An Efficient and High-Speed Overlap-Free Karatsuba-Based Finite-Field Multiplier for FGPA Implementation

Moslem Heidarpur, *Member, IEEE*, and Mitra Mirhassani, *Senior Member, IEEE*

***Abstract*— Cryptography systems have become inseparable parts of almost every communication device. Among cryptography algorithms, public-key cryptography, and in particular elliptic curve cryptography (ECC), has become the most dominant protocol at this time. In ECC systems, polynomial multiplication is considered to be the most slow and area consuming operation. This article proposes a novel hardware architecture for efficient field-programmable gate array (FPGA) implementation of Finite-field multipliers for ECC. Proposed hardware was implemented on different FPGA devices for various operand sizes, and performance parameters were determined. Comparing to state-of-the-art works, the proposed method resulted in a lower combinational delay and area–delay product indicating the efficiency of design.**

***Index Terms*— Binary polynomial multiplier, field-programmable gate array (FPGA) implementation, finite-field multiplier, Galois field, hardware cryptography, overlap-free Karatsuba.**

I. Introduction

CRYPTOGRAPHY is used to provide confidentiality, data security, and authentication in many applications, such as communication devices [1], [2], autonomous vehicles [3], Internet of Things (IoT) [4], and healthcare [5].

There are generally two types of cryptography techniques: symmetric-key cryptography [6] and public-key cryptography [7]. The public-key cryptography allows all communication parties to safely communicate without any previously shared secret information among them. For this purpose, it is required to provide key establishment and digital signature for secure communications. Examples of public-key cryptography include Diffie–Hellman [8], RSA [9], ElGamal [10], and elliptic curve cryptography (ECC) [11]. Among these algorithms, the ECC has become the most dominant public-key cryptosystem, mainly because of its relatively short key size with respect to the security strength and implementation effectiveness [12], [13].

Manuscript received November 2, 2020; revised January 14, 2021; accepted January 31, 2021. Date of publication February 24, 2021; date of current version April 1, 2021. This work was supported by FedDev Ontario under Grant 815405 and in part by Windsor Essex Economic Development Corporation (WE-EDC) under Grant 813674. *(Corresponding author: Moslem Heidarpur.)*

The authors are with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada (e-mail: heidarp@uwindsor.ca; mitramir@uwindsor.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2021.3058509.

Digital Object Identifier 10.1109/TVLSI.2021.3058509

Compared to software implementations, hardware cryptography results in a higher speed and a lower cost while satisfying the efficiency and low-power requirements of electronic devices. There exists a number of VLSI [11], [14] and field-programmable gate array (FPGA) [15], [16] implementation of ECC cryptography algorithms.

The computation of the elliptic curve points can be broken down into finite-field operations in which the finite-field multiplier becomes the major component in the hardware implementation of ECC, where multiplier's size and delay dominate the overall area and throughput of the system. Accordingly, numerous research specifically targeted designing fast and efficient finite-field multipliers for cryptosystems [17]–[26].

One of the well-known multiplication algorithms is the Karatsuba algorithm (KA) [27]. This method intends to reduce the number of multipliers by replacing them with addition operations. While in the conventional algorithm (CA) [28], multiplication of two n digit numbers requires n^2 single-digit products, this number for KA is $n^{\log_2^3} \approx n^{1.58}$. Having a lower space complexity will reduce the total area required for hardware implementation of the multiplier.

On the other side, KA is an iterative algorithm that has a higher time complexity and, as a result, a lower speed and performance. Therefore, there is a tradeoff between area and delay requirements while choosing KA over CA algorithm. The Karatsuba method can be modified according to the restriction of the hardware implementation in order to boost the speed or reduce its area. Hardware implementation techniques (e.g., pipelining) are also utilized to increase multipliers efficiency [29]–[32].

Overlap-free Karatsuba algorithm (OKA) [33]–[35] in one of the algorithms proposed by researchers to reduce the combinational delay of the KA. This algorithm aims to remove an XOR gate in the critical path of KA. This change results in a considerable reduction in the theoretical delay in comparison to the original KA method. However, the CA still outperforms OKA in terms of delay due to its noniterative structure.

In order to create an efficient multiplication scheme, which does not suffer from the high area and at the same time provide sufficient speed, a new multiplication hardware strategy is proposed. The algorithm takes advantage of a base unit, implemented similar to the conventional method at the lower level, and combines the result in a method similar to the OKA. The proposed multiplication strategy was demonstrated to be

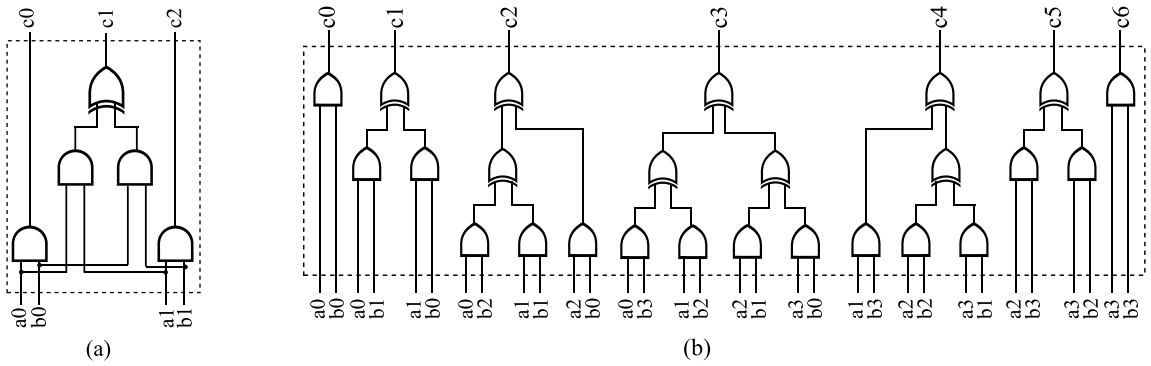

Fig. 1. DFG for hardware implementation of (a) 2- and (b) 4-bit conventional binary polynomial multiplier.

as fast as the CA with a resource utilization close to the Karatsuba.

Moreover, three algorithms, the conventional method of multiplication and the KA and OKA, were chosen for implementation on the FPGA. The outcome of this portion of work provides an overview of the FPGA implementations for various operand sizes in terms of area and delay. The result confirms that the CA is considerably faster than the other two algorithms. However, as the operand size increases, it requires more resources in terms of lookup tables. The proposed algorithms, on the other hand, demonstrated a high speed while having a lower area comparing to other algorithms. The contributions of this work are twofold. First, we implemented overlap-free Karatsuba binary polynomial multiplications on FPGA and compared the results with theoretical analysis as well as FPGA implementation of other algorithms for various operand sizes. Second, we proposed a lookup table-based overlap-free-based method to obtain a high-speed and efficient polynomial multiplier.

The rest of this article is organized as follows. Section II reviews conventional, KA, and OKA. Section III presents the details of the FPGA implementation procedure and results for the algorithms mentioned above and further introduces the proposed multiplication strategy. On FPGA performance and device utilization as well as comparison with similar and relevant works are discussed in Section IV. Finally, Section IV concludes this article.

