IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 5, May 2016

USB3.0 Driver Development for Embedded Platforms which Currently support USB2.0

Chintan Vadaliya[1], Urmil Vyas[2]

Student, M.E. (Embedded System Design), GTU PG School, Gandhinagar, India[1]

Director, Sibridge Technologies, Ahmedabad, India[2]

Abstract: Universal Serial Bus 3.0(USB3.0) is the next major revision in USB standards which provides very high data rate between PCs and devices. In this paper we are going to discuss how to enable USB3.0 support for embedded boards which can support Mass Storage Function, HID Function and Video Function. For the development purpose we are using Xilinx Zynq ZC702 Board as an embedded platform in which the new drivers will be loaded. So, in this work the ZC702 Board operates as a mass storage device. Similarly in the case of HID and camera, the ZC702 Board operates as a keyboard and camera device respectively.

Keywords: Universal Serial Bus(USB), Human Interface Device(HID), Mass Storage Device(MSD), Mass Storage Class(MSC), Field-Programmable Gate Array(FPGA), FPGA Mezzanine Connector(FMC).

I. INTRODUCTION

Universal Serial Bus (USB) is a communication protocol which is used for data connection and power supply between computer and Peripheral Devices. USB3.0 introduces new transfer mode called 'Super Speed' which is capable of providing data rate up to 5Gb/s which is ten times faster than the USB2.0 high-speed(480 Mb/s)[1]. Very large number of devices are still using USB2.0 interface, so in order to enable the support for USB3.0, significant changes are required at hardware level as well as software level. In this work, we are using Xilinx Zynq ZC702 board which only supports USB2.0 currently. We are going to enable the support for USB3.0 in ZC702 board by connecting a daughter card with the board. The daughter card will provide the required hardware support for the USB3.0 interface. After that we are going to develop the device drivers for the ZC702 board which will provide software support for the USB3.0 interface.

II. TECHNICAL BACKGROUND

The USB architecture can be saturated in two parts: Device and Host. Device can be a function which provide specific capability. For example, USB keyboard provides Human Interface Device(HID) functionality while a pen-drive or HDD provides Mass Storage Device(MSD) functionality. The devices are connected to the USB port which will provide attachment with USB host controller.

Host will interact with the connected USB devices through the USB host controller. USB host will control all the activity in the USB system like detection of device, configuration of device and removal of device etc. Host will initiate all the transfer requests and the targeted device will only answer the request[5].

Device and host will communicate with each other via pipe. Pipe is a logical connection among device and host. Pipe provides unidirectional or bidirectional path between device and host. At device side the pipe is connected to an endpoint.

Endpoint are like a IP socket. It is the source of the data. Host initiate any transfer on a specific endpoint and targeted device will only response on that endpoint. Endpoints can be considered as interface between the hardware and firmware of device.[5] Device firmware will read/write data on endpoint and for hardware it will be the source/destination to send/receive data. Every device has at least one endpoint(EP0) which is also called as control endpoint. EP0 is used to configure the device when its attached. Multiple endpoints can be provided depending on the functionality of the device. The logical connection between host and device is shown in Fig. 1.

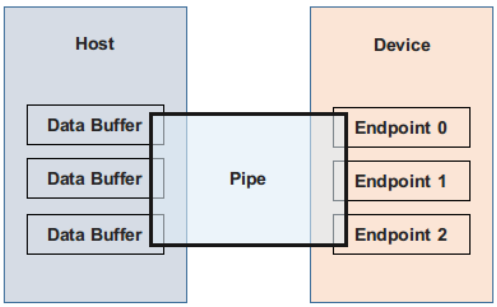

Fig. 1. Logical connection between device and host

USB devices has sets of descriptors which provides the information about device functionality, device manufacturer, supported USB version, numbers of endpoint etc. The hierarchy of USB descriptors is shown in Fig. 2[3].

Every device has only one device descriptor. Device descriptor will provide the details such as supported USB version, vendor and product ID and number of configuration device can provide[3].

