Efficient Hardware Architecture of Recursive Karatsuba-Ofman Multiplier

El hadj youssef wajih
Physic Department of Faculty of Sciences of Monastir
Electronics and Micro-Electronic Laboratory (LEME)
Monastir, Tunisia
elhadjyoussef_wajih@yahoo.fr

Zeghid Medien
Physic Department of Faculty of Sciences of Monastir
Electronics and Micro-Electronic Laboratory (LEME)
Monastir, Tunisia
LESTER-University of South Brittany
Lorient, France.
medienzeghid@gmail.com

Machhout Mohsen
Physic Department of Faculty of Sciences of Monastir
Electronics and Micro-Electronic Laboratory (LEME)
Monastir, Tunisia

Bouallegue Belgacem
Physic Department of Faculty of Sciences of Monastir
Electronics and Micro-Electronic Laboratory (LEME)
Monastir, Tunisia
LESTER-University of South Brittany
Lorient, France

Tourki Rached
Physic Department of Faculty of Sciences of Monastir
Electronics and Micro-Electronic Laboratory (LEME)
Monastir, Tunisia

Abstract— **The finite Field multiplication is the basic operation in all cryptographic applications. It can be performed by using Serial, Booth, Montgomery and Karatsuba-Ofman's divide-and-conquer technique. The Karatsuba-Ofman multiplier replaces a multiplication by three ones of half-length operands which are performed in parallel. The implementation of Karatsuba-Ofman multiplier has been made both in sequential and parallel architectures. In order to improve the performance's architectures over GF (2^m), we propose a new Sequential/Parallel architectures of Recursive Karatsuba-Ofman multiplier. In this paper, two Sequential/Parallel architectures are presented, developed and implemented on the Spartan 3 FPGA platform. Area and low Delay computation of the proposed architectures are improved. Mathematical Performances models (Area (n), Delay (n)) for large number (n) are elaborated for our proposed architectures. They can be established in order to expect the appropriate multiplier for the cryptographic applications.**

Keywords: Karatsuba-Ofman, polynomial multiplication, Galois fields

I. INTRODUCTION

In recent years, finite field multiplication in GF (2^m) has been widely used in various applications such as error-correcting code and cryptography. One of the motivations for fast and area efficient Hardware solutions for multiplications in finite fields GF (2^m) comes from the fact that they are the most time-consuming operations in cryptographic applications such as the Government Digital Signature Standard (DSS) [1] and also recently Elliptic Curve Cryptography (ECC) [2]. The multiplication is considered as the most fundamental task in arithmetic. So, the optimization is critical for overall performance of cryptographic implementation.

Since a field multiplier is crucial unit for overall performance of cryptographic implementation new multiplier's architectures whose performances can be chosen freely is necessary. In this paper two type of multiplier are treated: Parallel and Sequential multipliers. The fastest type of multiplier is the Parallel ones. Among these, the Karatsuba-Ofman Multipliers (KM) are the fastest. The Karatsuba-Ofman Algorithms (KOA) is discovered in 1962. It can successfully be applied to long integers multiplication step. The fundamental KM for polynomial in GF (2^m) is based on the idea of divide-and-conquer [3-4]. However the KM suffers from a bad Area occupation. The serial multiplier in the other hand is a sequential multiplier [5]. It requires smaller area and have a less complex structure, but generate partial product in each clock cycle. New proposed Sequential/Parallel recursive Karatsuba-Ofman architectures are developed an optimized. Only two architectures intended to perform the design of the cryptographic applications are presented in this paper.

The remainder of this paper is organized as follows: Section 2 describes the Karatsuba-Ofman Algorithm. The Serial multiplier is illustrated in section 3. Section 4 presents new approaches of adapted KOA, so that it can be implemented efficiently. Experimental results are given in Section 5.

978-1-4244-1577-9/08/$25.00 ©2008 IEEE

Mathematical models of Serial multiplier as well as our proposed Sequential/Parallel architecture's performances (Area, Delay) are elaborated in Section 6. Section 7 concludes the paper.

