Flexible Hardware Reduction for Elliptic Curve Cryptography in GF(2^m)

Steffen Peter, Peter Langendörfer, and Krzysztof Piotrowski
IHP GmbH, Frankfurt(Oder), Germany
{peter,langendoerfer,piotrowski}@ihp-microelectronics.com

Abstract

In this paper we discuss two ways to provide flexible hardware support for the reduction step in Elliptic Curve Cryptography in binary fields (GF(2^m)). In our first approach we are using several dedicated reduction units within a single multiplier. Our measurement results show that this simple approach leads to an additional area consumption of less than 10% compared to a dedicated design without performance penalties. In our second approach any elliptic curve cryptography up to a predefined maximal length can be supported. Here we take advantage of the features of commonly used reduction polynomials. Our results show a significant area penalty compared to dedicated designs. However, we achieve flexibility and the performance is still significantly better than those of known ECC hardware accelerator approaches with similar flexibility or even software implementations.

1 Introduction

Embedded devices are becoming integral part of safety critical and long living systems, e.g. as part of monitoring applications for buildings, cars etc. This automatically implies that strong security means have to be applied in order to ensure data integrity and authenticity. But the battery power of embedded devices is not sufficient to run applications for several decades, especially not if heavy weight crypto means are applied. In order to cope with the security versus life time issue normally hardware for crypto means is applied. But what happens after a couple of decades? Due to ever increasing computational power the once applied key length is no longer sufficient. Updating an ASIC is impossible, so the obvious solution is to use a key length that is by far not necessary at the time of deployment. In this paper we are investigating approaches that allow hardware support of elliptic curves with different key length.

Our first approach is to include several dedicated reduction units into a sufficiently large multiplier. We call this approach MHWR and present measurement results for two designs one supporting the NIST curves [6] B-283, B-233 and B-163 and a second one supporting all recommended named binary curves from B-163 to B-571. The MHWR approach has very good parameters, i.e. only 10% additional area is needed and the performance is not negatively affected. But, the flexibility is still limited. In our second approach, called FSR, we use a flexible reduction unit which exploits the properties of commonly used elliptic curves. This design allows full flexibility up to a predefined key length. We realized such an accelerator for key length up to 283 bit. The measurements show that it needs considerable more area than dedicated systems, but is still 500 times faster and needs about 200 times less energy than comparable software implementations.

The rest of this paper is structured as follows. First, we provide an overview of the reduction operation in GF(2^m) and the flexibility issue. In section 3 we investigate flexible approaches for the reduction. Here we present our flexible reduction unit. Implementations of flexible polynomial multiplier and full ECC designs that apply the reduction units are presented in section 4. There we also discuss our measured results. A short summary concludes the paper.

2 Background

Finite field arithmetic is the fundamental backbone of many approaches in cryptography and coding theory. Binary finite fields (GF(2^m)) provide efficient algorithms and implementations of the arithmetic operations. For example, additions and subtractions in GF(2^m) are very fast because they can be implemented as simple XOR operations without

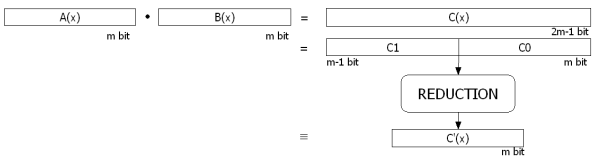

Figure 1: After a multiplication the result C has the double length. It must be reduced to an element C' with length m that is equivalent to C in the finite field, i.e the overlapping part C1 must be zero.

carry propagation. This renders these fields very favorable for cryptographic applications with long key lengths. Several elliptic curves that are for example recommended for ECC by the National Institute of Standards and Technology (NIST) [6] use these binary fields. Most hardware accelerators only support one field, i.e only one key length. We propose approaches that allow hardware implementations with high performance that provide a higher degree of flexibility. Flexibility means ability to compute different key lengths, i.e. binary fields GF(2^m) with different m.

