Journal of Engineering
journal homepage: www.joe.uobaghdad.edu.iq
Number 9 Volume 26 September 2020

Electrical, Electronics and communications, and Computer Engineering

Performance Evaluation of Scalar Multiplication in Elliptic Curve Cryptography Implementation using Different Multipliers Over Binary Field GF (2^{233})

Alaa Mohammed. Abdul-Hadi
Dr
Computer Engineering Dept.
Iraq, Baghdad
alaa.m.abdulhadi@coeng.uobaghdad.edu.iq

Yousra Abdul-Sahib S.aldeen
Dr
College of Science for Women
Iraq, Baghdad
yousraalkaalesi@gmail.com

Firas Ghanim Tawfeeq
Master Student
Computer Engineering Dept.
Iraq, Baghdad
f.ghanim1205@coeng.uobaghdad.edu.iq

ABSTRACT

This paper presents a point multiplication processor over the binary field GF (2^{233}) with internal registers integrated within the point-addition architecture to enhance the Performance Index (PI) of scalar multiplication. The proposed design uses one of two types of finite field multipliers, either the Montgomery multiplier or the interleaved multiplier supported by the additional layer of internal registers. Lopez Dahab coordinates are used for the computation of point multiplication on Koblitz Curve (K-233bit). In contrast, the metric used for comparison of the implementations of the design on different types of FPGA platforms is the Performance Index.
The first approach attains a performance index of approximately 0.217610202 when its realization is over Virtex-6 (6vlx130tff1156-3). It uses an interleaved multiplier with 3077 register slices, 4064 lookup tables (LUTs), 2837 flip-flops (FFs) at a maximum frequency of 221.6Mhz. This makes it more suitable for high-frequency applications. The second approach, which uses the Montgomery multiplier, produces a PI of approximately 0.2228157 when its implementation is on Virtex-4 (6vlx130tff1156-3). This approach utilizes 3543 slices, 2985 LUTs, 3691 FFs at a maximum frequency of 190.47MHz. Thus, it is found that the implementation of the second approach on Virtex-4 is more suitable for applications with a low frequency of about 86.4Mhz and a total number of slices of about 12305.

Keywords: Interleaved-multiplier, Montgomery-multiplier, ECC, Performance index, binary field GF (2^{233}).

*Corresponding author
Peer review under the responsibility of University of Baghdad.
https://doi.org/10.31026/j.eng.2020.09.04

Article received: 27/1/2020
Article accepted:18/5/2020
Article published:1/9/2020

تقييم أداء وحدة الضرب العددي في تشفير المنحني الأهليجي المنفذ بإستخدام دوائر ضرب رقمية مختلفة على الحقل الثنائي ($GF(2^{233})$

علاء محمد عبدالهادي
دكتور
قسم هندسة الحاسبات

يسرى عبد الصاحب سيف الدين
دكتور
كلية العلوم للبنات

فراس غانم توفيق
طالب ماجستير
قسم هندسة الحاسبات

الخلاصة

هذه الورقة تقدم معالج الضرب النقطي في الحقل الثنائي أوما تعرف بحقول غالوا ($GF2^{233}$) معزز بمسجلات توضع داخل وحدة الجمع النقطي لتحسين مؤشر الاداء (PI). التصميم المقترح يستخدم أحد نوعين من دوائر الضرب الرقمية في المجال المحدود ، إما دائرة ضرب Montgomery أو دائرة ضرب Interleaved. حيث تم إستخدام إحداثيات لوبيز دهب لحساب الضرب النقطي على منحنى كوبليتز (K-233bit). كما تم إستخدام مؤشر الأداء (PI) لغرض المقارنة بين تنفيذات التصميم المقترح على أنواع مختلفة من منصات FPGA.

تم الحصول من الطريقة الأولى على مؤشر أداء يبلغ حوالي 0.2176102. عندما تم التنفيذ على Virtex-6 (6vlx130tff1156-3) باستخدام دائرة الضرب (Interleaved) مع 3077 شريحة تسجيل و 4064 جداول بحث (LUTs) و 2837 (FFs) flip-flops كحد أقصى بتردد 221.6 ميجا هرتز ، هذا يجعله أكثرملائمة للاستخدام في تطبيقات التردد العالي. اما في الطريقة الثانية التي استخدم فيها مضاعف Montgomery فقد تم الحصول على مؤشر أداء (PI) مقداره 0.2281576 عند تنفيذه على (6vlx130tff1156-3) Virtex-4 حيث تم إستخدام 3543 شريحة و 2985 LUTs و 3691 FFs بتردد مقداره 190.47 ميجاهرتز. مما تقدم وجد أن تنفيذ الطريقة الثانية على Virtex-4 هو أكثر ملاءمة للتطبيقات التي تعمل بتردد منخفض يصل الى 86.4 ميجاهرتز ومجموعة شرائح تصل الى 12305.

الكلمات الرئيسية : دائرة ضرب Interleaved , دائرة ضرب منتغمري , ECC ,مؤشر الاداء, الحقل الثنائي (2^{233})

1. INTRODUCTION

Elliptic Curve Cryptography (ECC) is a type of asymmetric key **(Jwad, Abdulaah, and Effing, 2012)** cryptography that provides higher security than Rivest–Shamir–Adleman (RSA) for a smaller key size. A short key is a proper choice for hardware implementations of ECC, especially in devices with restricted resources as they require less area and processing time **(Kilts, 2006), (Kawther E. Abdullah, 2018).** Hardware implementations of cryptosystems produce systems with higher speeds and better security than software implementations. Point Multiplication (PM) is the heartbeat of ECC. Different projective coordinates can be used for point representation, but this work uses the Lopez Dahab coordinate system to skip the inversion process that consumes lots of resources **(Bilal and Rajaram, 2010)**. The efficiency of the high-performance hardware implementation of scalar multiplication depends on the polynomial representation. Both performance metrics, time, and area are desirable to be considered during the design. Still, incompatible features, as in some projects, can deliver a high speed within a compacted area while others attain lower area and speed. Consequently, hardware implementations require the consideration of speed and area parameters **(Strukov, 2006).** Different architectures are adopted to design and realize a multiplier unit such as a Montgomery, Karatsuba, Mastrovito, bit-parallel, and digit serial**.** This work considers two types of binary fields for GF (2^{233}) multipliers: Interleaved and Montgomery.

This paper aims to enhance the performance index of PM by adding internal registers within the data path of point multiplication, and the proposed designs of PM are implemented on different FPGA platforms. The FPGA which are appropriate for intensive computations **(Hassan, 2010)**.

The rest of this paper presents previous work in this field, the mathematical background of finite field and elliptic curves, simulation, implementation of the proposed design, followed by the results and discussion then finalized by the conclusion.

2.RELATED WORK

A proposed design by **(Urbano-Molano, Trujillo-Olaya, and Velasco-Medina, 2013),** presented parallel multiplication and bit-serial multipliers then obtained an execution time of 0.025µs and 1.62µs, respectively, with a value of k equal to 9**. (Fournaris, Dimopoulos, and Koufopavlou, 2017)** presented a strategy for digit serial multiplier based on binary Edwards curve scalar multiplier architectures. It relied on the use of GF (2^k) digit serial multiplication with a balance in speed and consumption resources in addition to parallelism for distributing GF (2^k) operations while keeping a high level of usability of units in each layer.

The design of point multiplication over the binary field GF (2^{233}) is presented by **(Kadu and Adane, 2018)** as a secured curve based on the recommendations of NIST.

Performances obtained from this design were assessed by comparing them with the Karatsuba-based point multiplier for area and delay. The results show that the Vedic multiplier occupied 22% less area on FPGA and caused 12% less delay than the Karatsuba-based scalar multiplier. The proposed design was coded using Verilog HDL and simulated and synthesized on Virtex-6.