II. KARATSUBA-OFMAN MULTIPLICATION (KM)

Karatsuba-Ofman's algorithm is considered as one of the fastest ways to multiply long integers [4]. In this section, we introduce the fundamental KOA. It can successfully be applied to polynomial multiplication step.

The fundamental Karatsuba-Ofman Multiplication for polynomial in GF (2^m) is a recursive divide-and-conquer technique. It replaces a multiplication by three multiplications of half-length operands [6]. The KOA becomes recursive if m/2 is even. A straightforward application of the KOA requires $\log_2(m)$ iteration steps for polynomials of degree m-1 [7].

Let $A = (a_0, a_1, ..., a_{m-1})$ and $B = (b_0, b_1, ..., b_{m-1})$, the binary representation of two integers. The operands A and B can be decomposed into two equal-size parts A_1 and A_0, B_1 and B_0 respectively, which represent the $m/2$ higher and lower order bits of A and B. We can split them in two parts as follows:

$$A(x) = \sum_{i=0}^{m-1} a_i x^i = \sum_{i=\frac{m}{2}}^{m-1} a_i x^i + \sum_{i=0}^{\frac{m}{2}-1} a_i x^i$$

$$= x^{\frac{m}{2}} \sum_{i=0}^{\frac{m}{2}-1} a_{i+\frac{m}{2}} x^i + \sum_{i=0}^{\frac{m}{2}-1} a_i x^i = x^{\frac{m}{2}} A_1 + A_0 \quad (1)$$

$$B(x) = \sum_{i=0}^{m-1} b_i x^i = x^{\frac{m}{2}} B_1 + B_0 \quad (2)$$

So the product $P = AB$ can be computed as follows:

$$A(x)B(x) = A_0B_0 + A_1B_1x^m + (A_1B_0 + A_0B_1)x^{m/2} \quad (3)$$

To improve the computation of the product P, (3) can be modified to (4) which requires three ($m/2$-*bits*) multiplications. In GF (2^m) the product A(x)B(x) is given by the equation :

$$A(x)B(x) = A_0B_0 + A_1B_1x^m + (A_0B_0 + A_1B_1 + (A_1 + A_0)(B_1 + B_0))x^{m/2} \quad (4)$$

The result is:

$$A(x)B(x) = C_0 + C_1x^{m/2} + C_2x^m \quad (5)$$

Field multiplication can be performed in two steps. Firstly, we perform an ordinary polynomial multiplication of two field elements as shown in (5). Then, we have a number that is (2m-1) bits long. In second step, this number must be reduced with the irreducible polynomial. The reduction only needs (m+k-1)/2 XOR gates for irreductible trinomials $F(x) = x^m + x^k + 1$. However, we want to focus on an efficient method based on KOA to calculate the polynomial multiplication. Therefore we only treat polynomial multiplication step in this paper. Fig. 1 shows the procedure of KOA where A, B are two m-bit length polynomials:

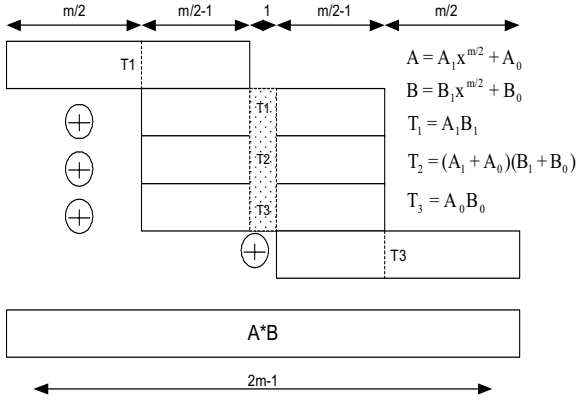


Figure 1. Karatsuba-Ofman's multiplication.

The multiplication over GF (2^m) is computed by an AND and XOR operations. According to (4), KOA can be applied recursively to the three polynomial multiplications. Hence, we can postpone the computations of the polynomial products A_0B_0, A_1B_1 and $(A_1+A_0)(B_1+B_0)$; and instead we can split again each one of these three factors into three polynomial products.

