

# A High Speed and Efficient Method of Elliptic Curve Encryption Using Ancient Indian Vedic Mathematics

Himanshu Thapliyal

Centre for VLSI and Embedded System Technologies  
International Institute of Information Technology,  
Hyderabad, 500019, India  
(thapliyalhimanshu@yahoo.com)

M.B Srinivas

Centre for VLSI and Embedded System Technologies  
International Institute of Information Technology,  
Hyderabad, 500019, India  
(srinivas@iiit.net)

**Abstract**—This paper presents efficient hardware circuitry for point doubling using square algorithms of Ancient Indian Vedic Mathematics. In order to calculate the square of a number, “Duplex” D property of binary numbers is proposed. A technique for computation of fourth power of a number is also being proposed. A considerable improvement in the point additions and doubling has been observed when implemented using proposed techniques for exponentiation

## I. INTRODUCTION

RSA and ECC (Elliptic Curve Arithmetic) are the two major standards used for public-key cryptography [1,2]. The key length of the RSA has increased over recent years and this has put heavier processing loads on application using RSA. Nowadays, ECC is emerging as a new generation of cryptosystems based on public key cryptography as it offers equal security for a smaller key size thereby reducing processing overhead [3,4,9,10,11,12]. The benefits of ECC, when compared with classical cryptosystems such as RSA, include higher speed, lower power consumption and smaller certificates, which are especially useful for wireless applications. The major time consuming arithmetic operations operation in ECC are point additions and doubling as exponentiation operations like square, cube and fourth power occur in these operations. This paper presents efficient hardware circuitry for point doubling using an algorithm for computation of square and fourth power of a number from Indian Vedic Mathematics.

## II. ELLIPTIC CURVE ARITHMETIC

An Elliptic curve is defined by an equation in two variables, with coefficients. The variables and coefficients are restricted to elements in finite field, which results in a finite Abelian group. The most important elliptic curve equations are  $y^2+xy=x^3+ax^2+b$  (Weierstrass equation in GF(2m)) and  $y^2=x^3+ax+b$  (Weierstrass equation in GF(p)). Figure 1 gives

the hierarchy of arithmetic operations involved in the elliptic curve arithmetic [4,5].

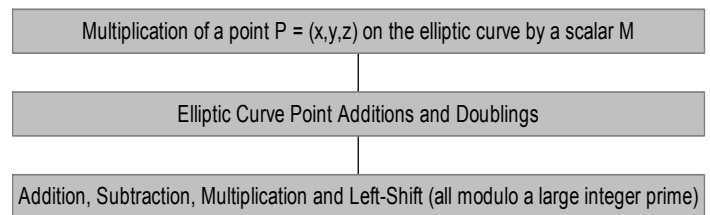


Figure 1. Elliptic Curve Arithmetic Hierarchy

Thus, it is evident from the Fig.1 that point additions and doubling are the prominent operations in ECC.

### A. Point Doubling (Using Projective Co-Ordinate System)

In the GF (P), the point doubling can be represented as follows

**Input:**  $P_1 = (X_1, Y_1, Z_1)$

**Output:**  $P_3 = (X_3, Y_3, Z_3)$

$$\lambda_1 = 3X_1^2 + aZ_1^4,$$

$$\lambda_2 = 4X_1Y_1^2,$$

$$X_3 = \lambda_1^2 - 2\lambda_2,$$

$$Z_3 = 2Y_1Z_1,$$

$$\lambda_3 = 8Y_1^4,$$

$$Y_3 = \lambda_1(\lambda_2 - X_3) - \lambda_3$$

**Return**  $(X_3, Y_3, Z_3)$ ;

Each elliptic curve addition and doubling requires a fixed number of modular multiplications, square, additions, shifts, and similar basic arithmetic operations. The actual number of these operations depends on the way the curve is represented; usually it is the multiplications, squares and fourth power operations that dominate the running time and running time will scale exactly with the number of arithmetic operations needed. Thus, it is evident from the above algorithm that square, cube and fourth power computation

are prominent operations in the computation of the point doubling, and efficient exponentiation architectures will significantly improve its computation time. In order to satisfy this need, this paper proposes a novel square and fourth power computation algorithm based on Vedic Mathematics and their hardware realization in the point doubling circuitry.

