New Polynomial Basis Versatile Multiplier over $GF(2^m)$ for Low-Power On-Chip Crypto-Systems

Mustafa Khairallah  
Electronics and Communications Department  
Ain Shams University  
Email: khairallah@ieee.org

Maged Ghoneima  
Mechatronics Engineering Department  
Ain Shams University  
Email: m_ghoneima@ieee.org

Abstract—This paper presents a low-power, reduced-area finite field multiplier over $GF(2^m)$ for ultra-low-power devices. The proposed design supports any field $GF(2^m)$ with low-weight irreducible polynomial. The different implementations presented in this paper support 99% of fields with prime $m < 4096$. The proposed design is advantageous in terms of flexibility and hardware complexity. The design can perform multiplication over fields whose $m > 1024$, and all standard elliptic curves consuming 28.7µW and 4.0µW respectively, using the TSMC 65nm technology library. The design is demonstrated to operate at frequencies up to 500 MHz, allowing various trade-offs between power, energy, and performance. The proposed design is shown to use around 40% less area and 40% less power than the other designs proposed in the literature. Hence, it enables implementing more secure ciphers for almost the lower cost than other available designs.

I. INTRODUCTION

Radio Frequency Identification (RFID), Internet of Things (IoT), Implantable Medical Devices (IMDs), Wireless Sensor Networks (WSN), and wearable devices are all new emerging technologies that are based on communications between embedded systems. Almost all devices that use or implement these technologies are limited in terms of power and area. With more complex systems and technologies emerging, the design goal is to include a lot of functionality in a small device that is characterized by a limited power and area budget while maintaining the minimum required performance. These devices are either battery powered or wirelessly powered. For example, an RFID tag is passively powered by the reader through an air interface. The ISO/IEC 18000-3 Standard has specified that an RFID tag operating at 13.56 MHz has to consume less than 15uW at 1.5V, to guarantee an operating range of 1 meter [1]. Besides, an RFID tag has to be almost a credit card size to be portable and easily used.

One of the main challenges facing the development of these new emerging devices is security processing, which is time and power consuming. Recent research shows that public key crypto-systems can consume 25–50% of the power available for an RFID tag [2]. In WSNs, they can consume around 72% of the energy available for handshaking between nodes, which is 5–10% of the WSN node’s budget [3]. Public key crypto-systems are used in vital security operations such as key exchange, digital signature and authentication. The major three algorithms for implementing public key cryptosystems, are RSA, Diffie-Hellman (DH) and Elliptic Curve Cryptography (ECC). While RSA relies on modular arithmetic, both DH and ECC [4], as well as the Galois/Counter Authenticated Encryption Mode (GCM) for the Advanced Encryption Standard (AES) [5], depend on finite field arithmetic. The widely adopted finite field arithmetic is based on either prime finite field operations or binary finite field operations. Binary finite field operations are known to be faster than prime finite field operations.

In this paper, we present a new binary finite field multiplier that can be used to multiply two field elements of any binary field $GF(2^m)$. The new multiplier takes the field elements $A$ and $B$, of width $GF(2^m)$ and irreducible polynomial $F(x)$, and computes $A \cdot B_{modF}(x)$. Using this multiplier, one can easily perform the standard ECC and DH algorithms. We also analyze the hardware complexity and power consumption of this new design.

II. MATHEMATICAL BACKGROUND

Public key cryptography depends on a hard mathematical problem, that cannot be evaluated practically, such as the discrete log problem in DH, the inverse function of scalar point multiplication in ECC, and integer factorization in RSA [6]. The discussion in this section is limited to ECC and DH, which are based on finite fields.

