Umm Al-Qura Univ. J. Eng.& Arch. Vol. 4, No.1, pp. 15-27 (2011) 31-06-011 © The University Press

Fast Elliptic Curve Scalar Multiplication with Pipelined Redundant Representation Multiplier

تركي فيصل الصماني
Turki F. Al-Somani

Computer Engineering Department, Faculty of Computer & Information Systems,
Umm Al-Qura University, Makkah, Saudi Arabia
E-mail : tfsomani@uqu.edu.sa

الملخص

لقد جذبت أنظمة التشفير المبنية على المنحنيات الإهليجية اهتمام العديد من الباحثين ، وتم إدراجها في عدة مقاييس ومعايير عالمية باعتبارها بديلاً جذاباً لأنظمة التشفير العمومي الأخرى مثل آر-إس-آ **(RSA)** لصغر حجم المفتاح السري من جانب ، وقوة الحماية العالية لكل بت من جانب آخر ~~لكل بت~~ . وقد ~~قدمت~~ أجريت العديد من الأبحاث التي تدرس الأساس الرياضي وقوة الحماية وكفاءة تطبيقات أنظمة التشفير المبنية على المنحنيات الإهليجية. وحتى الآن لم يتم رصد أي اختراقات ذات شأن تثبت ضعف قوة أمن أنظمة التشفير المبنية على المنحنيات الإهليجية والمبنية على مسألة اللوغاريتم المنفصل على نقاط المنحنى الإهليجي ، هذه الحقيقة والتي بينت أن هذه المسألة صعبة الاختراق للغاية ، جعلت أنظمة التشفير المبنية على المنحنيات الإهليجية بإمكانها استخدام أحجام مفاتيح أصغر ، مما جعلها تشكل تحدياً لأنظمة التشفير آر-إس-آ المهيمنة. تدرس هذه الورقة تأثير مضاعف خطوط عمليات تسلسلي عالي الأداء مبني على الحقل جي-إف (إم ٢) (**GF(2^m)**) على عمليات نقاط المنحنى الإهليجي. تمت دراسة مضاعف خطوط عمليات ثلاثي المراحل ذي وفرة في التمثيل مبني على الحقل **GF(2^m)** للقيم **$160 \le m \le 256$** من حيث المساحة المطلوبة وتحسين الإنتاجية. وقد استخدمت نماذج مساحة وسرعة البوابات المنطقية لتقدير إنتاجية مضاعف خطوط العمليات. تبين أن البنى المقترحة قدمت تطوراً ملحوظاً في الإنتاجية لمضاعف خطوط عمليات الثلاثي المراحل أفضل من استخدام ثلاثة مضاعفات غير مبنية لخطوط العمليات ، كما بين كذلك مقياس الأداء **AT^2** تطوراً ملحوظاً.

Abstract

Recently, Elliptic Curves Cryptosystems (ECCs) have attracted many researchers and have been included in many standards. ECCs are evolving as an attractive alternative to other public-key schemes such as RSA by offering the smallest key size and the highest strength per bit. Extensive research has been done on the underlying math, security strength and efficient implementations of elliptic curve cryptosystems. To date, no significant breakthroughs have been reported in determining weaknesses in the ECCs, which are based on the discrete logarithm problem over points on an elliptic curve. The fact that the problem appears so difficult to crack means that key sizes can be considerably reduced allowing ECCs to challenge the dominating position enjoyed thus far by the RSA cryptosystems. This paper studies the effect of a high performance pipelined GF(2^m) bit-serial multiplier on elliptic curve point operations. A 3-stage pipelined GF(2^m) redundant representation multiplier for $160 \le m \le 256$ was studied in terms of area overhead and throughput improvement. Simple gate area and delay models were used to estimate the throughput of the pipelined and the non-pipelined multipliers. The proposed pipelined architecture has been shown to have a significant improvement in throughput allowing a single 3-stage pipelined multiplier to have higher throughput than an architecture employing three parallel non-pipelined multipliers. The AT^2 performance metric has shown an even more significant improvement.

***Keywords*:** Normal Basis, Redundant Representation, Finite Fields, Elliptic Curve Cryptosystems, Projective Coordinates, Pipelining.

1. Introduction

Elliptic Curve Cryptosystems (ECCs) have been recently attracting increased attention (Koblitz 1987, Hankerson *et. al.* 2004). Standards for ECCs have been adopted by IEEE, ANSI, NIST, SEC and WTLS. The ability to use smaller key sizes and the computationally more efficient ECC algorithms compared to those used in earlier public key cryptosystems such as RSA (Rivest *et. al.* 1978) and ElGamal (ElGamal 1985) are two main reasons why ECCs are becoming more popular. They are considered particularly suitable for implementation on smart cards or mobile devices.

