# On Efficient Implementation of Accumulation in Finite Field Over $GF(2^m)$ and its Applications

Pramod Kumar Meher, *Senior Member, IEEE*

***Abstract*—Finite field accumulation is the simplest of all the finite field operations, but at the same time, it is one of the most frequently encountered operations in finite field arithmetic. In this paper, we present a simple but highly useful modification of the conventional hardware implementation of accumulation in finite field over $GF(2^m)$. The critical path, as well as, the hardware-complexity are reduced in the proposed design by performing the accumulation operation using $m$ number of $T$ flip-flops instead of using a combination of $m$ number of XOR gates with equal number of $D$ flip-flops in dependent loop structures. The conventional design is found to involve nearly 39% more area, 53% more delay, and 40% more maximum ac power consumption compared with the proposed accumulator. The proposed finite field accumulator is used further for the implementation of serial/parallel polynomial-basis finite field multiplication and bit-serial inter-conversion between polynomial basis representation and normal basis representation over $GF(2^m)$. The area-time complexity of the proposed bit-serial/parallel multiplier is less than half of the best of the corresponding existing structures. The structure proposed for digit-serial/parallel multiplication for trinomials is found to involve nearly 56% less area-time complexity compared with the best of the corresponding existing multipliers; and the existing design of bit-serial basis conversion is found to involve nearly twice area-time complexity compared with the proposed design using the proposed finite field accumulator.**

***Index Terms*—Elliptic curve cryptography (ECC), error control coding, finite field, finite field addition, finite field multiplication, galois field, VLSI.**

## I. Introduction

FINITE fields are of great interest for their applications in elliptic curve cryptography (ECC) and error control coding. In recent years, it has received more attention due to the emergence of ECC as a potential candidate for realizing robust cryoptosystems in resource-constrained environments [1], [2]. Addition operation in finite field over $GF(2^m)$ is simpler compared with other field operations, since there is no carry propagation, and addition of any two bits can be performed simply by a logical XOR operation. But at the same time, it is one of the most frequently encountered operations in finite field arithmetic, because not only is it required to perform the other field operations like multiplication, squaring, and inversion, but also it is required to perform the basic operations like point-additions and point-doubling in elliptic curve groups [1]. Conventionally, finite field accumulation over $GF(2^m)$ is performed by a combination of $m$ number of XOR gates with equal number of $D$ flip-flops in dependent loop structure. In this paper, we present a simple modification of the conventional design of finite field accumulator (FFA) where the combination of XOR gates and $D$ flip-flops are replaced by $T$ flip-flops. We have shown that the critical path, as well as, the hardware-complexity can be significantly reduced by performing the accumulation operation by $T$ flip-flops. Although it is an apparently trivial and simple modification of the widely used conventional design, the proposed FFA can lead to significantly more efficient implementation of other finite field operations and point operations for ECC. We have presented two examples in this paper to establish the advantages of the proposed FFA.

Multiplication is a basic arithmetic operation in finite field, which is relatively more complex compared with the other field operations like addition and squaring. Division operations on the other hand can be performed by a lookup table arrangement or through a series of multiplications. The time involved in performing the multiplications, consequently, is an important concern for efficient realization of point operations in elliptic curve groups and error control coding. Finite field multipliers with different bases of representation have been realized to be used for various applications. Multiplication in polynomial basis is relatively simpler, offers scalability for the fields of higher orders, and does not require a basis conversion [3]. The polynomial basis multipliers are, therefore, more efficient, and more widely used compared with the multipliers in the other bases of representations. A large number of architectures have been proposed in the literature for efficient polynomial basis multiplication over $GF(2^m)$ in dedicated hardware platform [4]–[31]. Bit-serial polynomial-basis multipliers are well-suited for small embedded systems since the cost and size of hardware and bandwidth are major constraints in such systems [11]–[13], [31]. Scalability of throughput, however, is an important issue in realization of finite field multipliers to have a balance between the speed-performance required by the application on one side, and bandwidth/logic-resources available in the implementation environment on the other side. Digit-serial architectures for polynomial-basis multiplications [17]–[22] are, therefore, suggested in the literature for scalable implementation by appropriate choice of digit-size. We have shown here that by using the proposed FFA, it would be possible to design more efficient hardware for bit/digit-serial/parallel polynomial-basis finite field multiplication over $GF(2^m)$.

Manuscript received August 17, 2007; revised February 02, 2008 and February 12, 2008. First published February 03, 2009; current version published March 18, 2009. A portion of this paper (Section III-B) has appeared in the Proceedings of the 6th International Conference on Information, Communications, and Signal Processing, (ICICS'07), Singapore.

The author is with the School of Computer Engineering, Nanyang Technological University, 639798 Singapore (e-mail: aspkmeher@ntu.edu.sg).

Digital Object Identifier 10.1109/TVLSI.2008.2005288

Three different bases of representation over $GF(2^m)$ are of particular interest. Those are: polynomial basis, normal basis, and dual basis. Out of the three bases, polynomial basis, and normal basis are more popular due to their higher practical relevance [32], [33]. Each of these bases has some advantages over the other for efficient realization of finite field arithmetic. Normal basis is a good choice for squaring of element over $GF(2^m)$ since squaring is performed in normal basis just by a cyclic right-shift, while in case of polynomial bases squaring is performed by bit-extension through insertion of 0 between the consecutive bits followed by modular reductions to reduce the extended polynomial of degree $2m-2$ to degree $m-1$. Similarly, inversion involves less area and time-complexity in normal basis. But polynomial basis has superior performance in finite field multiplications. For efficient hardware implementation of a given application, it would be useful to perform multiplication and addition in polynomial basis while squaring and inversion can be performed in normal basis. It is therefore useful to have an efficient hardware for conversion of normal basis to polynomial basis and vice versa. A hardware-efficient bit-serial design for basis conversion to be used for low-cost mobile and embedded systems is proposed by Li [34]. In this paper, we have reviewed the bit-serial converter presented in [34] and modified that to a more efficient form by using the proposed FFA.

The rest of this paper is organized as follows. A simple mathematical formulation for derivation of the proposed FFA is presented; and its efficiency over conventional implementation is discussed in Section II. Structures of bit-level and digit-level serial/parallel finite field multipliers using the proposed FFA is derived, and their advantages over the existing serial/parallel multipliers over $GF(2^m)$ are presented in Section III. In Section IV, we have reviewed an existing design [34] for bit-serial conversion from normal basis to polynomial basis and vice versa; and shown further that the basis converter can be implemented more efficiently by using the proposed FFA. Conclusions are presented in Section V.

## II. Finite Field Accumulator

Let the finite field $GF(2^m)$ be defined by an irreducible polynomial of degree $m$, given by

$$Q(x) = x^m + q_{m-1} \cdot x^{m-1} + \cdots + q_2 \cdot x^2 + q_1 \cdot x + 1 \quad (1)$$

where $\{q_i \text{ for } 1 \leq i \leq m-1\} \in GF(2)$. $Q(x)$ introduces a polynomial basis $\{1, z, z^2, \ldots, z^{m-1}\}$ (where $z$ is a root of $Q(x)$), which is used to represent the field elements. $A$ and $B$ be any two arbitrary elements in $GF(2^m)$, represented by the polynomial basis in the form of polynomials of degree $(m-1)$ as

$$A = \sum_{i=0}^{m-1} a_i z^i \qquad B = \sum_{i=0}^{m-1} b_i z^i \quad (2)$$

where $a_i$ and $b_i \epsilon \{0,1\}$, for $i = 0, 1, \ldots, m-1$.

Addition is the simplest operation in $GF(2^m)$, which is performed by bit-by-bit XOR operations of the pair of operand words, such that the addition of any two field elements, $S = A + B$, is given by

$$S = \sum_{i=0}^{m-1} s_i z^i \quad (3a)$$

where

$$s_i = a_i \oplus b_i \quad (3b)$$

for $i = 0, 1, \ldots, m-1$.

