REMOTE FPGA LABORATORY FOR TESTING VHDL IMPLEMENTATIONS OF DIGITAL FIR FILTERS

Martin Petrvalský, Oto Petura, Miloš Drutarovský
Department of Electronics and Multimedia Communications, Faculty of Electrical Engineering and Informatics, Technical University of Kosice, Letna 9, 042 00 Kosice, E-mail: martin.petrvalsky@tuke.sk, oto.petura@student.tuke.sk, milos.drutarovsky@tuke.sk

ABSTRACT

In this paper, we develop a new remotely testable setup based on Altera Cyclone III field-programmable gate array. The setup is part of the European project EDIVIᗡƎ *in which our department participates. The setup is prepared for an educational purposes in the field of the reconfigurable devices. The setup uses Nios II soft core processor with reconfigurable coprocessor for the digital filtration. The design of the coprocessor is provided in VHDL which can by modified by the end user. The Nios processor programmed in C language ensures an interface between the coprocessor and* EDIVIᗡƎ *project servers. The complete setup uses a Nios II Cyclone III embedded evaluation kit board with additional debug and monitoring interfaces for* EDIVIᗡƎ *project infrastructure. Besides the VHDL design and C implementation, we created a web interface for the remote testing and debugging of the end user designs.*

Keywords: *Remote setup, FPGA, Nios soft processor, FIR filters,* EDIVIᗡƎ *project, web interface*

1. INTRODUCTION

Education in the field of reconfigurable systems requires various elements. Lectors need software (development environment, simulation tools) and hardware equipment such as evaluation boards for each student. These requirements are often hard to achieve due to price, time and location of the lectures. One of the method to provide all the tools to students is usage of a remote setups. Students can have access to software and hardware at any time and from every place with the Internet connection.

We aim to provide students online access to toolchain and evaluation board. User of the remote setup provides design codes. After synthesis, they can test the design on the evaluation board using web interface with online video stream, control buttons and indicators. The remote system is part of EDIVIᗡƎ project [1] in which we participate.

We develop a novel system in the field of a university education focused on hardware design in a combination with a soft processor. Previously published and used educational remote setups [2–4] are mainly focused only on the hardware design. We added a functionality of the soft processor in the remote setup for educational purposes which shows students a possibility in interoperability between software and hardware design.

The paper is organized as follows. Brief information about EDIVIᗡƎ project is provided in section 2. The architecture of the remote setups are discussed in section 3. We give information about Nios II [5] setup with Finite Impulse Response (FIR) filter coprocessor and its web interface in section 4. Section 5 contains experimental results of our setup. We conclude the paper in section 6

2. EDIVIᗡƎ PROJECT

The project's main goal is to create and maintain a network of remotely accessible Field Programmable Gate Array (FPGA) setups shown in Fig. 1. Setups grant students who do not have direct access to the hardware an opportunity to test their designs on a real hardware using remote access. The access is provided using an unified web interface which enables students to check the device status and control the device using a provided controls.

Catholic University Leuven (KHLim), University of Oslo (UiO), Technical University of Kosice (TUKE) and University College Bonn-Rhein-Sieg (H-BRS) participate in the project. Each university provides four FPGA based setups. Two simple setups with easy to understand principles and two advanced setups with more complex solutions for exercises. Each setup is provided with exercises (usually 4-5 exercises) which are sequenced from the simplest to the most complex. Our simple setups teach students about Finite State Machines (FSMs) and linear feedback shift registers. Topics of the two advanced setups are Nios II with True Random Number Generator (TRNG) coprocessor [6] and Nios II with FIR filtration coprocessor which is the topic of this paper.