II. Background

Polynomial multiplication, followed by a modular reduction, is one of the widely used operations in GF(2^n) whose efficiency greatly impacts the overall performance and cost of the system [30].

The multiplier's efficiency is evaluated according to its required space and the total delay of the multiplication based on the theoretical boundaries of these quantities using ideal two-input AND and XOR gates [30], [33], [36]. These works do not consider the limitations of the actual hardware; for example, the issue of limited gate fan-out in these devices is ignored in most instances. The multiplier's delay and space are simply calculated by the ideal configuration of the system and linear addition of the standard gate delays and area requirements. However, in actual implementation, there are other considerations, such as when the hardware restrictions dictate inserting of buffers, which in theoretical works are not of concern [37]. This article incorporates both theoretical and implementation results.

A. Conventional Algorithm

This section briefly reviews the CA for binary polynomial multiplication. We start with simple 2- and 4-bit multipliers and then extend it to n-bit multipliers. Assume that $A(x)$ and $B(x)$ are two polynomials of degree one in GF(2^n) as; $A(x) = a_1x + a_0$ and $B(x) = b_1x + b_0$. Since we are in GF(2^n), this 1-bit addition and multiplication operations are performed using logical XORs and ANDs, respectively. The corresponding data flow graph (DFG) for the first-order example is shown in Fig. 1(a), while Fig. 1(b) shows DFG for a 4-bit multiplier.

As an example, the simple 2-bit multiplier could be implemented using one XOR and four AND gates. These numbers for an n-bit conventional multiplier are as follows:

$$\mathrm{CA}_{\mathrm{XOR}}(n) = (n-1)^2 \tag{1}$$

$$\mathrm{CA}_{\mathrm{AND}}(n) = (n)^2 \tag{2}$$

where ($\mathrm{CA}_{\mathrm{AND}}$) and ($\mathrm{CA}_{\mathrm{AND}}$) are the total number of AND and XOR gates, respectively. Assuming ideal hardware condition and signal strength (no buffers required), the delay of the CA multiplier for the given example in Fig. 1(a) is

$$T_{\mathrm{CA}}(2) = T_x + T_a \tag{3}$$

where it is assumed that T_x and T_a are the delay of an XOR and an AND gate, respectively.

Generally, the highest delay of CA for multiplying two n-bit polynomials happens at term $(n-1)$ of output and is equal to

$$T_{\mathrm{CA}}(n) = T_a + \log_2(n)\ T_x. \tag{4}$$

Next, the original KA will be briefly reviewed.

B. Karatsuba Algorithm

Since the conventional multiplication method is not the most efficient method, other methods, such as KA [27] and its variations, have been developed. We will briefly review the original KA in the following.

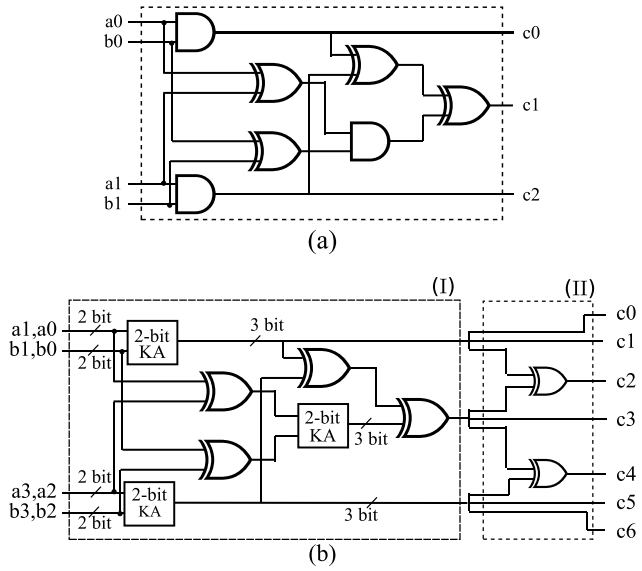

Fig. 2. DFG for hardware implementation of (a) 2- and (b) 4-bit Karatsuba binary polynomial multiplier, which uses three submultipliers. Block I performs splitting, submultiplication, and alignment, while block II calculates the overlaps of common terms.

For two polynomials of degree one (n = 2), $A(x) = a_1x + a_0$, and $B(x) = b_1x + b_0$, the DFG for hardware implementation is shown in Fig. 2(a). DFG for the 4-bit multiplier is represented in 2(b). It is worth noting that the 4-bit multiplier is constructed recursively using the 2-bit submultipliers.

DFG in Fig. 2 comprises two main blocks; block I includes splitting, submultiplication, and alignment stages. Furthermore, block II calculates the overlaps of common terms.

For an n-bit multiplier, consider two n-term polynomials, $A(x)$ and $B(x)$, which are in GF(2^n). These polynomials with $n-1$ degree are presented as follows:

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \tag{5}$$

$$B(x) = \sum_{i=0}^{n-1} b_i x^i \tag{6}$$

where a_i and b_i are polynomial coefficients.

$A(x)$ and $B(x)$ split into most (A_h, B_H) and least (A_L, B_L) significant halves as follows:

$$A(x) = x^m \sum_{i=0}^{m-1} a_{m+i}\ x^i + \sum_{i=0}^{m-1} a_m\ x^i = A_H\ x^m + A_L \tag{7}$$

$$B(x) = x^m \sum_{i=0}^{m-1} b_{m+i}\ x^i + \sum_{i=0}^{m-1} b_m\ x^i = B_H\ x^m + B_L \tag{8}$$

where $n = 2m$.

Using KA, the product of $A(x)B(x)$ could be calculated recursively as

$$\begin{aligned} A(x)B(x) &= \big(A_H x^m + A_L\big)\big(B_H x^m + B_L\big) \\ &= P_2(x)x^{2m} + [P_1(x) - P_2(x) - P_0(x)]x^m + P_0(x) \end{aligned} \tag{9}$$

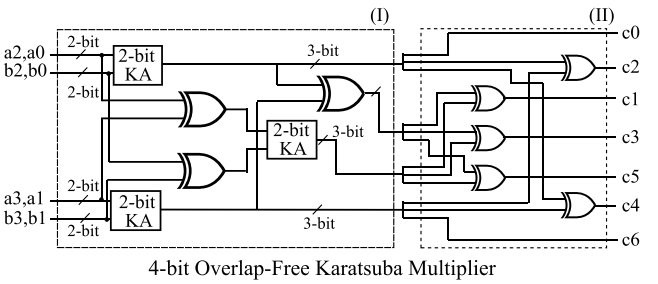

Fig. 3. DFG for hardware implementation of a 4-bit overlap-free Karatsuba binary polynomial multiplier. This method also uses three submultipliers. However, new splitting technique allows canceling extra T_x delay in KA.

where

$$\begin{aligned} P_2 &= A_H B_H \\ P_1 &= (A_H + A_L)(B_H + B_L) \\ P_0 &= A_L B_L. \end{aligned} \tag{10}$$

Equation (9) shows that three submultipliers are required in order to obtain the multiplication results using KA. In terms of complexity analysis, the number of gates required for implementation of an n-bit multiplier is

$$\mathrm{KA}_{\mathrm{XOR}}(n) = 3\ \mathrm{KA}_{\mathrm{XOR}}(n/2) + 4n - 4 \tag{11}$$

$$\mathrm{KA}_{\mathrm{AND}}(n) = 3\ \mathrm{KA}_{\mathrm{AND}}(n/2) \tag{12}$$

$$T_{\mathrm{KA}}(n) = 3\ T_x + T_{\mathrm{KA}}(n/2). \tag{13}$$

The nonrecursive forms of these equations are as follows:

$$\mathrm{KA}_{\mathrm{XOR}}(n) = 6\ n^{\log_2(3)} - 8n + 2 \tag{14}$$

$$\mathrm{KA}_{\mathrm{AND}}(n) = n^{\log_2(3)} \tag{15}$$

$$T_{\mathrm{KA}}(n) = T_a + \big(3\ \log_2(n) - 1\big)T_x. \tag{16}$$

Comparing (14) and (15) with (1) and (2) indicates a reduction of quadratic space complexity (n^2) in CA to subquadratic ($n^{\log_2(3)\approx=1.58}$) space complexity in KA.

