Jacek BORKO, Grzegorz DULNIK, Adam GRZELKA, Adam ŁUCZAK, Adam PASZKOWSKI
CHAIR OF MULTIMEDIA TELECOMMUNICATIONS AND MICROELECTRONICS, POZNAN UNIVERSITY OF TECHNOLOGY
3 Polanka St., 60 - 965 Poznan

A remote programming module of FPGA boards

Abstract

This paper presents a concept of implementation of a remote programmable FPGA module on selected hardware platforms. The possibilities of programmable matrix configurations and different ways of data transmission through an appropriate interface are described. The measurement results of operating times for the solution proposed by the authors are given.

Keywords: FPGA, remote programing.

1. Introduction

FPGA circuits are used in many different branches of industry, especially in prototype circuits where the constant modification of construction is needed.

Because of the wide variety of FPGA circuit applications, the place of a circuit activation is not always easily accessible by a human. This is the background to a problem strictly related to circuit reprogramming, since a standard programmer connection may be problematic. An exemplary situation is when a matrix that controls a set of cameras works at high attitude with a limited access to a circuit. The process of configuration of such circuits can be improved by the programmer, which does not require the direct-wired connection with a target system. In some cases, the target FPGA/uP device can be totally unreachable for programming e.g. satellite devices. It makes the problem even more complicated and some example solutions are described in [1][2]. The authors focused on the case where the target devices were connected to the Internet and were accessible through it. In this paper, a concept of a remote control programmer module of FPGA boards is presented. Figure 1 shows schematic representation of the example way the module works.

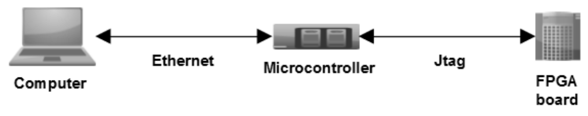

Fig. 1. Scheme of the FPGA module connection; devices and connection protocols between them are presented

Data transmission service aims at sending the configuration file from a computer to the programmer module. However after data transmission, the programming of the target circuit is carried out. A module connection to the network allows a developer to reprogram his/her FPGA board from every place that has an access to the Internet.

2. Hardware platform

The authors selected two significantly different hardware platforms to realize the programmer: Arduino Uno based on a microprocessor from the AVR family and Raspberry Pi with a microprocessor from ARM Company. In this way, the authors presented the capabilities of the module implemented in circuits with good technical parameters, as well as in circuits much cheaper with slightly fewer capabilities. The abovementioned hardware platforms helped the authors to verify and test how the programmer worked under different conditions.

The first circuit was selected because its software support by a powerful library was dedicated for the Arduino environment. For this platform a lot of additional module improvements are available, therefore it has gained great popularity among developers. This model contains e.g. a processor clocked at 16 MHz and a Flash memory of 32 KB, which imposes limitations to the size and complexity of the created program.

The second platform is Raspberry Pi model B+. It has been used for its very good technical parameters e.g. the ARM1176JZFS processor clocked at 700 MHz or the SDRAM memory of a capacity 512 MB. Undoubtedly, the advantage of this circuit is the adaptation to work with the Linux operating system, which gives great opportunities.

3. Data transmission

It is very important for a configuration file to reach the programmer undamaged, because it is required for correct target circuit programming. The selection of the proper data transmission protocol in which this application has to work depends mainly on the purpose of the program and the requirements it must satisfy. Due to the data transmission requirements, the TCP protocol was selected. It ensures lossless data flow thanks to the retransmission of damaged and lost packets or transmission in the connection mode.

Realization of data transmission service aims at sending a configurable file from the computer to the programmer circuit. From the point of view of a developer, sending the configurable file is carried out through window application. The program was created in the Qt environment.

The connection was realized in the client-server mode. The client service module was realized in the application which initializes the connection to a server. Therefore, the circuit programmer acts as a server, that waits for the connection on a specific port. In Figure 2, a two-sided graphical interface is presented. The left side is the information block which contains a text editor and a progress bar. The right side is a functional block with button activations of event procedure services.

