INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Area Efficient Modular Multiplier For Cryptography Applications

Prema.C
Sri Ramakrishna Engineering college, Vatamalaipalayam, Coimbatore, India
Manikandababu.C.S
Sri Ramakrishna Engineering college, Vatamalaipalayam, Coimbatore, India

Abstract:

The Modular multiplier is the basic building block of cryptograpic processors. This is used widely for unique applications like network security and high speed performance in multi-core processors. It mainly requires efficient and reliable hardware implementations. The classical methods have limitations in size of intermediate quotient. To overcome this, Column compression multiplier algorithm is used with the parallel implementation of Barett reduction and Montgomery reduction and Karatsuba algorithm. These are basic mathematical algorithms. Applying digit serial implementation among these three multipliers, low area is obtained compared to the existing multiplier algorithms. The experimental results shown here, which is obtained from XILINX ISE,SPARTAN 2E Family, mainly concentrates on area by considering the number of gates used for the digit serial implementation.

Key words: *Modular multiplier, Column compression, Elliptic curve cryptography, Public key cryptography .*

1.Introduction

The crucial operation in cryptography is generating the key. There are two types, secret key and public key. ECC and RSA algorithms are used to generate the keys. In ECC 160 bits key gives efficient security than 1024 bits in RSA. Likewise 224 bit key in elliptic curves provide same levels of security as 2048 bit key in RSA. The applications of Elliptic curve to cryptography was discovered by Koblitz and Miller. In ECC the factorization to primes of a composite number in elliptic fields modulo EF(n),is more difficult than the traditional methods over Galois Field GF(n). Hence the modular multiplier to generating the public key in ECC is very important for key management with large users.

This paper is structured as follows. Section 2 describes outlines basics, which are necessary for further discussion. In Section 3, explains our existing algorithm and section 4 discusses the trade-off between area and delay of our proposed algorithm, compared with existing algorithms. Hardware implementation results are discussed in Section 5 concludes the paper.

2.Review Of Related Work

In this paper, three type of algorithms used. The existing methods based on the Barret reduction, Montgomery modular multiplication, karatsuba's multiplication. Actually the multiplier is divided into two parts. The upper part is done by the classical method and the lower part of the multiplier is done by the Montgomery modular multiplier.The speed of the multiplier is increased by increasing parallelism among those by using karatsuba multiplier. Because the multiplicand also divided as MSB parts and LSB parts separately. Hence for long integer multiplications the complexity $o(n^2)$ will be reduced as $o(\log_2^3)$. The two elements are A and B, take modulus m then the required multiplication will be as follows,

$$A \times B = A.B \bmod M \qquad (1)$$

The modulus m value is an n-bit word, each word in the radix $r=2^k$.depends on word *i-th* of b is represented as b_i here i=0,1,…,n-1. Hence,

$$b = \sum_{i=0}^{n-1} b_i . r^i \qquad (2)$$

By the means of fully-parallel implementation we consider a very fast, one-clock-cycle modular multiplier. In order to execute Barrett or Montgomery multiplication in a single clock cycle, we need three n × n-bit multipliers, which is equivalent to the size of $n/2 \times n/2$ bits multipliers. The fully-parallel architectures supporting Barrett and Montgomery algorithms are illustrated in Fig.1 , respectively. We implement n = 16-bit modular multipliers in both cases.

2.1.Barrett Reduction

Input: M: $r^{n-1} < M < r^n$

A,B : $0 \leq A,B < M$

Output: S = A . B mod M (3)

Algorithm: S:= 0;

for *i* := *n-1* down to 0 do

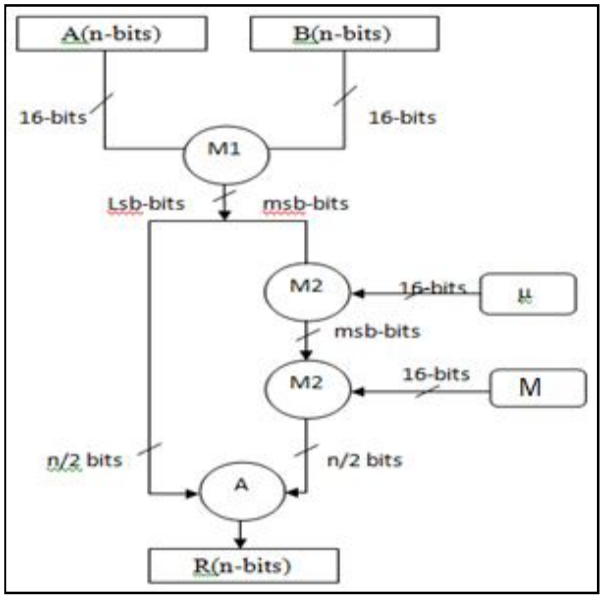
S:= r . s + $b_{i.}$ a;

