Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 401 – 406

ISSN 1896-7094

FPGA as a part of MS Windows control environment

Krzysztof Kołek, Andrzej Turnau

AGH University of Science and Technology
Department of Automatics, Cracow, Poland
{kko, atu}@agh.edu.pl

Abstract. The attention is focused on the Windows operating system (OS) used as a control and measurement environment. Windows OS due to their extensions may be used as a real-time OS (RTOS). Benefits and drawbacks of typical software extensions are compared. As far as hardware solutions are concerned the field programmable gate arrays FPGA technology is proposed to ensure fast time-critical operations. FPGA-based parallel execution and hardware implementation of the data processing algorithms significantly outperform the classical microprocessor operating modes. Suitability of the RTOS for a particular application and FPGA hardware maintenance is studied.

1 Introduction

MS Windows is widely used operating system. A great number of users insist to apply MS Windows in real-time control applications. In many cases the systems work satisfactory, however they may not meet the safety time-critical requirements. If these requirements are realized in a programming manner (under an operating system) then the shortest critical times are defined to be a few or tens microseconds usually. Due to the reconfigured FPGA chips we can reduce these times to a few or tens nanoseconds.

2 MS Windows and its extensions

As far as a real-time operation is concerned three Windows environments are tested and compared. In fact, one would rather call it "a soft real-time" [4]. The following Window applications are examined:

- excited by Windows timer message (WM Timer),
- excited by multimedia event timer,
- Real-Time Windows Target (RTWT).

To perform the tests we need a time reference for the measurements. Software timers are not accurate. They introduce unpredictable overhead to the measurements. Fortunately, we can use the 40 MHz clock available at the RT-DAC4/PCI board [2]. To measure the accuracy of the timer events the periodic tasks are started in each investigated environments. The only goal of these tasks is read the 40 MHz timer. The timer values serve as the time stamps and allow to estimate the jitter of the periodic task.

The first experiments related to Windows timer message have been tested. The periods of the timer is set to 100 ms or 10 ms. For both experiments the several individual tasks were running concurrently. It means that during the experiments MS Windows was heavy loaded. The results in the form of histograms are shown in Fig.1. There are two mean values: 109.9601 ms and 15.7571 ms. These values are far away from the desired values: 100 ms and 10 ms. Such an operation is not acceptable for the control purpose. It shows that the time

taken for the system to switch from a given task to the task triggered by the timer message is significantly large and does not satisfy measurement and control requirements in the case when the sampling time is below tens of milliseconds.

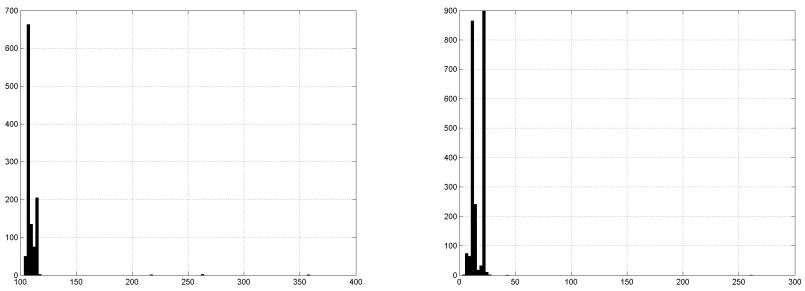

Fig. 1. Windows application excited by WM Timer; left) 100 ms – sampling, right) 10 ms – sampling period

Unlike the other solutions the jitter of the WM Timer events is relatively small for sampling periods greater than hundreds of milliseconds. Therefore, to demonstrate the jitter effects in the other cases the smaller sampling period will be used.

The next experiments correspond to the Windows application excited by the multimedia event timer. For example, this method is used by Real-Time Connection (RT-CON). The graphical results of two similar experiments of the sampling period equal to 10 ms are shown below (see Fig. 2). The mean values are correct but a large jitter happens occasionally (even 25 ns).

The final experiments correspond to Windows extended to RTWT. The graphical results of two experiments excited by the sampling periods equal to 10 ms and 1 ms are given in Fig. 3. The mean values are correct. We observe a small jitter. This solution works perfectly except that it does not support majority of the Windows API functions. For example, USB protocol.

The benefits and drawbacks of the presented experiments are summarized in Table 1. One can notice that it is hard to find an ideal omnipotent solution. In the two first cases to access I/O memory space one needs to apply a dedicated driver. In the RTWT case which seems to be the most suitable tool almost all the Windows API functions are not supported.

