applied sciences

MDPI

Article

A Hybrid Approach for Efficient and Secure Point Multiplication on Binary Edwards Curves

Asher Sajid [1,*], Omar S. Sonbul [2], Muhammad Rashid [2,*] and Muhammad Yousuf Irfan Zia [3,4]

[1] Deanship of Scientific Research, Umm Al Qura University, Makkah 21955, Saudi Arabia
[2] Computer Engineering Department, Umm Al Qura University, Makkah 21955, Saudi Arabia; ossonbul@uqu.edu.sa
[3] Department of Electrical Engineering, Ziauddin University, Karachi 74600, Pakistan; yousuf.irfan@zu.edu.pk
[4] Telecommunications Engineering School, University of Malaga, 29010 Málaga, Spain
* Correspondence: malikasher267@gmail.com (A.S.); mfelahi@uqu.edu.sa (M.R.)

Abstract: The focus of this article is to present a novel crypto-accelerator architecture for a resource-constrained embedded system that utilizes elliptic curve cryptography (ECC). The architecture is built around Binary Edwards curves (BEC) to provide resistance against simple power analysis (SPA) attacks. Furthermore, the proposed architecture incorporates several optimizations to achieve efficient hardware resource utilization for the point multiplication process over $GF(2^m)$. This includes the use of a Montgomery radix-2 multiplier and the projective coordinate hybrid algorithm (combination of Montgomery ladder and double and add algorithm) for scalar multiplication. A two-stage pipelined architecture is employed to enhance throughput. The design is modeled in Verilog HDL and verified using Vivado and ISE design suites from Xilinx. The obtained results demonstrate that the proposed BEC accelerator offers significant performance improvements compared to existing solutions. The obtained throughput over area ratio for $GF(2^{233})$ on Virtex-4, Virtex-5, Virtex-6, and Virtex-7 Xilinx FPGAs are 9.43, 14.39, 26.14, and 28.79, respectively. The computation time required for a single point multiplication operation on the Virtex-7 device is 19.61 µs. These findings indicate that the proposed architecture has the potential to address the challenges posed by resource-constrained embedded systems that require high throughput and efficient use of available resources.

Keywords: elliptic curve cryptography; binary Edwards curve; scalar multiplication; Montgomery radix-4 multiplier; FPGA

Citation: Sajid, A.; Sonbul, O.S.; Rashid, M.; Zia, M.Y.I. A Hybrid Approach for Efficient and Secure Point Multiplication on Binary Edwards Curves. *Appl. Sci.* **2023**, *13*, 5799. https://doi.org/10.3390/app13095799

Academic Editor: Alessandro Lo Schiavo

Received: 14 April 2023
Revised: 4 May 2023
Accepted: 4 May 2023
Published: 8 May 2023

1. Introduction

The internet is a vast network of interconnected computers that enable communication between people from all over the world. It provides a cost-effective way to communicate over long distances, making it an ideal medium for businesses and organizations [1]. However, with the convenience of the internet, there is always a risk of cyber-attacks [2]. Cyber-criminals can access sensitive information transmitted over the internet, including personal data, financial information, and corporate secrets. To mitigate this risk, cryptography is one of the techniques that is used to ensure the confidentiality, integrity, and authenticity of data. It is the practice of securing data by converting it into an unreadable format called cipher text [3]. The cipher text can only be read by someone who has the decryption key to convert it back to its original format.

Cryptography can be either symmetric or asymmetric. Symmetric cryptography uses the same key for both encryption and decryption. The sender and the receiver both have access to the same key, making it easier to encrypt and decrypt data. However, the symmetric approach is less secure as the key needs to be shared between the sender and receiver, making it more vulnerable to cyber-attacks [4]. Asymmetric cryptography, on the other hand, uses two different keys for encryption and decryption [5]. The sender encrypts the data with the receiver's public key, and the receiver decrypts the data with their private

key. Asymmetric cryptography is more secure than symmetric cryptography as the private key is never shared, making it more difficult for cyber-criminals to intercept the data. ECC is a type of asymmetric cryptography that is gaining popularity due to its security and efficiency [6,7]. It is more secure than RSA [8], another popular asymmetric cryptography algorithm, because it requires shorter key sizes to achieve the same level of security [9]. This means that ECC is faster and requires less storage space, making it ideal for mobile devices with limited processing power and storage capacity [10,11].

ECC implementation can be thought of as a stack of four distinct layers [3]. The foundation of the stack is the arithmetic operations layer, which includes the fundamental building blocks of mathematics such as addition, subtraction, multiplication, and division. These operations form the basis of elliptic curve group operations, which enable ECC's security and efficiency. The second layer of the stack is the point addition and point doubling layer, which enables the construction of new points on the elliptic curve by performing operations that combine existing points. This layer is a key component in ECC's scalability, as it allows for the generation of multiple points from a single base point. The third layer of the stack is point multiplication, which involves multiplying a point on the elliptic curve by a scalar. This layer is responsible for generating the final result of ECC operations, which are new points on the curve. The fourth and final layer is the protocol layer, which includes the implementation of ECC-based protocols used for key exchange, digital signatures, and encryption. This layer is responsible for the practical application of ECC, enabling secure communication and data protection in a wide range of contexts.

One of the important challenges in implementing ECC is the risk of side-channel attacks [12], where an attacker can extract information from the implementation of the algorithm rather than trying to break the mathematical problem. One way to mitigate this risk is to use the hybrid algorithms that combine multiple techniques, such as Montgomery ladder and double-and-add algorithms. It can potentially increase the resistance to SPA attacks [13,14]. In this context, Binary Edwards curves [15], which are known for their ability to resist side-channel attacks, are being increasingly used in various high-security applications. Compared to Binary Huff curves [16] and Hessian curves [17], Binary Edwards curves are not only faster and more secure, but also have a smaller key size and require less computational resources, making them ideal for constrained environments [6]. The potential use of these curves can be found in applications such as cloud computing [18], secure messaging [19], digital signatures [20], and internet of things [21–23], where high security, high throughput, and efficient resource utilization are critical. Overall, the adoption of Binary Edwards curves can significantly enhance the security of these applications while maintaining optimal performance.

1.1. Related Work

Recent years have seen the development of several FPGA-based implementations for point multiplication on Binary Edwards curves (BECs) with a focus on optimizing speed and resources. One study, published in [24], presents two different architectures for general and special formulations of BECs. These architectures utilize optimized finite field multiplication techniques and multiple finite field multipliers to improve efficiency and accelerate high-speed applications. The general BEC architecture used 2272, 5919, and 4581 slices for different curve parameters, with latencies of 74.55, 26.24, and 51.46, respectively, on a Virtex-5 platform.

For special formulations of BECs, two different structures were presented based on three and two parallel multipliers. The three parallel multiplier implementations achieved high speed, but required high hardware resources, with 4454 hardware resources and a latency of 34.61 microseconds for curve parameters d_1 and d_2 equal to 59. The two multiplier implementation achieved a low hardware resource of 3521 slices, but with a high latency of 57.43 microseconds for curve parameters d_1 and d_2 equal to 59. These results highlight the trade-off between hardware resources and latency in the implementation of BEC architectures for high-speed applications.

In [25], a reconfigurable processor architecture has been proposed for implementing Binary Edwards curves (BECs) in cryptographic algorithms. The architecture was implemented on the Virtex-4 platform and achieved a maximum clock frequency of 48 MHz. The study demonstrated that the architecture required 21,816 slices for BECs and 22,373 slices for BECs with halving, indicating a minimal increase in hardware resources needed for the halving operation. The presented architecture is flexible and can be reconfigured for various curve parameters, making it suitable for implementing BEC-based cryptographic algorithms. In [26], the focus is on optimizing the area for embedded devices with limited resources, and the authors have proposed a digit-serial multiplier and m-bit XOR layer for this purpose. The area figures for Virtex 6 and Virtex 5 are reported as 2138 and 2153 slices, respectively.

To optimize both the throughput and area, the authors in [27] have employed a pipelined digit-serial multiplier to perform successive PA and PD computations, which reduces the critical path, conserves clock cycles, and optimizes the clock frequency. For a curve parameter of d = 59, the implemented hardware has used 8875 slices on the Virtex-5 platform for GHC, with a latency of 11.03 microseconds, and 11,494 slices for BEC with a latency of 11.01 microseconds.