Since ECCs have been proposed in 1985, extensive research has been done on the underlying math, security strength and efficient implementations. Among the different fields that can underlie elliptic curves, prime fields GF(p) and binary fields GF(2^m) have been shown to be best suited for cryptographic applications. In particular, binary fields allow for fast computation in software as well as hardware (McEliece 1987, Lidl & Niederreiter 1994).

Inversion operations are the most expensive operations over Finite Fields (Blake *et. al.* 1999). The approach adopted in the literature is to represent Elliptic Curve points in projective coordinate systems in order to replace the inversion operations with repetitive multiplications (Lopez & Dahab 1998, Blake *et. al.* 1999). Almost all recently reported ECC cryptoprocessors use projective coordinate systems. Several projective coordinate systems have been reported and are in common use (Lopez & Dahab 1998, Blake *et. al.* 1999, Hankerson *et. al.* 2004). The selection of a particular projective coordinate system, however, has mostly been influenced by the number of arithmetic operations, mainly multiplications, required to perform elliptic curve point operations. This is expected due to the sequential nature of these architectures where a single multiplier is used.

For high performance servers, such sequential architectures are too slow to meet the ever increasing demand. For such servers, high-speed cryptoprocessors are becoming a necessity. One approach to meet this requirement is to exploit the inherent parallelism within Elliptic curve point operations in projective coordinate systems. Recently, several ECC cryptoprocessor architectures have been proposed where the choice of the projective coordinate system was influenced by the degree of its inherent parallelism (Gutub & Ibrahim 2003a, Gutub & Ibrahim 2003b, Al-Somani, *et. al.* 2006, Gutub *et. al.* 2007, Al-Somani 2008, Al-Somani 2010). These architectures have reported better (AT^2) area-time complexity compared to other architectures that are based only on a single multiplier (Thompson 1980). Recently, Al-Somani *et. al.* (2006) conducted a comparative study on sequential and parallel designs using different projective coordinate systems. Al-Somani *et. al.* in (2006) analyzed the dataflow of point operations for each projective coordinate system by finding the critical path that has the lowest number of the multiplication operations. Accordingly, the number of multipliers needed to meet this critical path is found.

This paper studies the effect of a proposed high-performance pipelined GF(2^m) multiplier on elliptic curve point operations. The rest of this paper is organized as follows: Section 2 gives a brief introduction to ECCs. Section 3 describes the proposed pipelined multiplier. Section 4 provides performance metrics for the pipelined versus the non-pipelined multipliers regarding their area overhead and system throughput. Section 5 presents comparisons between the two multiplier architectures for different projective coordinate systems. Finally, Section 6 concludes this work.

2. Elliptic Curve Cryptosystems Preliminaries

Elliptic curve cryptosystems, which are based on the discrete logarithm problem over points on an elliptic curve, were originally proposed by Niel Koblitz and Victor Miller in 1985 (Koblitz 1987). They are considered as a viable alternative to RSA cryptosystems but with much shorter key sizes. An ECC with key size of 128-256 bits has been shown to offer equal security to that of RSA with key size of 1-2K bits (Hankerson *et. al.* 2004). To date, no significant breakthroughs have been reported in determining weaknesses in the ECCs. The fact that the problem appears so difficult to crack means that key sizes can be considerably reduced allowing ECC to challenge the dominating position enjoyed thus far by the RSA cryptosystems. Because of their short key sizes, ECCs have been gaining popularity and are considered particularly suitable for implementation on smart cards and mobile devices.

An elliptic curve E over the finite field GF(p) defined by the parameters a, $b \in$ GF(p) with $p > 3$, consists of the set of points $P = (x, y)$, where $x, y \in$ GF(p), that satisfy the equation:

$$y^2 = x^3 + ax + b \qquad (1)$$

where a, $b \in$ GF(p) and $4a^3 + 27b^2 \neq 0 \bmod p$, together with the point at infinity O which is the additive identity of the group. The number of points $\#E$ on an elliptic curve over a finite field GF($q = p^m$) is defined by Hasse's theorem (McEliece 1987). The set of discrete points on an elliptic curve forms an abelian group, whose group operation is known as point addition. Elliptic curve point addition is defined according to the "chord-tangent process". Point addition over GF(p) is described as follows:

Let P and Q be two distinct points on an elliptic curve E defined over GF(p) with $Q \neq -P$ (Q is not the additive inverse of P). The addition of the two points P and Q is the point R ($R = P + Q$), where R is the additive inverse of S, with S being the third point on E intercepted by the straight line through points P and Q. The additive inverse of a point $P = (x, y)$ is the point $-P = (x, -y)$ which is the reflection of the point P with respect to the x-axis on E. When $P = Q$ and $P \neq -P$ the addition of P and Q is the point R ($R = 2P$), where R is the additive inverse of S with S being the third point on E intercepted by the straight line tangent to the curve at point P. This operation is referred to as point doubling.