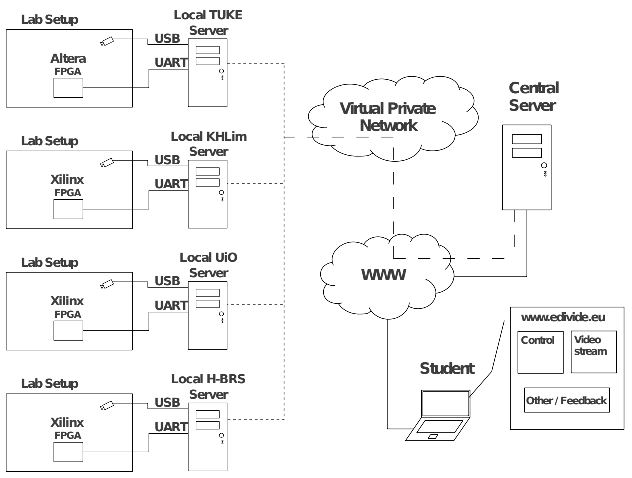

Fig. 1 : Architecture of distributed EDIVIᗡƎ laboratory

3. REMOTE SETUPS ARCHITECTURE

Each university is responsible for the local server setups, available FPGA hardware and development tools for the selected hardware. At our university we provide support for Altera FPGA by using Altera Quartus development tools [7]. Other partners in the project are using Xilinx FPGAs and development tools. We use Altera Nios II Cyclone III evaluation kit [8] as a common hardware platform for all EDIVIDE setups developed by our university. The Altera Nios II Kit includes full featured FPGA starter board and LCD multimedia card as shown in Fig. 2.

We use Altera Cyclone III EP3C25F324 FPGA [9] available on the starter board in all developed TUKE setups. The Altera EP3C25 contains 24,624 Logic Elements (LEs), 608,256 RAM bits, 66 18×18 bit embedded hardware multipliers and 4 Phase Locked Loops (PLLs) and all these resources are available for our designs. Our communication with local server uses defined Universal Asynchronous Receiver/Transmitter (UART) channels shown in Fig. 3 These channels were defined as basic unified communication channels for all projects developed by EDIVIDE partners.

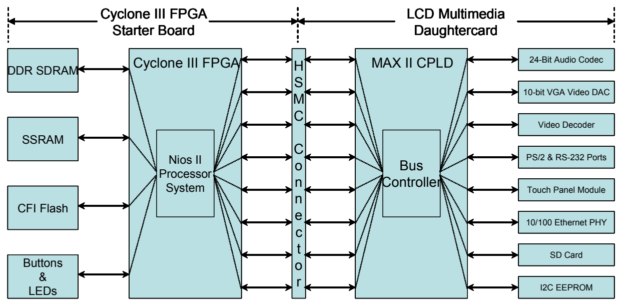

Fig. 2 : Structure of Altera Nios II Cyclone III evaluation kit and its embedded components

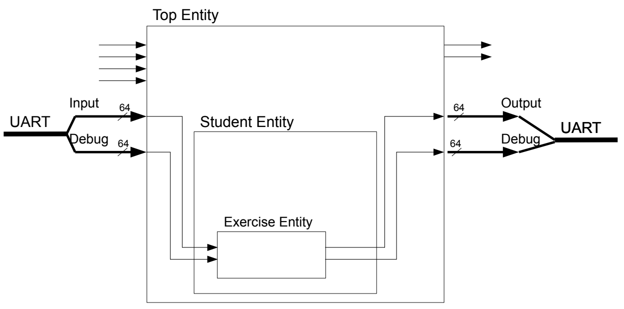

Fig. 3 : Basic Input/Output and Debug interfaces defined as basic communication channels for all EDIVIDE projects

As standard communication and debug channels we use 64-bit data and control packets in both directions. For TUKE designs we use additional available channels: 800×480 LCD screen and audio codec monitored by USB camera with microphone. We implemented interface to these channels as custom logic implemented in VHSIC Hardware Description Language (VHDL) [10] and controlled by a set of FSMs. A block diagram of available interfaces and additional hardware resources (pushbuttons, LEDs, oscillator, ...) is shown in Fig. 4.

Clear separation of these interfaces from our setups allows us to develop students' setups independently from hardware interfaces and/or optimize hardware interfaces (e.g. increase drivers resolution) in the future. In the next section we describe our second advanced setup combination of Nios II soft core with our implementation of FIR filter coprocesor.