The most important operation in GF(2^m) is the polynomial multiplication. Regarding flexibility the actual multiplication is not the biggest issue. For example, a 233 bit multiplier can multiply 163 bit values, if not used bit positions are padded with zeros. After a multiplication the product is too long for the finite field. The conversion to an equivalent element in the borders of the finite field is called reduction (see Fig. 1). Corresponding to classic finite fields the reduction can be performed by a division with remainder, i.e. long product is divided by the irreducible polynomial, which defines the finite field. But this modulo operation is too slow to be feasible. An alternative method is the repeated multiplications reduction (RMR) [2, 4]. This approach repeatedly subtracts the product of the overlapping part C1 (see Fig. 1) and the irreducible polynomial from C until the overlapping bits are zero. It is the only reported approach for flexible hardware reduction for ECs in $GF(2^m)$ that is not bound to specific irreducible polynomials. In [4] every reduction is performed by repeated multiplications of the irreducible and the overlapping part of the word. [2] presents a combined approach that performs reduction in specific named fields by hard wired reduction blocks, other fields are reduced by the repeated multiplication method.

The hard wired reduction (HWR) is the state of the art [2, 3]. It is very fast, very small, but it is tailored for exactly one field. With the knowledge of the irreducible polynomial it is possible to build a chain of XOR operations that performs the reduction within one step as a direct mapping from the long product to the m bit polynomial (see Fig. 2). But such a chain is bound to one field size and one irreducible polynomial. The slightest change results in a completely different XOR chain.

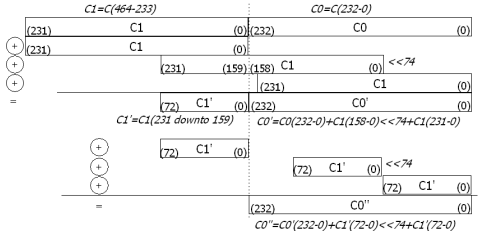

Figure 2: Reduction of a 465 bit word in $GF(2^{233})$. Finally the reduced $C0''$ can be determined by a direct mapping.

Table 1: Area consumption of combinatorial reduction blocks. 'Single' is the area for the single curve reduction. 'Combined' is the area for a reduction block that additionally can handle the smaller specific named fields, i.e. 283 can reduce the 163, 233 and 283 bit field

Size	Recommended	Area [mm^2]	
[bit]	irreducible[6]	Single	Combined
163	$x^{163}+x^7+x^6+x^3+1$	0.045	0.045
233	$x^{233}+x^{74}+1$	0.034	0.092
283	$x^{283}+x^{12}+x^7+x^5+1$	0.076	0.179
409	$x^{409}+x^{87}+1$	0.058	0.250
571	$x^{571}+x^{10}+x^5+x^2+1$	0.159	0.439

Thus, the actual flexibility issue is the reduction operation that must be performed after every multiplication. It is presumed that this operation cannot be supported flexible without losing much performance.

3 Flexible Reduction Approaches

Hard wired reduction is very efficient but without any flexibility regarding the key length. In contrast the state of the art flexible hardware reduction approach, the repeated multiplication reduction, requires much more clock cycles per multiplication. In this section we describe two approaches that allow flexibility without a significant performance loss.

3.1 Multiple hard-wired reduction

For a single curve the best solution is a hard wired reduction. The disadvantage of this approach is that it works exclusively for one field. But since other approaches are either very large or very slow, it is feasible to implement more than one of these blocks and select the one for the corresponding curve. In cases where the alternative curves are known, this method could result in a fast and competitive implementation. To examine this notion, we implemented reduction blocks for the five pseudo random curves in GF(2^m) recommended by the NIST, where each reduction block contains all smaller polynomials. Thus, the block of the 571-bit size includes reduction functionality for the 409, 283, 233 and 163 bit curves beside the 571 bit field. Table 1 shows a comparison of the resulting area consumption of the combined reduction blocks and the standalone blocks. The area overhead for the selection within the reduction logic is about 15% compared to the simple accumulation of the stand alone blocks. Initially we expected that logic of reduction blocks could be reused and a combination would lessen the total required area - but such effects could not be observed.

