FPGA Implementation of 16 bit RSA Cryptosystem for Text Message

Rohith S
Assistant Professor
Department of E&C
NCET, Bangalore,
Karnataka, India.

Poornima
Research scholar
Department of E&C
NCET, Bangalore,
Karnataka, India.

Mahesh C
Research scholar
Department of EC
NCET, Bangalore,
Karnataka, India

ABSTRACT
The rapid growth of the internet and electronic commerce has brought to the forefront the issue of privacy in electronic communication. In order to protect the information from unauthorized parties we need to mask the information before sending it through a communication channel. Currently RSA is one of the algorithms which are not broken by hackers due to its mathematical complexity. This paper presents a design and implementation of 16-bit RSA Cryptosystem. The entire cryptosystem is divided in to three parts: key generation, encryption and decryption. The key generation is carried out by using random number generator LFSR, "Sieve of Eratosthenes" algorithm for prime number detection, Booth multiplier for multiplication and Extended Euclidean algorithm to find GCD of the public key and Euler's Totient Function. Encryption and decryption is carried out by Modular Multiplication and Modular Exponentiation by using LR binary method. The encryption and decryption of the text message "NAGARJUNA COLLEGE" is verified using proposed RSA algorithm. The design is simulated using Modelsim10.2b simulator and finally implemented on Sparten-6 FPGA using Xilinx 13.4 software. Area and timing parameters are computed with respect to Spartan-6 FPGA. The results obtained from simulation are validated using MATLAB code.

Keywords:
RSA, FPGA, Verilog, Cryptography, Encryption, Decryption.

1. INTRODUCTION
Cryptography is concerned with keeping communications private. Modern cryptosystems are typically classified as either public-key or private-key [1]. Private-key encryption methods, such as the Data Encryption Standard (DES), use the same key to both encrypt and decrypt data. The key must be known only to the parties who are authorized to encrypt and decrypt a particular message. Public-key cryptosystems, on the other hand, use different keys to encrypt and decrypt data. The public-key (e, n) is globally available. The private-key (d, n) is kept confidential Private-key systems suffer from the key distribution problem. In order for a secure communication to occur, the key must first be securely sent to the other party. An unsecure channel such as a data network cannot be used. Public-key systems do not suffer from this problem because of their use of two different keys. Messages are encrypted with a public key and decrypted with a private key. No keys need to be distributed for a secure communication to occur. The RSA public-key cryptosystem is the most popular form of public-key cryptography [2-5].

The RSA algorithm capitalizes on the fact that there is no efficient way to factor very large numbers. As the number become larger and larger, finding out the factor for that number becomes extremely difficult. Even if the public key is known by the unintended users, the data encrypted by the RSA cannot be decrypted. This gives an additional level of security.

RSA algorithm crypto system is most widely used Public key cryptographic system[1-8]. RSA algorithm is very Good fit for real time data security, high performance applications like image capture with Encrypt/Decrypt, Financial application like Credit card swipper etc will take out the risk of tampering if implemented in hardware, High speed E-comers data transmission/reception over internet. With the high quality, guaranteed security and reliability the RSA algorithm represents a real turning point for practical applications. Many researchers proposed algorithm for RSA encryption and decryption process. VLSI architecture to compute modular exponentiation and modular multiplication for RSA public-key cryptosystem is discussed in [7]. The conventional H-algorithm is modified to find the modular exponentiation. Author shown modified H-algorithm will reduces computational complexity of the multiplication process. Result shows the architecture of the modified modular multiplication the iteration times are only half of Montgomery's algorithm and the H-algorithm. In [8] the architecture and modeling of RSA public key encryption/decryption systems is discussed. Architecture uses simple shift and add algorithm is used to implement the blocks. It makes the processing time faster and used comparatively smaller amount of space in the FPGA due to its reusability. The VHDL code is synthesized and simulated using Xilinx-ISE 10.1. It is verified that this architecture support multiple key of 128bits, 256bits, and 512 bits. RSA encryption algorithm, Montgomery unit for multiplication and FPGA implementation RSA architecture are discussed in[9].

This Paper discusses implementation of 16-bit RSA algorithm to encrypt and decrypt the text messages. The entire RSA cryptosystem is divided into three parts: key generation, encryption and decryption. The key generation stage aims to generate a pair keys, i) public key ii) private key. Anyone knowing the public key can encrypt messages, but cannot decrypt. Key generation is carried out by using LFSR , Sieve of Eratosthenes algorithm[ ] for prime number detection, Booth multiplier for multiplication and Extended Euclidean algorithm to find GCD. Encryption and decryption is carried out by modular multiplication and modular exponentiation by using LR binary method[ ]. The text message "NAGARJUNA COLLEGE" is encrypted and decrypted using proposed RSA algorithm. The architecture is synthesized using Xilinx-13.4 with respect to Spartan-6 FPGA and simulated using Modelsim.