### III. PARALLEL SQUARING UNIT

In most of the computations, the multiplier unit is used to compute the square of an operand. Since square is a special case of multiplication, a dedicated square hardware will significantly improve the computation time. A parallel square unit was proposed in [7,8] having faster area and lower latency time. The partial product array for the multiplier shown in the upper half of Fig. 2 is efficiently reduced to perform square operation as shown in the lower half of the figure below by combining the equivalent terms. Thus, there is a significant reduction in the partial product array of square, making square hardware much smaller than multiplier. Furthermore, square can be computed much faster than square.

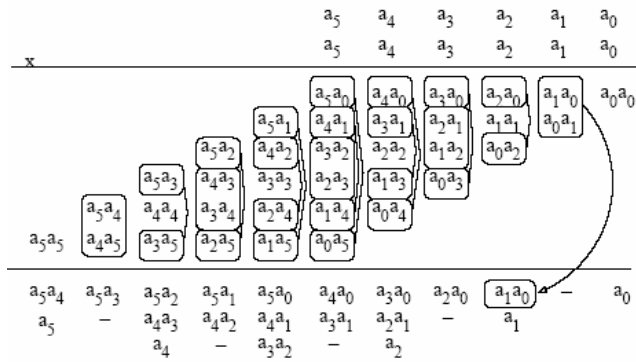


Figure 2. Square Algorithm proposed in [7,8]

### IV. PROPOSED SQUARE ALGORITHM

In order to calculate the square of a number “Duplex” D property of binary numbers has been proposed. In the Duplex, we take twice the product of the outermost pair, and then add twice the product of the next outermost pair, and so on till no pairs are left. When there are odd number of bits in the original sequence there is one bit left by itself in the middle, and this enters as such. Thus,

1. For a 1 bit number, D is the same number i.e  $D(X0)=X0$ .
2. For a 2 bit number D is twice their product i.e  $D(X1X0)=2 * X1 * X0$ .
3. For a 3 bit number D is twice the product of the outer pair + the e middle bit i.e  $D(X2X1X0)=2 * X2 * X0+X1$ .

4. For a 4 bit number D is twice the product of the outer pair + twice the product of the inner pair i.e  $D(X3X2X1X0)=2 * X3 * X0+2 * X2 * X1$

The pairing of the bits 4 at a time is done for number to be squared.

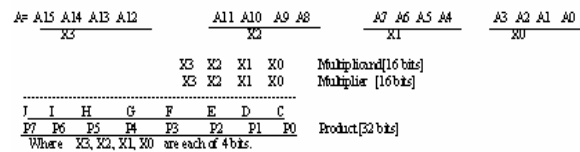
Thus  $D(1)=1$ ;

$$D(11)=2 * 1 * 1;$$

$$D(101)=2 * 1 * 1+0;$$

$$D(1011)=2 * 1 * 1+2 * 1 * 0;$$

The proposed square architecture is an improvement over partition multipliers in which the  $N \times N$  bit multiplication can be performed by decomposing the multiplicand and multiplier bits into  $M$  partitions where  $M=N/K$  (here  $N$  is the width of multiplicand and multiplier (divisible by 4) and  $K$  is a multiple of 4 such as 4, 8, 12, 16, .....  $4 * n$ ). The partition multipliers are the fastest multipliers implemented in the commercial processors and are much faster than conventional multipliers. In Fig. 3, the proposed square algorithm is explained adopting the partitioning scheme of 4 bits but improving its efficiency by applying the Duplex property of binary numbers. The proposed square algorithm can be generalized for any partition size.



