International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Implementation of RSA Cryptosystem Using Ancient Indian Vedic mathematics

Shahina M. Salim[1], Sonal A. Lakhotiya[2]

[1] G.H. Raisoni college of Engineering and Technology, Anjangaon bari Road, Amravati.

[2] G.H. Raisoni college of Engineering and Technology, Anjangaon bari Road, Amravati.

Abstract: ***RSA is one of the most safest standard algorithm based on public key, for providing security in network. The hierarchical overlay multiplier is used in RSA circuitry for multiplication operation. The most significant aspect is the development of division architecture based on Ancient Indian Vedic Mathematics and embedding it in RSA encryption/decryption circuitry for improved efficiency. Typically, modular-multiplication algorithm is used since no trial division is necessary, and the carry-save addition (CSA) is employed to reduce the critical path. The implementation of RSA encryption/decryption algorithm using the algorithm of Ancient Indian Vedic Mathematics that have been modified to improve performance. RSA circuitry implemented using vedic multiplication is efficient in terms of area, speed compared to its implementation using conventional multiplication.***

Keywords: RSA Cryptosystem, Modular Multiplication,, Modular exponentiation, Vedic Mathematics, FPGA, VHDL..

1. Introduction

Now a day's large number of internet and wireless communication users has led to an increasing demand of security measures and devices for protecting the user data transmitted over the unsecured network so that unauthorized persons cannot access it. The history of cryptography begins with secret writings in the Ancient civilizations. Cryptography is the practice and study of techniques for secure communication in the presence of third parties (called adversaries). Cryptography is closely related to the disciplines of cryptology and cryptanalysis. Cryptography includes techniques such as microdots, merging words with images, and other ways to hide information in storage or transit. More generally, it is about constructing and analyzing protocols that block adversaries, various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation are central to modern cryptography. Modern cryptography exists at the intersection of the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce. However, in today's computer-centric world, cryptography is most often associated with scrambling plaintext (ordinary text, sometimes referred to as cleartext) into ciphertext (a process called encryption), then back again (known as decryption). Individuals who practice this field are known as cryptographers. Modern cryptography concerns itself with the following four objectives:

1) **Confidentiality** (the information cannot be understood by anyone for whom it was unintended)
2) **Integrity** (the information cannot be altered in storage or transit between sender and intended receiver without the alteration being detected)
3) **Non-repudiation** (the creator/sender of the information cannot deny at a later stage his or her intentions in the creation or transmission of the information)
4) **Authentication** (the sender and receiver can confirm each others, identity and the origin/destination of the information).

Types of Cryptographic Algorithms:
There are several ways of classifying cryptographic algorithms. The three types of algorithms are:

- Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption
- Public Key Cryptography (PKC): Uses one key for encryption and another for decryption
- Hash Functions: Uses a mathematical transformation to irreversibly "encrypt" information.

The RSA algorithm was publicly described in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT; the letters **RSA** are the initials of their surnames. The standard techniques for providing privacy and security in data networks include encryption/decryption algorithms such as Advanced Encryption System (AES) (private-key) and RSA (public- key). Rivest–Shamir–Adleman (RSA) is one of the most widely preferred algorithms used in public-key.

Cryptography systems. RSA is one of the safest standard algorithms, based on public-key, for providing security in networks. RSA has a very slow ciphering rate if used in software. Security has become an increasingly important feature with the growth of electronic communication. The development of public-key cryptography (PKC) is the greatest and perhaps the only true revolution in the entire history of cryptography. Many PKC algorithms such as rivest–shamir–adleman (RSA) algorithm and Diffie–Hellman algorithm have been proposed. PKC is asymmetric involving the use of two separate keys, in contrast to symmetric conventional encryption, which uses only a single key. The use of two keys provides solution to key management and user authentication in a cryptosystem. RSA algorithm is the best known, the most versatile and widely used public key algorithm today RSA depends on the modular exponentiation of long integers, which is the critical operation for a variety

of the most widely accepted cryptosystems. Figure 1.1 shows public key encryption. A user of RSA creates and then publishes a public key based on the two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime numbers can feasibly decode the message.

Breaking RSA encryption is known as the RSA problem; whether it is as hard as the factoring problem remains an open question. Therefore, fast modular multiplication becomes the key to real-time encryption and decryption since a high throughput is needed in data communication. The most widely used algorithm for efficient modular multiplication is Montgomery's algorithm. The binary Montgomery's modular-multiplication algorithm employs only simple addition, subtraction, and shift operation to avoid trial division, a critical and time-consuming operation in conventional modular multiplication. The modular exponentiation is usually accomplished by performing repeated modular multiplications. There are approximately 10^{151} primes 512 bits in length or less. With approximately 10^{77} atoms in the universe, and if every atom in the universe needed a billion new primes every microsecond from the beginning of time until now, you would need only 10^{109} primes. So the odds of two people picking the same prime at random is, for all practical purposes, zero.

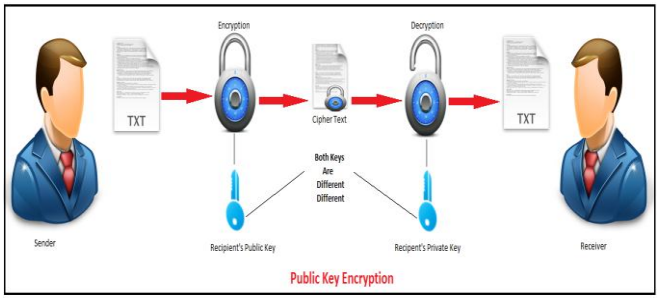

Figure 1.1: Public Key Encryption

The Sanskrit word 'Veda' means 'knowledge'. The Vedas consist of a huge number of documents there are said to be thousands of such documents in India, many of which have not yet been translated, which are shown to be highly structured, both within themselves and in relation to each other. Some documents, called 'Ganita sutras' (the name 'ganita' means mathematics), were devoted to mathematical knowledge. Sri Bharati Krishna Tirtha Maharaj, who is generally considered the doyen of this discipline, in his seminal book Vedic Mathematics, wrote about this special use of sutras. Vedic Mathematics" was the name given by him. He was the person who collected lost formulae from the writings of "Atharwa Vedas" and wrote them in the form of Sixteen Sutras and thirteen sub-sutras.