By applying recursively this strategy, each polynomial multiplication is transformed into three polynomial ones with their degrees reduced to about half of its previous value [8-9].The structural description of the recursive KOA is shown in Fig. 2.

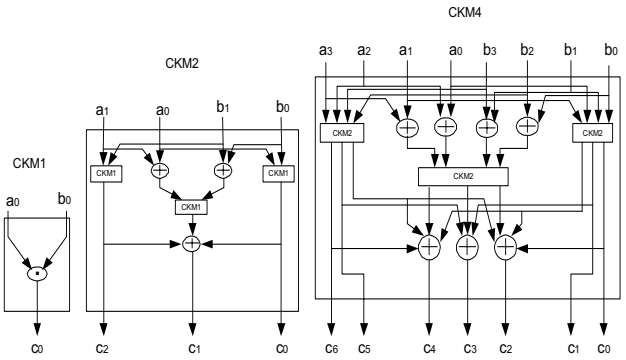
```
Algorithm KOA(X, Y)
if (Size(X) = 1) then
   Result: OneBitMultiplier (X, Y)
Else
 SumX := Hiqh(X) + Low(X) ;
 SumY := High(Y) + Low(Y) ;
 KOA 1 : KOA (Hiqh(X), High(Y), Productl);
 KOA 2 : KOA (Low(X),  Low(Y), Product2);
 KOA 3 : KOA (SumX, SumY, Product3);
Rigthshiftadd1 : RightShiftadd (Product1, Size(X)) +
RightShift (Product3 – Product1 – Product2,
        Size(X)/2) + Product2, Result;
end algorithm.
```

Figure 2. KOA Algorithm.

The recursive construction process of Karatsuba-Ofman Multiplication (RKM), consists of Combinational Karatsuba Multipliers (CKMs) of width $m = 2^i$ for arbitrary $i \in N$ (see Fig. 3) [10].

The KOA becomes recursive if n/2 is even. As can be concluded, Parallel multipliers (such as RKM) perform partial products in minimum clock cycle but require more space to be implemented [11].

So, new RKM architecture is needed. Performances such as Delay (ns) and Area occupation (slices) can be adjusted to meet the constraints.



a) CKM1 : 1 bit b) CKM2: 2 bit c) CKM4 : 4bit

Figure 3. Recursive Karatsuba-Ofman's multiplication (RKM).

III. SERIAL MULTIPLICATION

Multiplication in GF (2^m) with polynomial basis representation is defined as follows: Inputs $A = (a_0, a_1, ..., a_{m-1})$ and $B=(b_0,b_1, ...,b_{m-1}) \in$ GF(2^m). The product C=AB is given by $C(x) = A(x).B(x) \bmod F(x)$, where $F(x)$ is a constant irreducible polynomial of degree m (see Fig. 4) [5].

Algorithm serial (A(x), B(x), F(x))
C(x) <= 0
for i from m - 1 down to 0 do
C(x) <= C(x) x + A(x)bi
C(x) <= C(x) + c_m F(x)
End for
Return C(x)

end algorithm.

Figure 4. Serial multiplier algorithm.

The structural description of the Serial multiplier is shown in Fig. 5.

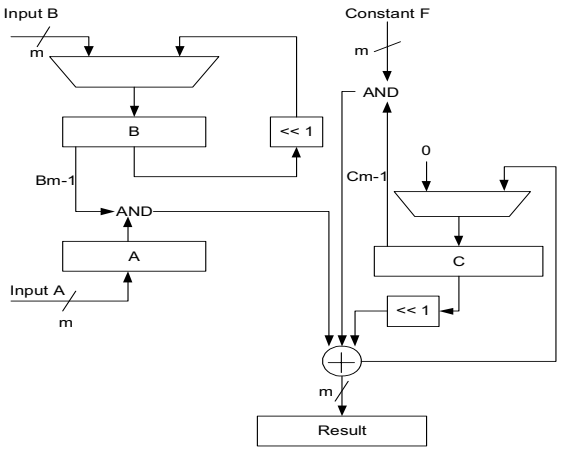


Figure 5. Serial Multiplier architecture.

