Towards strong security in embedded and pervasive systems: energy and area optimized serial polynomial multipliers in $GF(2^k)$

Zoya Dyka, Peter Langendoerfer, Frank Vater and Steffen Peter
IHP,
Im Technologiepark 25,
D-15232 Frankfurt (Oder), Germany
http://www.ihp-microelectronics.com/

***Abstract*—In this paper we discuss a theoretical approach to do early assessment of the area and power consumption of hardware accelerators for elliptic curve cryptography. For evaluation we developed several different one clock multipliers as building block for the final serial multipliers. The former are evaluated concerning their efficiency in comparison to literature and the results of our assessment technique are compared with synthesis results. Since the application areas we are interested in are embedded systems, pervasive computing or wireless sensor networks we investigated serial multipliers in order to achieve reduced area and energy consumption compared to the single clock multipliers. Our evaluation clearly shows that our approach provides good hints to select well suited implementation candidates.**

Keywords: ECC, GF(2^k), polynomial multiplication

I. INTRODUCTION

Embedded systems and wireless sensor networks have gained significant interest for being used in critical application areas such as industrial control systems. Reliability and by that security of those systems is of extremely high importance. But, for resource constraint devices the computational effort of strong security means, especially when public key cryptography is used are still too costly.

During recent years elliptic curve cryptography has gained significant attention. This is due to the fact that it provides higher security than RSA while consuming significant fewer resources. I.e. its keys are shorter and the computational overhead is less. But, for resource constraint devices such as embedded systems or wireless sensor nodes the computational effort when realized in software is still too high. The alternative is to implement ECC operations as hardware accelerators. The polynomial multiplication is costly and often executed when ECC is applied. Thus, it is the function that has the highest impact on area and/or energy consumption and by that it is the most often researched function for hardware implementation of ECC. Finding the "optimum" hardware accelerator is a time consuming and difficult task. On the one hand it needs to be as small as possible to reduce production costs, on the other hand it needs to be as energy efficient as possible since the target system is battery powered. One way to improve the hardware implementation of ECC is the optimization of polynomial multiplication. There exist several approaches to minimize the complexity of the polynomial multiplication [1]-[8]. A very promising approach to optimize polynomial multipliers is to combine these methods [9]. If an optimum combination is found this reduces the area and energy consumption significantly [10]. Another means to reduce the area of multipliers is to prolong the calculation time, i.e. to spend several clock cycles to calculate the polynomial product as a sum of partial products [11], [12]. Determining the multiplier with the minimal energy and/or area consumption is a tedious task especially when several multiplication methods are taken into account.

In this paper we present a means to assess the gate complexity of polynomial multipliers and discuss its applicability using 25 design alternatives as well as synthesis results of the most promising implementation candidates. Our evaluation shows that the complexity assessment provides good results well suited to select from different design choices.

The rest of this paper is structured as follows. In section II we shortly introduce the background of polynomial multiplication and our complexity assessment approach. After that we discuss and evaluate one clock multipliers that we use later for realizing serial multipliers, and their theoretical gate complexity. Then we investigate the resulting serial multipliers and their gate complexity. In section III we evaluate our approach using data from the literature as well as synthesis results. The paper closes with short conclusions.

II. POLYNOMIAL MULTIPLIERS

A. Definition of polynomial multiplication

First of all we give the definition of polynomial multiplications over extended binary fields $GF(2^k)$.

Let $A(x)=\sum_{i=0}^{k-1} a_i \cdot x^i$ and $B(x)=\sum_{i=0}^{k-1} b_i \cdot x^i$ be elements from $GF(2^k)$ and $f(x)$ be the irreducible polynomial of degree k generating $GF(2^k)$.

The multiplication of the elements $A(x)$ and $B(x)$ is defined as follows:

$$C'(x) = A(x) \cdot B(x) \mod f(x) = C(x) \mod f(x). \quad (1)$$

978-1-4673-0229-6/12/$31.00 ©2012 IEEE

The multiplication (1) can be performed in two steps: the calculation of the polynomial *C(x)* of degree *(2k-2)* and reduction of this polynomial *C(x)* to the degree *k*. In this paper we concentrate on the optimization of the first step of the multiplication only, i.e. on the polynomial multiplication:

$$C(x) = A(x) \cdot B(x) = \sum_{i=0}^{2k-2} c_i \cdot x^i \text{ , with } c_i = \bigoplus_{\substack{k>j\geq 0,\\ k>l\geq 0,\\ j+l=i}} a_j \cdot b_l \qquad (2)$$

The symbol $\oplus$ in (2) denotes the Boolean XOR operation. Elements of $GF(2^k)$ can be (and are usually) represented as *k*-bits long binary numbers: $A(x) = \sum_{i=0}^{k-1} a_i \cdot 2^i$.