According to him, there has been considerable literature on Mathematics in the Veda-sakhas. Unfortunately most of it has been lost to humanity as of now. This is evident from the fact that while, by the time of Patanjali, about 25 centuries ago, 1131 Veda-sakhas were known to the Vedic scholars, only about ten Veda-sakhas are presently in the knowledge of the Vedic scholars in the country. The Sutras apply to and cover almost every branch of Mathematics. They apply even to complex problems involving a large number of mathematical operations. Application of the Sutras saves a lot of time and effort in solving the problems, compared to the formal methods presently in vogue. Though the solutions appear like magic, the application of the Sutras is perfectly logical and rational. The computation made on the computers follows, in a way, the principles underlying the Sutras. The Sutras provide not only methods of calculation, but also ways of thinking for their application.

Vedic Mathematics is based on 16 sutras dealing with mathematics related to arithmetic, algebra, and geometry. These methods and ideas can be directly applied to trigonometry, plain and spherical geometry, conics, calculus and applied mathematics of various kinds. Application of the Sutras improves the computational skills of the learners in a wide area of problems, ensuring both speed and accuracy, strictly based on rational and logical reasoning. The knowledge of such methods enables the teachers to be more resourceful to mould the students and improve their talent and creativity. Application of the Sutras to specific problems involves rational thinking, which, in the process, helps improve intuition that is the bottom - line of the mastery of the mathematical geniuses of the past and the present such as Aryabhatta, Bhaskaracharya, Srinivasa Ramanujan, etc. The Vedic methods are direct, and truly extraordinary in their efficiency and simplicity. Research is being carried out in many areas, including the effects on children who learn Vedic maths and the development of new, powerful but easy applications of the Vedic sutras in geometry, calculus, computing etc. But the real beauty and effectiveness of Vedic mathematics cannot be fully appreciated without actually practicing the system. One an then see that it is perhaps the most refined and efficient mathematical system possible.

2. Literature Review

With the recent advent of hardware description languages (e.g. VHDL) and digital implementation for field-programmable gate arrays (FPGAs), substantial academic digital design projects become practicable. The time and effort to implement significant design projects may be undertaken without sacrificing the broad educational demands placed upon the modern engineering student. The authors have discussed the digital design, simulation, and FPGA implementation, using Xilinx design tools and simulation software. They have shown the digital architecture used to implement the system.

Sriraman, L, Kumar K.S, Prabakar, T.N [1] presented Vedic mathematics is one of the ancient Indian mathematics which contains sixteen sutras. These sutras can be used to solve problems in any branch of Mathematics in a faster way. The proposed squarer is based on sutra called Ekadhikena Purvena. It means that ―one more than the previous‖. This sutra is used for finding the square of decimal numbers ending with `5'. In this paper this sutra is generalized and used for squaring of binary numbers [2].

Gustavo D. Sutter, Jean-Pierre Deschamps, and José Luis Imana [3] presented Modular exponentiation with large modulus and exponent, which is usually accomplished by repeated modular multiplications, has been widely used in public key cryptosystems. Typically, the Montgomery's modular-multiplication algorithm is used since no trial division is necessary, and the carry–save addition (CSA) is employed to reduce the critical path. In this paper, we optimize the Montgomery's multiplication and propose architectures to perform the least significant bit first and the most significant bit first algorithm. The RSA operation is a modular exponentiation, and its security lies in its inability to efficiently factor large integers. Moreover, since the size of the modulus is at least 1024 b for long-term security, it means that high throughput rate is hard to achieve without the use of hardware acceleration. Efficient Montgomery's multiplication and associated exponentiation algorithms and circuit architectures have been presented.CSA was used to perform large word-length additions in conjunction with quotient precomputation and digit serial computation [4],[5].

F. Mace, F.-X. Standaert, and J.-J. Quisquater [6] presented, SEA is a scalable encryption algorithm targeted for small embedded applications. It was initially designed for software implementations in controllers, smart cards, or processors. Beyond its low cost performances, a significant advantage of the proposed architecture is its full flexibility for any parameter of the scalable encryption algorithm, taking advantage of generic VHDL coding. The letter also carefully describes the implementation details allowing us to keep small area requirements.

Xiaoming Tang [7] presented Most FPGA support floating-point IP core nowadays, but we have paid little attention to the acquisition of data source. A certain range of real number presented by ASCII code is converted to single precision floating-point by pipeline processing with VHDL language. Through functional simulation and download verification, the conversion time is about 10 us when the clock is 50 MHz .The conversion accuracy of this method can reach 10 calculated by MATLAB software.

Huddar, S.R. ,Rupanagudi, S.R. , Kalpana, M. , Mohan, S. presented [8] With the advent of new technology in the fields of VLSI and communication, there is also an ever growing demand for high speed processing and low area design. It is also a well known fact that the multiplier unit forms an integral part of processor design. Due to this regard, high speed multiplier architectures become the need of the day. The speed of a processor greatly depends on its multiplier's performance. This in turn increases the demand for high speed multipliers, at the same time keeping in mind low area and moderate power consumption. A novel high speed architecture for multiplication of two 8 bit numbers, combining the advantages of compressor based adders and also the ancient Vedic maths methodology. A new 7:2compressor architecture, based on 4:2 compressor architecture was also discussed [9].