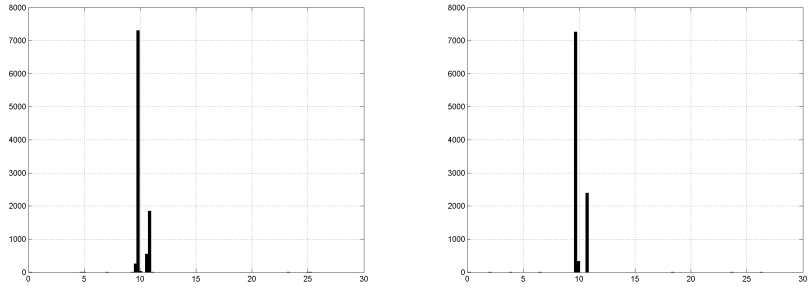

Fig. 2. Windows application excited by multimedia event timer; similar experiments - 10 ms sampling period

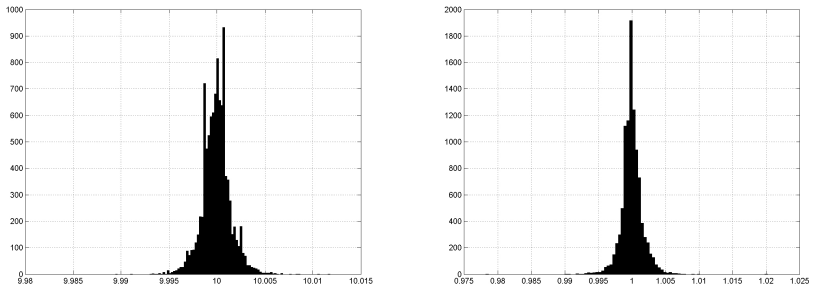

Fig. 3. Windows application extended to RTWT; left) 10 ms – sampling, right) 1 ms – sampling

Table 1 . Benefits and drawbacks of the presented MS Windows environments

	Benefits	Drawbacks
The Windows application excited by Windows timer message WMTimer	All functions from the Windows application programmer interface (API) operates properly	The average jitter is not acceptable. I/O memory space is available for drivers only
The Windows application excited by multimedia event timer	The average jitter is acceptable. Nearly all functions from Windows API operates properly. For example, support for USB	A large jitter happens occasionally. I/O memory space is available for dedicated I/O drivers only
Real-Time Windows Target RTWT	Small jitter. Very good time stability	Almost every function from Windows API is not supported. I/O memory space is available

We can create add-on functionality via extensions to MS Windows which may improve to a large extent responsiveness of original Windows. The primary goal is to minimize pre-emptive and interrupt latency. Unfortunately Microsoft kernels do not provide deterministic response and the jitter of the sampling may be significant.

3 FPGA based time-critical operations

The Field Programmable Gate Arrays (FPGA) contain a matrix of cells ready to be configured by a user to satisfy application requirements. The capacity of the FPGA chips is high enough to implement user-designed processors. As far as the data processing speed is considered the FPGA-based parallel execution and hardware implementation of the data processing algorithms significantly outperform the classical microprocessor operating modes. High speed is achieved due to advanced technology and through improved architecture, and supports system clock high rates (in our design 40 MHz). One has to remember that safety in control environment depends on a time delay that the system responds to a critical event. In a FPGA-based system it means 25 ns or less (in a case of a higher than 40 MHz clock rate). For the PC-based system the shortest time delay is a few μs. FPGA is hundreds or thousands times faster due to its parallel operation and due to the fact that the processing algorithms are implemented in hardware. Nevertheless high speed, it is hard to predict a worst-case guaranteed performance because delays in FPGA logic cells are layout dependent. Critical part of a design are shift registers and counters. They should run faster approximately two thirds of the specified toggle rate.

One can use a vast number of complex and advanced computer boards equipped with FPGA. We focus our attention to RT-DAC4/PCI board (see Fig. 4) which include two main elements: the PLX9030 PCI bridge and the FPGA chip [3]. The bridge is the local data bus master responsible for data transfer between the PCI and local bus. The local data bus is connected to the FPGA chip. Each analog and digital I/O function of the board must be implemented as a logic part of the FPGA chip. By the computer applications the RT-DAC4/PCI board is visible as a 64 double words located in the I/O address space. An extra 64 double words I/O space is accessible if interrupts are considered. The access to the board resources is done by the *inp*, *inpw* and *inpd* or *outp*, *outpw* and *outpd* C-language instructions.

