FPGA Chip as a System Master for Hardware Aided Parallel Computing

Jacek Pierzchlewski, Paweł Śniatała, Błażej Nowakowski, Andrzej Rybarczyk, Wojciech Wencel
Chair of Computer Engineering
Poznań University of Technology
Piotrowo 3A, 60-965 Poznań, Poland
Email: sniatala@et.put.poznan.pl

Abstract

This paper presents prototype board and its operating system dedicated for application specific parallel processing. The proposed architecture consists of two AVR microprocessors, FPGA Spartan3, SRAM and Flash EEPROM Memories, DA converters, and several serial communication ports. To make the system "designer friendly" a supervising algorithm, which can be called as a kind of "operating system" was elaborated. The algorithms were described in VHDL. The Spartan3 FPGA was chosen as a target platform to implement the master controller for the system. Necessary IO devices' controllers were implemented in AVRmicro. The designed board with elaborated libraries provides convenient solution to develop dedicated parallel processing systems.

1 Introduction

Parallel processing is a widely used approach in order to accelerate computing, signal processing or to increase the reliability of systems. The research efforts focus on both: new hardware structures dedicated for parallel processing and new algorithms to utilize the hardware components [1, 2]. The implementations can be done based on multiprocessor computer or computer clusters. However, for specific, particular problems clusters of computers can not be optimal solution in terms of power consumption, speed, size or economical factors. In such cases a dedicated multiprocessor platform can be a competitive proposal. In this paper we propose a prototype board, which can be used as a testing platform for such kind of solutions. Taking into account the popularity of FPGAs and AVR microcontrollers among designers [3], we decided to use these components as basic modules of our system. To make the system "designer friendly" we have elaborated a supervising algorithm, which can be called as a kind of "operating system" of our board. The following section presents the structure of the prototype board. Next, the concept of the operating system is introduced. Finally, testing applications for the board are proposed.

2 Hardware structure of the board

As was mentioned in the introduction the actual built prototype board consists of the following components:

- FPGA - Xilinx Spartan 3 with external connected 256KB static RAM, RS485, four A/D converters, LCD display,
- AVR ATMega128 microcontroller with external connected RS232 and SPI
- AVR ATMega128 microcontroller with external connectedRS232, RS485, SPI and 1.5MB EEPROM.

The configuration of the system is presented in Fig. 1.

The components listed above were actually put in our built prototype board. However, in general, the structure can be easy extended, because not all available pins of AVR micro and FPGA were used. The picture of the board, presented in Fig. 2, shows the testing board with the empty places for a possible structure extension.

3 Supervising "Operating System"

3.1 The OS concept and functionality

The hardware board presented above is a quite complex system to be controlled. In order to take this effort off from the board programmer, a supervising control system was elaborated. The system can be seen as an "operating system" (OS) of the board. Usually, the OS system is considered as a software. In our solution however, the OS system consists of several pieces of hardware. The algorithms, which are needed in the OS, can work in parallel, each in its hardware part. Hence, the FGPA available on the board,

Proceedings of the International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)
0-7695-2554-7/06/$20.00 © 2006 IEEE

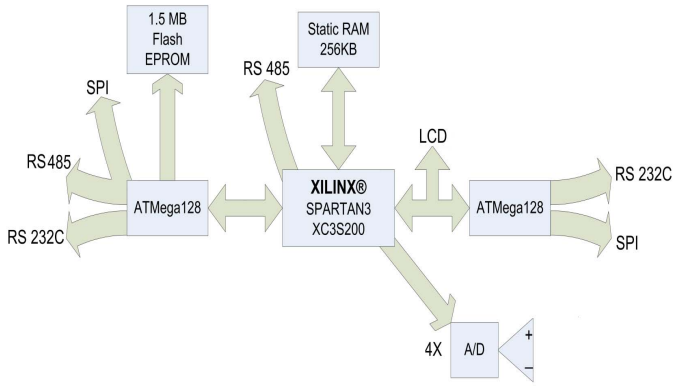

Figure 1. The structure of the proposed system

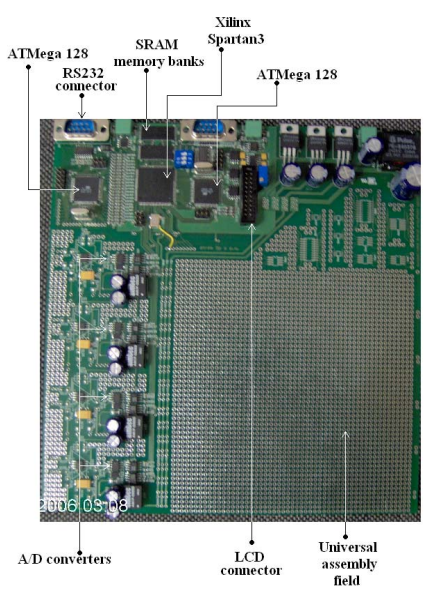

Figure 2. The prototype board

was chosen as a target platform to implement the proposed OS.

