Article

Elliptic-Curve Crypto Processor for RFID Applications

Muhammad Rashid ¹ , Sajjad Shaukat Jamal ² , Sikandar Zulqarnain Khan ³ , Adel R. Alharbi ⁴ ,
Amer Aljaedi ⁴ and Malik Imran ^{5,*}

¹ Department of Computer Engineering, Umm Al-Qura University, Makkah 24382, Saudi Arabia; mfelahi@uqu.edu.sa

² Department of Mathematics, College of Science, King Khalid University, Abha 61413, Saudi Arabia; shussain@kku.edu.sa

³ Department of Aeronautical Engineering, Estonian Aviation Academy, 61707 Tartu, Estonia; sikandar.khan@eava.ee

⁴ College of Computing and Information Technology, University of Tabuk, Tabuk 71491, Saudi Arabia; aalharbi@ut.edu.sa (A.R.A.); aaljaedi@ut.edu.sa (A.A.)

⁵ Department of Computer Systems, Tallinn University of Technology, 12616 Tallinn, Estonia

* Correspondence: malik.imran@taltech.ee; Tel.: +37-253-676-608

Abstract: This work presents an Elliptic-curve Point Multiplication (ECP) architecture with a focus on low latency and low area for radio-frequency-identification (RFID) applications over $GF(2^{163})$. To achieve low latency, we have reduced the clock cycles by using: (i) three-shift buffers in the datapath to load Elliptic-curve parameters as well as an initial point, (ii) the identical size of input/output interfaces in all building blocks of the architecture. The low area is preserved by using the same hardware resources of squaring and multiplication for inversion computation. Finally, an efficient controller is used to control the inferred logic. The proposed ECP architecture is modeled in Verilog and the synthesis results are given on three different 7-series FPGA (Field Programmable Gate Array) devices, i.e., Kintex-7, Artix-7, and Virtex-7. The performance of the architecture is provided with the integration of a schoolbook multiplier (implemented with two different logic styles, i.e., combinational and sequential). On Kintex-7, the combinational implementation style of a schoolbook multiplier results in power-optimized, i.e., 161 μ W, values with an expense of (i) hardware resources, i.e., 3561 look-up-tables and 1527 flip-flops, (ii) clock frequency, i.e., 227 MHz, and (iii) latency, i.e., 11.57 μ s. On the same Kintex-7 device, the sequential implementation style of a schoolbook multiplier provides, (i) 2.88 μ s latency, (ii) 1786 look-up-tables and 1855 flip-flops, (iii) 647 μ W power, and (iv) 909 MHz clock frequency. Therefore, the reported area, latency and power results make the proposed ECP architecture well-suited for RFID applications.

Keywords: elliptic-curve cryptography; crypto processor; RFID; hardware accelerator; FPGA

Citation: Rashid, M.; Jamal, S.S.; Khan, S.Z.; Alharbi, A.R.; Aljaedi, A.; Imran, M. Elliptic-Curve Crypto Processor for RFID Applications. *Appl. Sci.* **2021**, *11*, 7079. <https://doi.org/10.3390/app11157079>

Academic Editor: Arcangelo Castiglione

Received: 6 July 2021

Accepted: 26 July 2021

Published: 31 July 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Radio Frequency Identification (RFID) technology employs wireless communication for the tracking/identification/matching of an object. An RFID system includes tags and a reader such that the reader sends radio waves which are used by the tags for communication of the required information. Subsequently, the reader receives signals back from the tags [1]. Generally, there are two main types of RFID tags: active tags that are battery-powered and passive tags that drag the power from external sources, i.e., electromagnetic energy is transmitted to them from an RFID reader [2]. The RFID technology is extensively used in many applications, such as inventory control systems [3], wireless sensor networks [4], vehicle indoor localization [5], logistics [6], monitoring [7], warehousing [8], healthcare [9] and so on. Despite the frequent use of the RFID technology, the security issues are becoming more and more important [10]. In addition to the security issues, the RFID applications are resource-constrained [11,12].

From the security perspective, there exist several protocols/algorithms. However, two types of security algorithms are commonly involved and these include symmetric and asymmetric algorithms [13]. The symmetric algorithm contains a single key to perform encryption/decryption while the asymmetric algorithm requires a different pair of keys (public and private) for the said purpose. However, as the number of RFID tags increases, the potential risk in storing several symmetric keys also increases. Furthermore, it also adds up to the hardware cost and power consumption [11] of the system. Therefore, symmetric algorithms such as the Advanced Encryption Standard (AES) are not useful for RFID-related applications. In other words, asymmetric algorithms are more beneficial to achieve security and system requirements [14].

While the asymmetric algorithms provide better security for RFID applications, they infer higher computational overhead [11]. For example, the passive RFID tags take the energy they require from the radio signals where the power supply is limited [2]. Subsequently, these tags cannot utilize the energy-demanding asymmetric algorithms such as the Rivest–Shamir–Adleman (RSA) algorithm. Consequently, the Elliptic-curve Cryptography (ECC) algorithms are employed in many RFID applications due to their advantages over the traditional asymmetric algorithms [15,16]. For example, ECC offers smaller key sizes as compared to the RSA for the same security, i.e., the security of 163-bit ECC is considered equivalent to 1024-bits RSA [11]. That is why ECC is a considered a better choice for implementation of RFID tag chips.

Elliptic-curve cryptography involves four layers of operation [17]. The fourth layer is responsible to perform encryption and decryption. The most crucial operation is point multiplication (PM) which is computed in layer three. The point addition (PA) and point doubling (PD) as required for PM computation are performed in layer two. Layer one contains finite-field (FF) arithmetic operators (i.e., addition, multiplication, square and inversion). In addition, two types of coordinate systems, i.e., affine and projective, are involved. The latter is more frequently used to optimize the time required to perform one PM computation [16,18]. Additionally, two field representations, i.e., the prime ($GF(\mathbb{P})$) and binary ($GF(2^m)$), are available. The prior is commonly utilized for software-based implementations while the latter is frequently employed for hardware accelerators [11,18].

1.1. Hardware Accelerators for RFID Applications

The existing ECC-based state-of-the-art hardware accelerators for RFID applications are described in [14,19–22]. In [14], an ECC-based crypto processor for RFID tags over $GF(2^{163})$ is presented. The processor architecture is capable of performing the PM and modular arithmetic operations that include additions and multiplications. These operations are required for the ECC based crypto protocols. On 0.13 μm complementary metal–oxide–semiconductor (CMOS) technology, the achieved clock frequency is 1.13 MHz. The total number of required clock cycles to perform one PM operation is 275,816. Similarly, the time to compute one PM operation is 244.084 ms. The architecture utilizes 12,506 logic gates and consumes 36.63 μW power.