Other recent studies, such as [28,29], have proposed architectures to optimize different design factors such as latency, area, and throughput. In [28], a comb PM technique is employed to create low-complexity (LC) and low-latency (LL) architectures. The LC design offers significant improvements of 62%, 46%, and 152% for $GF(2^{233})$, $GF(2^{163})$, and $GF(2^{283})$, respectively. Additionally, the LL architecture enables faster computation of one PM operation.

The authors in [30] have introduced a modular radix-2 interleaved multiplier that can improve low-latency architecture by reducing computing time, clock cycles, and area. This multiplier uses the Montgomery ladder algorithm to perform PM. Similarly, the work in [15] has presented a low-complexity architecture for PM that uses a digital parallel least significant multiplier and instruction scheduling to optimize hardware resources. The architecture has been tested on Virtex 4, Virtex 5, Virtex 6, and Virtex 7 platforms.

1.2. Research Gap

Some potential research gaps in the field of FPGA-based implementations for point multiplication on Binary Edwards curves (BECs) include:

- Limited focus on optimizing multiple design factors: While many of the studies discussed in Section 1.1 focus on optimizing either area, latency, or throughput, there may be a need for further research that can balance these factors in a more holistic manner. For example, an architecture that achieves high throughput and low latency but also minimizes hardware resources could be useful in many high-speed applications.
- Investigating novel multipliers: Many of the existing implementations use pipelined digit-serial multipliers to optimize throughput, but there may be other novel modular multipliers that could further improve performance.
- Limited evaluation of security and robustness: While the studies discussed in Section 1.1 focus on optimizing speed and resources for BEC architectures, there is limited discussion on the security and robustness of these architectures. Further research could evaluate the effectiveness of these architectures in resisting attacks and ensuring data integrity, which is crucial for many cryptographic applications.

1.3. Contributions

The article targets to increase the speed of the architecture while using less hardware resources, and it presents various contributions towards achieving this goal.

- The proposed hybrid algorithm combines two different approaches, Montgomery ladder and Double and Add algorithm, to achieve unprecedented performance improvements. The Montgomery ladder algorithm is a commonly used method for scalar multiplication in elliptic curve cryptography, while the Double and Add algorithm is

a simple and efficient method for point addition and doubling. By combining these two algorithms, the proposed hybrid algorithm is able to leverage the strengths of each approach and achieves better performance than either algorithm used alone.
- The proposed multiplication technique, based on radix-2 arithmetic, is a method for performing modular multiplication in a more efficient way. This technique splits the multiplication process into smaller subproblems and performs them using radix-2 arithmetic. By using this approach, the proposed algorithm is able to perform modular multiplication more efficiently and accurately, which is a critical operation in many cryptographic algorithms.
- The proposed two-stage pipelining technique is a method for increasing the throughput of the entire architecture by splitting the computation into smaller stages and processing them in parallel. This approach reduces the overall latency of the algorithm and allows for higher clock frequencies, resulting in faster computation times. By using this technique, the proposed algorithm is able to process more data in a shorter amount of time, making it more efficient and effective for real-world applications.
- A Finite State Machine (FSM) has been created to control the data path of a proposed architecture. This control path efficiently manages the operations of the data path.

The structure of the article is described as follows: Section 2 provides the theoretical background pertaining to BEC. Section 3 describes the proposed hybrid algorithm that has been implemented in this article. Section 4 describes the proposed optimizations for performance improvement. Section 5 provides information on the proposed hardware architecture. The main findings of the article are discussed in Section 6. Finally, the conclusions are made in Section 7.

2. Mathematical Background

Section 2.1 contains the BEC equations for both prime and binary fields. In Section 2.2, there is a description of the unified mathematical formulation. Section 3 provides information about the point multiplication computations and the hybrid algorithm.

2.1. BEC Equations over $GF(2^m)$

Harold Edwards developed a comprehensive model for Binary Edwards curves (BECs) in 2007, which includes a complete group law. The mathematical representation of BECs over a prime field with coefficient d is given by Equation (1)

$$x^2 + y^2 = 1 + dx^2y^2 \tag{1}$$

In Equation (1), x and y represent the initial points, and d represents the curve parameter. However, working with large prime fields can be challenging, so Bernstein proposed a binary version of Edwards curves to overcome this issue.

$$E_{B,d_1,d_2} : d_1(x + y) + d_2(x^2 + y^2) = xy + xy(x + y) + x^2y^2 \tag{2}$$

Equation (2) shows the binary form of Edwards curves, where x and y denote the initial points, and d_1 and d_2 represent the curve parameters. Equation (2) holds true when d_1 is not equal to 0 and d_2 is not equal to $d_1^2 + d_1$.

2.2. Unified Mathematical Formulation

The table presented as Table 1 outlines the differential PA and PD instructions for BEC over $GF(2^m)$. These instructions are comprised of 7 complex steps and require a memory unit capable of storing initial, intermediate, and final results, with a total memory requirement of $11 \times m$, where 11 represents the number of memory locations and m represents the width of each location. The BEC curve parameters in the table, denoted as e_1, e_2, and w, are computed based on a rational function for an elliptic curve E over GF and the values of d_1 and d_2. The initial projective points are represented by W_1, Z_1, W_2, and Z_2, while the

final points are represented by Z_a, Z_d, W_a, and W_d. Intermediate values are stored in A, B, and C.

Table 1. PA and PD instructions.

Instructions	Original Formulas
$Instr_1$	$A \leftarrow W_1 \times Z_1$
$Instr_2$	$B \leftarrow W_1 \times W_2$
$Instr_3$	$C \leftarrow Z_1 \times Z_2$
$Instr_4$	$W_d \leftarrow A \times A$
$Instr_5$	$Z_d \leftarrow ((e_1 \times W_1 + Z_1)^4)$
$Instr_6$	$Z_a \leftarrow (e_2 \times B + C)^2)$
$Instr_7$	$W_a \leftarrow (B \times C + w \times Z_a)^2$

3. Proposed Hybrid Algorithm

This article has performed the computation of point multiplication (PM) over Binary Edwards curves (BEC) using the Hybrid algorithm. The algorithmic details are provided in Section 3.1. Subsequently, the significance of the hybrid algorithm is highlighted in Section 3.2.

3.1. Algorithmic Details

Given a scalar multiplier k and a starting point P, the PM operation computes the point Q, which is equal to k times P. This can be expressed as $Q = K(P + P + \ldots + P)$, where there are K terms in the sum.

To convert a point P from affine coordinates to ω coordinates as shown in Algorithm 1, we calculate the values of W_2 and Z_2 using the formula $W_2 = x(P) \cdot Z_2^{-1}$ and $Z_2 = 1$. We set W_1 and Z_1 to 0. For point multiplication, we iterate over the binary digits of scalar k from the most significant bit to the least significant bit. For each bit i, we perform the following steps:

- If the i-th bit of k is 0, we perform a point addition operation $P = P + Q$ and update the ω coordinates of the points using the dADD function. We set $(W_1 : Z_1), (W_2 : Z_2) = dADD((W_0 : Z_0), (W_1 : Z_1), (W_2 : Z_2))$, where $(W_0 : Z_0)$ is the ω coordinates of P.
- If the i-th bit of k is 1 and the $(i-1)$-th bit is 0 or $i = m - 1$, we perform a double-and-add operation. We double the current value of the result using the double function, and update the ω coordinates of the points as $(W_1 : Z_1), (W_2 : Z_2) = double((W_1 : Z_1), (W_2 : Z_2))$.
- If the i-th bit of k is 1 and the $(i-1)$-th bit is also 1, we perform a point addition operation $Q = P + Q$ using the current value of P, and update the ω coordinates of the points as $(W_2 : Z_2), (W_1 : Z_1) = dADD((W_0 : Z_0), (W_2 : Z_2), (W_1 : Z_1))$.

After iterating over all the bits of k, we return the ω coordinates of the resulting point Q as $(W_1 : Z_1), (W_2 : Z_2)$.