Since, no carries are generated during additions, the successive accumulation of $n$ number of finite field elements $D_j$ for $j = 0, 1, 2, \ldots, n-1$ can be given by

$$S = \sum_{j=0}^{n-1} D_j \quad (4a)$$

where

$$s_i = d_{i,0} \oplus d_{i,1} \oplus d_{i,2} \cdots \oplus d_{i,(n-1)} \quad (4b)$$

for $i = 0, 1, \ldots, m-1$, and

$$D_j = \sum_{i=0}^{m-1} d_{i,j} z^i \quad (4c)$$

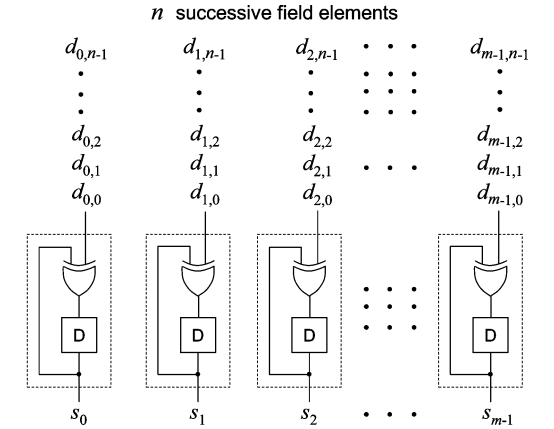
for $j = 0, 1, \ldots, n-1$.

The conventional design of an FFA over $GF(2^m)$ is shown in Fig. 1. It consists of $m$ number of bit-level accumulation cells, where each such cell consists of a two-input XOR gate and a $D$ flip-flop. Structure of each bit-level accumulation cell and its characteristic table are shown in Fig. 1(b). The input elements $D_j$ for $0 \leq j \leq n-1$ are fed sequentially in bit-parallel form to the FFA where each bit is fed to a bit-level accumulation cell. The accumulated output $S$ is obtained from the FFA after $n$ cycles. The duration of cycle period $T = T_X + T_{FF}$, where $T_X$ and $T_{FF}$ are the delays of two-input XOR gate and $D$ flip-flop, respectively. It can be observed that the characteristic table of bit-level accumulation cell [see Fig. 1(b)] is the same as that of a $T$ flip-flop. We can, therefore, replace each bit-level accumulation cell of the conventional FFA of Fig. 1 by a $T$ flip-flop; and can have an FFA consisting of $m$ number of $T$ flip-flops as shown in Fig. 2. It may be noted that the complexity of a $T$ flip-flop is nearly the same as that of a $D$ flip-flop since a $T$ flip-flop could be obtained by feeding the complementary output $\bar{Q}$ back as input to the $D$ flip-flop. The input for the resulting $T$ flip-flop is, however, required to be fed along with the clock to a NAND gate followed by an invertor to derive the clock derivation circuit in order to control the state toggling of the $T$ flip-flop according to the input bits. The states of all the $T$ flip-flops of the proposed FFA are reset at the beginning, and successive field elements to be accumulated are fed to the flip-flops in parallel. Since the state of a $T$ flip-flop toggles on arrival of each 1 as its input, the FFA performs the desired finite field accumulation when the input bits corresponding to all the elements are fed to the $T$ flip-flops in successive cycles.

TABLE I
AREA- AND TIME-COMPLEXITIES AND POWER CONSUMPTION OF THE PROPOSED AND THE CONVENTIONAL DESIGNS FOR FINITE FIELD ACCUMULATION OVER $GF(2^m)$

| Design | Area (sq.$\mu$m) | Delay (ns) | Max. AC Power ($\mu$W) | Area×Delay (sq.$\mu$m.ns) | Max. AC Power×Delay (J) |
|---|---|---|---|---|---|
| Conventional | $83.16m$ | $0.444n$ | $0.2257m$ | $36.9mn$ | $0.1002 \times 10^{-15}m$ |
| Proposed | $59.88m$ | $0.290n$ | $0.1608m$ | $17.37mn$ | $0.0466 \times 10^{-15}m$ |

Power estimation corresponds to maximum ac power consumption at switching frequency of 1 MHz in both cases at 25 °C and at 1.8 V operating voltage at unit drive strength. Note that $m$ is the order of finite field and $n$ refers to the number of field elements required to be accumulated.




| present state $Q$ | Input bit $X$in | next state $Q^+$ | remark on state |
|---|---|---|---|
| 0 | 0 | 0 | no change |
| 1 | 0 | 1 | no change |
| 0 | 1 | 1 | toggle |
| 1 | 1 | 0 | toggle |

Fig. 1. Typical design of an accumulator over $GF(2^m)$. (a) Conventional finite field accumulator. (b) Bit-level accumulation cell and its characteristic table.

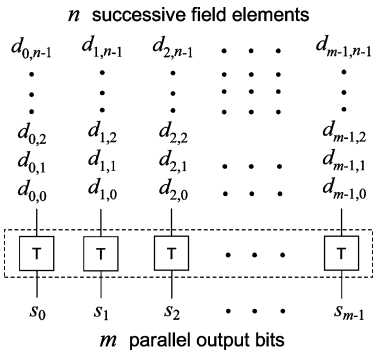



Fig. 2. Proposed $T$ flip-flop-based accumulator over $GF(2^m)$.

The area- and time-complexities of conventional FFA and the proposed FFA are listed in Table I. We have obtained the area of two-input XOR gate, $D$ flip-flop and $T$ flip-flop along with their worst-case intrinsic delays and maximum power consumption at unit drive strength using TSMC 0.18-$\mu$m process 1.8-V SAGE-X standard cell library databook [35]. Using those data, we have estimated the complexities mentioned in Table I. The complexity of $T$ flip-flop is derived from that of the equivalent $D$ flip-flop where the clock derivation circuit is replaced by a NAND gate followed by an invertor. The conventional accumulator is found to involve nearly 39% more area, 53% more delay, and 40% more maximum ac power consumption compared with the proposed accumulator.

## III. SERIAL-PARALLEL MULTIPLIER OVER $GF(2^m)$

In terms of the input/output (I/O) structuring, all these multipliers over $GF(2^m)$ can be classified into three basic forms: e.g., parallel-in parallel-out (or bit-parallel) architectures, serial-in serial-out (or bit/digit-serial) architectures, and serial-in parallel-out (or serial/parallel) architectures. In bit-parallel designs, a complete operand word is processed in every cycle, where the bits of input multiplicands are fed in parallel and the bits of output product word are also obtained in parallel. The bit-parallel designs [24]–[30] are intended mainly for high-speed implementation of the multiplication over $GF(2^m)$. They provide high throughput rate, but involve very high I/O bandwidth and large chip-area particularly for large values of the field order $m$. While large values of $m$ (160 or higher) are normally used in practice for ECC implementation to have adequate security [2], the portable and embedded devices where ECC is currently targeted are heavily constrained in terms or cost, size, and power-consumption. The bit-parallel architectures, therefore, are not well-suited for such resource-constrained systems. The bit/digit-serial structures, take only one new input bit/digit during a cycle and produce one output bit/digit per cycle. They are compact and may be opted for implementation of ECC in highly constrained systems [5]–[7], but cannot be used for high-speed applications. The digit-serial designs on the other hand, offer scalability of hardware and throughput, but the number of pipelining latches in the existing digit-serial systolic structures add substantial complexity to the overall area and time-complexity of the system [18]–[21]. Guo and Wang [18] have derived a systolic digit-serial/parallel architecture; and that has been improved further in [19]–[21] to reduce the critical path. Song and Parhi [17] have also proposed an efficient digit-serial/parallel architecture for finite field multiplication to achieve less area-complexity and a short critical path. In case of serial/parallel designs [11], [17], [22], the bits of one of the operands are fed in parallel and the bits of output are also obtained in parallel, while the other input operand is fed either in bit-serial or in digit-serial manner. The National Institute of Standards and Technology (NIST) [32] has recommended five binary finite fields for ECC implementation, out of which two are generated by the trinomials, $Q(x) = x^{233} + x^{73} + 1$ and $Q(x) = x^{409} + x^{87} + 1$. Efficient bit-parallel structures have,

therefore, been suggested in the last few years for finite field multiplications over $GF(2^m)$ based on irreducible trinomials [25]–[27] to achieve minimum critical path and least gate-complexity. But for large values of $m$, the critical path of these designs are too high. Efficient implementation of digit-serial/parallel multipliers in $GF(2^{233})$ are suggested in [22], [23], for high-throughput by concurrent multi-bit processing.

