USB-FPGA INTERFACE DOCUMENT

1 OVERVIEW

The overall framework of the USB-FPGA interface is shown below in Fig.1.

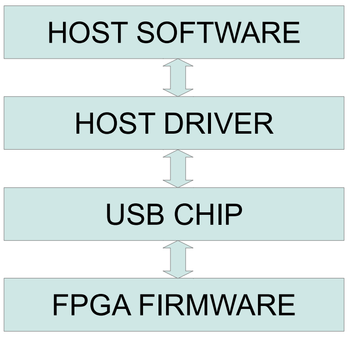

Fig. 1: Overall Framework

1.1 FPGA Firmware

The FPGA firmware serves as an interface between the USB chip and the internal FPGA logic. The details of the FPAG firmware will be introduced in section 2. Note that this FPGA firmware is developed for a specific setting of the USB chip. This means that, to use the FPGA firmware, the firmware of the USB chip must correctly pre-set the interfaces of the USB chip. The information of the USB chip interfaces will be introduced in section 3.

1.2 USB Chip

The USB chip on our board is CY7C60183A. Its interfaces are determined by the the USB chip firmware. The firmware can be loaded to the USB chip from the EEPROM chip or from the host PC by using some loader program. The information of the USB chip interfaces will be introduced in section 3.

1.3 Host Driver

The host drive for the USB chip used in our work is CyUSB.sys from Cypress. Generally two files are in the driver package. One is the .INF file and the other one is CyUSB.sys. The .INF file is used to map the devices to the .sys file. The PIDs and VIDs of the devices are used for the mapping. Therefore, two things need to be done. First, develop the USB chip firmware with the customized PID and VID. Then, add the PID and VID to the .INF file for the host PC to identify the driver of the USB chip.

1.4 Host Software

The software in my loop-back test is developed with Visual C++ Express. The CyAPI library is used for the control of the USB device. The program is just a simple Win32 console application with no GUI objects. More information will be introduced in section 4.

2 FPGA FIRMWARE

The interfaces of the FPGA firmware are shown in Fig.2. Its verilog file is "./fpga_firmware/src/usb_slave_fifo_interface.v"

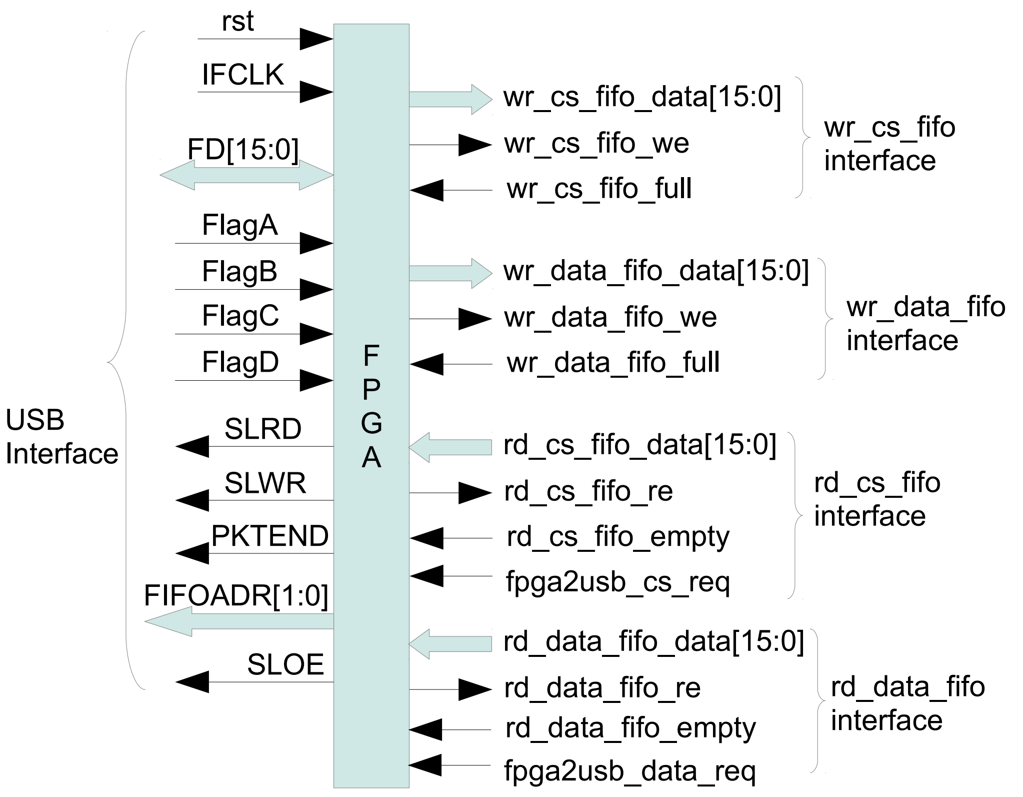

Fig. 2: FPGA Firmware

2.1 USB Interface

The requirements of the USB chip CY7C60183A are listed below.

All the signals in this interface should be configured as active high.

EP2 and EP4 should be configured as OUT end point.

EP6 and EP8 should be configured ad IN end point.

FLAGA should be configured as the empty flag of EP4 FIFO.

FLAGB should be configured as the empty flag of EP2 FIFO.

FLAGC should be configured as the full flag of EP8 FIFO.

FLAGD should be configured as the full flag of EP6 FIFO.

IFCLK should be configured as 48MHz and NO-INVERSE polarity.

WAKEUP should be configured as active low. It is used in the example design to generate reset.

