A&QT-R 2002 (THETA 13)
2002 IEEE-TTTC International Conference on Automation, Quality and Testing, Robotics
May 23 – 25, 2002, Cluj-Napoca, Romania

CONFIGURABLE PROCESSOR

Zoltan Baruch, Octavian Creț, Kalman Pusztai

Technical University of Cluj-Napoca, Romania, Computer Science Department
E-mail: {Zoltan.Baruch, Octavian.Cret, Kalman.Pusztai}@cs.utcluj.ro

Abstract. Configurable architectures can deliver the high performance required by computationally-demanding applications, similar to the ASIC circuits, while providing the flexibility of the programmable processors. The performances achieved by these architectures are often one or two orders of magnitude higher than those of processor-based alternatives. In this paper we describe the design and implementation of a configurable processor. The processor consists of a constant part and a configurable structure. The constant part allows to solve simple applications without changing the existing resources. The configurable part is defined by the user, based on the requirements of a specific application. This part contains application-specific functional blocks, controlled by special instructions. The integration of a classical processor and a configurable architecture within the same circuit allows to exploit the advantages of both architectures. For the design of the configurable processor we used the VHDL language. The implementation was performed using a *Xilinx* XCV600E FPGA device. This processor can be used in several types of applications: data encryption and compression, image processing, digital signal processing, special arithmetic.

Key Words: Configurable computing, Configurable architectures, Reconfigurable devices, FPGA devices.

1. INTRODUCTION

Regarding their advantages, the configurable computers or processors are situated between ASICs (*Application-Specific Integrated Circuits*) and programmable processors. Application-specific circuits allow the highest performance by sacrificing flexibility. General-purpose programmable processors allow the highest flexibility, but their performance is lower compared to a solution adapted to a particular application. Configurable computing ensure the performance required by computation-intensive applications, maintaining in the same time the advantages of the programmable processors.

Application-specific instruction sets, customized I/O solutions and optimized control can substantially improve the performance of programmable processors [9]. Before the advent of programmable circuits, modifying a processor’s instruction set required a writable control store and a custom microprogram for each particular application. Programmable circuits provide an adequate implementation platform for application-specific control units, due to the rapid development time and simplified design process.

An FPGA (*Field-Programmable Gate Array*) device contains logical blocks and interconnection lines between them. The operations that are to be performed by the device are specified by configuration bits, which define the various blocks’ function and the interconnections between blocks [1]. The FPGA devices were originally designed as an alternative to gate arrays configured by masks, in order to be “programmable” by the user. Like processors, FPGA devices can be programmed after fabrication in order to solve any computational problem allowed by their hardware resources. This programmability differentiates processors and FPGA devices from application-specific functional units, which can perform only a function or a limited number of functions.

Today’s FPGA devices are suitable for implementing configurable architectures. This is possible due to the devices’ higher capacity and their more flexible interconnection structure. In the same time, the configuration time and the size of configuration streams has been reduced significantly.

This paper describes the design and implementation of a configurable processor called ConP using an FPGA circuit. The processor contains the kernel of a general-purpose processor and it can be extended in order to be used for specific application. This extension is accomplished by adding new instructions and functional units, without the change of the general-purpose kernel. By combining a classical processor and a configurable architecture in the same device is possible to exploit the advantages of both.

The organization of this paper is as follows. In the next Section we introduce the main aspects of configurable computing and configurable architectures. In Section 3 we present the main features of the ConP processor and its general structure. The implementation of this processor and the design tools used are described in section 4. The conclusions are presented in Section 5.

2. CONFIGURABLE COMPUTING

The main characteristic of traditional processors is that they can be programmed to solve virtually any computational task. The configurable architectures maintain the general nature of computing, but this computing is organized in a different manner. In traditional processors, operations are composed *temporally* by sequencing them in time, using registers or memory to store intermediate results (Figure 1). In configurable architectures, tasks are implemented by *spatially* composing primitive operators, that is, by linking them together by wires. In this way, parallel processing is possible, since operators and functional units can perform their operations simultaneously, which results in a considerable performance improvement.