The finite field GF(2^m) is of particular importance in cryptography since it leads to efficient hardware implementations. Elements of the field are represented in terms of a basis. Most implementations use either a Polynomial Basis or a Normal Basis (Lidl and Niederreiter 1994). Let GF(2^m) be a finite field of characteristic two. A non-supersingular elliptic curve E over GF(2^m) is defined to be the set of solutions $(x, y) \in$ GF(2^m) × GF(2^m) to the equation,

$$y^2 + xy = x^3 + ax^2 + b \qquad (2)$$

where a and $b \in$ GF(2^m), $b \neq 0$, together with the point at infinity.

It is well known that E forms a commutative finite group, with O being the group identity, under the addition operation. Explicit formulas for the addition rule involve several field arithmetic operations (addition, squaring, multiplication and inversion) in the underlying finite field. The group operation in affine coordinate system involves finite field inversion, which is a very costly operation, particularly over prime fields. Projective coordinate systems are used to eliminate the need for performing inversion. Several projective coordinate systems have been proposed in the literature including homogeneous Jacobian and Lopez-Dahab coordinate system (Lopez & Dahab 1998, Blake *et. al.* 1999, Hankerson *et. al.* 2004). For elliptic curve defined over GF(2^m), many different forms of formulas may be used for point addition and doubling. For the Homogeneous coordinate system, an elliptic curve point (x, y) takes the form $(x, y) = (X/Z, Y/Z)$, while for the Jacobian coordinate system, a point takes the form $(x, y) = (X/Z^2, Y/Z^3)$. The Lopez-Dahab coordinate system, on the other hand, takes the form$(x, y) = (X/Z, Y/Z^2)$.

Adding a point P on the elliptic curve E to itself a number of times (k) is known as the scalar product (kP) of point P by the scalar k. Scalar multiplication is a basic operation for ECCs which, in the group of points of an elliptic curve, is analogous to exponentiation in the multiplicative group of integers modulo a fixed integer m. The scalar multiplication operation (kP) yields a point on the elliptic curve which is the result of adding point P to itself k times. Several scalar multiplication methods have been proposed in the literature (Hankerson *et. al.* 2004). Computing kP can be done using a straightforward binary method, the double-and-add method, based on the binary expression of the multiplier k. Computing kP using the binary method is described as follows:

Let $k = (k_{m-1},\ldots,k_0)$, where k_{m-1} is the most significant bit of k, be the binary representation of k. The multiplier k can be written as:

$$k = \sum_{0\le i<m} k_i 2^i = k_{m-1}2^{m-1} + k_{m-2}2^{m-2} + \cdots + k_1 2 + k_0 \quad (3)$$

Using the Horner expansion, k can be rewritten as:

$$k = (\ldots((k_{m-1}2 + k_{m-2})2 + \ldots + k_1)2 + k_0) \quad (4)$$

Accordingly, kP can be written as:

$$kP = 2(\ldots 2(2(2k_{m-1}P + k_{m-2}P) + \ldots + k_1P) + k_0P \quad (5)$$

The binary method algorithm is shown below:

Algorithm 1: The Binary Algorithm: most-to-least version

```
1. input $P$, $k$
2. $Q \leftarrow P$
3. for $i$ from $m$-2 downto 0 do
    3.1. $Q \leftarrow 2Q$
    3.2. if $k_i = 1$ then $Q \leftarrow Q + P$
4. end for
5. output $Q$
```

The binary scalar multiplication method is the most straightforward scalar multiplication method. It inspects the bits of the scalar multiplier k, if the inspected bit $k_i = 0$, only a point doubling operation is performed. If, however, the inspected bit $k_i = 1$, both a point doubling and a point addition operations are performed. The binary method requires m point doublings and an average of $\frac{m}{2}$ point additions.

3. The Proposed Pipelined Multiplier

Instead of using parallel multipliers to improve performance, in this work we study the effect of using a pipelined multiplier to increase the system throughput and, hence, overall performance. This approach saves the area taken by the parallel multipliers at the cost of a slight area increase due to the added pipeline registers. This, however, is minimized in the proposed approach since only 3-pipeline stages are put forward with only two added register stages required. The multiplier used here is the Wu *et. al.* (2002) redundant representation bit-serial multiplier. For a reasonable level of security, we will investigate private key sizes (m) in the range $160 \leq m \leq 256$. Results obtained, however, can be easily extended to larger key sizes.

Wu *et. al.* in (2002) have proposed an efficient redundant representation bit-serial multiplier over GF(2^m). Multiplication is performed in (Wu *et. al.* 2002) by converting the two operands into another basis (which is simply a permutation of the normal basis), performing the multiplication and converting the product back to the original normal basis.