A. Elliptic Curve Scalar Point Multiplication

In the standard ECC, the primitive operation is scalar multiplication over one of the Weierstrass equations elliptic curves over binary field $GF(2^m)$ as defined in the IEEE 1363 [4]. The standardized EC over $GF(2^m)$ is presented by equation (1), where $b=1$ is commonly used.

$$y^2 + xy = x^3 + ax^2 + b, \text{ where } a,b \in GF(2^m) \quad (1)$$

The main operation in elliptic curve cryptography is point multiplication, which is defined as $sP = P + P + P + \ldots + P$, $s$ times, where $P$ is a point on the curve, and $s$ is an arbitrary positive integer. Several algorithms have been proposed to compute the point addition operation. [7] presents an efficient algorithm that avoids the costly operations of finite field division/inversion, and uses only finite field multiplication, squaring and addition operations.
B. Finite Field Exponentiation

Let a be an arbitrary element of a finite field \( GF(2^m) \), and \( e \) be an arbitrary positive integer. Exponentiation over \( GF(2^m) \) can then be expressed by equation (2), where \( B_i \) is defined by equation (3) [8]. Both equations combined show that finite field exponentiation is based on repeated multiplication and squaring over \( GF(2^m) \).

\[
b = a^e = a^{\sum_{i=0}^{m-1} e_i 2^i} = a^{e_0} (a^2)^{e_1} (a^4)^{e_2} \ldots (a^{2^{m-1}})^{e_{m-1}} = \prod_{i=0}^{m-1} B_i \tag{2}
\]

\[
B_i = (a^2)^{e_i} = \begin{cases} a^2, & \text{if } e_i = 1 \\ 1, & \text{if } e_i = 0 \end{cases} \tag{3}
\]

C. Common Finite Fields

A binary finite field \( GF(2^m) \) is defined by two parameters; \( m \) and the irreducible polynomial \( F(x) \). The general form for \( F(x) \) is \( F(x) = x^m + x^{k_1} + x^{k_2} + \ldots + 1 \), where \( k_1 \) defines the weight of the polynomial. In [9], a table of low-weight irreducible polynomials \( \forall m \), where \( 2 \leq m \leq 10,000 \) is presented. Practically, we are only interested in finite fields whose \( m \leq 4096 \), which is the maximum practical width for DH fields. Moreover, there are many published attacks on the discrete logarithm problem over binary finite fields where \( m \) is composite, i.e. not prime, such as [10] and [11].

III. RELATED WORK

In [12], the author presented a parametrized architecture for polynomial basis multiplier over \( GF(2^m) \). The proposed circuit supports polynomial basis multiplication based on all types of irreducible polynomials of degree \( m \), where \( m \geq k_1 + 4 \). The author of [12] pointed out the architectural advantage of choosing a digit width \( D = 4 \) for Digital Serial Multiplication. However, the design in [12] has three drawbacks: the configuration needs to be decided before implementation, the hardware complexity increases linearly with the increase of field size, and finally it uses MSD-First Serial Multiplication algorithm. The MSD-First algorithm is characterized by lower performance compared to the LSD-First algorithm [8].

A versatile architecture for a polynomial basis multiplier over \( GF(2^m) \) was presented in [13]. It used tristate buffers to select the most significant bits of the result, so that it can support any finite field \( GF(2^m) \) with \( m \) less than the circuit width \( M \). However, the circuit in [13] does not support any field with \( m > M \).

In [14], the authors presented a low latency polynomial basis multiplier. The design supports all fields, where \( m \geq 2k_1 - 1 \), and has a theoretical latency of \( 2k_1 + 1 \). The proposed circuit is partially versatile, i.e. it supports all fields \( GF(2^m) \) where \( 1 \leq m \leq M, M \) is the width of the data-path.

In [15], a new high performance architecture for elliptic curve point multiplication was proposed, and implemented for the five recommended NIST polynomials. Their results for Digit-Serial Multiplication for different values of \( D \) show that the product of \( area \times D \) varies in a small range for \( 4 \leq D \leq 82 \), and is large for \( D < 4 \).