A design by **(Imran, Rashid, and Shafi, 2018)** presented a bit-parallel hybrid Karatsuba multiplier in the finite field layer**.** This design attained the number of slices, time of PM, and PI $= (slice * (K \cdot p))/10^6$]. where $\mathrm{k \cdot P}$ represent the time point mutliplication

(1) On Virtex-4, the result was (6884 slices, 53.5µs, 0.368); (2) On Virtex-5, the obtained results were (3636 slices, 32.3µs, 0.117); (3) On Virtex-6, the proposed design attained (3144 slices, 26.9µs, 0.084); (4) On the newer Virtex-7, it attained 3657 slices, 25.3µs, 0.092. Finally, the proposed Elliptic Curve Processor (ECP) outperforms on Virtex-6 in terms of both area and area performance index and is approximately 0.084 compared to the most relevant work.

The architecture of the proposed design by **(Rashidi, 2018)** was built on Virtex-5 XC5VLX110 and Virtex-4 XC4VLX100 FPGAs to achieve two fields, $F2^{163}$ and $F2^{233}$. The results show enhancement in execution time and area when compared to previous work.

Finally, the proposed design of a coprocessor **by (Parrilla *et al.*, 2019)** allowed the acceleration of secure services that can be applied in the next generations of FPGA. Thus, permitting to host in the same chip, a secure web or database server, and the cryptographic processor. This coprocessor provided an improvement over other hardware implementations in terms of area, performance, and scalability.

The purpose of the paper is to enhance the PI of scalar multiplication by adding a layer of registers, then compare the outcome PI among different FGPAs.

3. MATHEMATICAL BACKGROUND

3.1 Finite Field

A field with a finite number of elements is called a Finite Field F_q. It is used in cryptography, where $q=2^m$, to implement software or hardware with fast performance. The elements in a binary representation can be presented in a binary representation degree less than m, where $A(x)=\sum_{i=0}^{m-1} a_i x^i$; the arithmetic operations in a binary field are reduced using an irreducible polynomial that have an m degree. A polynomial with degree m can represented in the following formula:

$$f(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + a_{m-3}x^{m-3} \dots a_1x^1 + a_0x^0 \quad (1)$$

where x^i is called the ith terms of polynomial, and a_i represents the coefficient and m represents the length of key size.

For example, an 8-bit word is represented by a polynomial as follows in **Fig.1**:

1	**0**	**0**	**1**	**1**	**0**	**0**	**1**
↓	↓	↓	↓	↓	↓	↓	↓
$\mathbf{1x^7}$	$\mathbf{0x^6}$	$\mathbf{0x^5}$	$\mathbf{1x^4}$	$\mathbf{1x^3}$	$\mathbf{0x^2}$	$\mathbf{0x^1}$	$\mathbf{1x^0}$
First Simplification : $\mathbf{1x^7 + 1x^4 + 1x^3 + 1x^0}$							
Second Simplification: $\mathbf{x^7 + x^4 + x^3 + 1}$							

Figure 1. Polynomial representation of 8bit.

It is clear from **Fig.1** that the term of 0 coefficient is omitted; moreover, $\boldsymbol{x^0}$ is 1.

3.2 Arithmetic Operation on Polynomials

3.2.1 Addition of Polynomials

Adding two polynomials elements C(x) and d(x) requires a bitwise exclusive-or. For example, if C(x) and D(x) are two polynomials, then $C(x) = x^5 + x^2 + x$ *and* $D(x) = x^3 + x^2 + x$ in GF (2^8), so $E(x) = C(x) \oplus$ D(x)

$$0x^7 + 0x^6 + 1x^5 + 0x^4 + 0x^3 + 1x^2 + 1x^1 + 0x^0$$

$$0x^7 + 0x^6 + 0x^5 + 0x^4 + 1x^3 + 1x^2 + 0x^1 + 1x^0 \oplus$$

$$0x^7 + 0x^6 + 1x^5 + 0x^4 + 1x^3 + 0x^2 + 1x^1 + 1x^0 \rightarrow x^5 + x^3 + x + 1$$

Also, there is a simple way to add two polynomials in the field – by deleting common terms and retaining the uncommon terms.

3.2.1.1 Multiplication

This refers to multiplying two polynomials C(x) and D(x) based on normal multiplication and polynomial reduction f(x) and has a specific value based on the curve type.

3.2.1.2 Squaring

The squaring of polynomial $C(x)^2$ is too cheap, as it can be accomplished by inserting zero into the bit vector(Hankerson, 2004).

3.2.1.3 division

The division of two polynomials can be accomplished by dividing the polynomial on modulo f(x) and keeping the remainders, for example, the division of polynomial with degree 12 on modulo with 8 degrees, as shown below:

$$\begin{array}{r|l} & x^4+1 \\ \hline x^8+x^4+x^3+x+1 & x^{12}+x^7+x^2 \\ & x^{12}+x^8+x^7+x^5+x^4 \\ \cline{2-2} & x^8+x^5+x^4+x^2 \\ & x^8+x^4+x^3+x+1 \\ \cline{2-2} & Remainder: x^5+x^3+x^2+x+1 \end{array}$$

3.3 Elliptic Curve over GF (2^m).

The Elliptic curve, from a mathematical aspect, is a cubic equation in the standard form. Eq. (2) defines the elliptic curve over the binary field GF (2^m); the curve is set with points, and each point located on the Elliptic curve is represented by the x and y coordinates when using the Affine coordinate projective. The values of a and b in Eq.(1) specifies the shape of the curve, while $b \neq 0$ f(x) represents an irreducible polynomial.

$$y^2 + xy = x^3 + ax^2 + b \; mod \; f(x) \tag{2}$$

Operations in the Elliptic curve have a hierarchy model and contain four layers. Layer one represents the finite field arithmetic operations such as multiplication, addition, division, and inversion. Layer 2 consists of two main components: point addition and point doubling. Point multiplication (scalar multiplication) is layer 3 in layer 4 lie security schemes such as Elliptic Curve Digital Signature Algorithm (ECDSA) and Elliptic Curve Diffe-Halmen (ECDH). Different kinds of elliptic curves are available. This work is based on the Koblitz curve with on field 233 and specifications mentioned by NIST. Therefore, Eq. (3) is used instead of Eq. (2), which is denoted by E_0.

$$E_0 = y^2 + xy = x^3 + \; b \; mod \; f(x) \tag{3}$$

Where b = 1, a=0, and $f(x) = x^{233} + \; x^{74} + 1$

The Koblitz curve is attractive because of its advantages in computational aspects. These advantages lie in using Frobenius endomorphism (φ), and the point P (x, y) can be mapped such that:

$$\varphi: (x, y) \rightarrow (\, x^2, \; y^2) \; and \; \vartheta \; \rightarrow \; \vartheta \tag{4}$$

Clearly, the Frobenius endomorphism is very cheap: two or three square operations are required depending on the objective coordinates. Using Frobenius endomorphism instead of point doubling it, which is not a straightforward operation, it requisites a manipulation of the value of k.

$$\mu\varphi(p) - \varphi^2(P) = 2P \; where \; \mu = (-1)^{1-a} \tag{5}$$

Thus, φ can be realized as a complex number τ, satisfying $\mu\tau - \; \tau^2 \; = 2$ that is $\tau = (\mu + \sqrt{-7})/2$. If k is given in a $\tau - adic$ representation as $k = \sum_{i=0}^{l-1} k_i \tau^i$. Then, to apply fast Frobenius endomorphism, k must be converted to $\tau - adic$.

3.4 Multipliers over Finite Field.

3.4.1 Interleaved Multiplier

An easy model of multiplication over the finite field is the interleaved multiplier. The principal work of this multiplier is based on the shift and add algorithm, and the products of c(x) and d(x) is E(x) = c(x) d(x) mod f(x), as shown in **Fig.2.**

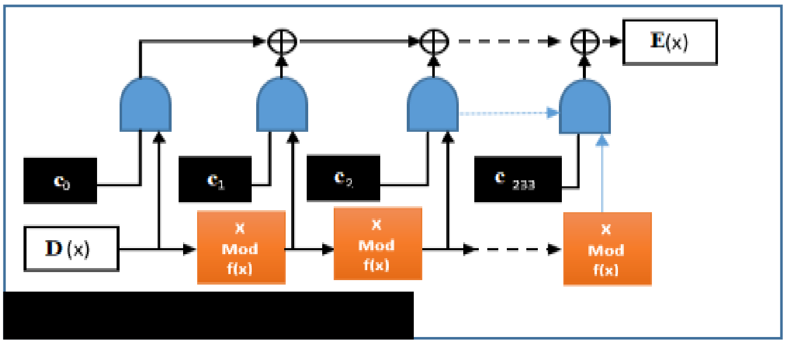

Figure 2. Interleaved multiplier for 233 field ,
(J. DESCHAMPS, J. IMANA, 2009).

