Implementation of a RISC Processor Core for SoC Designs – FPGA Prototype vs. ASIC Implementation

Dominik Langen, Jörg-Christian Niemann, Mario Porrmann, Heiko Kalte, Ulrich Rückert
Heinz Nixdorf Institute, University of Paderborn
Paderborn, Germany
E-mail: niemann@hni.upb.de WWW: wwwhni.upb.de/sct

Abstract

In this paper, an implementation of a RISC processor core for SoC designs is presented. We analyze the differences between a prototypical FPGA implementation and standard cell realizations in an 0.6µm and an 0.13µm technology, respectively. The core was developed by using the hardware description language VHDL, which offers the opportunity of adding special, optimized hardware blocks for various operations. The effects on area and power consumption as well as computational power are analyzed. A detailed overview of the implementation of additional hardware multipliers and their effects on the above mentioned topics concludes this paper.

I. Introduction

Today's deep submicron fabrication technologies enable design engineers to implement an impressive number of components like microprocessors, memories, and interfaces in a single microchip. With the emergence of 100nm processes, hundreds of millions of transistors can be integrated on one die to form a parallel system, consisting of thousands of components. As upcoming SoC devices have to make more and more computing power available and, at the same time, power consumption and production of heat have to be restricted to a minimum, a parallel processing approach is a promising alternative for many applications. This means that, instead of one high-clocked and immensely power-consuming CPU, a number of lower clocked and more power efficient processing units are instantiated. This paper addresses the implementation of a RISC processor core that can be used as a single processor as well as a template for a multiprocessor approach for SoC designs. As mentioned above, most SoC architectures have strict requirements regarding the area of each module and power consumption. For this reason, we have chosen a well-suited architecture with small demands on both of these aspects. The Motorola M-Core architecture has sufficient performance for a broad variety of embedded applications. Therefore, we designed the S-Core in the hardware description language VHDL, a processor core that is binary compatible to the Motorola M-Core architecture. This gives us the opportunity to enhance the M-Core's capabilities by adding special hardware blocks and to make use of 11% unimplemented opcode space. This opens up the possibility for an easy application-specific enhancement of the core.

II. S-Core Architecture

For the use in System-on-Chip (SoC) designs, we have developed the S-Core, a RISC processor core that is binary-compatible with Motorola's M-Core M200 architecture. Due to this compatibility, we benefit from the broad variety of tools available for this architecture. The processor has been developed as a soft core using the hardware description language VHDL. This gives us the opportunity of using the core for different target technologies. Furthermore, the architecture can easily be expanded by adding application-specific instructions to the core.

The Motorola M-Core architecture has been chosen because it is quite resource-efficient and still delivers a reasonable performance. In an 0.36µm ASIC technology Motorola's implementation of the microprocessor core needs an area of 2.2mm². The complete microcontroller MMC2001 consumes only 0.41mW/MHz. The performance of up to 50 MIPS is sufficient for many typical tasks of embedded systems. Another reason for the choice of the M-Core is its straightforward architecture [1] [2]. The S-Core is a 32-bit RISC one-address machine with a load/store architecture. The architecture has two banks of 16 32-bit registers, which can be alternatively used in the user mode. Each opcode possesses a fixed length of 16 bits, and each instruction, except for the load and store instructions, works only on the registers. The execution of most opcodes takes one clock cycle.