B. Gate complexity of polynomial multiplication

We describe the gate complexity (GC) of polynomial multiplication for *k*-bits operands by the exact number of the Boolean XOR operations *'#XOR'* and by the exact number of the Boolean AND operations *'#AND'*, because only these two operations are needed to calculate the polynomial product (see (2)) and because they are implemented in hardware by an XOR or an AND gate respectively.

$$GC_k^{MM} = (\#AND;\ \#XOR). \qquad (3)$$

In (3) the '*MM*' shows the used multiplication method. The minimal area and energy consumption of a multiplier can be calculated based on its gate complexity (3) and on the area and the energy consumption of the gates ($Area_{AND}$, $Area_{XOR}$ and E_{AND}, E_{XOR}) of the applied technology:

$$\begin{aligned} Area &= \#AND \cdot Area_{AND} + \#XOR \cdot Area_{XOR} \\ E &= \#AND \cdot E_{AND} + \#XOR \cdot E_{XOR} \end{aligned} . \qquad (4)$$

We use the following model for the average energy in (4):
In our model are considering two input AND and XOR gates only. The energy consumption of the gates depends on their input values. If the input values do not trigger any state change the consumed energy is negligible, otherwise i.e. if input values trigger state changes the consumed energy is by far higher and depend on the logic function implemented by the gate and on whether the output changes from zero to one or vice versa.
In order to estimate the energy consumption of design alternatives we use the average energy consumption of AND and XOR gates (E_{AND}, E_{XOR}) which we calculated from the energy consumption of the following four state changes for the IHP 0.13µ technology:
- first input is 0; second input is 0 and output changes to 1
- first input is 0; second input is 1 and output changes to 0
- first input is 1; second input is 0 and output changes to 1
- first input is 1; second input is 1 and output changes to 0.

An additional challenge when calculating the energy consumption stems from the fact that not all gates are always flipping when the polynomial product is calculating. This fact can be taken to account using a statistical coefficient. For example, if in average only 50% of all gates are flipping formula (4) can be modified as follows: $E = 1/2(\#AND \cdot E_{AND} + \#XOR \cdot E_{XOR})$. The problem is that the number of the flipping gates of the polynomial multiplier depends on both multiplicands, i.e. on the inputs. The using of such statistical coefficients in the energy consumption model of polynomials multipliers might result in a significant difference of the theoretical and practical results since the latter are obtained for the concrete multiplicand values.
We are aware of the fact that our model only very roughly approximates the real energy consumption but since we are mainly interested in comparing implementation alternatives, and not in predicting concrete energy consumption values, we are convinced that our model is well suited for assessing implementation candidates.

There exist many multiplication methods (MM) that can be applied to reduce the complexity of polynomial multipliers: the classical and the generalized Karatsuba MM [1] for $n>1$; Karatsuba MM [2] for 2- and Winograd MM [3] for 3-term operands, the latter two are the special cases of the generalized Karatsuba MM; Montgomery MM for 5-, 6- and 7-term operands [5]. These approaches segment the multiplicands into *n* smaller *m*-bits long parts (terms), which are then multiplied. Of all these MMs the classical MM requires the biggest number of partial multiplications. Other MMs lead to a reduced number of partial multiplications but require more XOR-operations compared to the classical MM. The gate complexity of these Karatsuba-like multiplication methods can be also described using a slightly modified i.e. recursive definition of the gate complexity:

$$GC_{k=nm}^{MM_i} = \left(\#MULT \cdot GC_m^{MM_j};\quad \#XOR\right), \text{ where} \qquad (5)$$

MM_i – the used MM; $\#MULT$ – the number of partial multiplications, and $GC_m^{MM_j}$ – the gate complexity of partial multipliers.