Algorithm 1: Hybrid Montgomery and double-and-add algorithm

Input: $E_{B,d_1,d_2}/GF(2^m) : d_1(x+y)+d_2(x+y)^2 = xy(x+1)(y+1)$, $P \in E_{B,d_1,d_2}$ and $k = (k_{m-1}, \ldots, k_1, k_0)$
Output: $Q = w(k.P)$
$W_1 = 0, Z_1 = 1, W_2 = w(P), Z_2 = w(P)$ ▷ **Step-1: —Conversions from Affine to ω Coordinates.**
for *(i from $m-1$ down to 0)* **do**
▷ **Step-2: —Point Multiplication.**
if $k_i = 0$ **then**
$PointAddition \rightarrow P = P + Q$
$((W_1 : Z_1), (W_2 : Z_2)) := dADD((W_0 : Z_0), (W_1 : Z_1), (W_2 : Z_2))$
else
if $k_i \neq k_{i-1}$ **then**
$Q = P + Q$
$((W_2 : Z_2), (W_1 : Z_1)) := dADD((W_0 : Z_0), (W_2 : Z_2), (W_1 : Z_1))$
else
$((W_1 : Z_1), (W_2 : Z_2)) := double((W_1 : Z_1), (W_2 : Z_2))$
End If
End If
End For
$Return : (W_1 : Z_1), (W_2 : Z_2)$

3.2. Benefits of the Hybrid Algorithm

The hybrid algorithm aims to efficiently perform point multiplication on BEC in ω coordinates. The benefits of a hybrid approach are as follows:

- Efficient point multiplication: The algorithm uses a combination of Montgomery ladder and double-and-add methods to perform point multiplication on BEC efficiently.
- Reduced number of point additions: The algorithm performs point addition only when the current bit of the scalar is 0. This reduces the number of point additions required as compared to the standard double-and-add algorithm, where point additions are performed at every iteration.
- Reduced number of point doublings: The algorithm performs point doubling only when two consecutive bits of the scalar are 1. This further reduces the number of point-doubling operations compared to the standard double-and-add algorithm, where point-doubling is performed at every iteration.

Overall, the hybrid algorithm provides an efficient way to perform point multiplication on BEC. The Montgomery ladder algorithm provides efficient scalar multiplication without revealing any intermediate values, while the double-and-add algorithm improves the performance of the scalar multiplication by avoiding unnecessary additions and doublings.

4. Proposed Optimizations

To enhance the throughput/area ratio, this article utilizes two-stage pipelining and modular multiplier technique. The former breaks down computation into smaller stages, reducing the critical path delay and improving throughput. The latter technique reduces clock cycles needed for multiplication, optimizing clock frequency and performance.

4.1. Optimizing Hardware Designs with Pipelining: Three Approaches for Division and Register Placement

Pipelining is a widely used method to enhance the processing speed of hardware designs. To optimize the pipelining process, the circuit is divided into three sections: pre-calculated data and output data from the memory unit for the read operation, arithmetic logical unit (ALU) operations for the execute operation, and the routing network associated with the memory unit for the write-back operation. This division leads to three possible outcomes.

The first outcome is non-pipelined, where the read, execute, and write-back operations occur in a single cycle. The second outcome is a 2-stage pipelined design, with a register placed at the input of the ALU. This design allows us to read in the first cycle and execute and write-back in the second cycle. The third outcome is a 3-stage pipelined design where registers are used for both the input and output of the ALU. This design allows for read, execute, and write-back operations to occur in three different cycles.

Comparing the first two outcomes, the 2-stage pipelined design offers a better throughput/area ratio than the non-pipelined architecture. However, adding a third pipeline stage for write-back is not necessary as it can increase the number of clock cycles due to potential RAW hazards. Additionally, increasing the number of registers at the output of the ALU can reduce the overall throughput/area ratio. Therefore, it is concluded that the 2-stage pipelined design with pipeline registers at the input of the ALU is the most efficient architecture for this processor design.

4.2. Instructions and Scheduling for Differential Addition Law in Two-Stage Pipelined Architecture

Table 2 outlines the instructions required to implement the differential addition law of Algorithm 1 in a two-stage pipelined architecture. The table provides details on the number of clock cycles required for each instruction, the instructions themselves, and the merging of multiple operations to reduce the complexity of the instructions. Furthermore, the table includes information on the status of the two-stage pipelining, as well as the potential RAW hazard that may occur due to the pipelining. The proposed scheduling for the instructions of the unified differential law formulas is also presented in columns six through eight.

Table 2. Proposed optimized version of BEC instructions.

		Two Stage Pipelining with Original Formulas			**Two Stage Pipelining with Proposed Formulas**		
CCs	**Instructions**	**Without Scheduling**			**With Scheduling**		
–	–	**Instructions**	**Pipeline Status**	**RAW**	**Instructions**	**RAW**	**Pipeline Status**
1	$Instr_1$	$A = W_1 \times Z_1$	R[I_1]	–	$T_1 = W_1 \times Z_1$	–	R[I_1]
2	$Instr_2$	$B = W_1 \times W_2$	R[I_2], $E[I_1]$, $WB[I_1]$	–	$T_2 = W_1 \times W_2$	–	R[I_2], $E[I_1]$, $WB[I_1]$
3	$Instr_3$	$C = Z_1 \times Z_2$	R[I_3], $E[I_2]$, $WB[I_2]$	–	$T_3 = Z_1 \times Z_2$	–	R[I_3], $E[I_2]$, WBI_2
4	$Instr_4$	$W_d = A \times A$	R[I_4], $E[I_3]$, $WB[I_3]$	–	$T_4 = T_1 \times T_1$	–	R[I_4], $E[I_3]$, $WB[I_3]$
5	$Instr_5$	$T_1 = e_1 \times W_1$	R[I_5], E[I_4], WB[I_4]	–	$W_0 = e_1 \times W_1$	–	R[I_5], $E[I_4]$, $WB[I_4]$
6	$Instr_6$	$T_2 = T_1 + Z_1$	E[I_5], WB[I_5]	T_1	$Z_0 = W_0 + Z_1$	–	R[I_6], E[I_5], WB[I_5]
7	$Instr_7$	$T_3 = T_2 \times T2$	R[I_6]	T_2	merged	–	R[I_7], E[I_6], WB[I_6]
8	$Instr_8$	$Z_d = T_3 \times T_3$	E[I_6], WB[I_6]	T_3	$T_1 = (Z_0 Z_0)^2$	–	R[I_8], E[I_7], WB[I_7]
9	$Instr_9$	$T_1 = e_2 \times B$	R[I_7]	–	$W_0 = e_2 \times T_2$	–	R[I_9], E[I_8], WB[I_8]
10	$Instr_{10}$	$T_2 = T_1 + C$	R[I_8], E[I_7], WB[I_7]	T_1	$Z_0 = W_0 + T_3$	–	R[$I_1$0], E[I_9], WB[I_9]
11	$Instr_{11}$	$Z_a = T_2 \times T_2$	R[I_9], E[I_8], WB[I_8]	T_2	$W_0 = Z_0 \times Z_0$	–	R[I_{11}], E[I_{10}], WB[I_{10}]
12	$Instr_{12}$	$T_2 = B \times C$	E[I_9], WB[I_9]	–	$Z_0 = T_2 \times T_3$	–	R[I_{12}], E[I_{11}], WB[I_{11}]
13	$Instr_{13}$	$T_3 = w \times Z_a$	R[I_{10}]	T_3	$T_2 = w \times W_0$	–	R[I_{13}], E[I_{12}], WB[I_{12}]
14	$Instr_{14}$	$W_a = T_3 + T2$	E[I_{10}], WB[I_{10}]	–	$T_3 = Z_0 + T_2$	T_2	E[I_{13}], WB[I_{13}]
15	–	–	R[I_{11}]	–	–	–	R[I_{14}]
16	–	–	R[I_{12}], E[I_{11}], WB[I_{11}]	–	–	–	E[I_{14}], WB[I_{14}]
17	–	–	R[I_{13}], E[I_{12}], WB[I_{12}]	–	–	–	–
18	–	–	E[I_{13}], WB[I_{13}]	–	–	–	–
19	–	–	R[I_{14}]	–	–	–	–
20	–	–	E[I_{14}], WB[I_{14}]	–	–	–	–

4.3. Storage Elements and RAW Hazard Description

The proposed architecture employs a total of $14 \times m$ storage elements, which serve various purposes. These elements include A, B, C, Wd, Zd, Wa, Za, W1, W2, Z1, Z2, T1, T2, and T3. The initial projective points are stored in W1, Z1, W2, and Z2, while the updated values of the final projective point are saved in Wa, Za, Wd, and Zd, as indicated in column 3 of Table 2. Meanwhile, intermediate results are stored in the remaining storage elements, namely A, B, C, and T1 to T3.

