Adv. Radio Sci., 6, 259–264, 2008
www.adv-radio-sci.net/6/259/2008/

Advances in
Radio Science

Hardware Accelerators for Elliptic Curve Cryptography

C. Puttmann[1], J. Shokrollahi[2], M. Porrmann[1], and U. Rückert[1]

[1]Heinz Nixdorf Institute, University of Paderborn, Germany
[2]Chair for System Security, Ruhr-University of Bochum, Germany

Abstract. In this paper we explore different hardware accelerators for cryptography based on elliptic curves. Furthermore, we present a hierarchical multiprocessor system-on-chip (MPSoC) platform that can be used for fast integration and evaluation of novel hardware accelerators. In respect of two application scenarios the hardware accelerators are coupled at different hierarchy levels of the MPSoC platform. The whole system is implemented in a state of the art 65 nm standard cell technology. Moreover, an FPGA-based rapid prototyping system for fast system verification is presented. Finally, a metric to analyze the resource efficiency by means of chip area, execution time and energy consumption is introduced.

1 Introduction

Data security is an important requirement for many applications in our daily life. Especially internet applications such as e-commerce need to transmit secret data via insecure communication channels. Therefore, various cryptographical methods exist that allow to protect sensitive data. Asymmetric cryptography, which is also known as public-key cryptography, does not only provide algorithms for encryption and decryption of data, but also for digital signatures and authentication. Recently asymmetric cryptography based on elliptic curves is gaining interest. Compared to traditional asymmetric techniques, e.g. the RSA algorithm, the elliptic curve cryptography (ECC) achieves an equivalent level of security with smaller key sizes. Using elliptic curve cryptography therefore results in memory as well as bandwidth savings.

Nevertheless, computational intensive operations emerge during the processing of ECC protocols. The scalar multiplication on elliptic curves represents a frequently required and complex operation. The acceleration of this task by dedicated hardware units can not only speed up the execution time, but also help to save energy. In this paper, we propose different types of hardware accelerators for scalar multiplication in respect of two application scenarios. On one hand, we consider mobile devices with limited power resources such as smartcards. On the other hand, security servers are considered, where high performance is more important than power consumption and costs in terms of chip area.

Our work presents various hardware accelerators for elliptic curve cryptography, which are evaluated by integration into a multiprocessor system-on-chip (MPSoC). This proposed evaluation platform is scalable and can be adapted to suit different application scenarios. Dedicated hardware blocks can easily be integrated at different hierarchy levels of the MPSoC, which allows a comfortable evaluation of novel accelerators. Furthermore, the whole system is mapped onto an FPGA-based rapid prototyping environment in order to speed up simulation and verification.

The proposed hardware accelerators are synthesized in a modern 65 nm standard cell technology and analyzed regarding their resource efficiency in terms of chip area, execution speed and power consumption. Furthermore, a metric is introduced to compare the resource efficiency of different implementations in respect of the considered application scenario.

The paper is structured as follows. Section 2 gives a brief overview of the mathematical background of elliptic curve cryptography. In Sect. 3 the evaluation platform that is used to integrate and verify the hardware accelerators is discussed. The proposed hardware accelerators as well as the metric for analyzing their resource efficiency are explained in detail in Section 4. Finally, Sect. 5 concludes the paper.

2 Elliptic Curve Arithmetic

In this section the elliptic curve cryptography based on binary field arithmetic is introduced. The general equation for a non-supersingular elliptic curve E over the binary finite field $\mathbb{F}_{2^m}$ is given by equation:

$$E : y^2 + xy = x^3 + ax^2 + b \qquad (1)$$

Correspondence to: C. Puttmann
(puttmann@hni.upb.de)