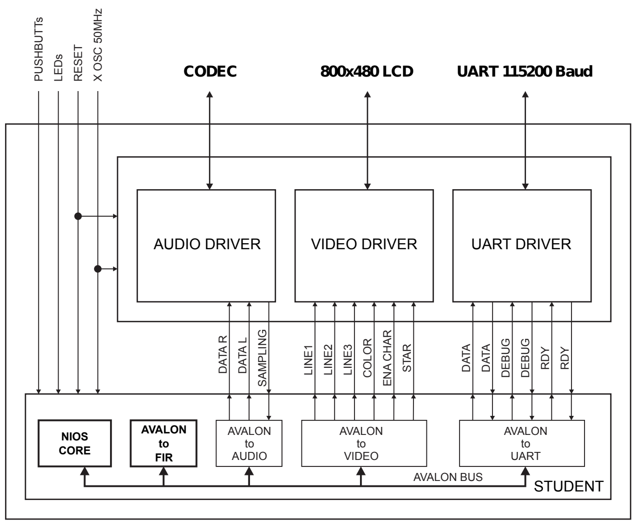

Fig. 4 : Interfaces and hardware components available for developed TUKE setups

4. ESSENTIAL PARTS OF THE NIOS II SETUP WITH FIR FILTRATION COPROCESSOR

The setup is designed to filter a real-time audio as well as digital signal samples received from a user via the Internet. Firstly, users must provide their design in VHDL code. After the synthesis takes place and the configuration is loaded to the FPGA board, the setup can be remotely tested. When filtering the real-time audio the user can set coefficients of the filter on the fly. When filtering digital signal samples the user can set coefficients, initial state of a delay line and send data samples to fulfill (1). The setup will then return filtered data which will be sent to the user.

$$\begin{aligned} y(n) &= h_0 x(n) + h_1 x(n-1) + \cdots + h_N x(n-N) \\ &= \sum_{i=0}^{N} h_i x(n-i) \end{aligned} \tag{1}$$

where $y(n)$ are output samples, $x(n)$ are input samples, h_i are coefficients of the FIR filter and N is order of the filter.

We prepared 4 exercises for students using our setup.

Demo exercise is prepared as a fully functional filter implementation. Students should get familiar with an FIR filter design [11] on an FPGA.

Symmetric FIR filter implementation. Student's task here is to implement their own implementation of a symmetric FIR filter.

Anti-symmetric filter implementation. Similarly to the second exercise the task here is to implement an anti-symmetric FIR filter.

N-band audio equalizer. The task here is to implement an N-band audio equalizer. It's goal is to point out one of the main uses of the FIR filters and performance of FPGA circuits.

In all exercises, students can upload a custom set of coefficients to the filter and use all capabilities of the setup. This provides students with a set of function to test their hardware implementation.

4.1. Nios II soft core processor

Nios II soft core processor is an embedded 32-bit processor architecture developed by Altera for use in FPGA circuits. The processor is responsible for handling communication between the server and the FPGA board. It also routes received data to appropriate peripherals according to the selected mode. There are several modes of operation:

Audio loopback mode: Nios receives data from an audio codec input and routes it back to the audio codec output not changing it in the process.

Audio filter mode: receiving data from the codec and sending back filtered data to the codec output.

Raw data filter mode: Nios receives all data from a serial line and sends it back also to the serial line. In this mode it is possible to set the initial state of the FIR filter delay line.

The Nios processor is also connected to an off-chip memory which is situated on the evaluation board besides FPGA (Fig. 5). The memory provides enough space for input and output data buffers for 96,000 signal samples. This amount is enough for 2 seconds of audio samples with a 48 kHz sampling rate.