The number of partial multiplications *#MULT* and the number of XOR gates *#XOR* depend on the selected multiplication method MM_i and on the segmentation *n* of the operands. The knowledge of the gate complexity $GC_m^{MM_j}$ of each partial multiplier as tuple (3) allows calculating the gate complexity of full *k*-bits multipliers.

C. One clock multipliers

A one clock multiplier, which implements the multiplication method MM_i and contains *#MULT* of partial multipliers (see (5)), can calculate the polynomial product in a single clock cycle. Each partial multiplication can be implemented by any multiplication method MM_j or even by any combination of multiplication methods. Optimal combinations of different MMs can be determined algorithmically [9]. In [10] we presented the determination of area and/or energy optimal combinations of 8 MMs with reduced number of XOR operations.

In this paper we describe the combination of 13 MMs as described in [10]. We determined the area-optimal combinations of these MMs for the IHP 0.13µ technology for all lengths of operands up to 600 bits. In order to benchmark our results we are using results from [1][1] and [9]. Since these articles give the gate complexity only for operands of a length up to 128 bits we reconstructed the gate complexity for longer operands i.e. up to 600 bits. We calculated the area of multipliers based on the reconstructed gate complexity using eq. (4). Fig. 1 shows the calculated area of multipliers for polynomials of a length of up to 600 bits for both approaches used as benchmarks and for our approach. The red curve in Fig.1 depicts the chip-area of multipliers that we calculated based on gate complexity reconstructed from [1]. The black curve depicts results we derived from [9]. The area consumption of our approach (green curve in Fig.1) is 33 per cent smaller than the results represented by the red curve and about 10 per cent smaller than the results represented by the black curve.

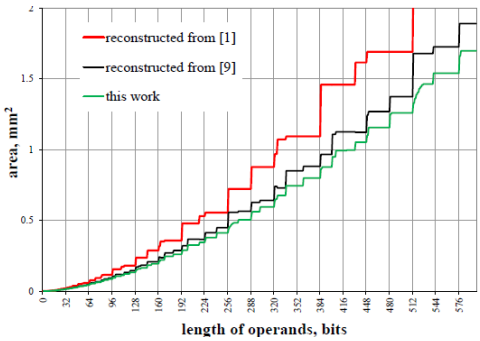

Figure 1. Calculated area of multipliers for an operand length of up to 600 bits for our approach (green line) and approaches presented in [1] (red line) and [9] (black line) respectively

Table I shows the parameters of the synthesized polynomial multipliers that we used as *m*-bits partial multipliers in our serial *nm*-bits polynomial multiplier.

TABLE I. PARTIAL MULTIPLIERS: CALCULATED PARAMETERS AND SYNTHESIZED VALUES

***m*, bits**	**Theory**			**Synthesis**	
	Gate Complexity	***area*, mm²**	***energy*, pJ**	***area*, mm²**	***energy*, pJ**
64	GC_{64}=(1296$_{AND}$; 2387$_{XOR}$)	0.0426	35.13	0.0396	29.82
59	GC_{60}=(1350$_{AND}$; 2094$_{XOR}$)	0.0391	32.76	0.0362	27.12
78	GC_{80}=(2025$_{AND}$; 3396$_{XOR}$)	0.0616	51.34	0.0575	42.90

The area and energy of all synthesized multipliers are smaller than the theoretical calculated values. This can be explained by the following fact: the calculation of the area of multipliers is based on the area of 2-inputs-AND and 2-inputs-XOR gates. In reality each technology provides also other gate types. For example, the IHP 0.13µ technology [13] has also 3-inputs-XOR gates. The Synopsis tools [14], that we use for the synthesis of the multipliers applies often a single 3-inputs-XOR gate instead two 2-inputs-XOR gates, reducing the resulting area and changing the energy consumption. The average deviation of the area of synthesized multipliers from the calculated one is about 7 %. The deviation of the energy values is much larger, i.e. approximately 16%. This deviation is caused by the following facts:

- not all gates are flipping causing the reduction of the theoretically determined energy consumption value
- the energy consumption of each gate depends on its input values and by that the average value used in the calculation already deviates stronger from the real values as in the case of the constant area parameter. To obtain energy consumption values from Table I we performed a sequence of more than 1000 multiplications using random inputs. For each gate of the multipliers its energy consumption was collected. Based on these values the total power consumption of the multiplier and its energy consumption per single clock cycle (i.e. per individual multiplication) were determined.