The Wu *et. al.* multiplier requires ($2m + 1$) registers and combinational logic. The combinational logic consists of a set of AND and XOR logic gates. The area complexity of the Wu *et. al.* (2002) multiplier is m AND gates + ($2m - 1$) XOR gates, while the time complexity is $T_A + [1 + log_2(m)]T_X$, where T_A and T_X are the delays of one AND gate and one XOR gate, respectively.

Figure 1 shows an example of a GF(2^5) Wu *et. al.* (2002) bit-serial field multiplier. The combinational logic of the multiplier (Figure 1) consists of one level of AND logic gates and four levels of XOR logic gates. In general, the combinational logic of the Wu *et. al.* (2002) bit-serial multiplier over GF(2^m) consists of one level of m AND logic gates and $[1 + log_2(m)]$ levels of XOR logic gates. For the range of m considered in this work, $160 \leq m \leq 256$, the number of levels of XOR logic gates is 9 levels. Thus, the proposed pipelined Wu *et. al.* (2002) multiplier will be partitioned into three stages. The first stage (Stage_0) contains one level of AND logic gates and three levels of XOR logic gates. The remaining two stages (Stage_1 & Stage_2), on the other hand, contain only three levels of XOR logic gates as illustrated in Figure 2.

4. Performance Analysis

To compare the proposed pipelined multiplier with original non-pipelined Wu *et. al.* (2002) multiplier in terms of area, we use an area metric corresponding to the sum of individual element estimated areas. The estimated area of various elements corresponds to the area of an equivalent number of simple logic gates. A simple logic gate is assumed to have a unit area of 1U. An AND logic gate is considered as one simple logic gate with one unit area 1U. An XOR logic gate has a typical area of two simple logic gates, i.e. 2U, while a Latch area is equivalent to four simple logic gates, or 4U. Accordingly, the area of the non-pipelined Wu *et. al.* (2002) multiplier of m AND gates + ($2m - 1$) XOR gates = m * 1U + ($2m - 1$) * 2U = ($5m$ -2) U.

Ignoring the wiring cost, the Latches that are inserted between the stages constitute the main area overhead. We estimate the required number of Latches between the stages, for a value of m = 256 bits. Accordingly, the first level of logic gates consists of 256 AND gates. The first level of XOR logic gates, on the other hand, will have 256 XOR gates. Each of the remaining levels of XOR gates will contain half the number of XOR gates in the preceding level. This implies that a 64-bit register is required between Stage_0 & Stage_1 and an 8-bit register is required between Stage_1 & Stage_2. Thus, the area of the proposed pipelined multiplier is m AND gates + ($2m - 1$) XOR gates + 72 register bits which is equivalent to an area metric of ($5m$ + 286)U. Table 1 lists the area of the proposed pipelined multiplier and the non-pipelined Wu *et. al.* (2002) multiplier.

To compare the proposed pipelined multiplier with the original non-pipelined Wu *et. al.* (2002) multiplier in terms of throughput, we compute the clock period for both designs. Delay calculations are normalized in terms of the delay of a simple logic gate τ. AND gates are assumed to be a simple logic gate of delay 1τ, while the delay of an XOR gate is equal to 2τ. Assumptions made regarding the relative delay and area figures have been verified through the study of several CMOS standard cell libraries for the concerned gates assuming 1x drive[1]. The Wu *et. al.* (2002) multiplier non-pipelined data path delay is given by $T_A + [1 + log_2(m)]T_X$. Accordingly, the clock period in this case for m = 256, is $(1\tau) + ((1 + 8) * 2\tau) = 19\tau$. For the proposed pipelined multiplier, on the other hand, the clock period is limited by the delay of the slowest stage, Stage_0 in this case, which is equal to $T_A + 3T_X$ or 7τ.

To compute the scalar multiplication, each loop iteration of the basic scalar multiplication algorithm (Algorithm 1) performs one point doubling operation and an average of 1/2 point addition operation. Table 2 shows the required number of multiplications for the two types of point operations, i.e. point addition and point doubling, for different coordinate systems (Al-Somani *et. al.* 2006). Following the approach detailed in (Al-Somani *et. al.* 2006) and considering various coordinate systems, the average number of multiplications per a single loop iteration of the basic scalar multiplication algorithm is given in Table 3 for cases where a single multiplier, 2 parallel multipliers, 3 parallel multipliers, and 4 parallel multipliers are available.

5. Results

Table 4 compares the minimum clock period required for the proposed pipelined multiplier with that of the non-pipelined Wu *et. al.* (2002) multiplier. In this table, **P** and **Non-P** refer to the **P**ipelined and **Non-P**ipelined multiplier respectively. The comparisons in Table 4 are based on the average number of multiplication cycles provided in Table 3.

