

CERN Summer Student Programme 2014

FPGA Programming using FX3

Student: Stefano Calleja

CERN ID: 134521

Supervisor: Magnus Mager

Department: PH-AID-DT

Date: 22/08/2013

Table of Contents

Abstract 2

Introduction..... 3

Method 4

Results..... 7

Conclusion 7

Appendix..... 8

Abstract

An FPGA is required to be programmed via USB3 cable. Connectivity to the host PC is achieved by using an FX3 chip. By changing the firmware of the FX3, one can alter the function of the FX3. To program the FPGA via USB3, the FX3 must act as a connector from the host to the FPGA. This type of connection is known as an FPGA link. This method of connection is required to avoid programming the FPGA and FX3 dedicated memories and thus not having to use different programming methods and cables to program the board. It is considered that the FX3 is suitable to be used as an FPGA link since its previous version, the FX2, was also used as an FPGA link in a similar project.

Firmware was downloaded on the FX3 using libusb and fx3load files from a Linux terminal. Some testing firmware was verified to perform as intended. However, the connection firmware intended to make the FPGA link truly functional has not been successful so far. Yet, through the FX3 documentation, it can be noted that an FPGA link is possible. UrJTAG shell should be appropriate to successfully program the FPGA once the FX3 acts as an FPGA link. However, connectivity via UrJTAG to an FPGA was not completed; thus this FPGA link can be deemed as a viable method but not truly confirmed.

Introduction

A module containing an FPGA and an FX3 chip is used as a slave FIFO device to act as a channel for the data transferred between the host computer and the sensing boards. The FPGA and FX3 chips are programmed through their respective memories as shown in Fig 1. Previous versions of the module integrated an FX2 chip which was also used as an FPGA link. Hence, the FX2 was used to program the FPGA and also, as a channel for data transfer. The programming upgrade entails that the FX3's firmware acts as an FPGA link; thus providing the same functionality previously obtained through the FX2 device.

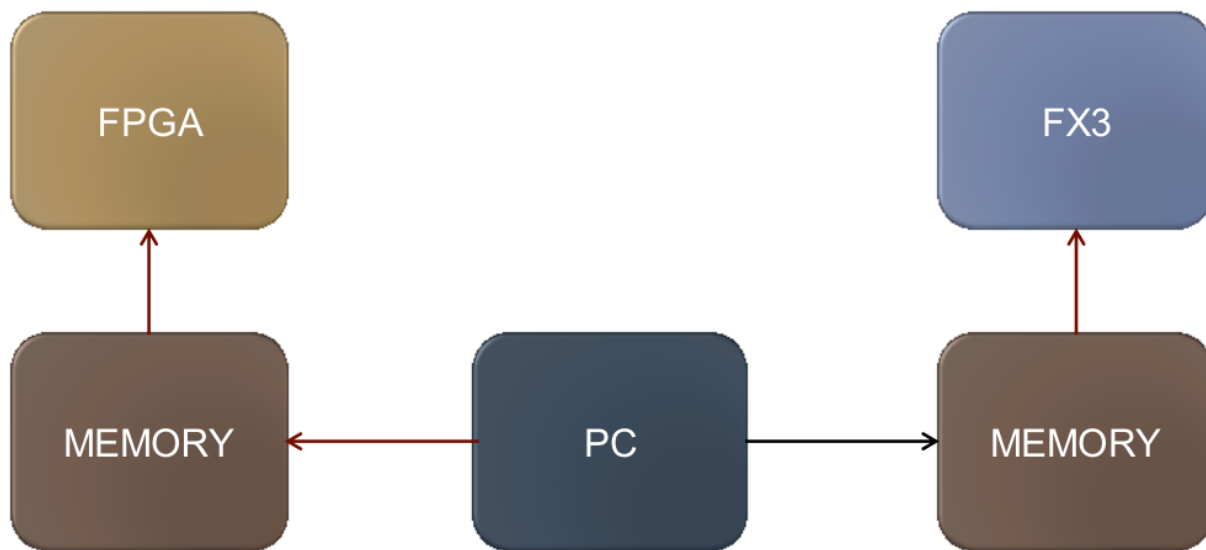


Figure 1 - Current Programming Configuration

Method

To use the FX3 as an FPGA link, the following steps are required:

- developing Firmware for FX3 using Cypress Tools
- downloading FX3 firmware using libusb libraries
- programming FPGA using UrJTAG through FX3

Cypress development tools provide the environment and examples to develop generic firmware. Making custom firmware requires the use of examples as well as making modifications according to the task's requirements. Cypress also provides the environment to download firmware to the FX3. However, it is necessary that the firmware is downloaded through other means such as terminal commands. Terminal commands can be used within Windows, Mac or Linux OS.

