Usage of FPGA in Network Security

Senthil Kumar M.[1] and S. Rajalakshmi[2]

[1] Department of Electronics and Communication Engineering
Sri Chandrasekharendra Saraswathi Viswa Maha Vidyalaya,
University, Enathur, Kanchipuram-631561, Tamil Nadu, India
[2] Department of Computer Science Engineering
Sri Chandrasekharendra Saraswathi Viswa Maha Vidyalaya,
University, Enathur, Kanchipuram-631561, Tamil Nadu, India
senthilkumar.murugesan@ymail.com,
raji.scsvmv@gmail.com

Abstract. This paper approaches to develop the RSA algorithm using FPGA that can be used as a standard device in the secured communication system. This RSA algorithm is implemented in the FPGA with the help of VHDL and works with radio frequency range to make the information safer. A simple nested loop addition and subtraction have been used in order to implement the RSA operation. This results in less processing time and less space in the FPGA. The information to encryption is in the form of statement or file and the same will appear in the decryption. The hardware design is targeted on Xilinx Spartan 3E device. The RSA algorithm design has made use of approximately 1000 total equivalent gate counts and achieved a clock frequency of 50.00MHz

Keywords: Cryptography, Communication, FPGA, Security, VHDL.

1 Introduction

The enormous advances in network technology have resulted in an amazing potential for changing the way we communicate and do business over the Internet. However, for transmitting confidential data, the cost-effectiveness and globalism provided by the Internet are diminished by the main disadvantage of public networks: security risks. As security plays a vital role in the communication channel, the development of a new and efficient hardware security module has become the primary preference. A vast number and wide varieties of works have been done on this particular field of hardware implementation of RSA algorithm. A hardware implementation of RSA scheme has been proposed by Hani, et al. [1], where they use Montgomery algorithm with modular multiplication and systolic array architecture. Ibrahimy, et al. [2] have proposed to implement RSA algorithm with flexible key size but they have given the input to the RSA encryption side in the form of binary values directly. This work approaches hardware implementation of RSA algorithm scheme using the modular exponentiation operation. In this design, it is possible to change the key size of RSA according to the application requirement and it could take the information either in the form of statement or file.

2 Design Methodology

An exceptional feature that can be found in the RSA algorithm [3] is that it allows most of the components used in encryption to be re-used in the decryption process, which can minimize the resulting hardware area. In RSA, a plaintext block M is encrypted to a cipher text block C by: $C = M^e \bmod n$. The plaintext block is recovered by: $M = C^d \bmod n$. RSA encryption and decryption are mutual inverses and commutative as shown in equation, due to symmetry in modular arithmetic. One of the potential applications for which this design of RSA has been targeted is the Secured Data Communication. In this application, the data input could be either a statement or a file which is fed into FPGA board directly via serial communication. The encryption module takes care of the security. The process at the receiving end is same as the process that has been followed at the sending end except that the sequence of the module is reverse. The RSA covers both the operation of Encryption and Decryption.

2.1 Complete Algorithm

The difficult part of RSA encryption/decryption lies on the modulus calculation of $c=m^e \; mod \; n$, to get the encrypted message "*c*". To calculate the encrypted message "*c*", it involves exponentiation that requires large amount of combinational logic, which increases exponentially with the number of bits being multiplied. The way to accomplish this is by using a sequential circuit, which realizes the exponentiation as a series of multiplications, and multiplication could also be realized as a series of shifts and conditional additions where an exponentiation is separated into a number of multiplications and squaring. Each multiplication can be realized by a series of additions. To reduce the hardware size, modulus (*mod n*) is performed as a number of subtractions inside the multiplication loops. Based on the algorithm described above, a Finite State Machine (FSM) has been developed for RSA. With the reference of this FSM, the VHDL code for RSA has been developed. The VHDL code for Data Control, UART and device drivers are also developed to feed the data to FPGA. Hence, RSA encryption requires a large loop to perform exponentiation, with a smaller inner loop to perform the multiplication. Within the multiplication loop, additions are used to substitute the multiplication. For each loop in addition, the divisor or modulus is subtracted from the working result whenever the working result becomes larger than the divisor, and leaving the modulus when encryption is done.

2.2 Steps for RSA Algorithm Implementation

The VHDL code comprises the following algorithm steps,

Step 1: Choose two prime numbers, such as P and Q & Compute N = PQ
Step 2: Compute Φ (N) = (P-1) * (Q-1)
Step 3: Choose any number $1 < e < \Phi$ (N) that is co prime to Φ (N) and 'e' is not a divisor of Φ (N). Find d value using (d * e) mod Φ (N) =1
Step 4: The Public Key is (e, N) $C = (M^e) \bmod N$
Step 5: The Private Key is (d, N) $M = (C^d) \bmod N$

3 VHDL Modeling

VHDL (VHSIC hardware description language) is a hardware description language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays and integrated circuits. This language is fully based on the IEEE 1164 Standards. Here, the whole system is designed with the help of VHDL. The RTL model of this approach is shown in the Figure.1.

