

# Parallel FPGA Processor Card for Distributed Information Processing

This article presents an FPGA-based board, the Parallel FPGA Processor Card PFPC-1V, designed for real-time DSP applications. The board is intended for operating as a hardware accelerator in conjunction with general purpose main processor on VME bus. PFPC-1V includes 1 RISC-like control unit and 6 processing units implemented in ALTERA FLEX 10K, and 2 bidirectional link ports. Architectures of control and processing units can be flexibly adapted to certain tasks, and for any particular architecture appropriate software can be generated.

## INTRODUCTION

In the paper a Parallel FPGA Processor Card (PFPC-1V) for distributed information processing is presented. In developing the device both parallelism and hardware implementation of algorithms were involved thus supplying high computation speeds for a number of applications. In order to eliminate the drawback of fixed non-universal architecture the FPGA are used. The use of FPGA chips permits the flexible change of the local architectures of processing units what makes it possible to implement computational algorithms of different kinds in hardware just by on-line reconfiguring the corresponding FPGA.

## THE HARDWARE

### The Structure and a Brief Description

The device is developed for VME bus systems and all its basic processing units are built on Altera FLEX 10K family FPGA chips. The card is designed as a hardware accelerator and must be always switched into the system as a slave on VME bus with a conventional universal processor as a master (host-machine), e.g. Intel 80x86 under OS MS-DOS or real-time OS VxWorks. A version with the PCI system bus is also possible. Operation frequency is 33 MHz.

The structure is shown in fig. 1. The device consists of the following units:

- Control Unit (CU);
- six Base Processing Units (BPU1-BPU6);
- Exchange Bus Controller (E-bus Controller);
- VME bus Controller;
- two SRAM arrays (SRAM1, SRAM2);
- high-speed serial receivers/transmitters.

The Control Unit controls the BPUs and manages the data flows on the card, it is can be considered as a RISC processor. The processor structure and instruction set can vary depending on the particular task requirements. The actual operations on the data are performed by the BPUs, whose architectures provide hardware implementations of the corresponding algorithms. In BPUs different kinds of adders, multipliers, decoders, ALUs, table functions and so can be implemented. The CU and BPUs are implemented in Altera FLEX 10K family FPGA chips, hence, their structure can

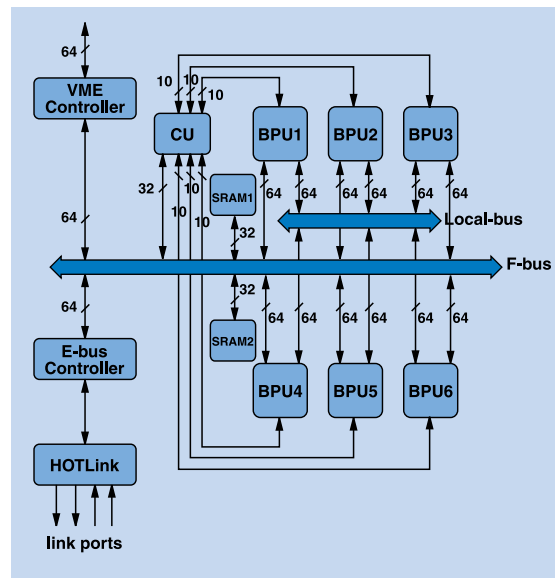


Figure 1. The Parallel FPGA Processor Card structure

be flexibly changed on-line by configuring the FPGA through VME bus with pre-designed configuration files. Two arrays of local SRAM are made up of 8 SRAM chips with each size of 0.5Mbyte thus providing total static memory size of 4Mbyte. Each array is arranged as 512K 8-bit words. Both arrays are connected with the CU and the VME Controller through individual address and control wires and can operate independently from each other.

Two or more separate Parallel FPGA Processor Cards or the cards and any other devices with the appropriate interfaces can be interconnected through the serial receiver/transmitter HOTLink™ channel, by Cypress corp. The E-bus Controller manages data transfers through the channel. It is implemented in the Altera FLEX 10K as well and consists of four FIFO pipelines and a number of data and control registers. A VME Controller provides interfaces with the host-machine through VME bus.