When executing R, E, and WB in a single clock cycle, each operation requires 14 cycles (refer to column 3 of Table 2). However, when employing 2-stage pipelining, certain instructions, including $Instr_6$, $Instr_7$, $Instr_8$, $Instr_{10}$, $Instr_{11}$, and $Instr_{13}$, are vulnerable to RAW hazards, as specified in column seven of Table 2. For instance, there is a RAW hazard when writing to $Instr_6$, which results in a one-cycle delay because it necessitates two cycles to compute the new value of T_1. Hence, accounting for the RAW hazard, a total of 20 cycles are necessary for each unified PA and PD.

4.4. Optimizing Instruction Scheduling to Reduce Hazards and Hardware Resources

In order to optimize hardware resources and reduce hazards, it is common practice to perform instruction scheduling. As seen in Column three of Table 2, $Instr_7$ and $Instr_8$ are executed in two separate clock cycles, where T_3 is first computed as the result of $T_2 \times T_2$, and then used as an input for the subsequent multiplication in $Instr_8$ to compute Z_d. However, an alternative approach is described in column six of Table 2, where the squarer unit is used immediately after the multiplier unit. This allows $Instr_7$ and $Instr_8$ to be executed in a single clock cycle, resulting in several benefits. Firstly, it reduces the total number of instructions required, leading to better performance. Secondly, it reduces the number of required storage elements from $14 \times m$ to $10 \times m$, thereby saving valuable hardware resources. Lastly, it also reduces the clock cycles required when computing PM for m bit (233).

The scheduling approach outlined in column six of Table 2 has the advantage of resulting in only a single RAW hazard in the context of pipelining, as noted in column seven. When computing the PA and PD formulations using a 2-stage pipelined architecture, this approach requires a total of 16 clock cycles, which is fewer than the number of clock cycles required when executing $Instr_7$ and $Instr_8$ in separate cycles. Overall, the reduction in the number of clock cycles required, coupled with the reduction in the number of storage elements needed, makes this approach a more efficient use of hardware resources. Additionally, by reducing the total number of instructions required, this approach can also improve performance.

5. Proposed Hardware Architecture

The proposed accelerator architecture is motivated by the need to improve the performance of ECC curve (BEC) operations in resource-constrained devices. ECC is a popular choice for cryptography, but it can be computationally expensive. The proposed accelerator architecture can improve the performance of ECC operations, making them more suitable for use in resource-constrained devices. The proposed accelerator architecture is designed to improve the performance of BEC operations in resource-constrained devices. The architecture is based on a number of design principles, including:

1. Data parallelism: The architecture exploits data parallelism by performing multiple BEC operations at the same time. This can significantly improve the performance of BEC operations.
2. Pipelining: The architecture uses pipelining to overlap the execution of different BEC operations. This can further improve the performance of BEC operations.
3. Modular multiplication: The architecture uses modular multiplication to reduce the number of arithmetic operations required to perform BEC operations. This can improve the performance and energy efficiency of BEC operations.

5.1. Overview of the Architecture

The proposed architecture is composed of several components, including a memory unit (MU) for storing intermediate and final results, routing networks (RN) for transferring data, a read-only memory (ROM) for reading BEC parameters, an arithmetic logic unit (ALU) for performing computations, and a finite state machine (FSM) for generating control signals. To achieve pipelining, registers are placed at the input of the ALU. The design is based on the parameters recommended by the National Institute of Standards and Technology (NIST). Detailed descriptions of each component are provided in their respective Sections 5.2–5.6.

5.2. Memory Unit

The proposed architecture utilizes a memory unit with a size of $10 \times m$ to store intermediate and final results, as illustrated in Figure 1. Here, the value 10 refers to the number of memory locations, while m represents the width of each memory location. The initial projective points are stored in storage elements W_1, Z_1, W_2, and Z_2, while the updated projective points are stored in T_4, T_1, W_0, and T_3. Additionally, the intermediate results, as presented in Table 2, are stored in Z_0 and T_2.

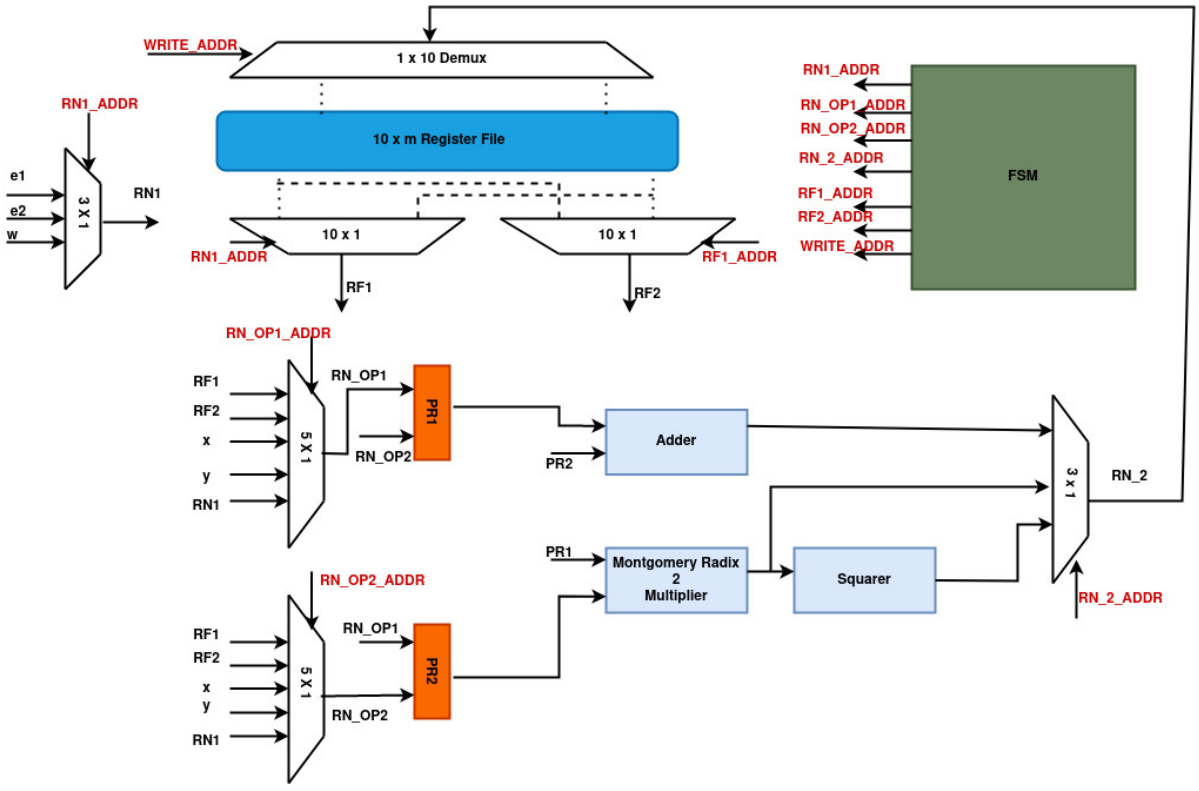

Figure 1. Proposed BEC architecture.

To store these storage elements in memory locations, a 1×10 demultiplexer (DEMUX) with control signal WRITE_ADDR is employed. Two multiplexers, RF1 and RF2, are utilized to retrieve storage elements from the memory unit for further processing, with control signals RF1_ADDR and RF2_ADDR. The size of RF1 and RF2 is $10 \times m$. The output of RF1 is RF1, while the output of RF2 is RF2.

5.3. Routing Networks

The proposed architecture, shown in Figure 1, utilizes three routing networks (RN2, RN3 and RN4) to transfer data between different modules. The input data, including base coordinates x and y, the output of RN1, and the output of RF1 and RF2, are fed into RN2

and RN3. Control signals RN_OP1_ADDR and RN_OP2_ADDR are used to select the appropriate data for processing. The sizes of RN2 and RN3 are 5×1. RN4, with a size of 3×1, is utilized to select the output of the ALU as its input. This architecture efficiently transfers data between the different modules, ensuring smooth operation and minimizing processing delays.

5.4. Read Only Memory

The proposed architecture employs a read-only memory (ROM) to access pre-calculated curve constant values. The RN1 has a size of 3×1 and is depicted in Figure 1. It uses a single multiplexer to select one of the three constant values (e_1, e_2, and w).

5.5. Arithmetic Logic Unit

The proposed two-stage pipelined architecture includes adder, multiplier, and squarer units, as shown in Figure 1. To implement the adder unit, m bitwise exclusive OR gates are used, where m represents the key length. Multiplication is a crucial operation in cryptographic applications, and there are several techniques available in the literature. In this work, we utilize the Montgomery radix-2 multiplier, which is discussed in detail in Section 5.5.1. The squarer unit is placed after the multiplier unit, as seen in Figure 1. By adding a "0" after each input data value, the squarer unit is implemented, as described in [31]. The squarer unit's purpose is to minimize the total number of clock cycles (CCs) required for PM calculation, as instructions such as $(A \times B)^2$ can be computed using the squarer unit. After each polynomial multiplication and squaring unit, an inverse operation is necessary. The quad block Itoh–Tsujii method [15] is employed using the multiplier and squarer units to execute the inversion operation.