Ali Ziya Alkara, Remziye Sonmez [10] presented Rivest–Shamir–Adleman (RSA) is one of the most widely preferred algorithms used in public-key cryptography systems. RSA has a very slow ciphering rate if used in software. The use of a specific hardware is the only reasonable solution in applications where performance is the key factor. To speed up the modular multiplication and squaring, bit level systolic arrays are used with the Montgomery's modular multiplication algorithm to constitute the core of modular exponentiation operation. RSA algorithm is the best known, the most versatile and widely used public key algorithm today. Therefore, fast modular multiplication becomes the key to real-time encryption and decryption since a high throughput is needed in data communication. A 1024-bit RSA algorithm using two single-row SA structures working in parallel where each cell is formed of cells defined by Shin. The results are obtained at the high-level synthesis stage and show the highest possible clock rate that can be achieved. The throughput is (n+2)(n+3)that corresponds to $1.0^5\_10^6$ cycles.

G.P. Saggese, L.Romano [11] presented an accelerator which can effectively improve the security and the performance of virtually any RSA cryptographic application. RSA is the most widely adopted standard for cryptographic systems, an accelerator can effectively be used to improve the dependability (namely security and performance) of a wide class of security services. This work makes three important contributions. The first contribution is the description of the architecture itself. The second contribution is the implementation of the architecture using Commercial Off The Shelf (COTS) FPGA technology, namely a CeloxicaRC1000programmable board mounting a Xilinx Virtex-E 2000 FPGA part. The third contribution is the experimental analysis of the tradeoffs—in terms of performance vs. area occupation—which result from different levels of parallelism, i.e. from different values of the digit size of the RSA crypto-processor [12].

R.Bhaskar, Ganapathi Hegde, P.R.Vaya, [13] presented the standard techniques for providing privacy and security in data networks include encryption/decryption algorithms such as Advanced Encryption System (AES) (private-key) and RSA (public-key). One of the most time consuming processes in RSA encryption/ decryption algorithm is the computation of a b mod n where "a" is the text, (b, n) is the key. Generally the prime number used for RSA Encryption system will around 100 to 150 decimal digits. The computations involved are tedious and time consuming. Also the hardware is quite complex. To increase the computation speed, the multiplication principle of Vedic mathematics is used and also an improvement is made in the conventional restoring algorithm which does the modulus operation. "Urdhva-tiryakbhyam" is the sutra (principle) which used to compute the multiplication. It literally means vertical and crosswise manipulation [14].

Jaina, D, Sethi, K. , Panda, R.[15] presented Real-time signal processing requires high speed and high throughput Multiplier-Accumulator (MAC) unit that consumes low power, which is always a key to achieve a high performance digital signal processing system. In this paper, design of MAC unit is proposed. The multiplier used inside the MAC unit is based on the Sutra "Urdhva Tiryagbhyam" (Vertically and Cross wise) which is one of the Sutras of Vedic

mathematics. Vedic mathematics is mainly based on sixteen Sutras and was rediscovered in early twentieth century. In ancient India, this Sutra was traditionally used for decimal number multiplications within less time. The same concept is applied for multiplication of binary numbers to make it useful in the digital hardware.

K.Z. Pekmestzi, N.K. Moshopoulos [16] presented a new systolic serial–parallel scheme that implements the Montgomery multiplier is presented. The serial input of this multiplier consists of two sets of data that enter in a bit-interleaved form. The results are also derived in the same form. The design, with minor modifications, can be used for the implementation of the RSA algorithm by realizing the square-and-multiply algorithm. The circuit yields the lowest hardware complexity reported and permits high-speed operation with 100% efficiency. The core of an RSA cryptosystem is the modular exponentiation, which can be constructed by a sequence of modular multiplications and squaring. These operations have to be performed in a serial pipelined way, because of the operands length (>1024 bits). The most efficient algorithm for modular multiplication was presented by Montgomery. One approach proposes a direct implementation of the Montgomery scheme by using two similar circuits: one for multiplication and one for squaring [17],[18]. Asymmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the different keys—one a public key and one a private key. It is also known as public-key encryption. Asymmetric encryption transforms plaintext into cipher text using a one of two keys and an encryption algorithm. Using the paired key and a decryption algorithm, the plaintext is recovered from the cipher text. Asymmetric encryption can be used for confidentiality, authentication, or both. The most widely used public-key cryptosystem is RSA. The difficulty of attacking RSA is based on the difficulty of finding the prime factors of a composite number. [19]

3. System Model Description

3.1 Introduction

Multipliers are extensively used in Microprocessors, DSP and Communication applications. For higher order multiplications, a huge number of adders are to be used to perform the partial product addition. The need of high speed multiplier is increasing as the need of high speed processors are increasing. Higher throughput arithmetic operations are important to achieve the desired performance in many real time signal and image processing applications. One of the key arithmetic operations in such applications is multiplication and the development of fast multiplier circuit has been a subject of interest over decades. Reducing the time delay and power consumption are very essential requirements for many applications. In the past multiplication was implemented generally with a sequence of addition, subtraction and shift operations. Two most common multiplication algorithms followed in the digital hardware are array multiplication algorithm and Booth multiplication algorithm. Due to the importance of digital multipliers in DSP, it has always been an active area of research.

3.2 Vedic Multiplication Technique

The use of Vedic mathematics lies in the fact that it reduces the typical calculations in conventional mathematics to very simple one. This is so because the Vedic formulae are claimed to be based on the natural principles on which the human mind works. Vedic Mathematics is a methodology of arithmetic rules that allow more efficient speed implementation. It also provides some effective algorithms which can be applied to various branches of engineering such as computing. A. Urdhva Tiryakbhyam Sutra The proposed Vedic multiplier is based on the "Urdhva Tiryagbhyam" sutra (algorithm). These Sutras have been traditionally used for the multiplication of two numbers in the decimal number system. In this work, we apply the same ideas to the binary number system to make the proposed algorithm compatible with the digital hardware. It is a general multiplication formula applicable to all cases of multiplication. It literally means "Vertically and Crosswise". It is based on a novel concept through which the generation of all partial products can be done with the concurrent addition of these partial products. The algorithm can be generalized for n x n bit number. Since the partial products and their sums are calculated in parallel, the multiplier is independent of the clock frequency of the processor. Due to its regular structure, it can be easily layout in microprocessors and designers can easily circumvent these problems to avoid catastrophic device failures. The processing power of multiplier can easily be increased by increasing the input and output data bus widths since it has a quite a regular structure. Due to its regular structure, it can be easily layout in a silicon chip. The Multiplier based on this sutra has the advantage that as the number of bits increases, gate delay and area increases very slowly as compared to other conventional multipliers.