$q_u := \lfloor S/M \rfloor$;

S: = M – Q_u . M;

End for

To reduce the complexity, separated the modular multiplication into two parallel processes. First one is multiplication process this is done for computing A × B,



Figur1: Parallel implementation for classical algorithm

while the later is is used to perform modular reduction by considering only one bit of at a time.

The intermediate quotient Q_u is done by integer division. Due to memory the division operation is considered as an expensive operation. To reduce the complexity of division pre-computed value μ is used and it takes the value as μ = 1/M [2]. Finally the subtraction gives nearly n-bits. This is implemented in finite fields [11][14] and also using ring of integers[5]. Fig 2.1 shows the parallel implementation of the Barrett reduction algorithm.

2.2.Montgomery Modular Multiplier

This algorithm mainly developed to avoid long integer division operation. Montgomery's algorithm is the most commonly utilized modular multiplication algorithm today. In contrast to the classical modular multiplication, it utilizes right to left divisions [10].The Montgomery multiplication of two images is defined as,

$$A * B = ABR^{-1} \bmod M \qquad (4)$$

The result is the image of U V mod M and needs to be converted back at the end of the process. For the sake of efficiency, one usually uses $R = r^n$ where $r = 2^w$ is the radix of each digit. For example, if p=11 then r=16 and r-1 =9. Because r. r^{-1} (mod 11) = 1. Let a = 5 and b = 4. Then a′= 5.16 (mod 11) = 3 and b′= 4.16 (mod11) = 9. Let c = a.b (mod p). Therefore c′=3.9.9(mod11)=1 and c=1.9(mod11)=9.The Montgomery modular reduction is faster than the regular reduction (divided by p). The total execution time will be reduced by performing more multiplication with the same operands. This is because of the p-residue needs to only being calculated once for each number. It can also be applicable for binary fields.

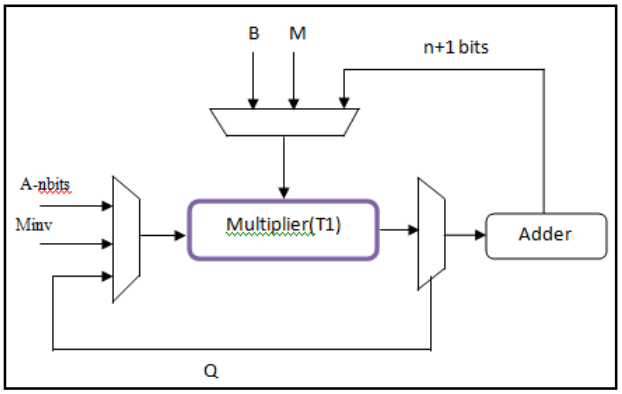


Figure2: Digit-serial implementation for Barrett and Montgomery algorithms

3.Bipartite Modular Multiplication

This section presents an existing modular multiplication. A new representation of residue classes modulo M used for performing the calculation. This is introduced by Montgomery which requires the constant R to be co prime to M and R>M and defined a new residue class representation using a new constant R= r^αn[5].

As per this multiplier the multiplicand is divided into two parts. The msb parts are done by Barrett reduction method or classical method and lsb bits are done by Montgomery method. Finally we get reduced number of bits. The only drawback is due to the intermediate value the storage of value is large.

Regular system	**Bipartite system**
$A \times B \equiv A \bullet B \bmod M$	$A \equiv X \bullet r \wedge \alpha n (\bmod M)$ $A \times B \equiv X \bullet Y \wedge (\alpha n) \bmod M$ $B \equiv Y \bullet r \wedge (\alpha n) (\bmod M)$

Table1: Transformation from original system to the new system

The table represents the transformation from original representation to the new system is accomplished by performing conventional modular multiplication between the integer value into the constant r^αn.Modular multiplication can be efficiently computed using this residue classes. Let us consider the multiplier B split into two parts $B=B_H$ .r^ αn+ B_L. therefore the images A and B using modulo M will be calculated as follows,

$A * B = (A \times BH + A * BL) \bmod M$ (5)

From equation (5),the left side is calculated by using Barrett reduction that process BH from the MSBs first, while the second term is calculated using the Montgomery algorithm which processes BL from the LSBs first. Both calculations performed parallel. Compare with operand B, BH and BL are shorter in length. Hence A x BH, A * BL are performed faster than the execution of the Barrett algorithm and the Montgomery algorithm with the unsplit operands.