IV. NEW MULTIPLIER DESIGN

A. Mathematical Equations

The new finite field multiplier, proposed in this paper, is based on the LSD-First Multiplication algorithm. Equation (4) describes the overall operation of the multiplier over \( GF(2^m) \).

\[
A \cdot B_{modF}(x) = (A \cdot B_{modF}(x)) + ((Ax^{D}_{modF}(x)) \cdot B_{1modF}(x)) + ((Ax^{2D}_{modF}(x)) \cdot B_{2modF}(x)) + ((Ax^{3D}_{modF}(x)) \cdot B_{3modF}(x)) + \ldots \tag{4}
\]

Observing equation (4), LSD multiplication can be simplified into two primitives, (5) and (6), that can be used iteratively to perform the full multiplication algorithm.

\[
A \cdot B_{i modF}(x) \tag{5}
\]

\[
Ax^{D}_{modF}(x) \tag{6}
\]

To avoid the drawbacks of the architecture presented in [12], two ideas are proposed:

1) Use an LSD-first approach.
2) Design a fixed width \( M \) circuit that can be used to perform multiplication over any field, under the conditions discussed later in this section.

As a result, all circuit parameters, except for latency, are field-independent, which provides flexibility to system designers working on tightly constrained systems in terms of power and area. To achieve so, equations (5) and (6) can be written in the form of equations (7) and (8).

\[
A \cdot B_{i modF}(x) = \left( \sum_{j=0}^{[\frac{M}{D}] - 1} A_j \cdot B_{i} x^{jM}_{modF}(x) \right) \tag{7}
\]

\[
Ax^{D}_{modF}(x) = \left( \sum_{j=0}^{[\frac{M}{D}] - 1} A_j x^{D} x^{jM}_{modF}(x) \right) \tag{8}
\]
A final full-width calculation still needs to be performed in equations (7) and (8), which is \( \text{mod}F(x) \). However, as mentioned in section II, there is a low-weight irreducible polynomial for most of the prime fields below \( m = 4096 \). For example, given the list of irreducible polynomials in [9], out of 564 irreducible polynomials for binary fields with prime \( m \) where \( 1 < m < 4096 \), there are 144 fields with \( k_1 > 125 \) (25.53%), 96 with \( k_1 \) greater than 252 (16.84%) and 6, \{2633, 3169, 3511, 3911, 3929, 3943\}, with \( k_1 \) greater than 1020 (1.06%). Consequently, \( M \) can be wisely chosen to reduce the \( \text{mod}F(x) \) operation to a single multiplication and addition operation.

One of the challenges of designing a versatile multiplier is the selection of the excess bits that should be reduced. The proposed idea is to left-align all the inputs, so that these \( D \) bits are always the \([M + 4 : M]\) bits in the \((\lceil \frac{m}{2} \rceil - 1)\)th iteration. Since addition is performed over \( GF(2) \), which is a simple carry-free bit-wise xoring, the results will be independent of the bit alignment.

Fig. 1 shows the straight forward data-path for equations (7) and (8). In the first iteration on \( A_{\lceil \frac{m}{2} \rceil - 1} \), the \( D \) MSBs are stored to be used for reduction, otherwise, they are xored with the \( D \) LSBs from the previous iteration. One final iteration is performed after all the \( M \)-bit words of \( A \) have been processed to multiply the \( D \) MSBs of the result by \( F(x) \), and add the result to the \( M \) LSBs of the output of the last iteration on \( A_0 \). Finally, the output word of each iteration is xored with its corresponding word from the previous iteration on \( B \). The only difference between equations (7) and (8) is that the multiplier is placed only on the data path of \( F(x) \) as the \( A_j \cdot B_i \) multiplication is replaced by a \( D \)-bit left shift operation on \( A_j \), as shown in the right half of Fig. 1.

The remaining question to answer is how to choose circuit width \( M \), and digit width \( D \) efficiently. The choice of \( D \) affects the power consumption of the circuit, and the complexity of \( \text{mod}F(x) \) operation. In [2], the authors analyzed the power consumption of a group ECC over \( GF(2^{163}) \) engines for different choices of \( D \). The minimum power consumption was achieved at \( D = 4 \). Moreover, at \( D = 4 \) the \( \text{mod}F(x) \) reduction includes the calculation of multiples of \( F(x) \) between 0 and \( 15F(X) \), which requires only 2 addition operations using \( 2M \) XORs as presented in [12].