2.2 wr_cs_fifo interface

This interface is a standard FIFO write interface. It connects to the USB out-endpoint EP4. The clock of this interface is IFCLK. This interface tries to transmit data from the EP4 FIFO as long as the EP4 FIFO is not empty and the signal wr_cs_fifo_full is 0.

2.3 wr_data_fifo interface

This interface is a standard FIFO write interface. It connects to the USB out-endpoint

EP2. The clock of this interface is IFCLK. This interface tries to transmit data from the EP2 FIFO as long as the EP2 FIFO is not empty and the signal wr_data_fifo_full is 0.

2.4 rd_cs_fifo interface

This interface is a standard FIFO read interface. It connects to the USB in-endpoint EP8. The clock of this interface is IFCLK. This interface tries to transmit data to the EP8 FIFO as long as the signal fpga2usb_cs_req is asserted, the EP8 FIFO is not full and the signal rd_cs_fifo_empty is 0.

2.5 rd_data_fifo interface

This interface is a standard FIFO read interface. It connects to the USB in-endpoint EP6. The clock of this interface is IFCLK. This interface tries to transmit data to the EP6 FIFO as long as the signal fpga2usb_data_req is asserted and the EP6 FIFO is not full, the EP6 FIFO is not full and the signal rd_data_fifo_empty is 0.

2.6 Priority

The priorities of the above four interfaces are listed below.

wr_cs_fifo > rd_cs_fifo > wr_data_fifo > rd_data_fifo.

3 USB CHIP

The behavior of the USB chip should be configured as required in section 2.1. A copy of the source code of the USB chip firmware is in the directory "./usb_firmware/". The firmware in that directory uses the VID and PID of Cypress. So, the users should customize these two numbers for their products. Note, after changing the VID and PID of the USB chip firmware, be sure to add the customized VID and PID to the .INF file of the driver.

The source code of the USB firmware is in "./usb_firmware/Fw/LP". The hex file is "fw.hex". The customized code is in the function "TD_Init" in the file "./usb_firmware/Fw/LP/periph.c".

The Keil uVision2 IDE is needed to compile the source code. This IDE can be obtained by installing the file SETUP_FX2LP_DVK_1004.exe, which can be downloaded from "http://www.cypress.com/?rID=14321". Installing this software also installs the general USB device driver from Cypress.

Finally, note that the above development environment is only for the WIN32 operating systems.

4 HOST PC DRIVER

In our test work, we have used the CyUSB.sys driver, which is encouraged by Cypress. The documentation of the supported IO codes can be found in the installation directory of SETUP_FX2LP_DVK_1004.exe.

5 HOST PC SOFTWARE

In our test work, we have used the CyAPI library, which is a higher level library of C++ classes for the control of Cypress USB devices. This library can be found in the installation directory of SETUP_FX2LP_DVK_1004.exe.

Visual C++ Express has been used for the test program development.

The test program just does loop-back test with randomly generated numbers.

4G bytes have been transferred from EP2 to EP6 and from EP4 to EP8 through the USB chip and the FPGA. No error was detected.

The source code of the software is in the directory "./test_usb_software".

6 Example Design

The example design of the FPGA firmware is shown in Fig.3. The verilog file is "./fpga_firmware/src/top.v".

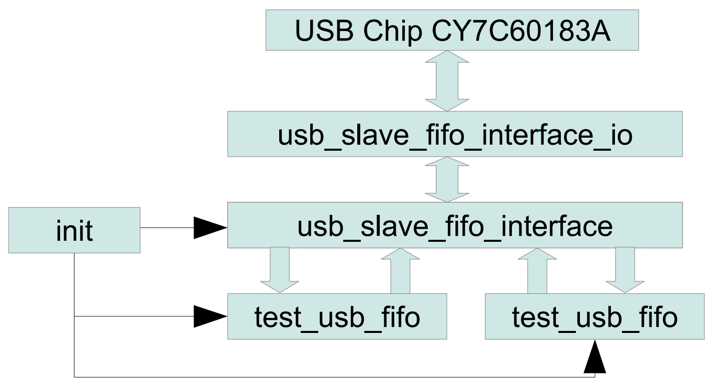

Fig. 3: Example FPGA Design

The "init" module is used to generate reset signals. The WAKEUP signal in the USB interface is connected to this module.

The "usb_slave_fifo_interface" module is the firmware introduced in section 2.

The "test_usb_fifo" modules connect to the four FIFO interfaces of the "usb_slave_fifo_interface" module.

The "usb_slave_fifo_interface_io" module is used to buffer the input and output USB signals and lock the USB interface clock.

The "top.ucf" file contains the firmware constraints. Note that the position of "IFCLK" is "p181". This is because the board I have is "PS_TDC_01_eval" not "TARGET". In the "TARGET" schematic , the position of IFCLK is "p148". Note that the position of IFCLK in "TARGET" may generate very poor timing performance, because "p148" is too far away from the clock buffers in the FPGA. Therefore, the pin mapping of the board "PS_TDC_01_eval" is better than that of "TARGET".

7 MISCELLANEOUS

When loading the USB firmware hex file into the USB chip with the EZ-USB console program provided by Cypress, be sure that the FPGA firmware has not been programmed yet. Otherwise, the contents in the registers in the USB chip may be destroyed and the driver may not be able to identify the device.

When programming the FPGA with the Xilinx Platform cable, if the USB chip is already working (which is the normal case), the in-endpoints of the USB chip may be filled with garbage data. Therefore, in this case, before using the in-endpoints for data transmission, the user should first flush the in-endpoints by performing reading from these endpoints until no garbage data is there.

8 OTHER REFERENCES

Refer to the "doc" directory for more documents.