Ernest JAMRO[1], Witold CIOCH[2]

Akademia Górniczo-Hutnicza (AGH), Katedra Elektroniki (1), AGH, Katedra Mechaniki i Wibroakustyki (2)

Digital signal acquisition and processing in FPGAs

Abstract. *Rapid grow of FPGAs resources allows to include a whole system into a single FPGA chip. The system usually include soft-processors, limited on-chip memory, input-output interfaces, etc. The main advantage of FPGAs solution is, however, incorporation of dedicated hardware signal processing units such as signal filtering, FFT for which signal processing speed significantly surpasses DSP or general purpose processors. This paper presents an example of a whole system solution incorporating an FPGA and high-speed analog-digital converters.*

Streszczenie. *Gwałtowny wzrost dostępnych zasobów układów FPGA pozwolił na integrację całego systemu w jednym układzie FPGA. Taki system z reguły zawiera soft-procesor, niewielką pamięć, interface'y wejścia wyjścia. Podstawową zaletą systemu opartego na układach FPGA jest możliwość dołączenia dedykowanych jednostek przetwarzania danych, umożliwiających np. filtrowanie sygnałów, obliczanie FFT. Dedykowane jednostki wykonują wybrane obliczenia dużo szybciej niż procesory DSP lub ogólnego przeznaczenia. Niniejszy artykuł przedstawia przykład właśnie takiego systemu składającego się układu FPGA i szybkich przetworników analog-cyfra.* ***(Akwizycja i przetwarzanie sygnałów cyfrowych w układach FPGA)***

Keywords: Digital Signal Processing, FPGAs, high-speed signals acquisition and processing.
Słowa kluczowe: Cyfrowe przetwarzanie sygnałów, FPGA, szybka akwizycja i przetwarzanie sygnałów.

Introduction

Electronic implementations of signal acquisition and digital signal processing procedures are often disregarded in academic considerations. Nevertheless they often determines the final procedures and results. Therefore in this paper Field Programmable Gate Arrays (FPGAs) electronic solutions for high speed signal acquisition and processing is considered.

Usually two different electronic solutions are adopted:

1) General-purpose processors or digital signal processors (DSPs)
2) Field Programmable Gate Arrays (FPGAs)

General purpose processors (e.g. Pentium) or DSPs are very flexible, easy to be programmed and relatively cheap. Nevertheless a signal sampling frequency and signal processing speed is relatively small. Consequently, this solution is only suitable for relatively slow real-time diagnostic systems, for which signal sampling frequency is below 100kHz. The given threshold frequency differs for different algorithms complexities, number of channels, processor computation power and so on, therefore the 100kHz should be regarded only as a rough number.

For systems for which data sampling frequency is above roughly 100kHz the FPGA solution is the best one. FPGAs can be programmed by an end-user in similar way as microprocessors, nevertheless they employ different design cycle which causes that they can be designed only by electronic engineers. For FPGAs, input signals sampling frequency is up to roughly 500 MHz, similarly data processing speed is also very high.

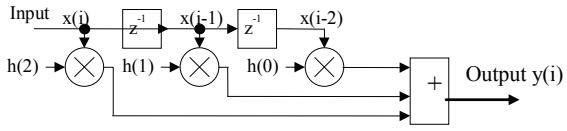

Fig. 1. Block diagram of a FIR filter

There are many signal-acquisition and digital signal processing devices based on FPGAs e.g. [1, 2], nevertheless they are either not dedicated signal processing, require PC connection or do not use hardware-software co-design approach. Therefore the main thought throughout this paper is to present a FPGA solution, its possibilities and design cycle, and finely to describe the developed FPGA-based system.

FPGAs and Digital Signal Processing

A Field Programmable Gate Arrays (FPGAs) e.g. [3] is roughly a semiconductor device containing programmable logic components such as AND, OR, XOR, NOT gates or more complex functions such as adders, multipliers, memories. This programmable logic elements are connected by programmable interconnects. In fact, in most cases, FPGAs do not contain standard gate logic – they incorporate relatively small memory cells (usually 16×1), denoted as Look-Up Tables (LUTs), which performs any logical function of up to 4 inputs. These logical functions can be grouped to form either Finite State Machines (FSM) suitable to control external components, e.g. Analog Digital Converters (ADC), memory. A (soft) processor e.g. MicroBlaze [4] is an example of a very complex FSM which can also be implemented in FPGAs. Besides these LUTs can be used to form arithmetic modules such as adders, multipliers, etc.