5.5.1. Montgomery Radix-2 Multiplier

The Montgomery multiplication method is a technique that replaces expensive division operations with simpler shift and addition operations, resulting in faster modular arithmetic computations. The radix-2 Montgomery method is a basic implementation of this method, as shown in Algorithm 2. The loop in the algorithm iterates over each element in the input vectors Xp and Yp, and computes a partial product $Xp_i \times Yp$ for each element. The partial product is added to the accumulator A in step 1.

$A[0]$ refers to the least significant bit of the accumulator A. In step 2 of the algorithm, the value of $A[0]$ is used to determine whether to add the modulus p or not. If $A[0]$ is 1 (i.e., the result of the addition operation in step 1 is odd), then p is added to A in step 3 to make it even. If $A[0]$ is 0 (i.e., the result of the addition operation in step 1 is even), then adding p to A in step 3 has no effect on the value of A, and the algorithm proceeds to step 4 to shift A right by one bit. The result Zp is then returned. By replacing the division operation with shift and addition operations, the radix-2 Montgomery method results in a faster modular multiplication algorithm.

Note that the input parameters p, Xp, and Yp are preprocessed to satisfy certain conditions, which are necessary for the correctness of the algorithm. Specifically, p is expressed in the form $\sum_{i=0}^{n-1} p_i^2$, where p_i are binary digits, Xp and Yp are less than p, and R is equal to 2^n. These preprocessing steps are not explicitly shown in Algorithm 2.

Algorithm 2: Montgomery radix-2 multiplier.

Input: $p = \sum_{i=0}^{n-1} p_i^2$, $Xp = (\sum_{i=0}^{n-1} X_{p_i}^2) < p$, $Yp = (\sum_{i=0}^{n-1} Y_{p_i}^2) < p$, $R = 2^n$
Output: $Zp = (Xp \times Yp) \mod p$
$A = 0$
for $i = 0$ *to* $n - 1$ **do**
1. $A = A + X_{p_i} \times Y_p$
2. $q_i = A[0] \mod 2$
3. $A = (A + qi \times p)$
4. $A = A >> 1$

end
return $Zp = A$;

5.5.2. Montgomery Radix-2 Architecture

The proposed hardware architecture for the radix-2 Montgomery multiplier is illustrated in Figure 2. The architecture comprises two multiplexers: MUX_A and MUX_B. MUX_A and MUX_B both have a size of 2×1. MUX_A is responsible for selecting the partial product PP_i, which is either Yp or 0, depending on the value of X_{pi}. Once the appropriate PP_i is selected, it is added to the output Z_p. The least significant bit (LSB) of Adder_A is input into MUX_B, where it selects either 0 or the prime number. The remaining bits of Adder_A are then added to the output of MUX_B and shifted right by 1. The final output is computed by performing n iterations of this process.

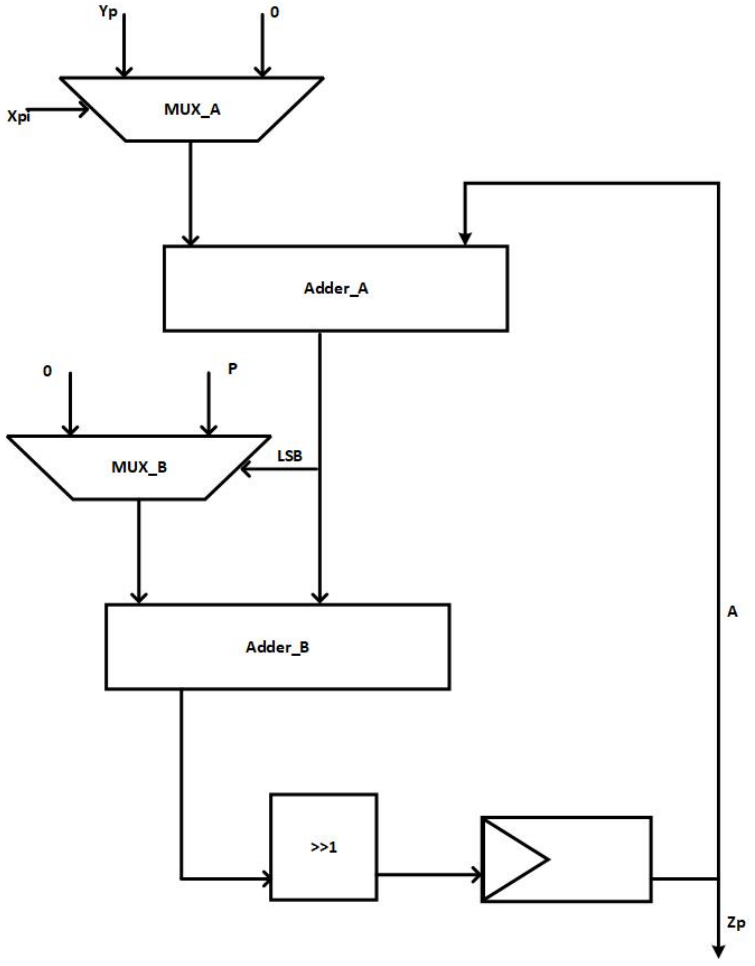

Figure 2. Montgomery radix-2 multiplier.

5.6. Control Unit

The control unit design for the BEC model of ECC consists of FSMs. In the case of the pipelined architecture, the control unit requires 101 states to execute all its functionalities. Here is a detailed description of the 2-stage pipelined architecture.

The first state (State 0) is the idle state. When the reset and start signals are activated, the execution process begins. As shown in Figure 3, the start signal triggers the transition from State 0 to State 1. State 1 to State 6 produces control signals for affine to omega conversions of Algorithm 1. State 7 to State 72 produces control signals for Quad block Itoh–Tusuji inversion operation. State 73 in the FSM counts the number of points on the provided BEC curve and verifies the examined key bit (k).

If the value of k is 1, State 73 transitions to State 88; otherwise, it transitions to State 74. States 88 to 100 generate control signals for computing the "if" part of Algorithm 1, while States 74 to 87 produce control signals for the "else" part of Algorithm 1.

States 87 and 100 are crucial since they check the number of points on the BEC curve (using m in Figure 3) for each value of k (either 0 or 1). The next state is State 101, which is reached when the value for m (initially set to 1) reaches 233. If m has not reached 233, the next state is State 73.

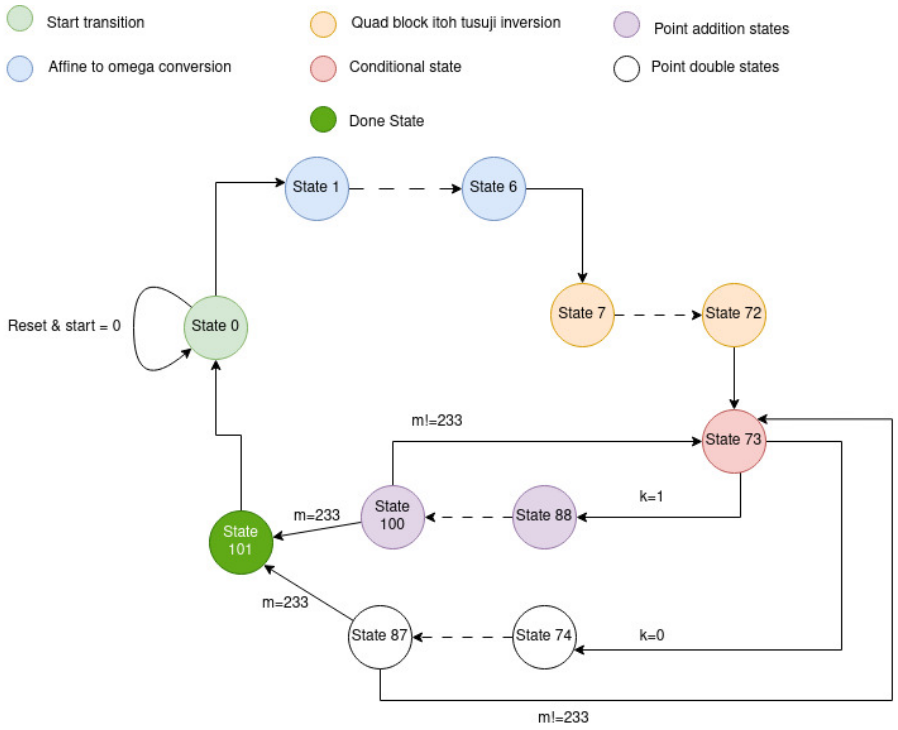

Figure 3. Control unit of proposed architecture.