Table II shows the relation of theoretically calculated values and synthesis results for area and energy for investigated multipliers using the 64-bit partial multiplier as normalization value.

TABLE II. RELATIONS OF PARAMETERS OF INVESTIGATED MULTIPLIERS

***m*, bits**	***area*, mm²**		***energy*, pJ**	
	Theory	***Synthesis***	***Theory***	***Synthesis***
64	1	1	1	1
59	0.92	0.91	0.93	0.91
78	1.58	1.59	1.57	1.58

D. *Serial multipliers*

All partial products can be calculated serial within *#MULT* clock cycles using a single *m*-bits partial multiplier only (see (5)). This reduces the area of the *k*-bits polynomial multiplier significantly [11], [12].

In serial multipliers the polynomial product will be accumulated in output registers. We explain this process using the example of the Winograd MM for 3-term polynomials, i.e. for the case of the segmentation of the operands into n=3 *m*-bits long terms. The multiplication formula for this case is the following:

$$\begin{aligned} \underbrace{C(x)}_{6m-1} = \underbrace{C_5}_{m-1}\underbrace{C_4}_{m}\underbrace{C_3}_{m}\underbrace{C_2}_{m}\underbrace{C_1}_{m}\underbrace{C_0}_{m} = \underbrace{A_2}_{m}\underbrace{A_1}_{m}\underbrace{A_0}_{m} \cdot \underbrace{B_2}_{m}\underbrace{B_1}_{m}\underbrace{B_0}_{m} = A_0B_0 \cdot 2^0 \oplus \\ \oplus (A_0B_0 \oplus A_1B_1 \oplus (A_0 \oplus A_1)(B_0 \oplus B_1)) \cdot 2^m \oplus \\ \oplus (A_0B_0 \oplus A_1B_1 \oplus A_2B_2 \oplus (A_0 \oplus A_2)(B_0 \oplus B_2)) \cdot 2^{2m} \oplus \\ \oplus (A_1B_1 \oplus A_2B_2 \oplus (A_1 \oplus A_2)(B_1 \oplus B_2)) \cdot 2^{3m} \oplus A_2B_2 \cdot 2^{4m} \end{aligned} \quad (6)$$

Each partial product PP_i=A_iB_i (or PP_{ij}=$(A_i \oplus A_j)(B_i \oplus A_j)$) in (6) is (*2m-1*)-bits long and can be written as a sum of two segments: the (*m-1*) most significant bits, which will be denoted as $PP_i[1]$ and the remaining *m*-bits denoted as $PP_i[0]$.

[1] by results of [1] we denote the data provided for the recursively applied generalized Karatsuba MM with minimal number of AND and XOR gates.

$$\underbrace{PP_i}_{2m-1\text{ bits}} = \underbrace{PP_i[1]}_{m-1\text{ bits}} \cdot 2^m + \underbrace{PP_i[0]}_{m\text{ bits}} \tag{7}$$

Eq. (8) is derived from (6) by applying (7):

$$C(x) = \underbrace{PP_0[0]}_{C_0} \cdot 2^0 \oplus \underbrace{\begin{pmatrix} PP_0[0] \oplus \\ PP_0[1] \oplus \\ PP_1[0] \oplus \\ PP_{01}[0] \end{pmatrix}}_{C_1} \cdot 2^m \oplus \underbrace{\begin{pmatrix} PP_0[0] \oplus PP_0[1] \oplus \\ PP_1[0] \oplus PP_1[1] \oplus \\ PP_2[0] \oplus PP_{02}[0] \oplus \\ PP_{01}[1] \end{pmatrix}}_{C_2} \cdot 2^{2m} \oplus$$
$$\oplus \underbrace{\begin{pmatrix} PP_0[1] \oplus PP_1[0] \oplus \\ PP_1[1] \oplus PP_2[0] \oplus \\ PP_2[1] \oplus PP_{12}[0] \oplus \\ PP_{02}[1] \end{pmatrix}}_{C_3} \cdot 2^{3m} \oplus \underbrace{\begin{pmatrix} PP_2[0] \oplus \\ PP_1[1] \oplus \\ PP_2[1] \oplus \\ PP_{12}[1] \end{pmatrix}}_{C_4} \cdot 2^{4m} \oplus \underbrace{PP_2[1]}_{C_5} \cdot 2^{5m} \tag{8}$$