PARALLEL COMPUTATION & METHODOLOGY									
1. $D(X0) = X0 * X0 = A$									
2. $D(X1X0) = 2 * X1 * X0 = B$									
3. $D(X2X0X1) = 2 * X2 * X0 + X1 * X1 = C$									
4. $D(X3X2X1X0) = 2 * X3 * X0 + 2 * X2 * X1 = D$									
5. $D(X3X3X1) = 2 * X3 * X1 + X2 * X2 = E$									
6. $D(X3X2) = 2 * X3 * X2 = F$									
7. $D(X3) = X3 * X3 = G$									

Figure 3. 16 x 16 bit Proposed Square Algorithm Using Duplex Square Algorithm

### V. PROPOSED TECHNIQUE FOR CALCULATING FOURTH POWER OF A NUMBER

This Paper also proposes an algorithm for calculating the fourth power of a number. The technique is developed as an extension of the duplex and Anurupya Sutra of Vedic Mathematics. If  $a$  and  $b$  are two digits, then according to proposed technique, the number  $M$  of  $N$  bits whose 4th power is to be calculated can be decomposed into two number  $a$  &  $b$  of  $N/2$  bits. Now, the 4th power of  $M$  can be calculated as follows.

$$M=(a+b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$$

The number can again further be decomposed until one can get the product through the smallest available 4x4 or 8x8 multiplier.

## VI. VERIFICATION AND IMPLEMENTATION

In this study, the algorithm is implemented in Verilog HDL and logic simulation is done in Veriwell Simulator; the synthesis and FPGA implementation is done using Xilinx Webpack 6.1. After gate-level synthesis from high level behavioral and/or structural RTL HDL codes, basic schematics are optimized. The design is optimized for speed and area using Xilinx, Device Family : VirtexE, Device : XCV300e, Package: bg432, Speed grade: -8. The device is made up of multiplexers and LUTs.

## VII. RESULT AND DISCUSSION

The proposed square architecture achieves significant improvement in performance over the square proposed in [7,8]. The results have been tested for partition size of 4. Table 1 shows the synthesis results of the square algorithm of [7,8] and the proposed Vedic Square. For the latest Xilinx, VirtexE family, it is observed that for 8 bit, the gate delay of the proposed square architecture is 15.193 ns with area of 190(device utilized) while it is 30.370 ns for the square proposed in [7,8] with area of 177. As the operand width is increased to 16, the gate delay of the proposed square architecture is 23.6 ns with area of 751(device utilized) while it is 60.646 ns for the square algorithm proposed in [7,8] with area of 727. Table 2 shows the synthesis results of the point doubling hardware embedding the square architectures. When 8 bit square architecture proposed in [7,8] is embedded in the point doubling architectures, the delay of the point doubling hardware is 604.81 ns with area of 25599 while it is 542.35 ns with area of 26290 for the proposed Vedic architecture. For 16 bit square architecture proposed in [7,8] the gate delay of the point doubling hardware was found to be 1327.809 ns with area of 96663, while the delay is 1207.677 ns with area of 96805 embedding the Vedic square architecture. Table-3 shows the comparison of the point doubling implementing the proposed method of fourth power computation, with the computation of the point doubling, using proposed square also for computation of fourth power. When implementing the point doubling hardware using the proposed method for fourth power computation, for 8 bit its delay is 537.885 ns with area of 33828 while for 16 bit its delay is 1348.971 ns with area of 139059. Thus, it can be concluded that the proposed method of fourth power computation is efficient for only small bit sizes while for higher bit sizes, the proposed method of square computation should be used also for the computation of the fourth power.

## VIII. CONCLUSION

The point doubling circuitry implemented with Vedic square algorithm exhibits improved efficiency in terms of speed and area. In FPGA, the proposed technique for

computation of fourth power should be used for small bit sizes only. For higher bit sizes, it is better to perform the fourth power operation by recursively using the proposed square rather than directly calculating using the proposed technique. The proposed techniques of square and fourth power computation will be highly beneficial for low power operation as the sub-modules can be switched on and off based on the requirement. Due to its parallel and regular structure the proposed architectures can be easily laid out on silicon and can work at high speed without increasing the clock frequency. It has the advantage that as the number of bits increases its gate delay and area increase very slowly as compared to point doubling circuitry employing traditional square algorithms. It is found that the design are quite efficient in terms of silicon area and speed and should result in substantial savings of resources in hardware when used for crypto and security applications.