3.2.1 Multiplication of two numbers 252 x 846:

To illustrate this scheme, let us consider the multiplication of two decimal numbers 252 x 846 by Urdhva-Tiryakbhyam method as shown in Fig. 3.1. The digits on the both sides of the line are multiplied and added with the carry from the previous step. This generates one of the bits of the result and a carry. This carry is added in the next step and hence the process goes on. If more than one line are there in one step, all the results are added to the previous carry. In each step, least significant bit acts as the result bit and all other bits act as carry for the next step. Initially the carry is taken to be zero.

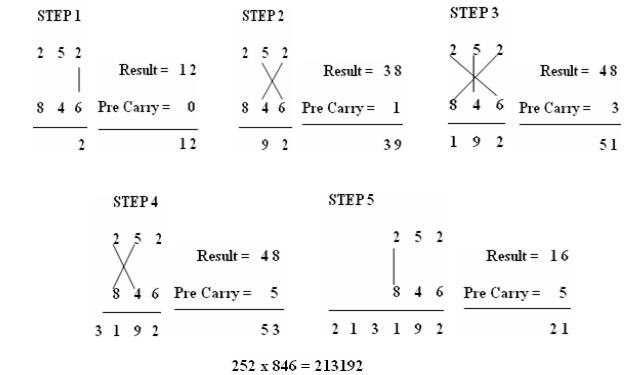

Figure 3.1: Multiplication of two decimal numbers –252 x 846

The hardware architecture of 2X2, 4x4 and 8x8 bit Vedic multiplier module are displayed in the below sections. Here, "Urdhva-Tiryagbhyam" (Vertically and Crosswise) sutra is used to propose such architecture for the multiplication of two binary numbers. The beauty of Vedic multiplier is that here partial product generation and additions are done concurrently. Hence, it is well adapted to parallel processing. The feature makes it more attractive for binary multiplications. This in turn reduces delay, which is the primary motivation behind this work.

3.2.2 Vedic Multiplier for 2x2 bit Module:

The method is explained below for two, 2 bit numbers A and B where A = a1a0 and B = b1b0 as shown in Fig.3.2. Firstly, the least significant bits are multiplied which gives the least significant bit of the final product (vertical). Then, the LSB of the multiplicand is multiplied with the next higher bit of the multiplier and added with, the product of LSB of multiplier and next higher bit of the multiplicand (crosswise). The sum gives second bit of the final product and the carry is added with the partial product obtained by multiplying the most significant bits to give the sum and carry. The sum is the third corresponding bit and carry becomes the fourth bit of the final product.

s0 = a0b0; (1)
c1s1 = a1b0 + a0b1; (2)
c2s2 = c1 + a1b1; (3)

The final result will be c2s2s1s0. This multiplication method is applicable for all the cases.

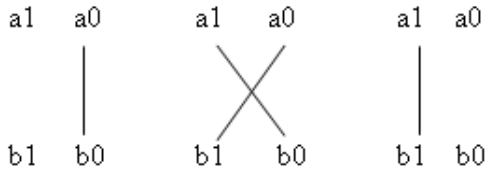

Figure 3.2: The Vedic Multiplication Method for two 2-bit Binary Numbers

Fig.3.2 The Vedic Multiplication Method for two 2-bit Binary Numbers The 2X2 Vedic multiplier module is implemented using four input AND gates & two half-adders which is displayed in its block diagram in Fig.3.3. It is found that the hardware architecture of 2x2 bit Vedic multiplier is same as the hardware architecture of 2x2 bit conventional Array Multiplier. Hence it is concluded that multiplication of 2 bit binary numbers by Vedic method does not made significant effect in improvement of the multiplier's efficiency. Very precisely we can state that the total delay is only 2-half adder delays, after final bit products are generated, which is very similar to Array multiplier. So we switch over to the implementation of 4x4 bit Vedic multiplier which uses the 2x2 bit multiplier as a basic building block. The same method can be extended for input bits 4 & 8. But for higher no. of bits in input, little modification is required.

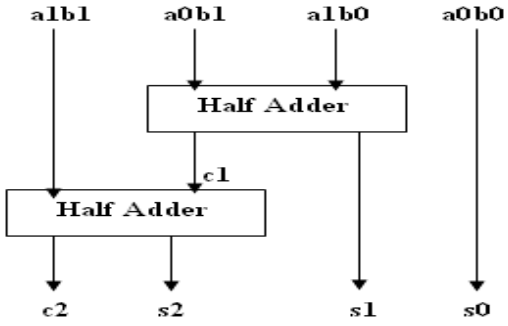

Figure 3.3: Block Diagram of 2x2 bit Vedic Multiplier

3.2.3 Vedic Multiplier for 4x4 bit Module

The 4x4 bit Vedic multiplier module is implemented using four 2x2 bit Vedic multiplier modules as discussed in Fig.3.4. Let's analyze 4x4 multiplications, say A= A3 A2 A1 A0 and B= B3 B2 B1 B0. The output line for the multiplication result is – S7 S6S5S4 S3 S2 S1 S0 .Let's divide A and B into two parts, say A3 A2 & A1 A0 for A and B3 B2 & B1B0 for B. Using the fundamental of Vedic multiplication, taking two bit at a time and using 2 bit multiplier block, we can have the following structure for multiplication as shown in Fig. 3.4.

