symmetry

MDPI

Article

Novel Hybrid-Size Digit-Serial Systolic Multiplier over $GF(2^m)$

Zhenji Hu [1] and Jiafeng Xie [2,*]

[1] School of Law, ShangHai University of Finance & Economics, Shanghai 200433, China; jenneyhu1986@163.com
[2] Department of Electrical & Computer Engineering, Villanova University, Villanova, PA 19085, USA
* Correspondence: jiafeng.xie@villanova.edu

Received: 26 September 2018; Accepted: 20 October 2018; Published: 24 October 2018

Abstract: Because of the efficient tradeoff in area–time complexities, digit-serial systolic multiplier over $GF(2^m)$ has gained substantial attention in the research community for possible application in current/emerging cryptosystems. In general, this type of multiplier is designed to be applicable to one certain field-size, which in fact determines the actual security level of the cryptosystem and thus limits the flexibility of the operation of cryptographic applications. Based on this consideration, in this paper, we propose a novel hybrid-size digit-serial systolic multiplier which not only offers flexibility to operate in either pentanomial- or trinomial-based multiplications, but also has low-complexity implementation performance. Overall, we have made two interdependent efforts to carry out the proposed work. First, a novel algorithm is derived to formulate the mathematical idea of the hybrid-size realization. Then, a novel digit-serial structure is obtained after efficient mapping from the proposed algorithm. Finally, the complexity analysis and comparison are given to demonstrate the efficiency of the proposed multiplier, e.g., the proposed one has less area-delay product (ADP) than the best existing trinomial-based design. The proposed multiplier can be used as a standard intellectual property (IP) core in many cryptographic applications for flexible operation.

Keywords: digit-serial; hybrid-size; low-complexity; pentanomial; systolic structure; trinomial

1. Introduction

Finite field multipliers have gained substantial attentions recently due to their critical roles in many cryptosystems such as elliptic curve cryptography (ECC), especially on hardware platforms [1]. Typically, there are three types of structuring related to the finite field multipliers, namely the bit-serial, bit-parallel, and digit-serial. Because of the efficient tradeoff in area–time complexities, digit-serial structures usually are more widely preferred than the other two in many applications [2].

Along with the recent advance in artificial intelligence technology, systolic structure has becoming more and more attracting in high-performance hardware platforms [3]. Accordingly, digit-serial systolization of finite field multipliers have the potential to be applied in high-performance cryptosystems due to their superior features such as high-throughput rate and regularity and modularity. Thus far, several efforts have been made on efficient implementation of digit-serial systolic finite field multipliers: (i) an efficient systolic finite field multiplier is presented in [3], where its complexity is significantly reduced compared with the previous reported one; (ii) a systolic-like digit-serial multiplier is reported in [4] and it is found that the systolic structure proposed is specifically suitable for Reed–Solomon Codec; (iii) an efficient digit-serial systolic multiplier is presented in [5]; (iv) the same authors reported a unified digit-serial systolic multiplier based on trinomials and all-one-polynomials [6]; (v) a low-complexity systolic multiplier is given in [7], where its complexity is optimized to be minimal; (vi) an efficient resource-sharing technique is employed in another digit-serial

systolic multiplier to achieve low critical-path and high-performance operation [8]; and (vii) an efficient systolic digit-serial multipliers is reported in [9], where the complexity is so far the least in the literature. These designs, undoubtedly, represent the major advance in the field of systolic digit-serial multipliers.

On the other side, however, the existing digit-serial systolic finite field multipliers, more or less, still have some drawbacks to be overcome: (i) although the digit-serial systolic multipliers have relatively few processing elements (PEs), the register-complexity of the multipliers is still large; and (ii) the current digit-serial multipliers are designed to be fixed field-size, and thus cannot provide enough flexibility to meet the current technology trend, i.e., one cryptosystem can meet different security level (field-size) need and the designers have to finalize different field-size multipliers with respect to different application requirement, which is some sort of inefficient in integrated chip (IC) design. Facing with these two challenges, in this paper, we have proposed a novel hybrid-size digit-serial systolic multiplier with low-complexity implementation. The proposed work is carried out through a combination of two coherent interdependent stages' efforts: (i) a novel hybrid-size digit-serial systolic multiplication algorithm is proposed which provides enough flexibility to both pentanomial- and trinomial-based multipliers; and (ii) the proposed algorithm is then mapped into a novel systolic structure through a series of optimization techniques. Thorough complexity analysis and detailed comparison have also been made to confirm the efficiency of the proposed design, i.e., it not only offers flexibility to be switched from one field-size to another one, but also has smaller area–time complexities compared with the existing single field-size digit-serial systolic multipliers. The proposed design can not only be used as a standard intellectual property (IP) core for various field-size cryptosystem, but also can be employed as a core computation unit in reconfigurable cryptographic processor (where demands flexible field-size choice).

