INTEGRATING HIGH SPEED USB FUNCTIONALITY FOR FPGA- AND ASIC-BASED DESIGNS

By Hridya Valsaraju, Cypress Semiconductor Corp., and Gopalakrishnan Vijayakumar, Cypress Semiconductor Corp.

The Universal Serial Bus has earned the popularity it now enjoys based on the merits of its ease of use, plug-and-play capabilities, and robustness. USB has, more or less, found its way into all the computer peripherals that once contained UARTs or parallel ports as their host interfaces and any product which requires an interface to a host computer now considers USB as its primary option. The various bandwidth choices that the USB offers -- Low, Full, High and now Super speed -- cater to a variety of computer peripherals as well as industrial and medical equipment.

The throughput offered by USB is sufficient even for high bandwidth applications like hard disk drives and scanners. Indeed, for most of the computer peripherals like keyboards and mice, PDAs, gamepads, joysticks, scanners, digital cameras and printers, USB is now the standard interconnection method.

Apart from simple computer peripheral devices, a wide range of FPGA-0based applications exist which can benefit greatly from the addition of a high speed USB interface. Digital Signal Oscilloscopes, ECGs, Video Cameras and Data acquisition systems are a few such devices. Adding a USB interface to a device will broaden its capabilities to a huge extent. For example, in case of data recording systems like DSOs and ECGs, the acquired data can be transferred real-time over to a host machine which, in turn, can transmit this data over the Internet as shown in Figure 1. For applications like remote data logging, connecting the data logger device via USB to a host can also enable the device to be controlled remotely from a master host machine that is located miles away but connected to the USB host over the Internet. This article explores possible design methodologies which can be utilized to implement an efficient high speed USB 2.0 interface in an FPGA- or ASIC-based system.

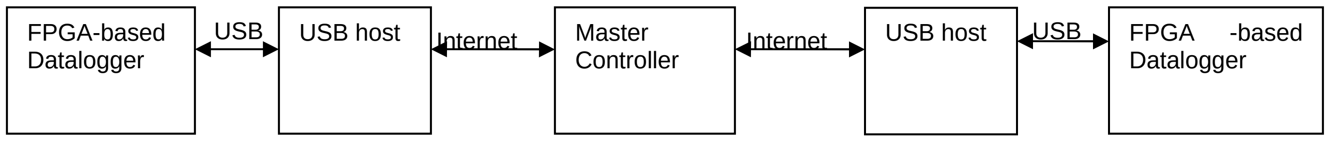

Figure 1. A USB-based distributed data acquisition system

In order to look at the various ways in which we can integrate a universal serial bus interface into an FPGA or an ASIC based system, we need to understand how a typical USB system works. A typical USB system would consist of a transceiver, a serial interface engine (SIE), and an interface controller as shown in Figure 2.

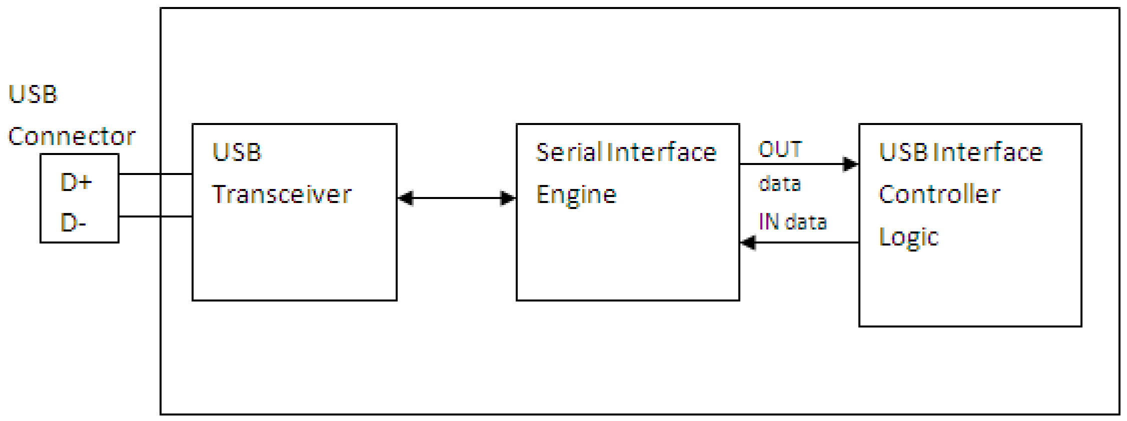

Figure 2. A typical USB system

The transceiver implements the physical layer of the USB protocol. The transceiver forms the two ends of the data link between the host and the device. USB transceivers will have both differential and single ended outputs. The SIE performs encoding and decoding of the serial data, bit stuffing, error correction, and other signal-level housekeeping tasks. It also converts parallel data to serial data and vice versa . The SIE should be interfaced to an intelligent master which should implement the high-level USB protocol by responding to host requests over the control endpoint. The master will also perform the application level tasks as required by the device functionality.

