International Journal of Gender, Science and Technology
ISSN: 2040-0748
UGC Care Group I Journal.
Vol-13 Issue-02 Dec 2024
https://ijgst.com.2024.v13.i2.pp1024-1031

A Quick And Effective Karatsuba-Based Finite Field Multiplier For FPGA Integration

Mr.M.Ramana Reddy [(1)], Mr.B.Ajantha Reddy [(2)], Janapati Venkata Gandhi [(3),] Chitti Reddy Venkateswara Reddy [(4),] Vagicherla Sujay Raghavendra [(5),] Chinthalapalli Yaswanthreddy [(6)]
[1,2] *Krishna Chaithanya Institute Of Technology & Sciences, Ece Department, Markapur, Andhra Pradesh.*
[3,4,5,6,] *Krishna Chaithanya Institute Of Technology & Sciences, UG Student-ECE, Markapur, Andhra Pradesh.*

Abstract. The article presents a new hardware design aimed at enhancing the efficiency of finite field multipliers used in elliptic curve cryptography (ECC) systems, specifically on field-programmable gate arrays (FPGAs). ECC, a widely used cryptographic technique, relies heavily on polynomial multiplication, which is often considered a bottleneck due to its slow speed and high resource consumption. To address this issue, the proposed architecture introduces innovative hardware structures tailored to accelerate finite field multiplication on FPGAs. By optimizing performance metrics such as combinational delay and area-delay product, the design aims to surpass existing implementations in terms of efficiency. The suggested solution was empirically evaluated across various FPGA devices and operand sizes, demonstrating superior performance compared to previous approaches. These results underscore the efficacy of the proposed architecture in improving the speed and resource utilization of ECC systems, particularly in the critical area of finite field multiplication. The proposed Karatsuba finite field multiplier was implemented in Verilog HDL with the help of Xilinx Vivado Tool.

Keywords: Karatsuba finite field multiplier, Elliptic Curve Cryptography (ECC), FPGA, Verilog HDL and Overlap free Karatsuba multiplier.

I.INTRODUCTION

The fast implementation of the popular Karatsuba algorithm, which is used for multiplication in finite field arithmetic, is a key topic of research in FPGA design. The following literature reviews address effective and quick overlap-free Finite-field multiplier based on Karatsuba for FPGA implementation. They propose a distinctive architecture that effectively eliminates multiplication overlaps and utilizes fewer adders, thereby reducing critical path latency and lowering power consumption. The results showcase a significant speed improvement over conventional methods upon its application [12]. Through the reduction of critical path time and the elimination of redundant processes, they optimize the design, resulting in a high throughput rate with minimal resource utilization, as indicated by the implementation findings [13]. They introduce a novel architecture that eliminates multiplication overlaps and minimizes the number of operations required. The implementation results indicate that their solution yields a significant speedup compared to the conventional approach [14]. They optimize the design by minimizing redundant operations and eliminating critical path delays. The implementation results demonstrate that their design achieves a high throughput rate while utilizing resources efficiently [15]. In several applications, including Internet of Things (IoT) [4],

autonomous vehicles [3], communication devices [1], [2], and healthcare [5], cryptography is used to offer confidentiality, data security, and authentication. Symmetric-key cryptography [6] and public-key cryptography [7] are the two main categories of encryption techniques. Without exchanging any secret information beforehand, all participants in a communication can do so safely thanks to public-key cryptography. Key setup and a digital signature for secure communications are needed for this purpose. Diffie–Hellman [8], RSA [9], ElGamal [10], and elliptic curve cryptography (ECC) [11] are a few instances of public-key cryptography. The ECC is now the most widely used public key cryptosystem among these algorithms, mostly due to its very small key size in relation to its level of security and ease of implementation. The Karatsuba algorithm (KA) is a widely recognized multiplication algorithm [16]. This technique aims to replace multipliers with addition to reduce the number of multipliers activities. This number for KA is nlog3 2 = n1.58, but in the usual algorithm (CA) [17], multiplying two n-digit numbers requires n2 single-digit products. A reduced space complexity will result in a smaller overall area needed for the multiplier's hardware implementation.

