ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Design and Implementation of Mastrovito Multiplier using New Serial-Out Bit-Level

Chaitanya Jami
M.Tech (VLSI),
Dept of ECE,
Baba Institute of Technology & Sciences.

P.K.Suresh
Associate Professor,
Dept of ECE,
Baba Institute of Technology & Sciences.

V Ramagowri Bobbili
Assistant Professor,
Dept of ECE,
Baba Institute of Technology & Sciences.

Abstract:
The Serial-out bit-level multiplication scheme is characterized by an important latency feature. It has an ability to sequentially generate an output bit of the multiplication result in each clock cycle. However, the computational complexity of the existing serial-out bit-level multipliers in GF (2^m) using normal basis representation, limits its usefulness in many applications; hence, an optimized serial out bit-level multiplier using polynomial basis representation is needed. In this paper, we propose new serial-out bit-level Mastrovito multiplier schemes. We show that in terms of the time complexities, the proposed multiplier schemes outperform the existing serial out bit-level schemes available in the literature. In addition, using the proposed multiplier schemes, we present new hybrid-double multiplication architectures. To the best of our knowledge, this is the first time such a hybrid multiplier structure using the polynomial basis is proposed. Prototypes of the presented serial-out bit-level schemes and the proposed hybrid-double multiplication architectures (10 schemes in total) are implemented over both GF (2^{163}) and GF (2^{233}), and experimental results are presented

Keywords:
Mastrovito multiplier, hybrid-double multiplication architectures.

I .INTRODUCTION:

FINITE field arithmetic has been widely applied in applications of different fields like error-control coding, cryptography, and digital signal processing.

The arithmetic operations in the finite fields over characteristic two GF (2m) have gained widespread use in public-key cryptography such as point multiplication in elliptic curve cryptography, and exponentiation-based cryptosystems. The finite field GF (2m) has 2melements and each of its elements can be represented by its m binary coordinates based on the choice of field-generating polynomial. For such a representation, the addition is relatively straight-forward by bit-wise XORing of the corresponding coordinates of two field elements. On the other hand, the multiplication operation requires larger and slower hardware. Other complex and time-consuming operations such as exponentiation, and division/inversion are implemented by the iterative application of the multiplication operations.

Much of the ongoing research in this area is focused on finding new architectures to implement the arithmetic multiplication operation more efficiently (see for example). Finite field multipliers with different properties are obtained by choosing different representations of the field elements. With the advantages of low design complexity, simplicity, regularity, and modularity in architecture, the standard or polynomial basis (PB) representation, is extensively used for cryptographic applications. In the PB, a multiplier requires a polynomial multiplication followed by a modular reduction. In practice, these two steps can be combined into a single step by using the so-called Mastrovito matrix [14], [15]. The properties and complexities of the PB multipliers depend heavily on the choice of a field-generating polynomial.

In this paper, we first consider an irreducible polynomial with !, ! 3, non-zero terms (denoted by !-nomials). We then obtain a further optimized structure for the special irreducible trinomial (! = 3). The implementation of finite field multipliers can be categorized, in terms of their structures, into three groups of parallel-level, digit-level and bit-level types. The bit-level multiplier scheme, which processes one bit of input per clock cycle, is area-efficient and suitable for resource-constrained and low-weighted devices. The bitlevel type multiplication algorithms, when the PB is used are classified as least significant bit first (LSB-first), and most significant bit first (MSB-first) schemes.

II. MASTROVITO MULTIPLIER:

Software and hardware implementations of the basic arithmetic operations (addition, multiplication, and inversion) in the Galois field GF(2m) are desired in coding theory, computer algebra, and cryptography [7, 4]. The cryptographic applications include elliptic curve cryptosystems [8, 2], in which m is quite large, usually around several hundreds. The efficiency of an algorithm is often measured by the number of bit-level or word-level operations. In the hardware implementations, it is often desired to reduce the total number of gates (space complexity) and the total gate delay (time complexity) of the algorithm. The representation of the field elements has a crucial role in determining the space and time complexity of the arithmetic operations, particularly the field multiplication. In this paper, we are interested in space and time complexity of the finite field multiplication operation, where the field elements are represented using the standard basis. The standard basis multiplication operation in GF(2m) is often accomplished in two steps: polynomial multiplication and modular reduction. Let a(x), b(x), c(x) ∈ GF(2m) and p(x) be the irreducible polynomial generating GF(2m). In order to compute c(x) = a(x)b(x) mod p(x), we first obtain the product polynomial d(x) which is of degree (at most) 2m − 2 as

$$d(x) = a(x)b(x) = \left(\sum_{i=0}^{m-1} a_i x^i\right)\left(\sum_{i=0}^{m-1} b_i x^i\right)$$