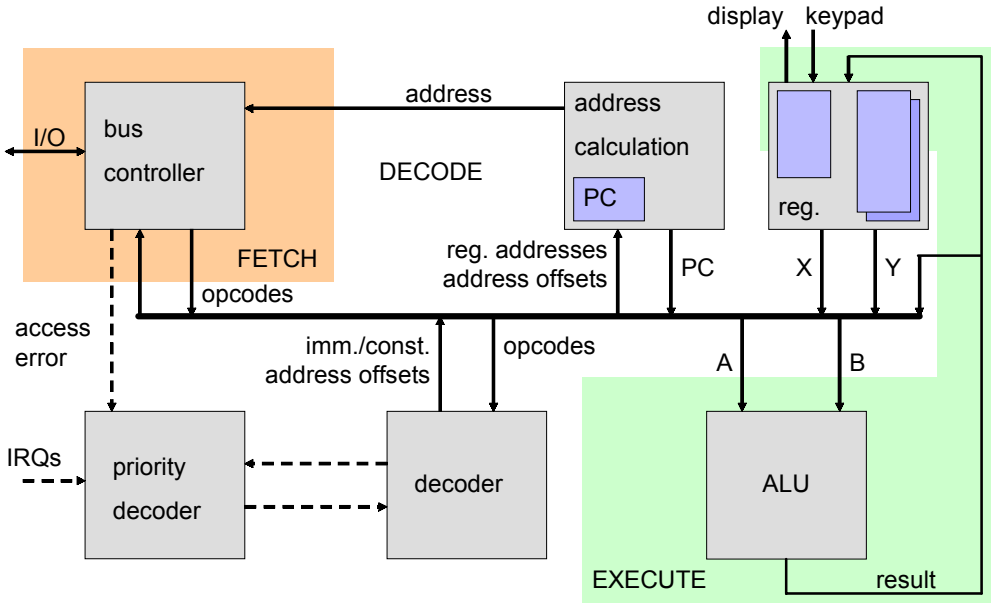

Figure 1: Architecture of the S-Core

Like Motorola's M200 architecture, our microprocessor core possesses a three-stage pipeline. The pipeline is controlled by the bus controller. If a transfer that takes more than one clock cycle is in progress the pipeline is being disabled. For an overview of the implemented architecture, see figure 1. For the software development the GNU C/C++ compiler, assembler and linker, which are available for the Motorola M-Core, are used to generate ELF (Executable and Linkable Format) binaries. In addition, we have developed a simulation environment that enables us to simulate the whole system. In the latter case, the binaries are used as image files for the memory models that are taken from the Synopsys SmartModels library.

III. Prototypical Implementation Using a Rapid Prototyping System

Especially when developing microelectronic systems, the time to market of a new product is a determinant of failure and success. Prototypical implementations help to convert ideas into products quickly and effectively. The Rapid-Prototyping System RAPTOR2000 (cf. figure 2), which has been developed by the Research Group for System and Circuit Technology, integrates all the important components to realize circuit and system designs with a complexity of up to 100 million transistors by means of FPGA-based application-specific modules (ASPs) [3] [4] [5]. The integration of new developments into a real system environment facilitates, besides a considerable acceleration and simplification of the tests, an increase in reliability and a significantly higher test coverage than would be possible with simulation tools only. The VHDL description of the S-Core has been mapped on a Xilinx Virtex FPGA that is integrated into the RAPTOR2000 system. The processor requires 3727 slices, i.e. 30% of a Xilinx Virtex 1000 FPGA. Currently, we are implementing various extensions of the processor core, ranging from instruction set extensions to additional components like timer and cache units. Using state-of-the-art FPGAs, it is even possible to implement on-chip multiprocessor architectures on one FPGA.

For this purpose, we have designed an AMBA bus interface that can be parameterized for different numbers of modules and that is automatically synthesized to any target technology [6].

Figure 2: Rapid Prototyping System RAPTOR2000

IV. ASIC Implementation

After the S-Core had been successfully tested by using the RAPTOR2000 environment, it has been fabricated in cooperation with Infineon Technologies AG (CPR ST, Prof. Ramacher) in an 0.13µm CMOS technology with four metal layers. Using a standard cell library, a maximum clock frequency of 160MHz has been achieved and the die size is 0.25mm². The core voltage of the ASIC implementation is 1.2V, leading to a power consumption of 0.165mW/MHz. As with the FPGA implementation, the ASIC implementation is embedded in the RAPTOR2000 system. Figure 3 depicts the S-Core testboard that can be used with the RAPTOR2000 system or as a stand-alone system with integrated I/O capabilities.

