Programmable LSB-first and MSB-first Modular Multipliers for ECC in $GF(2^m)$

Ravi Kumar Satzoda, Ramya Muralidharan and Chip Hong Chang
Center for High Performance Embedded Systems,
Nanyang Technological University, Singapore
{rksatzoda, ramy0003, echchang}@ntu.edu.sg

***Abstract*— In this paper, we propose programmable serial-in parallel-out LSB-first and MSB-first modular multipliers for elliptic curve cryptosystems (ECCs). The proposed multipliers can operate in any arbitrary field $GF(2^m)$ such that m is less than a maximum field order M. A linear array of processing elements is designed with a parallel switching circuitry to incorporate programmability in the fixed order multipliers. The proposed architectures are qualitatively compared against existing programmable multipliers in terms of gate count, delay and latency. The application specific integrated circuit (ASIC) implementation of the proposed multipliers using TSMC $0.18\mu m$ standard cell library is also analyzed.**

I. Introduction

LSB-first and MSB-first modular multipliers are commonly used for efficient implementation of elliptic curve cryptosystems (ECCs) in $GF(2^m)$ owing to their simple and regular systolic architectures [1]–[4]. Several variants of these multipliers are found in literature. While [1] discusses parallel multipliers, the area complexity of these multipliers is reduced by bit-serial implementations in [2]. The high latency of the bit-serial architectures in [2] is overcome by digitizing the modular multiplication (MM) algorithms in [3], [4].

In the current age of smartcards and internet banking, the security requirement is a dynamic parameter. This necessitates designing hardware architectures that are programmable to any field order. SEC2 [5] and NIST [6] define various secure field orders that define the keylength of ECCs. However, LSB-first and MSB-first architectures in [2]–[4] operate in a fixed field order, m, which dictates the operational field $GF(2^m)$ and the security strength of the ECC. In [1], in addition to bit parallel fixed order multipliers, programmble parallel pipelined multipliers are illustrated, that can operate in any field with field order m that is less than a maximum field order M. These programmable multipliers in [1] have an area complexity of $O(M^2)$ and a critical path delay of $(M-1)\Delta_{OR}$ where Δ_{OR} is the delay of a two-input OR gate. In the context of an ECC where $113 \leq m \leq 571$, these programmable multipliers will occupy large silicon area and incur high delay making them less feasible for actual hardware implementation.

In this paper, we present two new programmable serial-in parallel-out LSB-first and MSB-first modular multipliers that can operate for any field order $m \leq M$ and also reduce the area and delay costs significantly. The rest of the paper is organized as follows. The challenges in introducing programmability in the fixed order LSB-first and MSB-first multipliers are discussed in Section II. The proposed programmable multipliers are then derived and designed in Section III. In Section IV, the hardware cost metrics are discussed and conclusions are drawn in Section V.

II. Problem Statement

For the rest of the paper, the following notations are used. An m-bit vector is represented by boldfaced letter $\mathbf{X}$. The j-th bit of $\mathbf{X}$ in the i-th iteration is represented as $x_j^{(i)}$. The fixed order LSB-first MM of two polynomials, $a(x)$ and $b(x)$ in $GF(2^m)$ defined by a reduction polynomial $f(x)$, can be succinctly expressed as:

$$\begin{aligned} a_j^{(i)} &= a_{j-1}^{(i-1)} \oplus a_{m-1}^{(i-1)} f_j \qquad 0 \leq i \leq m-1,\ 0 \leq j \leq m-1 \\ t_j^{(i)} &= a_j^{(i-1)} \cdot b_{i-1} \oplus t_j^{(i-1)} \qquad 0 \leq i \leq m-1,\ 0 \leq j \leq m-1 \end{aligned} \tag{1}$$

where $\mathbf{A}^{(-1)} = a(x)$, $\mathbf{T}^{(-1)} = 0$. The MSB-first algorithm can also be similarly described by

$$t_j^{(i)} = t_{j-1}^{(i-1)} \oplus a_j \cdot b_{m-1-i} \oplus t_{m-1}^{(i-1)} \cdot f_j \qquad 0 \leq i, j \leq m-1 \tag{2}$$

where $\mathbf{T}^{(-1)} = 0$. In both (1) and (2), $\mathbf{T}^{(m-1)}$ is the final modular product. Both algorithms show a data dependency on the most significant bit (msb) of the intermediate vectors. In (1), the computation of the j-th bit of vector $\mathbf{A}$ in the i-th iteration, $a_j^{(i)}$, is dependent on the msb of $\mathbf{A}$ in the $(i-1)$-th iteration, $a_{m-1}^{(i-1)}$. Similarly, in (2), $t_j^{(i)}$ depends on $t_{m-1}^{(i-1)}$.