3.2 Flexible shift reduction (FSR)

Figure 2 depicts the construction diagram of the HWR block. It is shows that the overlapping part is shifted and XORed until it is zero. The shift operations are determined by the set positions of the irreducible polynomial. The idea of the flexible shift reduction (FSR) is to do the same for flexible word length and flexible irreducible polynomial. A general implementation of this idea would require up to m iterations of up to $m-1$ shifts, what would take too many clock cycles to be feasible. However, commonly used irreducible polynomials are not that general but have the following regularities:

- They are trinomials or pentanomials, i.e. with three or five set positions. Thus, not more than five shifts are needed per iteration.
- Since the second highest set position in the reduction polynomial is less than the half of its degree, it can be proved that only two successive multiplications are required for a complete reduction.

Thus, the number of shifts can be limited. Another problem is the addressing of the overlapping part. For GF(2^{233}) the overlapping part are bits 233 to 465, while for example for the smaller field GF(2^{163}) the overlapping part is bounded by bits 163 and 325. The FSR approach solves the problem by aligning smaller words to the middle of the hardware design so that the overlapping part is always accessible at the same position.

Figure 3 shows the FSR approach that realizes a reduction for irreducible polynomials having the described properties. It is a method that reduces data words up to a specified length for every irreducible polynomial as long as it is a trinomial or pentanomial. The maximum length is determined by the physical word size of the architecture (n). Figure 3 a) shows the reduction process for a polynomial of the full size, while Figure 3 b) of a smaller polynomial with the bit size m, where $m < n$. The full size reduction is quite obvious, since it is very similar to the basic idea of the hard-wired reduction. The overlapping part is shifted corresponding to the positions set in the reduction polynomial and is subsequently XORed to the word. After the first reduction step, more than the half of the overlapping part is zero. The positions of the data word that are set to zero are presented as shaded area in the picture. After the second step, which repeats the operations of the first step, the complete reduction process is completed. The scheme for smaller polynomials is almost the same. The only difference is that shorter words must be aligned, so that the least significant bit of the overlapping part has the same position as it does for the largest polynomial supported by the reduction unit. It corresponds to shifting the input word by $(n-m)$ bits to the left. Thus, the initial word is padded

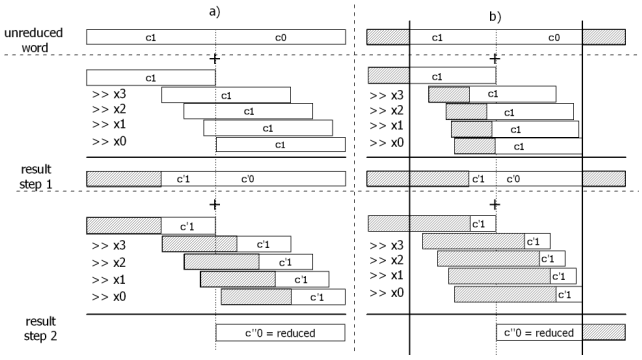

Figure 3: Scheme of a flexible reduction method. The left part a) shows a full size reduction. The right part b) is a reduction for a smaller curve. When the smaller polynomials are initially aligned so that the overlapping part starts at the same position as long polynomials, the same reduction logic can be used. The shaded parts represent zeros. The configuration of the reduction is determined by the four shift values x1, x2, x3, x4.

