“FPGA BASED VLSI DESIGN AND ITS IMPLEMENTATION OF RSA CRYPTOSYSTEM ALGORITHM USING VEDIC MATHEMATICS”

[1]DHANASHRI R. KADU
ME Scholar, Department of Electronics & Tele-Communication, Sipna College of Engineering. & Technology, Amravati, India
Kadudhanu6@gmail.com

[2]DR. G. P. DHOK
Head, Department of Instrumentation, Sipna College of Engineering. & Technology, Amravati, India
gajanandhok@rediffmail.com

ABSTRACT: *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). RSA is one of the safest standard algorithms, based on public-key, for providing security in networks. Even though the RSA Algorithm is an old and simple encryption technique, there is a scope to improve its performance. This paper proposes the implementation of RSA encryption algorithm using the sutras or principles of ancient Indian Vedic mathematics that have been modified the algorithm to improve performance. It is observed that, RSA algorithm is difficult to implement with high speed because one of the most time consuming processes in RSA encryption/ decryption algorithm is the computation of ab 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 in this paper. “Urdhva-tiryakbhyam” is the sutra (principle) which used to compute the multiplication. It literally means vertical and crosswise manipulation. The significance of this technique is that it computes the partial products in one step and avoids the shifting operation which saves both time and hardware. The VHDL code is simulated and synthesized on ModelSim SE 6.3f and Altera Quartus II 9.1 tools respectively.*

***Keywords*:** Vedic mathematics, RSA Algorithm, Decryption, Encryption, Key, Cipher Text.

1. INTRODUCTION

Cryptography is the study of methods for sending messages in secret (namely, in enciphered or disguised form) so that only the intended recipient can remove the disguise and read the message (or decipher it, thereby providing confidentiality). It is the art of using mathematics to address the issue of information security. Cryptography has, as its etymology krypton from the Greek, meaning hidden, and graphy, meaning to write. The original message is called the ‘Plaintext’ and the disguised message is called the ‘Cipher text’. The final message, encapsulated and sent, is called a ‘Cryptogram’. The process of transforming plaintext into cipher text is called ‘Encryption’ or ‘Enciphering’. The reverse process of turning cipher text into plaintext, which is accomplished by the recipient who has the knowledge to remove the disguise, is called ‘Decryption’ or ‘Deciphering’.
Two types of cryptography are private/secret/single key cryptography & Public key cryptography.RSA is public key algorithm.

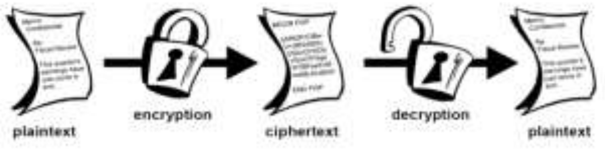


Figure 1: Encryption and decryption

The RSA Algorithm is the most popular and proven asymmetric key cryptographic algorithm. In 1977, Ron Rivest, Adi Shamir and Len Adleman at MIT developed the first major asymmetric key cryptography system. Hence it is called as RSA Algorithm Nowadays, more and more reconfigurable hardware devices are used in network applications due to their low cost, high performance and flexibility. Such applications include extensible network routers, firewalls and Internet-enable sensors, etc. These reconfigurable hardware devices are usually distributed in a large geographic area and operated over public networks, making on-site configuration inconvenient or infeasible. Therefore, robust security mechanisms for remote control and configuration are highly needed. The RSA algorithm is a secure, high quality, public key algorithm. 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 (typically thousands of bits long) integers.

The RSA algorithm has been adopted by many commercial software products and is built into current operating systems by Microsoft, Apple, Sun, and Novell. Commercial Application Specific Standard Products (ASSPs) like the security processors offered by several vendors have a much higher RSA performance than software implementation. However, their solution is inflexible and expensive. With the exponential increase in FPGA size over time, it is possible to implement a relatively high performance, user parameterizable RSA at low cost on FPGA.