In fixed order multipliers the msb of $\mathbf{A}$ and $\mathbf{T}$ are known and so the data dependencies do not pose an issue. However in programmable multipliers, the challenge is to select the $(m-1)$-th bit of the M-bit intermediate vectors. This issue is addressed in the programmable parallel LSB-first and MSB-first multipliers in [1] by introducing a signal that runs through a column of M processing elements (PEs) but this leads to a long critical path delay and high area costs for large M.

III. Proposed Programmable Multipliers

In this section we propose programmable serial-in parallel-out LSB-first and MSB-first modular multiplier which can cater for any generic field $GF(2^m)$ such that $m \leq M$.

A. Proposed Method and Signal Flow Graphs

Since the proposed multipliers are designed for a maximum field order M, all the parallel input and output vectors are M bits long. In the proposed method, $a(x)$ and $f(x)$, which are m and $m+1$ bits long respectively, are padded with zeros in the msb positions to convert them into **A** and **F** vectors. The actual field order is then computed by generating an M-bit vector **S**, which has a one at $(m-1)$-th position, i.e. $s_{m-1}=1$, and rest of its bits are set to zero. This is done by determining the first occurence of '1' in **F** when **F** is read from its msb. For example, if $f(x)=x^5+x^3+1$, $m=5$ and for $M=10$, **F** = 0000101001 and **S** can be computed as 0000010000 with $s_4=1$. We explain the architecture that computes **S** in the later part of this paper.

Let us first consider LSB-first MM. The following expressions compute the modular product $t(x) = a(x)b(x)\ mod\ f(x)$ for any field order $m \le M$.

$$\begin{aligned} a_j^{(i)} &= a_{j-1}^{(i-1)} \oplus a_{sw}^{(i-1)} \cdot f_j \\ t_j^{(i)} &= a_j^{(i-1)} \cdot b_i \oplus t_j^{(i-1)} \end{aligned} \quad (3)$$

where $0 \le i \le m-1$, $0 \le j \le M-1$, $\mathbf{A}^{(-1)}$ =**A** and $\mathbf{T}^{(-1)}$=0. In the fixed-order LSB-first MM in (1) $a_{m-1}^{(i-1)}$ was known and directly used to compute $a_j^{(i)}$. However, in (3) the $(m-1)$-th bit of $\mathbf{A}^{(i-1)}$ is stored in $a_{sw}^{(i-1)}$, which is computed using **S** as shown below.

$$a_{sw}^{(i-1)} = a_j^{(i-1)} \quad \text{if } s_j = 1 \quad (4)$$

where $0 \le i \le m-1$, $0 \le j \le M-1$. In this equation $a_j^{(i-1)}$ is assigned to $a_{sw}^{(i-1)}$ only when $s_j=1$. Since only s_{m-1}-th bit is set to '1', for any arbitrary field order, $a_{m-1}^{(i-1)}$ is assigned to $a_{sw}^{(i-1)}$ in (4). This enables programmability in LSB-first MM algorithm.

Similary, the programmable MSB-first variant can be expressed as

$$t_j^{(i)} = t_{sw}^{(i-1)} \cdot f_j \oplus b_{m-1-i} a_j \oplus t_{j-1}^{(i-1)} \quad (5)$$

where $0 \le i \le m-1, 0 \le j \le M-1$. $t_{sw}^{(i-1)}$ is computed using **S** in a similar fashion as in (4) to determine the $(m-1)$-th bit of $\mathbf{T}^{(i-1)}$. In both (3) and (5), the final modular product is registered at the end of m iterations in $\mathbf{T}^{(m-1)}$.