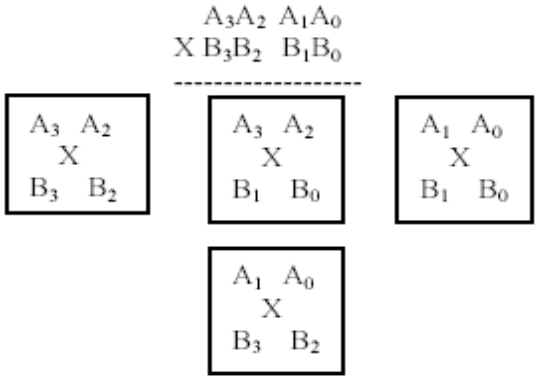

Figure 3.4: Sample Presentation for 4x4 bit Vedic Multiplication

Each block as shown above is 2x2 bit Vedic multiplier. First 2x2 bit multiplier inputs are A1A0 and B1B0. The last block is 2x2 bit multiplier with inputs A3 A2 and B3 B2. The middle one shows two 2x2 bit multiplier with inputs A3 A2 & B1B0 and A1A0 & B3 B2. So the final result of multiplication, which is of 8 bit, S7 S6S5S4 S3 S2 S1 S0. To get final product (S7 S6 S5 S4 S3 S2 S1 S0), four 2x2 bit Vedic multiplier (Fig. 3.5) and three 4-bit Ripple-Carry (RC) Adders are required. The proposed Vedic multiplier can be used to reduce delay. Early literature speaks about Vedic multipliers based on array multiplier structures. On the other hand, we proposed a new architecture, which is efficient in terms of speed. The arrangements of RC Adders shown in Fig.3.5, helps us to reduce delay. Interestingly, 8x8 Vedic multiplier modules are implemented easily by using four 4x4 multiplier modules.

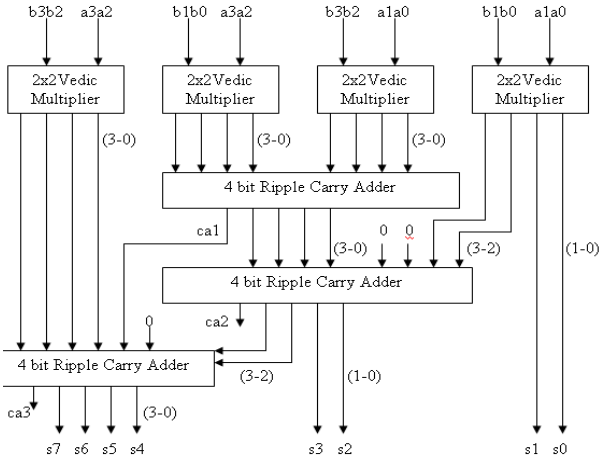

Figure 3.5: Block Diagram of 4x4 bit Vedic Multiplier

3.2.4 Vedic Multiplier for 8x8 bit Module:

The 8x8 bit Vedic multiplier module as shown in the block diagram in Fig. 3.6 can be easily implemented by using four 4x4 bit Vedic multiplier modules as discussed in the previous section. Let's analyze 8x8 multiplications, say A= A7 A6 A5 A4 A3 A2 A1 A0 and B= B7 B6 B5B4 B3 B2 B1B0. The output line for the multiplication result will be of 16 bits as – S15 S14 S13 S12 S11 S10 S9 S8 S7 S6S5S4 S3 S2 S1 S0. Let's divide A and B into two parts, say the 8 bit multiplicand A can be decomposed into pair of 4 bits AH-AL. Similarly multiplicand B can be decomposed into BH-BL. The 16 bit product can be written as:

P = A x B = (AH-AL) x (BH-BL)

= AH x BH + (AH x BL + AL x BH) + AL x BL

Using the fundamental of Vedic multiplication, taking four bits at a time and using 4 bit multiplier block as discussed we can perform the multiplication. The outputs of 4x4 bit multipliers are added accordingly to obtain the final product. Here total three 8 bit Ripple-Carry Adders are required as shown in Fig. 3.6.

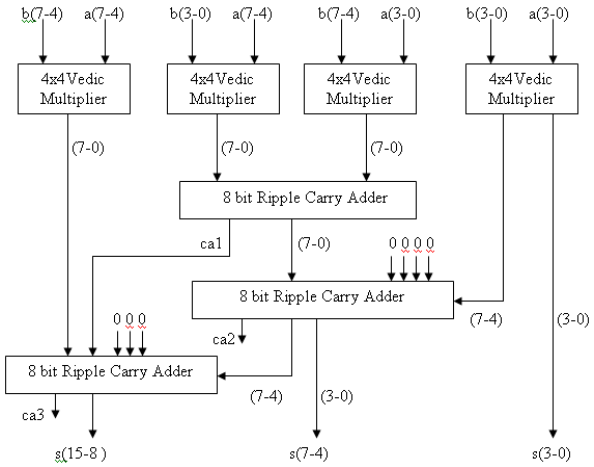

Figure 3.6: Block Diagram of 8x8 bit Vedic Multiplier

3.2.5 Generalized Algorithm for N x N bit Vedic Multiplier

We can generalize the method as discussed in the previous sections for any number of bits in input. Let, the multiplication of two N-bit binary numbers (where N = 1, 2, 3…N, must be in the form of 2N) *A* and *B* where A = AN.....A3 A2 A1 and B = BN….B3 B2 B1. The final multiplication result will be of (N + N) bits as S = S(N + N)....S3 S2 S1.

Step 1: Divide the multiplicand A and multiplier B into two equal parts, each consisting of [N to (N/2)+1] bits and [N/2 to 1] bits respectively, where first part indicates the MSB and other represents LSB.

Step 2: Represent the parts of A as AM and AL, and parts of B as BM and BL. Now represent A and B as AM AL and BM BL respectively.

Step 3: For A X B, we have general format as shown in Fig.3.7

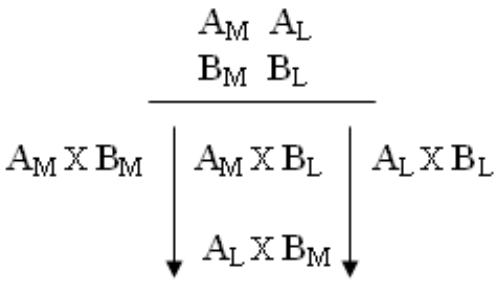

Figure 3.7 : General Representation for Vedic Multiplication

Step 4: The individual multiplications product can be obtained by the partitioning method and applying the basic building blocks.