3.4.2 Montgomery Multiplier

The Montgomery multiplier is a sequential multiplier model, and the products of the two polynomials c(x) and d(x) are defined in Eq. (6).

$$E(x) = c(x)d(x)\ M(x)^{-1}\ mod\ f(x) \tag{6}$$

where M(x) is a constant element in the field and gcd (M(x), f(x)) = 1; one can find two polynomials M $(x)^{-1}$ and f $(x)^{-1}$ so that

$$M(x)M(x)^{-1} + f(x)f(x)^{-1} = 1 \tag{7}$$

where $M(x)^{-1}$ is the inverse of M(x) modulo f(x). The two polynomials can be computed with the Extended Euclidean Algorithm (EEA). Therefore, the Montgomery multiplication over GF (2^m) can be computed using algorithm 1 and the data-path, as shown in **Fig. 3.**

Montgomery Algorithm.1
Step 1: Input Polynomials C(x), D(x), M(x), $F(x)^{-1}$ **Step 2:** T(x)=C(x)D(x). **Step 3:** Temp(x) = T(x) $F(x)^{-1}$ mod M(x) **Step 4:** E(x) = [T(x) + Temp(x) F(x)]/M(x) **Output:** E(x) = C(x)D(x) $M(x)^{-1}$ mod F(x)

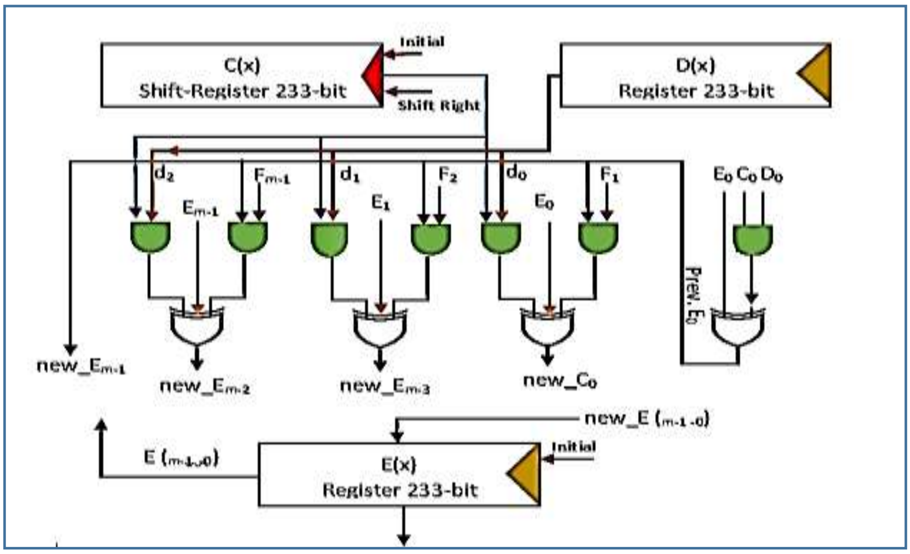

Figure 3. Internal architecture multiplier.
(J. DESCHAMPS , J. IMANA, 2009).

3.5 Squaring over Finite Field.

Classic squaring

In the classic squaring of polynomials E(x), inserting a zero value in bit vector is all that is required for getting $E(x)^2$. There is another method for squaring a polynomial, i.e., by applying the classic multiplication $E(x)^2=E(x)\,E(x) \bmod f(x)$.

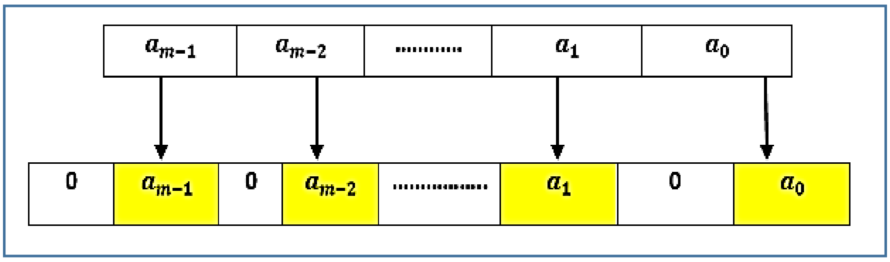

Figure 4. Polynomial squaring (Hossain, Saeedi and Kong, 2016).

3.6 Koblitz Point operations

3.6.1 Koblitz Point addition

This refers to adding two points P (x_1, y_1) and Q (x_2, y_2) on the Koblitz curve (E_0: $y^2 + xy = x^3 + 1$). Three arithmetic units are used for point addition: a multiplier, division unit, and squaring unit. The inversion component is not necessary for point operations when using the Koblitz curve. From a computational aspect, the third point R (X_3, Y_3) can be calculated using Eq. (8) and Eq. (9).

$$x_3 = z^2 + s + x_2 + x_1 + a \quad (8)$$

$$y_3 = z(x_1 + x_3) + x_3 + y_1 \quad , where\ z = \frac{y_2+y_1}{x_2+x_1} \tag{9}$$

Koblitz's point addition consists of two squaring components and one interleaved multipliers and a reduction component and binary division.

3.6.2 Koblitz point multiplication

All the points on the elliptic curve, including the infinity points, form a finite communicative group in point addition and point doubling. If there is a Generation point on the curve called P, and there is a positive number k, then the Q can be calculated as follows:

$$Q=k *P \rightarrow P+P+P.......+P \tag{10}$$

Algorithm 2: Lopez-Dahab-Algorithm
Input: **P=(**x_p, y_p) ∈ GF(2^m), Where K is a positive integer. K ← (k_{i-j},….,K_1, K_0) **Output:** K.P=(x_q, y_q) **Step1:Affine to Lopez Dahab Conversion** **1) S_2=(X_1)^2 2) X_2(S_2)^2 3) X_2(X_2+b)** **Step 2: Point Multiplication (PM)** *For int i=j-2 → 0 do* **1)W_1**=(X_1S_2) **2)W_2**=(X_2Z_1) **3) W_3**=(X_1Z_1) **4) T_3**=(Z_1) **5)T_3**=(T_3)^2 **6)Z_2**=(W_1+W_2) **7) S_2**=(S_2)^2 **8) Z_1**=(W_3)^2 **9)W_1**=(W_1W_2) **10)W_2=(X_PS_2)** **11)W_3=(bT_3)** **12) T_3**=(X_1)^2 **13) T_3**= (T_3)^2 **14)**X_2=(W_1+W_2) **15)X_1=(W_3+T_3)** **Step 3:**if (i = 0 & Ki = 1), *then* swap (X1, X2) *and* (Z1, S2) *end if end for* 1) **W_1**=Inv(Z_1) 2) **W_2**=Inv(S_2) 3)**W_3**=Inv(X_P) 4) **T_1**=(X_1W_1) 5)**W_1**=W_2(X_2W_2) 6) **T_3**=Inv(X_P) 7)**T_3**=T_3+Y_P 8) **W_1**=(X_P+T_1) 9) **W_2**=(X_P+V_2) 10**) W_1**=(W_1W_3**)** **11)W_2**=(W_1W_2) 12**)W_2**=(W_2+T_3) 13)**W_2**=(W_1W_2) 14**) T_2=(**W_2+Y_P) **Return: K. p= (x_q, y_q)** = (T_1, T_2)

The equation is called scalar multiplication or point multiplication. It is clear that the point multiplication is computed by repeating the adding and doubling of points, which absolutely depends on the components of the finite field arithmetic, such as polynomial multiplication, addition, inversion, and division.