The rest of the paper is organized as follows: Section 2 presents the mathematical formulation of the proposed digit-serial multiplication algorithm. Section 3 shows the detailed steps of the proposed systolic structure mapped from the algorithm. The analysis and comparison are provided in Section 4. The conclusion is given in Section 5.

2. Mathematical Formulation of the Proposed Multiplication Algorithm

Let the three elements A, B, and $C \in GF(2^m)$ and let the polynomial basis be the $\{1, x, x^2, \ldots\}$, where x is the root of $f(x)$ ($f(x)$ determines the field) [1]. Suppose for two field-sizes with m_1 and m_2, and $m_1 < m_2$, we can first define that

$$A_1 = \sum_{i=0}^{m_1-1} a_i x^i, B_1 = \sum_{i=0}^{m_1-1} b_i x^i, C_1 = \sum_{i=0}^{m_1-1} c_i x^i, \tag{1}$$

and

$$A_2 = \sum_{i=0}^{m_2-1} a_i x^i, B_2 = \sum_{i=0}^{m_2-1} b_i x^i, C_2 = \sum_{i=0}^{m_2-1} c_i x^i, \tag{2}$$

where a_i, b_i, and $c_i \in GF(2)$ and it is clear that a_i in both A_1 and A_2 are the same (for $0 \leq i \leq m_2 - 1$), and the same applies to b_i and c_i.

Suppose in the field-size of m_1, let C_1 be the product of A_1 and B_1 (corresponding field polynomial is $f_1(x)$), we can have

$$\begin{aligned} C_1 &= A_1 B_1 \bmod f_1(x) \\ &= \sum_{i=0}^{m_1-1} b_i (x^i \cdot A_1) \bmod f_1(x) \\ &= \sum_{i=0}^{m_1-1} b_i A_1^{(i)}, \end{aligned} \tag{3}$$

where $A_1^{(0)} = A_1$ and $A_1^{(i)} = x^i \cdot A_1 \bmod f_1(x)$. Similarly, for the field-size of m_2, we can have C_2 as the product of A_2 and B_2 (the field polynomial is $f_2(x)$):

$$\begin{aligned} C_2 =& A_2 B_2 \bmod f_2(x) \\ =& \sum_{i=0}^{m_2-1} b_i (x^i \cdot A_2) \bmod f_2(x) \\ =& \sum_{i=0}^{m_2-1} b_i A_2^{(i)}, \end{aligned} \tag{4}$$

where, similarly, we have $A_2^{(0)} = A_2$ and $A_2^{(i)} = x^i \cdot A_2 \bmod f_2(x)$. One can then have

$$C_2 = \sum_{i=0}^{m_1-1} b^i A_2^{(i)} + \sum_{i=m_1}^{m_2-1} b^i A_2^{(i)} \tag{5}$$

Then, after comparing Equation (3) with Equation (5), we can have

$$C_j = \sum_{i=0}^{m_1-1} b^i A_j^{(i)} + k_j \cdot \sum_{i=m_1}^{m_2-1} b^i A_2^{(i)}, \tag{6}$$

where $j = 1$ or 2 according to Equations (3) and (5), respectively, and

$$\begin{aligned} k_j &= 0, \text{ if } j = 1 \\ k_j &= 1, \text{ if } j = 2. \end{aligned} \tag{7}$$

Then, we can have the following definitions:

For any integer of m_2, we have $m_2 = w \cdot d$ (meanwhile, one can have $m_1 = w \cdot d_1$); then, we can define

$$\begin{aligned} B_1 &= [b_0, b_1, \ldots, b_{w-1}] \\ B_2 &= [b_w, b_{w+1}, \ldots, b_{2w-1}] \\ &\cdots \quad \cdots \quad \cdots \\ B_d &= [b_{m_2-w}, b_{m_2-w+1}, \ldots, b_{m_2-1}]. \end{aligned} \tag{8}$$

Similarly, we can have

$$\begin{aligned} A_1 &= [A_j^{(0)}, A_j^{(1)}, \ldots, A_j^{(w-1)}] \\ A_2 &= [A_j^{(w)}, A_j^{(w+1)}, \ldots, A_j^{(2w-1)}] \\ &\cdots \quad \cdots \quad \cdots \\ A_d &= [A_2^{(m_2-w)}, A_2^{(m_2-w+1)}, \ldots, A_2^{(m_2-1)}], \end{aligned} \tag{9}$$

where we assume $m_2 - m_1 > w$.