On the other hand, comparing equations for time complexity (4) with (16) indicates an increment from $\log_2(n)\ T_x$ to $3\ \log_2(n)T_x$. Conclusively, KA reduces the multipliers area, but the price is having a lower speed.

C. Overlap-Free Karatsuba Algorithm

The OKA is a speed-optimized version of the original Karatsuba. In this method, to improve the longest path delay, inputs are split into odd and even orders instead of the high and low parts. Once more, it is assumed that $A(x)$ and $B(x)$ are two polynomial in $GF(2^n)$ and $n = 2m$. These polynomials could be rewritten as

$$A(X) = \sum_{i=0}^{m-1} a_{2i}\ x^{2i} + \sum_{i=0}^{m-1} a_{2i+1}\ x^{2i+1} \tag{17}$$

$$B(x) = \sum_{i=0}^{m-1} b_{2i}\ x^{2i} + \sum_{i=0}^{m-1} b_{2i+1}\ x^{2i+1}. \tag{18}$$

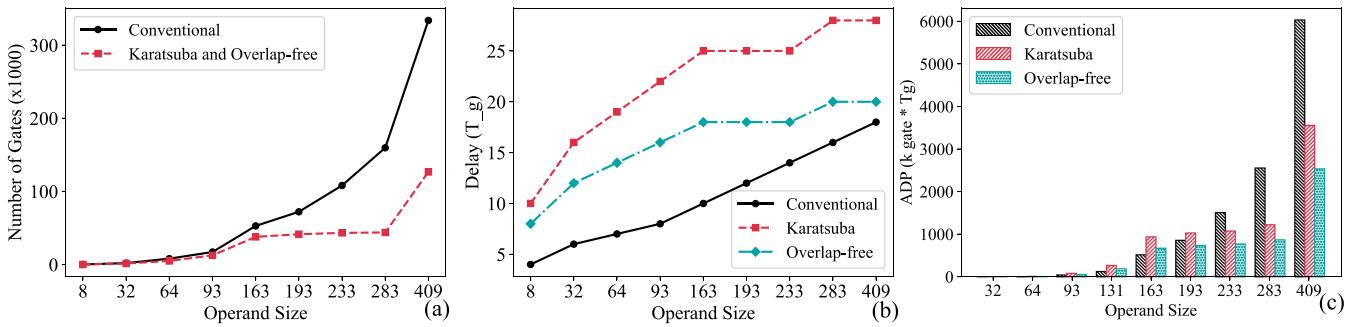

Fig. 4. (a) Total number of gates, (b) delay in terms of gate delay (T_g), and (c) ADP for hardware implementation of conventional, KA, and OKA, displayed as a function of their operand size. As the operand size grows, the number of gates required for hardware implementation of conventional becomes considerably larger than Karatsuba and overlap-free algorithms. In terms of the delay, conventional and overlap-free Karatsuba become closer, whereas Krartsuba's delay rises faster than these two algorithms. Overlap-free has the minimum ADP for operand sizes larger than 163.

Assuming that $y = x^2$, (17) and (18) can be rewritten as

$$A(x) = \sum_{i=0}^{m-1} a_{2i}\ y^i + x \sum_{i=0}^{m-1} a_{2i+1}\ y^i = A_e(y) + xA_o(y) \quad (19)$$

$$B(x) = \sum_{i=0}^{m-1} b_{2i}\ y^i + x \sum_{i=0}^{m-1} b_{2i+1}\ y^i = B_e(y) + xB_o(y) \quad (20)$$

where A_e and B_e include the even order and A_o and B_o include the odd-order terms of polynomials $A(X)$ and $B(x)$, respectively.

Following a similar approach to the KA, the polynomial multiplication could be calculated as:

$$\begin{aligned} A(x)B(x) &= (A_e(y) + xA_o(y)) \times (B_e(y) + xB_o(y)) \\ &= G_2(y)y + [G_1(y) - G_2(y) - G_0(y)]x + G_0(y) \end{aligned} \quad (21)$$

where

$$\begin{aligned} G_0 &= A_eB_e \\ G_1 &= (A_o + A_e)(B_o + B_e) \\ G_2 &= A_oB_o. \end{aligned} \quad (22)$$

Similar to the KA, the overlap-free algorithm also uses three submultipliers. However, term $G_0(y) + yG_2(y)$ has only odd exponents (x^{2n+1}), and term $G_1(y)$ contains only terms with even exponents (x^{2n}). Therefore, there is no overlap between the components of these two terms, which allows removing an XOR gate from the critical path of the Karatsuba multiplier.

As an example, the DFG of a 4-bit OKA multiplier is shown in Fig. 3. It should be noted that the DFG for a first-order OKA multiplier is similar to that of the KA, as shown in Fig. 2(a).

As shown in Fig. 3, for the 4-bit multiplier, the critical path of overlap-free is one XOR gate delay shorter than a Karatsuba multiplier.

By solving the recursive equation for area and space requirements, the estimated values of the OKA implementation are as follows:

$$\begin{aligned} \mathrm{OKA}_{\mathrm{XOR}}(n) &= 3\ \mathrm{OKA}_{\mathrm{XOR}}(n/2) + 4n - 4 \\ \mathrm{OKA}_{\mathrm{AND}}(n) &= 3\ \mathrm{OKA}_{\mathrm{AND}}(n/2) \end{aligned} \quad (23)$$

$$T_{\mathrm{OKA}}(n) = 2\ T_x + T_{\mathrm{OKA}}(n/2) \quad (24)$$

and solutions to recursive equations of (23) and (24) are as

$$\begin{aligned} \mathrm{OKA}_{\mathrm{XOR}}(n) &= 6\ n^{\log_2(3)} - 8n + 2 \\ \mathrm{OKA}_{\mathrm{AND}}(n) &= n^{\log_2(3)} \end{aligned} \quad (25)$$

$$T_{\mathrm{OKA}}(n) = T_a + \big(2\ \log_2(n) - 1\big)T_x. \quad (26)$$

Overlap-free Karatsuba space complexity is the same as the KA. However, its time complexity is decreased from $(3\ \log_2(n) - 1)T_x$ in KA to $(2\ \log_2(n) - 1)T_x$ in OKA.

