IJITE INTERNATIONAL JOURNAL OF INNOVATIVE TRENDS IN ENGINEERING (IJITE) ISSN: 2395-2946
ISSUE: 63, VOLUME 43, NUMBER 02, JULY 2018

FFT-Based Montgomery Multiplication - A Brief Survey

Sanskriti Yadav[1], Prof. Sujeet Mishra[2]

[1]Mteh. Scholar, [2]Research Guide

Department of Electronics and Communication Engg., Sanghvi Institute of Management and Science, Indore

Abstract- FPGAs are gaining importance both in commercial as well as research settings. The RSA algorithm is a secure public key algorithm if the modulus size is sufficiently large. It can be used in these applications as a method of exchanging secret information such as keys and producing digital signatures. However, the RSA algorithm is very computationally intensive, operating on very large integer. The RSA algorithm has been adopted by many commercial software products and is built into current operating systems by Microsoft, Apple, Sun, and Novell. Most FFT algorithms only work when the input size is the power of a small prime. FFT-Based Montgomery Multiplication is used to provide a treatment of the FFT that takes the perspective of both mathematicians and engineers into account so that these two communities may better communicate with each other. The Montgomery exponentiation and RSA require the calculation of the modulus in various steps. The steps of the division algorithm can somewhat be simplified in order to speed up the process. The reduction step can be achieved by making one of the well-known sequential division algorithms. In this work presents an extensive survey of literature on FFT-based multiplication.

Keywords- FFT, Montgomery multiplication, Fast multiplication, RSA algorithm, FPGA.

I. INTRODUCTION

In this age of universal electronic connectivity, of electronic eavesdropping and fraud, it is of utmost importance to store information securely. This led to a heightened awareness to protect the data from disclosure, to guarantee the authenticity of data and messages, and to protect systems from network-based attacks. Cryptography plays a major role in mobile phone communications, e-commerce, pay-tv, sending private emails, transmitting financial information, security of ATM cards, computer passwords, and electronic commerce digital signature and so on. Modular exponentiation a^b mod m, and implicitly modular multiplication a • c mod m are the operations intensively used, underlying many cryptographic schemes. In this investigation, we focus only on the Rivest Shamir and Adleman (RSA) cryptosystem requirements: the execution of multiplications modulo a large number, chosen in the order of 512-2048 bits to safeguard the information, at high throughput rates.

Arithmetic operations are the often thought of to be the most understood concepts. While this seems to be true for smaller bit-lengths, the situation gets trickier in that the school- book method that works so well in implementing smaller operand-arithmetic suddenly seems slower and cumbersome in implementing the arithmetic for large operands. Now, “large” is a subjective qualifier, and it is hard to put a number on it in generic terms. What is large for one application could be not-so-large or even small for another. So it is important to outline the application which is targeting the development of the current arithmetic towards. The operation in question throughout this work is the multiplier- based modular reduction of a large integer. Application-wise, this would be a typical step in modular exponentiation – which is routinely carried out in RSA cryptography.

The fastest multiplication algorithms use the fast Fourier transform. Although the fast Fourier transform was originally developed for convolution of sequences, which amounts to multiplication of polynomials, it can also be used for multiplication of long integers.

There are many Fourier primes, i.e., primes p for which FFTs in modulo p arithmetic exist. Moreover, there exists a reasonably efficient algorithm for determining such primes along with their primitive elements. From these primitive elements, the required primitive roots of unity can be efficiently computed. This method for multiplication of long integers using the fast Fourier transform over finite fields was discovered. It is described in detail by Knuth. A careful analysis of the algorithm shows that the product of two k-bit numbers can be performed using O(k logk loglogk) bit operations. However, the constant in front of the order function is high. The break-even point is much higher than that of Karatsuba-Ofman algorithm. It starts paying off for numbers with several thousand bits. Thus, they are not very suitable for performing RSA operations.