On the other hand, the choice of \( M \) mainly affects latency and energy consumption. Two versions of the design have been implemented, one for \( M = 1024 \) and another for \( M = 256 \). Fig. 4 shows the relationship between latency and field size \( m \) for \( 0 < m \leq 4096 \). The 1024-bit version is convenient for applications where both ECC and DH are computed as it allows computing operations over large fields with relatively low latency and area compared to bit-serial multipliers. The 256-bit version is convenient for applications where only ECC is needed. The maximum latency for the largest field recommended by NIST for ECC, \( m = 571 \), is \( \frac{571}{4} \) in the 1024-bit version and \( \frac{257}{1024} \) in the 256-bit version.

**B. Compact Implementation**

The straight forward implementation presented in Fig. 1 includes a lot of redundant logic that is only used for a short period of time during a single multiplication operation. For example, one of the multiply blocks is only used once every single multiplication operation. In order to avoid such redundancies, a new compact design is presented. The design is based on a processing element that consists of one \( M \)-bit-by-4-bit multiplier, one \( M \)-bit adder, one 4-bit adder block, one \( M \)-bit 2:1 multiplexer and one \( M \)-bit register. A binary finite field multiplier over \( GF(2^m) \) can be built using any number of processing elements (PEs), including only one PE. The target of the proposed design is to represent a new low power versatile binary field multiplier. The proposed design has a group of features, making it superior to its peers:

1. All circuit parameters, except latency, are independent of the field element width \( m \), but rather on the number of processing elements used.
2. For \( M \leq m \), latency is almost equal to the least

![Diagram](image-url)
reported latency for a digit-serial multiplier, $m/4$.

3) Ultra-low-power consumption. For example, it consumes only $4\mu W$ for $GF(2^{163})$ multiplication, where $M = 163$.

1) Processing Element: The processing element, presented in Fig. 2, multiplies a 4-bit number by an M-bit number resulting in a (M+3)-bit number, the most-significant 3 bits are outputs of the processing element, while the least-significant M-bits are added to another M-bit input. The addition results can be delayed (stored in a M-bit register) or routed directly to the final adder which adds the least-significant 4-bits of result to a 4-bit input value. The choice of the parameter $M$ is entirely based on the latency, power and area costs.

2) Overall Architecture: The overall architecture presented in Fig. 3 use the processing element to perform Digit Serial Multiplication. The control unit is responsible for selecting which inputs are fed to the PE at each clock cycle while the Regs block consists of a group registers used to store intermediate and output values.

3) Power Optimization: In contrary to the straightforward implementation in Fig. 1 or the MSD-First implementation in [12], most of the registers used in the proposed compact architecture are idle for large periods of times. Making use of this fact, the clock gating mechanism has been applied to these registers, minimizing the total consumed power.

V. RESULTS AND EXPERIMENTAL ANALYSIS

The hardware complexity of the proposed design is presented in Table I compared to the most recent similar designs in literature. In terms of area, the proposed design uses smaller area than similar designs. In spite of the low latency of [12], which is a function of $k_1$ ($k_1$ is typically much smaller than $m$), its area is a quadratic function of $m$, so it leads to a very large area for large fields. Latency lies in one of four regions, shown in Fig. 4:

1) If $m \leq M$, latency is equal to latency in [12], given $D = 4$.
2) If $M < m < D \cdot M$, latency lies between that of [12] and the basic bit serial multiplier.
3) If $m = D \cdot M$, latency is equal to that of the basic bit serial multiplier, but with much lower area.
4) If $m > D \cdot M$, latency increases rapidly with m, so it will be inconvenient to use the design.

To assess the previous results, the appropriate values of $M$ should be identified. Two significant values have been found, choosing $D = 4$:

1) If only ECC is required, choosing $M = 256$, given ECC uses the five recommended fields by NIST, $m \in \{163, 233, 283, 409, 571\}$, latency lies between 163 and 384.
2) If DH is required, choosing $M = 1024$, given DH fields range until 4096, latency is less than $m$ for all values $m = 4096$. Latency equals $m$ for $m = 4096$. Moreover, the implementation still supports ECC fields with latency $\frac{m}{4}$.