Vedic mathematics - a gift given to this world by the ancient sages of India. A system which is far simpler and more enjoyable than modern mathematics. The simplicity of Vedic Mathematics means that calculations can be carried out mentally though the methods can also be written down. There are many advantages in using a flexible, mental system. Pupils can invent their own methods; they are not limited to one method. This leads to more creative, interested and intelligent pupils. Vedic Mathematics refers to the technique of Calculations based on a set of 16 Sutras, or aphorisms, as algorithms and their upa-sutras or corollaries derived from these Sutras. Any mathematical problems (algebra, arithmetic, geometry or trigonometry) can be solved mentally with these sutras. Vedic Mathematics is more coherent than modern mathematics. Vedic Mathematics offers a fresh and highly efficient approach to mathematics covering a wide range - starts with elementary multiplication and concludes with a relatively advanced topic, the solution of non-linear partial differential equations. But the Vedic scheme is not simply a collection of rapid methods; it is a system, a unified approach. Vedic Mathematics extensively exploits the properties of numbers in every practical application

Further, when the integrated circuit is a field programmable gate array (FPGA), large amounts of valuable programming resources are required to implement digital division, thus limiting the size or precision of the data words that can be accommodated. In addition, the numerous data paths required to connect the related logic circuitry within a FPGA result in slow performance and in some cases may cause it to malfunction.

2. LITERATURE REVIEW

The paper published by Jainath Nasreen.P and Emy Ramola.P [1], used VHDL to implement a 16- bit RSA block cipher system. The whole implementation includes three parts: key generation, encryption and decryption process. The key generation stage aims to generate a pair of public key and private key, and then the private key will be distributed to receiver according to certain key distribution schemes. The memory usage and overhead associated with the key generation is eliminated by the proposed system model. The cipher text can be decrypted at receiver side by RSA secret key. They have been concluded that the proposed architecture yields better results than the existing one in all the parameters considered. The analysis results were obtained using Design Vision tool of Synopsis. The architecture generates all the possible random numbers for the given 16 bit input and stores it in a memory. Prime numbers from the generated random numbers were identified and again stored in a FIFO memory. The first two prime numbers stored in the FIFO is selected for encryption. But the proposed architecture eliminates the need for memories by checking for primality and selects two prime numbers simultaneously while the random numbers were being generated. Also the generation stops as soon as the system detects two prime numbers.

Modular Multiplication and Exponentiation Architectures for Fast RSA Cryptosystem, Based on Digit Serial Computation ,the paper published by Gustavo D. Sutter, Jean-Pierre Deschamps, and José Luis Imaña[2], in their paper, they optimized the Montgomery’s multiplication and proposed architectures to perform the least significant bit first and the most significant bit first algorithms. The developed architecture has the following distinctive characteristics: 1) use of digit serial approach for Montgomery multiplication. 2) Conversion of the CSA representation of intermediate multiplication using carry–skip addition. This allowed the critical path to be reduced, albeit with a small-area speed penalty; and 3) precompute the quotient value in Montgomery’s iteration in order to speed up the operating frequency. And they concluded that, CSA was used to perform large word-length additions in conjunction with quotient precomputation and digit serial computation. Another characteristic of the proposed architecture was the use of binary representation for intermediate exponentiation results and the use of efficient carry–skip addition at the end of a Montgomery’s multiplication.

For fair comparison, the circuits were implemented in Virtex 2 and Virtex 5 FPGA devices and in 0.18-μm ASIC technologies, presenting in all technologies the best results. A technological observation is that the Xilinx Virtex 5 implementation (65 nm) achieves throughput similar to a 0.18-μm ASIC technology and almost doubles the data rate of a Virtex 2 technologies (0.15 μm).

The paper published by Himanshu Thapliyal and M.B Srinivas [4] , proposed the hardware implementation of RSA encryption/decryption algorithm using the algorithms of Ancient Indian Vedic Mathematics that have been modified to improve performance. The recently proposed hierarchical overlay multiplier architecture is used in the RSA circuitry for multiplication operation. Due to its parallel and regular structure the proposed architecture can be easily laid out on silicon chip 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 RSA circuitry employing traditional multipliers and division algorithm.