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 5, May 2016

Configuration descriptor provides detail about the particular configuration such as required amount of power, device is self powered or bus powered and number of Interface it can have. Device can have multiple configuration descriptor as specified in device descriptor. After the device is connected the host will request for the device descriptor and then host can select the particular configuration. Only one configuration can be enabled at a time[3].

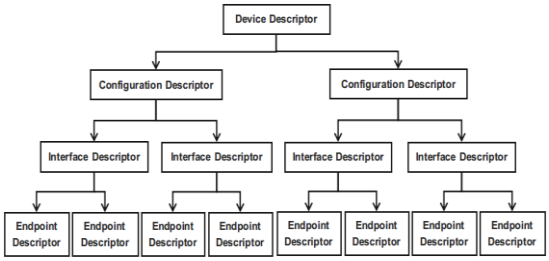

Fig. 2. Hierarchy of USB Descriptors

Device can have multiple interface descriptor as specified in configuration descriptor. Each interface can represent a feature of device. For example a multi-function device like printer/scanner/fax can have multiple interfaces, one for fax, one for printer and one for scanner. Device can have multiple interfaces enabled at a time. Each Interface will have one or more endpoint descriptor for different transfer types and direction[3].

III. HARDWARE

A. Xilinx Zynq ZC702

Xilinx Zynq ZC702 board is used as a USB device in this work. ZC702 provides embedded platform for the work and also includes industry-standard FPGA Mezzanine Connectors (FMC) which is used to connect and configure the daughter card. The board has ARM Dual-Core Cortex-A9 processor with Xilinx's 28nm programmable logic and has 1GB of RAM. ZC702 board has the support for USB2.0 transceiver only[4].

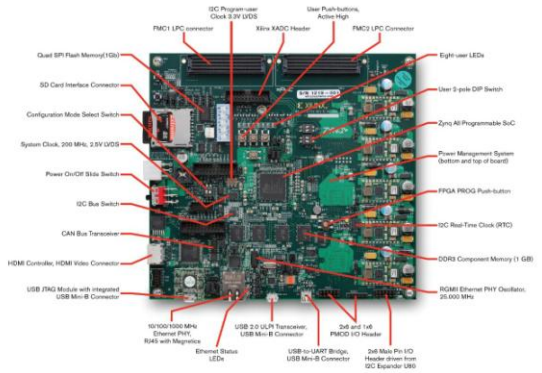

Fig. 3. Xilinx Zynq ZC702 Board

B. Daughter Card(USB3.0 PHY)

The daughter card will be connected to the Xilinx Zynq ZC702 board to provide the USB3.0 support. The daughter card is consists of FMC, USB3.0 PHY core, CMOS sensor connector and PS2 connector. The daughter card contains the TUSB1310A which is a single port, 4.8Gbps USB3.0 physical layer transceiver. The block diagram of the daughter card is shown in Fig. 4.

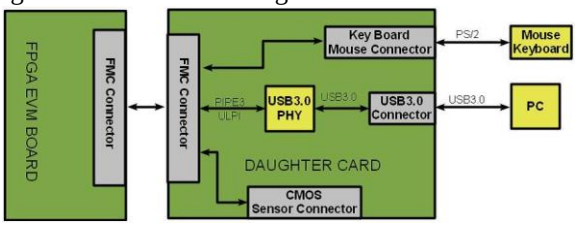

Fig. 4. Block Diagram of Daughter Card

IV. FIRMWARE

To enable the support for USB3.0 in the ZC702 board, we have to add device drivers for the USB3.0. We are going to use Linux kernel on the ZC702 board. ZC702 is an ARM platform, so we have to compile the Linux source using Xilinx ARM tool-chain after adding designed USB3.0 controller driver. The Linux USB driver framework is shown in Fig. 5[2].

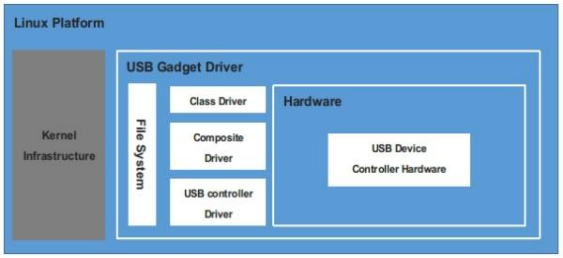

Fig. 5. Linux USB Driver Framework

The block diagram of USB driver interface between host and device is shown in Fig. 6.