The inversion module consumes more resources; therefore, most of the design uses projective coordinates for the representation of the point. Many projective coordinates exist, such as Affine, Lopez Dahap (LD), and Jacobean. The work in this paper is based on LD as the projective coordinates are shown in Algorithm3, and by using these types of coordinates, no inversion is used during the operation over a finite field, which quickens the point operation and consumes a few resources such as power, area, and low latency.

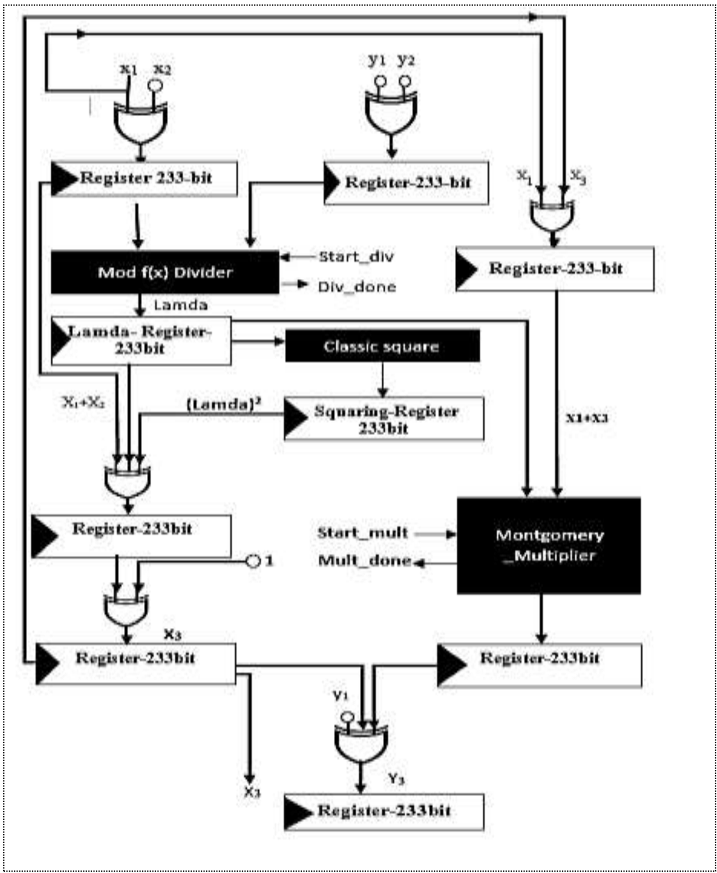

Figure 5. Data-path of point addition using Montgomery Multiplier (2^{233}).

3.7 Scalar Multiplication

Since the point multiplication is based on the Koblitz point addition, which consists of two squaring components, i.e., the interleaved multiplier, reduction component, and binary division algorithm. The proposed system can be defined over different scenarios to build the proposed elliptic curve point multiplication booster with internal registers on different platforms (Virtex-4, Virtex-5, Virtex-6, and Virtex-7), as shown below:

1. Building ECP with interleaved multiplier and classic squaring on different platforms.

2. Building ECP with Montgomery multiplier and classic squaring on different platforms.

Fig.5 illustrates the data-path of the Koblitz point addition over the 233-bit field and the proposed design using internal registers. These registers are used to store data after each operation.

4. RESULTS

The proposed design of scalar multiplication is based on point addition component. So, in order to get K.P, point P is added k-times to obtain the result. A strategy for architectural timing enhancement is to build intermediate layers of registers to the critical-path. This technique is used in pipeline design when latency, due to a few additional clock-cycles, does not affect specifications of the design. The throughput of the circuit is obtained using Eq. (11).

$$Throughput = \frac{frequency * number\ of\ bits}{cycles\ numbers} \tag{11}$$

To compare with previous work, PI is used instead of the throughput indicator, as PI is fairer to use for comparison among different platforms.

4.1 Simulation of the ECP for scenario 1

Fig.6 shows the time required for point multiplication, which is approximately 21.625us. It also shows the coordinates of the generation point (Xp, Yp) in Hexadecimal, as follows:

Xp:**17232ba853a7e731af129f22ff4149563a419c26bf50a4c9d6eefad6126**

Yp:**1db537dece819b7f70f555a67c427a8cd9bf18aeb9b56e0c11056fae6a3**

The number of registers, lookup tables, and flip-flops after synthesis are shown in **Fig.7.**

1.Simulation on Virtex-4

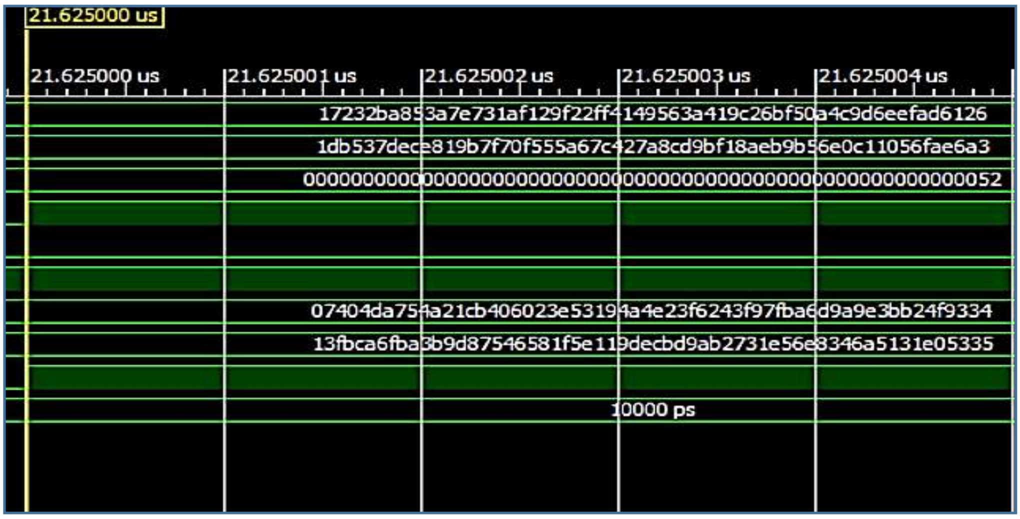

Figure 6. Simulation of ECP using interleaved multiplier.

```
Device utilization summary:
---------------------------

Selected Device : 4vlx80ff1148-12

 Number of Slices:                    3193  out of  35840     8%
 Number of Slice Flip Flops:          3772  out of  71680     5%
 Number of 4 input LUTs:              5340  out of  71680     7%
 Number of IOs:                        474
 Number of bonded IOBs:                474  out of    768    61%
 Number of GCLKs:                        1  out of     32     3%
```

Figure 7. Number of slices after synthesis.

Fig.8 shows the RTL schematic of scalar multiplication using interleaved multiplier with finite field GF (2^{233}) bits. The structure consists of three components, a Koblitz point addition, and two classic-squaring. The Point-addition in this scenario is composed of three components as follows: binary division, Interleaved multiplier, classic squaring, and some other components such as XOR gates for bitwise addition.

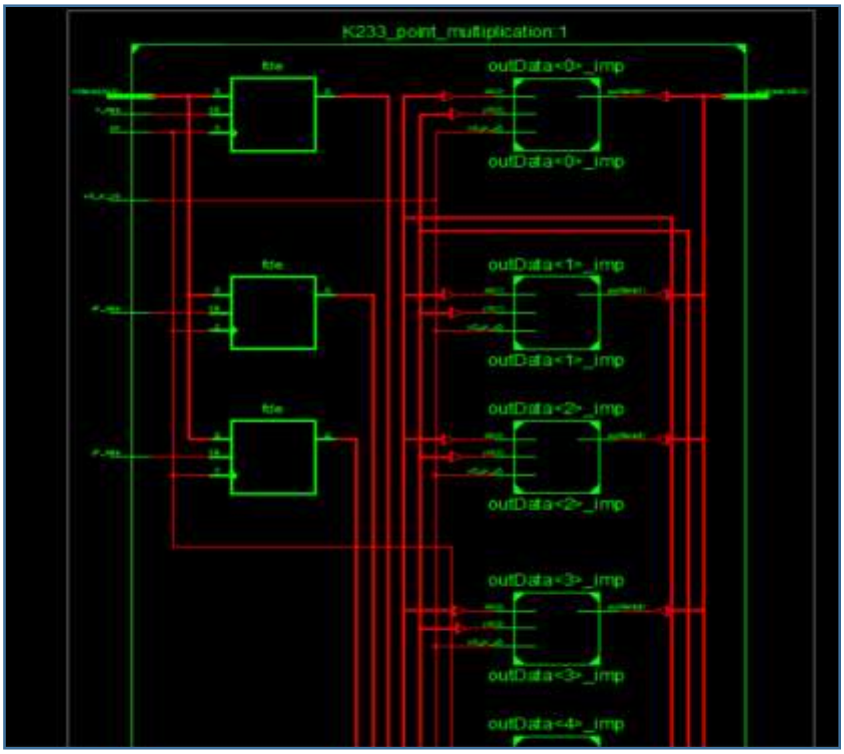

Figure 8. RTL schematic of ECP using interleaved multiplier.