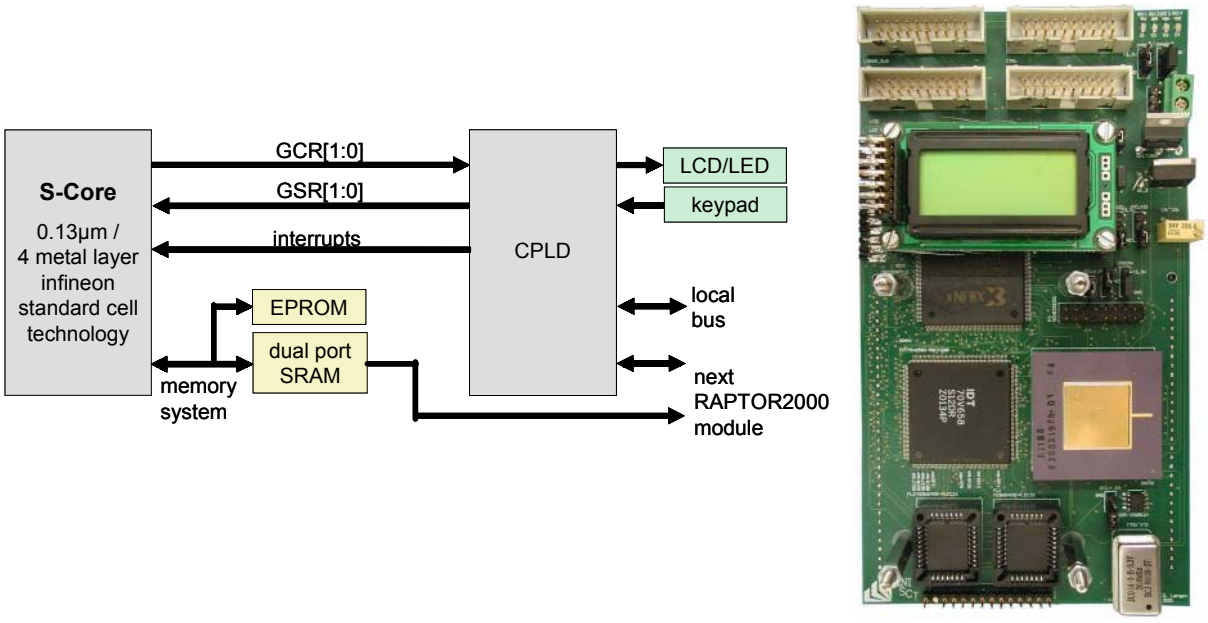

Figure 3: Test environment for the S-Core

In table 1, the performance and costs of the ASIC and FPGA implementation of the S-Core are compared. In addition to the 0.13µm Infineon technology, an 0.6µm AMS (Austria Micro Systems) technology is used for comparison. The basic version of the S-Core implemented in a XILINX Virtex 1000 FPGA (light grey highlight) serves as a reference in table 1. In this version, the 32 bit RISC CPU shows a performance of about 0.9MIPS/MHz. By adding additional optimized hardware blocks, the computing power of the CPU can be massively enhanced.

attributes	technology	voltage (V)	area	clock frequency (MHz)	power (mW/MHz)	power standardized (mW/MHz)	computing power		
							binary search	FIR32 filter	FIR16 filter
basic version, multiplication up to 18 cycles	Infineon 0.13 µm	1.2	0.25 mm² *(0.83%)*	160 *(1333%)*	0.165 *(0.66%)*	1.2 (5%)	1333%	1333%	1333%
	AMS 0.6 µm	5.0	**30 mm² *(100%)***	61 *(508%)*	20 *(80%)*	1.8 *(7%)*	508%	508%	508%
	FPGA Xilinx Virtex 1000-4	2.5	**3727 Slices *(100%)***	**12 *(100%)***	**25 *(100%)***	**25.0 *(100%)***	**100%**	**100%**	**100%**
multiplication in hardware in 2 cycles	AMS 0.6 µm	5.0	32 mm² *(107%)*	57 *(475%)*	20 *(80%)*	1.8 *(7%)*	475%	1553%	1340%
	FPGA Xilinx Virtex 1000-4	2.5	4345 Slices *(116%)*	12 *(100%)*	25 *(100%)*	25.0 *(100%)*	100%	327%	282%
additional MAC function	AMS 0.6 µm	5.0	32 mm² *(107%)*	58 *(483%)*	20 *(80%)*	1.8 *(7%)*	483%	2001%	1363%
	FPGA Xilinx Virtrex 1000-4	2.5	4499 Slices *(120%)*	12 *(100%)*	27.5 *(110%)*	27.5 *(110%)*	100%	414%	282%
additional MAC function and configurable data width	AMS 0.6 µm	5.0	33 mm² *(110%)*	51 *(425%)*	20 *(80%)*	1.8 *(7%)*	425%	1762%	1794%
	FPGA Xilinx Virtex 1000-4	2.5	4790 Slices *(129%)*	11 *(110%)*	27.5 *(110%)*	27.5 *(110%)*	92%	380%	387%