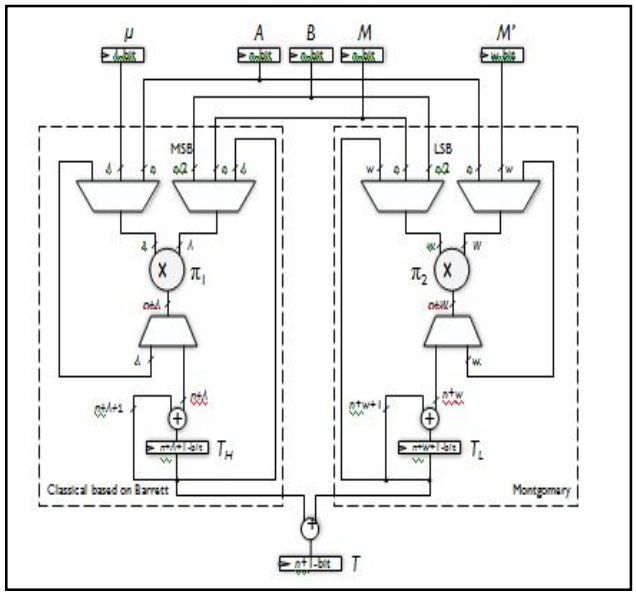


Figure 3: Digit-serial implementation for Bipartite algorithm

4. Digit-Serial Architecture For Tripartite Algorithm

Here, an existing modular multiplication algorithm used for encryption and decryption processes. The karatsuba algorithm used to reduce the sub-word multiplication and also it has complexity of $O(n\log_2^3)$. This algorithm can accelerate multiplication by representing two nw integers as,

$A = A_1R + A_0$ *and* $B = B_1R + B_0$,

Where, R = 2^k is chosen for an efficient implementation. Then, the product of AB can be computed as,

$$A * B = P_1 R_2 + (P_2 - P_0 - P_1) R + P_0 \qquad (6)$$

Where,

$P_0 = A_0 B_0$, $P1 = A_1 B_1$, $P = (A_0 + A_1)(B_0 + B_1)$

Therefore, we need three partial products by effectively using karatsuba algorithm. So the complexity in calculation of partial products will be reduced. Compare to other mathematical algorithms karatsuba which reduces the critical path in the main architecture of our algorithm. The generation of karatsuba algorithm is Toom-cook algorithm, which is very useful in convolution algorithms in DSP slices. The following equation derived from Karatsuba's algorithm.

$$ABR^{-1} \bmod M = (A_1R + A_0)(B_1R + B_0)R^{-1} \bmod M \qquad (7)$$

$$= A_1B_1R + (A_1B_0 + A_0B_1) + A_0B_0R^{-1} \bmod M$$

$$= P_1R \bmod M + (P_2 - P_0 - P_1) \bmod M + P_0$$

$R^{-1} mod\ M\ mod\ M$,

Where, $P_0 = A_0 B_0$, $P1 = A_1 B_1$, $P2 = (A_0 + A_1)(B_0 + B_1)$

In this case nw -digit inputs A and B are split into two blocks as A = (A_1 , A_0)R and B = (B1 , B0)R , and then Karatsuba's method is applied for performing multiplication of AB. Here, R is chosen as R=r^k where k= dn_w /2e for an