The rest of the paper is organized as follows. Section I describes proposed design. Implementation and simulation

results are given section II. In Section III gives the conclusion. The final section IV gives the references used

2. PROPOSED DESIGN

The RSA algorithm was invented by Rivest, Shamir, and Adleman in 1977 and published in 1978 [3]. It is one of the simplest and most widely used public-key cryptosystems. The RSA algorithm has 3 steps and the general description of the overall components of RSA algorithm.

Key Generation

In RSA [1] data encryption/ decryption uses public key and a private key. The public key can be known to 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. To generate the keys following algorithm is used.

1. Using Linear Feedback Shift Register (LFSR) with a polynomial $X^{16}+X^{14}+X^{13}+X^{11}+1$ 16 bit number is generated. Here integer number of bit length 16 is considered.
2. The so obtain number is random undergone for the primality test using Sieve Eratosthenes process. If the number is prime then assigned as a 'p' or 'q' else other number will be obtained from LFSR.
3. So obtained random number p and q are used to compute 'n' using n = p*q Where 'n' is the for both public and private keys. '*' is multiplication symbol implemented using 16 bit Booth algorithm.
4. Compute φ (n) = (p − 1)*(q − 1), where φ is is the Euler's Totient Function computed using booth multiplier.
5. Choose an integer e such that $1 < e < \phi$ (n) and gcd(e, ϕ(n)) = 1; i.e. e and φ(n) is a co prime 'e' is released as the public key exponent.
6. Determine d as $d \equiv e^{-1} \pmod{\varphi(n)}$, i.e., d is the multiplicative inverse of e (modulo φ(n)). This is more clearly stated as solve for d given de=1 (mod φ(n)). This is often computed using the extended Euclidean algorithm. 'd' is kept as the private key exponent.

By construction $de=1 \pmod{\varphi(n)}$.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.

Encryption

The encryption works using the public key (e, n). For any input message "M", the encrypted message "C" would be calculated as: $C=M^e \bmod n$. The modular exponentiation is carried out using standard LR binary algorithm[ ].

Decryption

The decryption works using the public key (d, n). For any input encrypted message "C", the original message "M" would be calculated is using: $M=C^d \bmod n$. Modular exponentiation is carried out using LR binary algorithm.

3. IMPLEMENTATION AND SIMULATION

The proposed design is implemented using Verilog HDL programming language and simulation is done using Modelsim 10.2b simulator. The design is synthesized with respect to Spartan-6 FPGA. RSA algorithm is developed using random number generator, prime number detector, multiplier, Extended Euclidean Algorithm, encryption and decryption. All these sub modules are implemented and integrated to generate required output so that to achieve the functional requirement of the project.

3.1 Random number generator

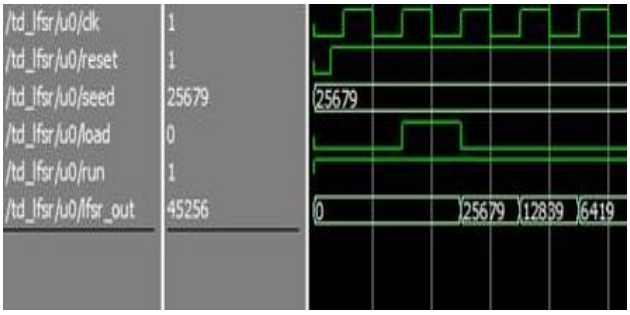

Fig 1: Simulation result of 16-bit random number generator

Fig 1 shows the simulated result of LFSR. For example seed value given to the 16-bit LFSR is 0110010001001111 [equivalent decimal value is 25679]. The sequences obtained are 0011001000100111 [equivalent decimal value is 12839], 001100100010011 [equivalent decimal value is 6419] etc. this is clearly depicted in waveform.