The next step is then the reduction operation c(x) = d(x) mod p(x) to obtain the m − 1 degree polynomial c(x). In practice, the multiplication and the reduction steps are often combined for efficiency reasons. An architecture for performing the field multiplication was proposed by Mastrovito [5, 6]. In this method, we represent the computation of d(x) as a matrix-vector product d = Mb, where (2m − 1) × m dimensional matrix M consists of the coefficients of the polynomial a(x). We then obtain an m × m dimensional matrix Z by reducing the matrix M using the generating polynomial p(x). The product c(x) is computed using the matrix-vector product c = Zb. The space complexity of the multiplier for the special generating trinomial xm+x+1 is shown to be m2−1 XOR and m2 AND gates [5, 6, 9, 10]. Paar [11] conjectured that the space complexity of the Mastrovito multiplier would be the same for all trinomials xm+xn +1, where 1 ≤ n ≤ m−1.

In this project, we describe an architecture for the Mastrovito type multiplier using a general trinomial of the form xm + xn +1, and show that the proposed architecture requires m2 − 1 XOR and m2 AND gates when n _= m/2. However, when m is even and n = m/2 there is further reduction: The proposed architecture requires only m2 − m/2 XOR gates. A few examples of irreducible polynomials of the form xm + xn +1 are given in Table 1. Furthermore, it is known [7] that a trinomial of the form xm + xm/2 +1 is irreducible over GF(2) if m is of the form m = 2· 3r for some r ≥ 0.

irreducible $x^m + x^n + 1$ encoded as (m, n)	
(4,3), (4,1)	(10,7), (10,3)
(5,3), (5,2)	(11,9), (11,2)
(6,5), (6,3), (6,1)	(12,9), (12,7), (12,5), (12,3)
(7,6), (7,4), (7,3)	(14,9), (14,5)
(9,8), (9,5), (9,4), (9,1)	(15,14), (15,11), (15,8), (15,7), (15,4), (15,1)

Table 1: Examples of irreducible polynomials.

III. ELLIPTIC CURVE SCALAR MULTIPLICATION

Considering the area overhead, this paper uses only one FF MAC and one FF squarer to achieve high performance ECSM. The FF MAC merges addition with the reduction in FF multiplication. This brings advantages that no extra clock cycle is needed for addition, and it would not deteriorate the critical-path delay. The multiplier can be implemented in serial, parallel, or digit-serial way. The serial and digit-serial multipliers consume less area, but require more clock cycles. For high-speed consideration, we use the parallel multiplier, which takes only one clock cycle to finish one multiplication. Later, we will talk about area reduction using the Karatsuba–Ofman algorithm.

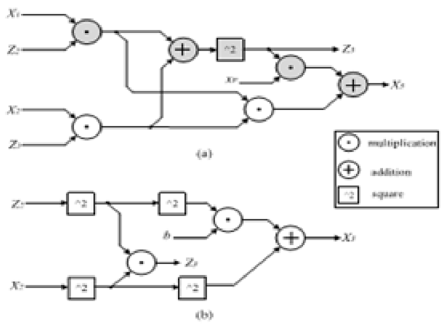


Fig 1. Data dependence graph of (a) point addition and (b) point doubling in the Montgomery ladder algorithm

IV. Proposed Architecture of Elliptic Curve Scalar multiplication

The post process stage of ECSM also requires careful consideration. While this stage is not the crucial part of ECSM, its optimization goal is to share the data path with the main loop as much as possible, rather than to reduce the required number of clock cycles. After proper scheduling of ECSM, we propose the high-performance architecture based on the improved Montgomery ladder scalar multiplication algorithm, as shown in Fig. 2.