Keeping these facts in view, by using the proposed finite field accumulator, in this section, we have derived a bit-serial/parallel multiplier based on a general irreducible polynomial and a digit-serial/parallel multiplier for polynomial basis multiplications over $GF(2^m)$ for trinomials.

### A. Bit-Serial-Parallel Multiplication Over $GF(2^m)$ Based on General Polynomials

In one of the early papers, Song and Parhi [11] have suggested a semi-systolic architecture for serial-parallel implementation of multiplication over $GF(2^m)$. To have higher throughput rate without proportionate increase in hardware, bidirectional data-flow schemes are used in semi-systolic designs for serial-parallel multipliers in [10] and [12]. A hardware-efficient LSB-first serial/parallel multiplier over $GF(2^m)$ is suggested in [13] for the trinomial-based binary extension fields $GF(2^{193})$ and $GF(2^{239})$. An efficient modular reduction technique is suggested in [14] to speedup the computation by partitioning the product expression of the traditional Mastrovito's serial multiplier, and concurrent by processing. In [15], Bharathwaj and Narasimham have simplified the modulo operation using the Itoh Tsujii algorithm [16], which could be used for area-time efficient realization for small values of field order $m$. In a recent paper [31], an area-time efficient serial-parallel semi-systolic architecture is suggested where the field multiplication is implemented by bidirectional modulo reduction operation and gate-level optimization. In the following, we derive a more efficient bit-serial architecture for finite field multiplication using the proposed FFA.

*Algorithm Formulation for the Bit-Serial/Parallel Multiplication:* The product of two finite field elements $A$ and $B$ in polynomial basis representation over $GF(2^m)$ is given by

$$C = A \cdot B \bmod Q(z). \tag{5}$$

To derive a recurrence relation for recursive implementation of the proposed bit-serial multiplier, (5) can be expanded and represented by the polynomial sum

$$C = \sum_{i=0}^{m-1} b_i \cdot \left(z^i \cdot A \bmod Q(z)\right). \tag{6}$$

Equation (6) can be expressed as a finite field accumulation

$$C = \sum_{j=0}^{n-1} X_j \tag{7}$$

where each $X_j$ is a polynomial of degree $(m-1)$, and given by

$$X_j = b_j \cdot A^j \tag{8}$$

for $A^0 = A$, and $A^j = [z^j \cdot A \bmod Q(z)]$, such that $A^{j+1}$ can be obtained from $A^j$ recursively as

$$A^{j+1} = z \cdot A^j \bmod Q(z). \tag{9}$$

By polynomial expansion of right-hand side of (9), we can find

$$cA^{j+1} = [a_0^j z + a_1^j z^2 + a_2^j z^3 + \cdots + a_{m-2}^j z^{m-1} + a_{m-1}^j z^m] \bmod Q(z) \tag{10a}$$

where

$$A^j = \sum_{i=0}^{m-1} a_i^j z^i. \tag{10b}$$

Since $z$ is a root of $Q(x)$ given by (1), one can have

$$z^m = q_{m-1} \cdot z^{m-1} + \cdots + q_2 \cdot z^2 + q_1 \cdot z + 1. \tag{11}$$

Substituting the expansion of $z^m$ on (10a), the reduced form of $A^{j+1}$ can be obtained as

$$A^{j+1} = a_0^{j+1} + a_1^{j+1} z + \cdots + a_{m-1}^{j+1} z^{m-1} \tag{12a}$$

where

$$a_0^{j+1} = a_{m-1}^j \tag{12b}$$

and

$$a_i^{j+1} = a_{i-1}^j \oplus q_i \cdot a_{m-1}^j \tag{12c}$$

for $i = 1, 2, \ldots, m-1$.

For bit-serial/parallel multiplication, (7)–(9) can be implemented recursively, where each recursion consists of three steps, e.g., the modular reduction of (9) [realized according to (12)], AND operations of (8), and finite field accumulation of (7).

*1) Proposed Structure For the Bit-Serial/Parallel Multiplication:* The proposed structure for bit-serial/parallel implementation of multiplication over $GF(2^m)$ is shown in Fig. 3. It consists of three units: such as the modular reduction unit (MRU), AND unit (AU), and an FFA. The MRU consists of $m$ number $D$ flip-flops and $(m-1)$ number of reduction cells "$RC(i)$" for $i = 1, 2, \ldots, M-1$. At the first cycle, the state of the $D$ flip-flops of MRU are initialized by loading the operand word $A$ in parallel. During each of the subsequent cycles, the MRU performs a modular reduction according to (12). The function of the reduction cells of MRU is depicted in Fig. 3(b). It may be noted that the structure and function of a reduction cell depends on the value of coefficient-bits "$q_i$" (for $1 \le i \le m-1$) of the field polynomial $Q(x)$. For $q_i = 1$, the $i$th reduction cell performs an XOR operation of its input from left with its input from top to produce an output to be fed to the $D$ flip-flop on its right. For $q_i = 0$, the reduction cell does not have any additional function other than transferring the input available from a $D$ flip-flop on its left to the $D$ flip-flop on its right. For $q_i = 0$, therefore, the reduction cell should be removed and $D$ flip-flop output should be fed directly to the next $D$ flip-flop on its right. In most practical ECC applications, the primitive irreducible polynomial $Q(z)$ is

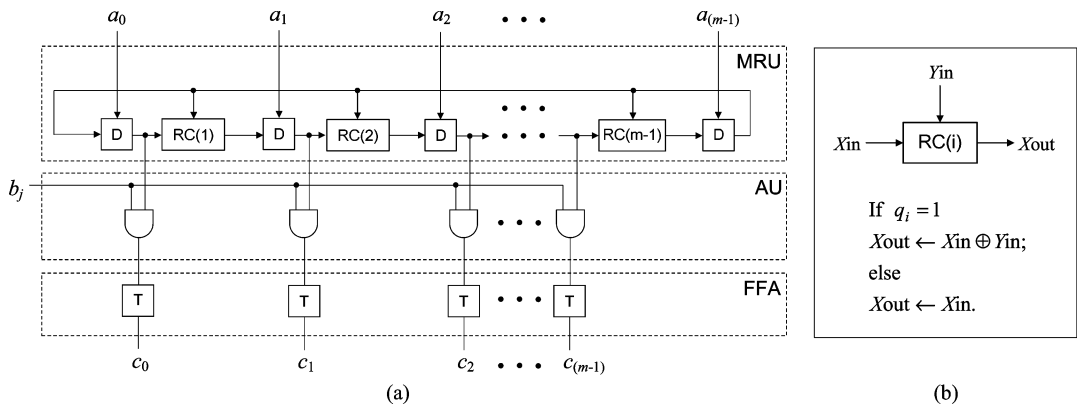



Fig. 3. Structure of the bit-serial/parallel multiplier over $GF(2^m)$ using $T$ flip-flop-based accumulator. (a) The proposed serial/parallel multiplier. (b) Function of the reduction cells (RC).