Table 1: Comparison of the ASIC and FPGA implementation of the S-Core

The table shows that the modification of the S-Core's ALU can lead to a significant increase in computational power for certain applications. In the basic version, the S-Core needs up to 18 cycles for one multiplication. The multiplication is based on a Booth 2 algorithm [7]. The partial products are generated and added sequentially. For the addition of the partial products, the built-in adder of the ALU is used. We have exchanged the sequential multiplier of the basic version for a parallel hardware multiplier. This multiplier uses the same Boot 2 algorithm as the serial one, but the partial products are generated and added by a carry-save adder tree in one clock cycle (figure 4).

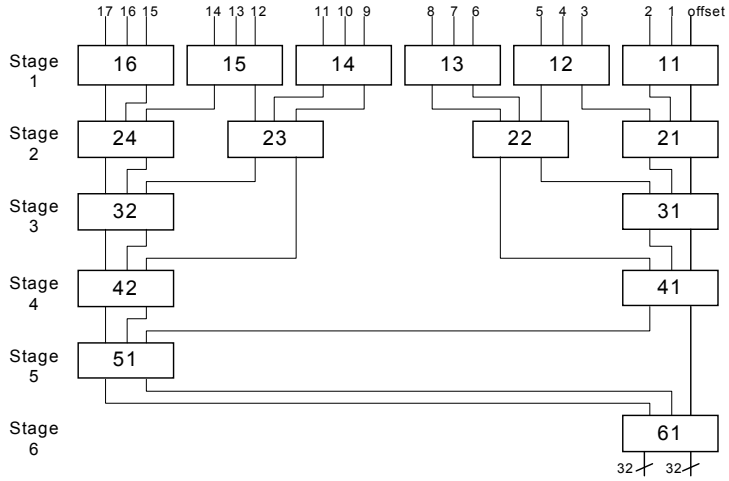

Figure 4: Carry-save adder tree for an S-Core hardware multiplier for a 32-bit datapath

Each rectangle symbolizes a carry-save adder. For this design, the input "offset" is hardwired to zero. In a second clock cycle, the two sums that come out of the carry-save adder tree are added

by the built-in adder of the ALU. Hence, for performing the multiplication in special hardware, only 2 cycles are needed. The processor modified in this way has been mapped to an AMS 0.6µm standard cell technology as well as to a Xilinx Virtex FPGA. For the standard cell technology, this modification leads to an increase of area by 7% and a drop of clock frequency by 7%. Consequently, applications that do not use the multiplier suffer from a 7% decrease in performance, but multiplication-intensive applications like FIR (finite impulse response) filters can be accelerated substantially. We have measured a 2-tab FIR filter with a precision of 16 and 32 bits. The hardware multiplier raises the performance by 306% and 264%, respectively. For the FPGA realization, no impact on the clock frequency could be noticed. However, the additional area consumption on the FPGA is 16%.

The performance of the multiplication unit can be improved further by integrating MAC (multiply-accumulate) capabilities. The input "offset" (cf. figure 4) is used to feed the value of an accumulator into the carry-save adder tree. For the AMS standard cell technology, this modification has almost no impact on area and clock frequency, but the performance increases by 394% and 268% for the FIR filters compared to the basic version. The FIR 16-bit filter cannot take as much advantage of the fast MAC-operation as the 32-bit filter as there are a couple of other operations necessary to adjust the results of the multiplication to 16-bit width. For the FPGA, the realization with MAC capabilities needs 4% more area than the version with the ordinary hardware multiplier.

