Hardware Implementation of Efficient Modified Karatsuba Multiplier Used in Elliptic Curves

Sameh M. Shohdy, Ashraf B. El-Sisi, and Nabil Ismail
(Corresponding author: Sameh M. Shohdy)

Computer Science Department, Faculty of Computers and Information, Menoufiya University
Gamal Abd-ElNaser St., Shebin Elkom 32511, Egypt
(Email: ashrafelsisi@hotmail.com, {s_mufic, nabil_a_ismail}@yahoo.com)
(Received Feb. 24, 2009; revised and accepted May 4 & June 27, 2009)

Abstract

The efficiency of the core Galois field arithmetic improves the performance of elliptic curve based public key cryptosystem implementation. This paper describes the design and implementation of a reconfigurable Galois field multiplier, which is implemented using field programmable gate arrays (**FPGAs**). The multiplier of Galois field based on Karatsuba's divide and conquer algorithm allows for reasonable speedup of the top-level public key algorithms. Binary Karatsuba multiplier is more efficient if it is truncated at n-bit multiplicand level and use an efficient classic multiplier algorithm. In these work three levels to truncate Binary Karatsuba algorithm (4 bits, 8 bits and 16 bits) are chosen showing that 8 bits is the best level for minimum number of slices and time delay to truncate Binary Karatsuba algorithm which is designed on an Xilinx VirtexE XCV2600 FPGA device. The VHDL hardware models are building using Xilinx ISE foundation software. This work is able to compute GF(2191) multiplication in 45.889 ns. experimental results of comparing block and stream ciphers when used to secure VoIP in terms of end-to-end delay and subjective quality of perceived voice.

Keywords: AES, CBC, MOS, QoS, VoIP

1 Introduction

During last decade cryptography took an important role in most of data exchange applications. Plain Data should be encrypted to cipher data before transferred to guarantee security necessities. On the other side the data decrypted to be ready to be processed in the main application. Cryptography algorithms divide in two categories: symmetric and asymmetric cryptosystems. In symmetric systems a single key is used to encrypt/decrypt the plain text to/from cipher text but due to the complex task for sharing this key between sender and receiver it's not preferred in applications that's cannot guaranteed secure key transferred. In asymmetric systems encryption and decryption operations done with two keys named private and public key. The private key owned by its owner and the public key is known for all other parties. When sender A needs to send data to receiver B, he encrypts his message with B's public key. This message will decrypted only with B's private key. So B only can read the message even if there is a third party spy on the channel.

Wireless Application Protocol (WAP) is a protocol for wireless devices [23]. WAP contains many layers to complete its function. One of them is Wireless Transport Layer Security (WTLS) which is a security layer to ensure privacy, data integrity and authentication between communication parties [10]. Secure transactions between client and server done by data encryption. But, first both parties must be authenticated before the beginning of transaction. Secure connection establishment consists of several steps includes key exchange and authentication. Because of the nature of wireless transmissions correspond to low bandwidth, limited processing power and memory capacity we need to speed up the security operation and in the same time makes a balance between security needs and energy consumption and area. Hardware accelerators for any public key algorithm is reduce calculation time due to parallel processing rather than sequential processing in software, but in the same time with large area use.

Elliptic Curve Cryptography (ECC) preferred when compared with classical cryptosystems such as RSA because of higher speed and lower power consumption which are particularly useful for wireless applications [1, 6, 9, 20]. Because of ECC guarantees the same security level as RSA but with shorter key size [4] and [8]. Elliptic curve used in many applications (e.g. Digital Signature, authentication protocols, etc.) [24]. But, the client and server authentication scenario illustrates how we can use it in Elliptic curve Diffie-Hellman authentication algorithm to authenticate client and server parties in WTLS layer at WAP protocol. Server and client should transfer cipher data using secret key to encrypt and decrypt the data. But, this is a problem for sharing this key between

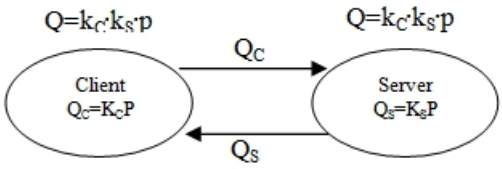

Figure 1: Elliptic curve Diffie-Hellman (ECDH)

the two parties. So there are some protocols useful for this problem. Diffie-Hellman key exchange protocol is one of them. ECC can be used for this protocol. First, client and server choose two random integer numbers K_C and K_S. Second, client computes Q_C using ECC scalar multiplication by multiply ECC point (P) by K_c. Also, server does the some operation but multiply P by K_S producing Q_S. Now, client transfers Q_C to the server and server transfers Q_S to the Client. Client receives Q_S and using Scalar multiplication again multiplies Q_S by K_C. On the other hand, server multiplies Q_C by K_S. The result in both sides is the same key K_c K_s P as shown in Figure 1. Now, the two parties have the same key which can be used as a secret key in the authentication process.