The proposed algorithms for the programmable LSB-first and MSB-first MM in (3) to (5) enable a serial-in parallel-out data flow as shown by the signal flow graphs (SFGs) in Fig. 1. For the sake of clarity, the iteration indices in the superscripts are not indicated in Fig. 1. Let us consider the programmable LSB-first multiplier in Fig. 1(a) which comprises a linear array of processing elements (PE). Each circle represents a PE that executes (3) in each iteration. The input operand, $b(x)$, is read into the PEs serially along with the parallel inputs, **A** and **F**. In every iteration i, the array produces two intermediate vectors, $\mathbf{A}^{(i)}$ and $\mathbf{T}^{(i)}$. The selection of the $(m-1)$-th bit of $\mathbf{A}^{(i-1)}$ is performed by a bank of M parallel switches that are controlled by **S**. According to (4), in any iteration only the switch corresponding to s_{m-1} is activated to produce $a_{sw}^{(i-1)} = a_{m-1}^{(i-1)}$. The value of $a_{sw}^{(i-1)}$ is then simultaneously broadcast to the entire array. The intermediate vectors $\mathbf{A}^{(i)}$ and $\mathbf{T}^{(i)}$ are also registered and fed back to the PEs via registers (●) for the next iteration. The SFG for MSB-first MM in Fig. 1(b) can be explained similarly. The switch array in this case selects $t_{m-1}^{(i-1)}$. In addition, only one intermediate vector $\mathbf{T}^{(i)}$, is registered and fedback for each iteration.

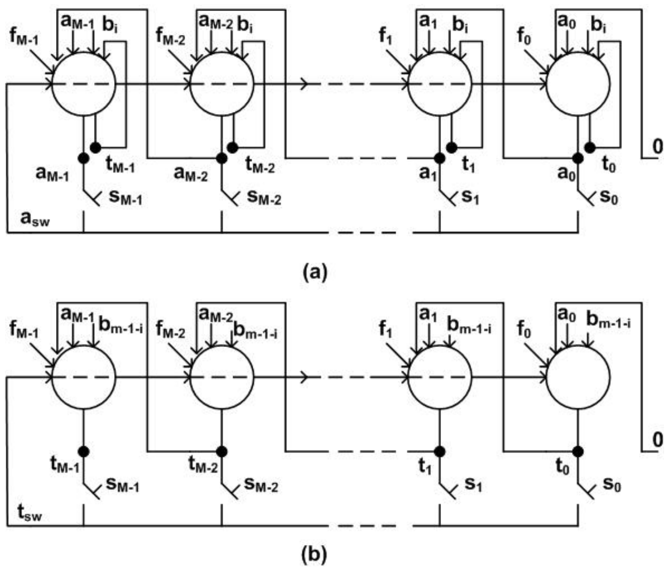

Fig. 1. Signal flow graphs (a) LSB-first modular multiplication (b) MSB-first modular multiplication for $GF(2^M)$

It was found that these SFGs cannot be directly translated into hardware architectures using application specific integrated circuit (ASIC) design flow. A 256-bit LSB-first multiplier was written in behavioral VHDL to implement the SFG. Synopsys Design Compiler v2004.12-SP2 was used to synthesize the architecture using TSMC 0.18 μm, 1.8 V Sage-X standard cell library. Design Compiler failed to synthesize the design even after running for 86 hours on a dual processor Solaris 8 Sun work station. It could however synthesize a smaller design with $M=128$. It was found that the assignment to $a_{sw}^{(i-1)}$ in (4) was implemented as an array of parallel tristate buffers (TBUFs). For the 256-bit multiplier, $a_{sw}^{(i-1)}$ signal was being driven by outputs from 256 TBUFs. Such high fan-in was not supported by the TSMC standard cell library. The direct translation of the SFGs being library dependent, the SFGs need to be divided in a systematic way.

B. Proposed Multiplier Architectures

We propose a multiplexer based bus architecture to obtain synthesizable designs. This is explained using the LSB-first multiplier but it is equally applicable to the MSB-first multiplier. The PE array shown in SFG is apportioned into p parallel processing blocks (PBs) of M/p PEs. In addition to $\mathbf{T}^{(i)}$ and $\mathbf{A}^{(i)}$, each PB generates a ternary output, $a_{swk} \in \{0,1,Z\}$ for $k=0,1,..,p-1$. Z represents high impedance state in a TBUF. Only one of the p sections, which has the $(m-1)$-th bit, will generate a valid binary (0 or 1) a_{swk} signal and the remaining sections result in Z. A control vector **P** multiplexes one of the four a_{swk} signals into $a_{sw}^{(i-1)}$ which corresponds to the PB containing the $(m-1)$-th bit of **A**. **P** is generated along with **S** as described in Section III-C.