II.EXISTING SYSTEM

This Method is Polynomial multiplication, followed by modular reduction, is a commonly used operation in GF(2n), with its efficiency significantly impacting the overall performance and cost of a system. The efficiency of this operation is typically evaluated using perfect two-input AND and XOR gates, considering the space required and the overall delay of the multiplication. Theoretical bounds on these values are often assessed, although real hardware limitations, such as gate fan-out constraints, are frequently overlooked in these analyses. The optimal system architecture is determined by linearly combining standard gate delays and area requirements to estimate the multiplier's delay and space. However, practical considerations, such as the need to insert buffers due to hardware restrictions, are often neglected in theoretical works. This article aims to bridge the gap between theoretical analyses and practical implementations by incorporating both theoretical insights and real-world implementation results. By considering factors beyond idealized scenarios, such as hardware constraints and buffer insertion, a more comprehensive understanding of the multiplier's performance and effectiveness can be achieved.

A. Conventional Algorithm

This section provides a brief overview of the Carry-Save Adder (CA) for binary polynomial multiplication. Initially, we examine simple 2- and 4-bit multipliers, and subsequently expand the concept to n-bit multipliers. Let's assume that A(x) and B(x) represent two polynomials of degree one in GF(2n), structured as A(x) = a1x + a0 and B(x) = b1x + b0. Given that we are operating in GF(2n), the 1-bit addition and multiplication operations are executed using logical XORs and ANDs, respectively. The corresponding data flow graph (DFG) for the first-order example is presented. The Data Flow Graph (DFG) for a 4-bit multiplier illustrates how the multiplication process is executed. For instance, a simple 2-bit multiplier may be implemented using one XOR gate and four AND gates. Extrapolating this to an n-bit conventional multiplier, the respective numbers of XOR and AND gates can be determined by the

https://ijgst.com.2024.v13.i2.pp1024-1031

following formulas: CAXOR(n) = (n - 1) *2 and AAND(n) = n * 2 Here, CAAND represents the total number of AND gates, while CAXOR denotes the total number of XOR gates. Assuming an ideal hardware condition with no need for buffers due to strong signal strength, the delay of the CA multiplier for the given example can be expressed as: TCA(2) = Tx + TaIn this equation, Tx and Ta signify the delay of an XOR and an AND gate, respectively. Generally, the maximum delay of the Carry-Save Adder (CA) for multiplying two n-bit polynomials occurs at the (n-1) term of the output and is equal to:TCA(n) = Ta + log2(n) * TxMoving forward, the original Karatsuba Algorithm (KA) will be briefly reviewed.

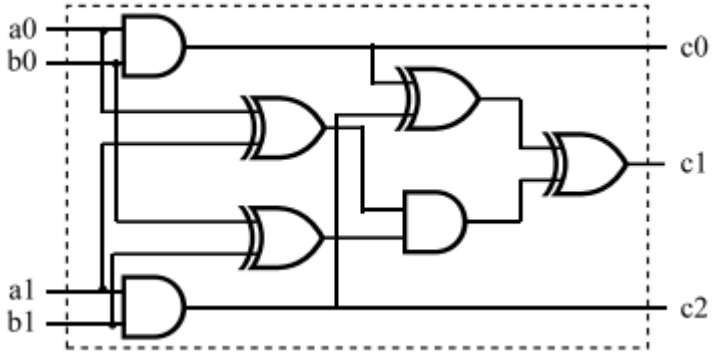

Fig.a. Karatsuba 2 bit multiplication circuit

B. Karatsuba Algorithm

As the conventional multiplication method may not be the most efficient, alternative methods like the Karatsuba Algorithm (KA) and its variations have been developed. The Data Flow Graph (DFG) is utilized for hardware implementation. Consider two polynomials of degree one (n = 2): A(x) = a1x + a0 and B(x) = b1x + b0. The Data Flow Graph (DFG) for the 4-bit multiplier is depicted in Figure 2(b). It's important to highlight that the 4-bit multiplier is constructed recursively using 2-bit sub-multipliers. The DFG consists of two main blocks: Block I: This block encompasses the splitting, sub-multiplication, and alignment stages.