The paper proposed by K.Z. Pekmestzi, N.K. Moshopoulos [5], in their paper a new implementation of a Montgomery multiplier is presented, which is based on the direct approach achieving higher performance than any other realization. The circuit is modified in an elegant way in order to implement both the modular multiplication and squaring in a bit-interleaved form. The modular exponentiation requires approximately 2n2 clock cycles with the minimum hardware complexity, reported so far. The proposed design is approximately 2 and 3 times more efficient than respectively. Compared to their circuit’s performance is about 20% higher. This is due to the direct implementation of the Montgomery algorithm, which yields a decrease of the circuit’s complexity, equal to 19 gates per bit.

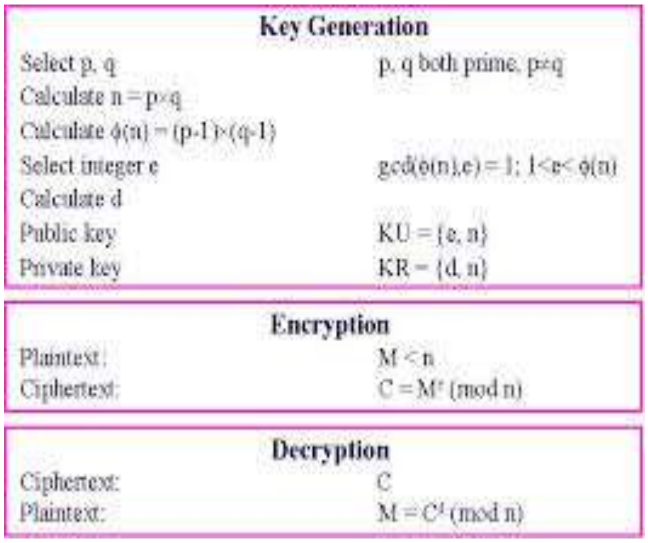
A hardware version of the RSA using the Montgomery's algorithm with systolic arrays has been proposed by Ali Ziya Alkar, Remziye Sonmez [7], where they use systolic arrays, to speed up the modular multiplication and squaring, bit level systolic arrays are used with the Montgomery's modular multiplication algorithm to constitute the core of modular exponentiation operation. The squaring systolic structure is also performed in parallel with the systolic multiplication in the modular exponentiation. The novel idea in their paper was to use the systolic array cells with increased performance of up to 20% and use them in a single row organization. The final RSA design is configurable and can operate both for encryption and decryption. The results are obtained at the high-level synthesis stage and show the highest possible clock rate that can be achieved.

3. PROPOSED WORK

A) RSA ALGORITHM

The RSA algorithm is a secure, high quality, public key algorithm. Fig shows steps involved in key generation , encryption & decryption of system. RSA encryption and decryption are mutual inverses and commutative due to symmetry in modular arithmetic.

The process of transforming plaintext into cipher text is called 'Encryption' or 'Enciphering'. The reverse process of turning cipher text into plaintext, which is accomplished by the recipient who has the knowledge to remove the disguise, is called 'Decryption' or 'Deciphering'. RSA algorithm can generally be further classified into key generation algorithm, encryption algorithm, and decryption algorithm. The RSA key generation algorithm and RSA encryption algorithm and decryption algorithm can be described in the following steps



Key Generation	
Select p, q	p, q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1) \times (q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