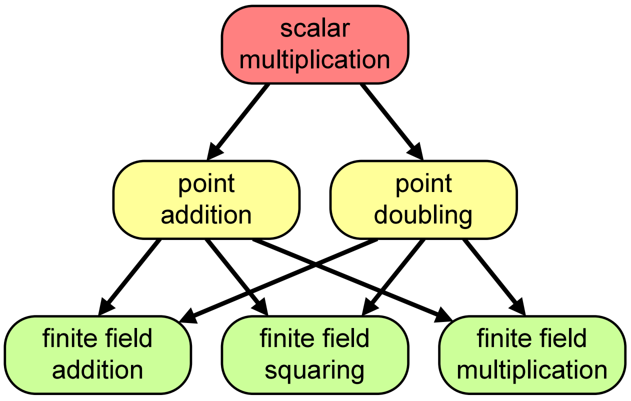

Fig. 1. Arithmetic hierarchy of elliptic curve cryptography.

for appropriate parameters $a, b \in \mathbb{F}_{2^m}$. The set of points $(x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$, which satisfy Eq. (1), together with the identity element $\mathcal{O}$, generate an additive abelian group. Let $\mathcal{P}=(x_p, y_p)$ and $\mathcal{Q}=(x_q, y_q)$ be two given points on the curve in Eq. (1). The point addition $\mathcal{P}+\mathcal{Q}$ as well as the point doubling $\mathcal{P}+\mathcal{P}$ are two operations defined on the elliptic curve E, which can geometrically be represented by the tangent and chord operation, respectively. By applying the point addition and point doubling operations, we are able to multiply an integer k with a point $\mathcal{P}$, which is the result of $k-1$ times adding the point $\mathcal{P}$ to itself. This operation is known as scalar or point multiplication $k \times \mathcal{P}$. Figure 1 depicts this hierarchical structure of arithmetic operations used for elliptic curve cryptography over finite fields.

2.1 Projective Coordinates

Naturally, point addition and point doubling also require a field inversion when using affine coordinates (x, y). Since inversion is a very expensive operation compared to multiplication, addition and squaring in finite fields, we use projective coordinates. In standard projective coordinates the points on the elliptic curve are represented as a triple (X, Y, Z) in such a way that $x \rightarrow X/Z$ and $y \rightarrow Y/Z$. By using projective coordinates only one finite field inversion is required at the end of a scalar multiplication in order to transform the projective coordinates back to affine coordinates.

2.2 Montgomery Ladder Algorithm

In Montgomery (1987) a very efficient method to perform the scalar multiplication is presented, which was applied to elliptic curve cryptography by López and Dahab (1999). The method is known as montgomery ladder and is shown at point level in Algorithm 1. Since in every loop iteration the same operations are performed, namely one point addition and one point doubling, the montgomery ladder algorithm is shielded against timing attacks and simple power analysis attacks.

Algorithm 1 The Montgomery ladder algorithm for scalar multiplication expressed at point level.

```
Input: A point $\mathcal{P}$ on the elliptic curve $E$, together with the binary
       representation of the scalar $k$ as $(k_{i-1}k_{i-2}\dots k_1k_0)_2$.
Output: $k \cdot \mathcal{P}$
  $\mathcal{P}_1 \leftarrow \mathcal{P}, \mathcal{P}_2 \leftarrow 2\mathcal{P}$
  for $j$ from $i-2$ downto $0$ do
     if $k_j = 1$ then
        $\mathcal{P}_1 \leftarrow \mathcal{P}_1 + \mathcal{P}_2, \mathcal{P}_2 \leftarrow 2\mathcal{P}_2$
     else
        $\mathcal{P}_2 \leftarrow \mathcal{P}_1 + \mathcal{P}_2, \mathcal{P}_1 \leftarrow 2\mathcal{P}_1$
     end if
  end for
```

2.3 Polynomial Basis Representation

As illustrated in Fig. 1, finite field arithmetic represents the base operations. Therefore, an efficient representation of the finite field elements in $\mathbb{F}_{2^m}$ is important. The polynomial basis representation can be described as a vector space of dimension m over the field $\mathbb{F}_2$ and is one of the most common representations in ECC. Field elements in polynomial basis representation are expressed as binary polynomials of degree $m-1$ as follows:

$$a(x) = \sum_{i=0}^{m-1} a_i \times x^i \quad \text{with} \quad a_i \in \{0, 1\} \tag{2}$$

One benefit of binary fields is that the finite field addition is calculated by a carry-free XOR operation of corresponding coefficients. Finite field squaring can be achieved by shifting each bit a_i to a_{2i} and filling the gaps with zeros. Similar to finite field multiplication, the result is a binary polynomial of degree $2m-2$, which has to be reduced modulo an irreducible, sparse polynomial of degree m. However, the finite field multiplication is equivalent to the product of the corresponding polynomials, which is, compared to addition and squaring, the most computational intensive operation.