Windows terminal requires the installation of several libusb files as well as an appropriate Terminal SHELL which gives the user a terminal command environment. In a variety of Linux OS, libusb files are readily available and a plugin is required to download files on the FPGA. Ultimately, this project will be using a Linux OS. Therefore, the firmware download should be executed through open source software and terminal commands.

The Linux plugin required to download the image file of the firmware to the FX3 is the fx3load. When the FX3's firmware is set to make it function as an FPGA link, UrJTAG shell is used to perform JTAG commands with the FPGA. If JTAG commands are successful, programming the FPGA through the FX3 would then be determined possible.

The first applications tested were present within the Cypress development tools. Cypress tools were primarily used to download firmware to the FX3. Firmware was modified to accommodate for the required FX3 pins (shown in the table below) to be connected with the JTAG port of the FPGA. The pin name is the actual BGA pin name of the FX3. The I/O number is used within the firmware to set the required pin as input or output. The pins chosen to be used on the FX3 development board happen to be the ones used for SPI communication. Therefore, SPI will be overridden and its pins will be used as an FPGA link.

FX3 Pin name	I/O number	Pins override names	Required pins for JTAG	SPI Header J34 pins used
D4	GPIO[53]	SPI_SCK	TCK	3
C1	GPIO[54]	SPI_SSN	TMS	5
C2	GPIO[55]	SPI_MISO	TDO	6
D5	GPIO[56]	SPI_MOSI	TDI	4

The configuration of the FPGA link is shown in Figure 2 while the table above describes connections of the system.

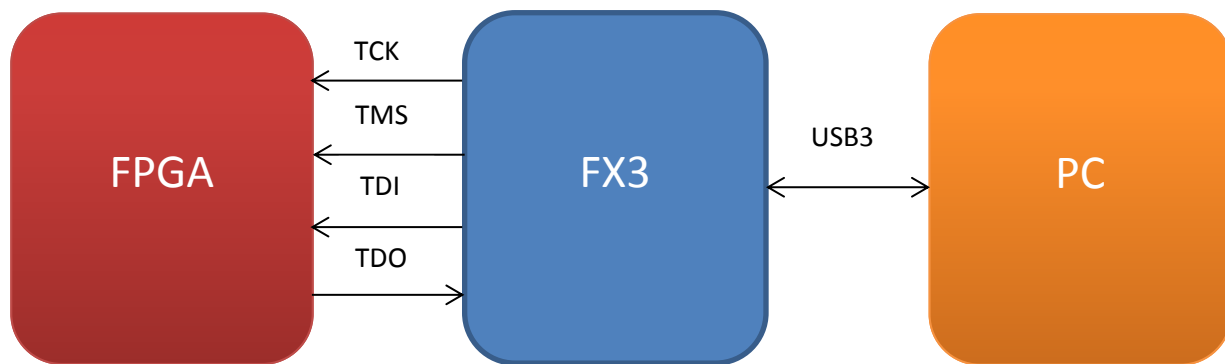


Figure 2 - Upgraded system block diagram

The image file was created using an Eclipse IDE on Windows 7 and downloaded on the FX3 through Scientific Linux 6. Some versions of scientific Linux come with libusb files readily installed while others require libusb library installations. The fx3load plugin is required to download firmware to the FX3 development board. Fx3load is not embedded in Linux OS. Hence, it needs to be installed. Fx3load is a tool that allows firmware in .hex .ihx and .img files to be downloaded into the FX3. Libusb commands are used to read the bus and device numbers as shown in Figure 3.

```
[Stef@pacific ~]$ lsusb  
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub  
Bus 001 Device 002: ID 03fd:0008 Xilinx, Inc.  
Bus 001 Device 003: ID 04b4:00f3 Cypress Semiconductor Corp.
```

Figure 3 - lsusb command to view USB connected devices

After acknowledging the bus and device numbers, the image file is then downloaded using the following command:

```
Fxload -I application.img -D /proc/bus/usb/001/003 -t fx3
```

From Figure 3, it should be noted that the FX3 chip is the one acknowledged as Cypress Semiconductor Corp, hence Bus 001 and Device 003, which may be observed in the command above. Other commands of the fx3load are found within its documentation. The firmware download was confirmed to be correct when the outputs from the testing firmware were verified on the oscilloscope.