The board can run at synchronous system clock rates of 40MHz applied to synchronize user-designed logic. The clock signal (40 MHz) born at the board and connected to the FPGA chip generates the local bus clock signal which is asynchronous to the PCI bus clock and is independent to the PC microprocessor clock. It means that FPGA works autonomously. Even the reset signal to the PC does not affect the FPGA operations. These features create a temptation to use an FPGA chip to be applied as safety time-critical device.

To illustrate safety time-critical operations realized by an FPGA circuit the control of the 3D Crane laboratory system is examined (see Fig. 5). The system can be used to develop even time-optimal control algorithms [5]. The real-time and free of delays operation is guaranteed due to functionality embedded in the FPGA logic.

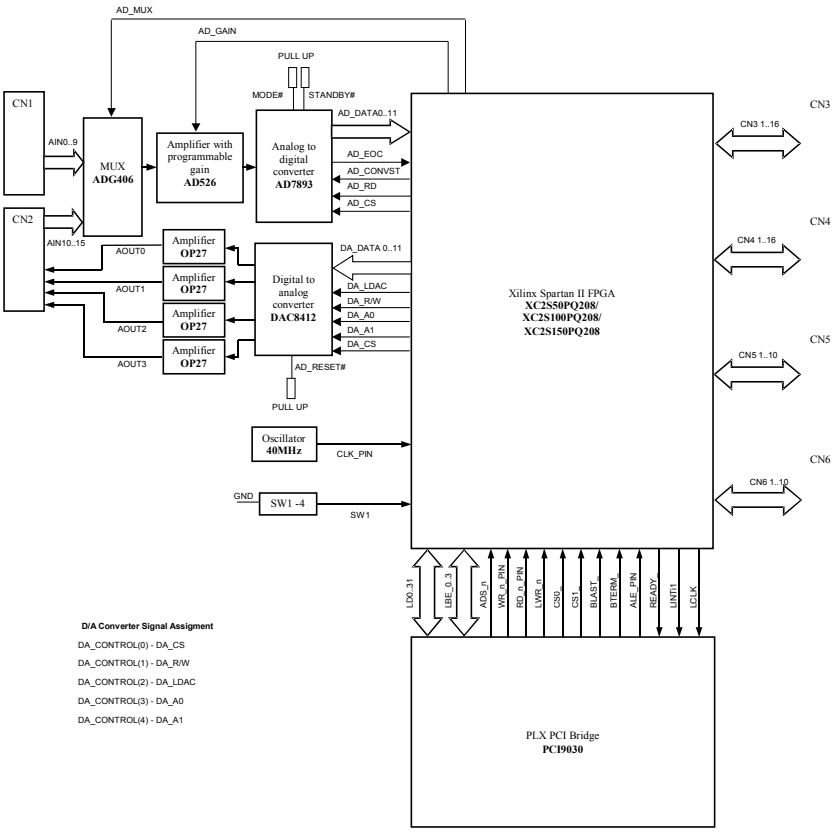

Fig. 4. Block diagram of the RT-DAC4/PCI board (courtesy Inteco Ltd)

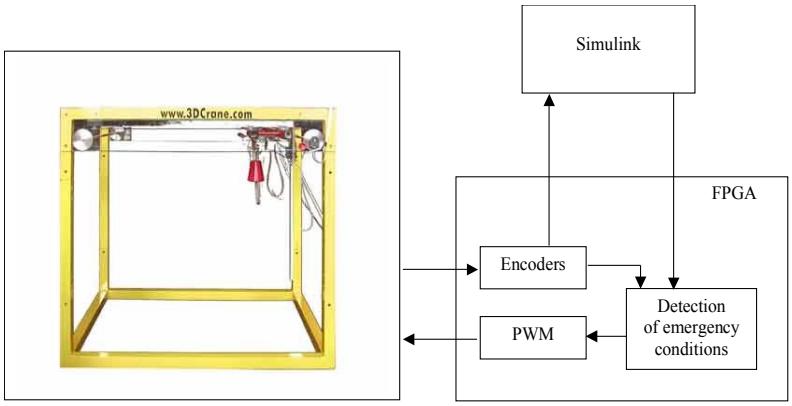

Fig. 5. Configuration of the 3D Crane measurement, control and detection of emergency conditions system

There are several critical constrains related to the crane construction and control. It is strictly forbidden to go beyond the 3D Crane system ranges. The bridge and the cart of the crane must remain inside their working space. The temperature and supply voltage of the power amplifier are permanently monitored and supervised. A violation of an arbitrary constraint rule has to result in a fast (without a delay) reaction of the system. These safety functionality is inserted (configured) directly in FPGA. It is guaranteed that the system responds to a violation of rules within 25 nanoseconds (see the *Detection of emergency conditions* block). Beside the critical safety rules the FPGA system is configured to become the interface to sensors (encoders) and actuators (the Pulse Width Modulation (PWM) generators applied to control the DC motors).