Here we will discuss the three different methods in which a USB interface can be added to an FPGA- or ASIC-based system.

Usage of USB Protocol Stack Intellectual property (IP) Along With an External Transceiver

In the very first method, one can implement the SIE within the FPGA or the ASIC (see Figure 3).

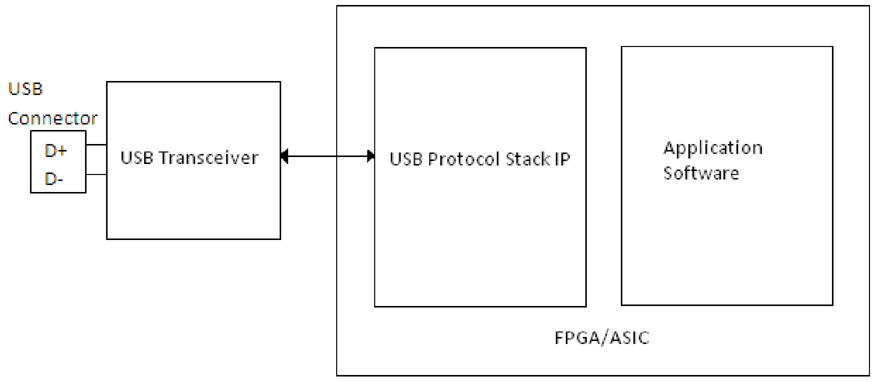

Fig 3. FPGA + USB Protocol Stack IP + USB Transceiver

The main functions that the SIE IP would have to perform are data encoding and decoding, CRC generation and error correction, bit stuffing, packet ID generation, serial-to-parallel and parallel-to-serial conversion. This IP can either be developed indigenously or can be bought from third party vendors. The FPGA would also have to perform high level USB protocol implementation. A transceiver IC would have to be used externally, such as the Cypress TX2UL transceiver chip. This transceiver is USB 2.0 certified and is compliant with the Tranceiver Macrocell Interface(UTMI) specification. All the required terminations, including 1.5 Kohm pull up on the Dplus line, are internal to the chip.

The advantage of this approach is that minimal external hardware is required. The disadvantages of using this approach are that developing the complicated USB stack IP requires time as well as engineering resources. The implementation of the SIE IP, the higher level protocol, and application functions also take up a significant amount of valuable FPGA resources. Usage of third-party IP may also prove costly.

Usage of a USB Bridge IC with integrated SIE and Transceiver

A second way to integrate a USB interface into an FPGA design would be to interface an external serial interface engine IC as well as a transceiver (see Figure 4).

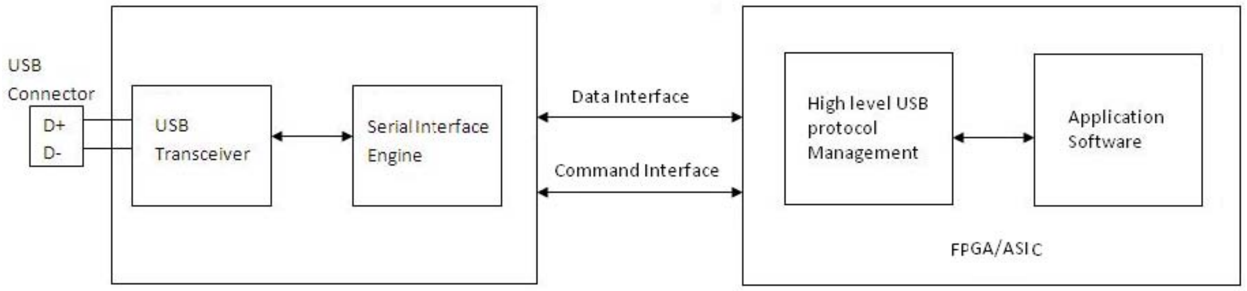

Figure 4. FPGA + Bridge IC(SIE + PHY)