3.2 Prime number detector

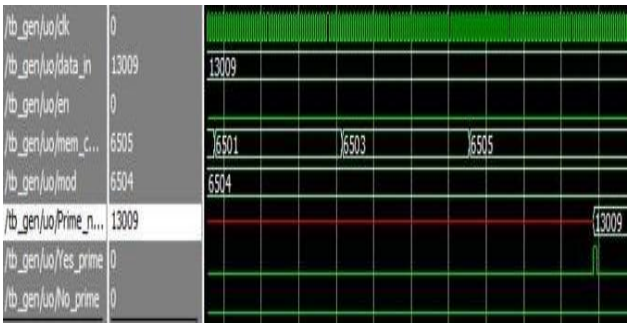

Fig 2: Simulation result of 16-bit prime number detector

The sequence generated from LFSR module is given as input to the prime number detector module. This module will test whether the number is prime or not. Fig 2 shows the simulation result of prime number detection module. For example input value given to the prime number detector is 13009. The number is detected as a prime number so that the output signal Yes_Prime is toggled and the output signal prime_out outputs the prime number.

3.3 32-bit Booth multiplier

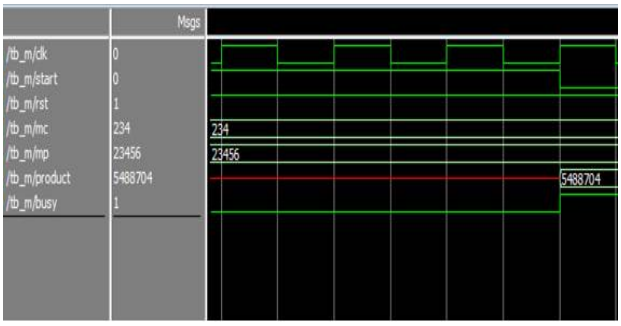

Fig 3: Simulation result of 32-bit Booth Multiplier

Fig 3 shows the waveform of the multiplication of two numbers. The inputs i.e. 32-bit multiplicand 'mc' is 234 and 32-bit multiplier 'mp' is 23456 given to the multiplier are 234 and 23456. When 'start' signal is high the multiplication operation will start and which gives the 64 bit product i.e. product=5488704.

3.4 Extended Euclidean algorithm

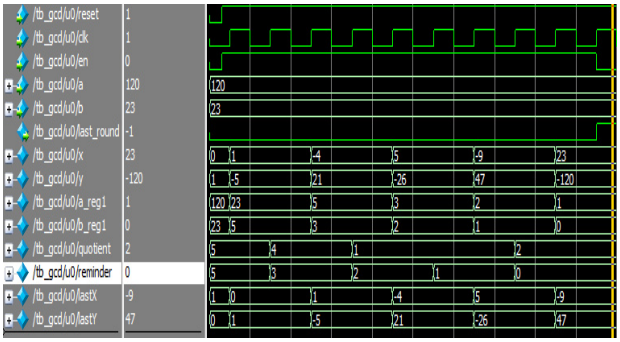

Fig 4: Simulation result of Extended Euclidean algorithm

Extended Euclidean algorithm has been implemented to find the multiplicative inverse of the e (modulo φ (n)). Fig 4 shows the simulation result of the Extended Euclidean algorithm. For example input 'e' is taken as 'a' and φ (n) is 'b' where a=120 and b=23. Last_Y is the required output that is 47.

3.5 Encryption

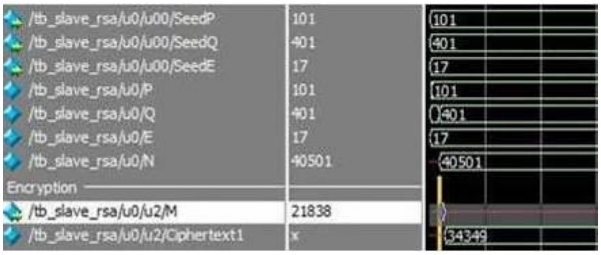

Fig 5: Simulation result of Encryption

The public key (e,n) obtained is used to Encrypt the data using RSA cryptosystem. The simulation result for the input M=21838 is shown in fig 5 Random number sequence is generated using LFSR with respect to the seed value given and it is tested by prime number detection algorithm. The prime numbers obtained are named as 'P' and 'Q'. So the prime detector gives two prime number as output that is 'P' and 'Q'. Then the modulus 'N' and 'φ (n)' is found using booth multiplier by performing N=p*q=401*101, φ(n)=(p-1)*(q-1)=400*100. The output value obtained are n= 40501 and φ(n)= 40000.The prime number in the range of 1 to 256 is entered to find 'E'. It will check for the condition $1< E <\varphi(n)$ and gcd(E,φ(n))=1. If the condition is satisfied then that number is taken as 'E' i.e. E=17. Then the data is encrypted using this public key 'E' according to cipher text $C=M^E$ mod N. Thus the encrypted data obtained is Ciphertext1=34349.