III. Proposed Multiplication Strategy

In this section, a new and efficient implementation of the finite-field multiplier is proposed. The proposed implementation strategy is obtained by studying the theoretical boundaries of area and delay of conventional, Karatsuba, and overlap-free methods. An observed trend is used as a guideline for creating finite-field multipliers of various sizes. Moreover, hardware resources requirements and combinational delay for two different implementation approaches, theoretical gate-based analysis and FPGA, are evaluated.

The total number of gates for hardware implementation of the binary polynomial multiplication algorithms for different operand sizes is presented in Fig. 4(a). From this figure, it can be observed that considering small operand sizes, the number of gates required for implementing CAs is lower than that of the KA. However, as the operand size grows, the number of gates for implementing CA becomes substantially higher than Karatsuba and overlap-free. As an example, for operand size of 409 bits, the CA requires almost 163% more gates than the Karatsuba or overlap-free.

Fig. 4(b) shows the total combination delay in terms of gate delays for all three algorithms. It was assumed that the delay of the XOR and AND gates is the same; $T_x = T_a = T_g$. In this figure, the CA has the minimum delay; however, as operand size grows, the delay of conventional and overlap-free Karatsuba converge to almost the same value. On the other hand, Krartsuba's delay rises at a faster rate compared to the other two algorithms. It is also worth noting that the delays of 163, 193, and 233 bits recursive multipliers are the same since they have an equal number of stages. In order to clarify this point, the construction of these multipliers is shown in Fig. 5.

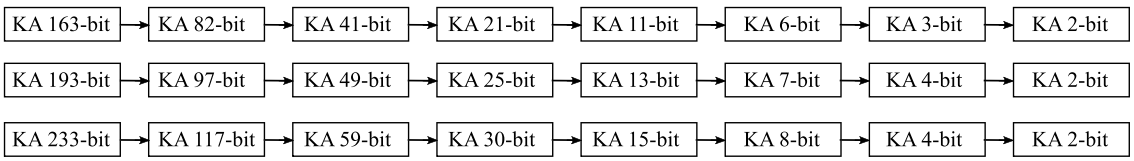

Fig. 5. Recursive Karatsuba polynomial multiplication for operand sizes of 163, 193, and 233 bits. Theoretical VLSI delay for implementation of these algorithms would be the same since they have an equal number of multiplier stages. The same structure was also used to implement an overlap-free algorithm.

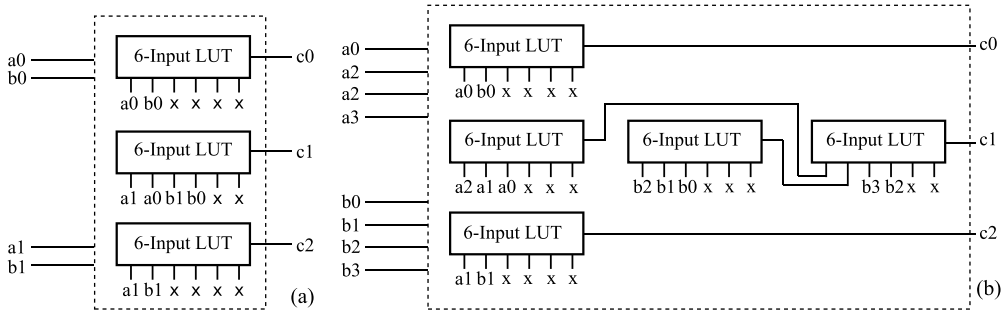

Fig. 6. DFG for FPGA implementation of (a) 2- and (b) 4-bit binary polynomial multipliers using six-input LUTs. Regardless of the different DFGs, FPGA implementation of all methods for 2- and 8-bit multipliers is the same. It is worth noting that this is a simple demonstration and actual architecture and routing is more complex.

Another interesting point is that the number of stages required to implement recursive multipliers, including KA and OKA, increases logarithmically with operand size rather than linearly. As an instance, in the case of the 233 bit, the first four stages recursively perform multiplication down to 15-bit multipliers. However, performing a 15-bit multiplication requires another four stages. It is also worth noting that the overall delay of these multipliers is determined by their corresponding number of stages.

To compare the efficiency of algorithms, area–delay product (ADP) was calculated for all algorithms and plotted in Fig. 4(c) where, on average, overlap-free has the minimum and conventional has the highest ADP.

Since our target platform is FPGAs, not digital gate-based devices, we investigated on-FPGA time and space analysis of these algorithms and validated the outcome by implementing algorithms. When it comes to the FPGA devices, the building blocks constructing most functions are lookup tables (LUTs) and not combinational gates. The LUTs are considered as universal gates where any function could be represented. Therefore, in order to accurately estimate the complexity and delay analysis, these structures should be implemented on the FPGA using the LUTs.

Fig. 6 shows an example DFG for FPGA implementation of a 2-bit and a 4-bit binary polynomial multiplier using six-input LUTs. As shown in this figure, irrespective of the number of gates and the difference in the DFGs, the LUT-based implementations are similar. The difference between LUT implementation of these algorithms in terms of performance and the number of LUTs becomes distinct as the operand size increases. Therefore, the theoretical estimations that are conventionally used to evaluate the efficiency of the algorithms cannot be simply extended to their actual FPGA implementation.

There are techniques to estimate the number of LUTs and delay for FPGA implementation of combinational circuits. However, in a pragmatic approach, all algorithms mentioned above for various operand sizes were implemented on FPGA. Fig. 7(a) and (b) shows the number of LUTs and combinational delay reported by the Vivado synthesizer tool for the implementation of conventional, KA, and OKA on an Artix-7 XC7A200TTFV1156-1 FPGA.

In terms of area, all algorithms utilize almost the same number of LUTs when their operand sizes are small. However, the difference in the area grows nonlinearly as the operand sizes increase. For example, for a 283-bit conventional multiplier, FPGA utilization is almost 69% higher than that of the KA. For 409 bit, the number of LUTs is approximately twice the number required for the Karatsuba implementation. It is expected that gap becomes wider for larger operand sizes.

The overlap-free implementation, however, utilizes a slightly higher number of LUTs comparing to Karatsuba. As an instance, a 409-bit overlap-free requires almost 2.7% more LUTs than the Karatsuba implementation.

Regarding the combinational delay, the CA is roughly 44% faster than Karatsuba on average. This number for overlap-free is smaller and is approximately 36%. The delays of Karatsuba and overlap-free are relatively close for small operand sizes. However, the difference grows with the size of multiplier operands in a way that for a 409-bit multiplier, overlap-free is almost 14% faster.

Despite the fact there are differences in detail, the results presented in Fig. 7 for FPGA implementation relate to the theoretical values represented in Fig. 4. It can be observed that the trend for the space complexity for both the theoretical and hardware implementation is similar; Karatsuba and overlap-free methods are more or less the same and require fewer resources than the conventional approach.