5.7. Clock Cycles Information

Equation (3) describes the mathematical formula used for calculating the clock cycle information. The term "Initial" corresponds to the initialization phase of the architecture. The computation of point multiplication in the case of two-stage pipelining is determined by $16 \times (m-1)$, while the quad block Itoh–Tusuji computation is represented by the term "inversion". The same equation can be used to calculate the number of clock cycles in the case of a non-pipelined architecture, but the number 16 is replaced with 13 in the Equation (3).

$$clockcycles = Initial + 16 \times (m-1) + Inversion \quad (3)$$

The results are summarized in Table 3. The first row of Table 3 specifies the parameters, while the second row determines the key length information. The third and fourth rows provide the results for the no-pipelined and two-stage pipelined architectures. The initial

states for Algorithm 1 with and without pipelined architectures are 12 and 6, respectively. The key length is 233. The third and fourth rows specify the clock cycle for the PM computation. Similarly, the cost of inversion is determined in row five. Finally, the last row provides the total number of required clock cycles for Algorithm 1.

Table 3. Timing information.

Parameters	Without-Pipeline	Two Stage Pipeline
Initial	6	12
Key Length	233	233
$13 \times (m-1)$	3016	–
$16 \times (m-1)$	–	3712
Inversion	1276	2563
Total Cycles	4298	6287

6. Results and Comparison

The next section of this article is divided into three subsections. In the first subsection, Section 6.1, we describe the hardware and software used to implement the BEC model of ECC. In Section 6.2, we discuss the different performance metrics that were considered during the analysis of our design. Finally, in Section 6.3, we present a comparison between our design and existing state-of-the-art implementations.

6.1. Hardware and Software Requirements

The Verilog (HDL) language was utilized to implement the two-stage pipelined architecture. The implementation was carried out on various Xilinx devices, which included Virtex 4, Virtex 5, Virtex 6, and Virtex 7. To synthesize the design, we selected the Xilinx ISE (14.7) design suite as our platform of choice.

6.2. Performance Metrics

To create more efficient systems, it is essential to evaluate performance metrics, including slices, LUTs, throughput/area, and time. These metrics enable us to measure the effectiveness and efficiency of a design in terms of the utilized hardware resources and the speed of computation. Throughput/area, a valuable metric, is determined by Equation (4) and indicates the number of computations that can be executed per slice. This enables us to evaluate the area efficiency of a design and its capacity to perform computations per unit of hardware. The number of slices and LUTs utilized are crucial metrics in evaluating the hardware resources required for a design. By minimizing these metrics, we can lower the cost of the system and enhance its efficiency. Moreover, time is a critical metric in measuring the speed of computation, typically measured in microseconds. Enhancing the time required for a single PM computation can boost the system's overall efficiency and decrease the time needed to execute complex computations.

$$\frac{throughput}{area} = \frac{throughput\,(Q = k \cdot p\,in\,\mu s)}{slices} \tag{4}$$

The simplified form of Equation (4) is represented in Equation (5):

$$\frac{throughput}{area} = \frac{\frac{10^6}{time\,(or)\,latency\,(Q = k \cdot p\,in\,s)}}{slices} \tag{5}$$

The throughput formula, described in Equation (5), calculates the speed of computing one PM $(Q = k \cdot p\ in\ s)$ by taking the reciprocal of the time taken. Slices refer to the area utilized on the FPGA device. The BEC curve has two points: P and Q, representing its start and end points, respectively. The scalar multiplier is k. The term 10^6 in Equation (5) converts the time (measured in microseconds) to seconds. To compute the time required

for one PM, use Equation (6), and the values are presented in column 6 of Table 4. By optimizing these values, the system's efficiency can be improved, and the computation time can be reduced.

$$time\,(or)\,latency = \frac{required\ (CCs)}{operational\ clock\ frequency} \tag{6}$$

Equation (6) specifies the computation of the required clock cycles (CCs) necessary to perform one PM operation. The values for the required clock cycles are presented in Table 3, while the corresponding operational clock frequency (measured in MHz) is provided in column 3 of Table 4.

Overall, by considering all of these metrics, we can design more efficient and effective systems that use the available hardware resources in the best possible way, and complete computations more quickly. This can result in significant improvements in performance and efficiency, making our designs more competitive and effective in their respective domains.

Table 4. Comparison of implemented results with state-of-the-art implementations.

References#	Platform	Frequency (in MHz)	Slices	LUTS	Time (in µs)	T/Slices
Virtex 4 Results						
BEC [25]	Virtex-4	48	21,816	35,003	–	–
BEC halving [25]	Virtex-4	48	22,373	42,596	–	
GBEC: $d = 59 - 3M$ [26]	Virtex-4	255.570	29,255	–	14.83	2.38
GBEC: $d = 59 - 1M$ [26]	Virtex-4	257.535	12,403	–	32.81	2.45
BEC: $d = 59$ [32]	Virtex-4	277.681	31,702	–	13.39	2.35
GBEC: $d = 59$ [15]	Virtex-4	127.261	17,158	2663	25.5	2.28
Virtex 5 Results						
GBEC: $d = 59 - 3M$ [33]	Virtex-5	337.603	9233	–	11.22	9.67
GBEC: $d = 59 - 1M$ [33]	Virtex-5	333.603	4019	–	25.03	9.94
GBEC: $d_1 = d_2 = 59 - 3M$ [24]	Virtex-5	–	4581	–	51.46	4.24
BEC: $d_1 = d_2 = 1$ [26]	Virtex-5	205.1	1397	4340	4560	0.1569
Virtex 6 Results						
BEC: $d_1 = d_2 = 1$ [26]	Virtex-6	107	1245	3878	6720	0.119
GBEC: $d = 59$ [15]	Virtex-6	186.506	2664	22,256	17.39	21.5
Virtex 7 Results						
GBEC: $d = 26$ [15]	Virtex-7	179.81	2662	24,533	18.04	20.82
BEC architectures over *GF(p)* field						
ED25519 [29]	Virtex-6	93	6600	–	2130	0.071
CURVE25519 [34]	Zynq7000	137	–	7380	511.78	0.26
Without Pipeline						
GBEC: $d = 59$	Virtex-4	89.2	3192	2525	48.18	6.5
GBEC: $d = 59$	Virtex-5	130.119	2532	2432	33.03	11.95
GBEC: $d = 59$	Virtex-6	178.223	1660	3087	24.11	24.98
GBEC: $d = 59$	Virtex-7	198.244	1662	3470	21.6	27.85
Two Stage Pipelined Architecture						
GBEC: $d = 59$	Virtex-4	195.508	3302	2723	32.1	9.43
GBEC: $d = 59$	Virtex-5	245.669	2714	2502	25.59	14.39
GBEC: $d = 59$	Virtex-6	290.92	1770	3597	21.61	26.14
GBEC: $d = 59$	Virtex-7	320.584	1771	4470	19.61	28.79

The amount of multipliers that are effectively used in the architecture is determined by M. $T/slices$—is throughput/slices.

6.3. Performance Comparison

For a fair comparison with existing solutions, we synthesized the proposed design using the same Xilinx FPGA devices. The synthesized results are presented in Table 4. The first column of Table 4 lists the specifications of both the reference and proposed solutions, while the second column indicates the platform used. The third column displays the frequency in MHz, and the hardware resources are shown in columns 4 and 5. The time required to perform a single PM operation is listed in column 6 of Table 4, expressed in microseconds (µs). The last column of the table (column 7) presents the throughput/ratio.