Hardware implementation of ECC passes through 3 main levels. First, underlying Galois field arithmetic which includes four operations field multiplication, field addition, field squaring and field inversion [14]. Second, elliptic curve preparation steps which includes Point doubling ($Q = 2P$) and point addition ($R = P + Q$). Finally, elliptic curve main operation (scalar multiplication ($Q = K \bullet P$)) as shown in Figure 2.

The long-term objective is to implement the first three levels layers to compute scalar multiplication for ECC using hardware to gain fast computation, reduce power and storage space. Many designs implement the scalar multiplication in hardware [2, 3, 15, 18, 21, 25]. This work concerns in implement Galois field multiplication operation [7, 13, 22, 26].

The organization of this paper is as follows. In Section 2 we describe mathematical background of elliptic curve cryptography and Galois field. Section 3 describes multiplication operation in $GF(2^m)$ and Karatsuba-Ofman multiplier. In Section 4 Architectural Design and Results for the implementation of Karatsuba multiplier implementing finite field arithmetic in $GF(2^{191})$ is presented. Finally, in Section 5 some conclusions remarks as well as future work are drawn.

2 Mathematical Background

2.1 Galois Field Arithmetic

Galois field or Finite field (F) defines as $GF(p^m)$ which is a field with finite number of elements (p^m elements with p is a prime number called characteristic of field) and two binary operation addition and multiplication satisfy

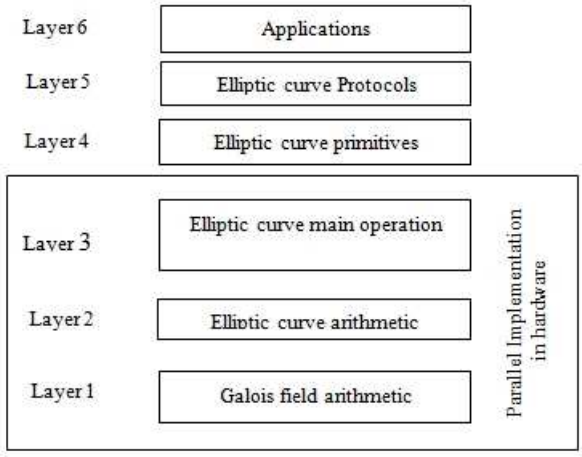

Figure 2: Elliptic curve cryptography hierarchical model

the following axioms: F is Abelian group with respect to '+', F {0} is an Abelian group with respect to '×' and distributive. Furthermore, Order of Galois field is the number of elements on the Galois field [4, 12].

Galois field arithmetic plays a critical role in elliptic curve cryptography implementation because it's the core of ECC scalar multiplication. So, more efficient implementation of underlying field operations results more efficient in the overall algorithm. Galois fields suitable for ECC implementation divides into two categories: prime field where $m = 1$ and binary field where $p = 2$ and $m > 1$. Binary Galois field preferred in hardware because of free carry propagation property in hardware which make addition operation only done with one n-bit XOR operation (equal to bit wise addition module 2), square operation done with no hardware resource rather than in (F_p) is cost as a general multiplication and faster Inversion operation in $GF(2^m)$.

Next two subsections discuss needed operations (addition, multiplication, squaring and inversion) for binary field needed for ECC implementation in hardware.

2.2 Binary Field

Finite field of order 2m is called binary field. Suppose Binary field ($F2^m$) and we have two elements $A, B \in F2^m$. Addition does not have any carry propagation. It can be done only with one n-bit XOR operation (equal to bit wise addition module 2), multiplication done by ordinary multiplication ($a \bullet b$) modulo irreducible polynomial $P(x)$ in $F2^m$, but does not have any carry propagation too in addition stage, squaring in F^2m done only by change bits order modulo irreducible polynomial $P(x)$ and Inversion computed by calculate A^{-1} which prove ($A \bullet A^{-1}$ mod $P(x) = 1$). For example: $F2^4$ is a Binary field with $m = 4$. $F2^4$ polynomial elements $\in$ {0, 1, x, $x + 1$ ... $x^3 + x^2 + x + 1$ }, suppose A, $B \in F2^4$ and $p(x) = x^4 + x + 1$. The four mathematical operations are illustrated in