Another ECC-based crypto processor architecture for RFID tags over $GF(2^{163})$ is presented in [19]. On 0.35 μm CMOS technology, the achieved clock frequency is 13.56 MHz. The total number of required clock cycles to perform one PM operation is 430,654. Moreover, the time for one PM computation is 31.8 ms with a utilization of 15,094 logic gates. Similar to [14,19], another ECC-based crypto processor architecture for RFID tags over $GF(2^{163})$ is presented in [20]. On a 0.18 μm technology, the reported clock frequency is 0.847 MHz. The total number of reported clock cycles for the computation of one PM operation is 296,964. Moreover, the time for one PM computation is 350.6 ms with a utilization of 13,200 logic gates.

A binary Edwards-curve (a special class/model of ECC) based crypto processor for extremely constrained applications is described in [21]. For area optimization, the size of the embedded register file is reduced. The power is improved by setting/enabling clock-gating during the synthesis process over the 0.13 μm technology. With several area and

power improvement techniques, the reported clock frequency is 0.400 MHz. Furthermore, the total number of required clock cycles for the computation of one PM operation is 219,148. Moreover, the time required to perform one PM computation is 547.87 ms with a utilization of 11,720 logic gates.

Similarly in [22], a crypto processor is presented to implement an Elliptic-curve Digital Signature Algorithm (ECDSA) over $GF(\mathbb{P}_{192})$ for application in RFID tags. The design objectives are to achieve low-energy, small chip area, robustness against cryptographic attacks and flexibility. On a 130 nm CMOS technology, the processor achieves 200 MHz frequency with a chip size 0.15 mm². For one signature generation, it takes 2500 μ s.

In addition to RFID specific architectures, there are several other PM hardware accelerators, described in [18,23–26]. A 2-stage architecture with pipelining to reduce clock cycles and optimize clock frequency is described in [18,24]. A low-area ECC accelerator architecture using a digit-serial multiplication method is described in [23]. Similar to [21], a low-cost and fast hardware implementations of the PM on Binary Edwards-curves is presented in [25]. Moreover, a $\frac{\text{throughput}}{\text{area}}$ architecture to optimize both throughput and area at the same time is provided in [26]. This is achieved by using a digit-serial multiplier in the datapath of the accelerator architecture.

1.2. Limitations in Existing Hardware Accelerators of RFID Applications

Based on the discussions presented in Section 1.1, it can be observed that the accelerator architectures, published in [14,18–26], are specifically designed to reduce hardware resources (area) and power parameters with the expense of higher clock cycles. Therefore, the increase in clock cycles ultimately reduces the performance of the entire architecture [14,19–22]. The reason for the utilization of extensive clock cycles is the use of different architectural styles, i.e., 8/16/32-bit (including datapath logic, memories, FFs), for the computation of PM operation. With low-area/power parameters, performance (latency) is also an essential parameter to be considered during the implementation of PM operation in ECC. Subsequently, the ambition of this work is to provide a low-latency (means high-performance) and low-area hardware accelerator architecture for RFID applications.

1.3. Our Contributions

To address the current challenge, we have provided a PM architecture—specifically for RFID applications over $GF(2^{163})$ —with a focus on low-latency (high-performance) and low-area parameters. Moreover, we have only used an 8-bit interface to load ECC parameters as well as the coordinates of an initial point externally. The building blocks of our architecture (shift buffers/registers, arithmetic unit and register array) are m -bit length, where m is the key size, i.e., 163. The additional contributions of this work to achieve low-latency and area reduction are given as follows:

- To achieve low-latency, we have reduced the clock cycles with the use of:
 - three shift buffers/registers to keep ECC parameters and a constant during the PM computation;
 - same size of the input/output interfaces in all blocks/units of the proposed architecture.
- Towards the area reduction, we have used (for the first time) the Dimitrov–Järvinen inversion algorithm with shared hardware resources of employed squarer and multiplier blocks for RFID applications.
- Finally, an efficient controller based on a finite-state-machine (FSM) that is used to control the inferred logic inside the proposed architecture.

1.4. Numerical Flavor of the Results

The proposed accelerator architecture over $GF(2^{163})$ is modeled in Verilog using the Vivado design tool. To evaluate the performance, we have synthesized our design with the integration of a schoolbook polynomial multiplier (in two different logic styles, i.e., combinational, and sequential) on three different 7-series FPGA devices, i.e., Kintex-7, Artix-7 and Virtex-7. The combinational implementation of a schoolbook multiplier in our proposed accelerator architecture results in power-optimized (161 μ W on both Kintex-7, Artix-7 and 183 μ W on Virtex-7) values with an expense of higher hardware resources (3561 look-up-tables (LUTs) and 1527 flip-flops (FFs) for all the three devices). Moreover, on both Kintex-7 and Virtex-7 devices, the achieved clock frequency is 227 MHz and the time to compute one PM operation (latency) is 11.57 μ s, respectively. These values on Artix-7 are 192 MHz and 13.68 μ s, respectively. On the other hand, the sequential implementation of a schoolbook multiplier in our proposed accelerator architecture results in low-latency (2.88 μ s on both Kintex-7, Virtex-7 and 3.41 μ s on Artix-7) and area-optimized (1786 LUTs & 1855 FFs for all the three devices) values with an expense of power (647 μ W on Kintex-7, Artix-7 while on Virtex-7, the consumed power is 733 μ W). Furthermore, on both Kintex-7 and Virtex-7 devices, the achieved clock frequency is 909 MHz while on Artix-7, 769 MHz is achieved.

It is important to note that the sequential implementation of a schoolbook multiplier allow us to achieve the objective of this paper (a low-latency (high-performance) and low-area architecture for RFID applications). Consequently, the state-of-the-art ASIC architectures, reported in [14,19–21] over $GF(2^{163})$ require 84,722, 11,041.6, 12,1736.1 and 190,208.3 times higher computation time as compared to our high-performance architecture synthesized on a modern Kintex-7 FPGA. Furthermore, the architecture of [22] over $GF(\mathbb{P}_{192})$ requires 868 times higher computation time as compared to our high-performance implementation on Kintex-7 over $GF(2^{163})$. As compared to the FPGA implementations of [18,23–26], our high-performance implementation utilizes lower hardware resources and takes lower computation time for the computation of one PM operation. Our obtained results (for high-performance and power-optimized) make our proposed architecture well-suited for RFID applications.

The remainder of this article is structured as follows: the background knowledge for the architecture of RFID tag chip and PM computation over $GF(2^m)$ is presented in Section 2. Section 3 reports the proposed architecture. Section 4 summarizes the implementation and comparison to state-of-the-art details. Section 5 concludes the paper.

2. Preliminaries

The persistence of this section is to describe the architecture of RFID tag chip, associated with the Elliptic-curve crypto processor (ECP), in Section 2.1. Moreover, the required mathematical background on ECC over $GF(2^m)$ is given in Section 2.2.

2.1. Architecture of RFID Tag Chip with ECC

An RFID tag chip embedded with ECC consists of four parts: (i) analog frond-end, (ii) random number generator, (iii) electrically erasable programmable read-only memory (EEPROM), and (iv) a digital baseband controller, as shown in Figure 1. The description of these parts is given as follows:

- Analog Front-End reads an input analogue signal through an embedded antenna and converts it into a digital format. The converted signal is fed as an input to the baseband controller as shown in Figure 1. It also performs carrier signal modulation (or) demodulation. Finally, it generates the clock and reset signals for the baseband controller.
- Random Numbers Generator (RNG) is required to introduce randomness in each authentication process so that the secret information/data are unpredictable.
- EEPROM is used to keep a pair of keys (i.e., private and public) and ECC parameters including the x and y coordinates of a base point and other required ECC constants.