2.4 Karatsuba Multiplication Method

In order to reduce the complexity of polynomial multiplication, the method of Karatsuba and Ofman (1963) is applied. Whereas "classically" the coefficients of the product $(a_1x+a_0)(b_1x+b_0)=a_1b_1x^2+(a_1b_0 \oplus a_0b_1)x+a_0b_0$ from the four input coefficients a_1, a_0, b_1, and b_0 are computed with 4 multiplications and 1 addition, the Karatsuba formula uses only 3 multiplications and 4 additions in binary fields:

$$\begin{aligned}&(a_1x + a_0)(b_1x + b_0) = \\ &a_1b_1x^2 + ((a_1 \oplus a_0)(b_1 \oplus b_0) \oplus a_1b_1 \oplus a_0b_0)x + a_0b_0.\end{aligned} \tag{3}$$

By applying the Karatsuba method to larger polynomials the costs of extra additions vanish compared to those of the

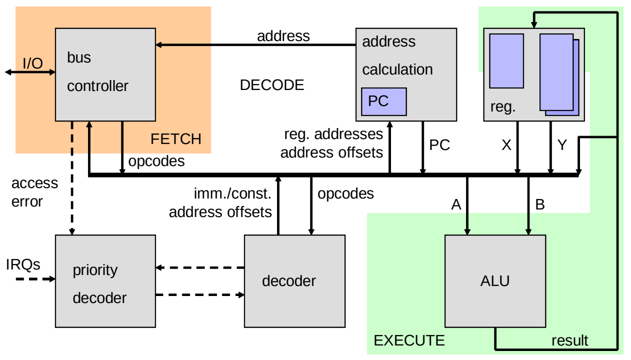

Fig. 2. N-Core processor architecture providing a three-stage pipeline.

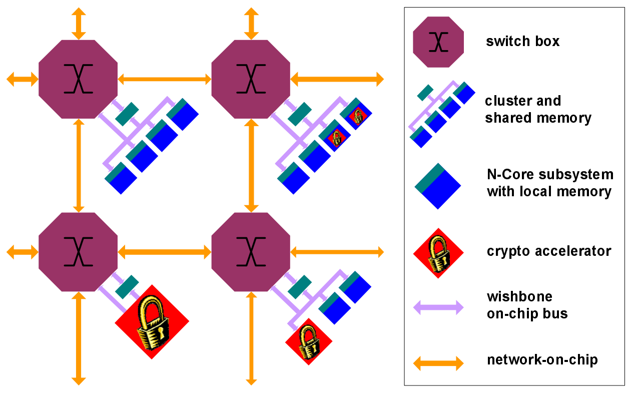

Fig. 3. Integration of hardware accelerators at different levels of our MPSoC-based evaluation platform.

saved multiplications and an asymptotical cost of $O(m^{1.59})$ compared to the classical cost of $O(m^2)$ is achieved (Hankerson et al., 2004).

3 Evaluation Platform

In Niemann et al. (2007) we presented the development of a scalable and hierarchical multiprocessor system-on-chip (MPSoC) architecture. This system architecture is used here as an evaluation platform for the integration of hardware accelerators.

3.1 N-Core Processor Element

The core processing element is our 32-bit RISC microprocessor named N-Core (Niemann et al., 2007). The N-Core comprises a common load-store architecture that is depicted in Fig. 2. The processor provides a three-stage pipeline, which performs instruction *fetch*, *decode* and *execute*. Two independent banks of sixteen 32-bit registers allow fast program context switching, e.g. for fast interrupt handling. Although the N-Core provides a 32-bit data bus, instructions have a fixed width of 16 bit, which delivers a high code density. Due to 11% free opcode space, instruction set extensions can be added to the microprocessor.

