Implementation of Data Transfer Protocol Using USB 2.0 on Blackfin ADSP

1Aditya N S, 2Chethana T, 3Sahana M, 4Sanjana BJ
1,2,3,4Department of ECE, SIR MVIT

Abstract-- The main objective of this project is to achieve data transfer through the USB 2.0 port between two Blackfin processors. The Blackfin is a family of 16- or 32-bit microprocessors developed, manufactured and marketed by Analog Devices. The processors have built-in fixed-point digital signal processor (DSP) functionality supplied by 16-bit Multiply-accumulates (MACs), accompanied by an on-chip small microcontroller. One of these processors (BF548) is configured as the host and the other (BF527) as the device. The motivation for the Universal Serial Bus (USB) came from the three interrelated considerations: connection of the PC to various devices, ease-of-use and port expansion. The more recent motivation for USB 2.0 stems from the fact that PCs have increasingly higher performance and are capable of processing vast amounts of data. USB 2.0 addresses this need by adding a third transfer rate of 480 Mb/s to the 12 Mb/s and 1.5 Mb/s originally defined for USB. USB 2.0 is a natural evolution of USB, delivering the desired bandwidth increase while preserving the original motivations for USB and maintaining full compatibility with existing peripherals. To facilitate the data transfer we use an IDE-VISUAL DSP ++ 5.0 software. VisualDSP++ includes all the tools that are required to build and manage processor projects. The programming of the processors required for the data transfer is done suitably using embedded C. Finally, the data transfer is verified by monitoring the contents of the Blackfin memory before and after transmission with the help of visual DSP.

Keywords: USB, Blackfin processors,

I. INTRODUCTION TO USB

The Universal Serial Bus (USB) is the technology that allows us to connect an electronic device to computer. It is a fast serial bus. A USB port is a standard cable connection interface for personal computers and consumer electronics devices. It is an industry standard for short-distance digital data communications. USB ports allow USB devices to be connected and transfer digital data over USB cables. USB ports can also supply electric power across the cable to devices that need it. The USB Specification defines a master-slave communications system and details two distinct roles - a host that is in control of all communications and a function that provides services to the host. Initial implementations partitioned these roles into individual silicon components. The recent “On-The-Go” (OTG) supplement extends the original usage model for USB by adding the capability to build a dual-role device. Both wired and wireless versions of the USB standard exist, although only the wired version involves USB ports and cables. USB ports and cables are used to connect hardware such as printers, scanners, keyboards, mice, flash drives, external hard drives, joysticks, cameras, and more to computers of all kinds, including desktops, tablets, laptops, net books, etc.

II. USB PINS

USB is a serial bus, using four shielded wires for the USB 2.0 variant: two for power (V_BUS and GND), and two for differential data signals (labeled as D+ and D− in pin outs). Non-Return-to-Zero Inverted (NRZI) encoding scheme is used for transferring data, with a sync field to synchronize the host and receiver clocks. D+ and D− signals are transmitted on a differential pair, providing half-duplex data transfers for USB 2.0.

Table-I Speeds of Operation

<table>
<thead>
<tr>
<th>Release name</th>
<th>Release Date</th>
<th>Maximum Transfer Rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB 1.0</td>
<td>January 1996</td>
<td>Low Speed (1.5 Mbps)</td>
</tr>
<tr>
<td>USB 1.1</td>
<td>August 1996</td>
<td>Full Speed (12 Mbps)</td>
</tr>
<tr>
<td>USB 2.0</td>
<td>April 2000</td>
<td>High Speed (480 Mbps)</td>
</tr>
<tr>
<td>USB 3.0</td>
<td>November 2008</td>
<td>Super Speed (6 Gbps)</td>
</tr>
<tr>
<td>USB 3.1</td>
<td>July 2013</td>
<td>SuperSpeed+ (10 Gbps)</td>
</tr>
</tbody>
</table>
III. USB WORKING

The design architecture of USB is asymmetrical in its topology, consisting of a host, a multitude of downstream USB ports, and multiple peripheral devices connected in a tiered-star topology. Additional USB hubs may be included in the tiers, allowing branching into a tree structure with up to five tier levels. A USB host may implement multiple host controllers and each host controller may provide one or more USB ports. Up to 127 devices, including hub devices if present may be connected to a single host controller. USB devices are linked in series through hubs. One hub—built into the host controller—is the root hub.

![USB Endpoints](image)

A physical USB device may consist of several logical sub-devices that are referred to as device functions. USB endpoints actually reside on the connected device: the channels to the host are referred to as pipes USB device communication is based on pipes (logical channels). A pipe is a connection from the host controller to a logical entity, found on a device, and named an endpoint. Because pipes correspond 1-to-1 to endpoints, the terms are sometimes used interchangeably. A USB device could have up to 32 endpoints (16 IN, 16 OUT), though it's rare to have so many. An endpoint is defined and numbered by the device during initialization (the period after physical connection called "enumeration") and so is relatively permanent, whereas a pipe may be opened and closed. There are two types of pipe: stream and message. A message pipe is bi-directional and is used for control transfers. A stream pipe is a uni-directional pipe connected to a uni-directional endpoint that transfers data using an isochronous, interrupt, or bulk transfer.