- Baseband Controller consists of several units, i.e., (i) pre-processing circuit, (ii) system controller, (iii) read access memory (RAM), (iv) memory interface, and (v) Elliptic-curve crypto processor (ECP). Moreover, as shown in Figure 1, a communication bus is used to integrate the aforementioned units. A pre-processing circuit is responsible to extract the relevant information from the incoming frame. If the input frame is available, then a system controller generates the corresponding signal to store the frame data into a RAM unit. Subsequently, the system controller generates a relevant signal to enable ECP for further computations on the stored frame data. Once all the required parameters of ECC and the frame data are present, the controller waits for the feedback signal from the ECP unit.

Figure 1. Structure of an RFID tag chip with Elliptic-curve Cryptography Processor (taken from [11]). We provided our architecture for the ECP block highlighted with red color dotted lines.

2.2. PM Computation over $GF(2^M)$

One PM operation over $GF(2^m)$ is performed by iterating k times the sum of a point P , i.e., $Q = k \cdot P = P + P + \dots + P$, where Q is the final point, k is a scalar multiplier, and P is the initial point. We have employed the following Lopez–Dahab PM algorithm (Algorithm 1).

The coordinates (i.e., x_p and y_p) of initial point P and a scalar multiplier k are inputs to Algorithm 1. Moreover, $k_{n-1}, \dots, k_1, k_0$ determine the scalar multiplier, i.e., key values in terms of 0 s and 1 s. The coordinates (i.e., x_q and y_q) of the final point Q are the output. For PM computation, Algorithm 1 consists of three parts: (i) affine to projective conversion, (ii) PM computation in projective coordinates and (iii) projective to affine conversions. The PM in projective coordinates is operated in a loop fashion. The control variable, i.e., m , in the *for* loop determines the key length.

Algorithm 1: Lopez–Dahab PM Algorithm [27]**Input:** $k = (k_{n-1}, \dots, k_1, k_0)$, $P = (x_p, y_p) \in GF(2^m)$ **Output:** $Q = (x_q, y_q) = k.(P)$

// Affine to projective conversion

1. $X_1 = x_p$
2. $Z_1 = 1$
3. $Z_2 = x_p^2$
4. $X_2 = Z_2^2$
5. $X_2 = X_2 + b$

// Point Multiplication in projective coordinates

for (i from $m-2$ down to 0) **do**

1. $V_1 = X_1 \times Z_2$
2. $V_2 = X_2 \times Z_1$
3. $V_3 = X_1 \times Z_1$
4. $R_3 = Z_1^2$
5. $R_3 = R_3^2$
6. $Z_2 = V_1 + V_2$
7. $Z_2 = Z_2^2$
8. $Z_1 = V_3^2$
9. $V_1 = V_1 \times V_2$
10. $V_2 = x_p \times Z_2$
11. $V_3 = b \times R_3$
12. $R_3 = X_1^2$
13. $R_3 = R_3^2$
14. $X_2 = V_1 + V_2$
15. $X_1 = V_3 + R_3$

if $i = 0$ and $k_i = 1$ **then**

1. **swap** (X_1, X_2)
2. **swap** (Z_1, Z_2)

end if**end for**

// Projective to Affine conversions

1. $V_1 = Inv(Z_1)$
2. $V_2 = Inv(Z_2)$
3. $V_3 = Inv(x_p)$
4. $R_1 = X_1 \times V_1$
5. $V_2 = X_2 \times V_2$
6. $R_3 = (x_p + 0)^2$
7. $R_3 = R_3 + y_p$
8. $V_1 = x_p + R_1$
9. $V_2 = x_p + V_2$
10. $V_1 = V_1 \times V_3$
11. $V_2 = V_2 \times V_1$
12. $V_2 = V_2 + R_3$
13. $V_2 = V_1 \times V_2$
14. $R_2 = V_2 + y_p$

Return: $(x_q, y_q) = (R_1, R_2)$

3. Proposed ECP Hardware Accelerator Architecture

The proposed architecture contains four blocks, i.e., (i) a register array, (ii) shift buffers/registers, (iii) an arithmetic unit, and (iv) a dedicated command controller, as shown in Figure 2. Furthermore, it contains *clk*, *rst* and *start* (1-bit) signals. A 1-bit *done* signal is generated once the crypto processor performs all the required computations. Similarly, it takes 8-bit input data. These input data are either the coordinates of an initial point *P* or related ECC parameters required for the computation. The *din* signal is used externally by the user while the result is in the form of an 8-bit output (*dout*). It is important to note that the size of the architecture is *m*-bit, where *m* is the key length. The architecture is modeled using Verilog HDL in Vivado IDE.

Figure 2. Proposed Hardware Accelerator Architecture of ECP.

3.1. Register Array

An array containing a register file is incorporated into the proposed accelerator architecture, as shown in Figure 2. It constitutes a total of ten registers, i.e., X_1 , X_2 , Z_1 , Z_2 , V_1 , V_2 , V_3 , R_1 , R_2 , and R_3 . These registers are required to contain intermediate and the final results when implementing Algorithm 1. Moreover, two 10×1 multiplexers (not shown in Figure 2) are used to read operands. Similarly, a 1×10 demultiplexer is incorporated (not shown in Figure 2) to update the values in a particular register. Each particular register from the employed register array is connected as an input to the multiplexers while the output is connected to *rdata1* and *rdata2* signals. The input to the demultiplexer is from the arithmetic unit (*wdata*) while the outputs are connected to each particular register in the employed register array. For each read/write operation, one clock cycle is required. A single red color line in Figure 2, from the command controller to register array, indicates the corresponding control signals (total three, one for the write and the remaining for the read) for the read/write operands.

3.2. Shift Buffers/Registers

The purpose of this block is to keep/store the coordinates of point *P* and a constant parameter (i.e., *b*) of ECC. As shown in Figure 2, the proposed architecture constitutes three *m*-bit shift registers, i.e., *xp*, *yp*, and *b*. Moreover, it contains one 3×1 multiplexer (not given in Figure 2) to select an appropriate ECC parameter using *rdata* for the arithmetic unit. The *wdata* with the corresponding control signal (shown with red color in Figure 2) is input to shift buffers/registers block from the command controller to write on a particular register with an 8-bit shift. Whenever the ECP is activated by setting the *start* signal, 8-bits

from LSB (least significant bit) to MSB (most significant bit) of either x or y coordinates of the initial point, $P = (x_p, y_p)$ are loaded into the corresponding x_p or y_p shift register. With the same m -bit operand length and shift register size, $\frac{m}{8}$ clock cycles are required. Therefore, our architecture deals with 163-bit; however, 21 additional clock cycles are required to load a 163 bit operand into a shift register. At the beginning (when start becomes 1), we have loaded (serially) the required ECC parameters and coordinates of point P into a corresponding shift register with an expense of 63 (3×21) clock cycles. It allows us to use the coordinates of point P and ECC parameters directly from the shift registers, instead of re-loading externally, during the implementation of Algorithm 1.