Table 1: Binary field $F2^4$ arithmetic operations

Polynomial elements	**Binary Form**	**Operation**
A = $X^3 + X^2$ B = $X^2 + X + 1$	A = 1100 B = 0111	**Addition** A + B = $X^3 + X^2 + X^2 + X + 1$ = $X^3 + X + 1$ A + B = 1100 $\oplus$ 0111 = 1011
A = $X^3 + X^2$ B = $X^2 + X + 1$	A = 1100 B = 0111	**Multiplication** $A \bullet B = X^5 + X^4 + X^3 + X^4 + X^3 + X^2$ = $X^5 + X^2$ mod $X^4 + X + 1$ $A \bullet B$ = 1100 $\bullet$ 0111 =100100 (reduction step) 100100 mod 10011 = 1011
A = $X^3 + X^2$	A = 1100	**Squaring** $A^2 = X^6 + X^4$ $A^2 = \sum a^i x^{2i}$ = 1010000
A = $X^3 + X^2 + 1$	A = 1101	**Inversion** $A^{-1} = X^2$, Since $(X^3 + X^2 + 1) \bullet (X^2)$ mod $(X^4 + X + 1) = 1$ A^{-1} = 00100

Table 1.

2.3 Elliptic Curve Cryptography Arithmetic

Elliptic curves are defined over chosen finite field. For example, an elliptic curve defined over real numbers is a set of points (x, y) which satisfy the equation:

$$y^2 + a_1xy + a_2y \quad = \quad x^3 + a_3x^2 + a_4x + a_5,$$

where a_1, a_2, a_3, a_4, $a_5 \in R$ and a_1, a_2, $a_3 = 0$. Different values of a_4, a_5 produce different curve defined over real numbers. For example, Let $a_4 = -9$, $a_5 = 9$ the equation will be present as:

$$y^2 \quad = \quad x^3 - 9x + 9.$$

The elliptic curve is additive group (its main operation is the addition). Let A and B two points on the curve then $R = A + B$ can geometry represents by draw a line through the two points that will intersect the curve at another point, call $-R$. The point $-R$ is reflected in the x-axis to get a point R which is the required point as shown in Figure 3.

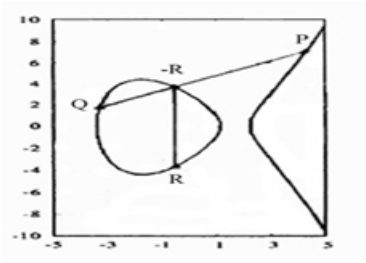

Figure 3: Addition operation in $y^2 = x^3 - 9x + 9$

ECC arithmetic: The main operation in ECC is the scalar multiplication operation ($Q = K \bullet P$, where k is integer and P is a point on the selected curve and Q is the scalar multiplication resulting from multiply K and P). There is no multiplication operation in elliptic curve groups However, the scalar product KP can be obtained by adding k copies of the same point P. The security of

elliptic curve systems is based on the difficulty of the elliptic curve discrete logarithm problem (ECDLP). ECDLP define as : Given an elliptic curve E defined over a Galois field (F_p) and two points Q and P that belong to the curve, the trick is to find the integer k which if multiplies by P we get Q. Pollard's rho is one of the popular algorithms known for solving the ECDLP. The largest ECDLP instance solved with Pollard's rho algorithm is for an elliptic curve over a 109-bit prime field as illustrates in [19].

Different schemes exist for performing elliptic curve scalar multiplication operation as mentions in [11]. For example, Montgomery algorithm which calculates the KP multiplication using two main blocks Point addition and Point doubling [21].

3 Multiplication Operations in $GF(2^m)$

Assume we have two elements $A(x)$, $B(x)$ belongs to binary field $GF(2^m)$ with irreducible polynomial $P(x)$. Field multiplication done by two steps:

- Polynomial multiplication of $A(x)$ and $B(x)$ $C'(x) = A(x) \bullet B(x)$;
- Reduction using irreducible polynomial $p(x)$ $C(x) = C'(x) \bmod p(x)$.

3.1 Karatsuba-Ofman Multiplier

As we mention above there are two steps to compute the multiplication operation over $GF(2^m)$. A lot of multipliers addressed the problem of compute polynomial multiplication some suitable for hardware and some for software. We addressed here two algorithms one suitable for composite field (where $m = ns$, $s = 2^t$, t integer) called Karatsuba multiplier first proposed in [16] and its modification-Binary Karatsuba multiplier-proposed in [17] which is suitable for $GF(2^m)$ where m an arbitrary number. The two algorithms depend on splitting the two multiplied elements to two parts for provide a parallel computation in hardware.