Eq. (8) can also be represented as a table In the rest of this paper named as table representation, (TR), see Table II. The leftmost column shows both segments of the partial products, i.e. all PP_i[d] and PP_{ij}[d], where $d \in \{0,1\}$. All other columns correspond to an individual product segment. If the product segment C_i is calculated by adding a partial product segment PP_j[d], the cell of the TR with 'coordinates' (PP_j[d], C_i) is filled with the symbol ⊕. Otherwise the cell is empty.

TABLE III. TABLE REPRESENTATION OF 3-TERM-WINOGRAD-MM, '⊕' INDICATES WHICH PARTIAL PRODUCTS NEED TO BE ADDED TO RETRIEVE C_i

$A_0 \cdot B_0$ [0]=PP_0[0]				⊕	⊕	⊕
$A_0 \cdot B_0$ [1]=PP_0[1]			⊕	⊕	⊕	
$A_1 \cdot B_1$ [0]=PP_1[0]			⊕	⊕	⊕	
$A_1 \cdot B_1$ [1]=PP_1[1]		⊕	⊕	⊕		
$A_2 \cdot B_2$ [0]=PP_2[0]		⊕	⊕	⊕		
$A_2 \cdot B_2$ [1]=PP_2[1]	⊕	⊕	⊕			
$(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)$ [0]=PP_{01}[0]					⊕	
$(A_0 \oplus A_1) \cdot (B_0 \oplus B_1)$ [1]=PP_{01}[1]				⊕		
$(A_0 \oplus A_2) \cdot (B_0 \oplus B_2)$ [0]=PP_{02}[0]				⊕		
$(A_0 \oplus A_2) \cdot (B_0 \oplus B_2)$ [1]=PP_{02}[1]			⊕			
$(A_1 \oplus A_2) \cdot (B_1 \oplus B_2)$ [0]=PP_{12}[0]			⊕			
$(A_1 \oplus A_2) \cdot (B_1 \oplus B_2)$ [1]=PP_{12}[1]		⊕				
	C_5	C_4	C_3	C_2	C_1	C_0

Table III contains 6 different partial products. They can be calculated within 6 clock cycles using a single partial multiplier. In this case only one partial product is available at the same time. For each segment C_i the sum of necessary segments of partial products can be accumulated within 6 clock cycles. Fig. 2 shows the structure of the serial multiplier and Table IV shows the corresponding accumulation plan.

Fig. 3 and 4 show the accumulation units for different segments of the polynomial products according to the accumulation plan.

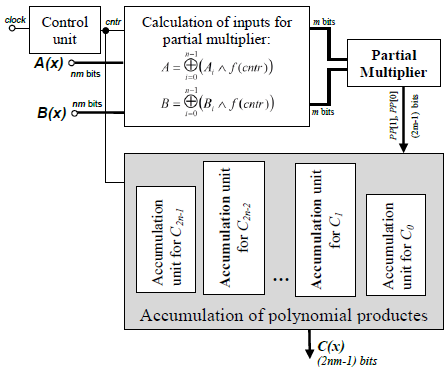

Figure 2. Structure of our serial polynomial multiplier

TABLE IV. ACCUMULATION PLAN FOR CALCULATION OF THE POLYNOMIAL PRODUCT ACCORDING TO TABLE II

Clock cycle	Available partial product *PP*	Accumulation plan
1	PP_0	C_0=C_0⊕PP[0]; C_1=C_1⊕PP[0]⊕PP[1]; C_2=C_2⊕PP[0]⊕PP[1]; C_3=C_3⊕PP[1]
2	PP_1	C_1=C_1⊕PP[0]; C_2=C_2⊕PP[0]⊕PP[1]; C_3=C_3⊕PP[0]⊕PP[1]; C_4=C_4⊕PP[1]
3	PP_2	C_2=C_2⊕PP[0]; C_3=C_3⊕PP[0]⊕PP[1]; C_4=C_4⊕PP[0]⊕PP[1]; C_5=C_5⊕PP[1]
4	PP_{01}	C_1=C_1⊕PP[0]; C_2=C_2⊕PP[1]
5	PP_{02}	C_2=C_2⊕PP[0]; C_3=C_3⊕PP[1]
6	PP_{12}	C_3=C_3⊕PP[0]; C_4=C_4⊕PP[1]