Decryption	
Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

Figure 2: Key Generation encryption and decryption steps of RSA Algorithm

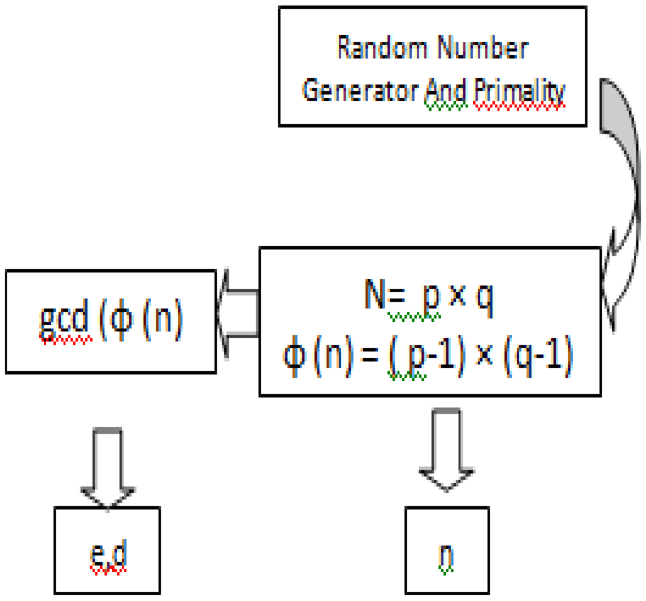


Figure 3: Proposed System architecture for RSA key generation

The system architecture for key generation is shown in Fig 3. A random number generator generates 16-bit pseudo random numbers and the primality tester takes a random number as input and tests if it is a prime ConfIrmed primes component pulls out two primes, and calculates n and ϕ(n) . N is stored in a register. ϕ (n) is sent to the Greatest Common Divider (GCD), where public exponent e is selected such that gcd [ϕ (n), e] = I, and private exponent d is obtained by inverting e modulo I (n). E and d are also stored in registers. Once n, d, and e are generated, RSA encryption/decryption s simply a modular exponentiation operation.

And the flow chart for the overall RSA algorithm is as shown below in figure 4,

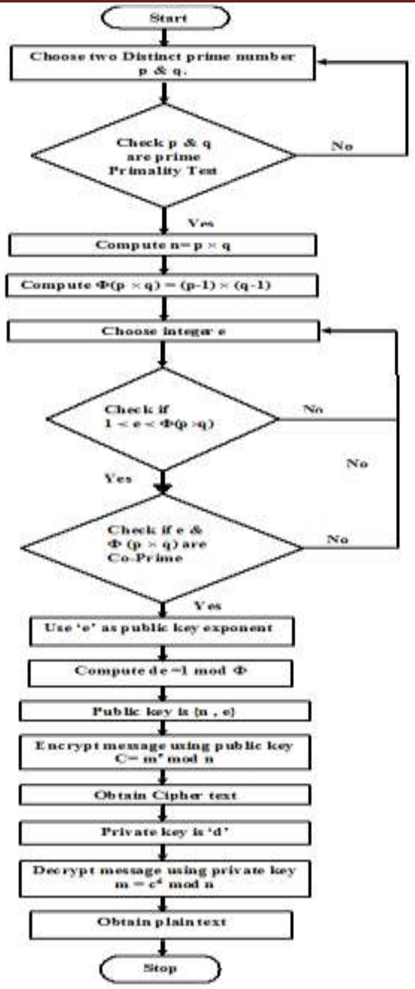

Start
Choose two Distinct prime number p & q.
Check p & q are prime Primality Test
No
Yes
Compute n= p × q
Compute Φ(p × q) = (p-1) × (q-1)
Choose integer e
Check if 1 < e < Φ(p ×q)
No
No
Yes
Check if e & Φ (p × q) are Co-Prime
No
Yes
Use 'e' as public key exponent
Compute d e =1 mod Φ
Public key is {n , e}
Encrypt message using public key C= m^e mod n
Obtain Cipher text
Private key is 'd'
Decrypt message using private key m = c^d mod n
Obtain plain text
Stop

Figure 4: Flow chart of RSA Algorithm

B) VEDIC MATHEMATICS

The basic sutras and upa sutras in the Vedic Mathematics helps to do almost all the numeric computations in easy and fast manner.

Vedic mathematics is mainly based on 16 Sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc. The multiplier is based on an algorithm Urdhva Tiryakbhyam (Vertical & Crosswise) of ancient Indian Vedic Mathematics. Urdhva Tiryakbhyam Sutra 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. Vedic multiplier is faster than array multiplier and Booth multiplier. As the number of bits increases from 8x8 bits to 16x16 bits, the timing delay is greatly reduced for Vedic multiplier as compared to other multipliers. Vedic multiplier has the greatest advantage as compared to other multipliers over gate delays and regularity of structures.