The results of Table 4 show that using 1 pipelined multiplier provides higher throughput than using 3 non-pipelined multipliers in all coordinate systems. It also shows that using two pipelined multipliers provide higher throughput than using four non-pipelined multipliers. Furthermore, Table 5 compares the proposed pipelined multiplier with the non-pipelined Wu *et. al.* (2002) in terms of the AT^2 performance metric and the

[1] Three libraries of different technologies were considered including TSMC 0.25 technology, the AMS 0.8 and the ES2 ECPD07 process.

results of Table 5 are depicted in Figure 3. Clearly, the results of Table 5 show that the proposed pipelined multiplier outperforms the non-pipelined Wu *et. al.* (2002) multiplier. The results of Table 5, alongside Table 4, show that using 1 pipelined multiplier provides better AT^2 than using 3 non-pipelined multipliers and using 2 pipelined multiplier provides better AT^2 than using 4 non-pipelined multipliers, which represents better than 50% improvement.

6. Conclusion

In this paper, the effect of a high performance pipelined $GF(2^{256})$ bit-serial multiplier on elliptic curve point operations has been investigated. A 3-stage pipelined version of the Wu *et. al.* (2002) $GF(2^m)$ redundant representation multiplier for $160 \leq m \leq 256$ was studied in terms of area overhead and throughput improvement. The proposed pipelined architecture has been shown to have a significant improvement in throughput allowing a single 3-stage pipelined multiplier to have higher throughput than an architecture employing three parallel non-pipelined multipliers. The AT^2 performance metric has shown an even more significant improvement.

Acknowledgment

The author would like to acknowledge the support of Umm Al-Qura University.

References

Al-Somani, T., Ibrahim, M. & Gutub, A. 2006. “High Performance Elliptic Curve GF(2^m) Crypto-Processor,” Information Technology Journal, 5(4), pp. 742-748.

Al-Somani, T. 2008. “Overlapped Parallel Computations of Scalar Multiplication with Resistance against Side Channel Attacks,” International Journal of Information and Computer Security (IJICS), Vol. 2, No. 3, pp. 261-278.

Al-Somani, T. 2010. Performance Evaluation of Elliptic Curve Projective Coordinates with Parallel GF(p) Field Operations and Side-Channel Atomicity, Journal of Computers, Vol. 5, No. 1, pp. 99-109.

Blake, I., Seroussi, G. & Smart, N. 1999. Elliptic Curves in Cryptography, Cambridge University Press: New York.

El Gamal, T. 1985. “A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms,” Advances in Cryptology: Proceedings of CRYPTO 84, Springer Verlag, pp. 10-18.

Gutub, A. & Ibrahim, M. 2003a. “High Performance Elliptic Curve GF(2^k) Cryptoprocessor Architecture For Multimedia,” Proc. IEEE International Conference on Multimedia & Expo, ICME 2003, Baltimore, Maryland, USA, pp. 81- 84.

Gutub, A. & Ibrahim, M. 2003b. “High Radix Parallel Architecture For GF(p) Elliptic Curve Processor,” Proc. IEEE Conference on Acoustics, Speech, and Signal Processing ICASSP 2003, Hong Kong, pp. 625-628.

Gutub, A., Ibrahim, M. & Al-Somani, T. 2007. “Parallelizing Gf(P) Elliptic Curve Cryptography Computations For Security And Speed,” The Proc. of the International Symposium on Signal Processing and its Applications ISSAP2007, Sharjah, United Arab Emirates (U.A.E.).

Hankerson, D., Menezes, A. & Vanstone, S. 2004. Guide to Elliptic Curve Cryptography, Springer-Verlag.

Koblitz, N. 1987. “Elliptic curve cryptosystems,” Mathematics of Computation, vol. 48, pp. 203-209.

Lidl, R. & Niederreiter, H. 1994. Introduction to finite fields and their applications, Cambridge University Press, Cambridge, UK.

Lopez, J. & Dahab, R. 1998. “Improved Algorithms for Elliptic Curve Arithmetic in GF(2^n),” SAC'98, LNCS Springer Verlag, pp. 201-212.

McEliece, R. 1987. Finite Fields for Computer Scientists and Engineers, Kluwer Academic Publishers, 1987.

Rivest, R. Shamir, A. & Adleman, L. 1978. “A method for obtaining digital signatures and public key cryptosystems,” Communications of the ACM, Vol. 21, No.2, pp. 120-126.

Thompson, C. D. 1980. A complexity theory for VLSI. Ph.D, dissertation, Carnegie Mellon University, Dep. Computer Science.

Wu, H., Hasan, A., Blake, I. & Gao, S. 2002. “Finite Field Multiplier Using Redundant Representation,” IEEE Trans. Computers, vol. 51, No. 11, pp. 1306-1316, 2002.

Table 1: Area Metrics of the proposed pipelined multiplier and the non-pipelined Wu *et. al.* (2002) multiplier.