3.2 Hierarchical MPSoC Architecture

The processor core is extended with local data and instruction memory, a programmable interrupt controller (PIC) and a timer module to form a *N-Core subsystem*. The N-Core subsystem represents the lowest level of hierarchy within the MPSoC system. At the next higher level, a parametrizable amount of N-Core subsystems are connected via an on-chip Wishbone bus. Together with a shared memory this multiprocessor arrangement forms a *cluster*. At cluster level the processor elements can send messages directly to each other via the Wishbone bus as well as exchange data via the common shared memory. In this way, applications can be parallelized and processed cooperatively by the N-Core subsystems. At the top level of hierarchy, several clusters are connected through switch boxes over the network-on-chip (NoC). The NoC communication structure is described in Puttmann et al. (2007) and features packet-switched wormhole routing with up to 42 GBit/s data throughput.

3.3 Coupling of Hardware Accelerators

Depending on the requirements of the application scenario, hardware accelerators can be coupled at different levels of hierarchy, resulting in several resource utilization variants. As depicted in the top right cluster of Fig. 3, hardware accelerators in terms of instruction set extensions can be added to the N-Core subsystem. This integration means a very close coupling to the processor with little hardware overhead. In the bottom right cluster of Fig. 3, the accelerator is connected to the local on-chip bus of the cluster. In this way, each processor of the corresponding cluster can access the hardware accelerator. Moreover, the direct coupling to a switch box (see bottom left cluster in Fig. 3) results in high data throughput due to the close connection to the network-on-chip. Here, every processor of the whole system can access the hardware accelerator. In order to easily integrate the same hardware unit at different levels, we developed wrapper modules that automatically provide an interface for the desired coupling (Niemann et al., 2007).

3.4 Rapid Prototyping Environment

Depending on the software program under test, the simulation of a complex multiprocessor system-on-chip can take very long. Our approach to speed up the system verification is based on fast hardware emulation instead of comparable slow software simulation. For this purpose, we have developed an FPGA-based rapid prototyping environment called RAPTOR2000 (Kalte et al., 2002). As depicted in Fig. 4, the prototyping system consists of the RAPTOR2000

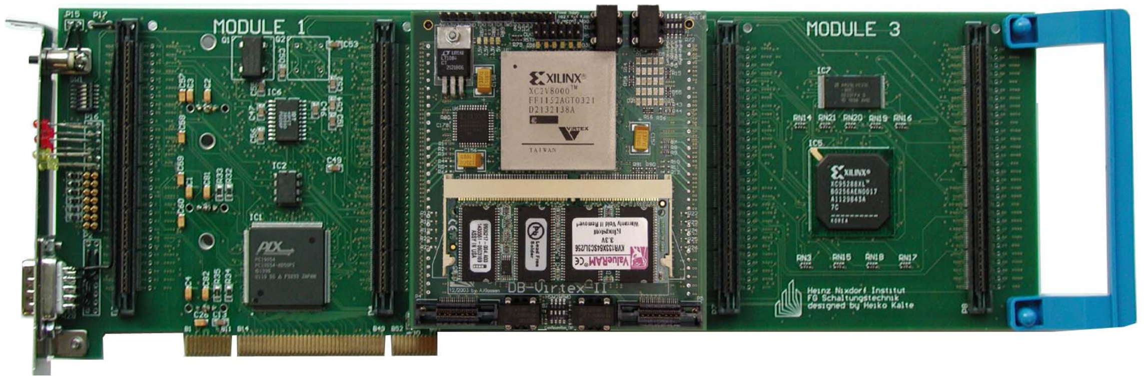

Fig. 4. Fast emulation of new hardware accelerators by using our FPGA-based rapid prototyping system RAPTOR2000.

Table 1. Synthesis results for a 65 nm standard cell technology and a Xilinx Virtex-II FPGA implementation, respectively.