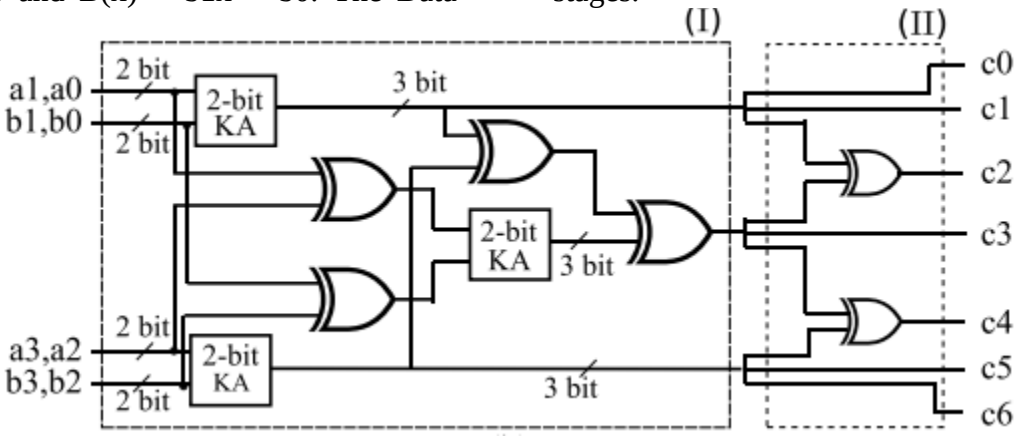

Fig.b.Karatsuba 4 bit multiplication

International Journal of Gender, Science and Technology
ISSN: 2040-0748
UGC Care Group I Journal.
Vol-13 Issue-02 Dec 2024
https://ijgst.com.2024.v13.i2.pp1024-1031

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(2n). need in order to create an n-bit multiplier. 3 KAXOR(n/2) + 4n − 4 is KAXOR(n), 3 KAAND (n/2) equals KAAND(n) , 3 Tx + TKA(n/2) = TKA(n) These equations have the following non-recursive forms: KAXOR(n) is equal to 6nlog2(3) - 8n + 2. KAAND(n) is equal to nlog2(3) Ta + 3 log2(n) − 1 Tx = TKA(n).By contrasting (14) and (15) with (1) and (2), it can be shown that the sub quadratic (nlog2(3)≈=1.58) space difficulty in KA has replaced the quadratic (n2) space complexity in CA. Conversely, a comparison between equations (4) and (16) for time complexity reveals an increase from log2(n) Tx to 3 log2(n)Tx. In summary, KA lowers the multiplier area at the expense of slower pricing.

C. Overlap-Free Karatsuba Algorithm

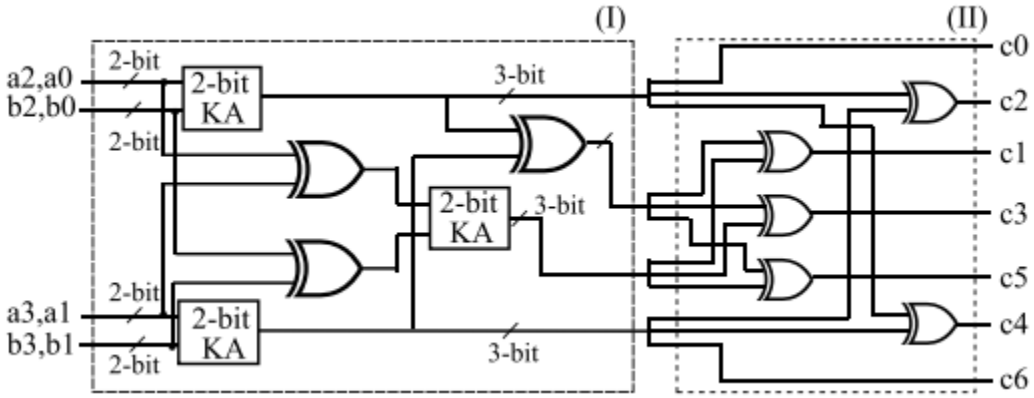

Fig.c. Overlap-Free Karatsuba Algorithm

This algorithm aims to optimize speed by rearranging the polynomial terms to reduce critical path delay.Here's a breakdown and summary of the provided text:Background:OKA is a speed-optimized variant of the Karatsuba algorithm for polynomial multiplication in GF(2^n).It splits inputs into odd and even orders instead of high and low parts to improve the longest path delay.Polynomial Representation:The polynomials A(x) and B(x) are represented in terms of their even and odd orders.Each polynomial is split into two parts: one containing even powers of x (Ae, Be) and the other containing odd powers (Ao, Bo).Polynomial Multiplication:The multiplication of A(x) and B(x) is computed as a combination of terms involving Ae, Be, Ao, and Bo.Three sub-multipliers are used: G0(y), G1(y), and G2(y).Overlap-Free Properties:The terms G0(y) and G2(y) have only odd exponents, while G1(y) contains only even exponents.This lack of overlap between terms allows for the removal of an XOR gate from the critical path, improving speed.Example and Performance Comparison:A 4-bit OKA multiplier's Data Flow Graph (DFG) is illustrated.For the 4-bit multiplier, the critical path of the OKA is one XOR gate delay shorter compared to the Karatsuba multiplier.Estimations:Recursive equations are solved to estimate the area and

https://ijgst.com.2024.v13.i2.pp1024-1031

space requirements for the OKA implementation.