Several requirements were formulated for the OS. The system should provide to a programmer a high level of abstraction. The user doesn't know about the actual hardware connections inside the board and should only view the system at the higher level of abstraction. It means that the OS should take care of the following exemplary tasks:

- allocate any hardware resources offered on the board,
- allocate memory area,
- provide a virtual memory mechanism,
- access to the allocated resources using handles,
- implement mutex algorithms to protect the resources from unsynchronized simultaneous access.

The system should allow scalability of the hardware. In particular, it should be able to handle additional processors, FPGAs or I/O modules. The implementation of the proposed OS in the FPGA must be concise enough to leave enough FPGA's resources for user applications.

3.2 The OS interface

As was mentioned, the proposed board can be extended with additional hardware resources. In the Fig. 3 the supervisor is a hardware (part of the FPGA), which implements the OS. In general, the FPGA presented in the Fig. 3, can be additional device or, as in our particular realization, the rest (unused by supervisor) part of the only one FPGA. In order to communicate between the supervisor and the resources the simple bus system was proposed. It consists of only 3 control lines and 16 bits data bus as presented in Fig. 3. This set of signals is enough to control 1 channel of data.

The supervising module, shown in Fig. 3, has connections to its clients: microprocessor/microcontrollers and others hardware modules on the board (eg.FPGAs). Referring to Fig. 3, let's describe the procedure required to establish connection between any client and supervisor. The client can need resources connected directly to the supervisor or to another client devices. In both cases the client "calls" the supervisor by asserting a low signal on the c-clk line. The supervisor answers the call by asserting a low signal on the s-clk line. Next, procedure depends on the physical location of the required resource. If it is connected directly to the supervisor and the resource is not busy, this resource can be allocated to the client. The data transmission can be initialized immediately or can be used later by the client. To use it later, client must call supervisor again, but this time only to initialize the transmission, since the resource is still allocated to this client. The client sends

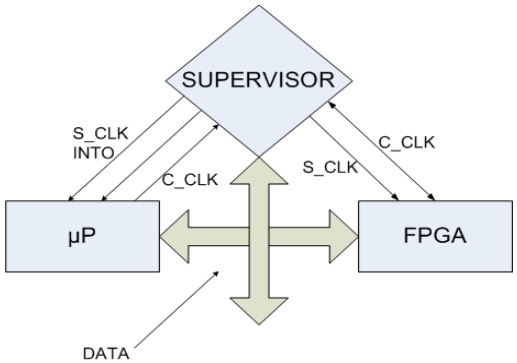

Figure 3. The proposed operating system bus.

only the handle and data. The resource is released by sending release request by the client to the supervisor. The data transmission uses 16 bits data bus and is synchronized by the c-clk clock signal. More complex situation is, when the required resource is not connected directly to the supervisor but is connected to another client. In such case, supervisor must call this client. If called device is busy, it is preemptived by the supervisor, but only if the client will allow to. For example, to preemptive the micro-controller, the supervisor sends the highest priority interrupt signal. After that, again the handshaking protocol establishes the connection. This time the data transmission is synchronized by the s-clk signal. In other, word in the second case, the supervisor established kind of DMA channel between client requesting the resource and the client, which can provide required resource. The actual driver, to handle the DMA transmission between clients, is located in the device, where the resource is connected to. This internal driver is responsible for the data transmission between clients through the DMA channel established previously by the supervisor.

This simply handshaking protocol was chosen to minimize number of pins required for the actual implementation. A frame, sending to the supervisor, has a form presented in Fig. 4. It contains the request code, handle and actual data.

Some of the possible requests are listed below:

- allocate the device of a given address,
- allocate n bytes of RAM,
- release the device/memory,
- send n bytes to the device of a given address,
- write a byte to the memory to a given address,
- write n bytes to the RAM starting at a given address.