3.6 Decryption

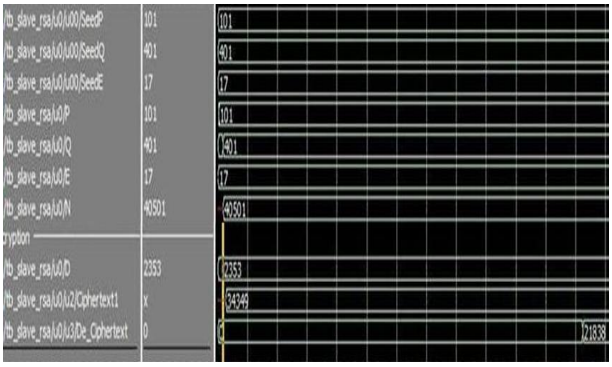

Fig 6: Simulation result of Decryption

Fig 6 shows the waveform of the decryption. The data is decrypted using the private key 'D' according to $M=C^D$ mod N. For example the inputs are private key D=2353, encrypted data Ciphertext1=34349 which is obtained in the encryption process and modulus N=40501. Decrypted data is same as the original data which was used during encryption i.e. De_Ciphertext=21838.

3.7 Simulation result for encryption of multiple characters

Simulation result shows the encryption and decryption of multiple characters "NAGARJUNA COLLEGE". In this two characters are taken at a time and it is converted to ASCII value, which is used for encryption and decryption.

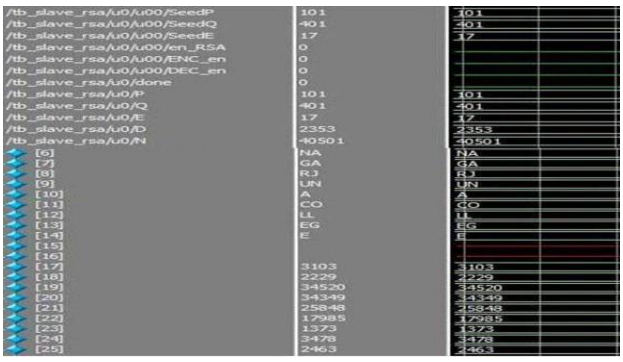

Fig 7: Simulation result for encryption of multiple characters

The figure 7 shows the encryption of multiple characters. The entire module operation will start with toggling the En_RSAsignal.All the control signals and information signal or the message signal are stored in memory. The address 0 is reserved for Seed_P, address 1 is reserved for Seed_Q, address 2 is reserved for Seed_E, address 3 is reserved for En_RSA, address 4 is reserved for En_Encryption and address

5 is reserved for En_Decryption. Messages are stored from address 6 to address 15, as "NAGARJUNA COLLEGE"; this is stored in the form of ASCII equal hex value, each ASCII character will take 8 bits or 1 byte ; as the input message for the module is 16 bits wide, the whole information is broken into 2 character for each message.

So once the En_RSA signal is toggled, LFSR module will take SEED_P, Seed_Q and generate a random number which is fed to the prime number detector. The output of prime number detector is named as P and Q where P=401 and Q=101. By using two detected prime numbers N=p*Q=40501 and φ(N)=(P -1)*(Q-1)=40000 are computed. Then the public key "E" is generated according to the condition 1<E<φ (N) and gcd(E,φ (N))=1. The obtainedE=17 and φ (N)=40000 is fed to "Extended Euclidean Algorithm" module to get "D" value.

Here the two characters of message are stored in the form of ASCII equal hex value i.e. NA=4E41, GA=4741, RG=524A, UN=554E, A=41, CO=434F, LL=4C4C, EG=4547, E=45. Once the P, Q, E, N and D are obtained and En_Encryption signal is toggled; the encryption module will take value of E, N and message from address 6 to address 15. The resultant encrypted data are NA=3103, GA=2229, RJ=34520, UN=34349, A=25848, CO=17385, LL=1373, EG=3478, E=2463. These encrypted data are stored between address 17 to address 26.