Multiplier	**Area (gates)**
The Proposed Pipelined	$(5m + 286)$U
Non-Pipelined Wu *et. al.* multiplier.	$(5m - 2)$U

Table 2: Number of Multiplications in Different Coordinate Systems (Al-Somani *et. al.* 2006).

Projective Coordinates (Pr)		**Jacobian Coordinates (J)**		**Lopez-Dahab Coordinates (L-D)**	
Add	Double	Add	Double	Add	Double
16M	7M	15M	5M	14M	5M

Table 3: Multiplication cycles for the coordinate systems using the non-pipelined Wu *et. al.* (2002) multiplier (Al-Somani *et. al.* 2006).

Coordinate System	**Average No. of Multiplication cycles**			
	1 Multiplier	**2 Multipliers**	**3 Multipliers**	**4 Multipliers**
Pr	15	8	6	4
J	12.5	7	5	4.5
L-D	12	6.5	5.5	5

Table 4: Minimum clock period for the pipelined versus the non-pipelined Wu *et. al.* (2002) multiplier in different coordinate systems.

Coordinate System	Time (gate time)							
	1 Mult.		2 Mults		3 Mults		4 Mults	
	P	Non-P	P	Non-P	P	Non-P	P	Non-P
Pr	105τ	285τ	56τ	152τ	42τ	114τ	28τ	76τ
J	87.5τ	237.5τ	49τ	133τ	35τ	95τ	31.5τ	85.5τ
L-D	84τ	228τ	45.5τ	123.5τ	38.5τ	104.5τ	35τ	95τ

Table 5: Comparison of results with different key sizes using the AT^2 performance metric.

m (bits)	Coordinate System	AT^2							
		1 Multiplier		2 Multipliers		3 Multipliers		4 Multipliers	
		P	Non-P	P	Non-P	P	Non-P	P	Non-P
160	Pr	11973150	64817550	3405696	18436992	1915704	10370808	851424	4609248
	J	8314688	45012188	2607486	14115822	1330350	7201950	1077584	5833580
	L-D	7662816	41483232	2248292	12171296	1609724	8714360	1330350	7201950
180	Pr	13075650	72940050	3719296	20747392	2092104	11670408	929824	5186848
	J	9080313	50652813	2847586	15884722	1452850	8104450	1176809	6564605
	L-D	8368416	46681632	2455317	13696521	1757949	9806385	1452850	8104450
200	Pr	14178150	81062550	4032896	23057792	2268504	12970008	1008224	5764448
	J	9845938	56293438	3087686	17653622	1575350	9006950	1276034	7295630
	L-D	9074016	51880032	2662342	15221746	1906174	10898410	1575350	9006950
220	Pr	15280650	89185050	4346496	25368192	2444904	14269608	1086624	6342048
	J	10611563	61934063	3327786	19422522	1697850	9909450	1375259	8026655
	L-D	9779616	57078432	2869367	16746971	2054399	11990435	1697850	9909450
256	Pr	17265150	103805550	4910976	29526912	2762424	16608888	1227744	7381728
	J	11989688	72087188	3759966	22606542	1918350	11533950	1553864	9342500
	L-D	11049696	66435552	3242012	19492376	2321204	13956080	1918350	11533950

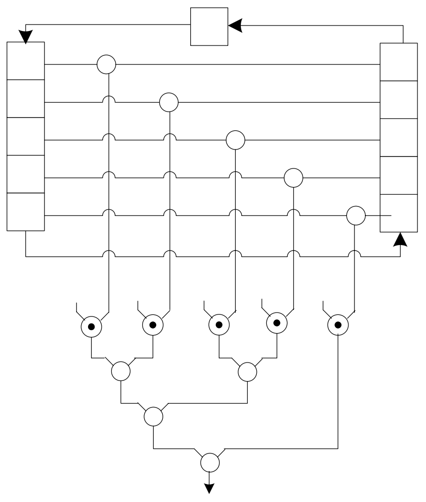


Figure1: $GF(2^5)$ bit-serial Wu *et. al.* multiplier.