2. Simulation on Virtex-5:

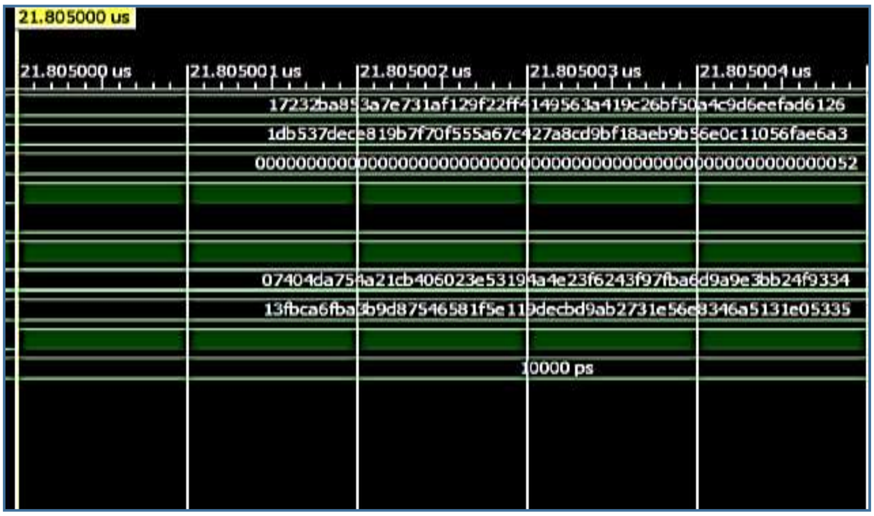

Figure 9. Simulation of ECP using interleaved multiplier.

Fig.9 shows the time required for scalar multiplication on Virtex-5, where the same generation point is used in Virtex-4. The maximum frequency of operation over this type of technology is 115.6MHZ. The number of slices distributed over different types of logic such as registers, LUTs, and FFs are shown in **Fig.10.**

```
Device utilization summary:
---------------------------

Selected Device : 5vlx155tff1738-3

Slice Logic Utilization:
 Number of Slice Registers:              3768  out of  97280     3%
 Number of Slice LUTs:                   4572  out of  97280     4%
    Number used as Logic:                4572  out of  97280     4%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:     5335
   Number with an unused Flip Flop:      1567  out of   5335    29%
   Number with an unused LUT:             763  out of   5335    14%
   Number of fully used LUT-FF pairs:    3005  out of   5335    56%
   Number of unique control sets:          13

IO Utilization:
 Number of IOs:                           474
 Number of bonded IOBs:                   474  out of    680    69%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                  1  out of     32     3%
```

Figure 10. Number of slices after synthesis process.

3. Simulation on Virtex-6:

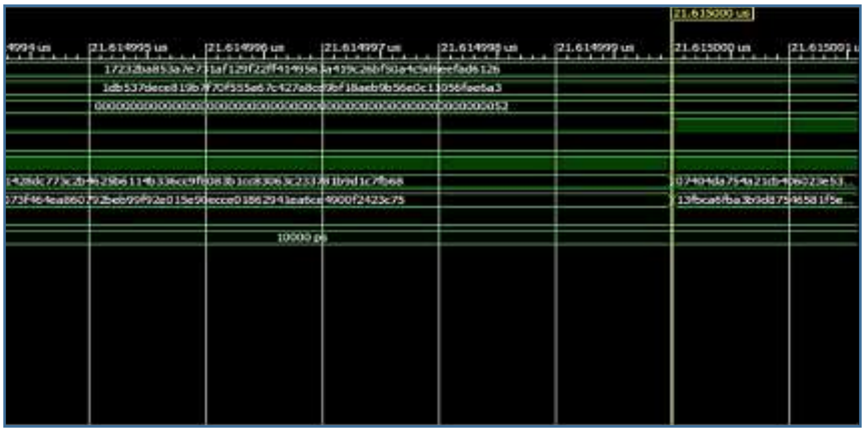

Figure 11. Simulation of point multiplication using interleaved multiplier.

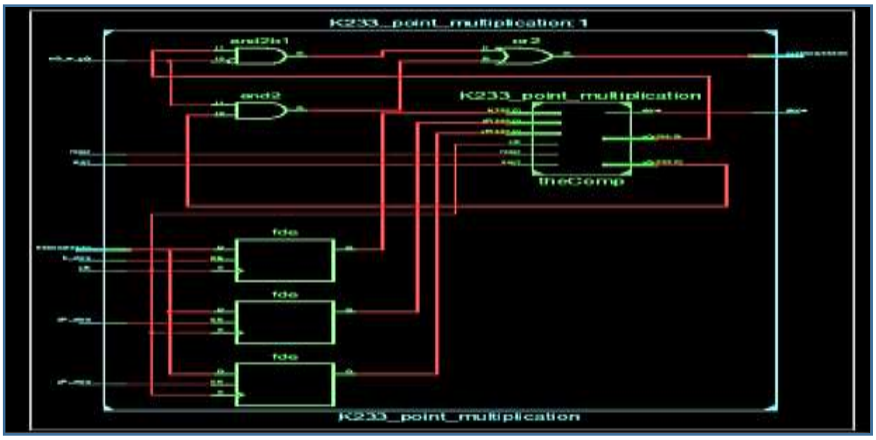

Figure 12. RTL schematic of ECP.

The RTL schematic of point multiplication over Virtex-6 is shown in **Fig.12**. K233_point_multiplication-1 represents the top component of the scalar multiplication.

The numbers of Slice-registers, Slice of LUTs, and FFs over this platform are 3077, 4064, and 2837, respectively, as shown in **Fig.13.**

top_K233_point_multiplication Project Status (12/28/2019 - 14:56:56)			
Project File:	ECP-Interleaved-V6.xise	**Parser Errors:**	No Errors
Module Name:	top_K233_point_multiplication	**Implementation State:**	Translated
Target Device:	xc6vlx130t-3ff1156	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	3 Warnings (0 new)
Design Goal:	Area Reduction	**• Routing Results:**	
Design Strategy:	Area Reduction with Physical Synthesis	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	3077	160000	1%
Number of Slice LUTs	4064	80000	5%
Number of fully used LUT-FF pairs	2837	4304	65%
Number of bonded IOBs	474	600	79%
Number of BUFG/BUFGCTRLs	1	32	3%

Figure 13. Virtex-6 Utilization of ECP using interleaved multiplier.

As shown in Table 1, the proposed design on Virtex-4 utilizes 3193 Slices, 340 LUTs, and 3772 Flip-flops and requires 21.805µs for point-multiplication. Thus, the PI approximates 0.38751206. On Virtex-5, the results show that 3768, 4572, and 5335 of a slice, LUTs, and Flip-flops are required, respectively. And the number of clock-cycles increases, leading to a maximum frequency of 115.9MhZ and an estimated PI 0. 0.308518945. By changing the platform to Virtex-6, the value of maximum frequency increases to 190MHz. Thus, the number of clock-cycles is approximately 4142, and the PI is 0.217610202. In Virtex-7, the results show that 3780 slices, 3646 LUTs, and 2682 Flip-flops are used, while the maximum frequency is 221.6Mhz. It is clear that the proposed design on Virtex-7 is working at a high-frequency 221.6Mhz of PI 0.2599156, which is higher than 0.217610202 on Virtex-6. However, the design on Virtex-6 is appropriate for limited area applications.
A higher Performance Index is obtained on Virtex-4 with a value of 0.38751206 and a low frequency of 86.6Mhz. In addition to a low number of clock-cycles (1884), when compared to other platforms. From the above, it can be seen that this proposal is appropriate for low-frequency applications.