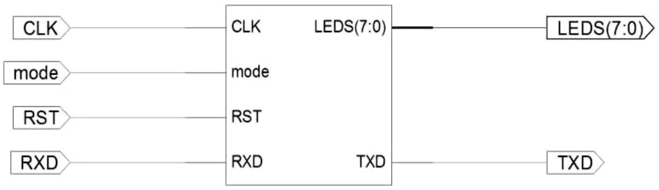


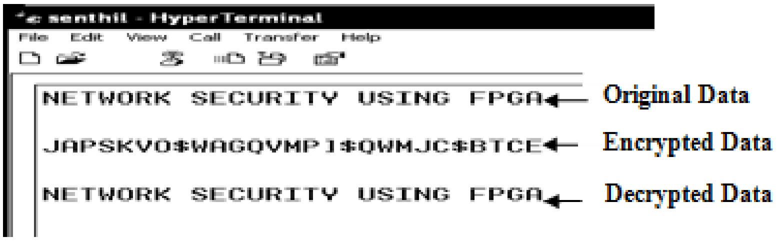
Fig. 1. Overall RSA RTL model in FPGA

4 Simulation, Synthesis and Discussion

VHDL is commonly used to write text models that describe a logic circuit. Such a model is processed by a synthesis program, only if it is part of the logic design. VHDL has file input and output capabilities, and can be used as a general-purpose language for text processing, but files are more commonly used by a simulation test bench for stimulus or verification of data. There are some VHDL compilers which build executable binaries. Here, we are using VHDL program (for RSA, Data Control, UART and device drivers) to write a test bench, to verify the functionality of the design using files on the host computer to define stimuli, to interact with the user, and to compare results with those expected. After the generation of codes that simulates successfully, it may not be synthesized into a real device or is too large to be practical. So, we are designing hardware in a VHDL IDE for FPGA implementation using Xilinx ISE to produce the RTL schematic of the desired circuit. Finally, the VHDL model is translated into the gates and wires that are mapped onto a programmable logic device FPGA. Hence it is the actual hardware which is configured as processor chip rather than the VHDL code which is used for implementing the RSA Algorithm. Once synthesis is over, the input message is in the form of data or file and can be given as input to FPGA from computer via Hyper Terminal or Ethernet in serial communication media. This message is moved to FPGA in binary form with selected bit values one by one. Now, RSA algorithm which is implemented using VHDL program in FPGA processes this message and produces encrypted data as output. So, the output, which is not in readable format, can be saved in note pad or it is directly available in a file. Now, the encrypted file can be sent for the decryption process to get the original message.

5 Output from FPGA via Hyper Terminal

The figure 2 shows the output of Xilinx Spartan 3E FPGA device. At first the input fed to the FPGA device under normal mode. Then, the same input can be given under encryption mode to get encrypted data. This encrypted data can be saved in note pad to transmit to the receiver side. Now, the encrypted data can be given under decryption mode in the receiver side to get original data.



```
NETWORK SECURITY USING FPGA

JAPSKVO$WAGQVMP1$QWMJC$BTCE

NETWORK SECURITY USING FPGA
```

Fig. 2. Output via Hyper Terminal

6 Conclusion

The primary goal of this research project is to develop RSA algorithm on FPGA which can provide a significant level of security and can be faster also. The maximum bit length for both the public and private key is 1024-bit. Beside the security issue, another major concern of this research project is to process the data or file as input. The VHDL language provides a useful tool of implementing the algorithm without actually drawing large amount of logic gates. Although the current key size of this RSA algorithm can provide a sufficient amount of security, a larger key size can always ensure better security.

References

1. Hani, M.K., Lin, T.S., Shaikh-Husin, N.: FPGA Implementation of RSA Public Key Cryptographic Coprocessor. In: TENCON, Kuala Lumpur, Malaysia, vol. 3, pp. 6–11 (2000)
2. Ibrahimy, M.I., Reaz, M.B.I., Asaduzzaman, K., Hussain, S.: FPGA Implementation of RSA Encryption Engine with Flexible Key Size. International Journal of Communications 1(3) (2007)
3. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
4. Mazzeo, A.: FPGA Based Implementation of a Serial RSA Processor: Design, Automation and Test in Europe, vol. 1. IEEE Computer Society, Washington, DC (2003)
5. Ghayoula, R., Hajlaoui, E.A.: FPGA Implementation of RSA Cryptosystem. International Journal of Engineering and Applied Sciences 2(3), 114–118 (2006)
6. Senthil Kumar, M., Rajalakshmi, S.: Effective Implementation of Network Security using FPGA. In: Recent Issues in Network Galaxy (RING 2011), Chennai, India, pp. 52–56 (2011)