Let A, B two elements in the Galois field $F(2^m)$ they can be represented as:

$$\begin{aligned} A &= \sum_{i=0}^{m-1} a_i x^i = \sum_{i=0}^{m-1} a_i x^i + \sum_{i=0}^{\frac{m}{2}-1} a_i x^i \\ &= X^{\frac{m}{2}} A^H + A^L, \end{aligned}$$

where $A^H = \sum_{i=0}^{\frac{m}{2}-1} a_{i+\frac{m}{2}} X^i$, $A^L = \sum_{i=0}^{\frac{m}{2}-1} a_i x^i$.

Also,

$$\begin{aligned} B &= \sum_{i=0}^{m-1} b_i x^i = \sum_{i=0}^{m-1} b_i x^i + \sum_{i=0}^{\frac{m}{2}-1} b_i x^i \\ &= X^{\frac{m}{2}} B^H + B^L, \end{aligned}$$

where $B^H = \sum_{i=0}^{\frac{m}{2}-1} b_{i+\frac{m}{2}} X^i$, $B^L = \sum_{i=0}^{\frac{m}{2}-1} b_i x^i$.

With some modifications we can say that:

$$\begin{aligned} C(x) &= X^m A^H B^H + A^L B^L + (A^H B^H \\ &+ A^L B^L + (A^H + A^L)(B^H + B^L)) X^{\frac{m}{2}}. \quad (1) \end{aligned}$$

According Equations 1 Suppose $M_A = (A^H + A^L)$, $M_B = (B^H + B^L)$ then we need three polynomial multiplications: (A^H B^H, $A^L B^L$ and $M_A M_B$) and four polynomial additions: ($A^H + A^L$, $B^H + B^L$, $R = A^H B^H + A^L B^L$ and $R + M_A M_B$).

Algorithm 1 shows the Karatsuba algorithm. As seen in Lines 8-10 the algorithm called recursively to reduce the complexity of large size operand until an efficient point we can use classic algorithm - Line 2 - to complete the multiplication process for small size operands [19].

Algorithm 1 Karatsuba multiplier for composite fields

1: **Input:** Two elements A, $B \in GF(2^m)$ with $m = rn = 2^t n$, and where A and B can be expressed as $A = X^{m/2} A^H + A^L$, $B = X^{\frac{m}{2}} B^H + B^L$
2: **Output:** A polynomial $C = AB$ with up to $2m-1$ coordinates, where $C = X^m C^H + C^L$
3: **Procedure** Kmul 2^t (C, A, B)
4: Begin
5: **if** ($r == 1$) **then**
6: C = classic_mul(A, B)
7: return;
8: **for** for i from 0 to $\frac{r}{n} - 1$ **do**
9: $M_{Ai} = A_i^L + A_i^H$
10: $M_{Bi} = B_i^L + B_i^H$
11: **end for**
12: $Kmul2^t(C^L, A^L, B^L)$;
13: $Kmul2^t(M, M_A, M_B)$;
14: $Kmul2^t(C^H, A^H, B^H)$;
15: **for** i from o to r-1 **do**
16: $M_i = M_i + C_i^L + C_i^H$
17: **end for**
18: **for** i from 0 to $r-1$ **do**
19: $C_{\frac{r}{2}} + i = C_{\frac{r}{2}+i} + M_i$
20: **end for**
21: **end if**
22: End

As mentions in the algorithm each m bit polynomial multiplication divides into three $m/2$ bit polynomial multiplication and some polynomial additions which make a recursive methodology to compute polynomial multiplication [17]. It's recommended for $GF(2^m)$ to choose prime m values [5]. Now, we can say $m = 2^t + d$. to use Karatsuba multiplier we make $m = 2^{t+1}$for both elements and put $(2^{t+1} - d)$ most significant bits equal to zero. But, this operation waste arithmetic operation in multiply zero values.

In [17], an approach called *Binary Karatsuba multiplier algorithm* works using the same technique of Karatsuba multiplier rather than it can use for arbitrary degree of m. This approach splits A, B many times as Karatsuba multiplier but it cut off all complete zero operands from the computation as shown in Algorithm 2.

Classic or binary Karatsuba multiplier is more efficient if we truncate them at n-bit multiplicand level and use an efficient classic algorithm which called hybrid Karatsuba multiplier. In this work we design hybrid binary Karatsuba multiplier for $GF(2^{191})$. Also we choose three levels to truncate binary Karatsuba algorithm - (4, 8 and 16 bits) and making a comparison of the performance of three levels of hybrid Karatsuba multiplier.