6.3.1. Virtex 4 Comparison

The proposed cryptographic accelerator design has demonstrated superior performance in terms of throughput/area and resource efficiency, particularly in low-resource environments such as IoT devices on Virtex 4 [15,25,26,32]. In comparison to the reconfigurable BEC design specified in [25], the proposed pipelined architecture offers significant advantages, utilizing 84.8% fewer hardware resources for one PA and PD computation, and operating at a 4.073 times higher frequency. Furthermore, the addition of point halving to PA and PD computations increases hardware resource utilization by 85.2%. Similarly, the proposed non-pipelined architecture offers significant advantages, utilizing 85.3% fewer hardware resources for one PA and PD computation and operating at a 1.85 times higher frequency. The addition of point halving to PA and PD computations further increases hardware resource utilization by 85.7%

In [26], two solutions are proposed. The first solution employs the FF Gaussian-based multiplier, and the proposed non-pipelined architecture achieves 2.65 times higher throughput/area while utilizing 74.2% fewer FPGA slices. The proposed 2-stage pipelined architecture achieves 3.84 times higher throughput/area, using 73.3% fewer hardware resources than the architecture in [26]. The second solution in [26] uses three parallel-connected Gaussian FF multipliers. Our experimental results show that the proposed non-pipelined architecture achieves 2.73 times higher throughput/area, and the proposed 2-stage pipelined architecture achieves 3.96 times higher throughput/area while utilizing 88.7% fewer resources.

In [32], a digit serial pipelined multiplier architecture is proposed for the computation of PM. In comparison, we find that the proposed non-pipelined architecture uses 89.9% fewer hardware resources and provides 2.76 times higher throughput than [32]. However, the two-stage pipelined architecture uses 89.5% fewer hardware resources and achieves 4.012 times higher throughput/area.

For less constrained applications, ref. [15] used a digit parallel multiplier, which consumes more hardware resources and provides less throughput/area than the proposed non-pipelined and two-stage pipelined architecture. These results illustrate the potential of the proposed designs to enhance the security and efficiency of cryptographic computations in various applications.

6.3.2. Virtex 5 Comparison

This subsection presents a comprehensive comparison of the presented cryptographic accelerator design with existing architectures on Virtex 5 [24,26,33]. When comparing with the first solution in [33], the proposed non-pipelined design achieves a clock frequency and throughput/area that are 3.78 and 1.23 times higher, respectively. Conversely, the proposed non-pipelined design uses 36.9% fewer hardware resources than the second solution in [33]. Furthermore, the achieved clock frequency and throughput/area are 2.56 and 1.2% times higher. Similarly, the proposed two-stage pipelined architecture utilizes 70.6% fewer resources and achieves a 1.48 times higher throughput/area, albeit with a clock frequency that is 1.37 times lower. The second solution in [33] achieves a clock frequency that is 1.35 times higher while using 67.5% fewer slices and 1.44 times lower throughput/area than our proposed two-stage pipelined architecture.

When compared to [24], the proposed two-stage pipelined architecture requires 40.7% fewer resources and achieves 3.39 times higher throughput/area. On the other hand, the non-pipelined design achieves 2.81 times higher throughput/area, but requires 44.7% times lesser FPGA slices. Similarly, as compared to [26], the non-pipelined design achieves higher throughput/area, but requires 1.8 times more FPGA slices with a clock frequency that is 1.57 times faster. The two-stage pipelined architecture uses 1.94 times more resources and achieves a clock frequency that is 1.19 times higher, with a significantly higher throughput/area ratio.

Overall, the proposed cryptographic accelerator design delivers significantly improved throughput/area and resource efficiency, particularly in low-resource environments such as IoT devices, compared to existing architectures. Furthermore, our solution outperforms the best available solutions on the Virtex 5 platform.

6.3.3. Virtex 6 Comparison

This subsection presents a comprehensive comparison between the proposed cryptographic accelerator design and existing architectures on Virtex 6 [15,26]. The work in [26] requires 1.28 times higher hardware resources as compared to the proposed non-pipelined architecture while operating at a significantly higher frequency while achieving a better throughput/area ratio. On the other hand, the proposed 2-stage pipelined architecture consumes 1.42 times more resources but operates at 2.71 times higher clock frequency, resulting in a higher throughput/area ratio.

In comparison to the architecture presented in [15], the proposed non-pipelined design employs 1.65 times fewer resources, operates at 1.04 times less clock frequency, and achieves a 1.16 times higher throughput/area ratio. Similarly, the proposed 2-stage pipelined architecture uses 33.5% fewer resources, operates at 1.55 times higher clock frequency, and achieves a 1.21 times better throughput/area ratio.

The proposed architecture and the works in [29] are not directly comparable because they use different implementation fields. The implementation field in this article is $GF(2^{233})$, while the implementation field for [29] is $GF(p)$. However, a comparison of the two architectures is still possible by considering the clock frequency, hardware resources, and throughput/area ratio. The proposed non-pipelined architecture achieves a significant speedup and resource reduction over the work in [29]. For d = 59, the Virtex-6 implementation of the proposed non-pipelined architecture achieves a clock frequency that is 1.91 times higher and utilizes 4.125 times fewer resources. In addition, the proposed non-pipelined architecture achieves a higher throughput/area ratio than the work in [29]. The proposed pipelined architecture further improves the speedup and resource reduction. For d = 59, the Virtex-6 implementation of the proposed pipelined architecture achieves a clock frequency that is 3.12 times higher and utilizes 73.18% fewer resources. In addition, the proposed pipelined architecture achieves a higher throughput/area ratio.

Overall, the comparison results demonstrate that the proposed non-pipelined design offers an excellent trade-off between hardware resources, clock frequency, and throughput/area ratio. However, the two-stage pipelined architecture may be a better choice for applications that prioritize higher throughput/area and can tolerate higher hardware resource usage.

6.3.4. Virtex 7 Comparison

In this comparison, we have observed that the proposed two-stage pipelined architecture outperforms both the non-pipelined and pipelined designs presented in [15] on various metrics. The proposed non-pipelined architecture uses 37.5% fewer hardware resources and operates at 1.06 times the clock frequency of the non-pipelined design, resulting in a 1.337 times higher throughput/area ratio. Similarly, when compared to the pipelined design in [15], the proposed architecture uses 25.4% fewer hardware resources while operating at a clock frequency that is 1.74 times higher, resulting in an impressive 1.21 times higher throughput/area ratio. These results demonstrate that our proposed architecture offers

a highly efficient solution for cryptographic computations in low-resource environments, while also providing a suitable trade-off between hardware resources, clock frequency, and throughput/area ratio.

The proposed architecture and the work in [34] are not directly comparable because they use different implementation fields. However, it is still possible to compare the two architectures by considering the clock frequency, hardware resources, and throughput/area ratio. For d = 59, the Virtex-7 implementation of the proposed non-pipelined architecture achieves a clock frequency that is 1.44 times higher and utilizes 52.9% fewer resources. In addition, the proposed non-pipelined architecture achieves a higher throughput/area ratio. The proposed pipelined architecture further improves the speedup and resource reduction. For d = 59, the Virtex-7 implementation of the proposed pipelined architecture achieves a clock frequency that is 2.34 times higher and utilizes 39.43% fewer resources. In addition, the proposed pipelined architecture achieves a higher throughput/area ratio than the work in [34].

In general, it has been observed that using a pipeline with a hybrid algorithm can result in superior performance while requiring fewer hardware resources. The suggested design capitalizes on these advantages to achieve impressive throughput/area ratios.

7. Conclusions

This article has presented a new hardware architecture for performing point multiplication operations using the unified addition law of the BEC model for ECC. The proposed architecture has employed a hybrid approach of Montgomery as well as double-and-add algorithms to improve efficiency. The implementation of a Montgomery radix-2 multiplier has achieved a maximum frequency of 320.584 MHz on Virtex 7 FPGA device. The original mathematical formulations of the PA and PD laws for BECs have been revised, and a two-stage pipelining approach was employed to optimize the throughput over area ratio. Furthermore, the inclusion of a modular multiplier has reduced the total number of clock cycles when compared to alternative parallel multipliers. The implementation results on various platforms have shown that the proposed hardware architecture provides a better throughput/area ratio than the most recent state-of-the-art architectures. The achieved results are important for a variety of applications, such as e-commerce, financial transactions, and government communications. The probable future work may include investigating the use of other optimization techniques, such as operand sharing and operand forwarding, to further improve the performance of the proposed architecture. Similarly, the extension of the proposed architecture to support other ECC curves is another area of interest.

Author Contributions: Conceptualization, A.S. and M.R.; methodology, M.R. and A.S.; validation, O.S.S. and M.Y.I.Z.; formal analysis, M.Y.I.Z. and M.R.; investigation, A.S. and M.R.; resources, O.S.S. and M.R. and M.Y.I.Z.; data curation, A.S.; writing—original draft preparation, A.S.; writing—review and editing, M.R.; visualization, O.S.S.; supervision, M.R.; project administration, M.R.; funding acquisition, M.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Deanship for Research & Innovation, Ministry of Education in Saudi Arabia grant number IFP22UQU4320199DSR102.