As far as for a programmer the Parallel FPGA Processor Card can be considered as a RISC processor (the Control Unit) and six vector processors (the BPUs), which execute MIMD instructions (see fig. 2). All the BPUs and the CU can operate in parallel, the BPUs are connected with each other through 64-bit bus, with

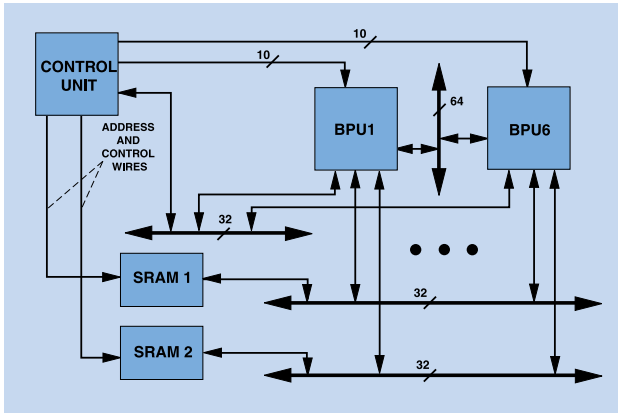


Figure 2. Simplified structure of the Parallel FPGA Processor Card

the CU through 10-bit buses and with the local SRAM arrays through 64-bit bus.

### The Control Unit

The Control Unit structure is shown in fig. 3. The CU consists of the following:

- Internal instructions memory (IIM);
- Instruction decoder;
- Data and constants registers;
- Address registers;
- BPUs registers;
- Control unit and status register (CSR);
- ALU;
- Program counter and stack pointer;
- Local SRAM controller.

The active program is stored in the Internal Instructions Memory (IIM), which is implemented in the FLEX 10k

Embedded Array Blocks (EAB). The IIM is arranged as 2048 32-bit words. The IIM is loaded when the FPGA is configured, moreover, there is a possibility to upload the program into IIM on-line through VME bus.

The IIM can be considered as a set of sub-programs (micro-codes) each of which implements a particular computational algorithm. The changeable parameters of micro-codes – sizes of matrices and vectors, numbers of cycles and so on, are stored in the corresponding registers which are loaded from the host-machine through VME bus before the program starts. Any instruction in the processor is executed in three clock ticks. The three-level pipeline allows for a new command to be uploaded every clock tick. For even pipeline filling all instructions are executed for the same number of tick counts.

Processor has eight 24-bit address registers RA0-RA7, six 32-bit data registers RD2-RD7, eight 8-bit constant registers, twenty one 10-bit BPUs registers and fast branch registers. The CSR register can be considered as the least significant data register RD0. The most significant bits of RD1 register are used as cycling counter for fast branch instructions and the least significant bits are for cycling the MIMD instructions sent to the BPUs. The rest data registers RD2-RD7 are used as general purpose registers.

The BPUs registers are to store the MIMD instructions the CU sets on the 10-bit buses that connect CU and BPUs. All the BPUs registers are divided into three groups – seven registers for every two BPUs. Only one of seven instructions for a BPU can be available at a moment – the instruction stored in one of the BPU registers – and usually it is well enough for the most of

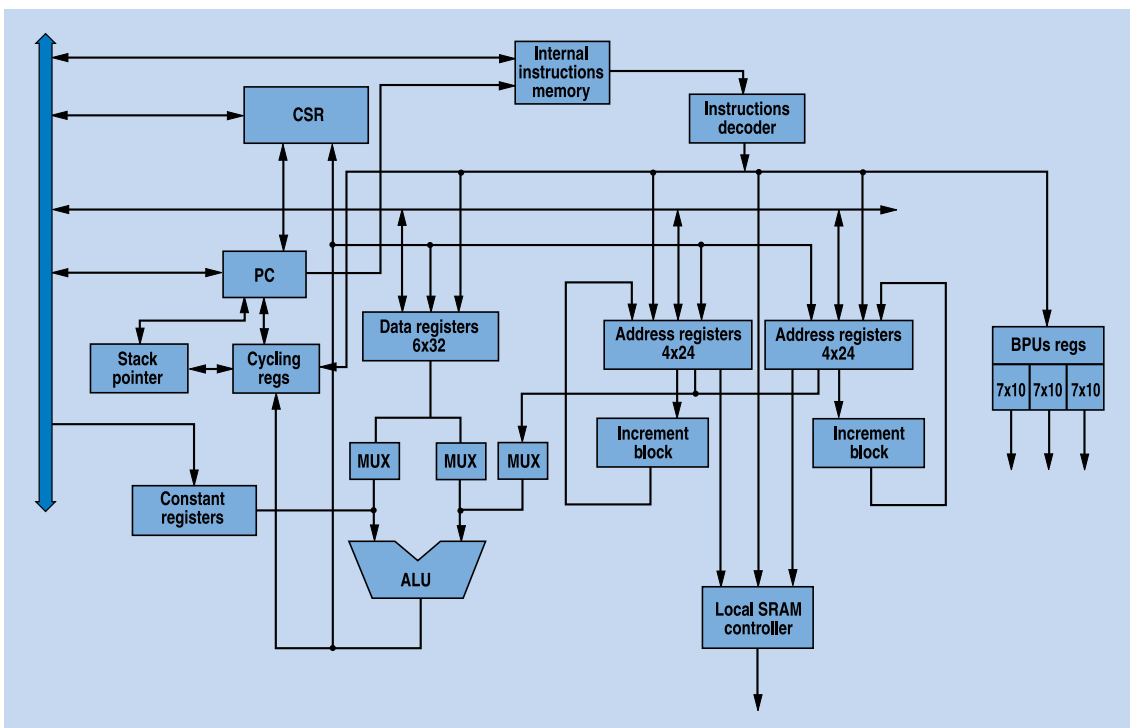


Figure 3. The Control Unit

applications. The BPUs registers are loaded in the micro-code body and if it is needed there can be uploaded more MIMD instructions.