It is clear that we can now transfer Equation (6) into

$$\begin{aligned} C_j &= A_1 B_1^T + A_2 B_2^T + \cdots + k_j A_d B_d^T \\ &= \sum_{u=1}^{d} \xi(k_j) A_u B_u^T, \end{aligned} \tag{10}$$

where $\xi(k_j)$ works for terms only when $m_1 \leq i \leq m_2 - 1$ and

$$\begin{aligned} \xi(k_j) &= 1, \text{ if } j = 2 \\ \xi(k_j) &= 0, \text{ if } j = 1, \end{aligned} \tag{11}$$

where we can see that Equation (10) can be used to perform two field-size finite field multiplications if we select the control signal properly.

The above equations can thus be summarized as Algorithm 1.

Algorithm 1. Proposed multiplication algorithm for hybrid field-size-based implementation

Inputs: A_1 and B_1 (also A_2 and B_2) are the pair of elements (polynomial basis representation) in $GF(2^m)$ for field-size of m_1 and m_2, respectively

Output: $C_j = A_j \cdot B_j \bmod f_j(x)$ for $j = 1$ or 2 and $f_j(x)$ is the field polynomial

1. Initialization step
   - 1.1 Define $B_1, B_2, \ldots, B_d$ for $m_2 = w \cdot d$ according to (8)
   - 1.2 Define $A_1, A_2, \ldots, A_d$ for $m_2 = w \cdot d$ according to (9)
   - 1.3 Define $D = 0$
2. Multiplication step
   - 2.1 According to the field-size selection signal, determine the value of j
   - 2.2 For $1 \leq u \leq d$
   - 2.3 $D = D + \xi(k_j) A_u B_u^T$
   - 2.4 End for
3. Final step
   - 3.1 Get $C_j = D$

The detailed processes of Steps 2.2 and 2.4 are the key multiplication processes.

Note that, due to the difference of the field polynomial $f_j(x)$, the process of deriving $A_j^{(i+1)}$ from $A_j^{(i)}$ is slightly different from each other. For instance, assume

$$A_j^{(i)} = \sum_{v=0}^{m_j-1} a_{j\to v}^{(i)} x^v, \tag{12}$$

where we can have

$$\begin{aligned} a_{j\to 0}^{(i+1)} &= a_{j\to m_j-1}^{(i)} \\ a_{j\to s}^{(i+1)} &= a_{j\to s-1}^{(i)} + a_{j\to m-1}^{(i)} \\ a_{j\to v}^{(i+1)} &= a_{j\to v-1}^{(i)}, \text{ for } 1 \leq v \leq m_j - 1, \end{aligned} \tag{13}$$

when $f_j(x)$ is a trinomial of $f_j(x) = x^{m_j} + x^s + 1$. While, for pentanomial $f_j(x) = x^{m_j} + x^{s_1} + x^{s_2} + x^{s_3} + 1$, we can have

$$\begin{aligned} &a_{j\to 0}^{(i+1)} = a_{j\to m_j-1}^{(i)}, \quad a_{j\to s_1}^{(i+1)} = a_{j\to s_1-1}^{(i)} + a_{j\to m_j-1}^{(i)} \\ &a_{j\to s_2}^{(i+1)} = a_{j\to s_2-1}^{(i)} + a_{j\to m_j-1}^{(i)}, \quad a_{j\to s_3}^{(i+1)} = a_{j\to s_3-1}^{(i)} + a_{j\to m_j-1}^{(i)} \\ &a_{j\to v}^{(i+1)} = a_{j\to v-1}^{(i)}, \text{ for } 1 \leq v \leq m_j - 1 \text{ and } v \neq s_1, s_2, s_3. \end{aligned} \tag{14}$$

Besides that, one has to note that the National Institute of Standards and Technology (NIST) has recommended five irreducible polynomials for ECC implementation [10,11] (three pentanomials and

two trinomials). Without loss of generality, we can assume m_1 is a pentanomial and m_2 is a trinomial. The corresponding structure presented below is also based on this assumption.