The main objective of this document is to provide a mathematical treatment of FFT algorithms, reviewing the work of previous literatures. Many of the algorithms contained in this work have appeared in the literature.

II. MONTGOMERY MULTIPLICATION

Cryptography is the art of designing and breaking ciphers. Traditionally, secret–key cryptography was used by the military and diplomatic services for providing secure communication, in which two communicating parties share a secret key that should be distributed in some secure way.

Development of mobile internet devices increased the need for cryptographic techniques for privacy and authentication of digital data. The invention of public–key cryptography, which assigns two keys (one public and one private) to each user, provided techniques for key distribution as well as signing and authenticating digital data.

Because of its complexity, public–key cryptography is mainly used for digital signatures and the management of secret keys between two points. The encryption of bulk data is mainly established with secret–key cryptosystems, whereas the secret keys to be shared for a pair of users, are distributed by public–key cryptosystems. For a standard public–key cryptosystem to be considered as "secure", the key length should be about thousand bits or more.

In public–key cryptography, input and output numbers are selected from finite element fields. All cryptographic operations are made in these finite fields, which map to modular multiplication and modular exponentiations in the digital world. Increasing demand for modular multiplication requires fast modular multiplication algorithms such as Montgomery multiplication, which will be described thoroughly in this investigation.

Montgomery reported an algorithm for modular multiplication. This new algorithm, called Montgomery Multiplication Algorithm, has the advantage of replacing division operations by bit shift operations. If the least significant bits to be shifted out are not zero, Montgomery's algorithm adds multiples of modulus to clear these bits before shifting them out.

In regular modular multiplication, after all bits of the multiplicand are processed, modulus is repeatedly subtracted from the result unless the result is less than the modulus. In Montgomery multiplication, bits are shifted out as each bit of the multiplicand is processed, leaving no need for the subtractions. The definitions of the terms unified and dual–radix are as follows.

a. Unified Architecture:

Architecture is said to be unified when it is able to work with operands in both prime and binary extension fields using the same hardware. It has been shown that a unified multiplier is feasible with only minor modifications to the multiplier for GF(p).

Figure 2.1 shows the block representation of Montgomery multiplications algorithm.

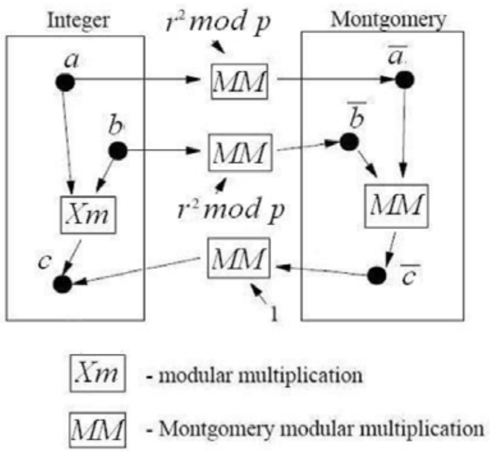

Figure 2.1 Montgomery multiplications.

b. Dual–Radix Architecture:

A unified multiplier is said to be dual–radix if it operates with a larger radix value for GF(2n) than the radix used for GF(p). The term, architecture, is used to represent the hardware of the Montgomery multiplier.

A radix–2n multiplier processes n bits of the multiplicand in every clock cycle. A radix (2,4) multiplier stands for a multiplier working in radix–2 for GF(p) and working in radix–4 for GF(2n). The new architecture in this exploration is a radix (2,4) multiplier architecture.

Dual radix multiplier design has critical time–area considerations, as the cost of extra radix should not effect the signal propagation time much while keeping the silicon area as low as possible.