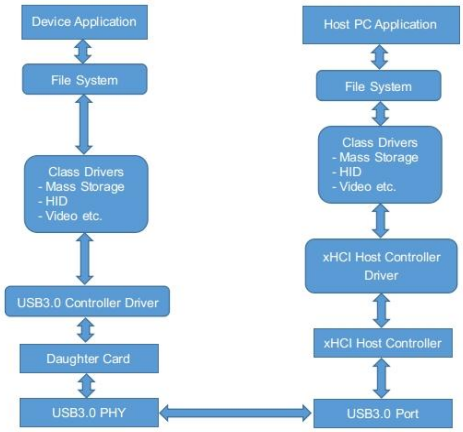

Fig. 6. USB Driver Interface between Host and Device

A. USB Controller Driver

USB3.0 controller driver will work as a Hardware Abstraction Layer(HAL) for the USB Device controller. In this work, the daughter card connected to the ZC702 will provide the USB3.0 device controller hardware. So, our task is to create the USB3.0 device controller driver to implement hardware specific routines which allows access to controller registers, memory space and interrupts.

USB3.0 controller driver will be added into the Linux source while compiling the source for the ARM platform.

DOI 10.17148/IJARCCE.2016.5550

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 5, May 2016

The drivers should be able to handle all the lower layer request for the hardware like data transmit/receive, device detection, speed detection, device management, allocating device resources, accessing controller registers to handle the USB transfer etc. The controller driver is also responsible to manage the device state. The state diagram of USB3.0 device is shown in Fig. 7[1].

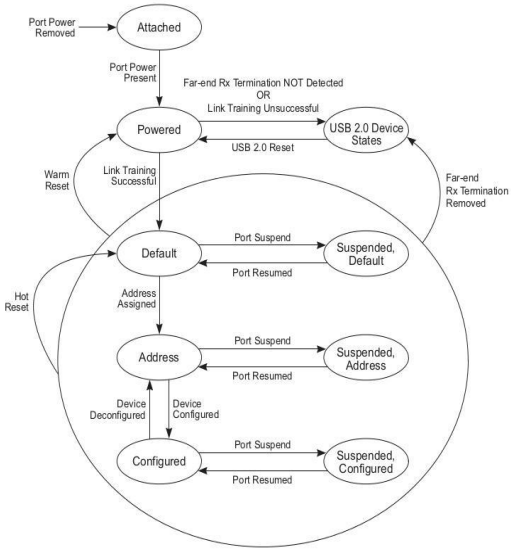

Fig. 7. USB Device State Diagram

B. Composite Driver

Composite Drivers are used to handle USB standard requests which are common for all the USB devices. It is also used to bind the class driver with the USB controller drivers. Some of the standard request which are handled in composite driver are listed below[1]:

- USB_REQ_SET_DESCRIPTOR
- USB_REQ_GET_STATUS
- USB_REQ_SET_CONFIGURATION
- USB_REQ_GET_CONFIGURATION
- USB_REQ_SET_INTERFACE
- USB_REQ_GET_INTERFACE
- USB_REQ_SET_FEATURE
- USB_REQ_GET_FEATURE

C. Class Driver

USB interface can support different types of functionality like Human Interface Device, Mass Storage Device, Audio Device, Video Device etc. So each functionality is considered as a USB class. Class driver provide the functionality support for the targeted device.[2] In this work, we are going to demonstrate the Mass Storage Class(MSC) driver. Mass Storage Class driver handles all file transfer operations between host and device. Some standard request are listed below:

- INQUIRY
- MODE_SELECT
- MODE_SENCE
- READ
- WRITE
- READ_CAPACITY
- READ_TOC
- READ_FORMAT_CAPACITIES

V. TEST AND RESULT

For the testing purpose we are using Mass Storage Class(MSC) driver. For this we have to provide any storage medium to the host to connect to. We are having 1GB of RAM in Xilinx Zynq ZC702 board, so we will use 512MB as the physical memory for the board and remaining 512MB will be used as the mass storage device. For this we have to make changes in the kernel configurations while compiling the Linux source. Once that has been made, we should be able to connect the 512MB of storage space to the host PC via USB3.0 interface. The screen shot from the host PC is shown in Fig. 8 after we connect the board to the USB3.0 port of the host PC.