3. Proposed Hybrid-Size Digit-Serial Systolic Multiplier

In this section, we propose several optimization technique to successfully map the corresponding algorithm into desired systolic structure. Specifically:

3.1. Novel Input Data Broadcasting Scheme

One major component of the register-complexity of a systolic finite field multiplier comes from the input data broadcasting. In this subsection, we propose a novel input data broadcasting that the main inputs to each PE are fed independent from each other and thus the relation of these data between the PEs is reduced to minimum, which can significantly reduce the related register-complexity among systolic array. In Figure 1, the proposed input data broadcasting technique is employed.

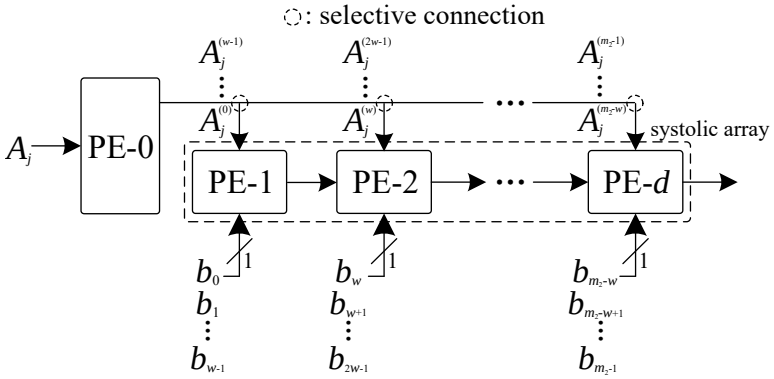

Figure 1. The proposed input data broadcasting technique.

As shown in Figure 1, according to Step 2.3 of Algorithm 1, each PE in the systolic array is fed with two inputs, namely the $A_j^{(i)}$ and the corresponding b_i. The output of each PE is then transferred to the next PE on its right. The complete output can be delivered after $(d + w)$ cycles, with the help of an extra accumulation cell. Since differences exist among all the $A_j^{(i)}$, we have used the selective connection to rightly connect each PE according to Algorithm 1. Because only one signal pipelining to the next PE is used, the register-complexity of the systolic array is significantly reduced. The details of the internal structures of these PEs are shown below. Note that, due to the simple internal structure of the PEs, i.e., critical-path of the PE is quite small, the proposed broadcasting technique has very limited influence on the overall time complexity.

3.2. Proper Arrangement on the Input Data Delivery

The two inputs, i.e., A_j and B_j, must be properly arranged to meet the data dependence requirement for hybrid-size operation. For B_j, according to Algorithm 1, all bits are delivered in a grouped-sequential way, which can be realized by the structure, as shown in Figure 2. One can see that the shift-register is producing the required output bits to each PE of Figure 1 based on Algorithm 1, while the hybrid-size selection is done by the inserting of an extra MUX (MUX is short for multiplexer) in the shifting path such that the shift-register can be working under the field-size of either m_1 or m_2 through the proper control of the MUX (control signal).

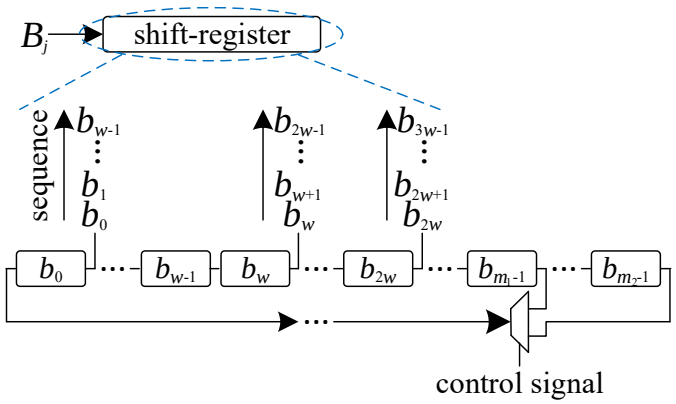

Figure 2. The proposed shift-register to deliver input data B_j.