Table 1. Comparison of PI among different platforms.

Ref	Point multiplication Algorithm	Area Information			Time Information		
		Slices	LUTs	FFs	CCS	Max Freq.	k.p(us)
This work	Lopez Dahab	3193	5340	3772	1884	86.4	21.805
	FF-Multiplier = Interleaved Multiplier PI=0.38751206, k=52 Platform =**Virtex-4** (xc4vlx80-12ff1148)						
	Lopez Dahab	3768	4572	5335	2527	115.9	21.805
	FF-Multiplier = Interleaved Multiplier PI=0.308518945, k=52 Platform = **Virtex-5** (5vlx155tff1738-3)						
	Lopez Dahab	3077	4064	2837	4142	190	21.8049
	FF-Multiplier = Interleaved Multiplier PI=0.217610202, k=52 Platform = **Virtex-6** (6vlx130tff1156-3)						
	Lopez Dahab	3780	3646	2682	4818	221	21.805
	FF-Multiplier = Interleaved Multiplier PI=0.2599156, k=52 Platform = **Virtex-7** (7vx550tffg1927-3)						
(Li and Li, 2016)	Montgomery	11708	21598	-	1926	194	9.9
	FF-Multiplier = bit-parallel PI=0.3297294, k=6 Platform = **Virtex-4** (Virtex4VLx-200)						

Fig.14 illustrates the total number of slices required by the proposed design using the first approach. The data shown in the figure demonstrate that the design utilizes less number of slices on Virtex-6.

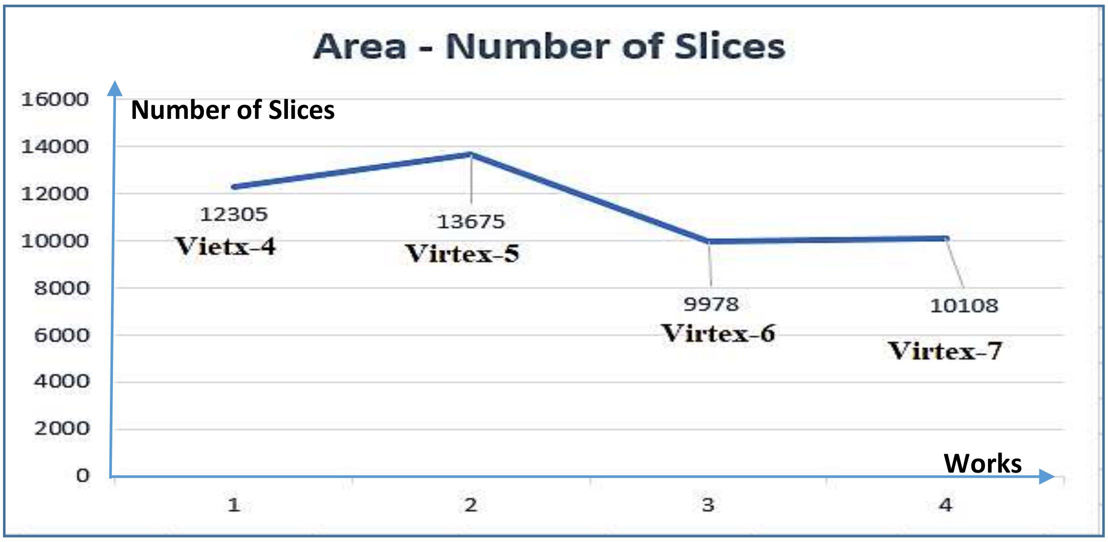

Figure 14. Slices of scalar multiplication among different FPGAs.

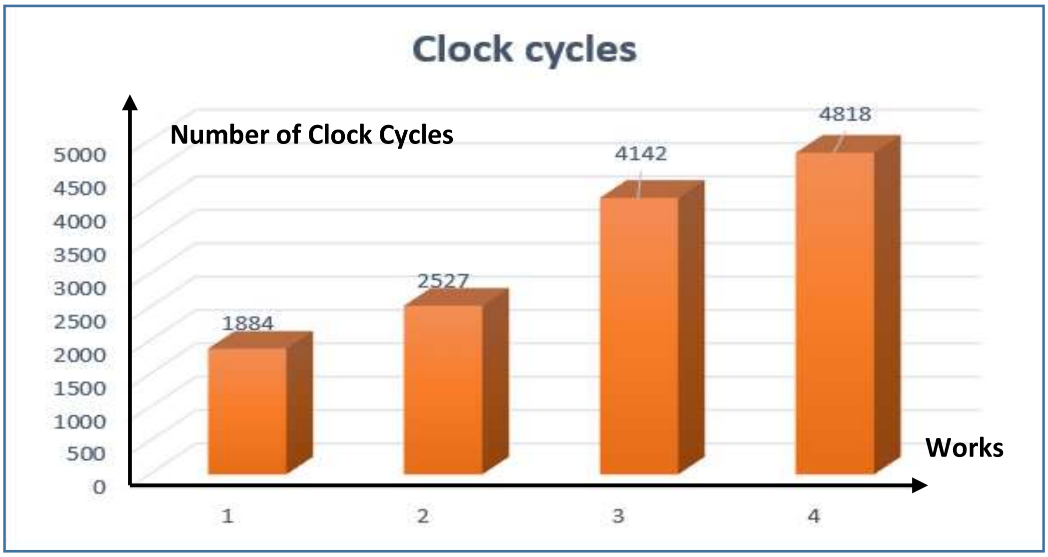

Figure 15. Clock cycles of ECP among different FPGAs.

Fig.15 shows the number of clock cycles of ECP among different FPGA, which makes it clear that Virtex-7 attains a high number of clock cycles of approximately 4818 when compared to others. Virtex-4 achieves the lowest number of the clock cycle with 1884, which makes suitable choices for low-frequency applications. **Fig.16** shows the performance index for scalar multiplication among different FGPA technologies using an interleaved multiplier, and the results show that ECP on Virtex-6 represents a better performance index with an estimated 0.21761202 when compared to the same design applied among different Xining platforms.

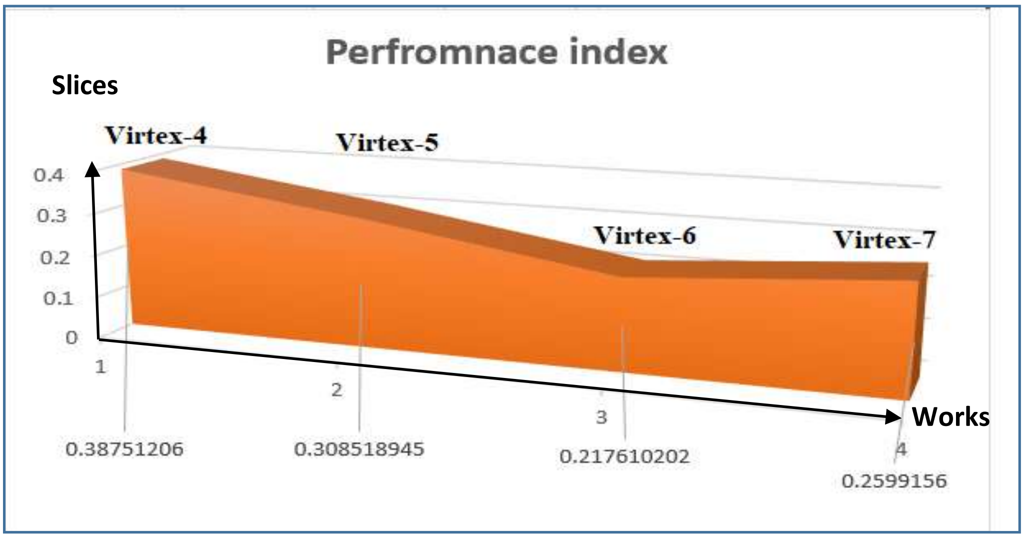

Figure 16. Performance index among different platforms.