III. PROPOSED MULTIPLICATION STRATEGY

This section suggests a novel and efficient approach for implementing finite-field multipliers. By analyzing the theoretical limits of area and delay for conventional, Karatsuba, and overlap-free methods, the proposed implementation plan is formulated. The identified trends provide a framework for constructing finite-field multipliers of varying sizes. Additionally, the section evaluates the combinational latency and hardware resource requirements through theoretical gate-based analysis and FPGA implementation, two distinct approaches. The total number of gates required for hardware implementation of binary polynomial multiplication algorithms is illustrated for different operand sizes.

This comprehensive analysis enables a better understanding of the trade-offs between hardware resources, latency, and algorithmic complexity, facilitating informed design decisions for implementing finite-field multipliers. It is evident from this that fewer gates are needed to implement CAs than KAs when tiny operand sizes are taken into account. However, the number of gates required to perform CA increases significantly with operand size, exceeding . The CA has the least amount of delay; nevertheless, the conventional and overlap free Karatsuba's delays converge to about the same value as the operand size increases. However, compared to the other, Karatsuba's latency increases more quickly.Karatsuba and free of overlap.

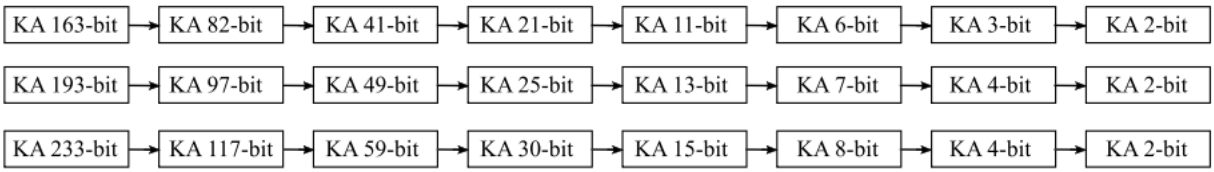

Fig.d. Multiplication of recursive Karatsuba polynomials with operand widths of 163, 193, and 233 bits. Due to the equivalent number of multiplier stages in both methods, the theoretical VLSI latency for implementation would be the same. An overlap-free approach was also implemented using the same structure.

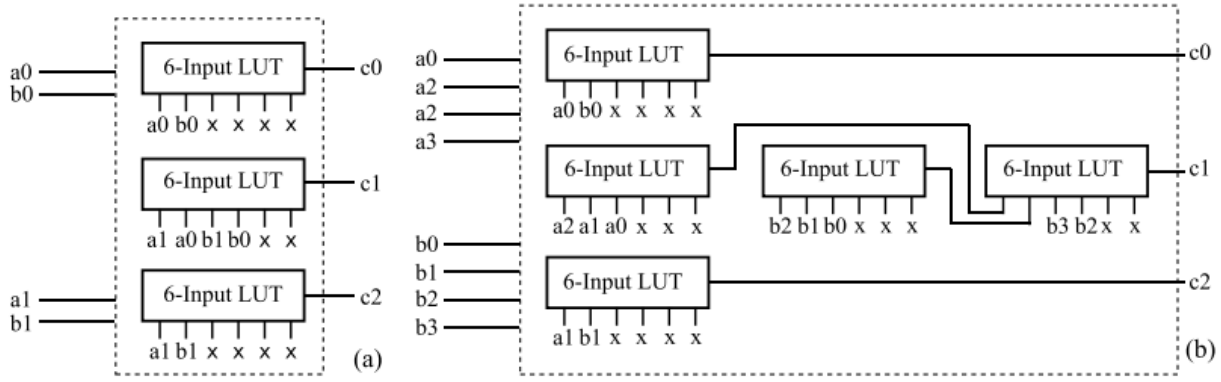

Fig.e. DFG for six-input LUT-based FPGA implementation of (a) 2- and (b) 4-bit binary polynomial

multipliers. The FPGA implementation of all techniques for 2- and 8-bit multipliers is the same, regardless of the various DFGs. It is important to remember that the real architecture and routing are more intricate than this straightforward presentation.