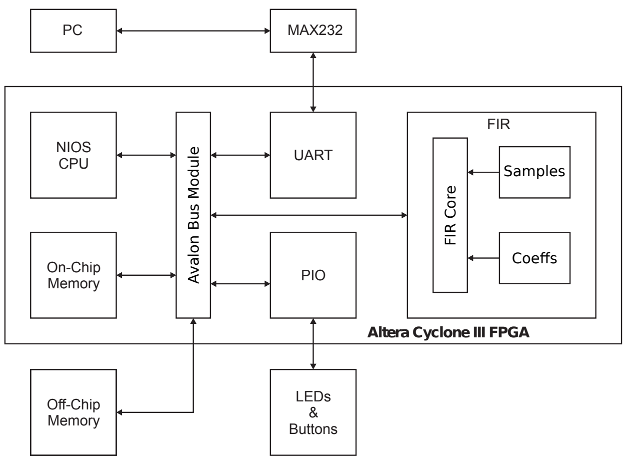

Fig. 5 : Block diagram of EDIVIDE setup with Nios II soft core and custom FIR filter coprocesor in Altera FPGA

4.2. FPGA based FIR filtering coprocessor

The setup provides student with an interface to the Nios II soft core processor as shown in the Fig. 6. Using this interface the student should implement a digital FIR filter. The interface provides 16-bit data input and output buses and 16-bit bus for coefficients input. It also provides the student with a FPGA embedded RAM blocks [12] for coefficients and data samples storage. Each of embedded RAM blocks is capable to store $1,024$ 16-bit values. This provides an interface suitable for filters with $1,024$ FIR coefficients.

The Nios processor along with the FIR filter are clocked by a 50 MHz clock. The audio sampling frequency of the audio codec is 48 kHz which allows the maximum of $1,041$ cycles for filtering and overhead during the filtering process as show in (2).

$$N_{maxCycles} = \frac{f_{clock}}{f_s} = \frac{50 \times 10^6}{48 \times 10^3} = 1,041.6 \text{ cycles} \qquad (2)$$

Our demo exercise implements a sequential FIR filter using the interface. The filter uses 16-bit fixed point arithmetics in 1.15 fractional format. The filter is controlled by a FSM, shown in Fig. 7, governing all operations. It uses RAM memory blocks for coefficients and data storage as depicted in Fig. 6. The demo exercise implements a general FIR filter based on (1) for the N^{th} order FIR filter.

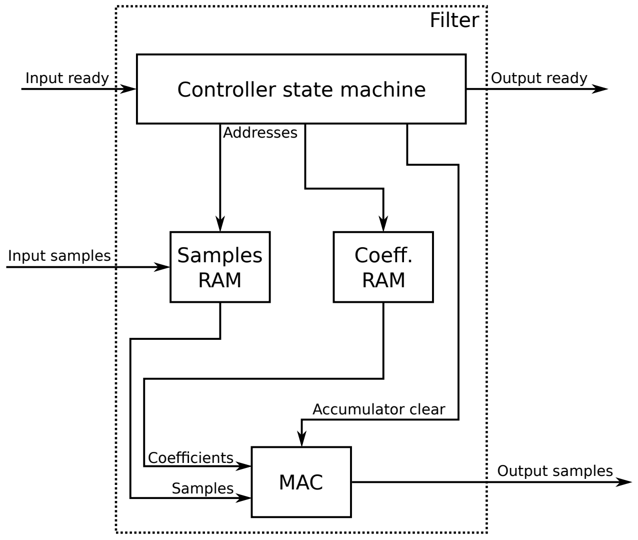

Fig. 6 : Block diagram of the digital FIR filter coprocessor

Embedded RAM blocks used for coefficients and data storage can store $1,024$ values each. That provides enough storage for implementing a $1,023^{rd}$ order FIR filter when the implementation directly matches (1).

The filtering process is controlled by the FSM as shown in Fig. 6. Fig. 7 shows internal states and possible transitions between them.

The reset signal is asynchronous and immediately puts the filter to a RESET state. In this state all internal registers are reinitialized with zeros. FSM then goes to RAM_CLR_WRITE state, which writes zero value to the sample memory. In RAM_CLR_INC state the memory address counter is incremented to erase next address. After whole memory is erased FSM goes into IDLE state. In IDLE state it waits for the input sample ready signal. After receiving the signal it goes into the WRITE_SAMPLE state. In the WRITE_SAMPLE state, the FSM writes the input sample into the samples memory. After the sample is written, the filtering process can start. The initialization phase of the filter is done during the START_FILTER state. During this phase, the FSM waits for the memory to finish writing of the sample. In the FILTER state, the filtering is performed. FSM sweeps through both memories and multiplies samples with appropriate coefficients. After it reaches the end of the memories, it changes the state to FINISH_FILTER. The FINISH_FILTER state compensates for the delay of the output registers of the memories. Similarly the FINISH_MAC state compensates for the delay of the MAC unit registers. After FINISH_MAC state, all samples are processed and the accumulator holds the output value. The FSM goes into the FINISH state, when it puts the 16-bit output value from the accumulator to the output register and triggers the output sample enable signal.