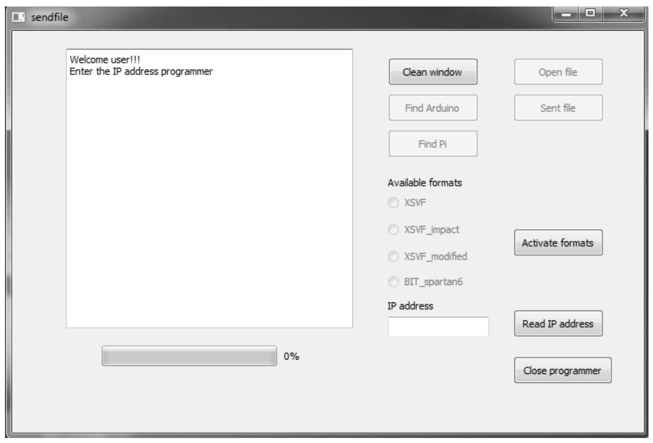
Welcome user!!!
Enter the IP address programmer

Fig. 2. Graphical interface available for the user

4. Configuration of FPGA boards

The possibility of configuring the FPGA matrix depends on the specific board. For example, the Spartan 6 matrix is configured with a bitstream. It means that the file is compiled from HDL language to a binary form and then is placed in the FPGA configuration memory. A file can be shifted from a FLASH memory, a microprocessor, a microcontroller and even from a computer. All those options use special configuration pins which allow configuring devices in a couple of modes: Master Serial, Master Select, MAP/BPI, Slave Serial, Slave Select MAP/BPI and JTAG.

5. Formats supported by the configuration file

In order to check the possibilities of matrix configuration, two formats of files were selected. This section presents the analysis of both formats.

The first one, which constitutes the base for chip configuration is a file with *bit* extension [3]. When this file is used all instruction of the JTAG interface of the specified chip must be known to the user. This file consists of a header and a bitstream, i.e. the data needed to configure the FPGA matrix. This data is needed to be shifted to a chip in the right moment on a TDI port in the JTAG interface. For a single matrix in a JTAG chain, configuration that uses only a bit file is really schematic, because instructions concerning TAP automat do not change. Bitstream is the only thing that changes. This is really comfortable because there is no need to complicate the implementation in order to modify the work of an integrated circuit. However, for various families of FPGA boards, the values in specified instructions can change, and also some new instructions and the arguments may appear. In the case of the proposed universal programmer, there is the need to create an additional database of commands to each family and each board. In the case of reconfiguration of many boards at the same time, additional commands for extended FPGAs control have to be placed in specified places of the programming instruction stream. This shows that such a variant of programmer realization is good only in the case when the developer would like to configure one board with one FPGA device.

The second format of the file is xsvf. It is a compact, binary version of svf (includes text in ASCII) which contains not only the data needed for FPGA board configuration but also instructions for the JTAG interface [4]. If the programmer is implemented correctly, every command for xsvf format is supported . This is an advantage, as Xilinx Company assures to reconfigure any of their FPGA boards. On the other hand, the disadvantage of this format is that every time when the matrix needs to be reconfigured, a new xsvf file has to be generated from the bit file and iMPACT program is needed. This operation takes time and tests are delayed.

The first solution to this problem is to use iMPACT program in the batch mode. This option allows implementing a script which has only a bit file. The script turns on the iMPACT in the background, generates the xsvf file and sends it to the microcontroller.

The second solution is more complicated and relies on changing the data needed to configure the xsvf file from the bit file. The effect of this process is that iMPACT is turned on just once to get the instructions for JTAG interfaces to configure the devices.

6. Implementation

The authors choose the Arduino system with an ATMega microcontroller as one of the testing platforms. The Arduino platform is a very cheap solution and is supported by many programming libraries. What is most important, there is a lot of libraries that support external interfaces and devices. Moreover, the software for the Arduino is developed and verified by a large community of Arduino users. Thanks to that the libraries are ease to use and are of high reliability.

The proposed programmer was composed of an Arduino board called Arduino UNO and together with a shield with the Ethernet module (WizNet device) (Fig.3). The JTAG interface is connected directly to ATMega microcontroller pins. The WizzNet shield comes with a card reader for a Micro-SD card so the data received by the Arduino is stored on a micro-SD card.

The SD card slot and the WizNet share the same SPI bus, therefore, it is not possible to read data from Ethernet or write them to SD card at the same time.

It means that a buffer for data storage had to be implemented. Due to a very limited memory in the ATmega microcontroller, the buffer is small, only for 128 bytes, but it is enough to maintain smooth data transfer.