	ASIC		FPGA	
	area	*power*	*utilization*	
	[mm²]	[mW]	[slices]	[RAM16]
N-Core processor	0.078	17.20	3206	0
N-Core subsystem[1]	0.613	45.16	3662	16
Cluster[2]	3.443	357.14	15362	80
Switch box	0.449	315.12	14133	0

[1] N-Core processor, 32 KB local memory, interrupt controller, timer module
[2] 4 N-Core subsystems, 32 KB shared memory, on-chip wishbone bus

motherboard, which can be equipped with up to six application specific daughterboard modules. Based on FPGA daughterboard modules, RAPTOR2000 is able to emulate circuits with a complexity of more than 200 million transistor gates. The host computer can communicate via a PCI bus interface with the RAPTOR2000 board and each attached daughterboard, respectively. Additionally, we have developed daughterboard modules that provide various communication standards. For example, an ethernet daughterboard can be used to stimulate the design under test with real world network traffic.

The implementation result of the MPSoC core components are shown in Table 1. We mapped the system to a 65 nm technology for ASIC fabrication as well as to a Xilinx XC2V8000-4 FPGA for rapid prototyping. The ASIC implementation is based on a low power (regular threshold voltage) standard cell library under typical operating conditions. The N-Core subsystem runs at a frequency of 400 MHz, which is also the operating frequency of a cluster. However, the network-on-chip as well as the switch box achieves a maximum clock frequency of 718 MHz. Each XC2V8000-4 FPGA can host one cluster including 4 N-Core subsystems and a switch box, which runs at 12.5 MHz. The system can be easily scaled by using more than one FPGA daughterboard (Niemann et al., 2007).

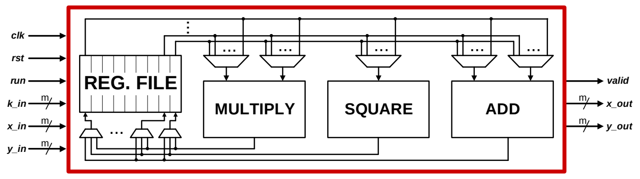

Fig. 5. Architecture of the high performance hardware accelerator for scalar multiplication.

4 Hardware Accelerators

In this section, various hardware accelerators for elliptic curve cryptography are presented. As explained in Sect. 2, we concentrate on the scalar multiplication in binary fields. Throughout the rest of this paper, we refer to the binary field $\mathbb{F}_{2^{233}}$, which is one of the recommended extension fields by the National Institute of Standards and Technology (2000). The hierarchy level for integration into the evaluation platform is chosen in respect to the application scenario. Furthermore, a metic for analyzing the resource efficiency is proposed and exemplarily applied to select the best hardware accelerator for a given application scenario.

4.1 High Performance Scenario

High performance hardware accelerators are required for systems such as internet servers, which have to handle many secure data transactions at one time, e.g. for online-banking applications. Fast computation and high data throughput are more important in this application scenario than power consumption and costs in terms of chip area.

We have developed a hardware accelerator for this application scenario, whose architecture is shown in Fig. 5. The hardware accelerator provides a modular structure comprising arithmetic units for finite field multiplication, squaring and addition (cf. Fig. 1). Addition as well as squaring is performed in one clock cycle, whereas multiplication

Table 2. Resource utilization for 65 nm standard cell technology at 200 MHz clock frequency and typical operating conditions.

	chip area		power consumption		execution time	
	absolute	relative	absolute	relative	absolute	relative
	[µm²]	[%]	[mW]	[%]	[ms]	[%]
N-Core	57190	0.00	9.23	0.00	57.59	0.00
N-Core (ISE1)	57519	0.58	9.49	2.82	50.92	-11.58
N-Core (ISE2)	59290	3.67	7.35	-20.37	26.42	-54.12
N-Core (ISE3)	64734	13.19	12.00	30.01	20.19	-64.94

based on the Karatsuba method requires 3 clock cycles in a pipelined fashion. Furthermore, the register file can store 7 finite field elements, which is sufficient for calculating the scalar multiplication. Due to the generic architecture, the hardware accelerator can easily be adapted to different field sizes. The scalar multiplication is calculated using the montgomery ladder algorithm as explained in Sect. 2. Moreover, the hardware accelerator handles coordinate transformation, i.e. affine coordinates are supported as input and output format, while internally projective coordinates are used for calculation.

The proposed hardware accelerator is coupled to a switch box port at the highest hierarchy level of the evaluation platform (cf. Fig. 3) for several reasons. At this level, the highest data throughput is achieved due to the close coupling to the network-on-chip. Furthermore, the hardware accelerator can operate autonomously without the need for an attached microprocessor.