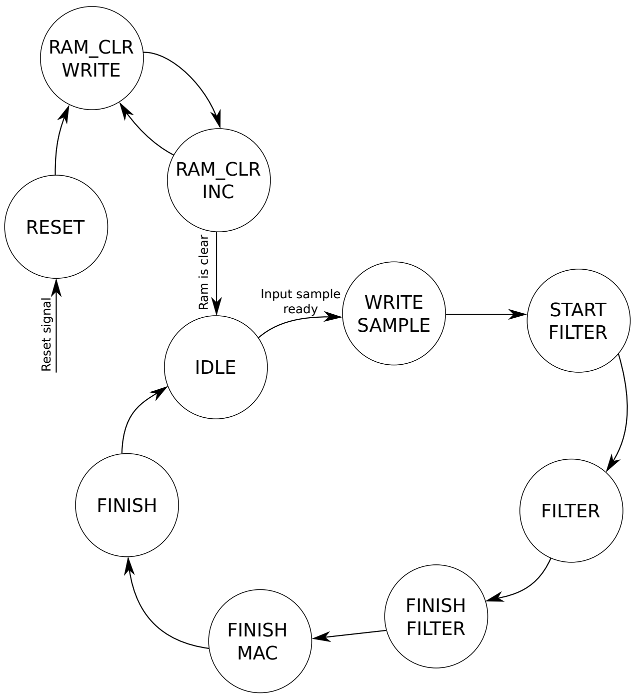

Fig. 7 : State diagram of the FIR filter's FSM

It is a known fact that linear-phase FIR filters posses a symmetry in their impulse response. There are two types of this symmetry. The first one is symmetric impulse response (3) and the second is anti-symmetric impulse response (4) [13].

$$h_i = h_{N-1-i} \tag{3}$$

$$h_i = -h_{N-1-i} \tag{4}$$

Two of student exercises are focused on symmetric and anti-symmetric filter implementations. Exploiting a symmetry of the impulse response may reduce memory requirements of the whole design because only half of coefficients need to be stored. This means that it is theoretically possible to implement a filter with $2,048$ coefficients with our setup using the embedded RAM blocks with $1,024$ elements. The time constraints are taken into consideration because of the maximal number of clock cycles available for the filtering is $1,041$ as shown in (2).

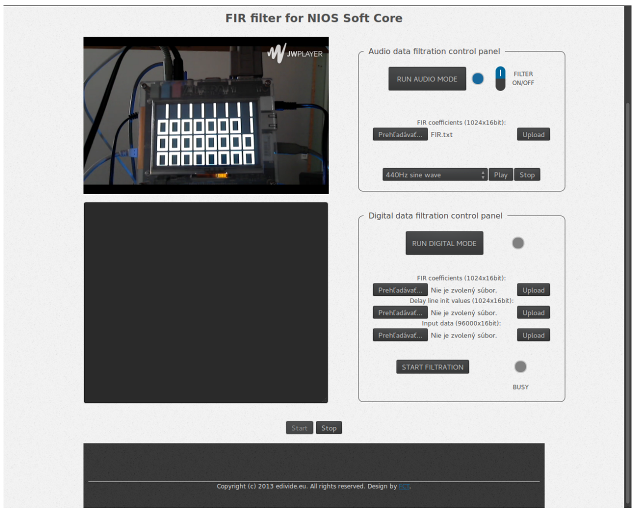
FIR filter for NIOS Soft Core
JWPLAYER
Audio data filtration control panel
RUN AUDIO MODE
FILTER ON/OFF
FIR coefficients (1024x16bit):
Prehľadávať...
FIR.txt
Upload
440Hz sine wave
Play
Stop
Digital data filtration control panel
RUN DIGITAL MODE
FIR coefficients (1024x16bit):
Prehľadávať...
Nie je zvolený súbor.
Upload
Delay line init values (1024x16bit):
Prehľadávať...
Nie je zvolený súbor.
Upload
Input data (96000x16bit):
Prehľadávať...
Nie je zvolený súbor.
Upload
START FILTRATION
BUSY
Start
Stop
Copyright (c) 2013 edivide.eu. All rights reserved. Design by FCT.