The Serial multiplier is slow because it takes m clock cycles to generate the product of two field elements. It consists of some simple multiplexers, a simple Combinational and right Shift operations. These operations are simple to implement and do not require a lot of Area [12].

IV. PROPOSED RKM ARCHITECTURES

Parallel multipliers perform partial products in minimum clock cycle but require more space to be implemented. Serial multipliers in the other hand require smaller area and have a less complex structure, but generate partial product in each clock cycle. In this section, we present new hybrid RKM architectures that can be easily adapted to various performance delay, area and power constraints for 128 bits multiplication. A block diagram of 128-bits RKM's architecture is developed as shown in Fig. 6

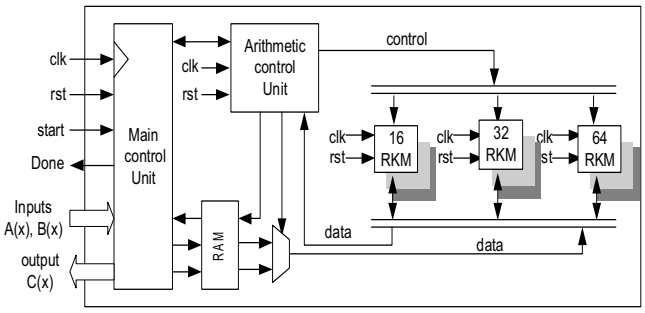
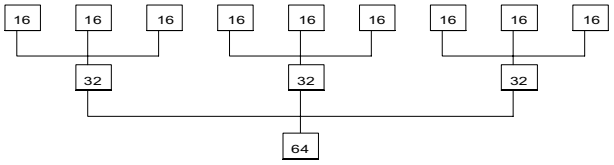
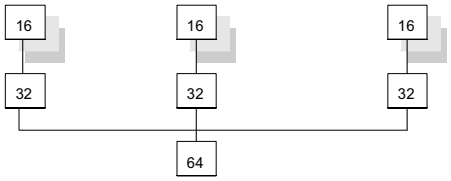


Figure 6. Architecture of Recursive 128-bit Karatsuba-Ofman's multiplication (RKM).

The major purpose of the proposed approaches is to obtain a small Area occupation of the circuit due to the fact that partial polynomial multiplications can be applied serially. The second purpose is to obtain a high speed computation by performing Parallel multiplication. In the following, two new schemes of 64-bits RKM architectures based on sequential and parallel applications of RKM were proposed and implemented as shown in Fig. 7.



a) 64-bit RKM1 (A1) (Parallel architecture)

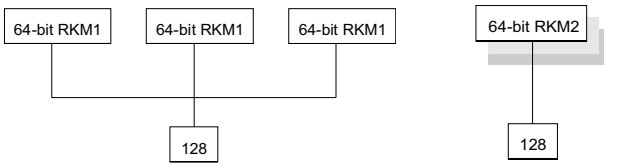


b) 64-bit RKM2 (A2) (Serial architecture)

Figure 7. A 64-bit sequential and parallel RKM using: a)Three parallel 32-bit and three parallel 16-bit RKM multipliers b) A 64-bit Three parallel 32-bit and three sequential 16-bit RKM multipliers.

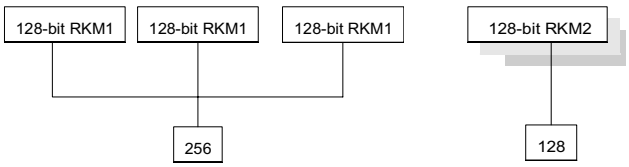
To perform a 64-bit RKM1 by applying a parallel architecture (A1), the parallel architecture of a 64-bit RKM is composed by: three 32-bits multiplier performed in parallel. Each multiplier requires three 16-bits RKM multiplier applied in parallel. The second architecture of 64-bits RKM2 each 32-bits multiplier uses one 16-bits multiplier applied sequentially.