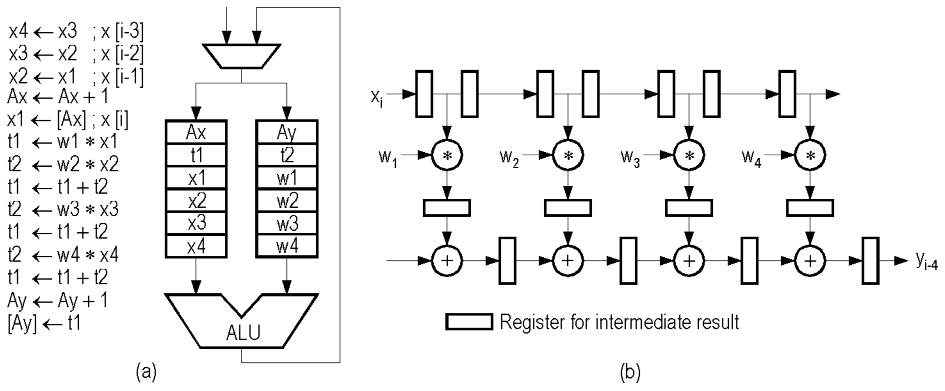
```
x4 ← x3   ; x [i-3]
x3 ← x2   ; x [i-2]
x2 ← x1   ; x [i-1]
Ax ← Ax + 1
x1 ← [Ax] ; x [i]
t1 ← w1 * x1
t2 ← w2 * x2
t1 ← t1 + t2
t2 ← w3 * x3
t1 ← t1 + t2
t2 ← w4 * x4
t1 ← t1 + t2
Ay ← Ay + 1
[Ay] ← t1
```

Figure 1. (a) Temporal and (b) spatial computations for the expression:
$y_i = w_1 * x_i + w_2 * x_{i-1} + w_3 * x_{i-2} + w_4 * x_{i-3}$.

To compare various architectures, DeHon introduced the *instruction depth*, which depends on the number of cycles required to perform a single iteration of a given task [3]. If a single instruction can be issued in each cycle for a complex task requiring many primitive operations (typical of scalar processors), a large number of instruction cycles are required to perform the task, which requires a deep instruction memory. If a task with low data dependencies and high throughput requirements is to be performed, it may require only a few instruction cycles on an array of spatial processing elements, and hence require only a shallow instruction memory.

In general, the functional and interconnection units of configurable architectures, such as FPGA devices, can store a single instruction, since they perform the same operation in each cycle. In this case, by instruction we mean the set of bits that control one cycle of the operation in the programmable device. Programmable architectures, such as processors, use deep instruction memory. Therefore, the instruction depth is an important characteristic that distinguishes configurable and programmable architectures.

The *computational density* represents the number of bit operations a device can perform per unit of area-time. A configurable device often can perform, in a single cycle, a computation that takes a processor hundreds of cycles. An FPGA device might require tens of cycles of latency to compute the first result, but because it performs the computation using a spatial pipeline composed of many active functional units, rather then sequentially with a small number of functional units, it achieves higher throughput.

Compared to traditional processors, the main reasons for the higher computational density of FPGA devices are the following:

- An FPGA device contains more active computing units in the same area as the processor, and therefore the degree of parallelism of the FPGA device is higher.
- FPGA devices can control operations at bit level, while processors can control their operators only at word level. As a result, processors often waste part of their computational capacity when operating on narrow-width data.

The disadvantage of configurable architectures is that they must perform the same operation in each cycle in order to take advantage of their high computational density. The programmable architectures contain relatively large cache memories for instructions and data, so that they can perform a large number of different operations with no decrease of their computational density.

Based on the previous observations, we can conclude that, for tasks that require high throughput and contain regular computations, the configurable architectures are more advantageous. For these tasks, the required throughput is obtained at lower cost than with programmable architectures. When a task contains a large number of computations that are not performed frequently, or when a high throughput is not required, it is better to use programmable architectures.

3. DESIGN OF THE CONFIGURABLE PROCESSOR

The ConP configurable processor consists of a constant part and an application-specific structure. The constant part allows to perform simple tasks without any change of the existing resources. The configurable part is defined by the user, based on the requirements of the specific application. This part consists of application-specific functional units that are controlled by special instructions. For the design of this processor, we used the VHDL hardware description language and the Active-HDL CAD system.

The main features of the ConP processor are the following:

- 16-bit external data bus and 12-bit address bus;

- Eight general-purpose registers: *R*0..*R*7;
- Three status flags: *S* (sign), *Z* (zero), *C* (carry);
- Load/store architecture;
- Direct, register-indirect, and immediate addressing modes;
- Stack memory organized in the internal memory;
- Expandable instruction set and addressing modes;
- I/O unit, that allow the addressing of 64 I/O registers and interrupt handling.

For the design of the ConP processor, we used a functional VHDL description. Therefore, the internal structure of the processor was generated by the synthesis system. A possible general structure of the processor is illustrated in Figure 2.