Fig. 8 : Printscreen of the web interface used with the second advanced setup.

4.3. Web interface for the FIR filter design testing

Web interface is used by students to control the design they developed. We developed two main parts of the interface - back-end and front-end. Printscreen of the front-end is in Fig. 8. In the front-end, video stream is in the upper left corner. It allows students to visually control the board. The text output is located in the lower left corner. It is used to inform the user about current processes in the design. Audio data filtration panel takes place in the upper right corner. It consists of activation button, signaling LED, switch for enabling FIR filtration, part for FIR coefficients upload and audio player with various sound samples. Raw digital data filtration panel takes place in the lower right corner. It consists of activation button, part for upload (FIR coefficients, delay line and input data), button to start the filtration and LED signaling busy state. Start and Stop buttons on the bottom are only for debug purposes. They are not available in the final version of the interface.

Second part of the web interface was the back-end. Part of the back-end was provided by EDIVIDE project and setup specific features was developed as a part of this work. The setup specific back-end consists of PHP script, CSS and JavaScript files. We created the functions which are uploading and processing the data files from the student. The functions forms packages for FPGA controlling and it sends data packages to the design. They also receive data from the board, process it and provide the filtered data to the end user.

5. EXPERIMENTAL RESULTS

5.1. FPGA resources used

Table 1 shows the FPGA resources used by all elements of the setup. We used 56% of the LEs and 76% of the embedded RAM bits. The rest of the resources is available for the students' design in submitted VHDL codes.

Table 1: FPGA resource usage

	LEs	Memory bits
FIR	268	32,768
Nios	cca 10,000	295,904
Total used	13,825	459,744
Total available	24,624	608,256
	PLLs	**DSP blocks**
FIR	0	2
Nios	1	0
Total used	2	2
Total available	2	132

5.2. Sound card characteristics

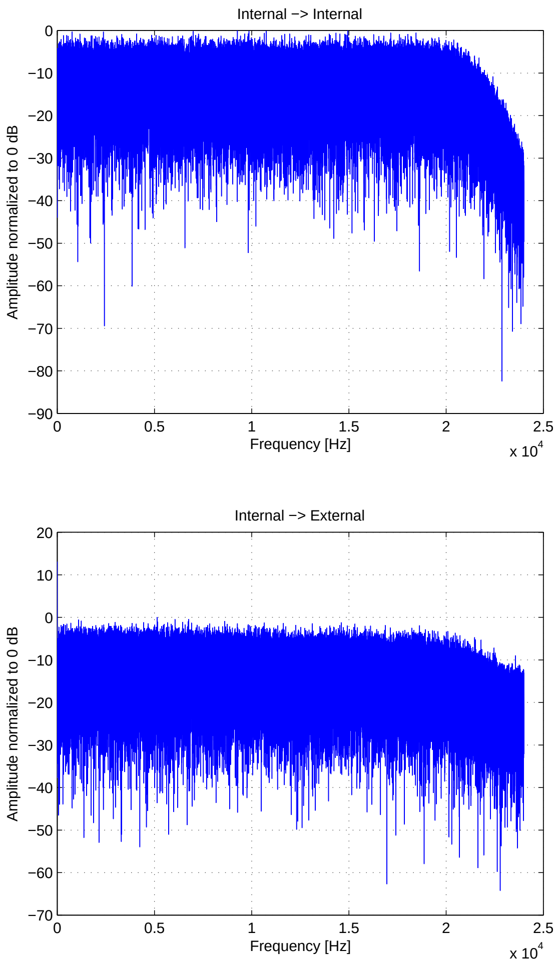

Fig. 9 : Characteristics of inputs of the sound cards when driven from the internal card