<table>
<thead>
<tr>
<th>Transfer Mode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Control</td>
<td>Exchange configuration, setup, and control information between the device and the host.</td>
</tr>
<tr>
<td>2. Isochronous</td>
<td>Isochronous transfer is used by client, server, device such as speakers and video camera.</td>
</tr>
<tr>
<td>3. Bulk</td>
<td>Bulk transfer is used by device like printer &amp; scanners, which receives data in one big packet.</td>
</tr>
<tr>
<td>4. Interrupt</td>
<td>Interrupt transfer is used by peripherals exchanging small amount of data, like mouse or keyboard.</td>
</tr>
</tbody>
</table>

When a USB device is first connected to a USB host, the USB device enumeration process is started. The enumeration starts by sending a reset signal to the USB device. The data rate of the USB device is determined during the reset signaling. After reset, the USB device's information is read by the host and the device is assigned a unique 7-bit address. If the USB host is restarted, the enumeration process is repeated for all connected devices. In USB 2.0, the host controller polls the bus for traffic, usually in a round-robin fashion.

IV. BLACKFIN PROCESSORS

The Blackfin is a family of 16- or 32-bit microprocessors developed, manufactured and marketed by Analog Devices. Blackfin DSPs are optimized for processing data, communications and video streams for penetration into new market spaces. It has Dynamic Power Management capabilities which deliver the lowest power consumption. The processors have built-in fixed-point digital signal processor (DSP) functionality supplied by 16-bit Multiply-accumulates (MACs), accompanied by an on-chip small microcontroller. The combination of Digital Signal Processing and Microcontroller functionality was designed to improve performance, programmability and power consumption over traditional DSP or RISC architecture designs.

A. FEATURES

**Controller**-
- L1 memory space for stack and heap
- Dedicated stack and frame pointers
- Byte addressability
- Simple bit-level manipulation

**DSP**-
- Fast, flexible arithmetic computational units
- Unconstrained data flow to/from computational units
- Extended precision and dynamic range
- Efficient sequencing
- Efficient I/O processing
• The DSP aspect of the Blackfin core is optimized to perform FFTs and Convolutions.

B. ARCHITECTURE

• Two 16-bit MACs, two 40-bit ALUs, four 8-bit Video ALUs support for 8/16/32 bit integer and 16/32-bit fractional data types.
• Concurrent Fetch of One instruction and two unique data elements.
• Two loop counters that allow for nested zero-overhead looping.
• A Modified Harvard architecture in combinational with a hierarchical memory.
• Arbitrary bit and bit field manipulation, insertion and extraction.
• Two data address generator (DAG) units with circular and bit-reversed addressing.
• Address register file consists of six 32-bit general purpose pointer registers and four 32-bit circular buffer addressing registers.
• Unified 4GB memory space.
• Mixed 16/32-bit instruction encoding for best code density.
• Memory protection for support of OS operation.

C. BF527

• Host DMA
• USB
• Ethernet MAC
• Internal Voltage Regulator
• TWI
• Two SPORTs
• Two UARTs
• SPI
• Eight GP Timers
• GP Counter
• Watchdog Timer
• RTC
• PPI
• 48 GPIOs

• L1 Instruction SRAM – 48K Bytes
• L1 Instruction SRAM/ Cache – 16K Bytes
• L1 Data SRAM – 32K Bytes
• L1 Data SRAM/ Cache – 32K Bytes
• L1 Scratchpad SRAM – 4K Bytes
• L1 ROM – 64K Bytes
• L2- 128K Bytes
• L3 Boot ROM – 4K Bytes

D. BF-548

• Lockbox Code Security
• SD/SDIO Controller
• Pixel Compositor
• NAND Flash Controller
• ATAPI
• High Speed USB OTG
• Host DMA
• Two CAN Ports
• Ethernet MAC
• Internal Voltage Regulator
• Two TWI Ports
• Four SPORTs
• Four UARTs Ports
• Three PPIs
• Three SPI Ports
• Eleven Timers
• One Up/Down Counter
• 152 GPIO Pins

• L1 Instruction SRAM – 48K Bytes
• L1 Instruction SRAM/ Cache – 16K Bytes
• L1 Data SRAM – 32K Bytes
• L1 Data SRAM/ Cache – 32K Bytes
• L1 Scratchpad SRAM – 4K Bytes
• L1 ROM – 64K Bytes
• L2- 128K Bytes
• L3 Boot ROM – 4K Bytes

• Maximum Core Instruction Rate – 533MHz

V. DATA TRANSFER

A transfer takes place every time data is moved between the host controller and the USB device. In general, USB transfers can be broadly categorized into control transfers and data transfers. All USB devices must support control transfers and can support endpoints for data transfers. Each type of transfer is associated with the type of USB endpoint (a buffer in the device). Control transfer is associated with the default endpoint and data transfers use unidirectional endpoints.