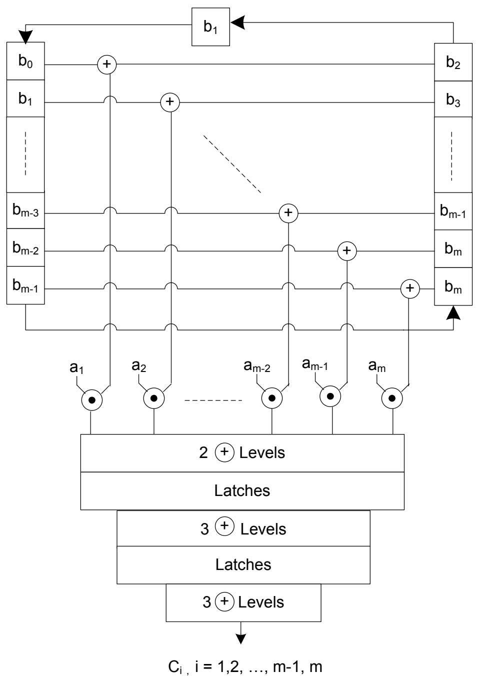
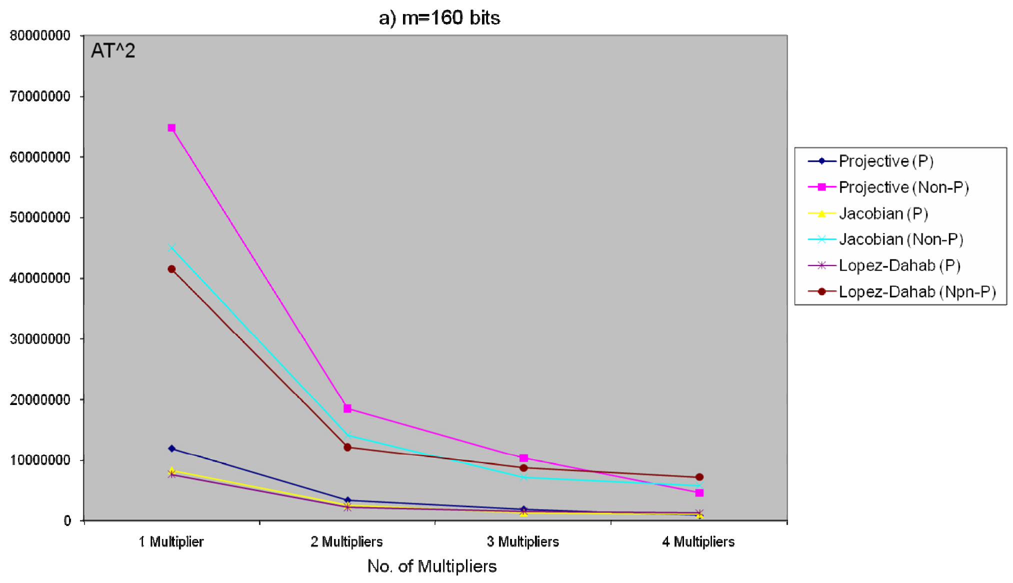
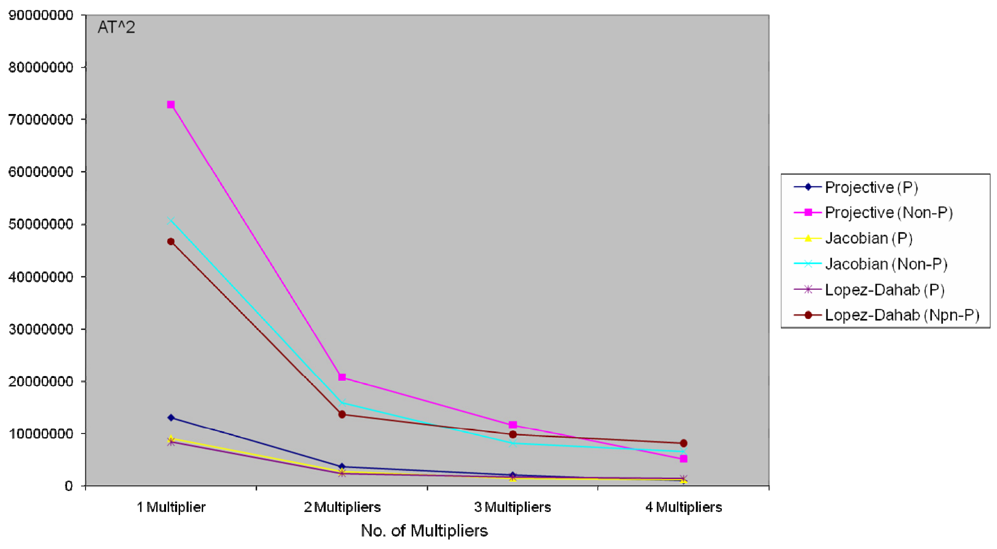