The operand A_j, through the help of PE-0, delivers the correct output bits to each PE according to Algorithm 1, which requires a more sophisticated structure, as shown in Figure 3a. From Equations (13) and (14), one can observe that there are one XOR gate involved when obtaining $A_2^{(i+1)}$ from $A_2^{(i)}$ (trinomial-based multiplication) while it requires three XOR gates when deriving $A_1^{(i+1)}$ from $A_1^{(i)}$. Thus, according to the data dependence requirement of Algorithm 1, one can find that there are d number of operands being delivered from PE-0 at the same cycle period, i.e., $A_j^{(wu)}, A_j^{(wu+1)}, \ldots, A_j^{(wu+d-1)}$. These operands, in fact, involve multiple identical bits between each other and thus can be shared. For simplicity of discussion, let assume $f_2(x) = x^{233} + x^{74} + 1$ and $f_1(x) = x^{163} + x^7 + x^6 + x^3 + 1$, we can have ($d_1 = 13$, $d = 16$, $a_0, a_1, a_2, \ldots, a_{162}$ for A_1, and $a_0, a_1, a_2, \ldots, a_{232}$ for A_2)

$$
\begin{aligned}
&[A_1^{(0)}\ A_1^{(1)}\ A_1^{(2)}\ \cdots\ A_1^{(12)}] \\
&= \begin{bmatrix}
a_0 & a_{162} & a_{161} & \cdots & a_{151} \\
a_1 & a_0 & a_{162} & \cdots & a_{152} \\
a_2 & a_1 & a_0 & \cdots & a_{153} \\
a_3 & a_2 + a_{162} & a_1 + a_{161} & \cdots & a_{154} + a_{150} \\
a_4 & a_3 & a_2 & \cdots & a_{155} \\
a_5 & a_4 & a_3 & \cdots & a_{156} \\
a_6 & a_5 + a_{162} & a_4 + a_{161} & \cdots & a_{157} + a_{150} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
a_{162} & a_{161} & a_{160} & \cdots & a_{150}
\end{bmatrix},
\end{aligned}
\tag{15}
$$

and

$$
\begin{aligned}
&[A_2^{(0)}\ A_2^{(1)}\ A_2^{(2)}\ \cdots\ A_2^{(15)}] \\
&= \begin{bmatrix}
a_0 & a_{232} & a_{231} & \cdots & a_{218} \\
a_1 & a_0 & a_{232} & \cdots & a_{219} \\
a_2 & a_1 & a_0 & \cdots & a_{220} \\
a_3 & a_2 & a_1 & \cdots & a_{221} \\
a_4 & a_3 & a_2 & \cdots & a_{222} \\
a_5 & a_4 & a_3 & \cdots & a_{223} \\
a_6 & a_5 & a_4 & \cdots & a_{224} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
a_{232} & a_{231} & a_{230} & \cdots & a_{217}
\end{bmatrix},
\end{aligned}
\tag{16}
$$

where we can see that the identical bits, e.g., $a_0, a_1, \ldots$, can be shared among these $A_j^{(i)}$ $(0 \leq i \leq 12)$, as shown by the example in Figure 3b (where we have shown how the MUXes are located to obtain hybrid-size implementation). Since other bits cannot be shared, we just use the MUX to connect with the two bits at the same position (according to Equations (8) and (9)) such that, through the proper working of these MUXes, the correct signals can be produced to the corresponding PE.

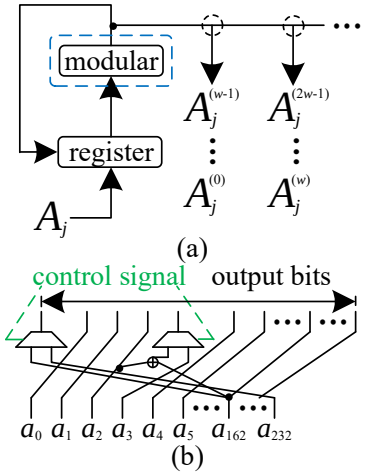

Figure 3. The internal structure of PE-0 to deliver input data A_j: (**a**) internal structure; and (**b**) an example of how the MUXes are inserted to obtain hybrid-size implementation for the modular cell in PE-0.