TABLE II
HARDWARE- AND TIME-COMPLEXITIES OF THE BIT-SERIAL/PARALLEL MULTIPLIERS FOR $GF(2^m)$-BASED ON GENERAL FIELD POLYNOMIALS

| designs | AND | XOR | Registers | Area | Latency | ACT | Cycle Time | Area-Time |
|---|---|---|---|---|---|---|---|---|
| Song and Parhi [11] | $2m$ | $2m-1$ | $4m+2$ | $412.47m$ | $m+1$ | $m$ | $T_A+T_X+T_{FF}$ | $239.40m^2$ |
| Batina *et al* [12] | $4m$ | $5m-2$ | $5m$ | $469.02m$ | $m/2+1$ | $m/2$ | $T_A+2T_X+T_{FF}$ | $172.01m^2$ |
| Garcia-Martinez *et al* [13] | $2m$ | $2m-1$ | $4m$ | $306.03m$ | $m+1$ | $m$ | $T_A+T_X+T_{FF}$ | $177.62m^2$ |
| Meher [31] | $2m$ | $5m-2$ | $4m-2$ | $385.86m$ | $m/2+1$ | $m/2$ | $T_A+T_{X3}+T_{FF}$ | $141.52m^2$ |
| Proposed | $m$ | $m-1$ | $3m$ | $212.89m$ | $m+1$ | $m$ | $T_A+T_X$ | $61.63m^2$ |

In case of proposed structure each of the input registers consists of $m$ $D$ flip-flops and the FFA contains equal number of $T$ flip-flops. The area, cycle period, and area-time are, respectively, in square-nm, ns, and sq-um-ns. $T_A$, $T_X$, $T_{FF}$, and $T_M$ are the gate delays of AND gate, XOR gate, latch, and multiplexer, respectively. Area of a three-input XOR gate is taken to be equivalent to that of 2 two-input XOR gates, and three-input XOR delay ($T_{X3}$) is taken to be two times that of two-input XOR delay. The structure of [11] requires 4 m number of 2:1 multiplexers in addition to the gate counts, which is not shown in this table, but the multiplexers are taken into account for computing the area-complexity of the structure.

a trinomial or pentanomial (of very high degree for ECC implementation) [2]. Except a few (one in case of trinomial and three in case or pentnomial), all the coefficients $q_i$ of the irreducible polynomial in the range $1 \leq i \leq m-1$) are, therefore, zero. For trinomial field polynomials, except only one reduction cell, all other reduction cells may be removed.

*2) Complexity Considerations:* Since there is no feedback loop in the structure at the output, the critical path of the structure $T_C = T_A + T_X$, where $T_A$ and $T_X$, are the delays of a two-input AND gate and a two-input XOR gate. It performs a multiplication over $GF(2^m)$ in $m$ cycles, where the duration of cycle period $T = T_A + T_X$. In Table II, we have listed the hardware requirements including the number of different gates, registers and multiplexors along with the time-complexity metrics, e.g., cycle time and average computation time (ACT) in terms of number of cycles of the proposed structure, as well as, the existing structures for bit-serial/parallel multiplication over $GF(2^m)$ based on any general polynomial. Using TSMC 0.18-$\mu$m process 1.8-V SAGE-X standard cell library data [35] for area and worst-case intrinsic delays of gates and flip-flops at unit drive strength, we have estimated the area-complexities and time-complexities of different structures; and listed in Table II. Time-complexities of different structures are estimated as the product of critical path $T_C$ with the ACT [$T = T_C \times \mathrm{ACT}$]. The proposed structure is found to have the significantly lower area complexity, shorter cycle times and less area-time complexity compared to those of the existing structures. The area-time complexity of the proposed design is found to be less than half of the best of the corresponding existing structures for serial/parallel multipliers in $GF(2^m)$.

### *B. Digit-Serial/Parallel Multiplier Over $GF(2^m)$ Based on Trinomials*

We derive here an efficient digit-serial/parallel structure for polynomial-basis multiplications in $GF(2^m)$ based on irreducible trinomials (which could also be extended for pentanomials), where the critical path, as well as, the hardware-complexity are reduced by the proposed FFA [23].

*Algorithm Formulation for the Digit-Serial/Parallel Multiplication:* To derive the recurrence relations for concurrent processing of the bits of a digit in the proposed multiplier, (6) can be broken into separate sums of $w$ terms as

$$C = \sum_{j=0}^{n-1} \sum_{i=0}^{w-1} b_{jw+i} \cdot \left[ z^{jw+i} \cdot A \bmod Q(z) \right] \qquad (13)$$

where $n = \lceil m/w \rceil$ and $b_j = 0$ for $m \leq j \leq (nw-1)$. Equation (13) can be expressed further in recursive form

$$C = \sum_{j=0}^{n-1} X_j \qquad (14)$$

where

$$X_j = \sum_{i=0}^{w-1} b_{jw+i} \cdot \left[z^i \cdot A^j \bmod Q(z)\right] \tag{15}$$

for $A^0 = A$, and $A^j = [z^{jw}.A \bmod Q(z)]$ can be written as

$$A^j = \sum_{k=0}^{m-1} a_k^j \cdot z^k \bmod Q(z). \tag{16}$$

Note that in the recursions defined by (14)–(16), the partial product generation and modular reductions are, respectively, performed according to (15) and (16), while the accumulation of the reduced polynomials are performed according to (14). Interestingly, all the reduced polynomials of (16) can be computed independently, and can be added in any desired sequence. After modular reduction, each individual term $X_j$ for $0 \le j \le (n-1)$ in (14) being transformed into a polynomial of order $(m-1)$, can be added by bit-by-bit XOR operations to obtain the desired product.

For efficient modular reduction of the individual terms of (15), we can find that

$$\begin{aligned} z^i \cdot A^j \bmod Q(z) &= \sum_{k=0}^{m-1} a_k^j \cdot z^{i+k} \bmod Q(z) \\ &= \sum_{k=0}^{m-i-1} a_k^j \cdot z^{i+k} + z^m \cdot \sum_{k=0}^{i-1} a_{(m-i+k)}^j \\ &\quad \cdot z^k \bmod Q(z). \end{aligned} \tag{17}$$

If $Q(z)$ is a trinomial of the form $Q(z) = z^m + z^l + 1$, then we can replace $z^m$ by $z^l + 1$ in (17) to find

$$\begin{aligned} z^i \cdot A^j \bmod Q(z) &= \left[\sum_{k=0}^{i-1} a_{(m-i+k)}^j \cdot z^k\right] \\ &\quad + z^l \cdot \left[\sum_{k=0}^{i-1} a_{(m-i+k)}^j \cdot z^k\right] + \left[\sum_{k=i}^{m-1} a_{k-i}^j \cdot z^k\right]. \end{aligned} \tag{18}$$

Note that the right-hand side of (18) is a polynomial of degree $(m-1)$ and the modular reduction is achieved by $i$ number of XOR operations required for adding the $i$ coefficients in the middle term, where $(l+w) \le (m-1)$.