Algorithm 2 Binary Karatsuba multiplier for arbitrary m

1: **Input:** Two elements $A, B \in GF(2^m)$ with m an arbitrary number, and where A and B can be expressed as $A = X^{\frac{m}{2}}A^H + A^L$, $B = X^{\frac{m}{2}}B^H + B^L$
2: **Output** A polynomial $C = AB$ with up to $2m-1$ coordinates, Where $C = X^m C^H + C^L$.
3: **Procedure** $BK(C, A, B)$
4: Begin
5: $k = [log_2 m]$
6: $d = m - 2^k$;
7: **if** $(d == 0)$ **then**
8: $C = Kmul2^k(A, B)$
9: return;
10: **for** i from 0 to d-1 **do**
11: $M_{Ai} = A_i^L + A_i^H$
12: $M_{Bi} = B_i^L + B_i^H$;
13: **end for**
14: $mul2^k(C^L, A^L, B^L)$;
15: $mul2^k(C^L, A^L, B^L)$;
16: $BK(C^H, A^H, B^H)$;
17: **for** i from 0 to $2k-2$ **do**
18: $M_i = M_i + C_i^L + C_i^H$
19: **end for**
20: **for** i from 0 to $2k-2$ **do**
21: $C_{k+i} = C_{k+i} + M_i$
22: **end for**
23: **for** i from 0 to $2k-2$ **do**
24: $C_{k+i} = C_{k+i} + M_i$
25: **end for**
26: **end if**
27: End

Figure 4 shows the architecture of Binary Karatsuba multiplier for $GF(2^{191})$ with the reduction Step demonstrates in next section.

4 Reduction Step

After calculating $C'(x) = A(x) \bullet B(x)$ the next step to completely compute polynomial multiplication is the reduction process $C(x) = C'(x) \bmod P(x)$. Once the irreducible polynomial $P(x)$ has been selected, the reduction step can be complete by using XOR gates only [19].

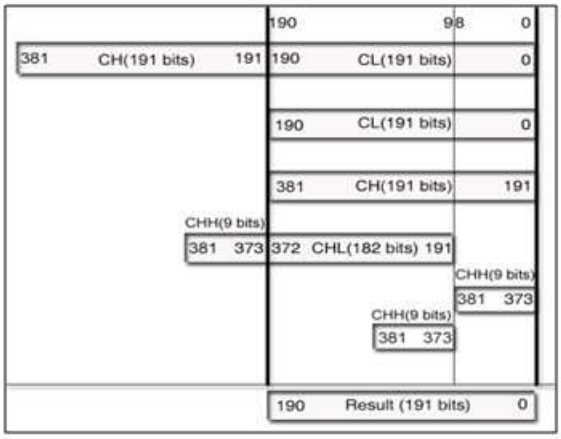

Figure 5: Reduction step in $GF(2^{191})$

$$
\begin{aligned}
C' &= \sum_{i=0}^{2m-2} Ci \\
C &= \sum_{i=0}^{m-1} CiwhereC = C' \bmod P(x) \\
C(x) &= C'_{[0,m-1]} + C'_{[m,2m-1]} + C'_{[m,2m-1-n]}X^n \\
&\quad + C'_{[2m-n,2m-1]} + (C'_{[2m-n,2m-1]}X^n). \quad (2)
\end{aligned}
$$

We select the irreducible polynomial $P(x) = X^{191} + X^9 + 1$ in the form of $X^m + X^n + 1$ for this work.

Figure 5 illustrates the reduction step in $GF(2^{191})$.

5 Architectural Design, Implementation, and Results

Depend on the architecture design illustrated in Figure 7. A parallel architecture was used for computing hybrid Binary Karatsuba multiplier truncated at 8 bit for $GF(2^{191})$ in 45.889 ns and 6, 265 slices. This work uses hybrid Binary Karatsuba multiplier (HBKM) and hybrid classic Karatsuba multiplier (HCKM) (both truncated at 4 bit and use any efficient classic multiplier) for 191 bits in different devices to illustrate the high cost of zero computation in classic Karatsuba algorithm. Also, to illustrates that the number of slices constant in all devices. But delay time change according to the device chosen and its speed this is because of different FPGA technology in each device as shown in Table 2.