One can also notice that, according to Equations (9), (13), and (14), with the help of a modular operation (done by the modular cell in Figure 3), the PE-0 delivers the corresponding output to each PE, i.e., obtaining A_u from A_{u-1} (for $1 \leq u \leq d/d_1$), which needs a delay time of $2T_X$ (T_X is the delay time of an XOR gate, and it takes T_X for trinomial-based multiplier and $2T_X$ for pentanomial-based one [9]). Besides that, one has to note that all the $A_j^{(i)}$ in one specific A_u can be obtained through the sharing of identical bits, as represented by the selective connection in Figures 1 and 3. Following this arrangement, the proposed hybrid-size structure operates in an ordered form according to Algorithm 1.

3.3. Hybrid Accumulation

The accumulation of the digit-serial operation also needs adjustment when compared with the conventional ones. As shown in Figure 4, where we have used a m_1-bit MUX cell to obtain the hybrid-size accumulation (where the accumulation cell is realized through the XOR cell connected with the register cell in a back-loop style). Note that these m_1 bit-level MUXes connect with the m_1-bit output of PE-d_1, while the remaining $(m_2 - m_1)$ bits of PE-d are directly connected with the accumulation cell. According to Equations (8) and (9), and Algorithm 1, we can let the MUX determine the multiplier is working under the condition of field-size of either m_1 bits or m_2 bits. Besides that, the number of output bits is also selected according to the specific chosen field-size, as shown in Figure 4, i.e., after designated number of cycle periods, the output is produced based on the value of the control signal.

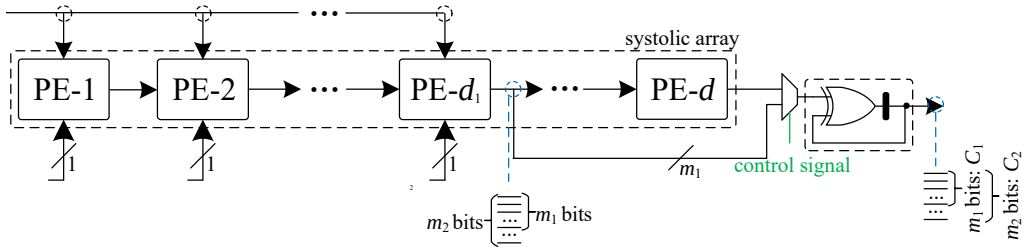

Figure 4. The detailed arrangement of how the accumulation cell operate to realize the hybrid-size implementation (only m_1 bit-level MUXes are employed and the remaining $(m_2 - m_1)$ bits of the output of PE-d are directly connected with the accumulation cell), where the black box in the accumulation cell denotes the register cell.

3.4. Final Structure

The internal structure of each PE is shown in Figure 5b, where it mainly consists of an AND cell, an XOR cell, and a register cell. With the combination of all the optimization techniques introduced above, we have presented the finalized proposed hybrid-size digit-serial structure, as shown in Figure 5a. All the control signals connected with the inserted MUXes collaborate together to switch the finite field multiplier from operating in one field-size to another. After designated cycle periods of accumulation, the multiplier delivers the desired output.

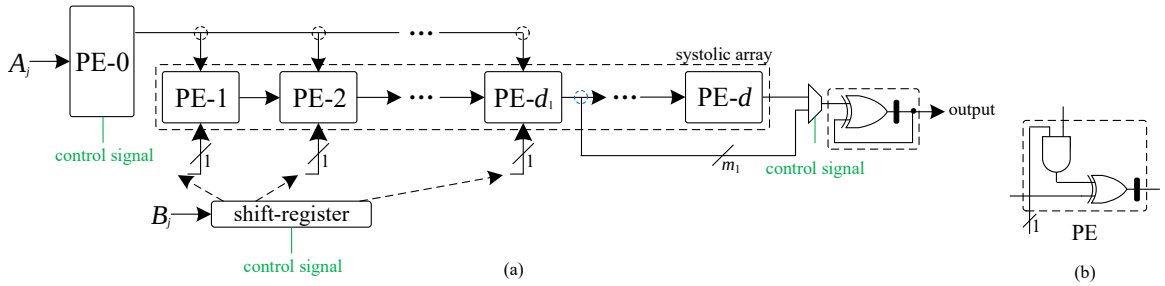

Figure 5. The final structure: (**a**) the proposed structure; and (**b**) the internal structure of a regular PE.