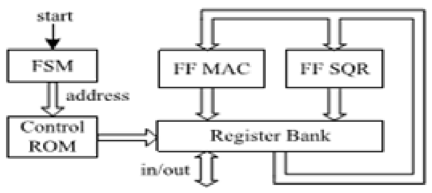


Fig. 2. Proposed architecture of ECSM.

The proposed ECSM architecture consists of one bit-parallel FF MAC, one FF squarer, a register bank, a finite-state machine, and a 6×18 control ROM. The FF MAC is implemented using the Karatsuba–Ofman algorithm, and is well pipelined. Then-stage pipelined FF MAC takes nclock cycles to finish one multiplication. The FF squarer is not pipelined, and one clock cycle is required to finish one square. The inputs to FF MAC, A, B,andC, and the input to FF squarer, S, are all registered. Another four registers T1, T2, T3,andT4 are used in the data path for data caching. Each register has a MUX before it. The control signals of MUXs are given at each clock cycle to switch between different operations in ECSM. The inputs fed to MUXs of registers are carefully allocated with the guideline that each MUX contains at most four branches. In this way, the input delay for registers is only the delay of a 4:1 MUX.

The control signals are different at every clock cycle for each iteration of the main loop and the post process stage. A heavy state machine is required to provide all the control signals in sequence. Here, we use a 6×18 control ROM to store the control signals for MUXs (16 bits) and the swapping selection (2 bits). A small state machine is used for conditional branching and jumping, and is providing the 6-bit address to the control ROM. For the FPGA implementation, the control ROM can be realized using Block RAMs in Xilinx devices or embedded memory blocks in Altera devices. Thus, it does not consume logic resources in FPGA. The data path of ECSM using a three-stage pipelined FF MAC is given for example in Fig. 3.

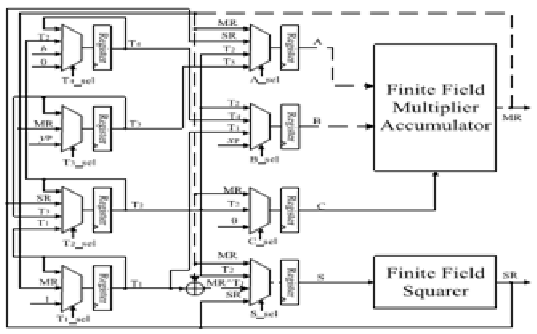


Fig. 3. Data path of ECSM using a three-stage pipelined FF MAC.

The termsX1, X2, Z1, and Z2 are not presented, because they are the intermediate results of the FF MAC or FF Squarer. The bold dashed line in Fig. 6 shows the critical path of the three-stage pipelined architecture, which consists of a pipelined FF MAC, an addition (XOR), and a 4:1 MUX. Data paths with other pipeline stages are similar to Fig. 6 except for different data connections. Control signals stored in the control ROM are also different. But, the critical path delay remains unchanged.

V. FPGA Design Flow

5.1 Design Entry:

There are different techniques for design entry. Schematic based, Hardware Description Language and combination of both etc. Selection of a method depends on the design and designer. If the designer wants to deal more with Hardware, then Schematic entry is the better choice. When the design is complex or the designer thinks the design in an algorithmic way then HDL is the better choice. Language based entry is faster but lag in performance and density. HDLs represent a level of abstraction that can isolate the designers from the details of the hardware implementation. Schematic based entry gives designers much more visibility into the hardware. It is the better choice for those who are hardware oriented. Another method but rarely used is state-machines. It is the better choice for the designers who think the design as a series of states. But the tools for state machine entry are limited. In this documentation we are going to deal with the HDL based design entry.

5.2 Synthesis:

The process which translates VHDL or Verilog code into a device netlist format. i.e. a complete circuit with logical elements(gates, flip flops, etc…) for the design. If the design contains more than one sub designs, ex. to implement a processor, we need a CPU as one design element and RAM as another and so on, then the synthesis process generates netlist for each design element Synthesis process will check code syntax and analyze the hierarchy of the design which ensures that the design is optimized for the design architecture, the designer has selected. The resulting netlist(s) is saved to an NGC(Native Generic Circuit) file (for Xilinx® Synthesis Technology (XST)).