The same technique is applied for 128-bits RKM1, 128-bits RKM2, 256-bits RKM1 and 256-bits RKM2 (see Fig. 8, 9).



a) 128-bit RKM1 (A3) b) 128-bit RKM2 (A4)

Figure 8. A 128-bit RKM using: a) 64-bit RKM1 b) 64-bit RKM2.



a) 256-bit RKM1 b) 256-bit RKM2

Figure 9. A 256-bit multiplication using : a) 128-bit RKM1 b) 128-bit RKM2.

V. EXPERIMENTAL RESULTS

All proposed algorithms of the multipliers for arbitrary (n) are simulated by using VHDL language and Xilinx ISETM tools V 6.1, implemented and synthesized on the Spartan 3s2000fg900-5 FPGA. Depending on the constraints imposed by the design specifications, the performance of the proposed Parallel/Serial RKM's architectures are measured by means of propagation Delay (ns), Area occupation (Slices) and Power consumption (mW) of all proposed architectures. The synthesis performances analyses for the 128-bits RKM's implementations are shown in Fig. 10.

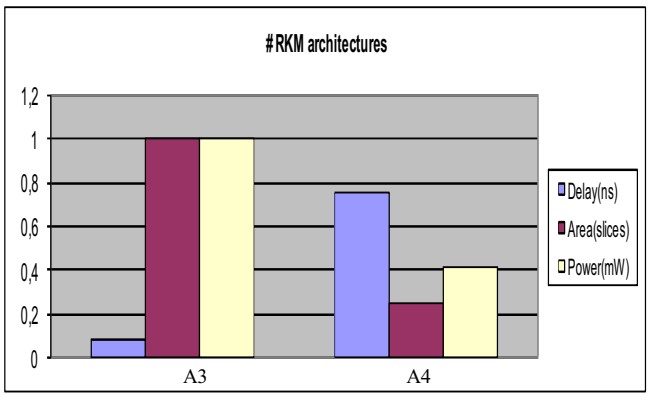


Figure 10. Normalized Delay, Area and Power histogram for the two 128-bit RKM's architectures on Spartan 3.

From Fig. 10, we notice that for Spartan 3 target device the architecture (A3) presents the minimum delay, however architecture A4 have the least area (slices) occupation.

In the following, the performances of the proposed (A3) and (A4) RKM's architectures are compared with the Serial multiplier. As shown in Table 1, the Serial multiplier implementation requires less area but it's slow because it takes m clock cycles to generate the product of two field elements.

TABLE I. AREA AND DELAY PERFORMANCES OF SERIAL, RKM-A3 AND RKM-A4 MULTIPLIERS ON SPARTAN 3.

Bit-length	Serial multiplier		RKM A3		RKM A4	
	Area (Slices)	Delay (ns)	Area (Slices)	Delay (ns)	Area (Slices)	Delay (ns)
32	182	278,66	1027	40,77	502	154,16
64	303	600,47	3268	49,9	1750	157,15
128	544	1323,29	10172	59,52	2528	515,64
256	1045	2855,87	-	69,77	8276	569,04

For 128-bits length, the implementation of these multiplier's architectures on Spartan 3 platform shows that: the A3 RKM's architecture multiplier is 22-times faster than Serial multiplier. Table 2 illustrates Area occupation and Delay reports of RKM by Serial multiplier's of the proposed RKM's architectures on Spartan 3.

TABLE II. AREA AND DELAY REPORT FOR THE TWO 128-BIT RKM'S ARCHITECTURES ON SPARTAN 3.

Architecture	Area report (RKM/Serial)	Delay report (Serial/ RKM)
A3	33,57096	22,23239
A4	8,34323	2,56631

Compared to the serial multiplier, the two RKM architectures A3 and A4 required more space Area but they are the fastest. Nevertheless, the choice of architecture A3 or A4 is imposed by the cryptographic application and by the constraints of the design specifications.