4. Complexity and Comparison

For simplicity of discussion, we just follow the assumption in Section 3 that m_1 comes from a pentanomial while m_2 is the field-size of a trinomial. The detailed complexity of the proposed multiplier is: (i) Systolic array: The systolic array has d number of PEs, where each PE has m_2 AND gates, m_2 XOR gates, and m_2 registers. (ii) Shift-register: The shift-register for B_j requires m_2 registers and one MUX. (iii) Accumulation cell: The accumulation cell requires m_1 MUXes, m_2 XORs, and m_2 registers. (iv) PE-0: There are in total $(3d_1 + d - 4)$ XOR gates, $(4d_1 - 4)$ MUXes, and $(m_2 + 3d_1 + d - 4)$ registers involved. Moreover, the proposed structure has a critical-path of $(2T_X + T_M)$ (T_M is the delay time of an MUX), and it takes $(d + w)$ cycles to produce the desired output for hybrid-size operation.

Overall, the complexity of the proposed design is listed along with the existing digit-serial multipliers (trinomial- or pentanomial-based designs) in Table 1 in terms of logic gates number, register number, latency (number of cycle periods), and critical-path. Note that the designs of [5,6] are based on all-one-polynomials (or used all-one-polynomials as a computation core), we thus do not list them in Table 1, just for a fair comparison. As shown in Table 1, one can see that the proposed hybrid-size digit-serial multiplier has relatively better area–time complexities than the existing ones, especially when considering that the proposed one can offer hybrid field-size operation (the existing ones are all single field-size based). To have a detailed comparison, we have also used the NanGate's Library Creator and the 45-nm FreePDK Base Kit from North Carolina State University (NCSU) [12] to estimate the area and time complexities of all the designs for $m_2 = 233, m_1 = 163$, $d = 16$, and $d_1 = 13$.

The obtained area, delay (latency time), power, area-delay product (ADP), and power-delay product (PDP) are listed in Table 2 for a comparison. Again, we can observe that the proposed one has better performance than the existing ones, e.g., it has at least 7.3% less ADP than the best trinomial one of [8], while it offers the flexibility to execute the pentanomial-based multiplier. Compared with the existing pentanomial ones, the proposed one still has better ADP when considering the scaling of the field-size. The proposed one also has 41.5% less ADP and 34.6% less PDP than the conventional hybrid field-size implementation (we have combined the best existing ones of [8,9] together to realize it).

Table 1. Comparison of Area–Time Complexities of Various Digit-Serial Systolic Multipliers.

Design	AND	XOR	Register	Latency	Critical-Path
Digit-serial systolic structures (trinomial of size m_2)					
[7] [1]	$m_2 d$	$d(2+m_2)+1$	$2m_2 d+3m_2$	$d+w$	T_A+T_X
[8] [1]	$m_2 d$	$m_2 d+m_2-d+1$	$2m_2 d+2m_2$	$d+w$	T_A+T_X
Digit-serial systolic structures (pentanomial of size m_1)					
[4] [2]	$2m_1 d_1+m_1$	$2m_1 d_1$	$m_1/d_1(10d_1+$ $1+9sd_1/2+s)$	$3m_1/d_1$	$d_1(T_A+T_X+$ $T_{MUX})/(s+1)$
DS-I [9]	$m_1 d_1$	$m_1 d_1+3m_1$ $-3m_1/d_1+3$	$2d_1 m_1+2m_1$	d_1+w_1	$2T_X$
Hybrid-size digit-serial systolic structures (pentanomial of size m_1 and trinomial of size m_2)					
Proposed [3]	$m_2 d$	$m_2 d+m_2$ $3d_1+d-4$	$m_2 d+2m_2$ $+3d_1+d-4$	$d+w$	$2T_X+T_M$

T_A, delay time of an AND gate; T_M, delay time of an MUX; [1]: For a fair comparison, the input bits of operand B_j fed to each PE is finalized as 1 for [7,8] (a shift-register of m_2 bits is also added). [2]: $(s+1)$ refers to the pipelined stage in the PE (here we choose s = 1), and $2m_1$ of MUX gates are not listed here. [3]: There are also (m_1+4d_1-3) MUXes involved.

Table 2. Comparison of Area–Time Complexities of Various Digit-Serial Systolic Multipliers.