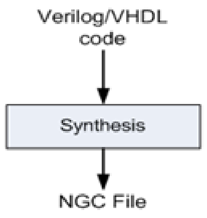


Figure 4 FPGA Synthesis

5.3 Implementation:

In this work, design of a DWT and IDWT is made using Verilog HDL and is synthesized on FPGA family of Spartan 3E through XILINX ISE Tool. This process includes following:

- Translate
- Map
- Place and Route

5.4 Translate:

Process combines all the input netlists and constraints to a logic design file. This information is saved as a NGD (Native Generic Database) file. This can be done using NGD Build program. Here, defining constraints is nothing but, assigning the ports in the design to the physical elements (ex. pins, switches, buttons etc) of the targeted device and specifying time requirements of the design. This information is stored in a file named UCF (User Constraints File). Tools used to create or modify the UCF are PACE, Constraint Editor etc.

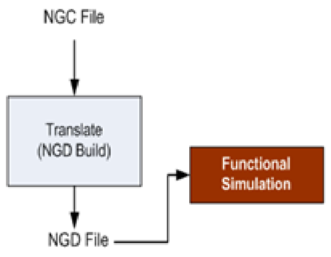


Figure 5 FPGA Translate

5.5 Map:

Process divides the whole circuit with logical elements into sub blocks such that they can be fit into the FPGA logic blocks. That means map process fits the logic defined by the NGD file into the targeted FPGA elements (Combinational Logic Blocks (CLB), Input Output Blocks (IOB)) and generates an NCD (Native Circuit Description) file which physically represents the design mapped to the components of FPGA. MAP program is used for this purpose.

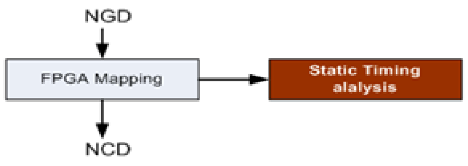


Figure 6 FPGA map

5.6 Place and Route:

PAR program is used for this process. The place and route process places the sub blocks from the map process into logic blocks according to the constraints and connects the logic blocks. Ex. if a sub block is placed in a logic block which is very near to IO pin, then it may save the time but it may affect some other constraint. So tradeoff between all the constraints is taken account by the place and route process The PAR tool takes the mapped NCD file as input and produces a completely routed NCD file as output. Output NCD file consists the routing information.

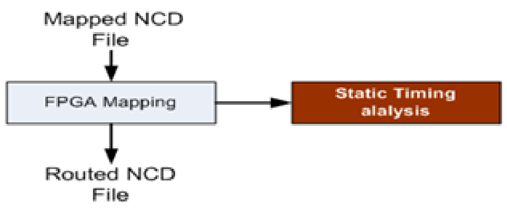


Figure 7 FPGA Place and route

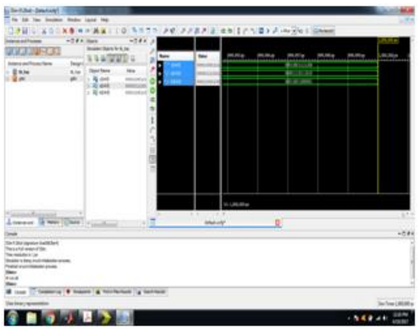
5.7 Device Programming:

Now the design must be loaded on the FPGA. But the design must be converted to a format so that the FPGA can accept it. BITGEN program deals with the conversion. The routed NCD file is then given to the BITGEN program to generate a bit stream (a .BIT file) which can be used to configure the target FPGA device. This can be done using a cable. Selection of cable depends on the design.

5.8 Behavioral Simulation (RTL Simulation):

This is first of all simulation steps; those are encountered throughout the hierarchy of the design flow. This simulation is performed before synthesis process to verify RTL (behavioral) code and to confirm that the design is functioning as intended. Behavioral simulation can be performed on either VHDL or Verilog designs. In this process, signals and variables are observed, procedures and functions are traced and breakpoints are set. This is a very fast simulation and so allows the designer to change the HDL code if the required functionality is not met with in a short time period. Since the design is not yet synthesized to gate level, timing and resource usage properties are still unknown.