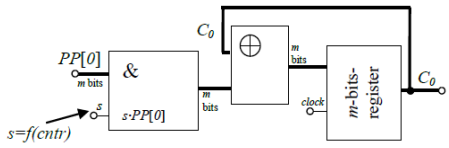

Figure 3. The accumulation unit for C_0: this unit contains m AND gates, m XOR gates and m flip-flops to realize the m-bits register. The structure of the accumulation unit for C_{2n-1} is the same, but the input value is the (m-1)-bits long PP[1].

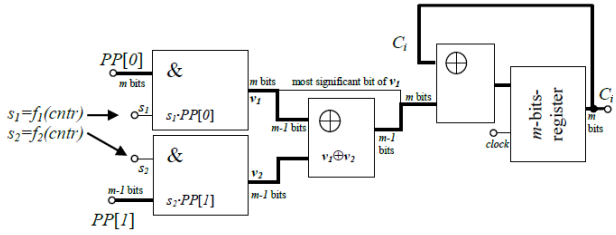

Figure 4. Accumulation unit for C_i, $1<i<2n-2$; this unit contains (m+m-1) AND gates, (m-1+m) XOR gates and m flip-flops to realize the m-bits register.

The gate complexity of the k-bits serial polynomial multipliers can be described according to its structure (see Fig. 2) as follows:

$$GC_{nm} = GC_{\substack{\text{Control} \\ \text{unit}}} + GC_{\substack{\text{calculation} \\ \text{of inputs for} \\ \text{partial} \\ \text{multiplier}}} + \sum_{i=0}^{2n-1} GC_{\substack{\text{Accumulation} \\ \text{unit for } C_i}} + GC_m \tag{9}$$

The gate complexity of the control unit is very small compared to the area of other units. Therefore (9) can be written for the implementation of 233-bits multiplication as follows:

$$GC_{nm} \approx 2\cdot(233_{AND};\ (233-m)_{XOR}) + \left(\begin{array}{l}(m_{AND};\ m_{XOR}) + \\ \sum_{i=1}^{2n-2}((2m-1)_{AND};\ (2m-1)_{XOR}) + \\ ((m-1)_{AND};\ (m-1)_{XOR})\end{array}\right) + \quad (10)$$
$$+ GC_m = GC_m + (467+4nm-2(n+m)_{AND};\ 234+4nm-2n-3m_{XOR})$$

For the described segmentation of multiplicands into n=3 terms the length of segments m in (10) is equal to 78 bits. Eq. (10) can also be used for the calculation of the gate complexity of other 233-bits serial multipliers, if they have the same structure. Eq. (10) doesn't contain the number of flip-flops (see Fig. 3 and 4) because each multiplier contains the same number of flip-flops for the output values independently of the implemented multiplication method.

III. Evaluation of Theoretical Data

In this section we discuss the theoretical performance parameters of the multipliers we investigated.

The execution time of the multiplication is an important parameter. We investigated the serial polynomial multipliers, which can calculate the polynomial product in less than 30 clock cycles. We calculated the area and energy consumption for the following MMs using eq. (10):

- n-term Karatsuba MM with $1<n\leq 7$
- n-term Winograd MM with $1<n\leq 7$
- generalized n-term Karatsuba MM with $1<n\leq 7$
- n-term Montgomery MM with $5\leq n\leq 7$

We obtained the multiplication formulae for the n-term Karatsuba MM in the following way: we derived the multiplication formulae for the 8-term multiplicands using the Karatsuba multiplication formula recursively. The expression for multiplication of 8-term-polynomials contains 27 partial products of m-bit long terms of multiplicands. The expression for the polynomial product of 7-term long operands can be obtained from the expression for 8-terms long operands by setting the most significant terms of both polynomials to zero and so on. We obtained the n-term Winograd MM formulae in the same way.