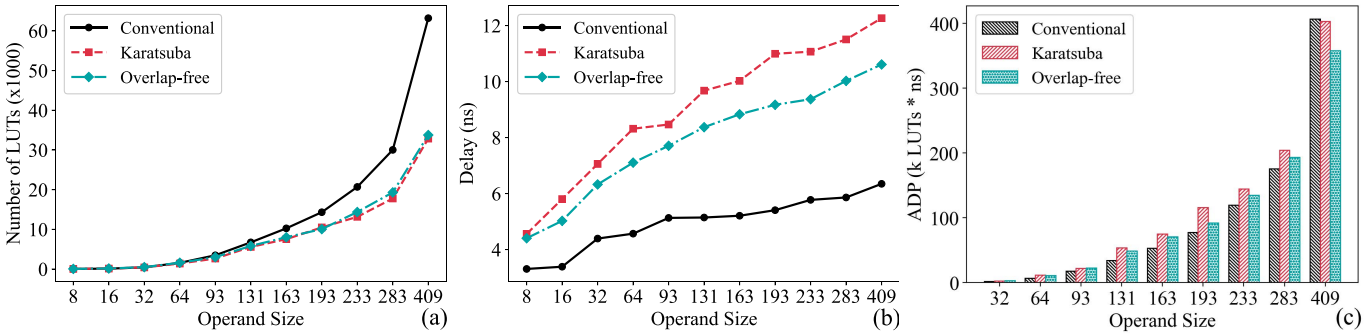

Fig. 7. (a) Number of LUTs, (b) combinational delay, and (c) ADP for FPGA implementation of bit-parallel binary polynomial multipliers using different algorithms. CA has the maximum LUTs and minimum delay. Overlap-free's resource utilization is close to Karatsuba, while it has a relatively higher speed. ADP for conventional method is smaller than other algorithms for all operand sizes except for 409 operand size where overlap-free has the minimum ADP.

TABLE I

NUMBER OF LUTS AND COMBINATIONAL DELAY FOR FPGA IMPLEMENTATION OF THE PROPOSED BIT-PARALLEL OVERLAP-FREE KARATSUBA-BASED MULTIPLICATION STRATEGY FOR DIFFERENT TRANSITION LEVELS. MINIMUM OF ADP FOR EACH OPERAND SIZE IS HIGHLIGHTED WITH A BOLD FONT

n	Level 1			Level 2			Level 3			Level 4		
	LUT	Delay	ADP	LUT	Delay	ADP	LUT	Delay	ADP	LUT	Delay	ADP
163	9307	5.786	53850	7996	6.460	51654	7102	6.832	**48520**	6634	7.565	50186
193	11786	6.811	80887	11640	6.683	77790	9683	6.887	**66686**	8843	7.780	68798
233	16814	6.841	115024	16797	7.475	125557	12349	6.930	**85578**	12439	6.930	86202
283	24251	6.846	166022	21596	7.155	154519	18746	7.662	**143631**	16475	8.720	143662
409	49211	6.856	337390	40199	8.085	325008	32880	7.889	**259390**	32361	9.170	296750
537	84199	7.524	633513	66957	8.098	542217	59863	8.559	51367	54177	8.906	**482500**
617	110119	7.528	828975	87007	8.106	705278	77623	8.556	664142	69967	9.405	**658039**

In terms of delay, for theoretical gate-based analysis as well as FPGA implementation results, the conventional method is the fastest, followed by the overlap-free and subsequently Karatsuba. In addition, the delay of overlap-free is close to Karatsuba on FPGA, while in theoretical results, it is closer to the CA.

Moreover, the ADP for all three algorithms was evaluated and plotted in Fig. 7(c). As results denote, the ADP for the conventional method is smaller than the other methods for operand sizes less than 409 bit. Numerically, a 283-bit multiplier using the conventional method is 14% more efficient than Karatsuba and 9% more efficient than the overlap-free method. These numbers for a 93-bit multiplier are 64% and 66% consecutively.

The trend indicates that the CA is the most efficient method for smaller operand sizes. It is also worth noting that for operand sizes larger than 93 bits, overlap-free is more efficient than the KA. It is expected that the overlap-free remains the most efficient method for larger operand sizes. Since the efficiency continues to trend toward the overlap-free method for large operand sizes, a hybrid approach should be considered to implement finite-field multiplication.

The DFG for the proposed overlap-free-based multiplication strategy (OBS) method is shown in Fig. 8. The highest level is based on the overlap-free. However, the conventional approach is used at the first level (level 1).

In order to illustrate the proposed hybrid approach, the structure of the OBS multiplier for each transition level is shown in Fig. 9. For each multiplier, the level in which the conventional method was used changed from level 1 to level 4, as shown in Fig. 9.

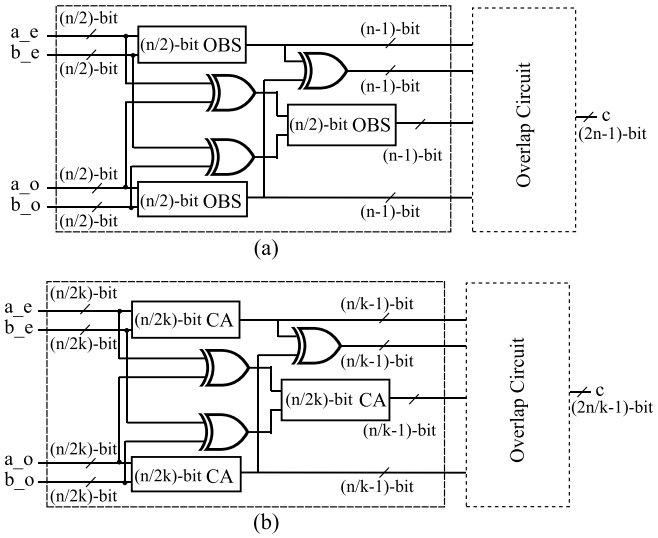

Fig. 8. Proposed overlap-free Karatsuba-based multiplication strategy for efficient FPGA implementation of binary polynomial multipliers. Higher level is recursive with three submultipliers similar to overlap-free Karatsuba. However, at a transition level (level k), it changes to a nonrecursive CA (CA submultipliers at the figure). (a) n-bit overlap-free-based multiplier (OBS) at the highest level. (b) (n/k)-bit overlap-free-based multiplier at transition level.

Table I summarizes FPGA resource utilization, combinational delay, and ADP for the proposed hybrid approach. As data shown in this table, using a CA at lower levels helps to reduce the number of LUTs required for the FPGA

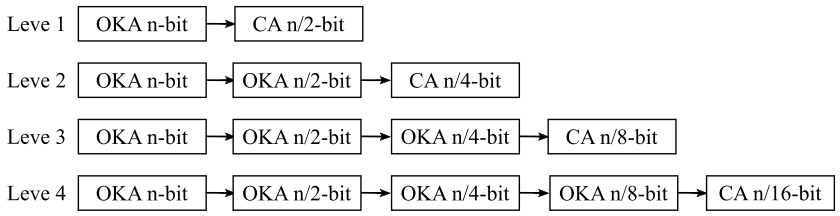

Fig. 9. Structure of the proposed multiplier where overlap-free transits to conventional at different levels. Choosing the transition level is a tradeoff between area and delay.