A. BULK LOOPBACK

As a first step in achieving this, we verify successful data transfer from a PC host to the device (blackfin ADSP) and back to the PC, forming a loop and hence the name “Loopback”. Loopback, or loop-back, refers to the routing of electronic signals, digital data streams, or flows of items back to their source without intentional processing or modification. This is primarily a means of testing the transmission or transportation infrastructure. By running loopback, we extract the protocol flow which forms the core part of the logic behind the data transfer.
VI. PROTOCOLS USED for DATA TRANSFER:

A. Attach Detection Protocol (ADP)

Allows an OTG device, embedded host or USB device to determine attachment status in the absence of power on the USB bus, enabling both insertion-based behavior and the capability to display attachment status. It does so by periodically measuring the capacitance on the USB port to determine whether there is another device attached, a dangling cable, or no cable. When a large enough change in capacitance is detected to indicate device attachment, an A-device will provide power to the USB bus and look for device connection. At the same time, a B-device will generate SRP and wait for the USB bus to become powered.

B. Session Request Protocol (SRP)

Allows both communicating devices to control when the link's power session is active; in standard USB, only the host is capable of doing so. That allows fine control over the power consumption, which is very important for battery-operated devices such as cameras and mobile phones. The OTG or embedded host can leave the USB link unpowered until the peripheral requires power.

C. Host Negotiation Protocol (HNP)

Allows the two devices to exchange their host/peripheral roles, provided both are OTG dual-role devices. By using HNP for reversing host/peripheral roles, the USB OTG device is capable of acquiring control of data-transfer scheduling. Thus, any OTG device is capable of initiating data-transfer over USB OTG bus. The latest version of the supplement also introduced HNP polling, in which the host device periodically polls the peripheral during an active session to determine whether it wishes to become a host.

VII. INTERFACE OVERVIEW

It is important to understand the functionality when a USB is configured as a host or as a device. When operating in host mode, the USB controller uses simple hosting capabilities to master point-to-point connections with another USB peripheral, initiating transfers on the bus for the peripheral to respond. USB IN packets are received into the RX FIFOs to be moved into the processor core memory, and data written into TX FIFOs is transmitted onto the bus as USB OUT packets. In this mode, the USB controller encodes, decodes, and checks USB packets sent and received. The controller automatically schedules isochronous and interrupt transfers from the endpoint buffers such that one transaction is performed every n frames, where n represents the polling interval programmed for the endpoint.

In peripheral/device mode, the USB controller encodes, decodes, checks, and directs all USB packets sent and received, responding appropriately to host requests. Data is transferred from the processor core memory into the device’s TX FIFOs to be transmitted onto USB as IN packets. In the other direction USB OUT packets are received into the RX FIFOs (having been sent from the host) and transferred to system memory for processing or storage. In peripheral mode, the USB controller acts as a slave device to another USB host; either a personal computer or another OTG host controller.

VIII. SOFTWARE IMPLEMENTATION:

VisualDSP++ 5.0

VisualDSP++ includes all the tools that are required to build and manage processor projects.

A. VDSP Features

- Integrated Development and Debugging Environment (IDDE) with VisualDSP++ Kernel (VDK) integration
- C/C++ optimizing compiler with run-time library.
- Assembler, linker, preprocessor, and archiver
Loader and splitter
Simulator
EZ-KIT Lite% evaluation system
Emulator

B. Source File Editing Features

VisualDSP++ simplifies tasks involving source files. All the activities necessary to create, view, print, move within, and locate information are easy to perform. Create and modify source files and view listing or map files generated by the code development tools. Source files are the C/C++ language or assembly language files that make up your project. Processor projects can include additional files such as data files and a Linker Description File (.ldf), which contains command input for the linker.

- Editor windows. Open multiple editor windows (source windows) to view and edit related files, or open multiple editor windows for a single file.
- Specify syntax coloring. This feature enhances the view and helps locate portions of the text, because keywords, quotes, and comments appear in distinct colors.
- Context-sensitive expression evaluation: Move the mouse pointer over a variable that is in the scope to view the variable’s value.
- Status icons: View icons that indicate breakpoints, bookmarks, and the current PC position.
- View error details and offending code.

C. Debugging Features

While debugging your project, one can:

- View and debug mixed C/C++ and assembly code.
- Use memory expressions. Use expressions that refer to memory.
- Use breakpoints to view registers and memory. Quickly add and remove, and enable and disable breakpoints.
- Set simulated watchpoints.
- Statistically profile the target processor’s PC (JTAG emulator debug targets only).
- Linearly profile the target processor’s PC (Simulation only).
- Generate interrupts using streaming I/O. Set up serial port (SPORT) or memory-mapped I/O.
- Create customized register windows.
- Plot values from processor memory
- Trace how your program arrives at a certain point and show reads, writes, and symbolic names.
- View pipeline depth of assembly instructions. Display the pipeline stage by querying the target processor(s) through the pipeline interface

D. VisualDSP++ Kernel

A project can optionally include the VisualDSP++ Kernel (VDK), which is a software executive between algorithms, peripherals, and control logic.

E. VisualDSP++ Help System

The VisualDSP++ Help system is designed to help us obtain information quickly. VisualDSP++ Help is comprised of multiple Help systems. The majority of the Help system files are VisualDSP++ manuals and hardware documentation.

REFERENCES