Fig. 2(a) shows the architecture of the LSB-first MM for a maximum field order of 572. Here, the array is divided into four parts, i.e. $p = 4$, giving four PBs. Fig. 2(b) shows a single PE and there are 143 such PEs in each of the four PBs. Each PB generates 143-bit wide vectors $t^{(i)}_{571:429}$, ..., $t^{(i)}_{142:0}$ and $a^{(i)}_{571:429}$, ..., $a^{(i)}_{142:0}$. The intermediate vectors $\mathbf{A}^{(i)}$ and $\mathbf{T}^{(i)}$ are registered through flip flops (FFs) represented by •, to generate $\mathbf{A}^{(i-1)}$ and $\mathbf{T}^{(i-1)}$. These vectors are then fed back to the PB for the next iteration. Each PB is connected to a bit-select block (`BSB`) comprising 143 bit selectors (`BSel`) shown in Fig. 2(c). It consists of an OR gate followed by a TBUF. The OR gates in `BSels` have two inputs $\mathbf{A}^{(-1)}$ and $\mathbf{A}^{(i-1)}$. $\mathbf{A}^{(-1)}$ is initialized with input $\mathbf{A}$ for the first iteration ($i = 0$) and for subsequent iterations it is set to logic '0' thus propagating $\mathbf{A}^{(i-1)}$ through the OR gate to the PE array. The TBUF is used to select the $(m-1)$-th bit of the vector propagated through OR gates in each iteration and assigns it to the common a_{swk} signal of the BSB. The four BSBs generate four single-bit signals - a_{sw0}, ..., a_{sw3}. For instance, if $m = 471$, $a_{sw3} = a_{470}$ and a_{sw0}, a_{sw1} and a_{sw2} are pulled to 'Z'. a_{sw3} is multiplexed into $a^{(i-1)}_{sw}$ by $\mathbf{P}$ which equals '11'.

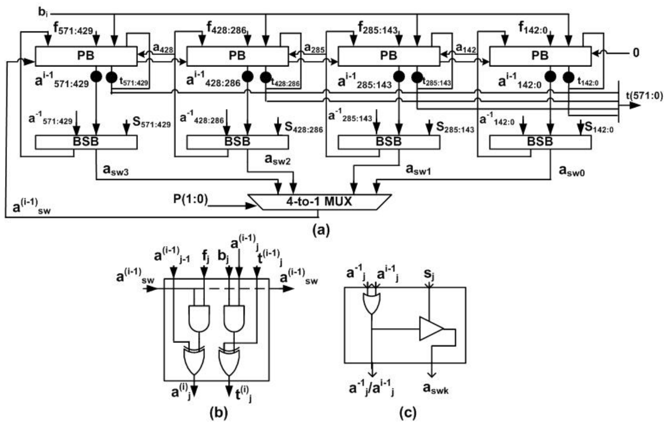

Fig. 2. Proposed programmable LSB-first modular multiplier for arbitrary $m \leq 572$ (a) Multiplier architecture (b) Processing element (PE) (c) Bit Selector (BSel)

Fig. 3 shows the architecture of the proposed MSB-first MM for $M = 572$. The architecture is analogous to the LSB-first multiplier discussed above. The BSB is simpler because only one intermediate vector $\mathbf{T}^{(i)}$ is selected in each iteration.

C. The S-Array Block

The generation of $\mathbf{S}$ and $\mathbf{P}$ vectors in the S-Array block is discussed in this section. $\mathbf{S}$ can be generated by cascading $M-1$ AND-OR gates in a linear array as in [1]. This however leads to a long critical path passing through $M-1$ OR gates. In the proposed multipliers such a logic severely affects the operating clock period because the critical path of the multiplier core is just one layer of PEs and is independent of M.

In order to equalize the delays of the S-array and multiplier blocks, we divide the array of $M-1$ AND-OR gates into p equal length blocks called the `S-subarray`. Each `S-subarray` is further divided into smaller units called the `S-cells` with no more than eight OR gates cascaded end

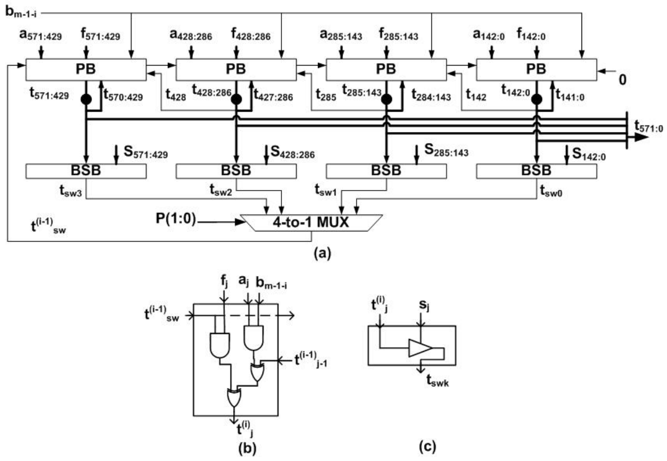

Fig. 3. Proposed programmable MSB-first modular multiplier for arbitrary $m \leq 572$ (a) Multiplier architecture (b) Processing element (PE) (d) Bit Selector (BSel)