The proposed architecture has been synthesized using the 65nm TSMC technology library for $D = 4$, and $m \in \{163, 233, 256, 1024\}$. A configurable version of the designs proposed in [12] and [13] have also been implemented, and synthesized using the same technology library. The area and power results are presented in Table II. The frequency of $140KHz$ is chosen for comparison with [2], but it should be noted that the design proposed in this paper can operate at frequencies up to $500MHz$. In [2], a full ECC engine was proposed. It used 11,831 gates, and 4 50$\mu W$ for a data path, where $M = 163$ while the multiplier is the largest block in the

<table>
<thead>
<tr>
<th>Proposed</th>
<th>$7M + 3D - 2$</th>
<th>$7M + 6D - 6$</th>
<th>$T_{XOR} + T_{MUX}$</th>
<th>$T_{XOR} + 2T_{MUX}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[12]</td>
<td>$6m + 18$</td>
<td>$14m + 7$</td>
<td>$T_{XOR}$</td>
<td>$T_{XOR} + T_{AND}$</td>
</tr>
<tr>
<td>[13]</td>
<td>$2M$</td>
<td>$0$</td>
<td>$+T_{PR}$</td>
<td>$T_{AND} + 2T_{MUX}$</td>
</tr>
<tr>
<td>[14]</td>
<td>$\frac{m^2 + m}{2}$</td>
<td>$5m - 1$</td>
<td>$+\log_2(m)T_{XOR}$</td>
<td>$+\log_2(m)T_{XOR}$</td>
</tr>
<tr>
<td>AND</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$6M + D$</td>
</tr>
<tr>
<td>OR</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$6m + 7$</td>
</tr>
<tr>
<td>Tristate buffer</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$4m$</td>
</tr>
<tr>
<td>INV</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$2k_1 + 1$</td>
</tr>
<tr>
<td>FFs</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$2k_1 + 1$</td>
</tr>
<tr>
<td>Latency</td>
<td>$2m/4$</td>
<td>$2m/4$</td>
<td>$2m/4$</td>
<td>$2m/4$</td>
</tr>
</tbody>
</table>

Fig. 3. The overall architecture of the proposed design

Fig. 4. Latency vs. Field Size for different designs: (a) Proposed design with $D = 4$ and $M = 256$ (b) Proposed design with $D = 4$ and $M = 512$ (c) Proposed design with $D = 4$ and $M = 1024$ (d) Basic Bit Serial Multiplier (e) [12] (f) [13] with M=256 (M is the maximum width of the operands)
design. Results show that ECC over $GF(2^m)$ where $m > 163$ can be used within RFID tags power budget. They also show that the proposed design uses around 40% less area and 40% less power than the design proposed in [12]. Although the design proposed in [13] has lower area, the proposed design consumes slightly less power and has much better latency.

Energy, represented by $E = L * P/f$, where $L$ is the latency, $P$ is the total power and $f$ is the operating frequency, will be used as a figure of merit for the sake of comparison. The operating clock frequency can be varied up to its maximum value, as long as the total power does not exceed the energy allocated by the higher level system. Examples are shown in Table III. The results show better energy results over both the designs proposed in [12] and [13]. Since the total power is composed of static leakage power, which is constant, and dynamic power, which varies linearly with the operating frequency. Thus, the total energy decreases with increasing the operating frequency. Hence, a trade-off exists between power, energy and response time. An example of such a trade-off is presented in the last three rows of Table III.

VI. Conclusion

The design of a new low-power, reduced area versatile polynomial basis multiplier over $GF(2^m)$ has been presented. The design uses the LSD first multiplication algorithm, with a fixed width data-path to perform multiplication over $GF(2^m)$ for any $m$. The hardware complexity, performance, power and energy consumption of the proposed design has been analyzed and discussed. The proposed design was shown to use around 40% less area and 40% less power than the similar designs proposed in the literature. Hence, the proposed design was shown to enable achieving higher security levels in ultra-low-power devices.

REFERENCES