3.3. Arithmetic Unit

The arithmetic unit of the proposed ECP accelerator architecture consists of an (i) adder, (ii) multiplier, (iii) squarer, (iv) controller (AU-controller as shown in Figure 2) and a routing multiplexer. The shaded portion with a gray color in Figure 2 determines the polynomial inversion with a combined use of squarer and multiplier blocks. The description of these arithmetic operators (adder, multiplier, squarer, reduction and inversion) and additional arithmetic unit blocks (Au-controller, routing multiplexer) is given as follows:

- **Adder (ADD) and Squarer (SQR):** The adder and squarer blocks require only one clock cycle for the computation. Therefore, the adder inputs two m -bit operands and gives one m -bit output after performing bitwise the Exclusive (OR) operation. On the other hand, the squarer inputs an m -bit operand and results a $2 \times m - 1$ -bit output. In $GF(2^m)$, a squarer is simply implemented by inserting 0's after two successive data bits.
- **Multiplier (MUL):** The performance of the entire crypto architecture is dependent on the performance of the utilized multiplier. Therefore, there are several approaches to perform polynomial multiplication, i.e., bit-serial, bit-parallel, digit-serial, and a digit-parallel. A comprehensive comparison over these multiplication approaches is presented in [28]. Comparatively, the bit-serial multipliers are more useful for low-area and low-power applications while for high-speed applications, a bit-parallel and digit-parallel approaches are more convenient. The digit serial multipliers are more beneficial where high-speed and low-area parameters are required to consider at the same time. Moreover, in bit-serial multipliers, the low-area and low-power values can be achieved with an additional cost on clock cycles. For example, for multiplication over two m -bit operands length, m clock cycles are required. On the other hand, one clock cycle is required for bit-parallel and digit-parallel multiplier approaches with an overhead over area and power parameters. There is always a design space between, area, power and speed/performance. The digit-serial multipliers take $\frac{m}{n}$ clock cycles, where m is the operand length and n determines the size of the digit. That is why the goal of this work is to provide a low-area and low-power architecture for extremely constrained RFID applications. Subsequently, a traditional schoolbook multiplication method (a type of bit-serial multipliers) is incorporated in this work to achieve low-area and low-power values. A schoolbook multiplication method with shift and add operations take two m -bit polynomials as input and results in $2 \times m - 1$ -bit polynomial as an output. It takes m clock cycles to perform one polynomial multiplication.
- **Polynomial reduction:** After each polynomial squaring and multiplication, a reduction is essential to transform $2 \times m - 1$ -bit polynomials into an m -bit. Therefore, we have performed a reduction using a sequence of routines given in Algorithm 2. For more description on these reduction routines, we refer the reader to [27]. In our proposed ECP architecture, the RTL (Register Transfer Level) development of Algorithm 2 in Verilog (HDL) is implemented using a combinational logic. Therefore, it takes one clock cycle for the polynomial reduction after multiplication. Similarly, a combinational logic is inferred for squarer block. However, squaring including reduction operation takes one clock cycle for the computation.

- **Polynomial inversion:** For polynomial inversion computation, we have employed the Dimitrov–Järvinen (DJ) algorithm [29]. It is formulated with an improvement to the most frequently utilized Itoh–Tsujii algorithm. The computational complexity of both IT and DJ inversion algorithms is the same. For example, these inversion algorithms require 9 multiplications and 162 squarings over $GF(2^{163})$. The key difference is the use of different register variables for the execution of the routines involved in these inversion algorithms (IT and DJ). Therefore, the IT inversion algorithm takes three 163-bit registers while the DJ algorithm needs only two. Based on this observation, our ECP architecture has implemented a DJ inversion algorithm. The sequence of routines in our work over $GF(2^{163})$ is given in Algorithm 5 of [11]. Moreover, we have used the same hardware resources of the squarer and multiplier blocks to implement the DJ inversion algorithm.
- **AU-controller and routing multiplexer:** As described earlier, the adder, squarer and reduction take one clock cycle for the implementation while the multiplier needs m clock cycles for two m -bit operands. Therefore, an AU-controller is required to make a SYNC of combinational (adder, squarer, reduction) and sequential (multiplier) logic inferred in the RTL development of our ECP architecture. It takes three operands (i.e., $rdata$, $rdata1$, and $rdata2$) as an input and results in one operand as an output ($wdata$). The $rdata$ is an output from shift buffers block and it contains a 163-bit value. This value is selected with a routing multiplexer (not shown in Figure 2) either from the ECC parameters (x_p, y_p) or a curve constant (b). The $rdata1$ and $rdata2$ are read operands to the arithmetic unit from the register array. Therefore, based on the control signals from command controller, the AU-controller is responsible to select the appropriate operands for the execution of adder, squarer and multiplier blocks. The output of each arithmetic operator is connected to a routing multiplexer for the written-back data ($wdata$) on the register array.

Algorithm 2: Reduction over $GF(2^{163})$ [27]

Input: Polynomial, $W(x)$ with $2 \times m - 1$ -bit length

Output: Polynomial, $X(x)$ with m -bit length

$Y = W[i] \oplus W[i + 163] \oplus W[i + 319]$

$Z = W[i] \oplus W[i + 157] \oplus W[i + 160]$

1. $for(0 \leq i \leq 1)$ $X[i] = Y \oplus W[i + 320] \oplus W[i + 323]$
 2. $for(i = 2)$ $X[i] = Y \oplus W[i + 320]$
 3. $for(3 \leq i \leq 5)$ $X[i] = Y \oplus W[i + 160] \oplus W[i + 316] \oplus W[i + 317]$
 4. $for(i = 6)$ $X[i] = Z \oplus W[i + 163] \oplus W[i + 313] \oplus W[i + 314] \oplus W[i + 316]$
 5. $for(7 \leq i \leq 10)$ $X[i] = Z \oplus W[i + 156] \oplus W[i + 163] \oplus W[i + 312] \oplus W[i + 314]$
 6. $for(11 \leq i \leq 12)$ $X[i] = Z \oplus W[i + 156] \oplus W[i + 163] \oplus W[i + 312]$
 7. $for(13 \leq i \leq 161)$ $X[i] = Z \oplus W[i + 156] \oplus W[i + 163]$
 8. $for(i = 162)$ $X[i] = Z \oplus W[i + 156]$
-

3.4. Dedicated Command Controller and Clock Cycles Calculation

We have described the FSM states involved in the command controller in Section 3.4.1. The clock cycles calculation is given in Section 3.4.2.

3.4.1. Number of States in the Command Controller

A dedicated command controller (i.e., FSM controller presented in Figure 3) generates control signals to execute, (i) affine-to-projective conversion, (ii) PM in projective coordinates and (iii) projective-to-affine conversions of Algorithm 1 and includes 60 states, as shown in Figure 3. The description of these states with respect to different parts of the Algorithm 1 follows:

Figure 3. FSM Controller of the Proposed Hardware Accelerator Architecture of ECP.