*1) Proposed Structure for the Digit-Serial/Parallel Multiplication:* A conceptual block diagram of the proposed structure for the digit-serial/parallel multiplication over $GF(2^m)$ is shown in Fig. 4. It consists of a product-generator-cum-modular-reduction (PGCMR) unit along with an $m$-bit input register and an $m$-bit finite field accumulator. The input register of the structure consisting of $m$ $D$ flip-flops is initialized by one of the multiplicands $A$ (by providing the bits of $A$ as SET/RESET signal); and reloaded on every cycle such that $A^j$ is loaded on the $j$th cycle for $1 \le j \le (n-1)$. During the $j$th cycle of computation, PGCMR performs modular reduction to transform the polynomials $[z^i \cdot A^j \bmod Q(z)]$ (for $0 \le i \le w-1$ ) of degree $(m+i-1)$ to a polynomial of degree $(m-1)$, and performs AND operations of every bit of the reduced output with $b_{jw+i}$ followed by the field additions. The detail structure of PGCMR for word size $w = 8$ [for any irreducible trinomial $Q(z) = z^m + z^l + 1$ as the field polynomial, and satisfying the condition $(l+w) \le (m-1)$] is shown in Fig. 5. It consists of three combinational sections: the modular reduction section, the AND section, and the addition section. The modular reduction section consists of eight modular reduction cells [shown in Fig. 5(b)] for $w = 8$, where the $i$th modular reduction cell performs $i$ number of XOR operations according to (18) in parallel to produce $[z^i \cdot A^j \bmod Q(z)]$ during the $j$th cycle. The $i$th modular reduction cell consists of $i$ number of two-input XOR gates to perform the $i$ number of XOR operations of (18) in parallel. In total, the modular reduction unit, therefore, consists of $w(w+1)/2$ number of two-input XOR gates and takes time $T_X$ to perform all the XOR operations.

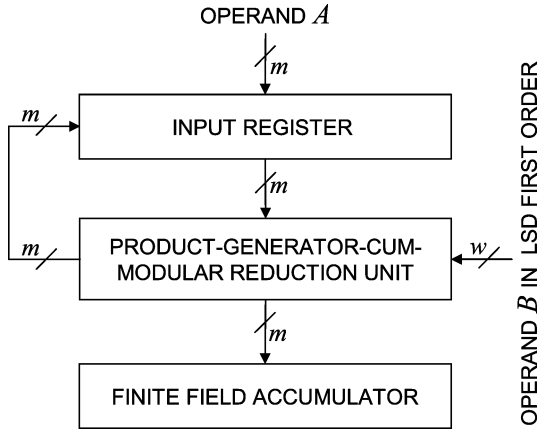



Fig. 4. Conceptual block diagram of the proposed architecture of the field multiplier over $GF(2^m)$.

The AND section consists of eight AND cells (AC). The function of an AC is shown in Fig. 5(c). It consists of $m$ number of two-input AND gates to perform the AND operations of (15). During each cycle, the AND section receives a digit (set of eight bits for $w = 8$) of the second operand $B$ in least significant digit (LSD)-first order, such that on the $j$th cycle it receives the bits $b_{jw+i}$ for $0 \le i \le w-1$. The $i$th AC performs $m$ number of AND operations of each bit of $[z^i \cdot A^j \bmod Q(z)]$ with $b_{jw+i}$. Each AC thus requires $m$ two-input AND gates. The AND section as a whole involves $wm$ number of two-input AND gates. It takes time $T_A$ to complete its operation, where $T_A$ is the propagation delay of a two-input AND gate. The finite field addition of the eight elements $[b_{jw+i} \cdot (z^i \cdot A^j \bmod Q(z))]$ for $0 \le i \le w-1$ of (15) are performed by bit-wise XOR operations in the addition section by an XOR logic tree consisting of seven XOR cells (XC). The function of each XC is shown in Fig. 5(d). It consists of $m$ number of two-input XOR gates to perform the bit-by-bit XOR operations of its pair of $m$-bit operands. The addition section requires seven XCs and requires $3T_X$ duration of time to complete its operations. The successive additions of (14) are performed by an FFA consisting of $m$ number of $T$ flip-flops. Note that the FFA also acts as the output register for this structure.

*2) Complexity Considerations:* The proposed structure for finite field multiplier over $GF(2^m)$ requires two $m$-bit registers and a PGCMR unit. The PGCMR unit, in general, requires $w$ modular reduction cells, equal number of AND cells and $(w-1)$ XOR cells. Since the $i$th modular reduction cell requires $i$ number of two-input XOR gates, the modular reduction unit as

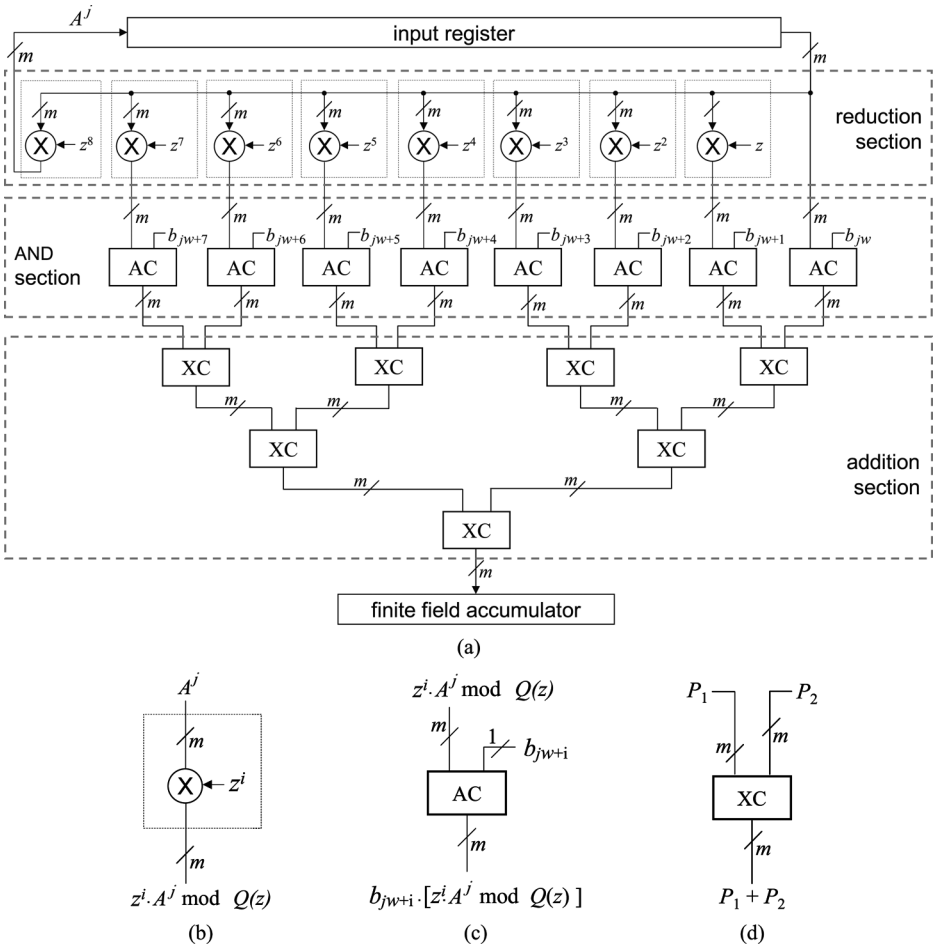



Fig. 5. Structure of the PGCMR unit of the multiplier in $GF(2^m)$ for $W = 8$. (a) The structure of PGCMR. (b) Function of a modular reduction cell. (c) Function of an AND cell (AC). (d) Function of an XOR cell (XC).

a whole requires $\lceil w(w+1)/2 \rceil$ number of two-input XOR gates. The AND section requires $wm$ number of two-input AND gates, while the addition section requires $(w-1)m$ number of two-input XOR gates. The computational delay of the modular reduction section, AND section and the addition section are $T_X$, $T_A$, and $(\log_2 w)T_X$, respectively. Since there is no feedback loop from the output register, the critical path of the structure $T_C = \max\{T_X + T_{FF}, T_A + (1+\log_2 w)T_X, T_{TF}\}$. The critical path of the structure thus amounts to $T_C = (1+\log_2 w)T_X$. It takes $\lceil m/w \rceil$ cycles, in general, to perform a finite field multiplication in $GF(2^m)$ based on any irreducible trinomial.