By adopting the above generalized algorithm we can implement Vedic Multiplier for any number of bits say 16, 32, 64, and so on, as per the requirement. Therefore, it could be possible to implement this Vedic multiplier in the ALU (Arithmetic Logic Unit) which will reduce the computational speed drastically & hence improves the processors efficiency.

3.3 RSA Cryptosystem

The RSA Algorithm is based on the mathematical fact that it is easy to find and multiply the large prime numbers together, but it is extremely difficult to factor their product. The public and private keys in RSA are based on very large prime numbers. The algorithm is simple but the complexity lies in the selection and generation of public and private keys. The RSA algorithm involves three steps: key generation, encryption and decryption.

3.3.1. Key Generation

RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.

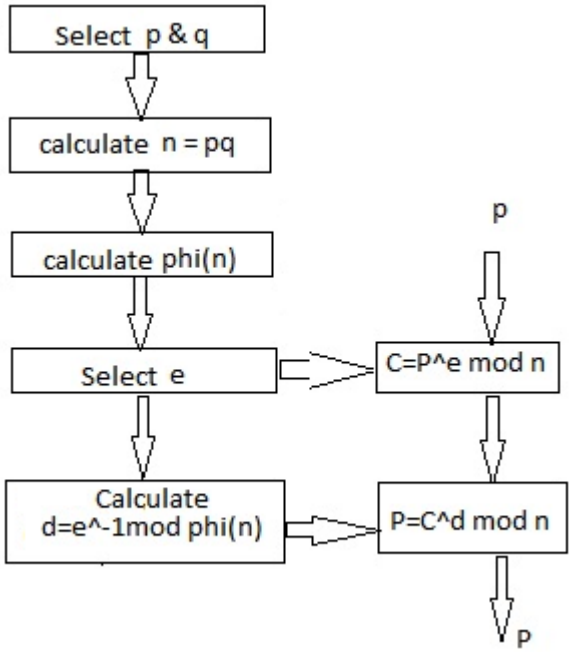

Figure 3.8: Flow chart of RSA algorithm

The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers p and q.
   - For security purposes, the integer's p and q should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a primarily test.
2. Compute n = p q.
   - n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
3. Compute φ(n) = φ(p)φ(q) = (p − 1)(q − 1) = n - (p + q -1), where φ is Euler's totient function.
4. Choose an integer e such that 1 < e <φ(n) and gcd (e, φ(n)) = 1; i.e., e and φ(n) are co prime.
   - e is released as the public key exponent.
   - e having a short bit-length and small Hamming weight results in more efficient encryption – most commonly 216 + 1 = 65,537. However, much smaller values of e (such as 3) have been shown to be less secure in some settings.
5. Determine d as d ≡ e−1 (mod φ(n)); i.e., d is the multiplicative inverse of e (modulo φ(n)
   - This is more clearly stated as: solve for d given d·e ≡ 1 (mod φ(n))
   - This is often computed using the extended Euclidean algorithm. Using the pseudo code in the Modular integers section, inputs a and n correspond to e and φ(n), respectively.
   - d is kept as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and φ(n) must also be kept secret because they can be used to calculate d.

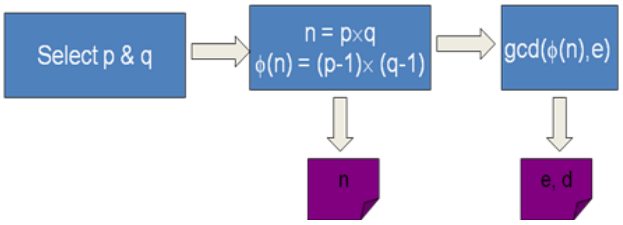

Figure 3.9: System architecture for Key generation

3.3.2. Encryption

A transmits its public key (n, e) to B and keeps the private key d secret. B then wishes to send message P to A , then computes the cipher text C corresponding to.

$$C = P^e \pmod n$$

This can be done efficiently, even for 500-bit numbers, using Modular exponentiation. B then transmits C to A.

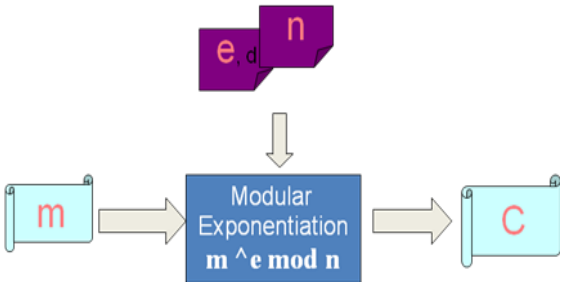

Figure 3.10: RSA Encryption Structure

3.3.3 Decryption

A can recover P from C by using its private key exponent d via computing

$P = C^d \pmod n$

Thus we get the original message.

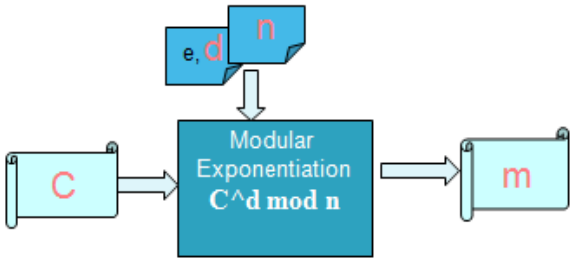

Figure 3.11: The RSA Decryption Structure

3.4 A Worked Example of RSA Encryption and Decryption

Here is an example of RSA encryption and decryption. The parameters used here are artificially small, but one can also use Open SSL to generate and examine a real key pair.