A fundamental signal processing procedure is filtering, therefore an example of implementation of a filter in FPGAs is given hereby. Finite Impulse Response (FIR) filters execute the following equation:

$$y(i) = \sum_{k=0}^{N-1} h(k) \cdot x(i-k) \tag{1}$$

where: *x(i), y(i)* – input and output samples, *i* – the index of a sample, *h(k)* – FIR filter coefficients, *N*- the number of filter taps.

For Digital Signal Processors (DSPs) every filter tap requires at least a single clock cycle [5]. For the number of filter taps *N= 100* and sampling frequency *f= 500 kS/s*, the required DSP clock frequency is well beyond 50 MHz. It should be noted that several independent channels are often processed and the DSP is also occupied by other activates (e.g. input data acquisition). Summing up, most DSPs are not capable of processing high speed input signals. Consequently an alternative solution - FPGAs is more and more often adopted for digital signal processing.

Block diagram of a FIR filter implemented in FPGAs is given in Fig. 1. Multiplier (*n×n-bit*) occupies less than n^2 4-input LUTs, furthermore for constant coefficient multipliers (employed e.g. when the filter coefficients are constant) the number of occupied resources can be significantly reduced [6]. For example, Xilinx XC3S1500 [3] contains roughly 30 000 LUTs. Consequently more than 100 16×16 bit multipliers working in parallel, each clocked by roughly

100 MHz can be implemented in this FPGA chip. Besides FPGAs incorporate dedicated multipliers, e.g. XC3S1500 contains 32 18×18-bit dedicated multipliers. Summing up, for selected operations, computation power of FPGAs is 10÷1000 (or even more) times larger than for DSPs.

Hereby only a simply example of the FIR filter is given, nevertheless the same hardware architecture can be employed for wavelet transforms and correlation calculations. A slight modification is only required for IIR filters. Similarly, efficient FPGA implementation of neural networks [7] or FFT [8] are available.

FPGAs and Software Solutions

FPGAs and microprocessors complement each other, i.e. usually the most computationally-intensive algorithms are data-driven algorithms (e.g. filtering, FFT), therefore they can be speed-up by FPGAs. On the others hand, program-driven algorithms, i.e. complex algorithms processed a limited number of times cannot be easily implemented in FPGAs. These algorithms are: initialization of hardware components, handling user interfaces (keyboard, LCD), displaying results, etc. Usually these algorithms are not computationally-intensive and therefore are well-suited for microprocessors. Partitioning an algorithm into software part (executed by microprocessors) and hardware part (executed by dedicated logics incorporated in FPGAs) is often denoted as hardware–software co-design [9].

FPGAs design cycle is relatively difficult and time-consuming. Therefore, a modular design approach is adopted. Modules, denoted as Intellectual Property (IP) cores, which functions are well-defined and tested, are supplied by different vendors. Connecting different modules to form a whole system is a fundamental problem which is tackled by Xilinx Embedded Development Kit (EDK) [10]. This software packet allows to graphically connect different modules and to integrate easily hardware and software part of the system.

In the proposed systems tens of different IP modules are employed. Some of them are supplied with the EDK. Nevertheless most of them are developed by the authors of this paper; e.g. ADC interface with IIR filter, FIR filter (wavelet transform) with adjustable decimation rate.

It should be noted that the core of the whole system is the soft-processor MicroBlaze [4] which controls all hardware modules. Consequently, the final procedures are designed in C/C++ language, which significantly simplifies the design cycle.

Programmable Unit for Diagnostics (PUD)

The proposed Programmable Unit for Diagnostics (PUD) is a device which core is FPGA XC3S1500 device [3]. The block diagram of the PUD is presented in Fig. 2. and it incorporates the following components:

- Four independent analog / digital modules on separate Printed Circuit Boards (PCB).
- Two independent SDRAM memory banks, 64MB each, employed to store acquired data from analogue / digital modules and other temporal data.
- Flash memory (4MB) to store FPGA configuration, MicroBlaze program and other non-volatile data e.g. filter coefficients, FFT coefficients.
- CPLD (Xilinx XC95144XL device) – module employed to configure FPGA and to control the PUD in power stand-by mode.
- LCD display employed to visualise the state of the device and results for acquired and processed data.
- Keyboard – allows user to control the PUD e.g. to start / stop data acquisition.
- PC computer communication by Ethernet, Parallel or Serial Ports.

Besides, the PUD incorporates some optional devices: hard disk driver (HDD), Compact Flash memory, VGA display, PC keyboard modules.