The sutra which we employ in this project is Urdhva Triyagbhyam (Multiplication). The Multiplier Architecture is based on the Vertical and Crosswise algorithm. The architecture is illustrated with two 4-bit numbers; the multiplier and multiplicand, each are grouped as 4-bit numbers so that it decomposes into 4x4 multiplication modules. After decomposition, vertical and crosswise algorithm is applied to carry out the multiplication on first 4x4 multiply modules. The results of first 4x4 multiplication module are utilized after getting the partial product bits parallel from the subsequent module to generate the final 16-bit product. Hence any complex NxN multiplication can be efficiently implemented by using small 4x4 multiplier using the proposed architecture where N is a multiple of 4 such as 8 , 16, ...2N. Therefore efficient multiplication algorithm implementation with small numbers such as 4-bits can be easily extended and embedded for implementing efficient NxN multiply operation. The proposed method of Urdhva Triyagbhyam can be implemented for binary system in the same way as decimal system.

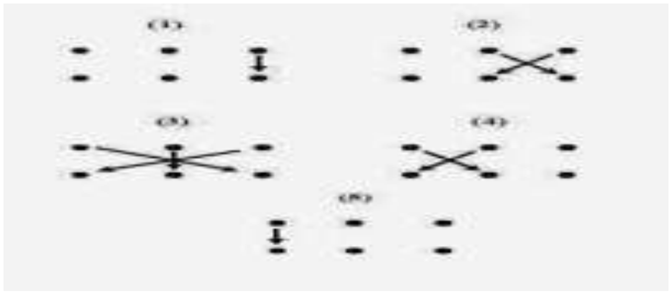


Figure 5: Multiplication steps using Urdhva Tiryakbhyam

The multiplication has been done in a single line in Urdhva method, whereas in shift and add method (Conventional) partial products have to be added to get the result. This implies the increase in speed. Urdhva Tiryakbhyam (Vertically and Crosswise), deals with the multiplication of numbers.

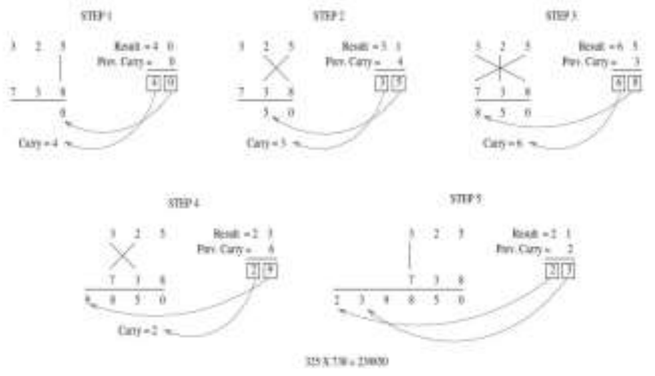


Figure 6: Multiplication of two decimal numbers by Urdhva Tiryakbhyam

This Sutra has been traditionally used for the multiplication of two numbers in the decimal number system. In this paper, we apply the same idea to the binary number system to make it compatible with the digital hardware. Let us first illustrate this Sutra with the help of an example in which two decimal numbers are multiplied. Line diagram for the multiplication of two numbers (325x728) is shown in Fig.6. The digits on the two ends of the line are multiplied and the result is added with the previous carry. When there are more lines in one step, all the results are added to the previous carry. The least significant digit of the number thus obtained acts as one of the result digits and the rest act as the carry for the next step. Initially the carry is taken to be zero.