Also, we design the hybrid Karatsuba multiplier with different n-bit truncated level (4, 8, and 16 bit) on an Xilinx VirtexE XCV2600 FPGA device. The results show different values of area and timing delay with different truncated level. The design shows that for an implementation of m bit hybrid Karatsuba multiplier 8 bit truncated level needs less number of FPGA slices and less

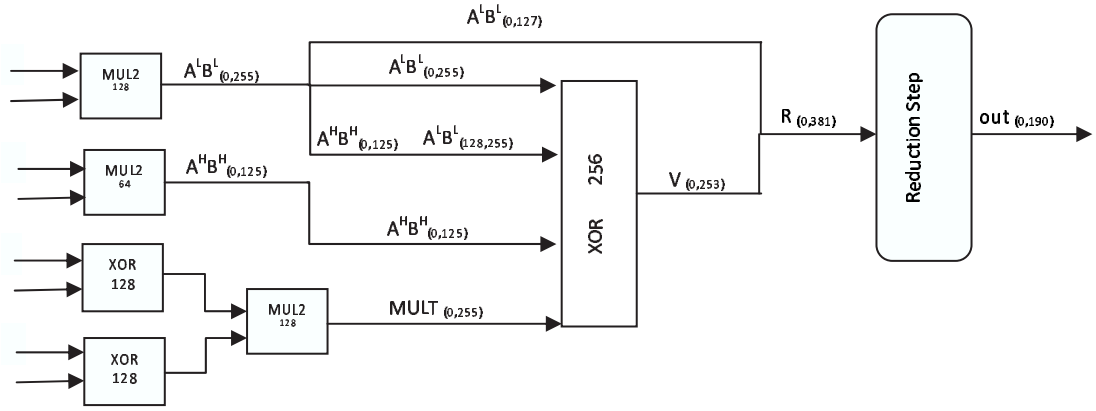

Figure 4: Karatsuba multiplier architecture for $GF(2^{191})$

Table 2: Classic/Hybrid Karatsuba multipliers for $GF(2^{191})$ design on different xilinx devices

Device	Algorithm	CLB slices	Time delay
xcv3200efg1156-8	HCKM (256 bits)	9,672	52.711 ns
xcv3200efg1156-8	HBKM(truncated at 4 bits)	6,632	48.950 ns
xcv3200efg1156-6	HCKM (256 bits)	9,672	63.967 ns
xcv3200efg1156-6	HBKM(truncated at 4 bits)	6,632	59.203 ns
xc2vp40ff1148 -7	HCKM (256 bits)	9,672	34.184 ns
xc2vp40ff1148 -7	HBKM(truncated at 4 bits)	6,632	32.632 ns
xcv2600efg1156-8	HCKM (256 bits)	9,672	50.708 ns
xcv2600efg1156-8	HBKM(truncated at 4 bits)	6,632	50.194 ns
xcv2600efg1156-6	HCKM (256 bits)	9,672	60.763 ns
xcv2600efg1156-6	HBKM(truncated at 4 bits)	6,632	59.203 ns

time delay on the XCV2600efg1156-8 device. Figures 6 and 7 shows the number of occupied FPGA slices and the time delay according to the size of the two multiplicand operands using Binary Karatsuba algorithm truncated at different levels designed on an Xilinx VirtexE XCV2600 FPGA device.

Also, making a comparison between different hardware implementation is not directly. This is because of different FPGA technology used for each implementation and the degree of finite field used. But, [19] shows an efficient measure to be the key of the comparison between different keys computed as:

$$S = \frac{number - of - bits}{number - of - slicing \times timing}.$$

In Table 3, this study is compared with several hardware implementations reported in previous works.

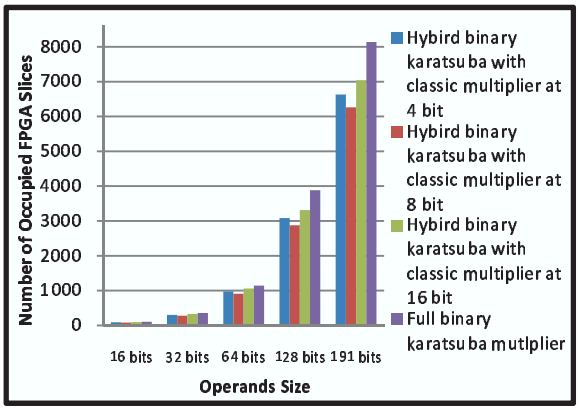

Figure 6: Number of occupied FPGA slices according to the size of the two multiplicand operands

6 Conclusions and Future Work

In this work, an architecture of implement polynomial multiplier for Binary Field $GF(2^{191})$ is presented using Xilinx xcv2600efg1156-8 FPGA device results 6, 265 occupied slices with time delay 45.889 ns. Also, this design use ISE 9.1 foundation tool for design HBKM and HCKM for 191 bits on different devices. Results show that truncated Binary Karatsuba multiplier at low level and use any efficient classic algorithm is more efficient rather than full Binary Karatsuba multiplier. Also, the results show that 8 bit truncated level is the best level for minimize number of slices and time delay to use classic Multiplier. But, delay time changes according to the device chosen and its speed. In the future work we will implement full ECC scalar multiplication for $GF(2^{191})$. Expected results will be minimums the number of slices and time delay needed according to efficient implementation for underlying field arithmetic.

