 Transceiver (Receiver) [7]

4. EXPECTED RESULTS AND DISCUSSIONS

Though the finished USB 2.0 Transceiver Macro cell will physically tested on an FPGA, it is hoped that this will have been achieved by the demonstration day. As usual with VLSI designs the testing phase will be expensive time wise. The testing procedure that was used for the project is given below

Initially the Verilog coding of the transmitter will be prepared which includes:

1. TX Shift/Hold Register
2. NRZI Encoder
3. Bit stuffer Logic
4. Transmit State Machine.

After the completion of the transmitter the coding of the receiver will start and simultaneously tested as shown:

1. Rx Shift/Hold Register.
2. NRZI Decoder.
3. Bit Unstuffed Logic.
4. Receive State Machine.
5. Full Speed DLL.

Now the transmitter and the receiver will completed, and the work on the USB controller started which included the Verilog coding and the testing of all the modules of the controller.

1. Controller state machine
2. SOP Detector
3. PID Checker
4. PID Generator

5. CONCLUSION

Reviewed USB Trans receiver microcell is useful in Linux and android operating system and hardware architecture of high speed data communication for USB 2.0 device using FPGA is well reviewed.

6. REFERENCES

[1] Babulu, K., & Rajan, K. S. (2008, July). FPGA Implementation of USB Transceiver Macrocell Interface with USB2. 0 Specifications. In Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on (pp. 966-970). IEEE.

[2] Jolfaei, F. A., Mohammadizadeh, N., Sadri, M. S., & FaniSani, F. (2009, December). High speed USB 2.0 interface for FPGA based embedded systems. In Embedded and Multimedia Computing, 2009. EM-Com 2009. 4th International Conference on (pp. 1-6). IEEE.

[3] Guo, G., Li, Z., & Yang, F. (2011, July). Design of high speed pulse data acquisition system based on FPGA and USB. In Multimedia Technology (ICMT), 2011 International Conference on (pp. 5374-5376). IEEE.

[4] Lun, C. C., bin Marzuki, A., & Wei, S. H. (2012, June). Analog front-end design implementation of USB2. 0 OTG Attach Detection Protocol. In Intelligent and Advanced Systems (ICIAS), 2012 4th International Conference on (Vol. 2, pp. 774-779). IEEE.

[5] Szecówka, P. M., & Pyrzynski, K. J. (2012, September). USB receiver/transmitter for FPGA implementation. In Signals and Electronic Systems (ICSES), 2012 International Conference on (pp. 1-6). IEEE.

[6] Pandey, M. K., Shekhar, S., Singh, J., Agarwal, G. K., & Saxena, N. (2013, July). A novel approach for USB2. 0 validation on System on Chip. In 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT) (pp. 1-4). IEEE.

[7]http://www.scribd.com/doc/97042848/USB-2-0 TransceiverMacrocellInterfface-UTMI-SpecXcvr-Macrocell-1-05#scribd

[8]http://resourcecentre.daiict.ac.in/resources/btp/full_catalogue_2006_10.pdf