with zeros at the left and the right end. After this operation, the reduction process is like the process for the full word.
For example, consider a reduction block for the maximum polynomial size of 233 bit. In this block we want to reduce a result of a multiplication within the recommended 163 bit NIST field. The reduction polynomial is $x^{163} + x^7 + x^6 + x^3 + 1$. To align the input word it is shifted to the left by the difference of field sizes, i.e. (233-163=) 70 bits. This means that the shaded area of the input word of Figure 3 b) represents 70 bits at the beginning and 70 bits at the end. The four shifting values are: $x3 = 163 - 7 = 156$, $x2 = 163 - 6 = 157$, $x1 = 163 - 3 = 160$, $x0 = 163 - 0 = 163$. After the reduction is completed, the result must be shifted back by 70 bits to obtain the usual right aligned data word.
After synthesizing the design, it becomes apparent that especially the shifters are connected with a crucial problem. Flexible shifters with a size of hundreds of bits are very slow and large. A design that corresponds to the idea of Figure 3 requires one left-shifter for the smaller input words, four concurrent right-shifters for each of the two reduction steps, and one right-shifter for the final alignment. Thus, ten large flexible shift operations are required for one reduction.
One approach to minimize the number of shifts would be the limitation of the irreducibles to trinomials. It decreases the number of shift operation by four, but the elimination of the pentanomials implies a significant loss of functionality. An approach that avoids one of the four concurrent shifters without losing functionality is shown in Figure 4. The term that belongs to the $x0$-shift is not shifted but directly forwarded to the final accumulation stage. It can be done because all irreducible polynomials have the structure: $r(x) = x^m + ... + 1$, i.e. the terms x^m and 1 are part of every reduction polynomial. In the original scheme (Fig. 3), the partial product that corresponds to x^m is not shifted, but the term

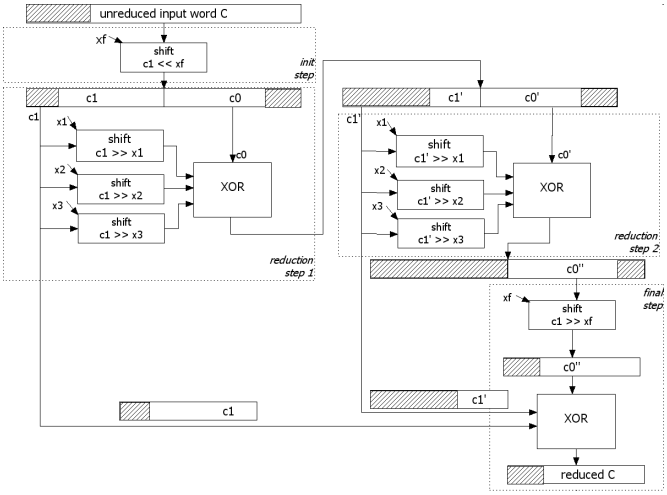

Figure 4: Scheme of the optimized flexible shift reduction method. The input is shifted in the init step to obtain the middle alignment. Then the two reduction steps are performed with three shift operations each (x1,x2,x3). The fourth shift (x0) can be saved when it is forwarded to the final step and accumulated after the final right shift.

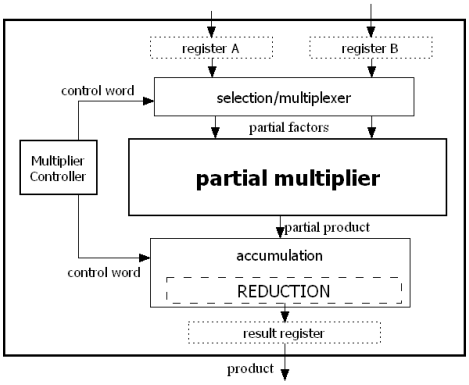

Figure 5: Structure of the polynomial multiplication unit based on the IKM approach [1]. Our preferred configuration requires 9 partial multiplication. The reduction is integral part of the multiplier.

that represents position 0 is shifted right by $x0$. Since $x0$ is the difference of m and the lowest set bit, which is always 0 ($x^0 = 1$), $x0$ is equivalent to m ($m - 0$).
In the final step of the reduction, the intermediate result, which is still aligned to the middle, is shifted so that it is aligned to the right. The amount of the bits for the final right shift is: $xf = n - m$, whereby n is the physical word length and m is the size of the calculated field.
Since the $x0$-terms have no impact on the intermediate overlapping part (c'1) it can be moved to the end. The two successive shift operations that have to be executed at the end ($>> m$ and $>> n-m$) can be merged to one shift operation ($>> n$). This n depends on the hardware design and not on the calculated field. Therefore, this shift operation is not an expensive flexible shift operation but a cheap readdressing. With this improvement still eight shift operations for one reduction are required.