The address registers are used for handling the addresses of the data stored in the local SRAM arrays. All address registers are divided into two groups with four registers per group. A group corresponds to the particular SRAM array. After execution of the MIMD instruction any address register can be incremented with the value stored in the special increment registers separately for each group. The need for increment is determined in the CU instruction body for each group individually, i.e. two address registers from different groups can be incremented at the same time. The local SRAM arrays are handled by the local SRAM controller that supports SRAM chips with access time of 20 ns and more.

The CU is assigned to manage the data flows and to control the on-card units and, for this reason, its ALU has a highly simplified structure. According to the requirements of algorithms implemented up to date ALU can perform addition, subtraction, shifting and some logical operations. The main CU instruction is for the BPUs control. The instruction says what BPU register the MIMD instructions should be taken from individually for each BPU, what address registers and what operation mode should be used for the local SRAM arrays, and whether the corresponding address registers should be incremented or not. The instruction can be repeated for the required number of times according to the value stored at the least significant bits of RD1 thus eliminating the losses on branches in the cycles when the MIMD instructions should be performed more than once.

To implement a particular computational algorithm one should determine a specific CU instruction set that would allow for the best performance to be achieved for the task. In a hardware design language (HDL), such as Verilog or VHDL, the appropriate CU architecture is described and the corresponding FPGA chip is configured with the architecture supporting the instruction set required. Thus one gets a specialized computational system adapted to the particular task requirements on the one hand and a universal one that can be effectively used for the various kinds of tasks on the other hand.

### The Base Processing Units

The BPUs are assigned to perform elementary operations such as addition, subtraction, multiplication, and kinds of arithmetic, logical and table functions. A BPU consists of register files, internal SRAM of 2Kbyte size, logical and arithmetical units. Such resources make it possible to implement, for example, various vector and matrix functions that are basic in image processing or artificial neural networks modeling. A generalized structure of BPU is shown in fig. 4. The particular architecture is determined by the particular task requirements. A BPU execute the instruction that comes from the CU and the data are taken either from the internal storage units or from the local SRAM arrays, or in some cases from other BPUs. That allows for the distributed information processing to be arranged.

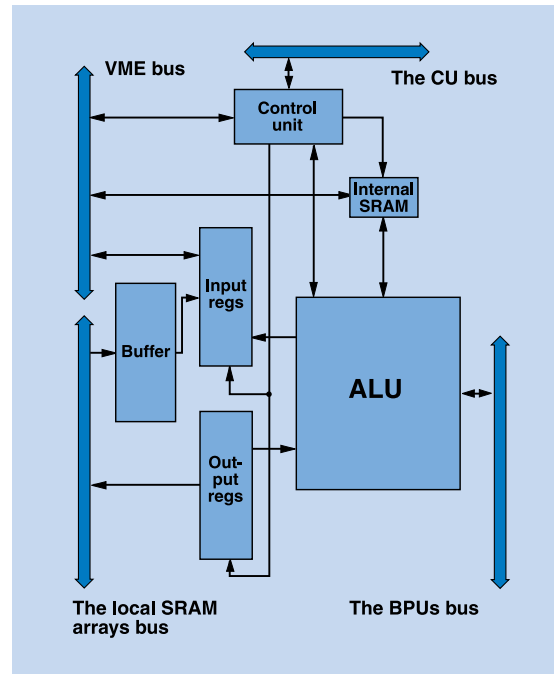


Figure 4. A generalized BPU structure

### THE SOFTWARE

The supporting software development is an essential part of designing a computer. In our case we faced a task of developing a flexible programming language with easily changeable sets of commands and instructions, with taking into account the need for some of the CU instructions to provide the parallel operating of different BPUs and the concurrent access to the local SRAM arrays. An assembler-like language was built basing on the Dialogue System of Structural Programming (DSSP). An appropriate compiler is developed that allows for any instruction to be added or deleted easily and the existing instruction codes to be modified in a simple and fast manner. The DSSP is a FORTH-like language, which has a lot of facilities for developing on its base an instrumental and low-level debugging and device testing software. In DSSP a new operator can be added by defining it in terms of previously defined operators. An assembler instruction can be considered as a DSSP instruction and then an assembler program will at the same time represent a program for DSSP. Thus the compiling process involves just a corresponding DSSP program execution. As a result a binary code is generated which is then loaded into the IIM of the CU.