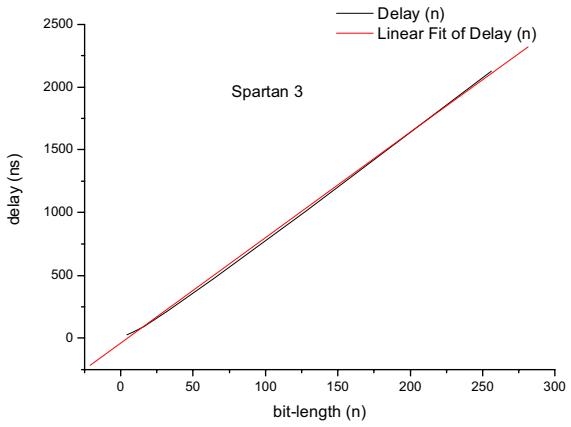
VI. ARCHITECTURE'S MODELIZATION

In this section, we present different mathematical models of our architectures performances (Area (n), Delay (n)) implemented on Spartan 3 FPGA. Mathematical models of Serial multiplier, (A3) and (A4) RKM's architectures are provided. A Clear idea is available for FPGA's user about area occupation and elapsed time of proposed RKM's architectures for large bit range.

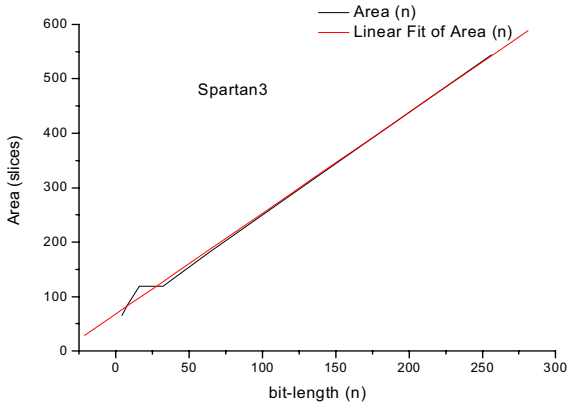
A. Serial architecture model

Fig. 11 shows the variation of Delay and the Area occupation of serial multipliers for different bit-lengths (n). These performances have linear pace. The linear fit of Delay and Area in Fig. 11 a) and b), is given by the equation $Y = A + B.n$, where the parameters a and B are by the method of least squares and they depend on the design specifications.

The determination of the values of A and b allows estimating the Area and the Delay of a Serial multiplier for any large number.



a)

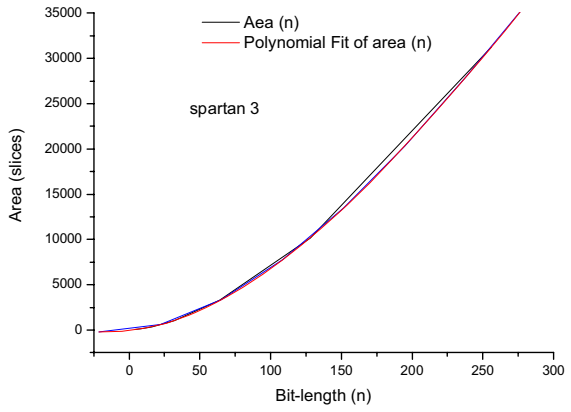


b)

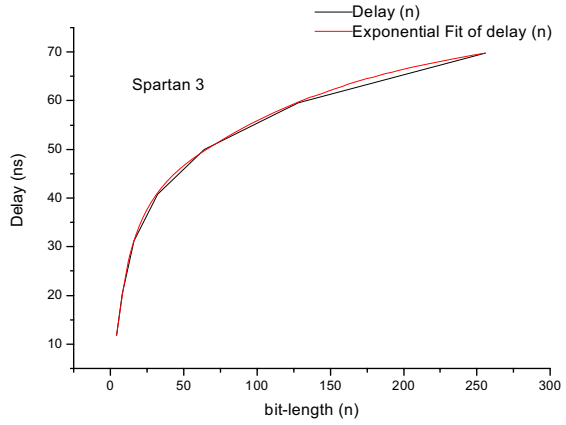
Figure 11. Serial 128-bit multiplications requirements: **a)** Delay (n) on Spartan 3 **c)** Area (n) on Spartan 3.