We implemented the shifter applying a multiplexer chain that shifts the input corresponding to the shift value consecutively by 1, 2, 4, ..., or 128. We could build and synthesize such a design that performs the reduction within one clock cycle. A 233 bit version of the reduction block requires about $1.3\mathrm{mm}^2$ in silicon in a $0.25\mu\mathrm{m}$ CMOS technology after synthesizing at a frequency of 70 MHz. This design computes the polynomial reduction within one clock cycle in every field up to a degree of 233 if the reduction polynomial is a trinomial or pentanomial.
The large area and relatively low clock speeds caused by the large shifters are the big disadvantages of this approach. Further investigations could improve these parameters. The presented design works for all field sizes up to a specified size. It even works for very small fields that have no practical use, e.g. $GF(2^3)$. A minimum field size of, e.g. 150 bit would reduce the complexity of the shifter logic, since several shift values are not required in such scenario. Experiments with sequential applications of the reduction process have shown that less silicon area and higher clock frequencies can be achieved if the reduction may take more than one clock cycle. However, in the following evaluation we only consider the initial design with the combinatorial FSR reduction.

4 Flexible ECC Designs

The flexible reduction units discussed in the previous section shall now be applied in complete ECC hardware designs that accelerate the elliptic curve point multiplication (ECPM). Beside the multiplication and reduction unit these designs comprise an ALU with embedded addition and squaring functionality, eight registers of word size n and a control logic that in particular controls the access to the system bus that transfers one data word per clock cycle. Before discussing the complete ECC designs we introduce the polynomial multiplication units, which are the most important functional units.

4.1 Polynomial Multiplication

All our multiplication units base on the iterative Karatsuba multiplication (IKM) approach proposed in [1]. They only differ in the word size and the reduction method, which is integral part of the multiplication unit. Our preferred multiplier setup requires 9 clock cycles for a polynomial multiplication. In each cycle partial factors are determined in the selection block, a partial multiplication is performed and the result is accumulated. The final accumulation step comprises the reduction. For the flexible designs the reduction

Table 2: Overview of flexible multiplication units supporting several fields in $GF(2^m)$.

Reduction method	Size [bit]	contains fields: 163	233	283	409	571	all upto	Area [mm^2]	Speed period[ns]	cycles
MHWR	233	X	X				-	0.67	10	9
	283	X	X	X			-	0.94	11	9
	409	X	X	X	X		-	1.54	12	9
	571	X	X	X	X	X	-	2.37	15	9
FSR	233	X	X				233	1.72	20	9
	255	X	X				255	1.94	21	9
	283	X	X	X			283	2.30	23	9

block is exchanged and additional inputs for the selection of the field length and the irreducible are added. It is premised that smaller polynomials are padded with zeros to meet the full multiplication factor length.

Table 2 shows an overview of multipliers for different fields applying the reduction methods as they were described in the previous section. One can see that the MHWR approach is very efficient. Both area and speed are barely affected by the additional reduction functionality. Contradictory results are observed for the FSR implementation. Both, area and speed are negatively affected with the factor of 2.

4.2 Evaluation of flexible ECC designs

The polynomial multiplication units with embedded reduction blocks are the core of our ECC designs. For the final evaluation of the impact of the flexible reduction approaches we implemented various ECC accelerators of different sizes and measured the effects on time, area, and energy. We realized two MHWR designs and one FSR implementation. In addition we implemented a design with the classic repeated multiplication reduction (RMR) as introduced in section 2.

MHWR 283 This three fields design bases on a 283 bit single curve design and includes the NIST curves B-163, B-233, and B-283. The reduction units in multiplier and the squarer have been replaced by an MHWR block for the three addressed fields. The embedded IKM requires nine clock cycles for a polynomial multiplication and has a factor size of 80 bit.