Again, the hardware accelerator was synthesized using a 65 nm standard cell technology with typical operating conditions. A total chip area of 0.279 mm^2 is required to support the binary field $\mathbb{F}_{2^{233}}$. The hardware accelerator achieves a clock frequency of 625 MHz, i.e. the complete scalar multiplication is calculated in 7.2 μs and consumes 72.8 mW power. Respectively, 15365 slices are needed on a Xilinx XC2V8000-4 FPGA, which runs with 50 MHz clock frequency, resulting in 90 μs calculation time.

4.2 Low Power Scenario

In contrast to the high performance scenario, we now consider low power devices like smartcards. These are commonly used in mobile devices with limited energy resources and restricted chip area. Instead of high performance, the motivation for hardware accelerators here is to save energy with little increase of chip area. In order to reach this aim, we analyzed several types of instruction set extensions (ISE). Regarding the evaluation platform, instruction set extensions are directly implemented to the N-Core processor and hence coupled at the lowest hierarchy level. For a smartcard scenario, the evaluation platform is scaled down to a minimum of only one single N-Core comprising instruction set extensions.

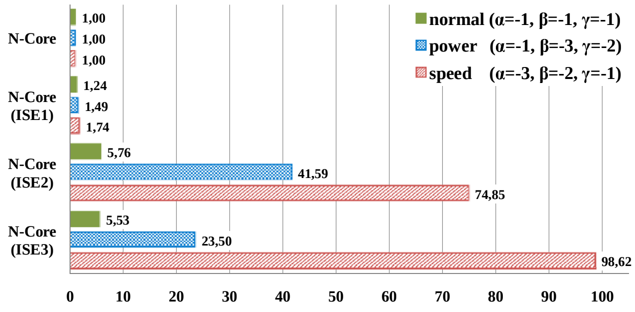

Fig. 6. Resource efficiency of different types of instruction set extensions.

In Puttmann et al. (2008), we present a two-stage framework for automatic instruction set extension. As a result, we implemented three different instruction set extensions. The first type of ISE combines already existing instructions, which are frequently used, to new *super instructions*. The analysis of the scalar multiplication showed that the instruction set of the N-Core processor can benefit from combinations of *shift* and *xor*. Therefore, we added three super instructions to the N-Core, namely `LSRIxor`, `LSLIxor` and `LSRIandadd`. This modification is in the following referred to as *ISE1*. Whereas ISE1 only combines existing instructions to new super instructions, the second type of instruction set extension (referred to as *ISE2*) introduces a new instruction, called `MULGF2`. This instruction multiplies two 32-bit binary field polynomials by using existing functions of the arithmetic logical unit (ALU). Multiplication with ISE2 is realized bit by bit utilizing the existing shift unit and xor function of the ALU. The third type of instruction set extension (*ISE3*) also supports this `MULGF2` instruction, but a dedicated hardware unit for binary field multiplication is implemented instead of using existing ALU functions.

Table 2 shows the absolute resource utilization as well as the relative changes compared to an original N-Core processor. The synthesis results are based on high threshold voltage standard cells, since we are considering a low power scenario. Therefore, the clock frequency of the N-Core is reduced to 200 MHz and only a chip area of 0.057 mm^2 is required. The execution time in Table 2 refers again to a scalar multiplication in $\mathbb{F}_{2^{233}}$.

4.3 Resource Efficiency

In order to select the best hardware accelerator for a given application scenario, a metric for resource efficiency is introduced in this section. The resource efficiency of the three variants of instruction set extensions is exemplarily analyzed. Therefore, we define resource efficiency (RE) as

$$RE = T^{\alpha} \times E^{\beta} \times A^{\gamma} \tag{4}$$

with *execution time* (T), *energy consumption* (E) and *chip area* (A). Each resource component can be weighted by an exponent (α, β, γ) to characterize the focus of the considered application scenario.