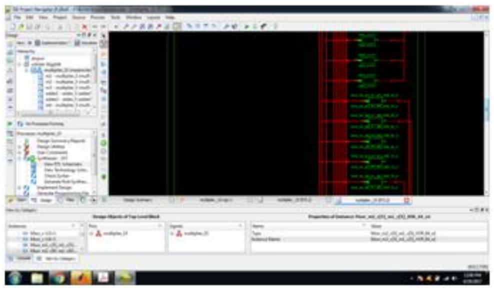
VI. SIMULATION RESULT

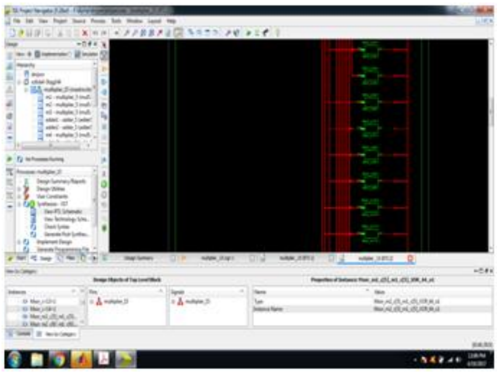


ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

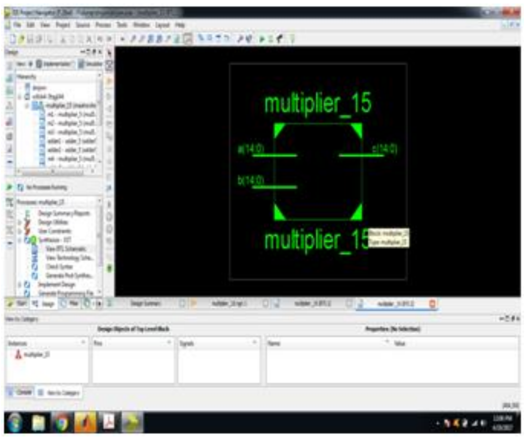
SYNTHESIS RESULTS:

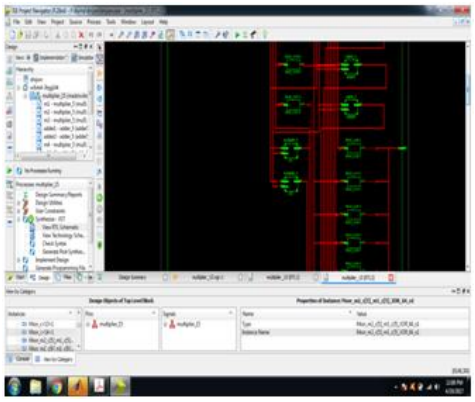
The developed project is simulated and verified their functionality. Once the functional verification is done, the RTL model is taken to the synthesis process using the Xilinx ISE tool. In synthesis process, the RTL model will be converted to the gate level netlist mapped to a specific technology library. Here in this Spartan 3E family, many different devices were available in the Xilinx ISE tool. In order to synthesis this design the device named as “XC3S500E” has been chosen and the package as “FG320” with the device speed such as “-4”. This design is synthesized and its results were analyzed as follows



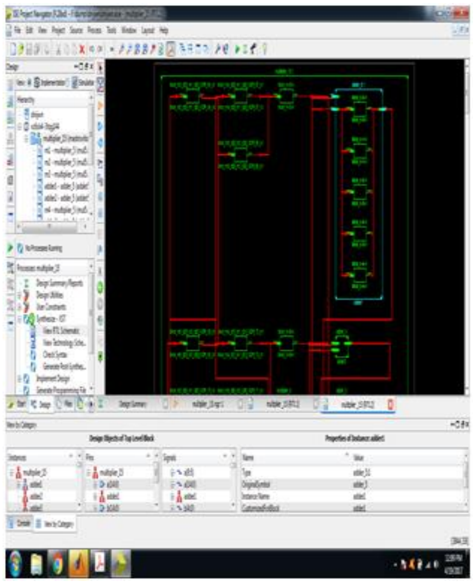


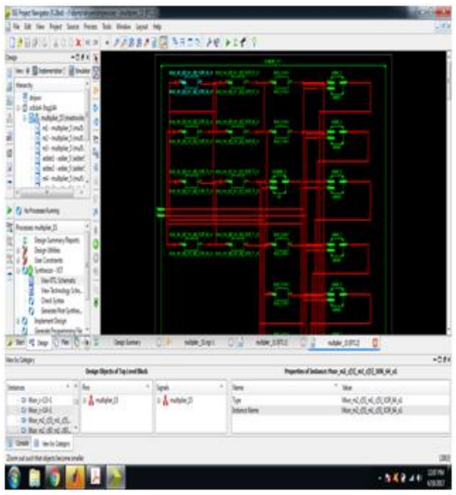
RTL SCHEMATIC:





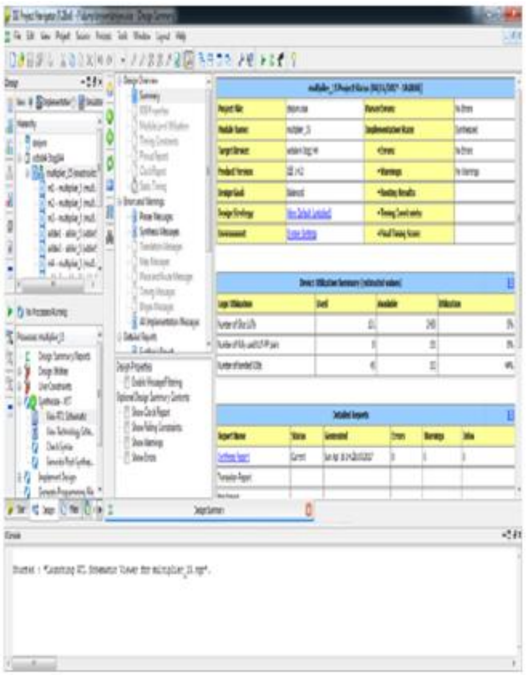
TECHNOLOGY SCHEMATIC:



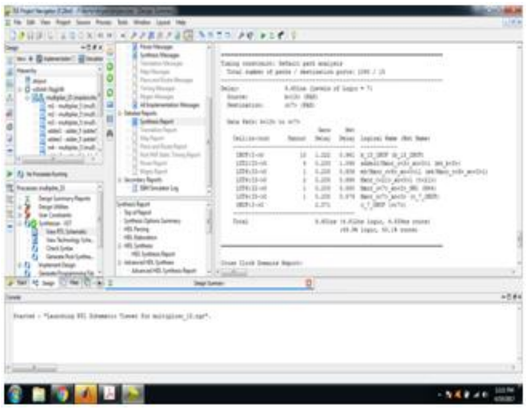


ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

DESIGN SUMMARY:



TIMING REPORT:



VII. CONCLUSION:

We have presented new hardware schemes for the serialout bit-level (SOBL) multiplier in PB representation over GF(2m) for both the !-nomial and the irreducible trinomial. Compared to previously published results in terms of time complexities, the work presented here outperform the existing SOBL multiplier schemes. We have also extended the traditional POBL multiplier schemes to new POBL double multiplication architectures, which perform two multiplications after 2m clock cycles.

Then, we proposed three hybrid-double multiplication architectures in PB over GF(2m). These hybrid multiplier structures perform two multiplications with latency comparable to the latency of a single multiplication, i.e., after m + 1 clock cycles. We have obtained the space and time complexities of the presented multipliers and have compared them with their counterparts. For the practical purposes, all the 10 schemes presented in this work have been implemented in ASIC technology over both GF(2163) and GF(2233), and the area, timing, power consumption, and energy results have been presented.

REFERENCES:

[1] R. Lidl, and H. Niederreiter, Introduction to Finite Fields and Their Applications. 2nd Ed., Cambridge Univ. Press, Cambridge, UK, Aug. 1994.

[2] R. E. Blahut, Theory and Practice of Error Control Codes. Addison- Wesley, Reading, MA, May 1983.

[3] A. J Menezes, I. F. Blake, X. Gao, R. C. Mullin, S. A. Vanstone, and T. Yaghoobian, Applications of Finite Fields. Kluwer Academic Publishers, Boston, MA, 1993.

[4] R. E. Blahut, Fast Algorithms for Digital Signal Processing. 1st Ed., Addison-Wesley, Reading, MA, Sept. 1985.