MHWR 571 Additionally to the 3 fields of MHWR 283 this design supports the fields B-409 and B-571. It is an adapted 571 bit single curve design with embedded 160 bit core multiplier. A multiplication in $GF(2^{409})$ and $GF(2^{571})$ requires nine clock cycles while multiplications on smaller fields are three times faster, due to a better utilization of the multiplier.

FSR 283 It is the same 283 bit ECC design as the MHWR 283, but with the 283 bit FSR logic embedded in multiplier and squarer. Except additional registers for the configuration of the reduction polynomial, the design is the same as for a single curve design. The reduction polynomial is stored in four registers of a size of nine bits, which describe set positions in the polynomial.

Table 3: Comparison of parameters of the RMR, the FSR, two MHWR ECC blocks, and corresponding single curve blocks.

	ECPM bit size	Single curve	MHWR 283	MHWR 571	FSR 283	RMR 283
Period [ns]	163	9	11	16	23	16
	283	11	11	16	23	16
	571	15	-	16	-	-
clk cycles	163	9251	9251	7383	9251	23990
	283	15922	15922	9215	15922	41900
	571	32275	-	32275	-	-
Time [μs]	163	83	102	118	212	384
	283	175	175	147	365	670
	571	484	-	517	-	-
Area [mm^2]	163	1.0	2.0	4.3	4.5	3.2
	283	1.9	2.0	4.3	4.5	3.2
	571	3.9	-	4.3	-	-
Energy [μWs]	163	12.8	25.8	23.2	81.6	78.4
	283	43.5	50.6	48.1	148.1	147.5
	571	191.7	-	248.8	-	-

RMR 283 In this 283 bit ECC design the standard multiplier was extended by a logic that performs the repeated multiplications. In order to achieve comparable results, we chose a bigger and faster core multiplier that allows the execution of a polynomial multiplication in three cycles. A default nine-cycle multiplier would require three times the time and merely 20% less area. The design does not contain a particular squaring unit. For the storage of the reduction polynomial a 283 bit register is required.

4.3 Comparision

The results for timing, area, and energy for the four flexible designs, together with a comparison to the single curve designs are shown in Table 3. The evaluation of these results indicates the following conclusions:

Area: The silicon area in both considered MHWR designs is less than 10% larger than for the single curve design. Therefore, the additional cost for the support of specific smaller curves is acceptble.

The FSR design requires more than double the area than a single curve design for the same word length. The RMR design is about 30% smaller than the version with FSR, but with $3.2\mathrm{mm}^2$ it still requires much more area than the 283 bit single curve design, which needs only $1.9\mathrm{mm}^2$.

Speed: The number of clock cycles required for an ECPM by MHWR is equal to the corresponding single curve implementation. An exception is the execution of small curves on the large 571 bit design. Here, the efficient application of the large internal multiplier reduces the number of clock cycles. The maximum clock frequency is slightly below the frequency for the single curve full size design. The FSR approach does not need more clock cycles but the maxi-

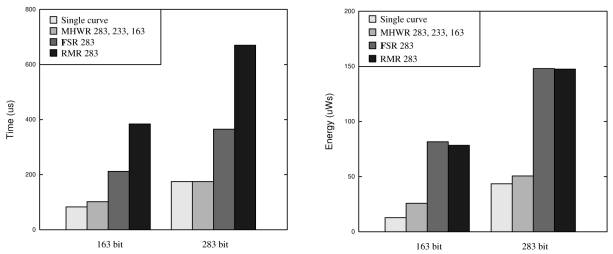

Figure 6: Required time (on the left) and total energy for an ECPM on B-163 and B-283. While FSR is much faster than RMR, the energy consumption of both full flexible designs is on a same level but is at least the threefold of the single curve and MHWR implementations.

mum clock frequency decreases significantly. For the implemented configuration even 50 MHz is not possible. In contrast RMR allows a faster frequency but suffers from the huge number of clock cycles so that it finally requires double the time of the FSR design. Figure 6 shows the comparison of the two full flexible designs, the 283 bit MHWR supporting three curves, and the corresponding single curve implementation. The FSR design, due to the slow clock, requires roughly double the time of the single curve operation, but is about twice as fast as the RMR implementation. In contrast, MHWR with support for 3 curves is only marginally slower than the 283 bit single curve design.