- **Affine-to-projective conversions:** State 0 is an idle state while state 1 to state 5 executes affine-to-projective conversions.
- **PM in projective coordinates:** State 6 to State 20 generate control signals for the sequence of routines (from 1 to 15 in PM part), presented in Algorithm 1. State 21 is a conditional state and is responsible to check *if* ($i = 0$ and $k_i = 1$) in Algorithm 1, where i counts the number of points on the ECC curve. Once the *if* statement becomes true, the next state is state 22, otherwise the next state is state 6. These states continue to repeat until the value of i becomes 0 (initially, the value of i is $m - 2$, see for statements in Algorithm 1). States 22 to 27 are for swapping purpose. Therefore, we have used 3 states (22 to 24) to execute the $\text{swap}(X_1, X_2)$ statement. The operations involved in these 3 states (22 to 24) are, (i) $R_1 = X_1$, (ii) $X_1 = X_2$, and (iii) $X_2 = R_1$. Similarly, in states 25 to 27, we have performed the $\text{swap}(Z_1, Z_2)$ statement. The operations involved in these 3 states (25 to 27) are (i) $R_1 = Z_1$, (ii) $Z_1 = Z_2$, and (iii) $Z_2 = R_1$.
- **Projective-to-affine conversions:** As shown in Algorithm 1, the projective-to-affine conversions consist of 14 sequences of routines. Three are for inversion, 5 are for multiplications, 6 are for additions and there is only one instruction for squaring. To compute each inversion operation, states 28 to 47 are responsible to generate control signals. Finally, the states 48 to 59 are required to implement the remaining sequence of routines (multiplications, addition and squaring) in the projective-to-affine conversion part of Algorithm 1.

3.4.2. Clock Cycle Calculation

The affine-to-projective conversion is carried out by simply transferring $(x_p, 1)$ to (X_1, Z_1) in two clock cycles. The X_2 and Z_2 contain x_p^2 and $x_p^4 + b$, respectively. It is computed by using three clock cycles (see sequence of routines from 3 to 5 in the affine-to-projective conversion part of Algorithm 1). The point multiplication in projective coordinates contains 15 sequences of routines (see Algorithm 1). Out of these 15 routines, 6 are for multiplications, 6 are for squaring and the remaining 3 routines are for additions. Therefore, nine clock cycles are required for the computation of 9 squaring and addition routines. For multiplication, using a traditional schoolbook method, a total of $6 \times m$ clock cycles are needed. As described in the previous section (Section 3.4.1), 6 states are required for the implementation of swap statements. Therefore, six clock cycles are required for this task. For each inversion computation, $m - 1$ squares followed with

9 multiplications are needed [29] over $GF(2^{163})$. Therefore, $m + (9 \times m)$ clock cycles are needed for each inversion, where m determines the key length. Finally, twelve clock cycles are required to perform the remaining sequence of routines in the projective-to-affine conversion. Consequently, it requires a total of 2627 clock cycles for one PM computation. Out of 2627 cycles, 5 cycles are for affine-to-projective conversion, 993 clock cycles are for PM computation in a projective coordinate and 1629 clock cycles are for projective-to-affine conversions. These clock cycles can be computed using Equation (1).

$$\text{Total clock cycles} = \underbrace{5}_{\text{Affine_to_Proj_Conv}} + \underbrace{15 + (6 \times m)}_{\text{PM_in_Proj_Coord}} + \underbrace{3 \times (\text{inversion}) + 12}_{\text{Proj_to_Affine_Conv}} \quad (1)$$

4. Implementation Results and Comparisons

This section includes two subsections where the implementation results are given in Section 4.1 and a comparison with the state-of-the-art is given in Section 4.2.

4.1. Results

We have modeled our architecture in Verilog over $GF(2^{163})$ using the Vivado design tool. The performance of the proposed ECP architecture is evaluated with the integration of a schoolbook polynomial multiplier in two different logic styles (i.e., sequential and combinational). Therefore, the implementation results in the state-of-the-art 7-series FPGA devices are summarized in Table 1. For Kintex-7, Artix-7 and Virtex-7 FPGA boards, the chosen devices for logic synthesis are XC7K325TFFG900-2, XC7A200TFFG676-2 and XC7VX485TFFG1761-2, respectively. The first column in Table 1 shows the implementation device. The provided clock period and the corresponding clock frequency (in MHz) are given in column two and column three, respectively. The time required to perform one PM computation, i.e., latency (in μs), is presented in column four. The area information in terms of look-up-tables (LUTs) and flip-flops (FFs) is shown in column five and column six, respectively. Finally, the last column (column seven) provides the utilized power (in μW). The latency of the architecture is computed by using Equation (2).

$$\text{Latency (in } \mu\text{s)} = \frac{\text{Clock cycles}}{\text{Freq (MHz)}} \quad (2)$$

Table 1. Implementation results of the proposed ECP accelerator architecture on modern FPGA devices.

Device	Clk. Period (in ns)	Freq. (in MHz)	Latency (in μ s)	LUTs	FFs	Power (in μ W)
Utilization of schoolbook multiplier as sequential logic (high-performance results)						
Kintex-7	1.1	909	2.88	1786	1855	647
Artix-7	1.3	769	3.41			647
Virtex-7	1.1	909	2.88			733
Use of schoolbook multiplier as combinational logic (power-optimized results)						
Kintex-7	4.4	227	11.57	3561	1527	161
Artix-7	5.2	192	13.68			161
Virtex-7	4.4	227	11.57			183

4.1.1. Use of a Polynomial Multiplier as Sequential Logic

The use of a schoolbook multiplier as a sequential logic results in a shorter critical path with an increase in both area and clock frequency. Therefore, Table 1 shows that the proposed accelerator architecture utilizes 1786 LUTs and 1855 FFs on modern 7-series FPGA devices (Kintex-7, Artix-7 and Virtex-7). With the utilization of same hardware resources (in terms of LUTs and FFs) for several 7-series FPGA devices, the achieved clock frequency on both Kintex-7 and Virtex-7 devices is 909 MHz which is comparatively 1.18 times higher as compared to the frequency achieved on Artix-7 (769 MHz). On the other hand, the power achieved on both Kintex-7 and Artix-7 devices is 0.647 mW which is comparatively 1.13 times lower than the power achieved on Virtex-7 FPGA (0.733 mW).

Therefore, there is a trade-off among several 7-series devices in terms of frequency and power results. The design of the fabric for the Artix-7 is customized for low-cost while the Kintex-7 and Virtex-7 are tuned for high-performance [30].