The controllers are build in Simulink equipped with the RTWT toolbox. These controller generate controls and define parameters responsible for safety. In this way one does not need to worry about control safety. Automatic (triggered) behaviors will take place free of delays of the operating systems.

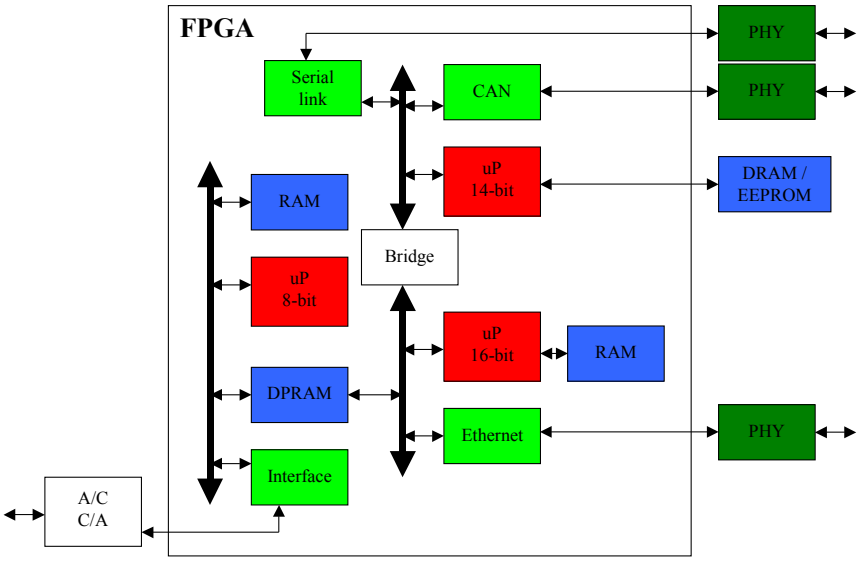

Fig. 6. System on the Chip configuration

The System on the Chip (SoC) configuration shown in Fig. 6. is an extended version of the configuration shown in Fig. 5. The SoC approach characterizes reach functionality – it means that multiple system components are placed on a single silicon chip [1]. The FPGA circuits contain resources sufficient to implement microprocessors, operational memory, interfaces to measurement and control signals, interfaces to communication channels, signal processing blocks, etc. One can decompose tasks and implement them as programs of

microprocessors or dedicated logic blocks. In peculiar, we can define the components responsible for time-critical safety operations.

The main benefit of FPGA is its reconfiguration flexibility. It might happens that urgent modifications to a logic are required. In this case we can introduce them in a very fast and relatively simple way. None modifications are introduced into the printed circuit board! A dedicated software is used to design and download a new FPGA configuration. The new implemented functionality is realised by the FPGA hardware.

Summary

MS Windows due to extensions pretends to be a real-time OS (RTOS). However, one has to be aware that the jitter of sampling time might be different in peculiar cases. It also might happen the crash of the OS (a well known “blue screen” effect).

In fact, to ensure the safety time-critical operation the FPGA logic is proposed. A dedicated hardware or microprocessor configured blocks may respond in a robust and fast way to the time-critical events independently to operating system programs.

References

1. Kołek K.: *FPGA as a Part of Control System,* 14th International Conference on Process Control ’03, Strbske Pleso, Slovak Republic, June 8-11, 2003, p. 207, CD.
2. RT-DAC4/PCI Board. Users Guide, `http://www.inteco.com.pl`
3. Spartan-II 2.5V FPGA Family: Functional description, `http://www.xilinx.com` .
4. Timmerman M.: *Windows NT Real-Time Extensions better or worse?* Real-Time Magazin 98-3, pp. 11-19.
5. Pauluk M., Korytowski A., Turnau A., Szymkat M.: *Time optimal control of 3d crane.* Proc. VII IEEE Int. Conf. “Methods and Models in Automation and Robotics”, Międzyzdroje, August 28-31 2001, pp. 927–932.