III. RELATED WORK

SR. NO.	Title	Author	Year	Approach
1	Area-Time Efficient Architecture of FFT-Based Montgomery Multiplication,	W. Dai, D. D. Chen, R. C. C. Cheung and Ç. K. Koç	2017	Reported an improved FFT-based Montgomery modular multiplication (MMM) algorithm achieving high area-time efficiency
2	Parameter Space for the Architecture of FFT-Based Montgomery Modular Multiplication,	D. D. Chen, G. X. Yao, R. C. C. Cheung, D. Pao and Ç. K. Koç,	2016	Reported improvements to FFT-based Montgomery Modular Multiplication (FFTM3) using carry-save arithmetic and pre-computation techniques

3	FPGA implementation of RSA based on carry save Montgomery modular multiplication,	R. Verma, M. Dutta and R. Vig,	2016	The efficiency of modular multiplication can be improved by algorithmic improvement
4	An extension of RSA_512 to RSA_1024 core under hardware platform based on montgomery powering,	W. Hentabli and F. Merazka, "	2015	Reported an extension of RSA core from 512 key lengths to 1024 key length under hardware platform
5	FPGA implementation of modified serial montgomery modular multiplication for 2048-bit RSA cryptosystems	B. Hanindhito, N. Ahmadi, H. Hogantara, A. I. Arrahmah and T. Adiono,	2015	Reported two modular multiplication architectures based on modified serial montgomery algorithm for 2048-bit RSA
6	Modified RSA public key algorithm	B. G. Aswathy and R. Resmi,	2014	Reported two architectures for FPGA implementation of modular exponentiation algorithm based on Montgomery technique
7	High throughput parallel montgomery modular exponentiation on FPGA,	A. Nadjia and A. Mohamed,	2014	Reported a high throughput architecture implementing a fast modular exponentiation based on the square-and-multiply method

W. Dai, D. D. Chen, R. C. C. Cheung and Ç. K. Koç, [1] The modular multiplication operation is the most time-consuming operation for number-theoretic cryptographic algorithms involving large integers, such as RSA and Diffie-Hellman. Implementations reveal that more than 75 percent of the time is spent in the modular multiplication function within the RSA for more than 1,024-bit moduli. There are fast multiplier architectures to minimize the delay and increase the throughput using parallelism and pipelining. However such designs are large in terms of area and low in efficiency. In this exploration, we integrate the fast Fourier transform (FFT) method into the McLaughlin's framework, and present an improved FFT-based Montgomery modular multiplication (MMM) algorithm achieving high area-time efficiency. Compared to the previous FFT-based designs, we inhibit the zero-padding operation by computing the modular multiplication steps directly using cyclic and nega-cyclic convolutions. Thus, we reduce the convolution length by half. Furthermore, supported by the number-theoretic weighted transform, the FFT algorithm is used to provide fast convolution computation. We also introduce a general method for efficient parameter selection for the reported algorithm.

Architectures with single and double butterfly structures are designed obtaining low area-latency solutions, which we implemented on Xilinx Virtex-6 FPGAs. The results show that our work offers a better area-latency efficiency compared to the state-of-the-art FFT-based MMM architectures from and above 1,024-bit operand sizes. We have obtained area-latency efficiency improvements up to 50.9 percent for 1,024-bit, 41.9 percent for 2,048-bit, 37.8 percent for 4,096-bit and 103.2 percent for 7,680-bit operands. Furthermore, the operating latency is also outperformed with high clock frequency for length-64 transform and above.

D. D. Chen, G. X. Yao, R. C. C. Cheung, D. Pao and Ç. K. Koç, [2] Modular multiplication is the core operation in public-key cryptographic algorithms such as RSA and the Diffie-Hellman algorithm. The efficiency of the modular multiplier plays a crucial role in the performance of these cryptographic methods. In this exploration, improvements to FFT-based Montgomery Modular Multiplication (FFTM3) using carry-save arithmetic and pre-computation techniques are presented. Moreover, pseudo-Fermat number transform is used to enrich the supported operand sizes for the FFTM3. The asymptotic complexity of our method is O(l log l log log l), which is the same as the Schonhage-Strassen multiplication algorithm (SSA). A systematic procedure to select suitable parameter set for the FFTM3 is provided. Prototypes of the improved FFTM3 multiplier with appropriate parameter sets are implemented on Xilinx Virtex-6 FPGA. Our method can perform 3,100-bit and 4,124-bit modular multiplications in 6.74 and 7.78 μs, respectively. It offers better computation latency and area-latency product compared to the state-of-the-art methods for operand size of 3,072-bit and above.