The serial multipliers have the same structure (see Fig. 2). The gate complexity of these multipliers does not depend on the applied MM and is approximately the same using the same segmentation and the same optimized partial multiplier (see eq. 10). The area of those multipliers is also approximately the same. Their energy consumption depends on the number of clock cycles, which are necessary to accumulate the polynomial product. The number of clock cycles is equal to the number of partial multiplications #MULT of the applied MM (see eq.5) and is given in table IV for each of the investigated MMs and also for the classical MM. The former determines the energy consumption of multipliers, i.e. it depends on the applied MM and is shown in Table V for each investigated MM.

TABLE V. Number of Clocks for Investigated MM

Investigated MM	Number of clocks for the segmentation n					
	n=2	n=3	n=4	n=5	n=6	n=7
classical	4	9	16	25	-	-
n-term Karatsuba	3	6	9	14	18	23
n-term Winograd	3	6	9	14	18	23
generalized n-term Karatsuba	3	6	10	15	21	24
n-term Montgomery	-	-	-	13	17	22

For all investigated multipliers we calculated the area and energy consumption using their gate complexity and the gate parameters of the IHP 0.13µ technology.

In addition we calculated the gate complexity of the serial multiplier that calculates the product using the classical MM. Such a multiplier doesn't contain the unit for calculating the inputs of the partial multiplier. Also its accumulation units are less complex than those needed for other MMs (see Fig. 4). We calculate the gate complexity of the serial classical multiplier using (11):

$$GC_{nm}^{clas} \approx GC_{m-TM} + (2n-2)m_{MUX} + \quad (11)$$
$$+ ((465+2nm)_{AND};(465+2nm-2m)_{XOR})$$

Fig. 5 shows the calculated area and energy consumption as a function of clock cycles required by each polynomial multiplier for the accumulation of the product. The red curve shows the energy consumption of all investigated multipliers and the green curve shows their area. The area and energy consumption of the optimized one clock 233-bits multiplier are given for comparison. Also the serial multiplier presented in [12] is shown in Fig.5.

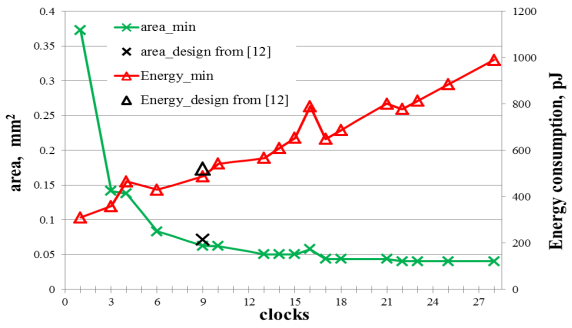

Figure 5. Calculated chip-parameters of all investigated serial polynomial multipliers

As selection criterion for the best multiplier we selected not only the area-power product but also the execution time of the multiplication. We determined 2 candidates for implementation: the 6-clock cycles multiplier using 3-term-Winograd MM (36 mm^2·pJ) and the 9-clock cycles multiplier using 4-term-Karatsuba MM (31 mm^2·pJ)[2]. Note that the area of other i-clock multipliers ($i>9$) is marginally smaller than the area of the 9-clock cycles multiplier but their energy consumption, area-power product and calculation time are much larger.

[2] For comparison, the 9-clock cycles multiplier presented in [12] has an area-power product of 37 mm^2·pJ.

For the evaluation of our theoretical results we synthesized the 6- and 9-clock cycles serial multipliers for the IHP 0.13µ technology using Synopsis tools [14]. Additionally we synthesized the design from [12] as benchmark for our practical results. Table VI shows the area and energy consumption of the synthesized polynomial multipliers and the chip parameters of the kP-designs using these polynomial multipliers.

TABLE VI. PARAMETERS OF SYNTHESIZED DESIGNS

Operation	Design details	*m*, bits	area, mm^2	Energy
polynomial multiplication	9-clock multiplier (as in [12])	64	0.1140	1.05 nJ
	9-clock multiplier (this work)	59	0.1090	0.95 nJ
	6-clockmultiplier (this work)	78	0.1323	0.72 nJ
kP	using 9-clock multiplier (from [12])	64	0.2480	2.29 µJ
	using 9-clock multiplier (this work)	59	0.2430	2.14 µJ
	using 6-clock multiplier (this work)	78	0.2662	1.74 µJ

The comparison of the chip-parameters of synthesized multipliers and kP-designs reveals:

- Applying the optimized 59-bits partial multiplier instead of the 64-bit partial multiplier in the 9-clock cycles serial polynomial multiplier reduces the area of the multiplier about 4.4% and at the same time the energy consumption about 9.5%.