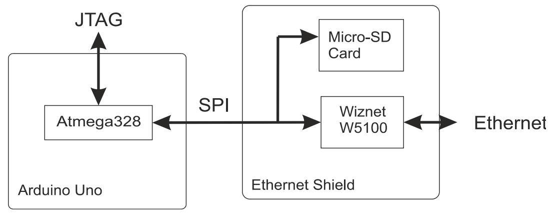

Fig. 3. Scheme of Arduino UNO

The next significant limitation is the size of the Flash memory in the Arduino UNO system. The program memory is limited to only 32 KBytes. For large projects it can be very troublesome. In this project, for full support of the xsvf programming file, the Ethernet communication (through the WizzNet module) and reading/writing from/to the SD card, the total size of the program code is about 28 KB.

A much more powerful system is offered by the Raspberry Pi platform. The more expensive Raspberry Pi platform offers an embedded Ethernet interface, a large RAM memory and ready to use Linux system. It can handle much larger programs and carry out more complicated tasks.

For the Raspberry Pi code, the size is smaller because there are libraries for Ethernet communication and SD card support which are already built into Linux. So the program contains only the control code and the code for managing the xsvf files.

7. Verification of the remote programmer

After the analysis of the programmer on two different devices measured, it was found that the Raspberry Pi provided better performance, as it was expected. Table 1 shows the time results for each operation. The procedure of data transmission in the network and the program process of FPGA boards took much more time using the Arduino Uno, than in the case of the Raspberry Pi.

Tab. 1. The results of programming and transmission times for different platforms

	Raspberry Pi	Arduino Uno
Transmission, s	1.3	120.2
Programming, s	3.6	40

For devices using over a dozen FPGA matrixes which require reconfiguration in a short time, the programmer implemented on the Raspberry Pi is doubtlessly better. However, for solutions in which a short time interval for reconfiguration is not needed, the cheaper Arduino Uno board can be used.

8. Conclusions

The paper presents the results of implementation of the FPGA programmer. Two programming platforms based on ***Raspberry Pi*** and ***Arduino Uno*** were tested.

In order to be compatible with the ISE Design Suite environment from XILINX, the proposed programming platform supports ***bit*** and ***xsvf*** files.

The proposed solution enables remote configuration of FPGA devices located in places which are hard to reach, which can greatly facilitate the work of developers.

The presented work has been funded by the Polish Ministry of Science and Higher Education within the status activity task "Signal processing and antenna optimization for acquisition, processing, analysis and presentation in 3D television systems" in 2015.

9. References

[1] Sosnowski J., Iwiński M.: Remote software reprogramming in embedded systems. Pomiary Automatyka Kontrola, PAK, vol. 59, nr 8, 2013, pp. 769-771.
[2] Iwinski M., Graczyk R., Sosnowski J.: Dependability issues in the PPLD-PSU subsystem for the BRITE-PL Hevelius microsatellite. PAK, vol. 60, no. 7, 2014.
[3] Xilinx: Spartan-6 FPGA Configuration User Guide. October 2014.
[4] Bridgford Brendan and Cammon Justin: SVF and XSVF File Formats for Xilinx Devices. August 2009.

Received: 16.04.2015 *Paper reviewed* *Accepted: 02.06.2015*

Jacek BORKO, eng.

He is a MSc student at the Chair of Multimedia Telecommunications and Microelectronics. His field of interests are microcontrollers, FPGA boards and high level programming.

e-mail: jacekborko@gmail.com

Grzegorz DULNIK, eng.

He is a MSc student at the Chair of Multimedia Telecommunications and Microelectronics. The main area of his interests are FPGA devices and microcontroller programming.

e-mail: dulnik.g@wp.pl

Adam GRZELKA, MSc

Received MSc degree from Poznan University of Technology in 2014. He is a PhD student at the Chair of Multimedia Telecommunications and Microelectronics. The main area of his professional activities are image processing, FTV (Free Viewpoint Television) and FPGA – especially implementation of compression algorithms and communication interfaces.

e-mail: agrzelka@multimedia.edu.pl

Adam ŁUCZAK, PhD

Received his MSc and PhD degrees from Poznan University of Technology in 1997 and 2001, respectively. In 1997 he joined the image processing team at Poznan University of Technology. Member of of Polish Society Theoretical and Applied Electrical Engineering (PTETiS). His research activities include video coders control, MPEG-4/H.264 systems and hardware implementations of digital signal processing algorithms. Currently he is involved in some project on video coding and video delivery.

e-mail: aluczak@multimedia.edu.pl

Adam PASZKOWSKI, eng.

He is a MSc student at the Chair of Multimedia Telecommunications and Microelectronics. The main area of his interests are FPGA boards and microcontrollers programming.

e-mail: adam9205@wp.pl