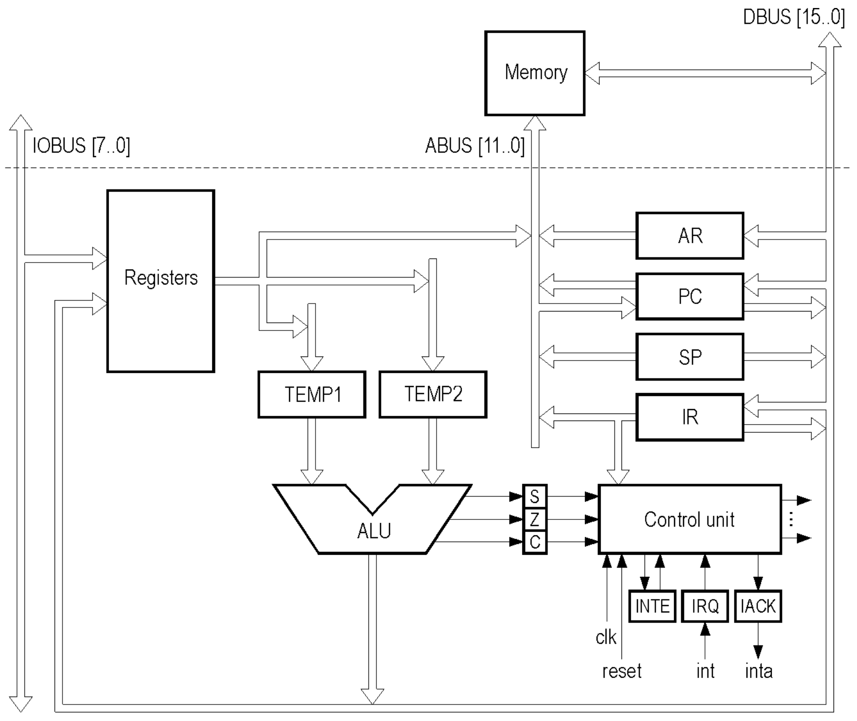

Figure 2. Block diagram of the ConP processor.

The processor contains three buses: a 16-bit data bus (*DBUS*), a 12-bit address bus (*ABUS*), and an 8-bit I/O bus (*IOBUS*). These buses are connected to the external pins of the processor. The I/O address space is isolated from the memory address space. The registers accessible by the programmer are the general-purpose registers *R*0..*R*7. Other registers, not accessible by program, are the program counter (*PC*), the stack pointer (*SP*), the instruction register (*IR*), the address register (*AR*), end two temporary registers (*TEMP*1, *TEMP*2). The ALU performs simple arithmetic and logical operations, as well as shift left, logical shift right, and arithmetic shift right. The control unit initiates and controls each transfer and data processing. All transfers are synchronized with the system clock.

For instruction execution, a number of clock cycles between 3 and 6 are required. These cycles are denoted by $T_0..T_5$. The instruction fetch requires two clock cycles, T_0 and T_1. Instruction decoding is performed in cycle T_2. The operations specified by the instruction are performed in one to three cycles. In each cycle, in addition to the operations specific to a particular cycle, other operations are performed that are needed to prepare for the next cycle.

4. IMPLEMENTATION OF THE CONFIGURABLE PROCESSOR

For the design and implementation of the ConP processor we used the VHDL language and synthesis software for FPGA devices. VHDL is the most used hardware description language standardized by the IEEE, so that it is supported by many design tools. After the simulation of the description with the Active-HDL VHDL simulator, we synthesized the description. For synthesis we used the Synopsys FPGA Express software, that generates a netlist from a VHDL description and then optimizes this netlist for the target FPGA device. The generated netlist is used to implement the processor in an FPGA device.