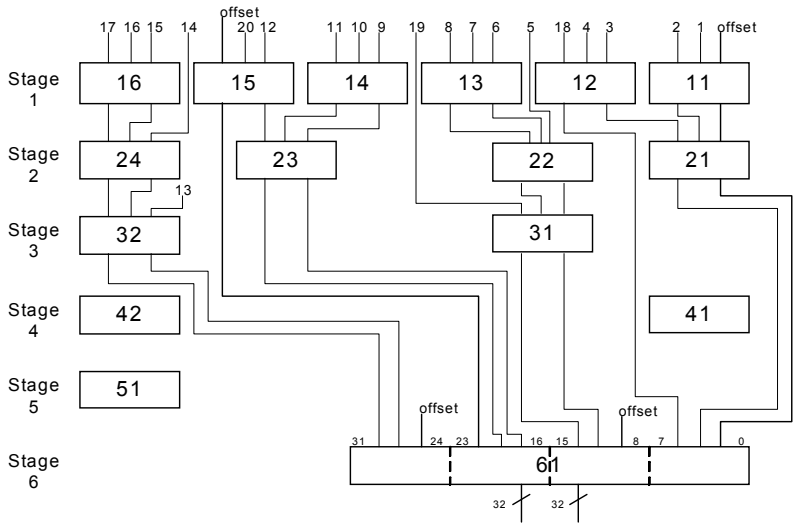

Figure 5: Carry-save adder tree for an S-Core hardware multiplier for a 4×8-bit datapath

A further modification addresses the problem of efficiently handling numbers with a low precision in a wide datapath, e.g. 8-bit and 16-bit numbers in a 32-bit datapath. These extensions are comparable to Intel's MMX extensions [8]. The extended ALU can add and multiply 1×32-bit numbers, 2×16-bit numbers, or 4×8-bit numbers at the same time (cf. figure 5). For the standard cell technology, the extensions increase the area consumption by another 3% compared to the ALU with MAC capabilities. For an FPGA, the additional area increase is 9%. The clock frequency drops for the standard cell technology by 16% and for the FPGA implementation by 8% compared to the basic version. Compared to the ALU with MAC capabilities, the 16-bit FIR filter can be accelerated by additional 85% (ASIC) and 105% (FPGA).

V. Conclusion

In this paper, we have presented an implementation of the S-Core RISC processor for SoC designs. After the prototypical FPGA realization, which was very useful for design verification and optimization, an ASIC has been fabricated in an 0.13µm Infineon standard cell technology. A comparison to an AMS 0.6µm technology realization has demonstrated the improvements in area and power consumption as well as a remarkable increase of the maximum clock frequency. We have shown that the implementation of special hardware blocks is a powerful method of increasing the computational power of the core. The area and power requirements caused by these increases in computational power are almost insignificant. As far as the area used by the S-Core is concerned, it becomes obvious that this processing unit is well fitted for SoC designs and is, beyond that, a good choice for a multiprocessor implementation. As for the 11% of free opcode space, the designer has enough architectural headroom for specific hardware-accelerated functions. This qualifies our core for computation-bound applications. In future, we are going to realize a SoC device with several S-Core units that will form a multiprocessor.

Acknowledgement

We would like to thank the Infineon Technologies AG and especially the department CPR ST, Prof. Ramacher, for their encouragement and their support of the ASIC fabrication.

References

[1] Motorola: M-Core Reference Manual, 1998.

[2] Motorola: MMC2001 Reference Manual, 1998.

[3] M. Porrmann, H. Kalte, U. Witkowski, J.-C. Niemann and U. Rückert: "A Dynamically Reconfigurable Hardware Accelerator for Self-Organizing Feature Maps". Proceedings of The 5th World Multi-Conference on Systemics, Cybernetics and Informatics, SCI 2001, Volume 3, pages 242-247, Orlando, Florida, USA, July 2001.

[4] Kalte, H.; Porrmann, M.; Rückert, U.: "Rapid Prototyping System für dynamisch rekonfigurierbare Hardwarestrukturen", AES2000, Karlsruhe, 18.-19. Jan. 2000, S. 150-157.

[5] Kalte, H., Porrmann, M., Rückert, U.: "Using a Dynamically Reconfigurable System to Accelerate Octree Based 3D Graphics", Tagungsband: ´International Conference on Parallel and Distributed Processing Techniques and Applicatons´ (PDPTA´2000), Vol. 5, 26-29 June, 2000, Monte Carlo Resort, Las Vegas, Nevada, USA, S. 2819- 2824.

[6] H. Kalte, D. Langen, E. Vonnahme, A. Brinkmann, U. Rückert: "Dynamically Reconfigurable System-on-Programmable-Chip", Proc. of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (PDP 2002), pp. 234-242, January 9th-11th, 2002, Gran Canaria Island, Spain.

[7] I. Koren: "Computer Arithmetic Algorithms", Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[8] A. Peleg, U. Weiser: "MMX Technology Extensions to the Intel Architecture", IEEE Micro, Vol. 16, No. 4, August 1996, pp. 42-50.