to end. This is because the critical path of the multiplier block is approximately the delay of eight 2-input AND gates. All these functional blocks are connected as shown in Fig. 4. In addition to generating $\mathbf{S}$, each `S-subarray` generates a `P-propagate` signal (pp) and a `block select` signal (bs). When m is detected in an `S-subarray`, bs of that `S-subarray` is set to '1'. Its pp output is also set to 1 which resets the values of bs in the subsequent `S-subarrays` to its right. When all the bits of $\mathbf{F}$ have been read, one of the four bs signals holds a '1'. These bs signals are further used to generate the 2-bit vector $\mathbf{P}$ as shown in Fig. 4.

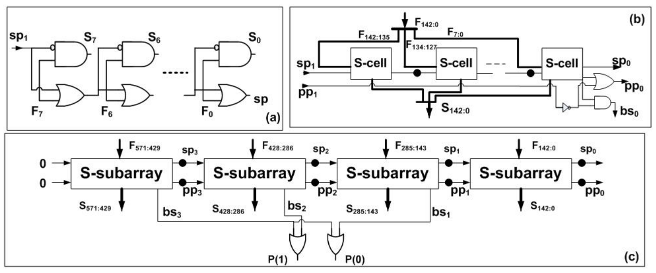

Fig. 4. $\mathbf{S}$ and $\mathbf{P}$ vectors generation circuit for $M = 572$ (a) Logic in S-cell (b) Logic in S-subarray (c) the S-array block

IV. Results and Discussion

The proposed LSB-first and MSB-first architectures are compared with the programmable architectures in [1] in the context of ECCs.

Gate count: Table I compares the basic cell (BC) complexity of the multiplier core and the S-array blocks of the proposed multiplier against equivalent computation cores in [1]. Each BC in the multiplier core of the proposed architectures comprises one PE and one `BSel`. In [1], a BC implements the logical functions of each iteration as described in [1].

TABLE I

AREA AND DELAY COMPLEXITY ANALYSIS

		Proposed		[1]	
		LSB-first	MSB-first	LSB-first	MSB-first
Multiplier core	Number of basic cells (BCs)	M BCs, 1 4-to-1 MUX		M^2 BCs	
	Basic cell (BC)	2 XOR, 2 AND, 1 OR, 1 TBUF, 2 FFs	2 XOR, 2 AND, 1 TBUF, 2 FFs	2 XOR, 4 AND, 1 OR, 2 FFs	2 XOR 4 AND, 1 OR, 1 FF
S-array block	Number of cells	$M/8$ S-cells, 2 OR		M cells	
	Cell	9 AND, 1 OR, 2 inverters, 2 FFs		1 AND, 1 OR, 1 inverter	
Total combinational gate count		$\left(\frac{53}{4}M + 13\right) A_2 + \left(\frac{21M}{4} + 4\right) A_1$	$\left(\frac{49}{4}M + 13\right) A_2 + \left(\frac{21M}{4} + 4\right) A_1$	$\left(11M^2 + 2M\right) A_2 + \left(2M^2 + 1M\right) A_1$	
Latency of multiplier L_m		$m + 1$ cycles		$M + 1$ cycles	
Critical path delay T_c		$9\Delta_{2inp} + 1\Delta_{TBUF}$		$(M - 1)\Delta_{OR}$	

$A_{XOR} = 3A_2 + 2A_1$; $A_{4to1MUX} = 11A_2 + 4A_1$; $A_{TBUF} = 3A_2 + 1A_1$
$\Delta_{4-to-1-MUX} = 5\Delta_{2inp}$; $\Delta_{XOR} = 2\Delta_{2inp}$

The total gate count for the combinational logic is expressed in terms of the number of equivalent inverters (A_1) and 2-input gates (A_2). As the proposed architectures comprise of a one-dimensional array as against a two-dimensional array in [1], an area savings of over 96% can be obtained by considering $M = 572$ for the gate count expressions in Table I.