While everything was working correctly in simulations, we noticed a problem when testing on a real hardware. We noticed that the audio signal at the output of the FPGA board seems filtered even if the FIR filter is not working. After further investigation of this issue we found out that the problem is the external USB sound card used on the server. Since the server does not have an internal sound card, we need to use the external device to provide the sound support. We decided to confirm this issue by testing it on a computer with an internal sound card so we directly interconnected inputs and outputs of both cards. We then used the white noise to determine the frequency characteristics of both cards.

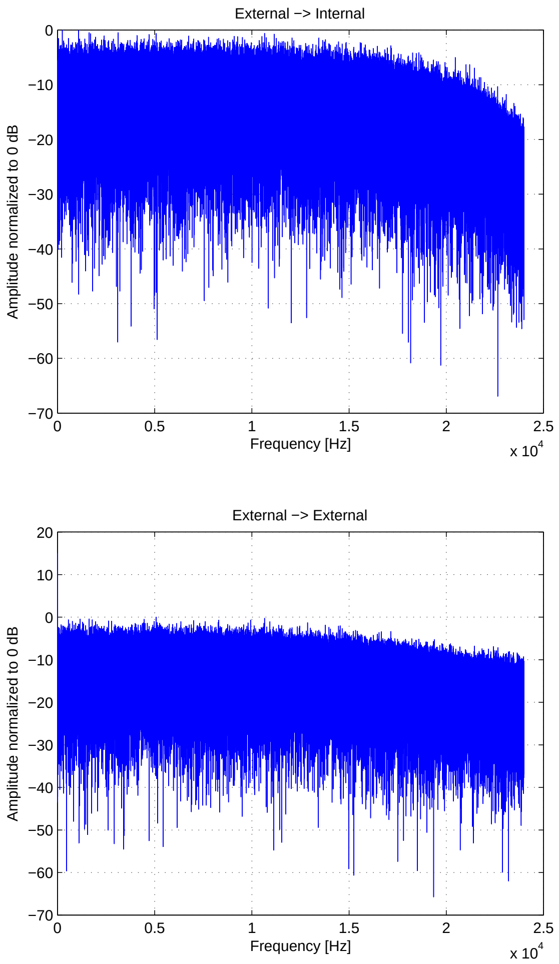

Fig. 10 : Characteristics of inputs of the sound cards when driven from the external card

The graphs in Fig. 9 and 10 show that the external sound card has an output filter with much lower cutoff frequency than the internal card. This affects every signal that is reproduced by the external sound card on the server. The input of the external card also does not have a filter which can suppress the DC component of the input signal.

Issues presented here are connected with the sound card used and can be suppressed only by choosing different hardware. This type of problem is hard to foresee, because characteristics of the input and output filters of the sound cards are not always public. Problems like this are also strongly dependent on the hardware and do not occur in simulations. That is one of the reasons, why testing with a real hardware is essential in the development process and in education. Students themselves will be aware of the fact that the simulation and hardware testing can behave differently.

6. CONCLUSION

In this paper we created the remote setup for Nios II processor using FIR filter coprocessor. We include this setup as second advanced setup for EDIVIᗡƎ project. We developed software for Nios II soft core processor, design of the FIR filter coprocessor, exercises for the students which will use this setup for education purposes and web interface for controlling and debugging the design. We provide FPGA resource usage for our setup and we measured sound card characteristics of the external sound card used on the local server. The setup allows students to design their own VHDL implementations of the FIR filter (e.g. a parallel FIR filtration). In the future work we will prepare Infinite Impulse Response (IIR) filtration coprocessor in order to allow students IIR filter testing.

ACKNOWLEDGEMENT

This work is done within European DIgital VIrtual DEsign (EDIVIᗡƎ) lab project #518565-LLP-1-2011-1-BE-ERASMUS-ESMO funded by LLP Erasmus.