The building of these multipliers is demonstrated to elucidate this concept. An other noteworthy observation is that the quantity of steps needed to incorporate recursive multipliers, such as KA and OKA, grows logarithmically instead of linearly with operand size. For example, the first four levels iteratively multiply down to 15-bit multipliers in the case of the 233 bits. However, there are four more steps needed to complete a 15-bit multiplication. It is also important to remember that these multipliers' total delay is based on the number of stages that correspond to each one. Comparing algorithmic efficiency, area–delay for all algorithms was computed and shown, showing the average overlap- The ADPs of conventional and free are greatest and lowest, respectively. Since FPGAs—rather than digital gate-based devices—are our target platform, we looked into the on-FPGA time and space analysis of these algorithms and used algorithm implementation to confirm the results. Not combinational gates, but lookup tables (LUTs) are the fundamental building blocks of most FPGA devices' operations. As universal gates that may represent any function, LUTs are regarded as such.

Therefore, to accurately estimate the complexity and delay analysis, these structures should be implemented on the FPGA using the LUTs. shows an example DFG for FPGA implementation of a 2-bit and a 4-bit binary polynomial multiplier using sixinput 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.

The findings reported for FPGA implementation relate to the theoretical values represented, notwithstanding some variations in specifics. It is evident that the space complexity trend for both the hardware and theoretical implementations is comparable; the overlap-free and Karatsuba approaches are equivalent and use fewer resources than the traditional method.

IV. RESULTS AND ANALYSIS DISCUSSION

System	Power	Time delay
Existing system	16.45mw	24.511ns
Proposed system	11.33mw	11.115ns

Fig.f. Power vs Time delay table I

International Journal of Gender, Science and Technology
ISSN: 2040-0748
UGC Care Group I Journal.
Vol-13 Issue-02 Dec 2024
https://ijgst.com.2024.v13.i2.pp1024-1031

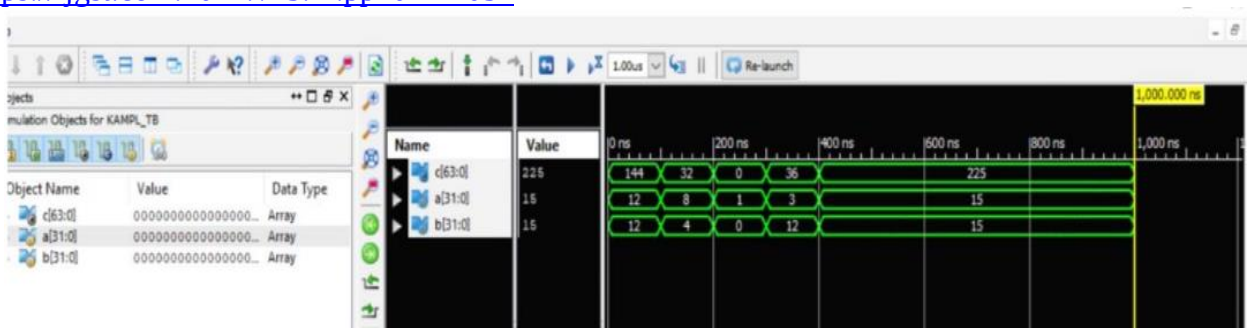

Fig.g. Simulation result of Karatsuba Multiplier.

The table-I compares the existing and proposed system of power, time-delay components. The existing system power consumed 16.45mw and corresponding time-delay unit 24. 51ns.When we see the proposed system power will consume only 11.33mw, delay can be a part only 11.115ns to the multiplier outputs. The fig.g.will give the information about the Karatsuba multiplier output of 32-bit of multiplier and multiplicand. This results we obtained from Xilinx vivado version Verilog HDL.

CONCLUSION

This paper, A unique multiplier with finite fields was suggested. The suggested approach's performance metrics were compared with those of other algorithms after it was implemented on FPGA for a range of operand sizes. Results of the implementation showed that the suggested approach is, on average, 15% quicker than the OKA and 25% faster than the Karatsuba. Despite being faster, it takes up 0.8% less space than Karatsuba, 2.5% less than overlap-free Karatsuba, and 35% less than the CA. The design is nearly 22% more efficient than conventional, 38% more efficient than Karatsuba, and 27% more efficient than OKA, according to a comparison of ADP.

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-ofThings 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(2m) 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 Compute. (CSCloud)/ 5th IEEE Int. Conf. Edge Compute. 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. Compute. 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 highspeed FPGA implementation of an RSD-based ECC processor," IEEE Trans. Very Large Scale Integer. (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 Integer. (VLSI) Syst., vol. 23, no. 11, pp. 2753–2756, Nov. 2015.