### APPLICATION FIELDS

The Parallel FPGA Processor Card can find a lot of applications to be effective in. In particular, various algorithms of image and signal processing in real-time can be implemented, such as filtering, convolution, Furie transform, correlation analyses, wavelets, pattern searches, compression/decompression, multidimensional analyses and other. The card can be used in telecommunications for tasks of coding/decoding and exchange protocols support. In the field of molecular biology the card is quite effective for the DNA, protein

ALGORITHM	PENTIUM-100	PENTIUMII-333	HYPER SPARC	PFPC-1V
Convolution with filter 4_4	0.65	0.11	0.76	0.02
Median filter	1.97	0.49	0.75	0.001
Sharpening	0.51	0.13	1.31	0.004
Comparing with mask 32_32	43.78	7.14	58.89	0.142
Matrix multiplication	8.61	0.60	12.31	0.011

Table 1. Algorithm Execution Times

### Notes:

- 1) Image size is 256x256 bytes;
- 2) Pentium-100, RAM 16 Mb;  
PentiumII-333, RAM 128 Mb;  
Hyper SPARC, 200 MHz, RAM 64 Mb;

and other sequences alignments: finding similarities using dynamic programming and Smith-Waterman algorithms, searching the databases like GenBank for the closest alignments and other analyses. Another possible use is the artificial neural networks modeling and different kinds of neuron-like nets can be successfully implemented. It can be also used for systems monitoring and control – intellectual controllers can be implemented. In Table 1 there are shown times required to perform some special algorithms of image processing on the PFPC-1V and on some PC or workstation platforms to compare speeds.

### REFERENCES

- [1] M. Gokhale et al., "Building and Using a Highly Parallel Programmable Logic Array", *Computer*, No.1, Jan. 1991, pp. 81-89.
- [2] M. Aubury et al., "Advanced Silicon Prototyping in a Reconfigurable Environment", IOS Press, 1998.
- [3] B. Denby, "Neural Networks and Cellular Automata in Experimental High Energy Physics", *Computer Physics Communications*, Vol. 49, 1988, pp. 429-448.
- [4] AMPP Catalog, June 1998, by Altera Corporation.
- [5] D. Frantov, M. Shumakov "DED – DSSP Editor and Debugger", *Proceedings of EuroForth 97*, Oxford.
- [6] G. R. Goslin, "A Guide to Using Field Programmable Gate Arrays (FPGA) for Application-Specific Digital Processing Performance", Xilinx Inc., V.1.0, 1995.

### World Wide Web sites

- [7] <http://compugen.co.il/products/bioc-t.html>

### SUMMARY

Real-time DSP applications require both properly fast and complicated computers to solve numbers of quite sophisticated modern tasks at appropriate rates. In the paper a Parallel FPGA Processor Card (PFPC-1V) for

distributed information processing is presented. In developing the device both parallelism and hardware implementation of algorithms were involved, thus supplying high computation speeds for a number of applications. In order to eliminate the drawback of fixed non-universal architecture the FPGA are used. The use of FPGA chips permits the flexible change of the local architectures of processing units what makes it possible to implement computational algorithms of different kinds in hardware just by on-line reconfiguring the corresponding ■

*Sergey Aryashev is a junior researcher in the Department of Distributed Systems at Institute for System Studies in Moscow, Russia. He received his diploma in engineering physics from the Moscow State Engineering Physics Institute, Russia, in 1996. His research interests include ASIC and FPGA based hardware, signal and image processing systems, parallel processing, Verilog and VHDL hardware design languages.*

*Sergey Bobkov is a head of Department of Distributed Systems at Institute for System Studies in Moscow, Russia. He received his diploma in engineering physics and Ph.D. degree in electrical engineering from the Moscow State Engineering Physics Institute, Russia in 1978 and 1983, respectively. His research interests include industrial and special purpose computers, VLSI devices designing, advanced computer architectures.*

*Evgueni Sidorov is a junior researcher in the Department of Distributed Systems at Institute for System Studies in Moscow, Russia. He received his BS degree in physics and diploma in engineering physics from the Moscow State Engineering Physics Institute, Russia, in 1998 and in 1999, respectively. His research interests include ASIC and FPGA based hardware, signal and image processing systems, advanced data processing techniques.*

*Ilya Yudin is a junior researcher in the Department of Distributed Systems at Institute for System Studies in Moscow, Russia. He received his diploma in system engineering from the Moscow State Institute of Electronics and Mathematics, Russia, in 1999. His research interests include data base engineering and real-time software designing.*