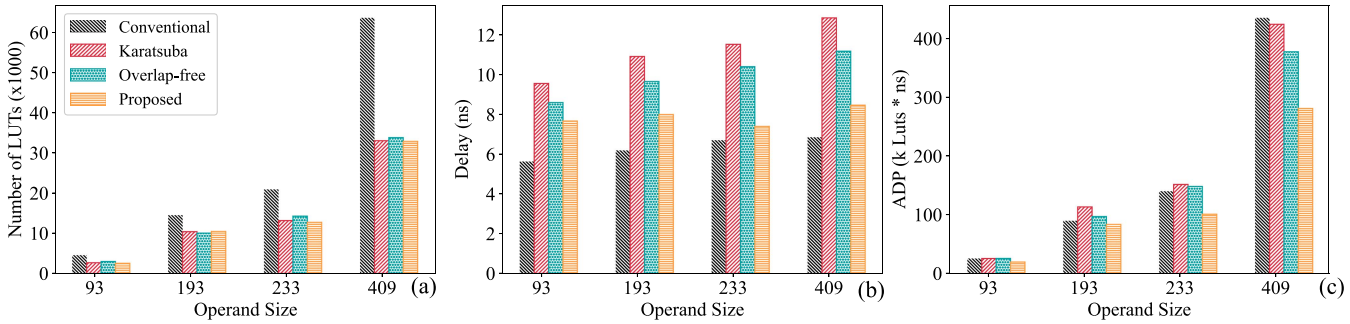

Fig. 10. (a) Number of LUTs and (b) combinational delay for FPGA implementation of bit-parallel finite-field multiplier using different algorithms. The number of LUTs for FPGA implementation of the proposed OBS is lower than conventional and is close to Kratsuba and overlap-free. However, in terms of delay, it is nearly as fast as CA.

TABLE II

AREA–DELAY IMPROVEMENT PERCENTAGE OF THE PROPOSED OVERLAP-FREE-BASED MULTIPLIER COMPARING TO OTHER ALGORITHMS

n	Conventional	Karatsuba	Overlap-free Karatsuba
93	23.8	23.7	24.0
193	5.2	25.0	12.2
233	32.8	38.5	36.5
409	36.1	34.5	26.4

TABLE III

SPEED CHANGE PERCENTAGE OF THE PROPOSED ALGORITHM WITH REGARD TO OTHER ALGORITHMS

n	Conventional	Karatsuba	Overlap-free Karatsuba
93	-36.1	19.8	10.8
193	-29.1	26.8	17.2
233	-10.4	35.8	28.9
409	-23.6	34.2	24.3

implementation of the multipliers. As an example, a 283-bit proposed multiplier with a CA at level 4 utilizes 32% less LUTs compared to that of the CA at level 1.

On the other hand, delay for level 4 is 27% higher than that for level 1. Therefore, choosing the transition level is a tradeoff between area and delay. In this work, the level that resulted in the minimum value of ADP (highlighted with a bold font in Table I) was selected to construct the proposed multipliers for each operand size. The most optimum transition level may vary with the operand size as in this table for a 537-bit multiplier level 4 has the minimum ADP, while for other sizes, transition to CAs at level 4 was the most efficient.

IV. RESULTS AND DISCUSSION

This section discusses the implementation results of the proposed method and comparison with other relevant works in this field.

A. Overlap-Free-Based Multiplier Hardware

Generally, to evaluate the effectiveness of an algorithm, theoretical boundaries for the area, delay, and ADP were calculated. However, a closer look reveals that when implemented on an FPGA, this might not be the case. Since the hardware utilization in an FPGA is based on the LUTs, such theoretical analysis needs to be revised. In addition, since the proposed method is based on LUT implementation, which is the building block of the FPGA, the estimates are more reflective of the actual cost and performance.

In most applications, binary polynomial multiplication is followed by a modular reduction to construct a finite-field multiplier. To compare algorithms, 93-, 193-, 233-, and 409-bit finite-field multipliers were developed using irreducible trinomials and implemented on FPGA (Artix-7 XC7A200TTFV1156-1) where device utilization and speed are compared with other algorithms in Fig. 10(a) and (b).

As shown in Fig. 10, the proposed method's resource utilization is considerably lower than conventional and is close to KAs, whereas it is nearly as fast as the conventional approach. Nevertheless, the main advantage of the proposed algorithm is its efficiency, which is investigated in the following.

Fig. 10(c) compares the ADP for the proposed and other algorithms, where the proposed method has the lowest ADP. Tables II and III summarize the ADP and speed improvement when compared to the other algorithms. In terms of ADP, the proposed method is, on average, 25% more efficient than the conventional, 31% more efficient than the Karatsuba, and 25% more efficient than the overlap-free algorithm.

TABLE IV

NUMBER OF LUTS AND DELAY FOR FPGA IMPLEMENTATION OF GF(2^{233}) FINITE-FIELD MULTIPLIERS ON DIFFERENT DEVICES USING VARIOUS SYNTHESIS TOOLS

Device Synthesis Tool	Virtex-5 XC6SLX75 XILINX ISE		Zynq XC7Z030 XILINX ISE		Spartan-7 XC7S100 XILINX Vivado		Cyclone-IV EP4CGX150DF Intel Quartus Prime	
Method	LUTs	Delay	LUTs	Delay	LUTs	Delay	LUTs	Delay
Proposed	14275	9.144	14275	4.955	12720	4.929	18901	7.633
Conventional	22821	6.319	22821	4.168	20920	4.146	37142	7.191
Karatsuba	17613	15.342	17644	7.756	13315	6.868	19814	10.792
Overlap-free Karatsuba	21376	14.116	21359	7.345	14255	6.340	20830	9.783

TABLE V

COMPARING FPGA RESOURCE UTILIZATION AND DELAY OF THE PROPOSED MULTIPLIER WITH THOSE OF RELEVANT WORKS

Reference	Algorithm	Slices	LUTs	Delay (ns)	ADP	n	Device & Tool	I/O Buffer	Reduction
Samanta et al. [38]	Modified Karatsuba	36	62	13.95	1367	8	Spartan-3E XC3S100E	Yes	No
This work	OBS	24	46	11.01	770		Xilinx ISE		
Zhou et al. [39]	Modified Karatsuba	3320	11404	7.68	113080	233	Virtex-5 XC5VLX50	No	Yes
This work	OBS	3857	11476	6.61	101351		Xilinx ISE		
J. Imaña [40]	Imaña's method	2354	5501	20.56	161498	113	ARTIX-7 XC7A200T	No	Yes
This work	OBS	1084	3792	10.52	51295		Xilinx Vivado		
Xie et al. [35]	Digit-serial Karatsuba	NA	1420	49.5	70290	233	Stratix II EP2S180F	No	Yes
This work	OBS	1872	17738	9.39	166559		Quartus Prime		
Arish et al. [41]	Karatsuba-Urdhva	972	1018	13.00	25870	24	Virtex-4	Yes	No
This work	OBS	184	360	12.51	6805		Xilinx ISE		
Rashidi et al. [42]	Pipelined Bit-parallel	21195	36812	7.19	3702090	233	Virtex-4 XC4VLX200	No	Yes
This work	OBS	1147	19804	8.29	173683		Xilinx ISE		

In terms of delay of operations, the proposed method is, on average, 25% slower than the CA. On the other side, it outperforms the Karatsuba and overlap-free by 30% and 20% accordingly.