- Applying our serial 6-clock cycles polynomial multiplier instead of the 9-clock multiplier from [12] reduces the energy consumption of a single polynomial multiplication about 31.4%. The energy consumption of the full kP-designs and the execution time of the kP-operation are reduced about 24% and 28%, respectively.

CONCLUSION

In this paper we have presented a theoretical approach to do early assessing of the area and power consumption of hardware accelerators for elliptic curve cryptography. We developed one clock multipliers for up to 600 bits long operands using optimal combinations of 13 multiplication methods. We have calculated the area and energy consumption of 25 serial polynomial multipliers for the multiplication in $GF(2^{233})$ that we use as an example. This approach can be applied also for the development of a serial multiplier for other operand length. We used the results to select three multipliers for further evaluation. These multipliers were synthesized and the synthesis values are given in the article. The comparison of the theoretical calculated data and the synthesized values shows that our approach provides a good and reliable means to do an early assessment for selecting implementation candidates well suited for resource constraint, low cost devices such as embedded systems or wireless sensor nodes.

REFERENCES

[1] Weimerskirch, A., Paar, C.: Generalizations of the Karatsuba Algorithm for Efficient Implementations. Report 2006/224, Cryptology ePrint Archive, 2006, http://eprint.iacr.org/2006/224.pdf

[2] Karatsuba, A., Ofman, Y.: Multiplication of Many-Digital Numbers by Automatic Computers. Doklady Akad. Nauk SSSR, Vol. 145 (1962), pp: 293–294. Translation in Physics-Doklady, 7 (1963), pp: 595–596.

[3] Winograd, S.: Arithmetic Complexity of Computations. SIAM, 1980

[4] Sunar, B.: A Generalized Method for Constructing Subquadratic Complexity $GF(2^k)$ Multipliers. IEEE Transactions on Computers, Vol. 53, Nr. 9, 2004, pp: 1097-1105

[5] Montgomery, P.L.: Five, Six, and Seven-Term Karatsuba-Like Formulae. IEEE Transactions on Computers, Vol. 54, Nr. 3, 2005, pp: 362-369

[6] Haining Fan and M. Anwar Hasan. Comments on "Five, six, and seven-term Karatsuba-like formulae". IEEE Trans. Comput., 56 (5), 2007, pp:716–717

[7] Cenk, M., Koc, C. K., Ozbudak, F.: Polynomial multiplication over finite fields using field extensions and interpolation, in 19th IEEE Symposium on Computer Arithmetic, IEEE Computer Society Press, Portland, Oregon, 2009, pp: 84-91.

[8] Oseledets, I.: Improved n-term Karatsuba-like Formulae in GF(2). IEEE Transactions on Computers, Vol. 60, Nr. 8, 2011, pp: 1212-1216

[9] J. von zur Gathen and J. Shokrollahi: Efficient FPGA-based Karatsuba multipliers for polynomials over F2, in Selected Areas in Cryptography (SAC) 12th International Workshop, Canada, 2005, Revised Selected Papers, Vol. 3897 of LNCS, Springer-Verlag, 2006, pp: 359–369

[10] Dyka, Z., Langendoerfer, P.: Algorithmically determining the optimum polynomial multiplier in $GF(2^k)$, extended abstract, WEWORC 2011, available at http://2011.weworc.org/

[11] Dyka, Z., Langendoerfer, P.: Area efficient hardware implementation of elliptic curve cryptography by iteratively applying Karatsubas method, In Proceedings of the Design, Automation and Test in Europe (DATE 2005), 2005, Vol.3, pp: 70-75

[12] Peter, S., Langendoerfer, P.: An Efficient Polynomial Multiplier GF(2m) and its Application to ECC Designs. In Proceedings of the Design, Automation and Test in Europe (DATE 2007), 2007, pp:1253-1258

[13] Innovations for High Performance Microelectronics, http://www.ihp-microelectronics.com/

[14] Synopsis, http://www.synopsys.com/