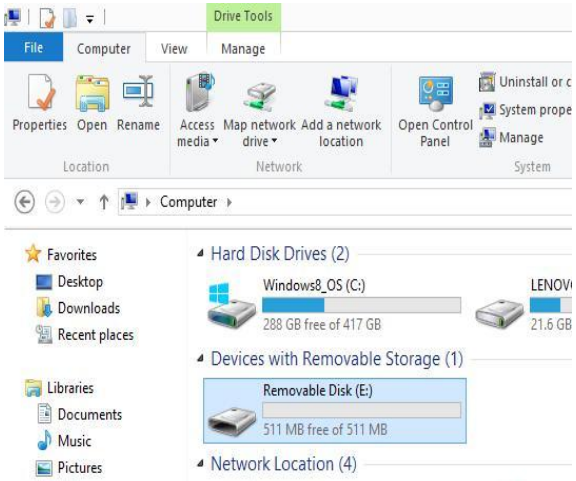

Fig. 8. USB3.0 Storage Device Mounted on Host PC

As you can see that the 512MB of removable storage device is detected on the host PC. Now we will run the data transfer test on the device using Crystal Disk Mark disk benchmark software[7]. The result is shown in Fig. 9.

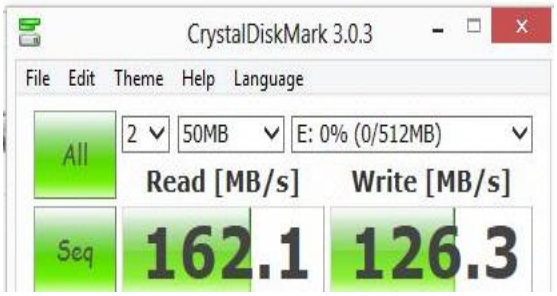

Fig. 9. Read/Write Speed on Storage Device

VI. CONCLUSION

We are able to provide the USB3.0 support in Xilinx Zync ZC702 board by applying specific changes in hardware(Daughter card) and Software(USB3.0 Controller Driver). We can also add support for HID/Audio/Video Class drivers. Similar can be done for other embedded board to enhance the data connection speed using USB3.0 interface.

DOI 10.17148/IJARCCE.2016.5550

ACKNOWLEDGMENT

I would like to thank Sibridge Technologies and especially **Urmil Vyas** (Director), **Bhavik Kothari**(Mem. Of Technical Staff), **Piyush Mehta**(Mem. Of Technical Staff) and **Tejas Patel**(Associate Engineer) for their help and support. It wouldn't have been possible to do this work without their guidance.

REFERENCES

[1]. Hewlett-Packard Company, Intel Corporation, Microsoft Corporation, NEC Corporation, ST-NXP Wireless and Texas Instruments. "Universal Serial Bus 3.0 Specification", 2nd Edition, Nov2012
[2]. Rajaram Regupathy, "Bootstrap yourself with Linux-USB Stack", 30-Jul-2012, pp. 156-315.
[3]. Beyond Logic Ltd "USB in Nutshell" Website,2nd Edition, Aug2011, http://www.beyondlogic.org/
[4]. www.xilinx.com/support/documentation/boards_and_kits/zc702_zvik/ug850-zc702-eval-bd.pdf
[5]. Jodeit M & Johns M. "USB Device Drivers: A Stepping Stone into Your Kernel", Computer Network Defense (EC2ND), 2010 European Conference, pp. 46-52
[6]. www.usb.org/
[7]. http://crystalmark.info/software/CrystalDiskMark/

DOI 10.17148/IJARCCE.2016.5550