In Section IV-A, the proposed OBS approach is implemented on various hardwares, and comparisons with similar hardware implementations are made.

B. Different FPGA Devices

Delay, area, and ADP obtained from the conventional complexity evaluation methods are not consistent and often change with FPGA device or synthesizer software. Some FPGAs, such as Intel's Cyclone IV, series are made of four-input LUTs, whereas Xilinx's modern FPGAs, such as the 7 and 6 series, have six-input LUTs. It is shown that FPGAs with six-input LUT have advantages over those with four-input LUTs [43].

Furthermore, these FPGAs are fabricated using different technologies. For instance, Spartan 6 series are fabricated with a 45-nm technology node, whereas Spartan-7 is manufactured with the 28-nm channel length transistors. As a result, it is not reasonable to directly compare the delay of a design on these two FPGA series.

Therefore, the proposed method was implemented on various FPGAs to study the effect of the device variations on its efficiency. Table IV shows the number of LUTs and combinational delay for FPGA implementation of GF(2^{233}) multiplier using different algorithms, synthesizer tools, and FPGA devices.

As data shown in this table, even though device utilization and speed change with FPGA and synthesis tool, the proposed method is the most efficient implementation method. Moreover, the conventional method remains the fastest.

Calculating the average number of LUTs and delay for all FPGA types, the proposed method is almost 35% faster than Karatsuba and 29% faster than overlap-free. At the same time, it requires 12% less area than Karatsuba, 15% less area than overlap-free, and 42% less than the CA.

C. Comparison With Relevant Works

In this section, FPGA device utilization and speed for FPGA implementation of the proposed method is compared with other relevant works in this field. For a fair comparison, the proposed method was implemented using the same synthesis tool, FPGA device, and operand size as used in those works.

Device utilizations and speeds reported so far in this work are without considering I/O buffers. However, the proposed method was implemented with I/O buffers enabled to be compared with works that performance parameters reported considering I/O buffers. A similar approach was also applied to reduction modules.

Table V shows FPGA utilization, delay, and ADP for the proposed methods and some of the works published in the literature. Samanta *et al.* [38] proposed a modified Karatsuba for 8-bit operands where product terms are split into alternative formats to shorten the delay of operation. However, the proposed OBS approach outperformed it and has a lower ADP. Zhou *et al.* [39] proposed a 233-bit Karatsuba-based algorithm. It still shows a higher delay and an overall larger ADP than our proposed approach.

Imaña [40] proposed a bit-parallel polynomial multiplier where a new splitting approach was applied to type I irreducible polynomials. Compared to this work, the proposed method is almost twice faster, while its ADP is smaller by a third.

A low area pipelined digit-serial multiplier is presented by Xie *et al.* [35]. This work has a considerably low area and is ideal for applications where minimizing the cost is the priority. Comparatively, the design presented in this work has a larger area but, on the other hand, its total delay is smaller.

A Karatsuba-UrdHa-based bit-parallel multiplier is presented in [41]. The combination delay reported in this work is higher than the proposed design, while its area is almost three times larger. Rashidi [42] proposed a pipelined version of the bit-parallel binary field multipliers. This design has a smaller delay than the hardware presented in this work; however, its ADP is considerably higher. Overall comparisons show that the proposed method outperforms the state-of-the-art and provides an efficient solution for large operand sizes.

V. Conclusion

In this article, a novel finite-field multiplier was proposed. The proposed method was implemented on FPGA for different operand sizes and its performance parameters were compared with other algorithms. Implementation results indicated that the proposed method is, on average, 30% faster than Karatsuba and 20% faster than the OKA. While being faster, it requires 1% less area than that of Karatsuba, 4% less area than that of overlap-free Karatsuba, %43 less than that of the CA. Comparing ADP also indicated that the design is almost 25% more efficient than conventional, 30% more efficient than Karatsuba, and 25% more efficient than OKA. Comparing with state-of-the-art works also indicated that the design has higher speed and lower ADP, which demonstrates the efficiency of the design.

References

[1] R. Abu-Salma, M. A. Sasse, J. Bonneau, A. Danilova, A. Naiakshina, and M. Smith, "Obstacles to the adoption of secure communication tools," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 137–153.

[2] B. Vembu, A. Navale, and S. Sadhasivan, "Creating secure communication channels between processing elements," U.S. Patent 9 589 159, Mar. 7, 2017.

[3] J. Yoo and J. H. Yi, "Code-based authentication scheme for lightweight integrity checking of smart vehicles," *IEEE Access*, vol. 6, pp. 46731–46741, 2018.

[4] K. Shahbazi and S. B. Ko, "Area-efficient nano-AES implementation for Internet-of-Things devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 1, pp. 136–146, Jan. 2021.

[5] P. Aparna and P. V. V. Kishore, "Biometric-based efficient medical image watermarking in E-healthcare application," *IET Image Process.*, vol. 13, no. 3, pp. 421–428, Feb. 2019.

[6] S. Raza, L. Seitz, D. Sitenkov, and G. Selander, "S3K: Scalable security with symmetric keys—DTLS key establishment for the Internet of Things," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 3, pp. 1270–1280, Jul. 2016.

[7] X. Zhang, J. Long, Z. Wang, and H. Cheng, "Lossless and reversible data hiding in encrypted images with public-key cryptography," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 9, pp. 1622–1631, Sep. 2016.

[8] A. Faz-Hernandez, F. Rodriguez-Henriquez, E. Ochoa-Jimenez, and J. Lopez, "A faster software implementation of the supersingular isogeny Diffie-Hellman key exchange protocol," *IEEE Trans. Comput.*, vol. 67, no. 11, pp. 1622–1636, Nov. 2018.

[9] X. Zhou and X. Tang, "Research and implementation of RSA algorithm for encryption and decryption," in *Proc. 6th Int. Forum Strategic Technol.*, vol. 2, Aug. 2011, pp. 1118–1121.

[10] F.-Y. Rao, "On the security of a variant of ElGamal encryption scheme," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 4, pp. 725–728, Jul. 2019.

[11] Z. U. A. Khan and M. Benaissa, "High-speed and low-latency ECC processor implementation over GF(2^m) on FPGA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 1, pp. 165–176, Jan. 2017.

[12] F. Mallouli, A. Hellal, N. S. Saeed, and F. A. Alzahrani, "A survey on cryptography: Comparative study between RSA vs ECC algorithms, and RSA vs El-Gamal algorithms," in *Proc. 6th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/ 5th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Jun. 2019, pp. 173–176.

[13] S. R. Singh, A. K. Khan, and S. R. Singh, "Performance evaluation of RSA and elliptic curve cryptography," in *Proc. 2nd Int. Conf. Contemp. Comput. Informat. (IC3I)*, Dec. 2016, pp. 302–306.

[14] G. Chen, G. Bai, and H. Chen, "A high-performance elliptic curve cryptographic processor for general curves over GF(p) based on a systolic arithmetic unit," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 5, pp. 412–416, May 2007.