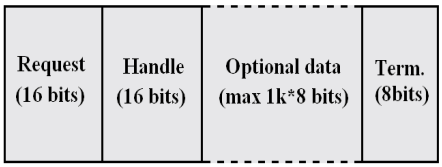

Figure 4. An example of the protocol frame.

As was mentioned above, the designed OS should control any number of channels presented in Fig. 3. For the multi-channel version of the system, the possible extension depends on the supervisor device capability.

3.3 Hardware Structure of the OS

The system was described in VHDL. As a target platform the XILINX Spartan 3 was chosen. For a bigger structure, it could be necessary to use a more advanced FPGA, mainly because of the need of additional I/O pins. The structure of the supervisor module is presented in Fig. 5. The module is connected to the channels using the bus signals located on the left side. The signals were described in Fig. 3. On the right side of the Fig. 5, the connections to the devices connected directly to the FPGA are presented. In our prototype board SRAM, RS485 and A/D converters are connected directly to the FPGA. After receiving a request from a client the interrupt controller decides, whether sends the acknowledge to the client or not. The decision is taken based on the priority rules, which are collected in the controller based on the information given from the instruction decoder. When the transmission is established the data frame is receiving from the client. The frame, in the form presented in the Fig. 4, is decoded in the instruction decoder. Each channel has its bus driver or device driver. Based on the decoded request the needed resources are scheduled in the resource and MUTEX tables. Finally, the DMA controller make all necessary data connections between clients and resources. It is a multichannel controller, since more than one connection can be established simultaneously if possible. To control all time restrictions, programmable timers are included in the system.

4 Practical experiments

In order to check the functionality of the designed board and OS, image processing algorithms will be implemented. In particular the MPEG7 color descriptors extraction will

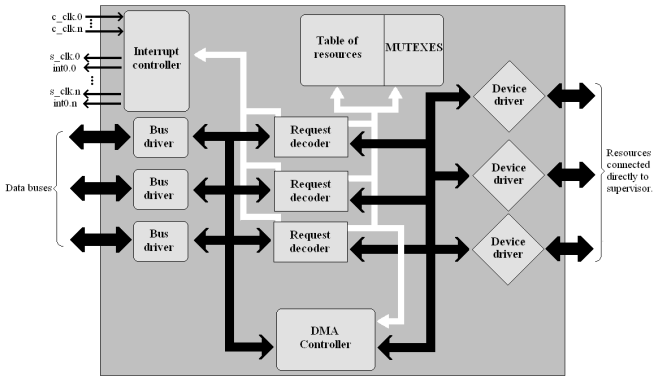

Figure 5. The supervisor hardware structure.

be tested. The goal is to parallelize calculation of different descriptors. The chosen descriptors are:

- Scalable Color,
- Color Structure,
- Color Layout.

The chosen descriptors were already implemented in a non parallel approach. More details about the descriptions extraction algorithms and their hardware implementation can be found in our paper [4]. Having results from the non parallel implementation, it will be possible to compare those results with the implementation based on the new designed board.

5 Conclusion

The paper presents a new hardware system suitable for parallel processing. An original concept of the ”operating system” for the board was presented. The prototype board was build and tested. The idea of the board was to make it as flexible as possible, keeping a low complexity of the system. The board contains components widely used by engineers: AVR micro-controllers, FPGAs. The designed operating system makes the programming task of the board easier for a final designer. The proposed board can be a good solution for a multithread tasks.

References

[1] P.P. Jonker and W. Caarls. Application driven design of embedded real-time image processors. In Proceedings of Acivs 2003 (Advanced Concepts for Intelligent Vision Sys- tems). Ghent University, September 2-5 2003.

[2] Seinstra F.J., Koelma D. Lazy parallelization: A finite state machine based optimization approach for data parallel image processing applications. In Proc. of the Workshop on Parallel and Distributed Computing in Image Processing, Video Processing, and Multimedia (PDIVM 2003), 2003. Held in conjunction with IPDPS 2003.

[3] James-Roxby P., Schumacher P., Ross Ch. A Single Program Multiple Data Parallel Processing Platform for FPGAs, In Proc. of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM04).

[4] Savakis A., Śniatała P., Rudnicki R., Hardware Implementation of MPEG-7 Color Descriptors. In Proc. Mixed Design of Integrated Circuits and Systems, MIXDES’2004, Szczecin Poland, Vol.1, pp..