In Table III, we have listed the hardware requirements including the number of different gates, multiplexers, and registers along with the time-complexity metrics, e.g., cycle time, ACT, and latency in terms of number of computational cycles of the proposed structure, as well as, the existing structures. As shown in Table III, the structures of [18]–[21] require more than four times the number of registers, and involve nearly two times the number of two-input AND/XOR gates with $2m$ number of additional multiplexers compared with those of the proposed structure. They have nearly the same ACT but involve nearly three times more latency. The structure of [21] has the same critical path as the proposed one but the structures of [18] and [19] have substantially higher critical path. The structures of [17] and [22] involve the same number of cycles of ACT as the proposed structure, but involve relatively more area and longer critical path than the latter. The proposed one thus involves substantially lower area-time complexity compared with the other two. The area and time-complexities of different structures, estimated by using the TSMC 0.18-$\mu$m process 1.8-V SAGE-X standard cell library data [35] for area and worst-case intrinsic delays at unit drive strength, are listed in Table IV. It can be observed from Table IV that the proposed structure involves nearly 56% less area-time complexity compared with the best of the existing structures for $w = 8$.

## IV. Bit-Serial Converter of Basis Over $GF(2^m)$

The sets $\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$ and $\{\alpha, \alpha^2, \alpha^4, \ldots, \alpha^{2^{m-1}}\}$ are, respectively, called as the polynomial basis and the normal basis over $GF(2^m)$, where $\alpha$ is a root of the irreducible polynomial $Q(x)$ as given by (1). Let $A$ be a finite field element in polynomial basis representation, and $B$ be the normal basis representation of the same element. There exists one-to-one correspondence between these elements in two representations, such that one can be obtained from the other by linear transformations of the forms

$$\mathbf{B}^T = \mathbf{A}^T \mathbf{P} \tag{19a}$$

TABLE III
HARDWARE- AND TIME-COMPLEXITIES OF THE DIGIT-SERIAL STRUCTURES FOR FIELD MULTIPLICATION OVER $GF(2^m)$ BASED ON TRINOMIALS

| Designs | AND | XOR | Register | MUX | Latency | ACT | Cycle Time |
|---|---|---|---|---|---|---|---|
| Guo and Wang [18] | $w(2w+1)n$ | $2w^2n$ | $10wn$ | $2wn$ | $3n$ | $n$ | $w(T_A+2T_X+T_M)-T_M$ |
| Kim *et al* [19], [20] | $w(2w+1)n$ | $2w^2n$ | $10wn+n$ | $2wn$ | $3n$ | $n$ | $w(T_A+T_X+T_M)-T_M$ |
| Kim *et al* [21] | $2w^2n$ | $2w^2n$ | $8wn+4w$ | $2wn$ | $3n+2$ | $n$ | $T_A+\lceil\log_2(w+1)\rceil.T_X$ |
| Tang *et al* [22] | $wm$ | $wm+(w^2+w)/2$ | $2m+w$ | 0 | $n-1$ | $n-1$ | $T_A+(\lceil\log_2 w\rceil+2).T_X+T_{FF}$ |
| Song and Parhi [17] | $wm$ | $wm+(w^2+w)/2$ | $2m+w$ | $m$ | $n+1$ | $n$ | $T_A+(\lceil\log_2 w\rceil+1).T_X+T_{FF}$ |
| Proposed | $wm$ | $(w-1)m+(w^2+w)/2$ | $2m+w$ | 0 | $n$ | $n$ | $T_A+(\lceil\log_2 w\rceil+1).T_X$ |

$n=\lceil m/w\rceil$. $T_M$ is the delay of a 2:1 line multiplexer. The delays of three-input and four-input XOR gates is taken to be two times that of two-input XOR delay. The register size is represented in terms of number of bits. In case of proposed structure the input register consists of $m$ $D$ flip-flops and the finite field accumulator contains equal number of $T$ flip-flops. The ACT is represented in terms of number of computational cycles.

TABLE IV
AREA- AND TIME-COMPLEXITIES OF THE PROPOSED AND EXISTING DESIGNS FOR FIELD MULTIPLICATION OVER $GF(2^m)$ FOR $w=8$

| Design | Area (sq.um) | Cycle Period | Area-Time |
|---|---|---|---|
| Guo and Wang [18] | $1270.7m$ | 4.4837 ns | $712.2m^2$ |
| Kim *et al* [19], [20] | $1270.7m$ | 3.2589 ns | $517.6m^2$ |
| Kim *et al* [21] | $1144.3m$ | 0.7488 ns | $107.1m^2$ |
| Tang *et al* [22] | $432.4m$ | 1.1928 ns | $64.5m^2$ |
| Song and Parhi [17] | $459.0m$ | 1.0397 ns | $59.7m^2$ |
| Proposed | $409.1m$ | 0.7488 ns | $38.3m^2$ |

The area and area-time complexities are approximated for large values of $m$. The value of $n$ is approximated to $(m/w)$ for [18]–[21] although $n \geq (m/w)$. The area-time values are in sq.um.ns.

and

$$\mathbf{A}^T=\mathbf{B}^T\mathbf{T} \tag{19b}$$

where $\mathbf{P}$ and $\mathbf{T}$ are conversion matrices of size $m\times m$, and the elements of both these matrices $P_{i,j}$ and $T_{i,j}\in\{1,0\}$, for $0\leq i,\ j\leq m-1$. $\mathbf{A}$ and $\mathbf{B}$ are column vector representation of the elements $A$ and $B$.

For simple presentation of the proposed structure we use here the same example as that of [34] for basis conversion.

### A. Conversion From Polynomial Basis to Normal Basis

Let us consider a conversion from polynomial basis to normal basis over $GF(2^5)$ for $m=5$, and consider a primitive polynomial $Q(z)=z^5+z^4+z^2+z+1$. The polynomial basis and the normal basis may, respectively, be given by the linearly independent sets $\{1,\alpha,\alpha^2,\alpha^3,\alpha^4\}$ and $\{\alpha,\alpha^2,\alpha^4,\alpha^8,\alpha^{16}\}$. Since $\alpha$ is a root of $z^5+z^4+z^2+z+1=0$, we can have

$$\alpha^5=\alpha^4+\alpha^2+\alpha+1. \tag{20}$$

Besides, one can find that the elements of normal basis satisfy the condition [33]

$$\alpha^{16}+\alpha^8+\alpha^4+\alpha^2+\alpha=1. \tag{21}$$

Using (20) and (21), it is possible to map the normal basis to polynomial basis according to the following relations:

$$\begin{bmatrix}1\\ \alpha\\ \alpha^2\\ \alpha^3\\ \alpha^4\end{bmatrix}=\begin{bmatrix}1&1&1&1&1\\ 1&0&0&0&0\\ 0&1&0&0&0\\ 1&0&0&1&0\\ 0&0&1&0&0\end{bmatrix}\begin{bmatrix}\alpha\\ \alpha^2\\ \alpha^4\\ \alpha^8\\ \alpha^{16}\end{bmatrix}. \tag{22}$$

The normal basis representation $B=\Sigma_{i=0}^{4}(b_i\alpha^{2^i})$ of the polynomial basis representation $A=\Sigma_{i=0}^{4}(a_i\alpha^i)$ of a field element over $GF(2^5)$ may thus be given by

$$\begin{bmatrix}b_0\\ b_1\\ b_2\\ b_3\\ b_4\end{bmatrix}^T=\begin{bmatrix}a_0\\ a_1\\ a_2\\ a_3\\ a_4\end{bmatrix}^T\begin{bmatrix}P_{00}&P_{01}&P_{02}&P_{03}&P_{04}\\ P_{10}&P_{11}&P_{12}&P_{13}&P_{14}\\ P_{20}&P_{21}&P_{22}&P_{23}&P_{24}\\ P_{30}&P_{31}&P_{32}&P_{33}&P_{34}\\ P_{40}&P_{41}&P_{42}&P_{43}&P_{44}\end{bmatrix} \tag{23}$$

where the conversion matrix $\mathbf{P}$ is given by

$$\mathbf{P}=\begin{bmatrix}1&1&1&1&1\\ 1&0&0&0&0\\ 0&1&0&0&0\\ 1&0&0&1&0\\ 0&0&1&0&0\end{bmatrix}. \tag{24}$$