B. RKM architecture models

From the Fig. 12, we notice that the Area variation curves of RKM architecture (A3) have a third order polynomial fit. The Delay one has a second order Exponential fit.



a)



a)

Figure 12. A128-bit RKM requirements of architecture (A3) on Spartan 3: a) Area (n) b) Delay(n) .

These models are available for designers before hardware and software RKM's implementation. So, we can have a clear idea about the previewed performances in order to have suitable trade-offs between Area and Speed for implementing cryptographic applications.

VII. CONCLUSION

In this paper we have presented and developed two new RKM architectures for cryptographic applications. For each architecture, we detailed the hardware design and then compared them with respect to Area and Delay requirements.

Finally, a mathematical models of Delay (n) and Area (n) performances for both Serial architecture and our proposed RKM architecture (A3) are elaborated. Such models can provide expectations about the performances of multipliers architectures to have suitable trade-offs between Area and Speed for implementing cryptographic schemes in embedded systems.

REFERENCES

[1] Recommended Elliptic Curves for Federal Government Use, July 1999.Appendix 6 to FIPS publication 186-2, Digital Signature Standard (DSS), US Dept. of Commerce, Nat'l Inst. of Standards and Technology, Jan. 2000, http://www.itl.nist.gov/fipspubs/fip186.htm.

[2] Z. Dyka, P. Langendoerfer, "Area efficient hardware implementation of elliptic curve cryptography by iteratively applying karatsuba's method", Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05), Compute Society pages 70–75, 2005.

[3] P. L. Montgomery, "Five, Six, and seven-Term Karatsuba-Like Formulae," IEEE Transactions on Computers, vol. 54, no. 3, pp. 362–369, March 2005.

[4] N. Nedjah, L. de Macedo Mourelle, "A Review of Modular Multiplication Methods and Respective Hardware Implementations", Informatica 30 (2006) 111–129.

[5] Dong-Ho Lee Jong-Soo Oh, "Multi-Segment GF(2^m) Multiplication and its Application to Elliptic Curve Cryptography", GLSVLSI'07, Stresa-Lago Maggiore, Italy, pages 546-551, March 11–13, 2007.

[6] S. Peter, P. Langendorfer, "An Efficient Polynomial Multiplier in GF (2^m) and its Application to ECC Designs", 978-3-9810801-2-4/DATE07 © 2007 EDAA.

[7] J. von zur Gathen and J. Shokrollahi, "Efficient FPGA-based Karatsuba multipliers for polynomials over F2," B. Preneel and S. Tavares (Eds.): SAC 2005, LNCS 3897, pp. 359–369, 2006.

[8] L. A. B. Kowada, R. Portugal, C. M. H. Figueiredo, "Reversible Karatsuba's Algorithm", Journal of Universal Computer Science, vol. 12, no. 5 (2006), 499-511.

[9] L.S. Cheng, A. Miri, and T.H. Yeap, "Improved FPGA implementations of parallel Karatsuba multiplication over GF (2^n)", In 23rd Biennial Symposium on Communications, 2006.

[10] M. Ernst, M. Jung, F. Madlener, S. Huss, R. Blumel, "A Reconfigurable System on Chip Implementation for Elliptic Curve Cryptography over GF(2^n)", CHES 2002, LNCS 2523, pp. 381–399, 2003.

[11] N.S. Chang, C.H. Kim, Y.-H. Park, and J. Lim, "A non-redundant and efficient architecture for Karatsuba-Ofman algorithm", In ISC, pages 288–299, 2005, Springer-Verlag Berlin Heidelberg 2005.

[12] Manfred Schimmler, Bertil Schmidt, Hans-Werner Lang, Sven Heithecker, " An area-efficient bit-serial integer and GF(2^n) multiplier", Microelectronic Engineering 84 pp. 253–259, 2007.