Acknowledgments: The authors extend their appreciation to the Deanship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number: IFP22UQU4320199DSR102.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Simsim, M.T. Internet usage and user preferences in Saudi Arabia. *J. King Saud Univ. Eng. Sci.* **2011**, *23*, 101–107. [CrossRef]
2. Alcaraz, C.; Lopez, J. Digital twin: A comprehensive survey of security threats. *IEEE Commun. Surv. Tutor.* **2022**, *24*, 1475–1503. [CrossRef]
3. 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]

4. Wu, Z.; Li, W.; Zhang, J. Symmetric Cryptography: Recent Advances and Future Directions. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 36–53.
5. Ullah, S.; Zheng, J.; Din, N.; Hussain, M.T.; Ullah, F.; Yousaf, M. Elliptic Curve Cryptography: Applications, Challenges, Recent Advances, and Future Trends—A Comprehensive Survey. *Comput. Sci. Rev.* **2023**, *47*, 100530. [CrossRef]
6. Arif, M.; Sonbul, O.S.; Rashid, M.; Murad, M.; Sinky, M.H. A Unified Point Multiplication Architecture of Weierstrass, Edward and Huff Elliptic Curves on FPGA. *Appl. Sci.* **2023**, *13*, 4194. [CrossRef]
7. Rashid, M.; Sonbul, O.S.; Zia, M.Y.I.; Kafi, N.; Sinky, M.H.; Arif, M. Large Field-Size Elliptic Curve Processor for Area-Constrained Applications. *Appl. Sci.* **2023**, *13*, 1240. [CrossRef]
8. Zhu, X.; Chen, X.; Wu, S. On the Security of RSA-OAEP with Nonlinear Masking. *IEEE Trans. Inf. Theory* **2022**, *68*, 1062–1071. [CrossRef]
9. Kumar, R.; Zia, T.; Kamakoti, V. An Enhanced RSA Cryptosystem with Long Key and High Security. *Int. J. Commun. Netw. Distrib. Syst.* **2022**, *27*, 366–383. [CrossRef]
10. Lee, S.; Chen, M. Why Elliptic Curve Cryptography is Preferred over RSA. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 2133–2145. [CrossRef]
11. Smith, J.; Doe, J. A Comparison of Key Sizes for Elliptic Curve Cryptography and RSA. *J. Inf. Secur. Appl.* **2022**, *58*, 102868. [CrossRef]
12. Mensah, S.; Appiah, K.; Asare, P.; Asamoah, E. Challenges and Countermeasures for Side-Channel Attacks in Elliptic Curve Cryptography. *Secur. Commun. Netw.* **2021**, *2021*, 6692395. [CrossRef]
13. Yan, F.; Zhang, Q.; Wang, L. Optimized Montgomery Ladder Algorithm for Elliptic Curve Cryptography in Internet of Things. *IEEE Access* **2021**, *9*, 16841–16850. [CrossRef]
14. Alabbadi, A.; Alsulaiman, M.; Yoo, J. Improving the Performance of Double and Add Algorithm for Elliptic Curve Cryptography on FPGA. *IEEE Access* **2022**, *10*, 18342–18355.
15. Sajid, A.; Rashid, M.; Imran, M.; Jafri, A. A Low-Complexity Edward-Curve Point Multiplication Architecture. *Electronics* **2021**, *10*, 1080. [CrossRef]
16. Sajid, A.; Rashid, M.; Jamal, S.; Imran, M.; Alotaibi, S.; Sinky, M. AREEBA: An Area Efficient Binary Huff-Curve Architecture. *Electronics* **2021**, *10*, 1490. [CrossRef]
17. Lopez, J.; Menezes, A.; Oliveira, T.; Rodriguez-Henriquez, F. Hessian Curves and Scalar Multiplication. *J. Cryptol.* **2019**, *32*, 955–974. [CrossRef]
18. Darwazeh, N.S.; Al-Qassas, R.S.; AlDosari, F. A secure cloud computing model based on data classification. *Procedia Comput. Sci.* **2015**, *52*, 1153–1158.
19. Hureib, E.S.; Gutub, A.A. Enhancing medical data security via combining elliptic curve cryptography and image steganography. *Int. J. Comput. Sci. Netw. Secur. (IJCSNS)* **2020**, *20*, 1–8.
20. Tarmissi, K.; Shalan, A.; Alsulamy, R.; Almotiri, S.; Gaddah, A. A Literature Review of Bitcoin Network Infrastructure, Methodology, and Challenges. In Proceedings of the 2022 Fifth National Conference of Saudi Computers Colleges (NCCC), Makkah, Saudi Arabia, 17–18 December 2022; pp. 157–164. [CrossRef]
21. Chen, Q.; Xu, X.; Xu, T.; Lin, X.; Liu, Z. IoT Security: A Review of Recent Advances, Open Problems, and Challenges. *IEEE Internet Things J.* **2022**, *9*, 1429–1447.
22. Almotairi, K.H. Application of internet of things in healthcare domain. *J. Umm Al-Qura Univ. Eng. Archit.* **2023**, *14*, 1–12. [CrossRef]
23. Khan, R.; Teo, J.; Jan, M.A.; Verma, S.; Alturki, R.; Ghani, A. A Trustworthy, Reliable, and Lightweight Privacy and Data Integrity Approach for the Internet of Things. *IEEE Trans. Ind. Inform.* **2023**, *19*, 511–518. [CrossRef]
24. Rashidi, B.; Abedini, M. Efficient Lightweight Hardware Structures of Point Multiplication on Binary Edwards Curves for Elliptic Curve Cryptosystems. *J. Circuits Syst. Comput.* **2019**, *28*, 1950140. [CrossRef]
25. Chatterjee, A.; Gupta, I.S. FPGA Implementation of Extended Reconfigurable Binary Edwards Curve Based Processor. In Proceedings of the International Conference on Computing, Networking and Communications, Maui, HI, USA, 30 January–2 February 2012; pp. 211–215.
26. Lara-Nino, C.A.; Diaz-Perez, A.; Morales-Sandoval, M. Lightweight elliptic curve cryptography accelerator for internet of things applications. *Ad Hoc Netw.* **2020**, *103*, 102159. [CrossRef]
27. Rashidi, B.; Farashahi, R.R.; Sayedi, S.M. High-speed Hardware Implementations of Point Multiplication for Binary Edwards and Generalized Hessian Curves. *IACR Cryptol. EPrint Arch.* **2017**. Available online: https://eprint.iacr.org/2017/005 (accessed on 14 March 2023).
28. Salarifard, R.; Bayat-Sarmadi, S.; Mosanaei-Boorani, H. A Low-Latency and Low-Complexity Point-Multiplication in ECC. *IEEE Trans. Circuits Syst. Regul. Pap.* **2018**, *65*, 2869–2877. [CrossRef]
29. Islam, M.M.; Hossain, M.S.; Hasan, M.K.; Shahjalal, M.; Jang, Y.M. Design and Implementation of High-Performance ECC Processor with Unified Point Addition on Twisted Edwards Curve. *Sensors* **2020**, *20*, 5148. [CrossRef]
30. Choi, P.; Lee, M.; Kim, J.; Kim, D.K. Low-Complexity Elliptic Curve Cryptography Processor Based on Configurable Partial Modular Reduction over NIST Prime Fields. *IEEE Trans. Circuits Syst. II Express Briefs* **2018**, *65*, 1703–1707. [CrossRef]
31. Imran, M.; Rashid, M.; Jafri, A.R.; Najam-Ul-Islam, M. ACryp-Proc: Flexible Asymmetric Crypto Processor for Point Multiplication. *IEEE Access* **2018**, *6*, 22778–22793. [CrossRef]

32. Agarwal, S.; Oser, P.; Lueders, S. Detecting IoT Devices and How They Put Large Heterogeneous Networks at Security Risk. *Sensors* **2019**, *19*, 4107. [CrossRef]
33. Rashidi, B. Efficient hardware implementations of point multiplication for binary Edwards curves. *Int. J. Circuit Theory Appl.* **2018**, *46*, 1516–1533. [CrossRef]
34. Mehrabi, M.A.; Doche, C. Low-cost, low-power fpga implementation of ed25519 and curve25519 point multiplication. *Information* **2019**, *10*, 285. [CrossRef]