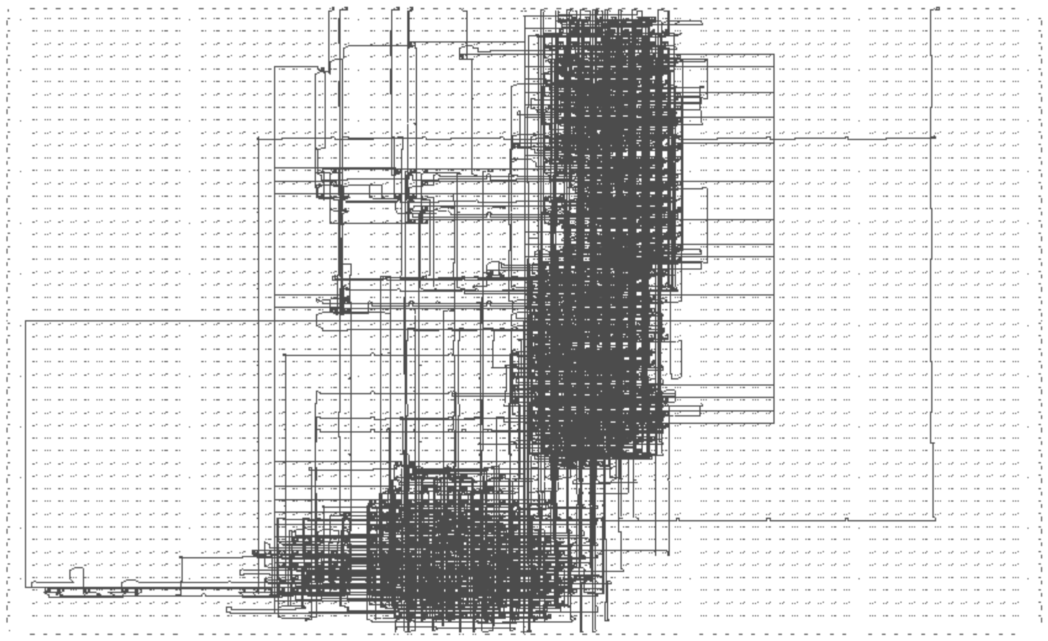

Figure 3. Layout of the ConP processor in the XCV600E FPGA device.

For implementation we used a Xilinx XCV600E FPGA device. This device contains 3456 (48x72) configurable logic blocks. The implementation was performed using the Xilinx Foundation Series software. From the netlist generated by logic synthesis the Foundation software performs the required operations for the configuration of the FPGA device and allows the simulation of the design based on the real signal delays. The layout of the ConP processor in the XCV600E FPGA device is shown in Figure 3.

Table 1. Main implementation parameters of the ConP processor.

Parameter	Value
Maximum delay on the interconnection lines	11.96 ns
Maximum clock frequency	52.42 MHz
Number of configurable blocks (CLB)	425
Number of flip-flops	576
Number of I/O blocks (IOB)	18
Number of equivalent gates	14,109

Table 1 presents the main implementation parameters of the ConP processor. As a synthesis option, we specified an operating frequency of 50 MHz. By changing the synthesis options, it is possible to increase this frequency. The processor uses about 12% of the FPGA device resources, and therefore it is possible to extend the instruction set with instructions that require complex logic.

5. CONCLUSIONS

In this paper we introduced the configurable computing, the configurable architectures, and then the design and implementation of the ConP configurable processor. This processor consists of a constant part and a configurable part. The constant part can be used to solve simple applications without changing the existing resources. The configurable part is defined by the user, based on the requirements of a specific application. This part contains application-specific functional blocks, controlled by special instructions. The integration of a classical processor and a configurable architecture within the same device allows to exploit the advantages of both architectures. Classical processors allow the highest flexibility, but their performance is lower than that of application-specific solutions. Application-specific circuits ensure the highest performance by sacrificing the flexibility.

For the design of the configurable processor we used the VHDL language. The implementation was performed using a Xilinx XCV600E FPGA device. This processor can be used in several types of applications: data encryption and compression, image processing, digital signal processing, special arithmetic.

REFERENCES

[1] Brown, S., Rose, J. [1996] FPGA and CPLD Architectures: A Tutorial, *IEEE Design & Test of Computers*, Summer, pp. 42–57.

[2] Callahan, T. J., Hauser, J. R., Wawrzynek, J. [2000] The Garp Architecture and C Compiler, *Computer*, Vol. 33, No. 4, pp. 62–69.

[3] DeHon, A. [2000] The Density Advantage of Configurable Computing, *Computer*, Vol. 33, No. 4, pp. 41–49.

[4] Ebeling, C., Cronquist, D. C., Franklin, P. [1996] RaPiD – Reconfigurable Pipelined Datapath. In *Proceedings of the 6th International Workshop on Field-Programmable Logic and Applications*, Springer-Verlag, pp. 126–135.

[5] Goldstein, S. C. et al. [2000] PipeRench: A Reconfigurable Architecture and Compiler, *Computer*, Vol. 33, No. 4, pp. 70–77.

[6] Guccione, S. A., Gonzales, M. J. [1995] Classification and Performance of Reconfigurable Architectures. In *Proceedings of the 5th International Workshop on Field-Programmable Logic and Applications*, Springer-Verlag, pp. 439–448.

[7] Haynes, S. D., Stone, J., Cheung, P. Y. K. [2000] Video Image Processing with the Sonic Architecture, *Computer*, Vol. 33, No. 4, pp. 50–57.

[8] Salcic, Z., Smailagic, A. [1999] Digital Systems Design and Prototyping Using Field Programmable Logic, Kluwer Academic Publishers.

[9] Wirthlin, M. J., Hutchings, B. L. [1995] DISC: A Dynamic Instruction Set Computer. In *Proceedings of the 4th IEEE Symposium of FPGAs for Custom Computing Machines FCCM '95*, Los Alamitos, IEEE CS Press, pp. 99–107.