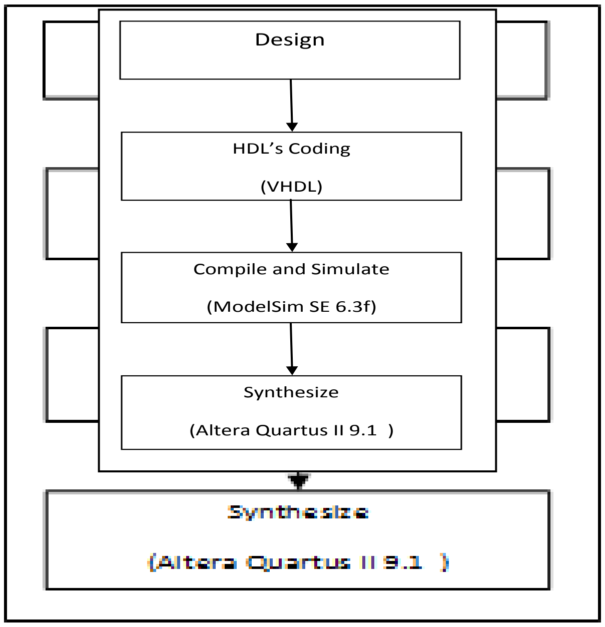


Figure 7: Design Flow

As shown in the project design flow, firstly design the project with the appropriate information then the next step will be of VHDL coding. After HDL designing, the code will simulate and its functionality will verified using simulation software on ModelSim SE 6.3f simulator. ModelSim is a verification and simulation tool for VHDL, Verilog, SystemVerilog, and mixed language designs. The code is simulated and the output is tested for the various inputs. If the output values are consistent with the expected values then we proceed further else necessary corrections are made in the code. Simulation is a continuous process. Small sections of the design should be simulated and verified for functionality before assembling them into a large design. After several iterations of design and simulation the correct functionality is achieved. Once the design and simulation is done then another design review by some other people is done so that nothing is missed and no improper assumption made as far as the output functionality is concerned. And finally we will synthesize and analyze the VHDL design on Altera Quartus II 9.1. The Altera Quartus II design software provides a complete design environment that easily adapts to your specific design requirements.

4. RESULT AND DISCUSSION

In the project design flow, firstly design the specification of the RSA cryptosystem. Then the next step will be of VHDL coding. After HDL coding of the system, the code will simulate and its functionality will verified using simulation software on ModelSim SE 6.3f simulator. ModelSim is a verification and simulation tool for VHDL, Verilog, SystemVerilog, and mixed language designs. The code is simulated and the output is tested for the various inputs. If the output values are consistent with the expected values then we proceed further else necessary corrections are made in the code. Simulation is a continuous process. Small sections of the design should be simulated and verified for functionality before assembling them into a large design. After several iterations of design and simulation the correct functionality is achieved. Once the design and simulation is done then another design review by some other people is done so that nothing is missed and no improper assumption made as far as the output functionality is concerned. And finally we will synthesize and analyze the VHDL design on Altera Quartus II 9.1. The Altera Quartus II design software provides a complete design environment that easily adapts to your specific design requirements.

The fig 8 shows the simulation results of the conventional multiplier obtained using the ModelSim SE 6.3f and the table shows the corresponding design summary obtained using Alter Quartus II9.1 tool.

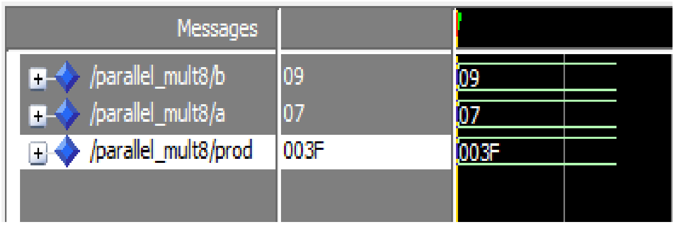


Figure 8: Simulation Result of Conventional multiplier

The fig 9 shows the simulation results of the 8 bit vedic multiplier obtained using the ModelSim SE 6.3f and the table shows the corresponding design summary obtained using Alter Quartus II9.1 tool.