4.1.2. Integration of a Polynomial Multiplier as Combinational Logic

Table 1 reveals that the use of a schoolbook multiplier as a combinational logic results in a longer critical path which ultimately shows the decrease in the clock frequency (227 MHz on both Kintex-7 and Virtex-7 devices whereas a 192 MHz on Artix-7). Moreover, with the same clock cycles utilization, it takes two times more hardware resources in terms of FPGA LUTs (3561) as compared to the sequential multiplier circuit (where this value is 1786). Furthermore, on Kintex-7, Virtex-7, and Artix-7 FPGA devices, it requires 4 times more computational time (latency) for the execution of instructions shown in Algorithm 1. Despite all other parameters (i.e., hardware resources, clock frequency, and latency), the use of a combinational multiplier circuit in our proposed ECP architecture results in 4-fold decrease in power consumption (0.161 mW on both Kintex-7 and Artix-7 devices while 0.183 mW on a Virtex-7) as compared to the sequential multiplier circuit.

In summary, the sequential logic results in a low critical path with an increase in the clock frequency. On the other hand, the combinational logic infers the longer critical path with a decrease in the operational frequency. Apart from this, the sequential logic for the polynomial multiplication results in higher power consumption as in the employed schoolbook multiplier where we have incorporated two m -bit registers. The first register is employed to perform a one-bit shift operation in one clock cycle while another register is utilized to accumulate a shifted result during the polynomial multiplication computation. In the combinational style, we have applied the dedicated circuit logic for implementation. On FPGA devices, with the expense of latency (computational time), more power can be reduced by utilizing a single shared buffer rather than three (as we used in this work—see Section 3.2). With use of one shared buffer, further power can be optimized through clock gating when running a synthesis for ASIC commercial nodes (as used in [21]).

4.2. Comparison with State-of-the-Art

Before describing the comparison to the state-of-the-art, it is essential to note that we have provided our implementation results using a schoolbook polynomial multiplier (implemented with two different, i.e., sequential and combinational, logic styles). The sequential implementation of a schoolbook multiplier in our proposed ECP architecture allows us to achieve the objective of this paper (a low-latency (high-performance) and low-area architecture for RFID applications). Therefore, the performance comparison to the existing state-of-the-art is provided with our high-performance implementation results.

The comparison with the state-of-the-art is shown in Table 2. The first column in Table 2 provides the reference solution (Ref. #). The targeted key length, i.e., m and the implementation devices are shown in column two and three, respectively. Column four and five present the clock frequency (Freq. in MHz) and clock cycles (CCs), respectively. The time required to perform one PM computation, i.e., latency (in ms), is presented in column six. The area information for FPGAs (in terms of LUTs/FFs) and ASIC (in terms of # of gates/chip size in mm²) devices is shown in column seven. Finally, the last column (i.e., column eight) provides the power (in μ W) information.

Table 2. Comparison to state-of-the-art ECC-based accelerator architectures.

Ref. #	m	Device	Clock Freq. (in MHz)	Clock Cycles (CCs)	Latency (in ms)	Area LUTs/FFs for FPGAs # of Gates/mm ² for ASICs	Power (in μ W)
ASIC implementations of ECC, provided specifically for RFID applications							
[14]	$GF(2^{163})$	0.13 μ m	1.13	275,816	244	12,506/–	36.63
[19]	$GF(2^{163})$	0.35 μ m	13.56	430,654	31.8	15,094/–	–
[20]	$GF(2^{163})$	0.18 μ m	0.847	296,964	350.6	13,200/–	–
[21]	$GF(2^{163})$	0.13 μ m	0.400	219,148	547.8	11,720/–	7.27
[22]	$GF(\mathbb{P}_{192})$	130 nm	200	500,000	2.50	–/0.15	–
This work	$GF(2^{163})$	Kintex-7	909	2627	2.88×10^{-3}	1786/1855	647
FPGA based ECC architectures for various constrained cryptographic applications							
[18]	$GF(2^{163})$	Virtex-7	369	3960	10.7×10^{-3}	2207 slices	–
[23]	$GF(2^{163})$	Virtex-5	359	–	110×10^{-3}	473 slices	–
[24]	$GF(2^{163})$	Virtex-7	383	–	9.9×10^{-3}	4162/1832	–
[25]	$GF(2^{163})$	Virtex-5	288.5	–	24.5×10^{-3}	3122 slices	–
[26]	$GF(2^{163})$	Virtex-7	397	–	10.5×10^{-3}	4721/1886	–
This Work	$GF(2^{163})$	Virtex-5	571	2627	4.60×10^{-3}	2546/1858	821
	$GF(2^{163})$	Virtex-7	909	2627	2.88×10^{-3}	1786/1855	733

Clock cycles for Ref # [22] are calculated using the product of frequency to latency. FPGA slices for our Virtex-5 and Virtex-7 implementations are 637 and 447.

4.2.1. Comparison for ASIC Implementations (Described for RFID Applications)

As shown in Table 2, the ECC accelerator architectures (specifically tailored for RFID applications) are synthesized on different ASIC commercial technologies. Therefore, the area comparison for these architectures is not possible as we have used an FPGA while the ASIC platform is considered in [14,19–22] for implementations. Comparison with respect to other parameters, i.e., clock frequency, CCs, latency and power (where given in state-of-the-art implementations), is provided in the text that follows.

Comparison in terms of clock cycles: All the RFID-related architectures, reported in [14,19–22], require more clock cycles as compared to the proposed accelerator architecture. This is due to the use of different architectural styles, i.e., 8/16/32-bit (including datapath logic, memories, FFs), for the computation of PM operation. In our case, we have only used an 8-bit interface to load ECC parameters and coordinates of the initial point externally from EEPROM (shown in Figure 2), while the building blocks of our architecture (shift buffers/registers, arithmetic unit and register array) are m -bit length, where m is the key size, i.e., 163. Subsequently, the proposed accelerator architecture requires 105 (ratio of 27,5816 over 2627), 164 (ratio of 430,654 over 2627), 113 (ratio of 296,964 over 2627), 83.4 (ratio of 219,148 over 2627) and 190.3 (ratio of 500 k over 2627) times lower clock cycles as compared to [14,19–22], respectively.

Comparison in terms of clock frequency, latency and power: We have provided a comparison with respect to operational clock frequency, latency and power parameters with our Kintex-7 FPGA implementation, as we have achieved low-power and low-latency values for this modern device (see Table 1 for implementation results). Concerning only the clock frequency for comparison, our accelerator architecture is 804.4 (ratio of 909 over 1.13), 67 (ratio of 909 over 13.56), 1073.1 (ratio of 909 over 0.847), 2272.5 (ratio of 909 over 0.4) and 4.5 (ratio of 909 over 200) times faster as compared to the solutions described in [14,19–22], respectively. Furthermore, the accelerator architectures, described in [14,19–21] over $GF(2^{163})$ require 84,722 (ratio of 244 over 2.88×10^{-3}), 11,041.6 (ratio of 31.8 over 2.88×10^{-3}), 121,736.1 (ratio of 350.6 over 2.88×10^{-3}) and 190,208.3 (ratio of 547.8 over 2.88×10^{-3}) times higher computation time as compared to this work. Moreover, the architecture of [22] over $GF(\mathbb{P}_{192})$ requires 868 (ratio of 2.50 over 2.88×10^{-3}) times higher computation time as compared to our PM architecture over $GF(2^{163})$.