The PUD incorporates four independent analog (/ digital) boards. Each analog board incorporates either 10 MS/s 14-bit ADC or two channels 500kS/s 16-bit ADC or four channels 250kS/s. In total, the PUD can sample either four independent channels at 10MS/s each, 8 independent channels at 500 kS/s each or 16 independent channels at 250 kS/s.

Besides, the PUD incorporates a HDD, which allows for recoding acquired signals. In order to speed-up data transfer, a special HDD file format was designed to allow high speed sequential data writes to the HDD.

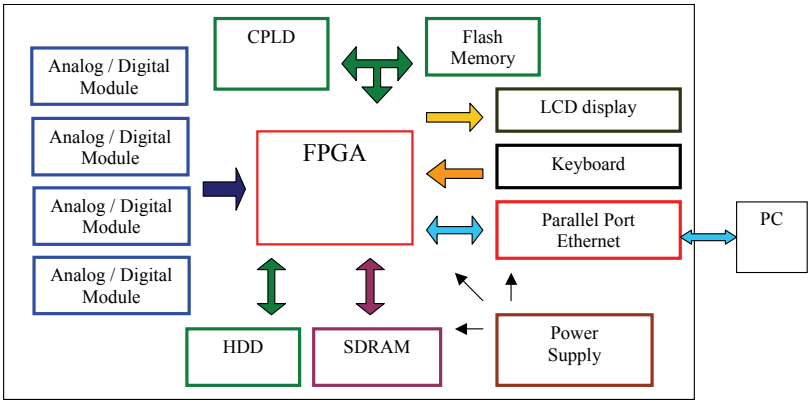

Fig. 2. Block diagram of the PUD

Basic Signal Processing

In this section a basic signal processing operations implemented in FPGAs are described. This operation ideally suite the FPGA. These operations are:

- calculation the signal range (MIN, MAX values)
- calculating the signal average value (AVG)
- calculating Real Mean Squares (RMS)
- the signal integration.

Calculating the signal range is required to adjust input signal amplifiers levels. Usually, the input signal to noise ratio is the lowest when the Analog/ Digit Converter (ADC) operates on its full range. Therefore input signal amplification level should be properly adjust based on the input signal range. Calculating signal range in software is a very basic operation - only 2 lines of C code – see Listing 1. Nevertheless this operation consumes significant amount of

processor time, as it must be calculated for every input signal.

Similar conclusions can be driven for calculating average and RMS signal values. Signal integration is required for ICP sensors for which vibration acceleration is measured, nevertheless vibration speed or position is the point of interests. It should be noted that calculation of the integration as stated in Listing 1 can be performed provided that the average signal value is equal 0.

Listing 1. An example of C language code for calculating MIN, MAX, AVG, RMS and integration.

```
// initialisation
int min= 0x7FFF, max= 0x8000, sum=0, sum2= 0; // for 16-bit data
// main calculation
for (int i=0; i<N; i++) // N- number of all samples
{ if (data_in > max) max= data_in;
  else if (data_in<min) min= data_in;
  sum+= data_in;
  sum2+= data_in * data_in;
  integral(i)= sum;
}
// final calculation
avg= sum/N; RMS= sqrt(sum2/N);
```

The hardware implementation of the above operation significantly speed-ups calculation process. It should be noted that similar code as in *C* can be written in VHDL (hardware description language). Besides this code consumes insignificant amount of FPGA resources. For example *n*-bit adders or comparator requires only *n*-LEs (logic elements ≈ LUT); XC3S1500 contains roughly 30 000 LEs. Another problem for processors is arithmetic operation bit-widths. For example for 16-bit input data and number of input samples greater than 2^{16}= 65536, the 32-bit operation is not sufficient. The solution for this problem is to substitute e.g. one 48-bit addition with two 32-bit additions. This however doubles the processor calculation time. For hardware solution the arithmetic width is defined by a parameter which can be freely selected.

Calculation speed of the hardware solution is far beyond the requirements, for example, all above operation can be carried out in parallel with speed greater 100MHz. Therefore to further reduce the hardware requirement input signals from different channels are time-multiplexed into a single arithmetic unit. Therefore only a single arithmetic unit is required for several different channels.

Listing 1 contains 3 different calculation stages:

- initialization
- main calculation process
- final calculation.