Energy: As expected the energy consumption increases with larger word sizes and more flexibility. Smaller fields are computed more energy efficient on MHWR571 than on the smaller MHWR283. This is due to the improved utilization of the bigger field multiplication unit.

Figure 6 compares the required energy for one ECPM for the corresponding single curve and flexible design. Both FSR and RMR require more than three times the energy than the single curve design. For small 163 bit ECPM the energy consumption is increased by the factor of five. The improved execution time of FSR does not lead to an reduced energy consumption.

5 Conclusion and Summary

Elliptic Curve Cryptography features such as key length and calculation effort, make them a good choice for embedded systems compared to RSA. But each curve has its own irreducible polynomial and efficient hardware designs usually exploit the properties of this polynomial. Thus, normally hardware accelerators can support only one dedicated elliptic curve. In this paper we have introduced two ways to provide flexible hardware support for elliptic curves. Our first approach is quite straightforward, since we included several dedicated reduction units into a polynomial multiplier. Even if this looks like a waste of area, our measurements show that the area penalty is less than 10 per cent. In addition the execution speed is not affected negatively. For example the 283 bit single curve design synthesized for a 0.25μ CMOS technology has a size of $1.9\mathrm{mm}^2$, the ECPM-operation requires 175μs and 43.5μWs. A design that additionally supports B-233 and B-163 curves has a size of $2.0\mathrm{mm}^2$, and the 283 bit ECPM needs 175μs and 50.6μWs.

Our second approach is more sophisticated. Here we use a reduction unit which adjusts polynomials that have to be reduced in such a way that all polynomials up to a certain length, i.e. the size of the reduction unit, can be reduced. The full flexibility has its costs. The silicon area for the FSR approach is more than 2 times the area of a single curve. Also time and energy consumption increase significantly compared to the optimized single curve design. Here, FSR is about 50% faster than traditional flexible reduction methods as RMR. For our knowledge, the FSR based ECC design is the fastet flexible ECC accelerator. In comparison to an examplary software implementation based on the MIRACL[5] library and running on a MIPS R4000 33MHz, also assembled in 0.25 μ CMOS technology, the benefits of that design become even more apparent. Such a software implementation provides full flexibility but, varying with the field size, requires at least 200 times the time and 550 times the energy. For very small devices as wireless sensor nodes that numbers become even more distinct.

These parameters qualify our flexible designs as well suited solutions for mobile and embedded designs that demand not only good performance but low energy consumption.

Acknowledgment

This work was partially funded by the German Ministry of Education and Research under grant 01AK060B and in part by EU under contract 26820.

References

[1] Z. Dyka and P. Langendoerfer. Area efficient hardware implementation of elliptic curve cryptography by iteratively applying karatsuba's method. In *DATE*, pages 70–75, 2005.

[2] H. Eberle, N. Gura, and S. C. Shantz. A cryptograhpic processor for arbitrary elliptic curves over. In *ASAP*, 2003.

[3] N. A. Saqib, F. Rodríguez-Henríquez, and A. Díaz-Pérez. A parallel architecture for fast computation of elliptic curve scalar multiplication over $GF(2^m)$. In *IPDPS*, 2004.

[4] A. Satoh and K. Takano. A scalable dual-field elliptic curve cryptographic processor. *IEEE Trans. Comput.*, 52(4):449–460, 2003.

[5] M. Scott. *MIRACL—A Multiprecision Integer and Rational Arithmetic C/C++ Library, Version 5.0*. Shamus Software Ltd, Dublin, Ireland, 2005.

[6] F. U.S. Department of Commerce/NIST. Digital Signature Standard (DSS), FIPS PUB 186-2, Jan. 27, 2000.