1. Choose two distinct prime numbers, such as p= 61 and q=53
2. Compute $n = pq$ giving

n=61×53=3233

3. Compute the totient of the product as $\varphi(n) = (p - 1)(q - 1)$ giving

Φ (3233) = (61-1) (53-1) = 3120

4. Choose any number $1 < e < 3120$ that is co prime to 3120. Choosing a prime number for e leaves us only to check that e is not a divisor of 3120.
   Let e = 17
5. Compute d, the modular multiplicative inverse of e (mod φ(n)) yielding,
   d = 2753

Worked example for the modular multiplicative inverse:
e × d mod Φ(n) =1
17×2753 mod 3120

The **public key** is (n = 3233, e = 17) .For a padded plaintext message m, the encryption function is
$c(m) = m^{17} \bmod 3233$

The **private key** is (d = 2753). For an encrypted cipher text c, the decryption function is
$m(c) = c^{2753} \bmod 3233$

For instance, in order to encrypt m = 65, we calculate

$$c = 65^{17} \bmod 3233 = 2790$$

To decrypt c = 2790, we calculate

$$m = 2790^{2753} \bmod 3233$$

Both of these calculations can be computed efficiently using the square- and -multiply algorithm for modular exponentiation. In real-life situations the primes selected would be much larger; in our example it would be trivial to factor n, 3233 (obtained from the freely available public key) back to the primes p and q. Given e, also from the public key, we could then compute d and so acquire the private key.

4. Result and Discussion

The objectives of this project are to design and implement the RSA CRYPTOSYSTEM to improve speed performance, area reduction and throughput. The implementation of RSA cryptosystem includes Key generation, Encryption and Decryption in Modelsim simulation Environment which was an interesting task to design that module. To begin the testing of encryption /decryption process, a set of RSA parameters has calculated. Then the calculated parameters have been fed into the RSA_CORE module and the results have been compared with the theoretical values. Assumed that p = 61, q = 53. To generate the two keys, the product is computed as, n = p x q = 3233=0CA1**.** The Euler's Totient function of n is computed as, φ = 3120. Public key e is randomly chosen such that e and φ is relatively prime. Here, e = 11 has been chosen. Finally, the Extended Euclid's algorithm is used to compute the decryption key, d. The private key for decryption obtained as d = 0AC1. For each of the key size following input set is used:

Encryption: Message, $M = 7_{16}$
Public key, $E = 11_{16}$
Modulus, $N = 0CA1_{16}$
Decryption: Message, $M = 0941_{16}$
Public key, $E = 0AC1_{16}$
Modulus, $N = 0CA1_{16}$
Thus, public key = (11, 3233), private key = (0AC1).
Fig. 6.1 shows the simulation result of key generation.

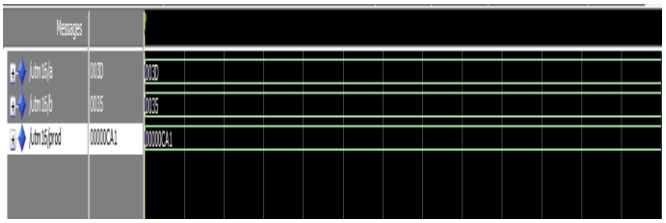

Figure 4.1: Simulation result of Key generation

Table 1: Design Summary of Key generation

Logic Utilization	Used	Available	Utilization
Number of Slices	498	4656	10%
Number of 4 input LUTs	902	9312	9%
Number of IOs	64		
Number of bonded IOBs	64	232	27%
Delay	239.976ns (Levels of Logic = 168)		

Fig 4.2 shows the simulation result of Encryption, which gives the plain text. The Table II shows the Device Utilization Summary for Encryption. The Timing Summary is shown in Table III.

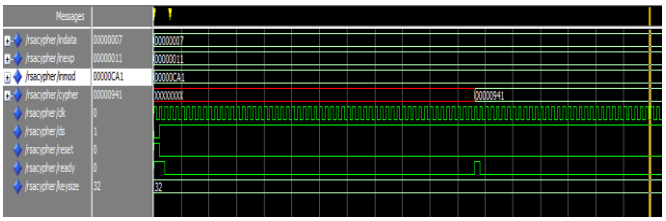

Figure 4.2: Simulation result of Encryption

Table 2: Design Summary of Encryption

Logic Utilization	Used	Available	Utilization
Number of Slices	517	4656	11%
Number of Slice Flip Flops	459	9312	4%
Number of 4 input LUTs	936	9312	10%
Number of IOs	132		
Number of bonded IOBs	132	232	56%
Number of GCLKs	1	24	4%

Table 3: Timing Summary of Encryption

Speed Grade	-4
Minimum period	12.266ns (Maximum Frequency: 81.523MHz)
Minimum input arrival time before clock	5.241ns
Maximum output required time after clock	5.169ns

Fig 4.3 shows simulation result of decryption. The Device utilization summary of decryption is summarized in Table IV. The timing summary is shown in Table V.

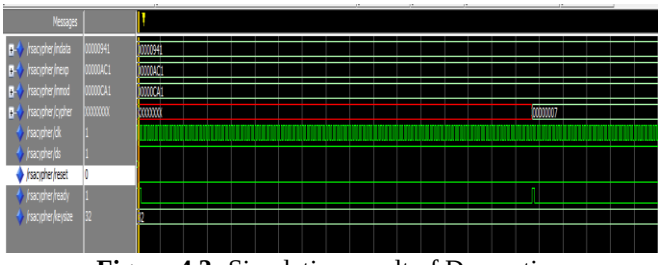

Figure 4.3: Simulation result of Decryption

Table 4: Design Summary of Decryption

Logic Utilization	Used	Available	Utilization
Number of Slices	517	4656	11%
Number of Slice Flip Flops	459	9312	4%
Number of 4 input LUTs	936	9312	10%
Number of IOs	132		
Number of bonded IOBs	132	232	56%
Number of GCLKs	1	24	4%

Table 5: Timing Summary of Decryption.

Speed Grade	-4
Minimum period	12.266ns (Maximum Frequency: 81.523MHz)
Minimum input arrival time before clock	5.241ns
Maximum output required time after clock	5.169ns

5. Advantages

1) The biggest advantage of RSA is that it uses asymmetric keys.
2) Very high speed of encryption.
3) Private Keys are completely unknown to everybody, even the Trust Authority's manager !