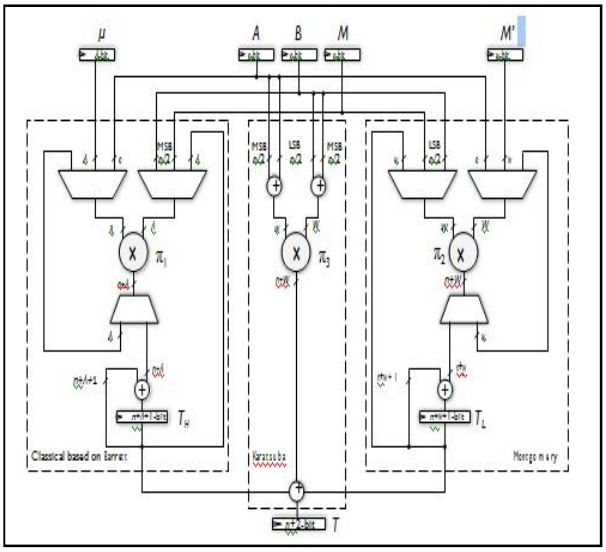


Figure4: Digit-serial architecture for tripartite multiplier [7]

efficient implementation although k can be arbitrarily chosen. This is also called as U split version. Here both multiplier and multiplicand separated as two terms like MSB part and LSB part.

In total, we have three terms that can be computed independently by using the existing algorithms described in the above equation. In that equation three different terms computed parallel. Figure 4.1 explains the main idea of the tripartite algorithm and compares with the bipartite algorithm. This architecture for the u = 2-split version [7] is depicted in Figure 4. In this architecture three type of multiplication algorithms established by the way of parallel processing. The left hand side dotted part is fully developed as per classical method that is Barrett reduction.

The computation fully depends on the precomputed value and common Modulus value. Then the second term that is right hand side dotted part is developed according to Montgomery modular multiplication method. In between classical and Montgomery method the karatsuba multiplier is successively enhance the speed of producing partial products. But the drawback is only depends on the

intermediate quotient. This will depends on the partial products. To overcome this we use column compression multiplier instead of ordinary multiplier to reduce the complexity in partial products.

4.1.Column Compression Multiplier

This is a hardware multiplier design[15]. This is slightly faster and requires fewer gates. In a parallel multiplier the partial products are generated by using array of AND gates. The main problem is the summation of the partial products, and it is the time taken to perform this summation which determines the maximum speed at which a multiplier may operate.

This scheme essentially minimizes the number of adder stages required to perform the summation of partial products. This is achieved by using full and half adders to reduce the number of rows in the matrix number of bits at each summation stage.This is a refinement of the parallel multipliers presented as Row multipliers. This is consists of three stages. The partial product matrix is formed in the first stage by N2 AND stages.In the second stage, the partial product matrix is reduced to a height of two. This will be replaced by using Wallace Pseudo adders with parallel (n, m) counters. A Parallel (n, m) counter is a circuit which has n inputs and produce m outputs which provide a binary count of the Ones present at the inputs. A full adder is an implementation of a (3, 2) counter which takes 3 inputs and produces 2 outputs. Similarly a half adder is an implementation of a (2, 2) counter which takes 2 inputs and produces 2 Outputs.

In compression multipliers that reduce the number of rows as much as possible on each layer, these multipliers have less expensive reduction phase, but the numbers may be a few bits longer, thus requiring slightly bigger adders. In a parallel multiplier, the terms yi ^ (xn-1 - . . . x0) are known as the partial products and are generated using an array of AND gates. For a parallel multiplier, the shifting term 2i is inherent in the wiring and does not require any explicit hardware. Hence the area of our multiplier is reduced in terms of gates. This leads to reduce the cell delay.

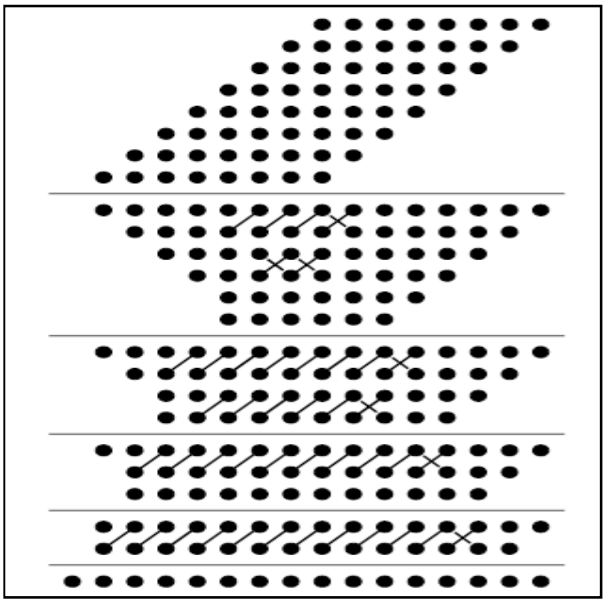


Figure 5: Dot diagram for Column Compression Multiplier

Parameters	**Bipartite**	**New Modular Multiplier**
AREA(Gate count)	8,436	7,911
POWER (mW)	69	25
DELAY (ns)	83.541	57.56