Critical path delay: Critical path delay T_c is taken as the longest combinational delay between two flip-flops (FFs) and it defines the minimum operating clock period. From Table I it can be seen that T_c of the proposed multipliers is independent of the size of the multiplier, which is defined by M. It runs through one PE, one BSB and a 4-to-1 multiplexer. In contrast, T_c of the programmable multipliers in [1] equals $(M - 1)\Delta_{OR}$ because of the long chain of cascaded OR gates in each column of PEs, that determines the $(m - 1)$-th bit of intermediate vectors. These values show a clear ascendancy of the proposed architectures over [1].

Latency: Latency is defined as the number of clock cycles required to compute a valid output for a given input. In the proposed multipliers, the total latency, L_{total}, can be spilt into two parts - L_s and L_m. L_s is a one time setup latency to determine the actual field order m in the S-array block whereas L_m is the latency of the multiplier core. During a typical encryption or decryption process in ECC, MM is repeatedly employed to compute a point multiplication [7]. So for the entire encryption or decryption process, the S-array operates only once. Therefore, L_m is the more dominant component in the total latency. From Table I, we see that the proposed architectures compute the modular product in lesser number of clock cycles as compared to [1]. In [1], though the architectures are pipelined the latency is still dependent on M due to the inherent limitations of the algorithms.

ASIC Implementation: The proposed programmable MSB-first multiplier and that of [1] were implemented for the largest field order, $M = 572$ [6] on an ASIC platform. The designs were synthesized using Synopsys Design Compiler v2004.12-SP2 on a Sun Solaris 8 dual-processor system with a RAM of 4 GB and TSMC 0.18 μm standard cell library with 1.8 V supply. Design Compiler could not synthesize the parallel architecture of [1] due to the complexity of the design requiring a RAM of over 4.4 GB. This shows the impracticality of implementing the fully parallel architectures of [1] for large M as required by ECCs. Furthermore, the proposed multiplier could be synthesized using the same platform.

The two programmable MSB-first multipliers were implemented for a smaller $M = 128$ for comparison. Our architecture resulted in a minimum clock period of $1.98ns$ giving a speed up of 16.7 times over the multiplier in [1]. A typical point multiplication in ECC requires approximately $(6\lfloor log_2 m\rfloor + 10)$ MMs [7]. For an ECC operating in $GF(2^{113})$, all MMs in a point multiplication were completed in $11.49\mu s$ using the proposed multiplier which is about 10 times faster than its counterpart in [1].

V. CONCLUSION

In this paper, we presented programmable LSB-first and MSB-first modular multipliers that can operate in arbitrary field order m where $m \leq M$, M being the highest possible field order. Signal flow graphs were derived for the proposed methods and were translated to synthesizable architectures. Our serial-in parallel-out multipliers are suitable for ASIC implementation owing to their reduced complexity and can cater for ECCs of orders as high as 572. For a 128-bit multiplier, a 16.7 times speedup in operating frequency is achieved with an area savings of 96% over existing programmable multiplier.

REFERENCES

[1] S. K. Jain, L. Song and K. K. Parhi, "Efficient semisystolic architectures for finite-field arithmetic," *IEEE Trans. on Very Large Scale Intergration (VLSI) Systems*, vol. 6, no. 1, pp. 101-113, Mar. 1998.

[2] C.-L. Wang and J.-L. Lin, "Systolic array implementation of multipliers for finite fields $GF(2^m)$," *IEEE Trans. on Cir. and Sys.-I*, vol. 38, no. 7, pp. 796-800, July 1991.

[3] J.-H. Guo and C.-L. Wang, "Digit-serial systolic multiplier for finite fields $GF(2^m)$," *IEE Proc. Comput. Digit. Tech.*, vol. 145, no. 2, pp. 143-148, Mar. 1998.

[4] C. H. Kim, C. P. Hong and S. Kwon, "A digit-serial multiplier for finite field $GF(2^m)$," in *IEEE Trans. on VLSI*, vol. 13, no. 4, pp. 476-483, Apr. 2005.

[5] Certicom Research, "SEC 2: Recommended elliptic curve domain parameters," *Standards for Efficient Cryptography*, Technical document, Sept. 20, 2000.

[6] National Institute of Standards and Technology (NIST), United States of America, http://csrc.nist.gov/csrc/fedstandards.html

[7] J. López and R. Dahab, "Fast Multiplication on Elliptic Curves over $GF(2^m)$ without Precomputation," *Lecture Notes In Computer Science, Proc. of the First International Workshop on Cryptographic Hardware and Embedded Systems (CHES 1999)*, vol. 1717, pp. 316-327, 1999.