R. Verma, M. Dutta and R. Vig [3] Modular multiplication determines the efficiency of RSA cryptosystem as modular multiplication is core operation in RSA. The efficiency of modular multiplication can be improved by algorithmic improvement. The long operands in Montgomery modular multiplication can be added with carry save adders. Implementation of carry save adders on FPGAs require more area. This exploration presents the implementation results of RSA on FPGAs based on carry save Montgomery.

W. Hentabli and F. Merazka, [4] A hardware implementation of RSA encryption based on Montgomery algorithm with modular multiplication and systolic array architecture is presented. In this exploration, we present an extension of RSA core from 512 key lengths to 1024 key length under hardware platform. The design uses two block multipliers as the main functional unit and Block-RAM as storage unit for the operands. To extend the core from 512 bits to 1024 bits, the design keeps the same IP-Core architecture, it will only adjust the radix used in the multipliers, and number of words to meet the system requirements such as available resources, precision and timing constraints. The architecture, based on the Montgomery modular multiplication algorithm, utilizes a pipelining technique that allows concurrent operation of hardwired multipliers.

B. Hanindhito, N. Ahmadi, H. Hogantara, A. I. Arrahmah and T. Adiono, [5] RSA (Rivest, Shamir, Adleman) is one of the most widely used cryptographic algorithms worldwide to perform data encryption and decryption. An essential step in RSA computation lies on its modular multiplication which is relatively expensive and time consuming to be implemented in hardware. This exploration reported two modular multiplication architectures based on modified serial montgomery algorithm for 2048-bit RSA. By limiting the integer modulo that has sequence of A094358, a very simple and fast modular multiplication hardware can be developed. The first archictecture which incorporates 2048-bit adders performes better in term of latency (19010 Logic Cells, 2048 clock cycles or 0.0022 s), while the second architecture utilizing multiple smaller 128-bit adders offers less area consumption (8926 Logic Cells, 36864 clock cycles or 0.0031 s). An area multiplied with squared latency (AT2) can be used as trade-off parameter for choosing the most suitable design for certain need. For prototyping purpose, we have successfully synthesized and implemented reported designs written in VHDL using Altera Quartus II with Cyclone II EP2C70F896C6 FPGA as a target board.

B. G. Aswathy and R. Resmi, [6] Modern cryptography is heavily based on mathematical theory and computer science practice, cryptographic algorithms are designed around computational hardness assumptions. Among the various techniques adopted in cryptographic technology the RSA (Rivest, Shamir and Adleman) is the most widely used public key cryptosystem. The basic operation for this algorithm is modular exponentiation. Modular multiplication is the core computation of all modular exponentiation algorithms, Montgomery's method is considered as the most efficient algorithm for performing the modular multiplication operation. This exploration presents two architectures for FPGA implementation of modular exponentiation algorithm based on Montgomery technique - one with parallel technique and another with maximum sequential operation. Finally these architectures are analyzed with respect to area, speed, and power using Xilinx ISE.

A. Nadjia and A. Mohamed, [7] Modular exponentiation is the key operation in public key cryptosystems such as RSA (Rivest, Shamir Adelman). It is implemented by repeated modular multiplications which are time consuming for large operands. Accelerating RSA requires reducing the number of modular multiplications with speeding up the modular multiplication. In this exploration, we present a high throughput architecture implementing a fast modular exponentiation based on the square-and-multiply method, called binary method which allows the parallel execution of squares and multiplications by using two fast Montgomery modular multipliers. The Montgomery multiplication is based on a high radix-216 to reduce the iterations number of this operation where the multiplication of two 1024-bits numbers is performed in only 65 iterations. The CS (Carry Save) representation is advantageously used to overcome the carry propagation then the iteration cycle is independent of the data path length. The implementation results showed that the architecture computes a 1024 bits modular exponentiation in only 0.66 ms.