The last step required to verify the FPGA link is to use UrJTAG to transmit commands from the PC to the FPGA. UrJTAG software is available in Windows and Linux. Installation is done by following the documentation of the UrJTAG website. Within this project, UrJTAG was installed and used within both operating systems. UrJTAG requires the user to set the Cable driver for the required FPGA and soon after detect the FPGA connected. If the driver is correct and the cable is declared correctly, one could send commands or program the FPGA.

Results

Programming the FPGA board through the FX3 was not achieved but from the documentation of the FX3 and UrJTAG, it is understood to be possible. Windows 7 and Scientific Linux 6 (as virtual machine) were used for this project even though this project could be managed in a simpler way by using only Scientific Linux 6 or a more updated Linux OS. The unsuccessful part of the project was in not having all the appropriate means (cable drivers and commands) to test the code used to connect the control and bulk endpoints to the required output pins of the FX3. It was deemed to be unsuccessful because the JTAG communication was not conclusive and since UrJTAG would determine the functionality of firmware, the firmware developed could not be tested sufficiently. Primarily, UrJTAG was used to communicate via a JTAG connection with the Xilinx Virtex 6. Lack of communication with the Xilinx Virtex 6 board could be attributed to the lack of drivers and/or proper commands. Even though the FPGA link was not fully realised, programming the FX3 using libusb, fx3load and a Linux terminal was successful. Configuring UrJTAG correctly would allow for the realisation of the FPGA link as described in the method.

Conclusion

The Linux operating systems used in the process were CENT OS, Scientific Linux 5 and 6. All the operating systems were installed as a virtual machine. Thus, some limitations from the virtual machine were encountered, specifically USB connectivity. The issue of USB connectivity was solved by reducing the capability of USB 3.0 to USB 2.0. Some problems were also encountered when installing UrJTAG in Scientific Linux 6 due to UrJTAG not recognising that libusb files were actually installed in the system. The same problems were not however encountered when UrJTAG was installed on Scientific Linux 5.

The method chosen to program the FPGA board in this project is determined to be efficient in terms of requirements. If implemented in a Linux environment, the only tools required would be a firmware, libusb files, fx3load files and UrJTAG shell. This approach would allow the user a high level of control over the data being sent to the FPGA.

Appendix

Below are snippets of the program to set GPIOs.

```
// Configure GPIO 56 as input with interrupt enabled for both edges
gpioConfig.outValue = CyTrue;
gpioConfig.inputEn = CyTrue;
gpioConfig.driveLowEn = CyFalse;
gpioConfig.driveHighEn = CyFalse;
gpioConfig.intrMode = CY_U3P_GPIO_INTR_BOTH_EDGE;
apiRetStatus = CyU3PGpioSetSimpleConfig(56, &gpioConfig);
if (apiRetStatus != CY_U3P_SUCCESS)
{
    /* Error handling */
    CyFxAppErrorHandler(apiRetStatus);
}

// Overriding GPIO 53 and setting it as an output with no interrupt
apiRetStatus = CyU3PDeviceGpioOverride (53, CyTrue);
if (apiRetStatus != 0)
{
    /* Error Handling */
    CyFxAppErrorHandler(apiRetStatus);
}

/* Configure GPIO 53 as output */
gpioConfig.outValue = CyFalse;
gpioConfig.driveLowEn = CyTrue;
gpioConfig.driveHighEn = CyTrue;
gpioConfig.inputEn = CyFalse;
gpioConfig.intrMode = CY_U3P_GPIO_NO_INTR;
apiRetStatus = CyU3PGpioSetSimpleConfig(53, &gpioConfig);
if (apiRetStatus != CY_U3P_SUCCESS)
{
    /* Error handling */
    CyFxAppErrorHandler(apiRetStatus);
}
```

Following available examples of Cypress EZ USB Suite and the Programming Manual, one could also find code which sets bulk and control endpoints. The bulk and control endpoints have not been documented due to not successfully using UrJTAG to verify the firmware's functionality. Toggling of outputs and other functions related to tasks are done through thread functions such as GpioOutputThread_Entry() found in the cyfxgpioapp.c of GpioApp example.