It is important to note that the power comparison is only possible for solutions, described in [14,21]. The remaining ECC architectures for RFID applications do not consider power for implementations, as shown in Table 2. The use of three m -bit shift registers in

our high-performance implementation results in more power consumption as compared to state-of-the-art solutions. Therefore, our high-performance implementation consumes 17.6 (ratio of 647 over 36.63) and 88.9 (ratio of 647 over 7.27) times higher power as compared to [14,21] respectively. When comparing our power-optimized implementation with [14,21], this figure reduced to 4.3 (ratio of 161 over 36.63) and 22.1 (ratio of 161 over 7.27). There is always a trade-off between several design parameters (i.e., performance, area, and power). For example, higher performance results in a higher power. As provided earlier in Section 4.1.2, further power consumption of our architecture on FPGA devices can be reduced with the expense of additional CCs by using a single shift register/buffer instead of three.

4.2.2. Comparison with Fpga Based Architectures

For a realistic comparison with the state-of-the-art, we have synthesized our proposed architecture in similar devices that have been utilized in recent state-of-the-art publications that include [18,23–26].

Comparison on Virtex-5 [23,25]: Comparison in terms of clock cycles and power with [23,25] is not possible as this information is not reported. Therefore, the proposed accelerator architecture is 1.59 (ratio of 571 over 359) times faster in terms of clock frequency as compared to the PM architecture, reported in [23]. This is due to the use of a faster register array in our work while a flexible memory is utilized in [23] to support multiple PM algorithms (Binary, Montgomery, and Frobenius map) in a single design. A larger memory size drives a longer critical path delay as compared to the memories having shorter size. The longer critical path delay(s) ultimately reduces the clock frequency.

As compared to [25], the proposed architecture is 1.97 (ratio of 571 over 288.5) times faster as we have utilized an array of a faster register file whereas a BRAM (block read access memory) is employed in [25]. As far as the latency is concerned, the architectures of [23,25] require 23.91 (ratio of 110 over 4.60) and 5.32 (ratio of 24.5 over 4.60) times higher computation time as compared to this work. The FPGA slices reported in [25] are 4.90 (ratio of 3122 over 637) times higher as compared to our architecture. When comparing FPGA slices to [23], the proposed accelerator utilizes 1.34 (ratio of 637 over 473) times more hardware resources as we have used three additional shift buffers. There is always a trade-off between computation time and area.

Comparison over Virtex-7 implementations [18,24,26]: The required clock cycles as reported in a 2-stage pipelined architecture in [18] is 1.50 (ratio of 3960 over 2627) times higher as compared to this work. The power comparison is not possible. Similarly, the comparison over clock cycles and power is not possible with [24,26] as this information is not available (see Table 2). Our architecture is 2.46 (ratio of 909 over 369), 2.37 (ratio of 909 over 383) and 2.28 (ratio of 909 over 397) times faster in terms of clock frequency as compared to the solutions that are described in [18,24,26], respectively. Moreover, the architectures of [18,24,26] require 3.71 (ratio of 10.7 over 2.88), 3.43 (ratio of 9.9 over 2.88) and 3.64 (ratio of 10.5 over 2.88) times higher computation time as compared to our architecture. For comparing the hardware resources, the 2-stage pipelined architecture of [18] utilizes 4.93 (ratio of 2207 over 447) times higher FPGA slices in comparison to this work. This is due to the use of a bit-serial FF multiplier in the datapath of the proposed architecture while a digit-parallel with digit size of 32-bit multiplier architecture is incorporated in [18].

Similar to [18], another 2-stage pipelined architecture is published in [24] where a digit-parallel multiplier with a digit size of 41-bit is employed in the datapath to reduce the required clock cycles. Consequently, a digit-parallel multiplier results 2.33 (ratio of 4162 over 1786) times higher hardware resources in terms of FPGA LUTs as compared to the bit-serial multiplier in the proposed architecture. An optimized PM accelerator architecture for $\frac{\text{throughput}}{\text{area}}$ is described in [26] where a digit-serial multiplier is used to reduce hardware resources and clock cycles. Therefore, the architecture of [26] utilizes 2.64 (ratio of 4721 over 1786) times higher hardware resources in terms of FPGA LUTs as compared to this work.

5. Conclusions

This article has proposed a low-latency and low-area Elliptic-curve crypto processor architecture for an efficient use in RFID applications. The low-latency has been achieved by reducing the number of clock cycles in the datapath for loading the Elliptic-curve parameters. Furthermore, the same size of input/output interfaces has been used in other blocks of the architecture. The low area is preserved by using the same hardware resources for the squarer and multiplier operators. The proposed architecture has been validated by implementing it in state-of-the-art 7-series FPGA devices, i.e., Kintex-7, Artix-7, and Virtex-7. The obtained results indicate that the proposed accelerator utilizes 1789 LUTs and 1855 FFs in each of the Kintex-7, Artix-7 and Virtex-7 FPGA devices. However, differences in the achieved clock frequency, latency and power consumption have been recorded. For example, the achieved clock frequency and time to perform one PM operation on Kintex-7 and Virtex-7 are found to be 909 MHz and 2.88 μ s, respectively. However, the achieved clock frequency and latency on Artix-7 FPGA are 769 MHz and 3.41 μ s. From the power perspective, 647 μ W is achieved on Kintex-7 and Artix-7 FPGA devices while on a Virtex-7 FPGA, the achieved value is 733 μ W.

Author Contributions: Conceptualization, S.Z.K. and M.R.; data extraction, S.Z.K. and S.S.J.; results compilation, M.R. and A.R.A. and A.A.; validation, M.R. and M.I.; writing—original draft preparation, M.R.; critical review, M.R. and S.Z.K.; draft optimization, M.R.; supervision, S.S.J. and M.R.; funding acquisition, S.S.J. and A.R.A. and A.A. All authors have read and agreed to the published version of the manuscript.

Funding: The author Sajjad Shaukat Jamal extends his gratitude to the Deanship of Scientific Research at King Khalid University for Funding this work through a research group program under grant number R.G.P. 1/72/42.

Acknowledgments: We are thankful to the support of Deanship of Scientific Research at King Khalid University, Abha, Saudi Arabia.

Conflicts of Interest: There are no conflicts of interest among the authors.