Figure 6 shows the resource efficiency of the three ISE variants (cf. Table 2), which are normalized to the resource efficiency of the original N-Core processor. We characterized three resource weightings, namely *normal*, *power* and *speed*. The exponents α, β and γ are all negative, so that a better utilization of any resource results in a better resource efficiency. The solid bars depict the resource efficiency with each component equally weighted, i.e. chip area, energy consumption and execution time are of the same importance. However, for the smartcard application scenario, we focus on low power implementations with little hardware overhead. Therefore, the checkered bars depict the resource efficiency, where energy consumption ($\beta = -3$) is more important than chip area ($\gamma = -2$), which in turn is more important than execution time ($\alpha = -1$). Therefore, ISE2 would be the best suited implementation for this application scenario, since it achieves the highest resource efficiency with emphasis on power. In contrast, the striped bars show the resource efficiency when focusing on speed. Consequently, the execution time ($\alpha = -3$) is here more important than the energy consumption ($\beta = -2$), which again is more important than the chip area ($\gamma = -1$). In this case, ISE3 represents the best suited hardware accelerator, because it achieves the highest resource efficiency with respect to execution speed.

5 Conclusions

In this paper we have presented various hardware accelerators for cryptography based on elliptic curves. The proposed scalable multiprocessor system-on-chip evaluation platform allows an easy integration of hardware accelerators at different levels of hierarchy. In order to speed up the verification of novel hardware accelerators, the evaluation system can be emulated by using our FPGA-based rapid prototyping environment RAPTOR2000. A high performance hardware accelerator was developed, which calculates a scalar multiplication in $\mathbb{F}_{2^{233}}$ in 7.2 μs. Moreover, different types of instruction set extensions were introduced that provide energy savings up to 63%. Finally, a metric for analyzing the resource efficiency was defined, which helps to select the best suitable hardware accelerator for a given application scenario.

Acknowledgements. The research described in this paper was funded in part by the Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung – BMBF), registered there under grant number 01AK065F (NGN-PlaNetS), and by the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG) under project RU 477/8.

References

Hankerson, D., Menezes, A. J., and Vanstone, S. A.: Guide to Elliptic Curve Cryptography, Springer Professional Computing, Springer, Berlin, 2004.

Kalte, H., Porrmann, M., and Rückert, U.: A Prototyping Platform for Dynamically Reconfigurable System-on-Chip Designs, in: Proceedings of the IEEE Workshop Heterogeneous reconfigurable Systems-on-Chip (SoC), Hamburg, Germany, http://www.raptor2000.de, 2002.

Karatsuba, A. A. and Ofman, Y.: Multiplication of Multidigit Numbers on Automata, Soviet Physics Doklady, 7, 595-596, 1963.

López, J. and Dahab, R.: FastMultiplication on Elliptic Curves over GF(2m) without Precomputation, in: CHES 99: Proceedings of the First InternationalWorkshop on Cryptographic Hardware and Embedded Systems, Springer-Verlag, London, UK, 316-327, 1999.

Montgomery, P. L.: Speeding the Pollard and Elliptic Curve Methods of Factorization, Math. Comput., 48, 243-264, 1987.

National Institute of Standards and Technology (NIST): Digital Signature Standard (DSS), U.S. Department Of Commerce, chap. Recommended Elliptic Curves for Federal Government Use, FIPS 186-2, 24-48, 2000.

Niemann, J.-C., Puttmann, C., Porrmann, M., and Rückert, U.: Resource Efficiency of the GigaNetIC Chip Multiprocessor Architecture, J. Syst. Architect., special issue on Architectural premises for pervasive computing, 53, 285-299, 2007.

Puttmann, C., Niemann, J.-C., Porrmann, M., and Rückert, U.: GigaNoC - A Hierarchical Network-on-Chip for Scalable Chip-Multiprocessors, in: Proceedings of the 10th EUROMICRO Conference on Digital System Design (DSD), 29–31 August, Lübeck, Germany, IEEE Computer Society, 495–502, 2007.

Puttmann, C., Shokrollahi, J., and Porrmann, M.: Resource Efficiency of Instruction Set Extension for Elliptic Curve Cryptography, in: Proceedings of the 5th International Conference on Information Technology: New Generation (ITNG), 7–9 April, Las Vegas, Nevada, USA, IEEE Computer Society, 131–136, 2008.