3.8 Simulation result for decryption of multiple characters

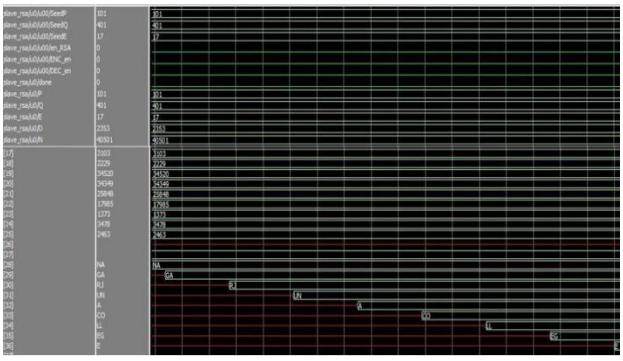

Fig 8: Simulation result for decryption of multiple characters

Fig 8 shows the simulation result for decryption of multiple characters. The inputs to the decryption module are private key 'D' which is found using Extended Euclidean algorithm, encrypted data obtained in the encryption process and modulus N. Decryption is carried out by taking two characters at a time. It will be the original message given in the encryption process i.e. "NAGARJUNA COLLLEGE". When the En_Decryption signal is toggled the decryption module will take the D,N and encrypted data from address 17 to address 26 and stores the decrypted data between address 27 to address 36.

3.9 Area utilization of RSA cryptography

The below chart shows the area utilized for RSA cryptography by Xilinx Spartan 6

Table 1: Utilized area for RSA cryptography

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	1804	4656	38%
Number of Slice Flip Flops	1811	9312	19%
Number of 4 input LUTs	3212	9312	34%
Number of bonded IOBs	100	232	43%
Number of MULT18X18SIOs	3	20	15%
Number of GCLKs	9	24	37%

The above table shows the device utilization summary of whole RSA module in the device. Spartan 6 (XC6SLX16). The whole implementation of RSA cryptography takes 1804 slices.

3.10 Latency & Clock Frequency of RSA cryptography

The below chart shows the maximum frequency of RSA in Xilinx Spartan 6

```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 16.308ns (Maximum Frequency: 61.321MHz)
   Minimum input arrival time before clock: 5.141ns
   Maximum output required time after clock: 4.063ns
   Maximum combinational path delay: No path found
```

Fig 7: Time summary of RSA cryptography

The timing summary shows the maximum operating frequency is 61.321MHz for RSA cryptography.

4. CONCLUSION

Cryptographic algorithm plays an important role for the transmission of the data over the internet. There have been lots of the cryptographic algorithms that have designed in the past. RSA is one of most widely used cryptographic algorithm in recent times. In this project 16-bit RSA cryptosystem is implemented on FPGA. The RSA cryptosystem includes random number generator LFSR, "Sieve of Eratosthenes" algorithm for prime number detection, Booth multiplier for multiplication, Extended Euclidean Algorithm for key generation, L-R binary algorithm for encryption and decryption. The design is simulated using Modelsim simulator 10.2b, and also synthesized and implemented on Sparten-6 FPGA using Xilinx 13.4 software, and the area and speed are tabulated with respect to Sparten-6 FPGA. It shows implementation of RSA cryptosystem takes 1804 slices and maximum operating frequency is 61.321MHz .The results obtained from simulation are validated using MATLAB code.

5. REFERENCES

[1] Whitfield Diffie and Martin E. Hellman "New Directions in Cryptography" IEEE Transactions On Information Theory, vol. It-22, no. 6, november 1976.

[2] Woei-JiunnTsaur and Chih-Hung Wang "A new message-recovery combined fair blind signature scheme with provable security using self-certified pairing-based

cryptosystem" International Journal of Innovative Computing, Information and Control Volume 8, Number 2, February 2012.

[3] R.L .Ri vest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Pub lic- Key Cryptosystems", communicatios of the ACM 2 1 (1 9 7 8)

[4] 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 1 I ISSN 2229-55 1 8

[5] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". February, 1978 Volume 21, Number 2 pp. 120-126.

[6] Ridha Ghayoula, ElAmjedHajlaoui, TalelKorkobi, MbarekTraii, HichemTrabelsi, "FPGA Implementation of RSA Cryptosystem", World Academy of Science, Engineering and Technology 20 2008.d. 7, No 4, pp. 241_250.

[7] Jen-Shiun Chiang, Cheng-Chih Chien" An Efficient VLSI Architecture for Rivest Shamir-Adleman Public-key Cryptosystem**"** Tamkang Journal of Science and Engineering, Vol. 7, No 4, pp. 241_250.

[8] Sushanta Kumar Sahu "FPGA Implementation of RSA Encryption System" International Journal of Computer Applications (0975 – 8887) Volume 19– No.9, April 2011.