References

1. Su, J.; Sheng, Z.; Leung, V.C.M.; Chen, Y. Energy Efficient Tag Identification Algorithms For RFID: Survey, Motivation and New Design. *IEEE Wirel. Commun.* **2019**, *26*, 118–124. [\[CrossRef\]](#)
2. Landaluce, H.; Arjona, L.; Perallos, A.; Falcone, F.; Angulo, I.; Muralter, F. A Review of IoT Sensing Applications and Challenges Using RFID and Wireless Sensor Networks. *Sensors* **2020**, *20*, 2495. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Doss, R.; Trujillo-Rasua, R.; Piramuthu, S. Secure attribute-based search in RFID-based inventory control systems. *Decis. Support Syst.* **2020**, *132*, 113270. [\[CrossRef\]](#)
4. Zhu, X. Complex event detection for commodity distribution Internet of Things model incorporating radio frequency identification and Wireless Sensor Network. *Future Gener. Comput. Syst.* **2021**, *125*, 100–111. [\[CrossRef\]](#)
5. Motroni, A.; Buffi, A.; Nepa, P. A Survey on Indoor Vehicle Localization Through RFID Technology. *IEEE Access* **2021**, *9*, 17921–17942. [\[CrossRef\]](#)
6. Giusti, I.; Cepolina, E.M.; Cangialosi, E.; Aquaro, D.; Caroti, G.; Piemonte, A. Mitigation of human error consequences in general cargo handler logistics: Impact of RFID implementation. *Comput. Ind. Eng.* **2019**, *137*, 106038. [\[CrossRef\]](#)
7. Bouzaffour, K.; Lescop, B.; Talbot, P.; Gallée, F.; Rioual, S. Development of an Embedded UHF-RFID Corrosion Sensor for Monitoring Corrosion of Steel in Concrete. *IEEE Sens. J.* **2021**, *21*, 12306–12312.
8. Park, J.; Kim, Y.J.; Lee, B.K. Passive Radio-Frequency Identification Tag-Based Indoor Localization in Multi-Stacking Racks for Warehousing. *Appl. Sci.* **2020**, *10*, 3623. [\[CrossRef\]](#)
9. Abuelkhail, A.; Baroudi, U.; Raad, M.; Sheltami, T. Internet of things for healthcare monitoring applications based on RFID clustering scheme. *Wirel. Netw.* **2021**, *27*, 747–763. [\[CrossRef\]](#)
10. Munoz-Ausecha, C.; Ruiz-Rosero, J.; Ramirez-Gonzalez, G. RFID Applications and Security Review. *Computation* **2021**, *9*, 69. [\[CrossRef\]](#)
11. Liu, Z.; Liu, D.; Zou, X.; Lin, H.; Cheng, J. Design of an Elliptic Curve Cryptography Processor for RFID Tag Chips. *Sensors* **2014**, *14*, 17883–17904. [\[CrossRef\]](#) [\[PubMed\]](#)
12. Suresh, T.; Ramakrishnan, M. Design of low power NFSR for RFID system with irregular clock pulse. *Microprocess. Microsyst.* **2020**, *73*, 102983. [\[CrossRef\]](#)
13. Rashid, M.; Imran, M.; Jafri, A.R.; Al-Somani, T.F. Flexible Architectures for Cryptographic Algorithms—A Systematic Literature Review. *J. Circuits Syst. Comput.* **2019**, *28*, 1930003. [\[CrossRef\]](#)

14. Lee, Y.K.; Sakiyama, K.; Batina, L.; Verbaudhede, I. Elliptic-Curve-Based Security Processor for RFID. *IEEE Trans. Comput.* **2008**, *57*, 1514–1527. [\[CrossRef\]](#)
15. Liu, D.; Liu, Z.; Yong, Z.; Zou, X.; Cheng, J. Design and Implementation of An ECC-Based Digital Baseband Controller for RFID Tag Chip. *IEEE Trans. Ind. Electron.* **2015**, *62*, 4365–4373. [\[CrossRef\]](#)
16. Noori, D.; Shakeri, H.; Niazi, T.M. Scalable, efficient, and secure RFID with elliptic curve cryptosystem for Internet of Things in healthcare environment. *Eurasip J. Inf. Secur.* **2020**, *2020*, 13. [\[CrossRef\]](#)
17. Sajid, A.; Rashid, M.; Jamal, S.S.; Imran, M.; Alotaibi, S.S.; Sinky, M.H. AREEBA: An Area Efficient Binary Huff-Curve Architecture. *Electronics* **2021**, *10*, 1490. [\[CrossRef\]](#)
18. Imran, M.; Rashid, M.; Jafri, A.R.; Kashif, M. Throughput/area optimised pipelined architecture for elliptic curve crypto processor. *IET Comput. Digit. Tech.* **2019**, *13*, 361–368. [\[CrossRef\]](#)
19. Kumar, S.S.; Paar, C. Are Standards Compliant Elliptic Curve Cryptosystems Feasible on RFID? Available online: http://sandeep.de/my/papers/2006_RFIDSec_TinyECC.pdf (accessed on 21 June 2021).
20. Hein, D.; Wolkerstorfer, J.; Felber, N. ECC Is Ready for RFID—A Proof in Silicon. In *Selected Areas in Cryptography*; Avanzi, R.M., Keliher, L., Sica, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 401–413.
21. Kocabaş, U.; Fan, J.; Verbaudhede, I. Implementation of binary edwards curves for very-constrained devices. In Proceedings of the ASAP 2010—21st IEEE International Conference on Application-Specific Systems, Architectures and Processors, Rennes, France, 7–9 July 2010; pp. 185–191.
22. Furbass, F.; Wolkerstorfer, J. ECC Processor with Low Die Size for RFID Applications. In Proceedings of the 2007 IEEE International Symposium on Circuits and Systems, New Orleans, LA, USA, 27–30 May 2007; pp. 1835–1838.
23. Khan, Z.U.A.; Benaissa, M. Low area ECC implementation on FPGA. In Proceedings of the 2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS), Abu Dhabi, United Arab Emirates, 8–11 December 2013; pp. 581–584.
24. Imran, M.; Pagliarini, S.; Rashid, M. An Area Aware Accelerator for Elliptic Curve Point Multiplication. In Proceedings of the 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Glasgow, UK, 23–25 November 2020; pp. 1–4.
25. Rashidi, B. Low-Cost and Fast Hardware Implementations of Point Multiplication on Binary Edwards Curves. In Proceedings of the Iranian Conference on Electrical Engineering (ICEE), Mashhad, Iran, 8–10 May 2018; pp. 17–22.
26. Khan, Z.U.A.; Benaissa, M. Throughput/Area-efficient ECC Processor Using Montgomery Point Multiplication on FPGA. *IEEE Trans. Circuits Syst. II Express Briefs* **2015**, *62*, 1078–1082. [\[CrossRef\]](#)
27. Zhang, Y.; Chen, D.; Choi, Y.; Chen, L.; Ko, S.B. A high performance ECC hardware implementation with instruction-level parallelism over GF(2163). *Microprocess. Microsyst.* **2010**, *34*, 228–236. [\[CrossRef\]](#)
28. Imran, M.; Rashid, M. Architectural review of polynomial bases finite field multipliers over GF(2^m). In Proceedings of the 2017 International Conference on Communication, Computing and Digital Systems (C-CODE), Islamabad, Pakistan, 8–9 March 2017; pp. 331–336.
29. Dimitrov, V.; Järvinen, K. Another Look at Inversions over Binary Fields. In Proceedings of the 2013 IEEE 21st Symposium on Computer Arithmetic, Austin, TX, USA, 7–10 April 2013; pp. 211–218.
30. Xilinx. 7 Series FPGAs: Breakthrough Power and Performance, Dramatically Reduced Development Time. Available online: https://www.xilinx.com/publications/prod_mktg/7-Series-Product-Brief.pdf (accessed on 29 May 2021).