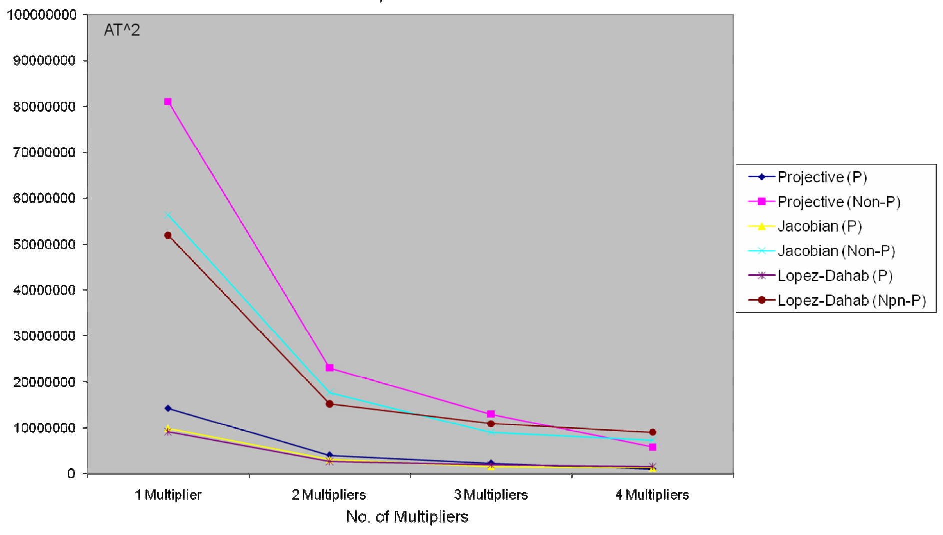
Figure2: Proposed pipelined bit-serial multiplier.

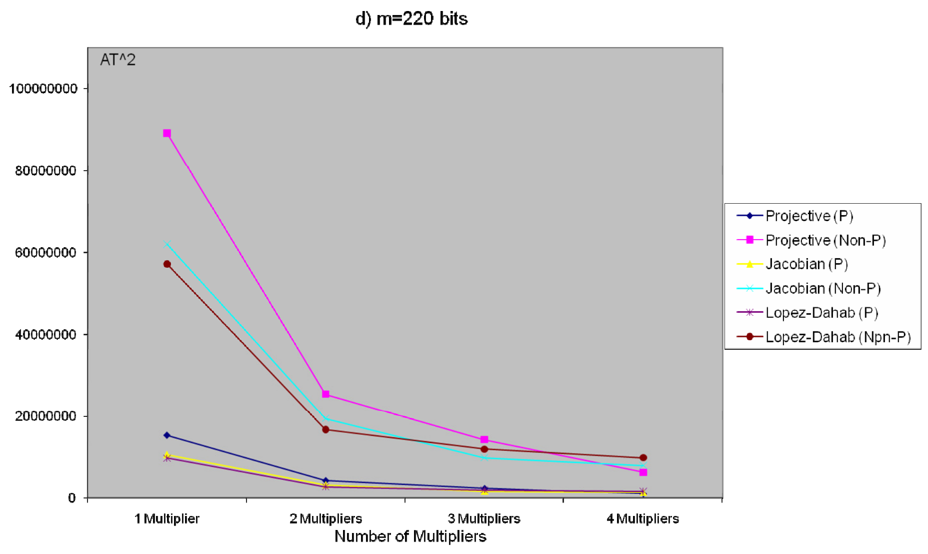


b) m=180 bits



c) m=200 bits





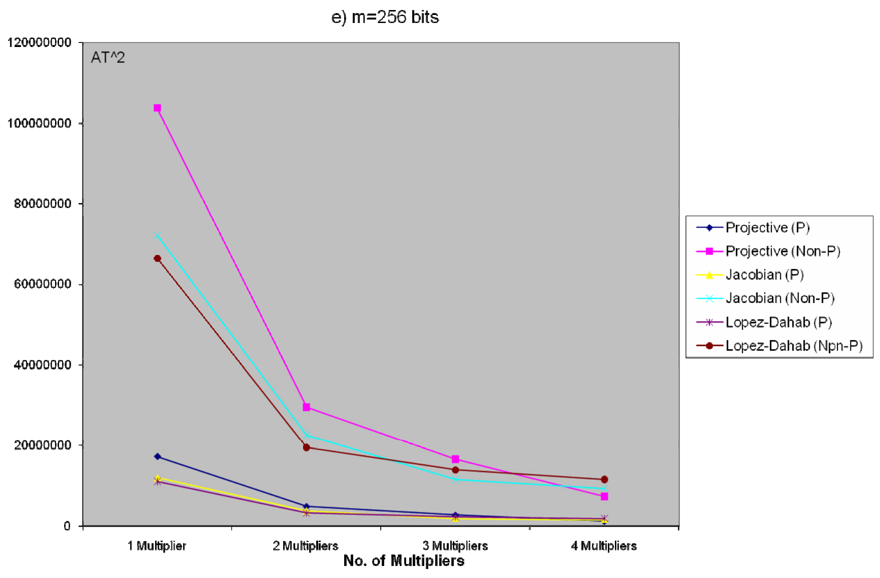


Figure 3: AT^2 Comparison results for different key sizes.

Received 10/6/1431;24/5/2010, accepted 1 /6 /1432; 4/5/2011