REFERENCES

[1] EDIVIᗡƎ . (2015) EDIVIᗡƎ home webpage. [Online]. Available: http://www.edivide.eu

[2] MORGAN, F. – CAWLEY, S.: *Enhancing learning of digital systems using a remote FPGA lab*, Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC) **2011**, No. 1 (1) 1–8

[3] ELUCIDARE. (2015) Remote FPGA laboratory for online teaching and prototyping. [Online]. Available: http://www.elucidare.co.uk/assignments/project_fpga/abst_fpga.php

[4] VICILOGIC. (2015) Remote FPGA Labs and Online Learning of Digital Systems. [Online]. Available: http://vicilogic.com/fpl2014tutorial/

[5] ALTERA CORPORATION. (2015) Nios II processor webpage. [Online]. Available: http://www.altera.com/devices/processor/nios2/ni2-index.html

[6] DRUTAROVSKY, M. – VARCHOLA, M. – PETRVALSKY, M.:: *Remotely testable setup of soft CPU with cryptographic TRNG coprocessor extension embedded into Altera FPGA*, Radioelektronika (RADIOELEKTRONIKA), 23rd International Conference **2013**, No. 21 (2) 136–140

[7] ALTERA CORPORATION. (2015) Design software webpage. [Online]. Available: http://www.altera.com/products/software/sfw-index.jsp

[8] ALTERA CORPORATION. (2015) Nios II Embedded Evaluation Kit, Cyclone III Edition User Guide, Altera, July 2010. [Online]. Available: http://www.altera.com/products/devkits/altera/kit-cyc3-embedded.html

[9] ALTERA CORPORATION, Cyclone III Device Handbook, Altera, v 12.0, August 2012.

[10] IEEE Std 1076-2008 (Revision of IEEE Std 1076-2002), IEEE Standard VHDL Language Reference Manual, January 2009. [Online]. Available: http://web02.gonzaga.edu/faculty/talarico/polibari/documents/VHDLrefManual.pdf

[11] IOWA HILLS SOFTWARE. (2015) Digital and Analog Filters. [Online]. Available: http://iowahills.com/Index.html

[12] ALTERA CORPORATION. (2015) Embedded Memory User Guide. [Online]. Available: http://www.altera.com/literature/ug/ug_ram_rom.pdf

[13] RICE UNIVERSITY. (2015) Four Types of Linear-Phase FIR Filters. [Online]. Available: http://cnx.org/contents/32e866a8-cfb7-4bc1-acae-596823fd8260@3/Four_Types_of_Linear-Phase_FIR

Received March 11, 2015, accepted July 10, 2015

BIOGRAPHIES

Martin Petrvalský was born on the 16th of September, 1988 in Pribram. He received his Bachelor's degree in 2010 and Master's degree in 2012 in Technical University of Kosice. He graduated from Department of Electronics and Multimedia Communications from Faculty of Electrical Engineering and Informatics. He is a PhD. student in Department of Electronics and Multimedia Communications. His current research focuses on side channel attacks on embedded devices and its countermeasures.

Oto Petura was born on the 15th of February, 1991 in Michalovce. He received his Bachelor's degree in 2013 in Technical University of Kosice. He graduated from Department of Electronics and Multimedia Communications from Faculty of Electrical Engineering and Informatics. He is a Master's student in Department of Electronics and Multimedia Communications. His current research focuses on UWB sensor networks.

Miloš Drutarovský was born in Presov in Slovak Republic, in 1965. He received his Ing. (M.Sc.) degree and PhD. degree in Radioelectronics from the Faculty of Electrical Engineering, Technical University of Kosice, in 1988 and 1995, respectively. He defended his habilitation work - Digital Signal Processors in Digital Signal Processing in 2000. He is currently working as an associate professor at the Department of Electronics and Multimedia Communications of the Faculty of Electrical Engineering and Informatics, Technical University of Kosice. His current research focuses on embedded electronics, applied cryptography, algorithms and architectures for embedded cryptographic architectures, digital signal processing, digital signal processors, field programmable devices and soft microcontrollers embedded into FPGA circuits.