## REFERENCES

- [1] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, 21 (2), pp. 120-126, February 1978
- [2] Walter C.D.; *Exponentiation Using Division Chains*; IEEE Transactions on Computers, IEEE Inc.; New York, U.S.; vol. 47, No. 7; Jul. 1, 1998, pp. 757-765.
- [3] Sangook Moon,"Elliptic Curve Scalar Point Multiplication Using Radix-4 Booth's Algorithm", *International Symposium on Communications and Information Technologies 2004(ISCIT 2004)*,pp. 80-83, Japan, October 26-29,2004.
- [4] Siddaveerasharan Devarkal and Duncan A. Buell," Elliptic Curve Arithmetic", *Proceedings, MAPLD 2003*.
- [5] Duncan A. Buell, James P. Davis, and Gang Quan, "Reconfigurable computing applied to problems in communication security," *Proceedings, MAPLD 2002*.
- [6] *Energy Scalable Reconfigurable Cryptographic Hardware for Portable Applications*. James Ross Goodman, PhD Dissertation, MIT.
- [7] Albert A. Liddicoat and Michael J. Flynn, "Parallel Square and Cube Computations", *34th Asilomar Conference on Signals, Systems, and Computers*, California, October 2000.
- [8] Albert Liddicoat and Michael J. Flynn," Parallel Square and Cube Computations", *Technical report CSL-TR-00-808*, Stanford University, August 2000.
- [9] Karatsuba A.; Ofman Y. *Multiplication of multidigit numbers by automata*, *Soviet Physics-Doklady* 7, p. 595-596, 1963
- [10] Bailey, D. V. and Paar, C.," Efficient Arithmetic in Finite Field Extensions with Application in Elliptic Curve Cryptography", *Journal of Cryptology*, vol. 14, no. 3, 153-176. 2001
- [11] A.Weimerskirch, C.Paar and S.C.Shantz (2001), "Elliptic Curve Cryptography on a Palm OS Device", *Proceeding of 6th Australasian Conference on Information Security and Privacy (ACISP 2001)*, 11-13 July 2001, Macquarie University, Sydney, Australia.
- [12] M. Rosner, *Elliptic Curve Cryptosystems on reconfigurable hardware*. Master's Thesis, Worcester Polytechnic Institute, Worcester, USA, 1998.

**Table I. Comparison Results of Square**

Name of Multiplier		Vendor	Device Family & Device	Package	Speed Grade	Cell Use	Estimated Delay (ns)
Square Proposed by Flynn et.al	8 x 8 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	177	30.370
	16 x 16 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	727	60.646
Proposed Square	8 x 8 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	190	15.193
	16 x 16 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	751	23.600

**Table II. Comparison Results of Point Doubling**

Point Doubling Using Square		Vendor	Device Family & Device	Package	Speed Grade	Cell Use	Estimated Delay (ns)
Using Square Proposed by Flynn et.al	8 x 8 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	25599	604.861
	16 x 16 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	96663	1327.809
Using Proposed Square	8 x 8 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	26290	542.325
	16 x 16 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	96805	1207.677

**Table III. Results of Point Doubling When Using Proposed Technique for computation of 4th Power**

Point Doubling		Vendor	Device Family & Device	Package	Speed Grade	Cell Use	Estimated Delay (ns)
Using Proposed Square	8 x 8 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	26290	542.325
	16 x 16 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	96805	1207.677
Using Proposed Fourth Power Technique	8 x 8 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	33828	537.885
	16 x 16 bit	Xilinx	VirtexE Xcv300e	Bg432	-8	139059	1348.971