Table 3: Comparison between different hardware $GF(2^M)$ multipliers

Reference and algorithm	FPGA Platform	Field	Number of slices	Timing	S
Ref[1] Montgomery multiplier	Virtex	160	1427 Slices	1.66 μs	0.0675
Ref[17], KaratsubaVOfman multiplier.	Virtex 2	163	5840 Slices	14.73μs	1.895
Ref[3], Binary multiplier	VirtexE XCV2600	163	351 Slices	2.2μs	0.2110
Ref [13], KaratsubaVOfman multiplier.	VirtexE XCV3200E	191	8721 Slices	43.1 μs	0.5081
This work, KaratsubaVOfman multiplier.	VirtexE XCV2600	191	8721 Slices	45.889 μs	0.6645

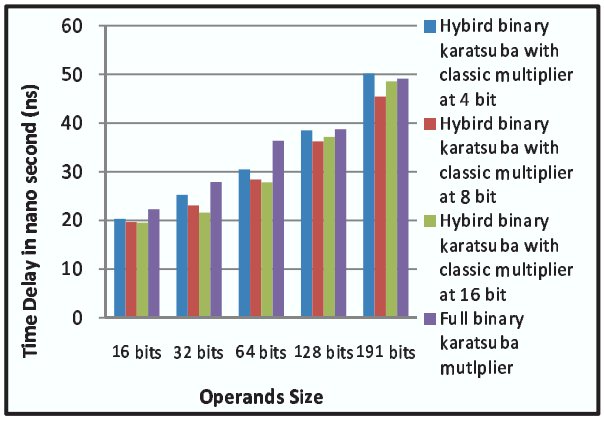

Figure 7: Time delay according to the size of the two multiplicand operands

References

[1] Z. Chen, L. Xu, and C. Wu, "Construction of large families of pseudorandom subsets of the set $\{1, 2, \cdots, N\}$ using elliptic curves," *International Journal of Network Security*, vol. 11, no. 3, pp. 122-127, 2010.

[2] C. C. R. Cheung, L. Wayne, and Y.K. P. Cheung,"Reconfigurable elliptic curve cryptosystems on a chip," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)*, pp. 24-29, 2005.

[3] M. Ernst, M. Jung, F. Madlener, S. Huss, and R. Blumel, "A reconfigurable system on chip implementation for elliptic curve cryptography over $GF(2n)$," *Cryptographic Hardware and Embedded Systems CHES 2002*, LNCS 2523, pp. 175-192, Spriger-Verlag, CA, USA, 2002.

[4] D. Hankerson, M. Alfred, and V. Scott, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, ISBN 038795273, 2004.

[5] IEEE P1363, *Standard Specifications for Public-Key Cryptography*, Draft ver. 7, Sep. 1998.

[6] B. King, "Mapping an arbritrary message to an elliptic curve when defined over $GF(2^n)$," *International Journal of Network Security*, vol. 8, no. 2, pp. 169-176, 2009.

[7] J. Lee, H. Kim, Y. Lee, S. M. Hong, H. Yoon, "Parallelized scalar multiplication on elliptic curves defined over optimal extension field," *International Journal of Network Security*, vol. 4, no. 1, pp. 99-106, 2007.

[8] A. Lenstra and E. Verheul, "Selecting cryptographic key sizes," *Proceedings Workshop on Practice and Theory in Public Key Cryptography*, LNCS 1751, pp. 446-465, Springer-Verlag, 2000.

[9] T. C. Lin, "Algorithms on elliptic curves over fields of characteristic two with non-adjacent forms," *International Journal of Network Security*, vol. 9, no. 2, pp. 117-120, 2009.

[10] Y. L. Liu, W. Gao, H. X. Yao, and X. H. Yu, "Elliptic curve cryptography based wireless authentication protocol," *International Journal of Network Security*, vol. 5, no. 3, pp. 327-337, 2007.