4.2 Simulation of ECP for scenario 2

In this approach, the Montgomery multiplier with internal registers is applied in a Koblitz-point-addition instead of an interleaved multiplier, and the proposed design is implemented on different FPGA devices. **Fig.17** shows the time required for point multiplication, which is approximately 21.625000us. The maximum operating frequency is illustrated in **Fig.18,** respectively.

Simulation on Virtex-4:

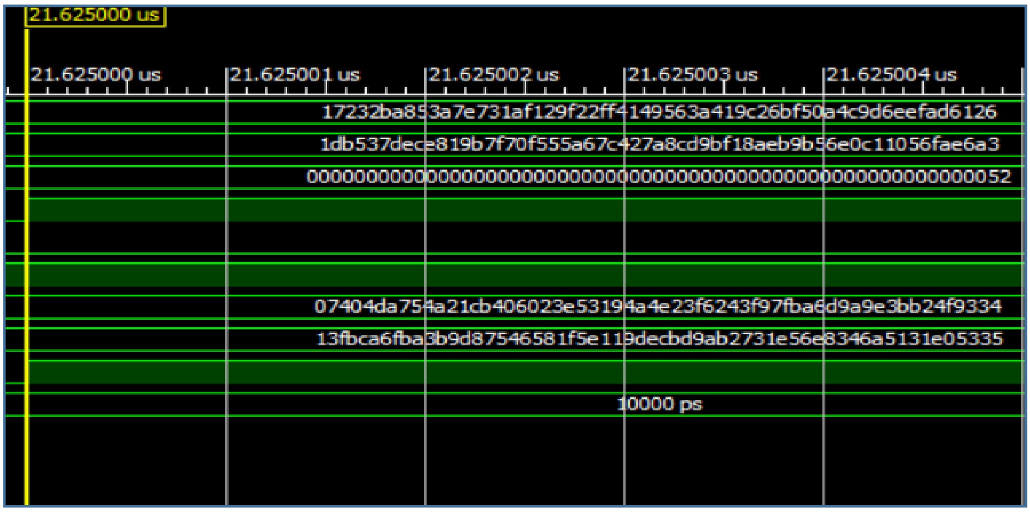

Figure 17. Time of point multiplication.

```
Timing Summary:
---------------
Speed Grade: -12

   Minimum period: 11.564ns (Maximum Frequency: 86.475MHz)
   Minimum input arrival time before clock: 2.430ns
   Maximum output required time after clock: 5.281ns
   Maximum combinational path delay: 5.937ns
```

Figure 18. Maximum frequency.

As shown in **Table.2** and **Fig.19,** the second proposed design based on the Montgomery multiplier representing the main component of ECP was implemented on different FGPA platforms. On Virtex-4, the design utilizes over 3340 slices, 3772 LUTs, and 5575 Flip-flops and requires 21.805µs for scalar multiplication. Thus, the PI approaches 0.276640035. On Virtex-5, the results showed that 3768, 4573, and 3005 are used from slices, LUTs, and Flip-flops, respectively. And the number of clock-cycles is increased. Thus, the maximum frequency was 115.9MhZ, and the estimated PI was 0.24739953. By changing the platform to Virtex-6, the value of maximum frequency is increased to 190MHz. Thus, the number of Clock-cycles approximates 4142, and the PI is 0.222815076. In Virtex-7, the results show that with 3077 slice, 4065 LUTs, and 4305 Flip-flop, the maximum frequency is 196Mhz. It is clear that the proposed design on Virtex-7 is working on a high frequency of 196Mhz with PI 0.249601835, which is higher than 0.217610202 on Virtex-6. This makes it a better choice for high-frequency applications. However, the design implemented on Virtex-6 is appropriate for low area applications occupying a small area. Due to the higher Performance Index obtained on Virtex-4 with 0.276640035 with a low frequency of 86.6Mhz and a low number of clock-cycles (1883) as compared to other platforms, this proposed design is more appropriate for low-frequency applications. The number of slices that represent registers, LUTs, and FFs for this approach over Virtex-4, Virtex-5, Virtex-6, and Virtex-7 is clearly shown in **Fig.20**. Obviously, the proposed design attains a smaller index of performance estimated at 0.22815076 as compared to the same design on other platforms.**Fig.21** shows the comparison of PI for the two approaches over different FPGAs.

Table 2. Comparison of PI among different platforms.

Ref	Point multiplication Algorithm	Area Information			Time Information		
		Slices	LUTs	FFs	CCS	Max Freq.	k. p (us)
This work	Lopez Dahab	3340	3772	5575	1883	86.4	21.605
	FF-Multiplier = Montgomery Multiplier PI= 0.276640035, k=52 Platform =**Virtex-4** (xc4vlx80-12ff1148)						
	Lopez Dahab	3768	4573	3005	2507	115.9	21.805
	FF-Multiplier = Montgomery Multiplier PI=0.24739953, k=52 Platform = **Virtex-5** (5vlx155tff1738-3)						
	Lopez Dahab	3543	2985	3691	4142	190	21.80499
	FF-Multiplier = Montgomery Multiplier PI=0.222815076, k=52 Platform = **Virtex-6** (6vlx130tff1156-3)						
	Lopez Dahab	3077	4065	4305	4273	196	21.805
	FF-Multiplier = Montgomery Multiplier PI=0.249601835, k=52 Platform = **Virtex-7** (7vx550tffg1927-3)						
(Li and Li, 2016)	Montgomery	11708	21598	-	1926	194	9.9
	FF-Multiplier = bit-parallel PI=0.3297294, k=6 Platform = **Virtex-4** (Virtex4VLx-200)						

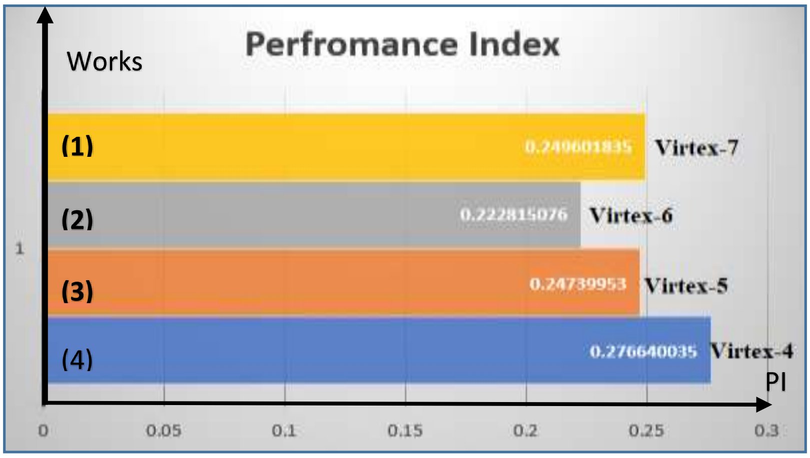

Figure 19. Performance Index among different FPGAs.

The total number of slices utilized in this approach is 12687, 11346, 10219, and 11447 on Virtex-4, Virtex-5, Virtex-6, and Virtex-7, respectively. It can be seen that the proposed design on Virtex-6 consumes a lesser number of slices (10219) as compared to the same design on other platforms.