[5] V. S. Miller, “Use of Elliptic Curves in Cryptography,” In Proc. of Advances in Cryptology-CRYPTO’85, LNCS, 1986, vol. 218, pp. 417-426.

[6] N. Koblitz, “Elliptic Curve Cryptosystems,” Mathematics of Computation, vol. 48, no. 177, pp. 203-209, Jan. 1987.

[7] T. Elgamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms,” IEEE Trans. Inf. Theory, vol. 31, no. 4, pp. 469-472, Jul. 1985.

[8] W. Diffie, and M. Hellman, "New Directions in Cryptography," IEEE Trans. Inf. Theory, vol. 22, no. 6, pp. 644-654, Nov. 1976.

[9] M. A. Hasan, A. H. Namin, and C. Negre, "Toeplitz Matrix Approach for Binary Field Multiplication Using Quadrinomials," IEEE Trans. VLSI Systems, vol. 20, no. 3, pp. 449-458, Mar. 2012.

[10] H. Wu, "Bit-Parallel Polynomial Basis Multiplier for New Classes of Finite Fields," IEEE Trans. Computers, vol. 57, no. 8, pp. 1023- 1031, Aug. 2008.

[11] A. Hariri, and A. Reyhani-Masoleh, "Bit-Serial and Bit-Parallel Montgomery Multiplication and Squaring over GF(2m)," IEEE Trans. Computers, vol. 58, no. 10, pp. 1332-1345, Oct. 2009.

[12] I.S. Hsu, T. K. Truong, L. J. Deutsch, and I. S Reed, "A Comparison of VLSI Architecture of Finite Field Multipliers Using Dual, Normal, or Stnadard Basis," IEEE Trans. Computers, vol. 37, no. 6, pp. 735-739, Jun. 1988.

[13] D. Hankerson, A. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography. New York: Springer-Verlag, 2004.

[14] E. D. Mastrovito, "VLSI Designs for Multiplication over Finite Field GF(2m)," Proc. Sixth Symp. Applied Algebra, Algebraic Algorithms, and Error Correcting Codes (AAECC-6), pp. 297-309, Jul. 1988.

[15] E. D. Mastrovito, "VLSI Architectures for Computation in Galois Fields," PhD thesis, Link¨oping Univ., Link¨oping, Sweden 1991.

[16] T. Beth, and D. Gollmann, "Algorithm Engineering for Public Key Algorithms," IEEE J. Selected Areas in Communications, vol. 7, no. 4, pp. 458-466, May 1989.

[17] R. Azarderakhsh, and A. Reyhani-Masoleh, "Low-Complexity Multiplier Architectures for Single and Hybrid-Double Multiplications in Gaussian Normal Bases," IEEE Trans. Computers, vol. 62, no. 4, pp. 744-757, Jan. 2012.

[18] R. Azarderakhsh, K. J¨arvinen, and V. Dimitrov, "Fast Inversion in GF(2m) with Normal Basis Using Hybrid-Double Multipliers," IEEE Trans. Computers, in process.

[19] A. Reyhani-Masoleh, "A New Bit-Serial Architecture for Field Multiplication Using Polynomial Bases," In Proc. of CHES 2008, Aug. 2008, LNCS 5154, pp. 300-314.

[20] H. Wu, "Bit-Parallel Finite Field Multiplier and Squarer Using Polynomial Basis," IEEE Trans. Computers, vol. 51, no. 7, pp. 750- 758, Jul. 2002.

[21] F. Rodriguez-Henriguez, and C¸ . K. Koc¸, "Parallel Multipliers Based on Special Irreducible Pentanomials," IEEE Trans. Computers, vol. 52, no. 12, pp. 1535-1542, Dec. 2003.

[22] B. Sunar, and C¸ . K. Koc¸, "Mastrovito Multiplier for All Trinomials," IEEE Trans. Computers, vol. 48, no. 5, pp. 522-527, May 1999.

[23] A. Halbuo˘ gullari, and C¸ . K. Koc¸, "Mastrovito Multiplier for General Irreducible Polynomial," IEEE Trans. Computers, vol. 49, no. 5, pp. 503-518, May 2000.