Design	Area (μm^2)	Delay (ns) [1]	Power (μW/GHz)	ADP ($\mu m^2 \times$ ns)	PDP (μW/GHz$\times$ ns)
Digit-serial trinomial-based ($m_2 = 233, d = 16$)					
[7]	46,958	4.48	56,068	210,372	251,185
[8]	46,174	4.48	55,726	206,860	249,652
Digit-serial pentanomial-based ($m_1 = 163, d_1 = 13$)					
[4]	22,675	45.63	35,412	1,034,660	1,615,850
[9]	26,998	4.16	33,139	112,312	137,858
Hybrid-size ($m_1 = 163$ and $m_2 = 233$)					
Traditional [2]	73,172	4.48	88,865	327,811	398,115
Proposed	29,961	6.4	40,672	191,750	260,301

[1]: delay = latency cycle number $\times$ critical-path. [2]: Refers to the conventional implementation of two field-size finite field multipliers; we have used the best existing ones of [8,9] to be combined together.

The proposed hybrid-size digit-serial systolic multiplier, undoubtedly, can be extended as a standard IP core in various cryptosystems that demand different security levels. On the other hand,

due to the low-complexity of the proposed design, it can also be used in cryptosystem for flexible operation, in the case the user of that cryptosystem needs to change/upgrade the system. Moreover, it is worth mentioning that the proposed hybrid field-size strategy can also be extended to multiple filed-size implementation.

5. Conclusions

This paper presents a novel implementation of a hybrid field-size digit-serial systolic multiplier over $GF(2^m)$. A novel digit-serial multiplication algorithm suitable for hybrid field-size realization is proposed first. Then, through a series of optimization techniques, the proposed algorithm is successfully mapped into a high-performance digit-serial systolic multiplier. The complexity analysis and detailed comparison have been given to confirm the efficiency of the proposed design. Future work may focus on the application of the proposed design in various cryptosystems.

Author Contributions: Resources, Z.H.; Writing—Review & Editing, J.X.

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IP	Intellectual property
ECC	Elliptic curve cryptography
PE	Processing elements
IC	Integrated chip
NCSU	North Carolina State University

References

1. Blake, I.; Seroussi, G.; Smart, N.P. *Elliptic Curves in Cryptography*; London Mathematical Society Lecture Note Series; Cambridge University Press: Cambridge, UK, 1999.
2. Xie, J.; Meher, P.K.; He, J. Low-latency area-delay-efficient systolic multiplier over $GF(2^m)$ for a wider class of trinomials using parallel register sharing. In Proceedings of the 2012 IEEE International Symposium on Circuits and Systems, Seoul, Korea, 20–23 May 2012; pp. 89–92.
3. Systolic Array. Available online: https://en.wikipedia.org/wiki/Systolic-array (accessed on 25 September 2018).
4. Kim, C.H.; Hong, C.P.; Kwon, S. A digit-serial multiplier for finite field $GF(2^m)$. *IEEE Trans. Very Large Scale Integr. Syst.* **2005**, *13*, 476–483.
5. Meher, P.K. Systolic and non-systolic scalable modular designs of finite field multipliers for Reed-Solomon Codec. *IEEE Trans. Very Large Scale Integr. Syst.* **2009**, *17*, 747–757. [CrossRef]
6. Talapatra, S.; Rahaman, H.; Mathew, J. Low complexity digit serial systolic montgomery multipliers for special class of $GF(2^m)$. *IEEE Trans. Very Large Scale Integr. Syst.* **2010**, *18*, 847–852. [CrossRef]
7. Talapatra, S.; Rahaman, H.; Saha, S.K. Unified digit serial systolic montgomery multiplication architecture for special classes of polynomials over. In Proceedings of the 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, Lille, France, 1–3 September 2010; pp. 427–432.
8. Pan, J.-S.; Lee, C.-Y.; Meher, P.K. Low-latency digit-serial and digit-parallel systolic multipliers for large binary extension fields. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2013**, *60*, 3195–3204. [CrossRef]
9. Xie, J.; Meher, P.K.; Mao, Z. High-throughput digit-level systolic multiplier over $GF(2^m)$ based on irreducible trinomials. *IEEE Trans. Circuits Syst. II* **2015**, *62*, 481–485. [CrossRef]
10. Xie, J.; Meher, P.K.; Mao, Z.-H. Low-latency high-throughput systolic multipliers over $GF(2^m)$ for NIST recommended pentanomials. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2015**, *62*, 881–890. [CrossRef]

11. NIST. Boulder, CO, USA. Available online: http://www.csrc.nist.gov/publications (accessed on 25 September 2018).
12. Nangate Standard Cell Library. Available online: http://www.si2.org/openeda.si2.org/projects/nangatelib (accessed on 25 September 2018).