### B. Conversion From Normal Basis to Polynomial Basis

Using (20) and (21), one can also map the polynomial basis to normal basis according to the following relations:

$$\begin{bmatrix}\alpha\\ \alpha^2\\ \alpha^4\\ \alpha^8\\ \alpha^{16}\end{bmatrix}=\begin{bmatrix}0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&1\\ 0&1&0&1&0\\ 1&0&1&1&1\end{bmatrix}\begin{bmatrix}1\\ \alpha\\ \alpha^2\\ \alpha^3\\ \alpha^4\end{bmatrix}. \tag{25}$$

The polynomial basis representation $A$ of the normal basis representation $B$ of a field element over $GF(2^5)$ may thus be given by

$$\begin{bmatrix}a_0\\ a_1\\ a_2\\ a_3\\ a_4\end{bmatrix}^T=\begin{bmatrix}b_0\\ b_1\\ b_2\\ b_3\\ b_4\end{bmatrix}^T\begin{bmatrix}T_{00}&T_{01}&T_{02}&T_{03}&T_{04}\\ T_{10}&T_{11}&T_{12}&T_{13}&T_{14}\\ T_{20}&T_{21}&T_{22}&T_{23}&T_{24}\\ T_{30}&T_{31}&T_{32}&T_{33}&T_{34}\\ T_{40}&T_{41}&T_{42}&T_{43}&T_{44}\end{bmatrix} \tag{26}$$

TABLE V
HARDWARE- AND TIME-COMPLEXITIES OF THE BIT-SERIAL STRUCTURES FOR CONVERSION OF BASIS OVER $GF(2^m)$

| designs | D Flip-flop | T Flip-flop | XOR | Switch | Area | Cycle Time | Throughput | Area-Time |
|---|---|---|---|---|---|---|---|---|
| Li *et al* [34] | $m$ | 0 | $m$ | $m$ | $93.14m$ | $T_S + T_X + T_{FF}$ | $1/m$ | $45.08m^2$ |
| Proposed | 0 | $m$ | 0 | $m$ | $69.85m$ | $T_S + T_{TF}$ | $1/m$ | $23.06m^2$ |

The area and time required for switch are taken to be the same as that of a NAND gate in the existing design and the proposed modified design.

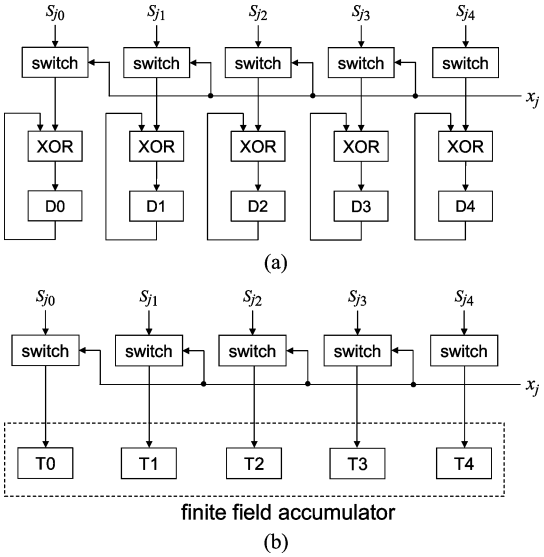



Fig. 6. Structures for basis conversion over $GF(2^5)$. (a) The existing structure. (b) Proposed structure for basis conversion.

where the conversion matrix $\mathbf{T}$ is given by

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{27}$$

The existing bit-serial structure [34] for basis conversion over $GF(2^m)$ is shown in Fig. 6(a). It consists of $m$ number of switches, $m$ number of two-input XOR gates and equal number of $D$ flip-flops. The bits of a finite filed element $X$ are broadcast to all the switches of the structure in bit-serial order, while the rows of conversion matrix $\mathbf{S}$ is fed in parallel to the switches, such that successive elements of a column of $\mathbf{S}$ are fed to a particular switch in successive cycles. It can be used as a universal basis converter by feeding with appropriate conversion matrix to the circuit. Conversion from normal basis to polynomial basis of representation can be performed when $\mathbf{S} = \mathbf{T}$ and $X = B$ according to (26). Conversion from polynomial basis to normal basis of representation can be performed when $\mathbf{S} = \mathbf{P}$ and $X = A$ according to (23). In the existing structure of Fig. 6(a), the output of each of the switches are accumulated in successive cycles by the combination of XOR gate and one bit-register. The combination of XOR gates and the bit-registers can be replaced by an FFA as shown in Fig. 6(b) to have a more efficient implementation.

The area- and time-complexities of the proposed structure and the existing structure [34] are listed in Table V for comparison. As shown in the table, the proposed design involves significantly less area and less time-complexity compared with the existing design of bit-serial basis converter in $GF(2^m)$. The existing design is found to involve nearly twice the area-time complexity of the design modified by using the proposed FFA.

## V. CONCLUSION

A simple but highly useful modification of conventional hardware implementation of accumulation in finite field over $GF(2^m)$ has been suggested, where the cycle time is substantially reduced by implementing successive additions in every clock period by $m$ $T$ flip-flops instead of using $m$ number of $D$ flip-flops and XOR-gates in data-dependent loops. The conventional accumulator is found to involve nearly 39% more area, 53% more delay, and 40% more maximum ac power consumption compared with the proposed accumulator. The proposed finite field accumulator is used to design serial/parallel polynomial-basis multipliers for $GF(2^m)$. The structure proposed for bit-serial/parallel multiplier for $GF(2^m)$ based on any general polynomial is found to have significantly lower area complexity, shorter cycle time and less area-time complexity compared to those of the existing structures. The area-time complexity of the proposed bit-serial/parallel multiplier is less than half of the best of the corresponding existing structures. The structure proposed for digit-serial/parallel multiplication over $GF(2^m)$ based on trinomials is also found to be significantly more efficient compared with the existing designs. By using the proposed accumulator, we have modified a low-cost bit-serial hardware design [34] for conversion of polynomial basis to normal basis and vice versa. The modified design of basis converter involves significantly less area and less time complexity compared with the existing design [34]. The existing design is found to involve nearly twice the area-time complexity of the design modified by using the proposed FFA. Further studies can still be made to find the advantages of this finite field accumulator in various other circuits for finite field arithmetic.

## REFERENCES

[1] I. Blake, G. Seroussi, and N. P. Smart, "Elliptic curves in cryptography," in *London Mathematical Society Lecture Note Series*. Cambridge, U.K.: Cambridge University Press, 1999.

[2] K. Nguyen and A. Weigl, "Fast Arithmetic in Hardware," in *Handbook of Elliptic and Hyperelliptic Curve Crytography*, H. Cohen and G. Frey, Eds. Boca Raton, FL: Taylor & Francis, ch. 26. [Online]. Available: http://www.csrc.nist.gov/publications

[3] 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 standard bases," *IEEE Trans Computers*, vol. 37, no. 6, pp. 735–739, Jun. 1988.

[4] C.-S. Yeh, I. S. Reed, and T. K. Truong, "Systolic multipliers for finite fields $GF(2^m)$," *IEEE Trans Computers*, vol. C-33, no. 4, pp. 357–360, Apr. 1984.