All keys are written into chips and are not accessible to humans or other machines. This Guarantees the privacy of all the end-users.
4) New applications can be added without updating the chip.
5) RSA has it's advantages of being a reliable and safe system

6. Applications

RSA algorithm is the most used algorithm in commercial systems.
1) It is used to protect web traffic, in the SSL protocol (Security Socket Layer).
2) It is used to guarantee email privacy and authenticity in PGP (Pretty Good Privacy),Remote connection in SSH (SecureShell).
3) Furthermore it plays an important role in the modern payment systems through SET Protocol (Secure Electronic Transaction).
4) RSA has been used in most digital data, information and telephone security applications.
5) It is used worldwide to secure Internet, banking and credit card transactions.

7. Conclusion

The RSA encryption/decryption system is implemented using the Vedic Mathematics algorithm to increase its computation speed. The advantage of the Vedic multiplier is that it calculates the partial products in one single step and there are no shift operations which saves the time and the hardware. As the number of message bits increases the gate delay as well as the area increase slowly. Hence it can be used effectively in all the cryptographic applications. It is found that this design is 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.

8. Future Scope

There is a huge implication for the Internet technology industry. The algorithm which is created runs with far greater efficiency than one of the industry's best algorithms. The implementation of this algorithm into websites for e-commerce or email could mean huge speedups in performance. The increase in speed could also allow cryptographers to use larger key sizes for greater security, without having to compromise speed. Ideas for improving this algorithm include implementing a finite field in base 3 or even finding variations of this algorithm that improve certain security issues such as resistance to side-channel attacks. Another possibility for future work is implementing this algorithm with special large integer (> 232 bits) manipulation parallelization or programming NIST recommended elliptic curves. Efficient security mechanisms for resource constrained devices. Quite efficient in terms of silicon area and speed and should result in substantial saving of resources when used for crypto and security applications.

References

[1] Sriraman, L. Dept. of Electron. &Commun. Eng., Oxford Eng. Coll., Trichy, India;Kumar, K.S. ;Prabakar, T.N.*"Design and FPGA implementation of binary squarer using Vedic mathematics*"IEEE Trans. Ind. Electron., July 2013.
[2] Prabir Saha, Arindam Banerjee, Partha Bhattacharyya, Anup Dandapat "*High Speed ASIC Design of Complex Multiplier Using Vedic Mathematics*" Proceeding of the 2011 IEEE Students' Technology Symposium 14-16 January, 2011, lIT Kharagpur.
[3] Gustavo D. Sutter, Member, IEEE, Jean- Pierre Deschamps, and José Luis Imaña , *"Modular Multiplication and Exponentiation Architectures for Fast RSA Cryptosystem Based on Digit Serial Computation"* IEEE Trans. Ind.Electron.,vol.57,no.10, p.p3308–3316, Oct. 2010.
[4] E. Monmasson and M. N. Cirstea, "*FPGA design methodology for industrial control systems—A review*," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1824–1842, Aug .2007.
[5] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, "*Features, design tools, an Application domains of FPGAs*", IEEE Trans. Ind. Electron., vol. 54, no. 4, pp. 1810,1823, Aug. 2007.
[6] F. Macé, F. -X. Standaert, and J.-J. Quisquater *, "FPGA Implementation of a Scalable Encryption Algorithm*" , IEEE Transactions on Very large Scale Integration (VLSI) systems, vol. 16, no. 2, February 2008.
[7] Xiaoming Tang ; Res. Inst. of Inf. Fusion, Naval Aeronaut. & Astronaut. Univ., Yantai, China; Tao Zhang ; Zhenjie Wang ; Wenliang Yuan, "*A novel data format conversion method based on FPGA*", IEEE Trans. Ind. Electron ,July 2011.
[8] Huddar, S.R. ; WorldServe Educ., Bangalore, India ; Rupanagudi, S.R. ; Kalpana, M. Mohan, S, *"Novel high speed vedic mathematics multiplier using compressors"*, IEEE Trans. Ind. Electron ,March 2013.

[9] M. Ramalatha, K. Deena Dayalan, S. Deborah Priya, "*High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques*," Advances in Computational Tools for Engineering Applications, 2009, IEEE Proc., pp 600-603.
[10] A.Z. Alkar, R. Sonmez, An ASIC Implementation of the RSA Algorithm 18th MUG International Conference, February 2002.
[11] G.P. Saggese a, L. Romano a, N. Mazzocca b, A. Mazzeo, "*A tamper resistant 50 hardware accelerator for RSA cryptographic applications"*, Journal of Systems Architecture (2004) 711–727.
[12] IEEE Std 1363–2000: Standard specifications for Public- Key cryptography. IEEE, August 2000.
[13] R. Bhaskar, Ganapathi Hegde, P. R. Vaya, "*An efficient hardware model for RSA Encryption system usingVedic mathematics"*,International Conference on Technology Communication and System Design 2011.
[14] Nitish Aggarwal , Kartik Asooja , Saurabh Shekhar Verma, Sapna Negi, "*An Improvement in the Restoring Division Algorithm*", IEEE 2009.
[15] Jaina, D. Dept. of Electron. & Telecommun. Eng., VSS Univ. of Technol., Burla, India; Sethi, K. ; Panda, R, "*Vedic Mathematics Based Multiply Accumulate Unit*" IEEE Trans. Ind. Electron ,Oct. 2011.
[16] K.Z. Pekmestzi*, N.K. Moshopoulos, "*A bit-interleaved systolic architecture for a High speed RSA system*", INTEGRATION, the VLSI journal 30 (2001) 169–175.
[17] R.L.Rivest, A. Shamir, and L.Adleman, "*A method for obtaining digital signature and public-key cryptosystems"*,Commun. ACM, vol. 21, no. 2, pp. 120–126, Feb 1978.
[18] C. Yang, T. Chang, C. Jen, "*A new RSA cryptosystem hardware design based on Montgomery algorithm"*, IEEE Trans.Circuits Systems II 45 (7) (1998) 908–913.
[19] William Stallings,―Cryptography and Network Security, Third Edition, Pearson Education, 2003.