[11] J. Lopez and R. Dahab, "Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation," *Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems (CHES '99)*, LNCS 1717, pp. 316-327, Springer-Verlag, Cancun, Mexico, 1999.

[12] R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic Publishers, 1987.

[13] N. A. Moldovyan, "Acceleration of the elliptic cryptography with vector finite fields," *International Journal of Network Security*, vol. 9, no. 2, pp. 180-185, 2009.

[14] S. Moon, "A binary redundant scalar point multiplication in secure elliptic curve cryptosystems," *International Journal of Network Security*, vol. 3, no. 2, pp. 132-137, 2006.

[15] G. Orlando and C. Paar, "A scalable $GF(p)$ elliptic curve processor architecture for programmable hardware," *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, LNCS 2162, pp. 348-363, Springer-Verlag, Paris, France, May 14-16, 2001.

[16] C. Paar, P. Fleischmann, and P. Soria-Rodriguez, "Fast arithmetic for public-key algorithms in galois fields with composite exponents," *IEEE Transactions on Computers*, vol. 48, no. 10, pp. 1025-1034, 1999.

[17] F. Rodriguez-Henriquez and Q. K. Kog, "On fully parallel Karatsuba multipliers for $GF(2^m)$," *Inter-*

national Conference on Computer Science and Technology (CST), pp. 405-410, 2003.

[18] F. Rodriguez-Henriquez, N. A. Saqib, and A. Diaz-Pérez, "A fast parallel implementation of elliptic curve point multiplication over $GF(2^m)$," *Microprocessors and Microsystems*, vol. 28, no. 5-6, pp. 329-339, Aug. 2004.

[19] F. Rodriguez-Henriquez, N. A. Saqib, A. Diaz-Perez, and K. C. Kaya, *Cryptographic Algorithms on Reconfigurable Hardware*, Springer-Verlag, ISBN 0387338837, 2006.

[20] H. Sahu and B. K. Sharma, "An MSS based on the elliptic curve cryptosystem," *International Journal of Network Security*, vol. 12, no. 1, pp. 1-3, 2011.

[21] N. A. Saqib, F. Rodríguez-Henruez, and A. Díaz-Pérez, "A reconfigurable processor for high speed point multiplication in Elliptic Curves," *International Journal of Embedded Systems*, vol. 1, no. 3-4, pp. 237-249, 2005.

[22] Z. J. Shi and H. Yun, "Software implementations of elliptic curve cryptography," *International Journal of Network Security*, vol. 7, no. 1, pp. 141-150, 2008.

[23] D. Singelee and B. Preneel, "The wireless application protocol," *International Journal of Network Security*, vol. 1, no. 3, pp. 161-165, 2005.

[24] M. Toorani and A. A. B. Shirazi, "Cryptanalysis of an elliptic curve-based signcryption scheme," *International Journal of Network Security*, vol. 10, no. 1, pp. 51-56, 2010.

[25] E. h. Y. Wajih, G. Zied, M. Mohsen, and T. Rached, "Design and implementation of elliptic curve point multiplication processor over $GF(2^m)$," *International Journal of Computer Sciences and Engineering Systems*, vol. 2, no. 2, pp. 121-129, 2008.

[26] D. Yong, Y. F. Hong, W. T. Wang, Y. Y. Zhou, and X. Y. Zhao, "Speeding scalar multiplication of elliptic curve over $GF(2^m n)$," *International Journal of Network Security*, vol. 11, no. 2, pp. 70-77, 2010.

Sameh M. Shohdy received his B.S. in Computer Science from Faculty of Computers and Information, Menoufiya University in 2005. He is pursuing his M.S. in Computer Science from Menoufiya University. His current research interests include Information Security, and Reconfigurable Computing.

Ashraf B. El-Sisi received the B.S. and M.S. in Electronic Engineering and Computer Science Engineering from Menoufyia University, Faculty of Electronic in 1989 and 1995, respectively and received his PhD in Computer Engineering & Control from Zagazig University, Faculty of Engineering in 2001. His research interest includes His research interest includes privacy preserving data mining, AI approaches in software testing, Intelligent Agent, Implementation of security and graphics algorithms based on FPGA, Testing biometric security algorithms and devices.

Nabil Ismail is used to be the Dean of the Faculty of Computers and Information, Menoufia University, (Aug. 2006- Aug. 2008). He is now a Professor of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufia University. Prof. Ismail obtained his PhD degree in Computer Eng. from Durham University, England, in 1983. His main research area includes Computer security, Computer architecture, high performance cryptography processor design, light-power smart devices security applications, and EIT algorithms.