[5] C.-L. Wang and J.-L. Lin, "Systolic array implementation of multipliers for finite fields $GF(2^m)$," *IEEE Trans. Circuits Syst.*, vol. 38, no. 7, pp. 796–800, Jul. 1991.

[6] M. A. Hasan and V. K. Bhargava, "Bit-serial systolic divider and multiplier for finite fields $GF(2^m)$," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 972–980, Aug. 1992.

[7] W. C. Tsai and S.-J. Wang, "Two systolic architectures for multiplication in $GF(2^m)$," *IEE Proc.—Comput. Digit. Techniques*, vol. 147, pp. 375–382, Nov. 2000.

[8] C.-Y. Lee, E.-H. Lu, and J.-Y. Lee, "New bit-parallel systolic multipliers for a class of $GF(2^m)$," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'01)*, May 2001, vol. 4, pp. 578–581.

[9] C.-Y. Lee, E.-H. Lu, and J.-Y. Lee, "High-speed bit-parallel systolic multipliers for a class of $GF(2^m)$," in *Proc. Int. Symp. VLSI Technol., Syst., Appl.*, Apr. 2001, pp. 291–294.

[10] M. C. Mekhallalati, M. K. Ibrahim, and A. S. Ashur, "New low complexity bidirectional systolic structures for serial multiplication over the finite field $GF(q^m)$," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 1, pp. 101–113, Mar. 1998.

[11] L. Song and K. K. Parhi, "Efficient finite field serial/parallel multiplication," in *Proc. Int. Conf. Appl. Specific Syst., Arch. Processors (ASAP)*, Aug. 1996, pp. 72–82.

[12] L. Batina, N. S. Mentens, B. Ors, and B. Preneel, "Serial multiplier architectures over $GF(2^n)$ for elliptic curve cryptosystems," in *Proc. 12th IEEE Mediterranean Electrotech. Conf. (IEEE MELECON)*, May 2004, vol. 2, pp. 779–782.

[13] M. A. Garcia-Martinez, R. Posada-Gomez, G. Morales-Luna, and F. Rodriguez-Henriquez, "FPGA implementation of an efficient multiplier over finite fields $GF(2^m)$," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs (ReConFig)*, Sep. 2005, p. 5.

[14] S. Moon, J. Park, and Y. Lee, "Fast VLSI arithmetic algorithms for high-security elliptic curve cryptographic applications," *IEEE Trans. Consumer Electron.*, vol. 47, no. 3, pp. 700–708, Aug. 2001.

[15] S. V. Bharathwaj and K. L. Narasimhan, "An alternate approach to modular multiplication for finite fields $[GF(2^m)]$ using Itoh Tsujii algorithm," in *Proc. 3rd Int. IEEE-NEWCAS Conf.*, 2005, pp. 7103–7105.

[16] J. Guajardo and C. Paar, "Itoh-Tsujii inversion in standard basis and its application in cryptography," *Des., Codes, Cryptography*, vol. 25, pp. 207–216, 2002.

[17] L. Song and K. K. Parhi, "Low-energy digit-serial/parallel finite field multipliers," *J. VLSI Digit. Process.*, vol. 19, pp. 149–166, 1998.

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

[19] C. H. Kim, S. D. Han, and C. P. Hong, "An efficient digit-serial systolic multiplier for finite fields $GF(2^m)$," in *Proc. 14th Ann. IEEE Int. ASIC/SOC Conf.*, Sep. 2001, pp. 361–365.

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

[21] C. H. Kim, S. Kwon, and C. P. Hong, "A fast digit-serial systolic multiplier for finite field $GF(2^m)$," in *Proc. Asia South Pacific Des. Autom. Conf. (ASP-DAC)*, Jan. 2005, vol. 2, pp. 1268–1271.

[22] W. Tang, H. Wu, and M. Ahmadi, "VLSI implementation of bit-parallel word-serial multiplier in $GF(2^{233})$," in *Proc. 3rd Int. IEEE-NEWCAS Conf.*, Jun. 2005, pp. 399–402.

[23] P. K. Meher, "High-throughput hardware-efficient digit-serial architecture for field multiplication over $GF(2^m)$," presented at the 6th Int. Conf. Inf., Commun. Signal Process. (ICICS), Singapore, 2007.

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

[25] B. Sunar and C. K. Koc, "Mastrovito multiplier for all trinomials," *IEEE Trans Computers*, vol. 48, no. 5, pp. 522–527, May 1999.

[26] H. Wu, "Bit-parallel finite field multiplier and squarer using polynomial basis," *IEEE Trans. Computers*, vol. 51, no. 7, pp. 750–758, Jul. 2002.

[27] J. L. Imaña, J. M. Sánchez, and F. Tirado, "Bit-parallel finite field multipliers for irreducible trinomials," *IEEE Trans Computers*, vol. 55, no. 5, pp. 520–533, May 2006.

[28] A. Reyhani-Masoleh and M. A. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over $GF(2^m)$," *IEEE Trans. Computers*, vol. 53, no. 8, pp. 945–959, Aug. 2004.

[29] 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.

[30] H. Wu, "Low complexity bit-parallel multiplier for a class of finite fields," in *Proc. Int. Conf. Commun., Circuits Syst.*, Jun. 2006, vol. 4, pp. 565–568.

[31] P. K. Meher, "Systolic formulation for low-complexity serial-parallel implementation of unified finite field multiplication over $GF(2^m)$," in *Proc. 18th IEEE Int. Conf. Appl.-Specific Syst., Arch. Processors (ASAP)*, Montreal, QC, Canada, Jul. 2007, pp. 134–139.

[32] *National Institute of Standards and Technology, FIPS 186-2, Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication 186-2, 2000.

[33] P. Din, C. Wang, and J. Omura, "Normal basis of finite field $GF(2^m)$," *IEEE Trans. Inf. Theory*, vol. 32, no. 2, pp. 285–287, Mar. 1986.

[34] H. Li, "Design of reconfigurable general finite field multiplier in $GF(2^m)$," in *Proc. 2nd Ann. IEEE Northeast Workshop Circuits Syst. (NEWCAS)*, Jun. 2004, pp. 337–339.

[35] *TSMC 0.18 $\mu$m Process 1.8-Volt SAGE-X™ Standard Cell Library Databook*, Release 4.1, Artisan Components, Sep. 2003.

**Pramod Kumar Meher** (SM'03) received the B.Sc. degree (first class honors) and the M.Sc. degree (first class)(with electronics specials), both in physics, and the Ph.D. degree from Sambalpur University, Sambalpur, India, in 1976, 1978, and 1996, respectively.

He has a wide scientific and technical background covering physics, electronics and computer engineering. Currently, he is a Senior Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. He was a Professor with Utkal University, Bhubaneswar, India, in 1997–2002, a Reader in Electronics with Berhampur University, Berhampur, India, during 1993–1997, and a Lecturer in Physics in various Government Colleges (in India) during 1981–1993. His research interests include design of dedicated and reconfigurable architectures for computation-intensive algorithms pertaining to signal processing, image processing, secured communication, and bioinformatics. He has published more than 100 technical papers in various reputed journals and conference proceedings.

Dr. Meher was a recipient of the Samanta Chandrasekhar Award for excellence in research in Engineering and Technology for the year 1999. He is a Chartered Engineer of the Engineering Council of the United Kingdom, a Fellow of The Institution of Electronics and Telecommunication Engineers (IETE) of India, and a Fellow of the Institution of Engineering and Technology (IET), (formerly known as the Institution of Electrical Engineers), U.K. Currently, he is serving as Associate Editors of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and *The Journal of Circuits and Systems for Signal Processing*. More information about the author is available at http://www.ntu.edu.sg/home/aspkmeher/.