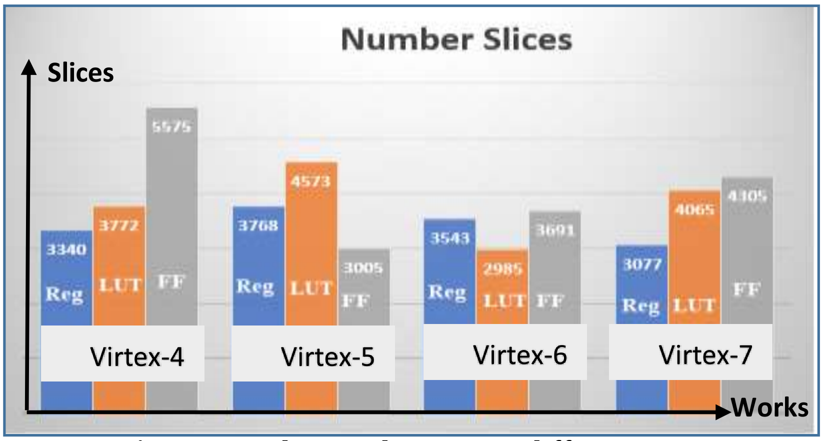

Figure 20. Slices utilization on different FPGAs.

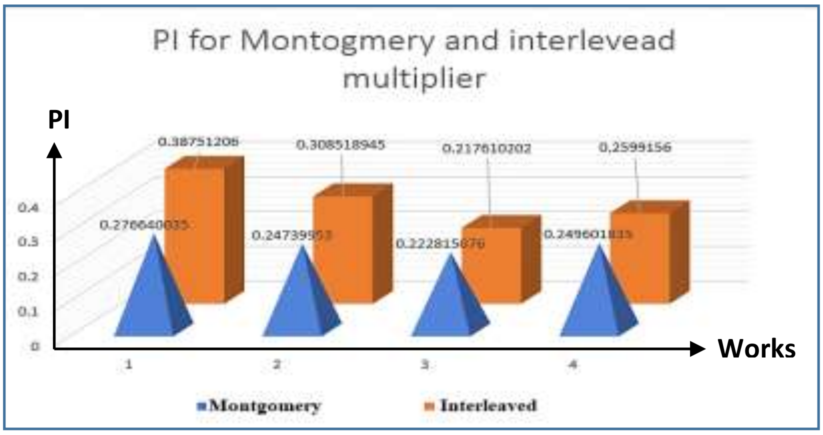

Figure 21. Performance index using different multipliers among different FPGAs.

To achieve fair comparison for the results obtained in this paper, with those obtained from the more related work done by (Li and Li, 2016) and implemented on Virtex-4, the proposed design (1) of both approaches in this work is chosen. The authors in the previous work chose k=6, while in this proposed design, k=52. This explains the difference between the time consumed in their work 9.9 us and the time consumed in design (1) which was approximately 21.805 us for both multipliers, as shown in **Tables 1** and **2**.
The PI of the previous work is 0.3297294, while the PI of design (1) using the first approach is 0.38751206, and of the second approach, 0.276640035.
So, their design is better than design (1), as shown in Table.1, when using an interleaved multiplier, but the design (1) shown in **Table.2** provides better PI than that obtained from previous work.

5. CONCLUSIONS

This paper presents the implementation of scalar multiplication based on the Lopez-Dahab algorithm. This algorithm uses point addition and squaring units as the cornerstone of point multiplication. The proposed design relies on the Koblitz curve with a binary field GF (2^{233}) bit. The index of performance equation was used as an analysis tool for comparison of the proposed design among different Xilinx's platforms using two different types of multipliers, either Montgomery or interleaved. It is shown that the proposed design on Virtex-6 with interleaved multiplier outperforms the other designs of the two approaches with a performance index of approximately 0.217610202 and a low number of total slices 9978. However, in general, ECP with Montgomery multiplier achieves a good performance index among all different technologies compared to ECP with the interleaved multiplier. The proposed design implemented on Virtex-7 in the first approach is appropriate for applications with high frequency since its maximum operational frequency approximately 221.6Mhz. In contrast, the proposed design on Virtex-4 is more suitable for applications with low-frequency, since its maximum frequency is approximately 86.4Mhz. Design (1) in Table.2, when using the Montgomery multiplier, provides a performance index better than the previous design of Li and Li in 2016.

REFERENCES

- Bilal, R., and Rajaram, M., 2010. 'High speed and Low space complexity FPGA based ECC processor', *International Journal of Computer Applications*, 8(3), pp. 5–10. DOI: 10.5120/1195-1673.
- Fournaris, A. P., Dimopoulos, C. and Koufopavlou, O., 2017. 'A Design Strategy for Digit Serial Multiplier Based Binary Edwards Curve Scalar Multiplier Architectures', *Proceedings - 20th Euromicro Conference on Digital System Design, DSD 2017*, pp. 221–228. DOI: 10.1109/DSD.2017.82.
- Hankerson, D., 2004. *Guide to Elliptic Curve Cryptography*, *Guide to Elliptic Curve Cryptography*. DOI: 10.1007/b97644.
- Hassan, A. A., 2010. 'Coloring of gray-scale image using FPGA', *Journal of Engineering*, 16(4), pp. 5932–5945.
- Hossain, M. S., Saeedi, E., and Kong, Y., 2016 'High-Performance FPGA Implementation of Elliptic Curve Cryptography Processor over Binary Field GF(2^163)', (Icissp), pp. 415–422. DOI: 10.5220/0005741604150422.
- Imran, M., Rashid, M. and Shafi, I., 2018. 'Lopez Dahab based elliptic crypto processor (ECP) over GF(2163) for low-area applications on FPGA', *2018 International Conference*

on Engineering and Emerging Technologies, ICEET 2018, 2018-Janua(February), pp. 1–6. DOI: 10.1109/ICEET1.2018.8338645.

- JEAN-PIERE DESCHAMPS, JOSE LUIS IMANA, G. D. S., 2009. *Hardware Implementation of Finite-Field Arithmetic*. The McGraw-Hill Companies,.
- Jwad, Z. A., Abdulaah, S. N. and Effing, W., 2012. 'Secured smart card simulation', *Journal of Engineering*,18(4), pp. 459–471.
- Kadu, R. K. and Adane, D. S., 2018. 'A novel efficient hardware implementation of elliptic curve cryptography scalar multiplication using vedic multiplier', *International Journal of Simulation: Systems, Science and Technology*, 19(6), pp. 38.1-38.10. DOI: 10.5013/IJSSST.a.19.06.38.
- Kawther E. Abdullah, N. H. M. A., 2018. A Secure Enhancement for Encoding/ Decoding data using Elliptic Curve Cryptography, Iraqi Journal of Science, Vol. 59, No.1A, pp: 189-198.
- Kilts, S. (no date) *Advanced FPGA Design: Architecture, Implementation, and Optimization: Steve Kilts: 9780470054376: Amazon.com: Books*. Available at: http://www.amazon.com/Advanced-FPGA-Design-Architecture-Implementation/dp/0470054379?tag=donations09-20.
- Li, L., and Li, S., 2016. 'High-performance pipelined architecture of elliptic curve scalar multiplication over GF(2m)', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(4), pp. 1223–1232. DOI: 10.1109/TVLSI.2015.2453360.
- Parrilla, L. *et al.*, 2019. 'Elliptic Curve Cryptography hardware accelerator for high-performance secure servers', *Journal of Supercomputing*. Springer US, 75(3), pp. 1107–1122. DOI: 10.1007/s11227-018-2317-6.
- Rashidi, B., 2018. 'Low-Cost and Fast Hardware Implementations of Point Multiplication on Binary Edwards Curves', *26th Iranian Conference on Electrical Engineering, ICEE 2018*. IEEE, pp. 17–22. DOI: 10.1109/ICEE.2018.8472703.
- Strukov, D., 2006. 'The area and latency tradeoffs of binary bit-parallel BCH decoders for prospective nanoelectronic memories', *Conference Record - Asilomar Conference on Signals, Systems and Computers*, pp. 1183–1187. DOI: 10.1109/ACSSC.2006.354942.
- Urbano-Molano, F. A., Trujillo-Olaya, V. and Velasco-Medina, J., 2013. 'Design of an elliptic curve cryptoprocessor using optimal normal basis over GF(2233)', *2013 IEEE 4th Latin American Symposium on Circuits and Systems, LASCAS 2013 - Conference Proceedings*, pp. 1–4. DOI: 10.1109/LASCAS.2013.6519014.