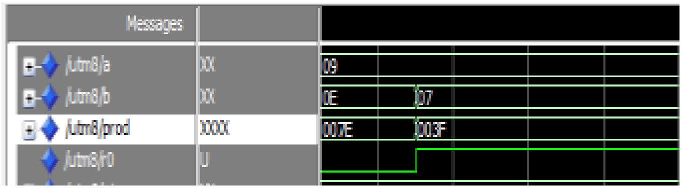


Figure 9: Simulation Result of Vedic multiplier

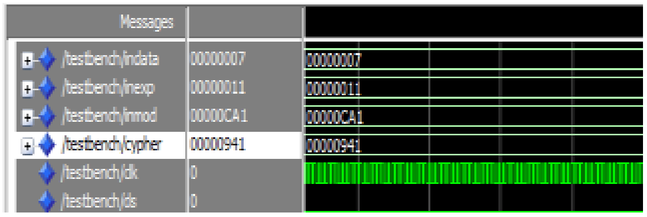
Table 1: Design Summary of Vedic and Conventional multiplier

Parameter	**Conventional Multiplier**	**Vedic Multiplier**
Number of logic elements	189	167
Number of combinational functions	189	167
Propogation delay	161.089ns	57.608ns
Maximum frequency	6.207MHz	17.3MHz
Power	70.58mW	70.49mW

The summary of overall design for both conventional multiplier and vedic multiplier is summarized as in the table 1.From this summary we can say that we achieves a significant improvement in performance using the Vedic multiplier as reflected by the results shown in table 1 . It is found that when implemented with Vedic multiplier, the RSA circuitry has less timing delay compared to its implementation using traditional multiplier. So from summarized table, it is proved that Vedic multiplier is efficient than conventional multipliers in terms of area/speed.

In this chapter we present the test environment and the experimental results of our design modules. Our objectives of this paper are to design and implement the RSA cryptosystem to improve speed performance, area reduction and throughput. In this project we have been implemented the RSA cryptosystem is simulated using Modelsim environment and a tool from Altera Quartus II9.1 used for FPGA design.
Here we firstly implemented RSA cryptosystem which includes Encryption and Decryption in Modelsim simulation Environment which was an interesting task to design that module.

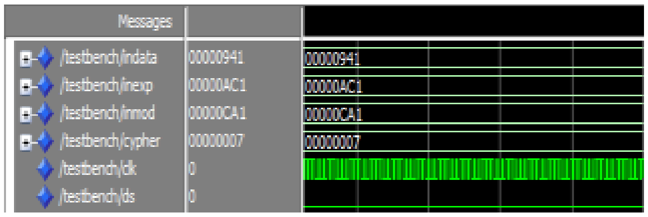
To begin the testing of encryption /decryption process, a set of RSA parameters has calculated. Then the calculated parameters have been fed into the RSA_CORE module and the results have been compared with the theoretical values. Assumed that p = 61, q = 53. To generate the two keys, the product is computed as, $n = p \ x \ q = 3233$**.** 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 = 17 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 = 2753. For each of the key size following input set is used:
Encryption: Message, $M = 7_{16}$
Public key, $e = 11_{16}$
Modulus, $n = \text{CA1}_{16}$



```
Messages
/testbench/indata   00000007   00000007
/testbench/inexp    00000011   00000011
/testbench/inmod    00000CA1   00000CA1
/testbench/cypher   00000941   00000941
/testbench/clk      0
/testbench/ds       0
```

Figure 10: Simulation Result of Encryption

Decryption: Message, $C = 941_{16}$
Private Key, $d = \text{AC1}_{16}$
Modulus, $n = \text{CA1}_{16}$
Thus, public key = (17, 3233), private key = (2753, 3233).



```
Messages
/testbench/indata   00000941   00000941
/testbench/inexp    00000AC1   00000AC1
/testbench/inmod    00000CA1   00000CA1
/testbench/cypher   00000007   00000007
/testbench/clk      0
/testbench/ds       0
```

Figure 11: Simulation Result of Decryption

Table 2: Design Summary of RSA cryptosystem

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Percentage of use**
Number of logic elements	912	18,752	5%
Number of combinational functions	808	18,752	4%
Number of logic registers	459	18,752	2%
Number of pins	132	315	42%
Number of memory bits	0	239,616	0%
Number of embedded multiplier 9-bit elements	0	52	0%

Table 3: Performance analysis of RSA algorithm using Vedic and conventional multiplier

Parameter	**Conventional Multiplier**	**Vedic Multiplier**
Total pins	148	132
Propogation delay	8.201ns	7.220ns
Maximum frequency	121.9MHz	138.5MHz
Power	84.21mW	82.20mW

The performance was compared in terms of speed and power. The analysis result was shown above in table 5.4.2. From the above table it can conclude that the proposed architecture yields better results than the existing one in all the parameters considered. The efficiency will be increased by a considered amount in terms of the above mentioned parameters.