IV. PROBLEM STATEMENT

Among customary finite field arithmetic operations, namely, addition, subtraction, multiplication and inversion of nonzero elements, the computation of the later is the most time-consuming one. Unfortunately enough, cryptographic designers have historically shown some resistance to use -related techniques for computing multiplicative inverses when using polynomial basis representation. Montgomery algorithm is an effective way to perform modular multiplication, while avoiding division by the large modulus. The majority of existing RSA algorithms use Montgomery method and high radix number systems. Montgomery algorithm has been also combined with RNS, which proves to be a promising alternative. The aim of this investigation is to investigate alternative base extension methods that have lower computational demands such that the Montgomery modular multiplication can be executed faster and the overall RSA performance (throughput) is improved.

V. CONCLUSION

This work presents an extensive survey of literature on efficient architecture of FFT-based Montgomery multiplication. The purpose of this work is to explore several types of FFT algorithms and to explore several common applications of the FFT algorithm. The multiplicative FFT algorithms are extensively covered in the literature and this research the objective of this work is to review various relevant algorithms. Various techniques for implementing modular multiplication were discussed in this work. This work contained background material for a study of FFT algorithms. After presenting some historical information, mathematical prerequisites, and a brief discussion on algorithm complexity, are explored.

REFERENCES

[1]. W. Dai, D. D. Chen, R. C. C. Cheung and Ç. K. Koç, "Area-Time Efficient Architecture of FFT-Based Montgomery Multiplication," in IEEE Transactions on Computers, vol. 66, no. 3, pp. 375-388, March 1 2017.

[2]. D. D. Chen, G. X. Yao, R. C. C. Cheung, D. Pao and Ç. K. Koç, "Parameter Space for the Architecture of FFT-Based Montgomery Modular Multiplication," in IEEE Transactions on Computers, vol. 65, no. 1, pp. 147-160, Jan. 1 2016.

[3]. R. Verma, M. Dutta and R. Vig, "FPGA implementation of RSA based on carry save Montgomery modular multiplication," 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), New Delhi, 2016, pp. 107-112

[4]. W. Hentabli and F. Merazka, "An extension of RSA_512 to RSA_1024 core under hardware platform based on montgomery powering," 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, 2015, pp. 448-453.

[5]. B. Hanindhito, N. Ahmadi, H. Hogantara, A. I. Arrahmah and T. Adiono, "FPGA implementation of modified serial montgomery modular multiplication for 2048-bit RSA cryptosystems," 2015 International Seminar on Intelligent Technology and Its Applications (ISITIA), Surabaya, 2015, pp. 113-118.

[6]. B. G. Aswathy and R. Resmi, "Modified RSA public key algorithm," 2014 First International Conference on Computational Systems and Communications (ICCSC), Trivandrum, 2014, pp. 252-255.

[7]. A. Nadjia and A. Mohamed, "High throughput parallel montgomery modular exponentiation on FPGA," 2014 9th International Design and Test Symposium (IDT), Algiers, 2014, pp. 225-230.

[8]. R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, pp. 120–126, 1978.

[9]. R. L. Rivest, “A description of a single-chip implementation of the RSA cipher,” Lambda, vol. 1, no. Oct.–Dec., pp. 14–18, 1980.

[10]. “Recommendation for key management,” NIST, Tech. Rep. Spe- cial Publication 800-57, Part-1, Rev.-3, 2012.

[11]. P. L. Montgomery, “Modular multiplication without trial divi- sion,” Mathematics Comput., vol. 44, no. 170, pp. 519–521, 1985.

[12]. A. Karatsuba and Y. Ofman, “Multiplication of multidigit num- bers on automata,” Soviet Physics Doklady, vol. 7, 1963, Art. no. 595.