Table2: Comparison result using Mentor Graphics

5.Conclusion and Result

In this project, Bipartite and New Modular Multiplier has been designed by using XILINX 8.1, Design suit SPARTAN 2E family - target device xc2s50e7. Parallel implementation for both presented in order to reduce the complexity in multiplication. The Digit-serial implementation for our Modular Multiplier constructed by inserting karatsuba algorithm between classical and Montgomery Modular multiplier. It is used to reduce the complexity of parallel implementation and reduce the delay from 83,541 ns to 57.56 ns and also reduce the area by utilization of gates from 8,436 to 7,911. The total power is reduced from 69mW to 25 mW. Table II Comparision results shows that high speed for the presented Multiplier is obtained when compared to existing. Also low power and low area is achieved when compared to previous method. It is proved by the

HDL analysis also. In the proposed work, the Modular Multiplication is performed for 16- bits in future the work can be extended for 64 bits. The proposed modification can also be employ in Elliptic Curve Cryptography algorithm. To reduce the delay pipelining can also be incorporated in the structure.

6.Acknowledgement

The authors thank the Management and Principal of Sri Ramakrishna Engineering College, Coimbatore for providing excellent computing facilities and encouragement.

7.Reference

1. P. Barrett. “Communications Authentication and Security using Public Key Encryption - A Design for Implementation” Master Thesis, Oxford University, 1984.
2. P.Barrett. “Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor”. In Proc. CRYPTO’86, pages 311–323, 1986.
3. J.-F. Dhem. “Design of an Efficient Public-key Cryptographic Library for RISC-based Smart Cards”. Ph.D Thesis, 1998.
4. J.Fan, K. Sakiyama, and I.Verbauwhede. “Montgomery Modular Multiplication Algorithm on Multi-Core Systems”. In IEEE Workshop on Signal Processing Systems: Design and Implementation (SIPS 2007), pages 261–266, Shanghai, China, 2007. IEEE.
5. M.E.Kaihara and N.Takagi. “Bipartite Modular Multiplication”. In J.R.Rao andB.Sunar, editors, Proceedings of 7th International Workshop on Cryptographic Hardware and Embedded Systems (CHES), number 3659 in Lecture Notes in Computer Science. Springer – Verlag 2006.
6. A.Karatsuba and Y.Ofman. “Multiplication of Many-Digita Numbers by Automatic Computers”. Translation in Physics - Doklady, 145:595–596, 7 1963.
7. Kazuo Sakiyama, Miroslav Knez evic, Junfeng Fan, BartPreneel, Ingrid Verbauwhede, “Tripartite modular multiplication” , INTEGRATION, the VLSI journal 44(2011) 259–269, www.elsevier.com/locate/vlsi
8. N. Koblitz. “Elliptic Curve Cryptosystem”. Math. Comp., 48:203–209, 1987.
9. V.Miller. “Uses of Elliptic Curves in Cryptography”. In H. CWilliams, editor, Advances in Cryptology: Proceedings of CRYPTO’85, number 218 in Lecture Notes in Computer Science, pages 417–426. Springer- Verlag, 1985.
10. P. Montgomery. “Modular Multiplication without Trial Division”. Mathematics of Computation, 44(170):519– 521, 1985.
11. M.J. Potgieter and B. J. van Dyk. “Two Hardware Implementations of the Group Operations Necessary for Implementing an Elliptic Curve Cryptosystem over a Characteristic Two Finite Field”. In Africon Conference in Africa, 2002. IEEE AFRICON. 6th, pages 187–192, 2002.

12. R.L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM 21 (2) (1978) 120–126.
13. D. Suzuki, "How to maximize the potential of FPGA resources, Berlin, 2007, pp. 272–288
14. H. Wu, "Montgomery multiplier and squarer for a class of finite fields", IEEE Transactions for modular exponentiation", in: CHES '07: Proceedings of the 9th International Workshop on Cryptographic Hardware and embedded Systems, Springer-Verlag on Computers 51 (5) (2002) 521–529.
15. Whitney J.Townsend, Earl E.Swartzlander, Jr.,Jacob Abraham, "A Comparision of Dadda and Wallace multiplier delays".