5. CONCLUSION

The RSA algorithm is important to network security because they are the components (i.e. encryption and decryption key) which interact with the security system, without them the system will be useless as RSA are used to fire a particular encryption and decryption keys process because of which security system is build. The RSA algorithm has the same importance as of the system in the cryptography over network security. Since RSA are used to provide the authentication and privacy to whole system. The main advantage of RSA algorithm is enhanced security and convenience. Using public key encryption is also an advantage of this algorithm. Only the RSA lacks in encryption speed because of its mathematical calculation and to prevent this, the multiplier based on Vedic mathematics is used which is one of the fast and low power multiplier. Employing this technique in the algorithm which reduces the complexity, execution time, power etc. Urdhva tiryakbhyam sutra is the most efficient sutra, giving minimum delay for multiplication of all types of numbers, either small or large and it eliminates unwanted multiplication steps as compared to conventional multiplication. So the RSA algorithm is implemented using the high speed multiplier sutras exhibits improved efficiency in terms of speed. And in this paper the RSA algorithm using vedic mathematics will be prototype using VHDL and its analysis will be base on the FPGA

6. REFERENCES

[1] Jainath Nasreen.P Emy Ramola.P, A Novel Architecture for VLSI Implementation of RSA Cryptosystem, 2012 IEEE
[2] Gustavo D. Sutter, Jean-Pierre Deschamps, and José Luis Imaña, Modular Multiplication and Exponentiation Architectures for Fast RSA Cryptosystem, Based on Digit Serial Computation, IEEE Transactions on industrial electronics, VOL. 58, NO. 7, JULY 2011
[3] A.R.Landge, A.H. Ansari , RSA algorithm realization on FPGA ,International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 7, July 2013
[4] Himanshu Thapliyal and M.B Srinivas, VLSI Implementation of RSA Encryption System, 2005
[5] K.Z. Pekmestzi*, N.K. Moshopoulos, A bit-interleaved systolic architecture for a high-speed RSA system 19 October 2001
[6] Chiranth E Chakravarthy H.Y.A, Nagamohanareddy P, Umesh T.H, Chethan Kumar M, " Implementation of RSA Cryptosystem Using Verilog', International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-20 11 I ISSN 2229-55 1 8
[7] Ali Ziya Alkar, Remziye Sonmez, A hardware version of the RSA using the Montgomery's algorithm with systolic arrays, 2004
[8] G.P. Saggese, L. Romano, N. Mazzocca, A. Mazzeo , A tamper resistant hardware accelerator for RSA cryptographic applications 2005
[9] Tzong-Sun Wu, Han-Yu Lin," Secure Convertible Authenticated Encryption Scheme Based on RSA ", Infor matica 3 3 (2009) 4 8 1 -486.
[10] Guilherme Perin, Daniel Gomes Mesquita, and Jo-ao Baptista Marti ns, "Montgomery Modular Multiplication on Reconfigurable Hardware Systolicversus Multiplexed Implementation ", Hindawi Publishing Corporation International Journal of Reconfigurable Computing Volume 2011.
[11] Cryptography and Network Security by William Stallings, Third Edition, Pearson Education
[12] Cryptography and Network Security by Atul Kahate, Tata McGraw Hill,2003

7. AUTHOR PROFILE

	Miss. Dhanashri R. Kadu Received Bachelor's Degree in Electronic & Telecommunication from Sant Gadge Baba Amravati University in 2012 & Pursuing Master Degree in EXTC from Sipna College of Engineering. Amravati-444605.
	DR. G. P. Dhok Head, Department of Instrumentation, Sipna College of Engineering Amravati -444605.