Here, the FPGA will only need to implement the high-level USB protocol layer. Any command received from the USB host would be relayed by the SIE to the FPGA. The FPGA would need to contain logic to respond appropriately to these commands. For example, in the process of enumeration, the USB device would get a command from the host requesting for its descriptors. In this implementation, the external hardware, i.e., the transceiver and the SIE, would receive the packet and notify the FPGA via an interrupt that a command has been received. The FPGA would then have to read a register in the SIE to understand that the device has received a GET Descriptor command from the host. It would then have to send the correct descriptors to the SIE. Rather than using an SIE IC with an external transceiver, USB bridge ICs such as the Cypress Mobl-USB can be used which have a built-in SIE as well as transceiver. The controller has two double buffered high-speed capable endpoints that share a 2 KB FIFO space for maximum flexibility and throughput, as well as the Control Endpoint 0.

The major advantage of such an implementation is that half of the complexity (i.e., signal-level protocol management) has been shifted out of the FPGA. However, even in this scenario, some amount of FPGA resources would still have to be utilized for the higher level USB protocol implementation. In this case, a decision would have to be arrived at by analyzing the cost of the external hardware required vs. the amount of FPGA resources saved.

Usage of a USB Controller with FPGA

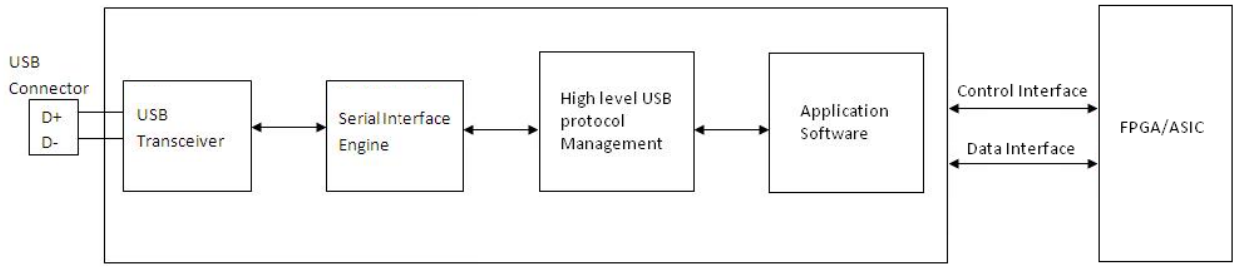

Figure 5. USB Controller + FPGA

The final technique is that of interfacing a full-fledged USB controller to the FPGA (see Figure 5). In this case, all the USB protocol level management would take place outside the FPGA. A dedicated USB controller IC will contain the PHY, the SIE, and the protocol control logic.

Controllers can offer the flexibility of a configurable number of endpoints, FIFO size, and may even contain a microcontroller to handle high-level USB protocol management. Such peripheral controllers may also contain the capability to handle some application-level functions, thus freeing the FPGA or ASIC from the need to perform them. If they contain a RAM-based architecture, the developer is also offered the flexibility of in-field firmware upgrades.

The FPGA will now be required to hold only the logic to interface to this chip. The interface between the USB controller and the FPGA can be any of the standard interfaces like SPI, I^{2}C, or HPI. It might also be a custom interface that is particular to the USB controller IC.

The Cypress FX2LP is a dedicated peripheral controller chip that can reduce the development time for the integration of USB into an FPGA- or ASIC-based system. It contains a simple “Slave FIFO” interface which enables a simple interconnection with an FPGA-based system. The Slave FIFO interface functions with simple Slave Read and Slave Write signals which can read or write data into the FX2LP’s 4K FIFO space.

For interfacing with an ASIC-based system, the FX2LP contains a GPIF or General Programmable Interface logic which can generate interface waveform for any standard interface that the ASIC possesses for communication. The GPIF also offers the option of configuring the FPGA over the USB interface thus eliminating the need for a separate configuration chip like a PROM or a processor. Configuring the FPGA over USB can also eliminate JTAG headers from the board. Hence, a versatile, dedicated USB controller can act as more than a just a data conduit by reducing board size and cost.

All the three design techniques mentioned in this article have their own merits and demerits. A selection has to be made based on the careful analysis of the trade-offs acceptable between cost, board space, and FPGA resources.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com