[15] H. Marzouqi, M. Al-Qutayri, K. Salah, D. Schinianakis, and T. Stouraitis, "A high-speed FPGA implementation of an RSD-based ECC processor," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 1, pp. 151–164, Jan. 2016.

[16] K. C. C. Loi and S.-B. Ko, "Scalable elliptic curve cryptosystem FPGA processor for NIST prime curves," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2753–2756, Nov. 2015.

[17] P. H. Namin, C. Roma, R. Muscedere, and M. Ahmadi, "Efficient VLSI implementation of a sequential finite field multiplier using reordered normal basis in domino logic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 11, pp. 2542–2552, Nov. 2018.

[18] C.-Y. Lee, C.-S. Yang, B. K. Meher, P. K. Meher, and J.-S. Pan, "Low-complexity digit-serial and scalable SPB/GPB multipliers over large binary extension fields using (b,2)-way Karatsuba decomposition," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 11, pp. 3115–3124, Nov. 2014.

[19] P. Chen, S. N. Basha, M. Mozaffari-Kermani, R. Azarderakhsh, and J. Xie, "FPGA realization of low register systolic all-one-polynomial multipliers over $GF(2^m)$ and their applications in trinomial multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 725–734, Feb. 2017.

[20] M. Esmaeildoust, D. Schinianakis, H. Javashi, T. Stouraitis, and K. Navi, "Efficient rns implementation of elliptic curve point multiplication over GF(p)," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 8, pp. 1545–1549, Aug. 2013.

[21] H. Alrimeih and D. Rakhmatov, "Fast and flexible hardware support for ECC over multiple standard prime fields," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 12, pp. 2661–2674, Dec. 2014.

[22] 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.

[23] P. K. Meher, "Systolic and non-systolic scalable modular designs of finite field multipliers for Reed–Solomon codec," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 6, pp. 747–757, Jun. 2009.

[24] S. H. Namin, H. Wu, and M. Ahmadi, "Low-power design for a digit-serial polynomial basis finite field multiplier using factoring technique," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 441–449, Feb. 2017.

[25] J.-S. Pan, C.-Y. Lee, A. Sghaier, M. Zeghid, and J. Xie, "Novel systolization of subquadratic space complexity multipliers based on Toeplitz matrix–vector product approach," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 7, pp. 1614–1622, Jul. 2019.

[26] J. Xie, C.-Y. Lee, P. K. Meher, and Z.-H. Mao, "Novel bit-parallel and digit-serial systolic finite field multipliers over $GF(2^m)$ based on reordered normal basis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 9, pp. 2119–2130, Sep. 2019.

[27] A. Karatsuba and Y. Ofman, "Multiplication of many-digital numbers by automatic computers," *Doklady Akademii Nauk SSSR*, vol. 145, no. 2, pp. 293–294, 1962.

[28] W. Liu, S. Fan, A. Khalid, C. Rafferty, and M. O'Neill, "Optimized schoolbook polynomial multiplication for compact lattice-based cryptography on FPGA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 10, pp. 2459–2463, Oct. 2019.

[29] L. Li and S. Li, "High-performance pipelined architecture of elliptic curve scalar multiplication over GF(2^m)," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 4, pp. 1223–1232, Apr. 2016.

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

[31] L. Henzen and W. Fichtner, "FPGA parallel-pipelined AES-GCM core for 100G Ethernet applications," in *Proc. ESSCIRC*, Sep. 2010, pp. 202–205.

[32] Y. Li, Y. Zhang, and W. He, "Fast hybrid Karatsuba multiplier for type II pentanomials," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 11, pp. 2459–2463, Nov. 2020.

[33] H. Fan, J.-G. Sun, M. Gu, and K.-Y. Lam, "Overlap-free Karatsuba-ofman polynomial multiplication algorithms," *IET Inf. Secur.*, vol. 4, no. 1, pp. 8–14, Mar. 2010.

[34] C.-Y. Lee and J. Xie, "Low area-delay complexity digit-level parallel-in serial-out multiplier over GF(2^m) based on overlap-free Karatsuba algorithm," in *Proc. IEEE 36th Int. Conf. Comput. Design (ICCD)*, Oct. 2018, pp. 187–194.
[35] J. Xie, P. K. Meher, M. Sun, Y. Li, B. Zeng, and Z.-H. Mao, "Efficient FPGA implementation of low-complexity systolic Karatsuba multiplier over $GF(2^m)$ based on NIST polynomials," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 7, pp. 1815–1825, Jul. 2017.
[36] Y. Li, Y. Zhang, X. Guo, and C. Qi, "N-term Karatsuba algorithm and its application to multiplier designs for special trinomials," *IEEE Access*, vol. 6, pp. 43056–43069, 2018.
[37] A. D. Piccoli, A. Visconti, and O. G. Rizzo, "Polynomial multiplication over binary finite fields: New upper bounds," *J. Cryptograph. Eng.*, vol. 10, pp. 197–210, Apr. 2019.
[38] J. Samanta, R. Sultana, and J. Bhaumik, "FPGA based modified Karatsuba multiplier," in *Proc. Int. Conf. VLSI Signal Process. (ICVSP)*, vol. 10, 2014, p. 12.
[39] G. Zhou, H. Michalik, and L. Hinsenkamp, "Complexity analysis and efficient implementations of bit parallel finite field multipliers based on Karatsuba-Ofman algorithm on FPGAs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 7, pp. 1057–1066, Jul. 2010.
[40] J. L. Imana, "Fast bit-parallel binary multipliers based on type-I pentanomials," *IEEE Trans. Comput.*, vol. 67, no. 6, pp. 898–904, Jun. 2018.
[41] S. Arish and R. K. Sharma, "An efficient floating point multiplier design for high speed applications using Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm," in *Proc. Int. Conf. Signal Process. Commun. (ICSC)*, Mar. 2015, pp. 303–308.
[42] B. Rashidi, "Throughput/area efficient implementation of scalable polynomial basis multiplication," *J. Hardw. Syst. Secur.*, vol. 4, pp. 120–135, Jan. 2020.
[43] A. Percey, "Advantages of the virtex-5 FPGA 6-input LUT architecture," White Paper Virtex-5 FPGAs, Xilinx WP284 (v1. 0), 2007.

Moslem Heidarpur (Member, IEEE) received the B.Sc. degree in electrical engineering and the M.Sc. degree in electronic engineering from the Department of Electrical Engineering, Razi University, Kermanshah, Iran, in 2012 and 2014, respectively, and the Ph.D. degree from the University of Windsor, Windsor, ON, Canada, in 2020.

He is currently a Postdoctoral Research Fellow with the University of Windsor. His research interests include analog and digital electronic circuit design and optimization, neuromorphic engineering, and cybersecurity.

Mitra Mirhassani (Senior Member, IEEE) is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada. Her research interests include hardware security, neuromorphic engineering, and arithmetic optimization.

Prof. Mirhassani is serving as an Associate Editor for IEEE ACCESS journal and a Guest Associate Editor for IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. She is the Advisor to the Women in Cybersecurity (WiCyS) Windsor Chapter. She was recognized as one of the Top Women in Cybersecurity by IT World Canada.