The stage 2 requires the most processor time, as the same operations are carried out millions of times. Therefore only this stage is implemented directly in hardware. Stages 1 and 3 are carried out only once, therefore they are implemented in processors MicroBlaze. Implementing *sqrt()* in hardware would require significant hardware resources (far beyond implemented adders / comparators), nevertheless high speed of this operation would reduce calculation time insignificantly. The above calculation process is a good example for hardware-software co-design.

Procedure of Linear Decimation (PLD)

One of the diagnostic procedure carried out on the PUD is the PLD [12]. The main idea behind the PLD that for a rotation machine, an input signal is sampled the same number of times for a rotation. Therefore the signal is, approximately, sampled according to the rotation angle rather than constant sampling frequency.

Consequently the PLD requires the following steps:

1) Sampling input signal with high frequency – much higher than the input signal frequency.
2) Calculating decimation ratio – which depends on the rotation speed and changes with rotation speed.
3) Filter and decimate the signal.

The most calculation-consuming part of the PLD is signal filtering before decimation. As was mentioned earlier FPGAs can implement filtering operation very efficiently, therefore the whole PLD procedure is appropriate for FPGA implementations.

Conclusions

This paper presents a hardware solution for diagnostic systems. Several arithmetic procedures are adopted in the PUD [11,12,13]. A detail description of them is outside the scope of this paper. Nevertheless by employing the PUD, input signal sampling frequency together with signal processing speed increases more than 10 times. This allows for significant increase of the overall system performance. For example, the PUD adapts the Procedure of Linear Decimation (PLD) [12] directly in hardware which results in real-time signal processing, e.g. in switching-off a rotating machinery already in run-up procedure in the case when an machine malfunction is detected [13].

REFERENCES

[1] Lin T., Zhengou Z.: The implementation of 100MHz data acquisition based on FPGA, *Proc. System-on-Chip for Real-Time Applications,* pp. 287 – 291, 2003

[2] Toscher, S. Reinemann, T., Kasper, R.: An Adaptive FPGA-Based Mechatronic Control System Supporting Partial Reconfiguration of Controller Functionalities, *NASA/ESA Conference on Adaptive Hardware and Systems*, pp. 225 – 228, 2006

[3] XILINX Inc.: Spartan-3 1.2V FPGA Family: Introduction and Ordering Information, DS099 (v1.0), *www.xilinx.com*, 2003

[4] XILINX Inc.: *MicroBlaze Processor Reference Guide* Embedded Development Kit EDK 6.3i, *Xilinx*, 2004

[5] Jamro E., Wiatr K.: Implementation of convolution operation on general purpose processors, *Proceedings of the Euromicro Conf.* Warszawa, pp. 410-417, 2001

[6] Jamro E.: Parameterised automated generation of convolvers implemented in FPGAs, *Ph.D. Thesis*, AGH University of Science and Technology, Kraków, Poland 2001.

[7] Jamro E., Wiatr K.: A Novel Parallel-Serial Architecture for Neural Networks Implemented in FPGAs, *Proc. of IEEE Design and Diagnostics of Electronics Circuits and Systems Workshop*, Sopron, pp.121-128, 2005

[8] XILINX Inc.: Fast Fourier Transform v3.2, LogiCore, *www.xilinx.com*, 2006

[9] Staunstrup J., Wolf W.: Hardware/software co-design: principles and practice, *Kluwer Academic*, Boston, 1997

[10] XILINX Inc.: Embedded System Tools Reference Manual, *www.xilinx.com*, 2006

[11] Batko W., Mikulski A.: New method of pit-shaft reinforcement evaluation with the application of pulse test. *Problemy Eksploatacji*, no. 4, Radom, pp. 173-180, 2006.

[12] Krzyworzeka P., Adamczyk J., Cioch W., Jamro E.: Monitoring of nonstationary states in rotating machinery. *Biblioteka Problemów Eksploatacji*, Radom 2006

[13] Batko W., Cioch W., Jamro E.: Monitoring system for grinding machine of turbine-engine blades. *Journal of Polish Cimac Explo – Diesel & Gas Turbine '07*. Gdańsk – Stockholm – Tumba, Poland – Sweden, pp. 37-42, 2007

***Authors**: dr inż. Ernest Jamro, AGH-University of Science and Technology, Department of Electronics, Al. Mickiewicza 30, 30-059 Kraków, Poland e-mail: jamro@agh.edu.pl*
dr inż. Witold Cioch, AGH- University of Science and Technology, Department of Mechanics and Vibroacoustics , Al. Mickiewicza 30, 30-059 Kraków, Poland, e-mail: cioch@agh.edu.pl