Abstract—This paper discusses the implementation of the 1024-bit RSA algorithm based on Altera Cyclone EP1C12 FPGA with Montgomery algorithm and key scanning from left to right. This paper completes the partition of the total architecture, solves the complicated operations, gives the implementation methods and simulation results. According to the implementation methods, we get a configuration file and download it to FPGA chip, and implement the function of data encryption/decryption successfully.

Index Terms—RSA, FPGA, Montgomery algorithm, Information Security

I. INTRODUCTION

With the rapid development of computer technology, we have entered an information society. Every day the network is transporting, exchanging and processing lots of information. How to ensure the security of information and access issues is increasing attention. The importance of network is information security. In order to ensure information security, we can use cryptography technology to encrypt, certificate and sign. Encryption technology is an effective method of core technology to ensure information security and network security.

Modern encryption algorithm is divided into symmetric and asymmetric cryptosystem. RSA is an asymmetric cryptosystem. RSA is not only used in the data encryption, but also in the digital signature. Therefore it provides one essential method for the information encryption and the certification of the public network. It produces a pair of key. One of them is the private key which is preserved by the users, another is the public key which may be opened to outside, even to network. People will send the files which are encrypted by public key to others who accept the files and decrypt the files by private key. For enhancing the security, the RSA key length is 512 bits at least. In this paper we concentrate on the RSA which key length is 1024 bits.

FPGA is a semiconductor device that can be configured by the customer or designer after manufacturing. FPAGAs are programmed by using a logic circuit diagram or a source code in a hardware description language to specify how the chip will work. They can be used to implement any logical function that an application specific integrated circuit can perform.

This paper has implemented the encryption and decryption of 1024-bit RSA by Montgomery algorithm and enhanced the rate of running.

II. RSA ALGORITHM

In number theory, Euler’s theorem (also known as the Fermat-Euler theorem or Euler’s totient theorem) states that if \( n \) is a positive integer and \( \alpha \) is a positive integer coprime to \( n \), then \( \alpha^{\phi(n)} \equiv 1 \pmod{n} \). The theorem is a generalization of Fermat’s little theorem, and is further generalized by Carmichael’s theorem. Euler’s theorem also forms the basis of the RSA encryption system: encryption and decryption in this system together amount to exponentiating the original text by \( \phi(n) \), so Euler’s theorem shows that the decrypted result is the same as the original.

The method of realizing RSA is as follows: Firstly, choose two big prime numbers which are not different. Secondly, calculate \( n = p \cdot q \) (publicly), \( \phi(n) = (p-1)(q-1) \), \( \phi(n) \) is the Euler function value of \( n \). Thirdly, choose \( e \) which is integer and \( \text{gcd}(e, \phi(n)) = 1 \). At last, \( d = e^{-1} \pmod{\phi(n)} \). The encryption/decryption chain of RSA is described as follows: the algorithm of encryption, \( c = E(m) = m^e \pmod{n} \); the algorithm of decryption, \( m = D(c) = c^d \pmod{n} \); In the algorithm, \( m \) is the original message data, \( c \) is the encrypted message or cipher text, \( e, m \) is the publicly available encryption key, and \( d \) is the decryption key. The following is a practical example of RSA key generation and an RSA-based encryption cryptographic exchange. For example:

1) Generator primes create the modulus: \( p = 47, q = 71 \), \( m = p \cdot q, m = 3337 \).
2) Calculate the key: \( 47 - 1 = 46, 71 - 1 = 70 \), \( \text{gcd}(46, 70) = 1, e = 79 \).
3) Key calculation: \( (d \cdot 79) \pmod{3220} = 1, d = \text{Egcd}(79, 3220) = 1019 \).
4) Message block to the value of 688, encrypted using the public key: \( m = 688, c = 688^{79} \pmod{3337 = 1570} \).
5) Cipher text decrypted with the private key to obtain the original data block: \( c = 1570, m = 1570^{1019} \pmod{3337 = 688} \).
Beyond this asymmetric cryptography example, RSA Math has practical applications in symmetric cryptography.

III. RSA IMPLEMENTATION METHOD BASED ON FPGA

A. The Implementation of Modular Exponentiation

Modular exponentiation is a type of exponentiation performed over a modulus. It is particularly useful in computer science, especially in the field of cryptography. For the encryption and decryption of RSA, Modular exponentiation is same.

\[ X^e \mod M, \text{ where } e = (e_1, ..., e_k); k \text{ is the number of bits of } e. \]

\[ X^e = X \times X^{2^0} \times (X^{2^1})^e \times \ldots \times (X^{2^{k-1}})^{e_{k-1}} \]

Input: X, e, M;

Output: Z = X^e \mod M;

if \( e_{k-1} = 1 \) then Z=X \else Z=1;

for i = k-2 downto 0 begin
  Z = Z \times Z \mod M;
  if e[i]=1 then Z = Z \times X \mod M
end

return Z

It is required that the exponent e be converted to binary notation. That is, e can be written as:

\[ e = \sum_{i=0}^{n-1} a_i \times 2^i \]

In such notation, the length of e is n bits. \( a_i \) can take the value 0 or 1 for any i such that 0<i<n-1.

B. The Implementation of Modular Multiplication

Montgomery introduced an efficient algorithm for calculating modular multiplication. The speed of modular multiplication affects the speed of RSA encryption algorithm and modular multiplication is the most important factor which affects the speed. We define multiplication mode for R=A*B mod n. There are many algorithms, such as Blakley algorithm, Barrett algorithm, Brickell algorithm, Montgomery algorithm. The Montgomery algorithm is the most suitable algorithm for hardware implementation, the widely used method is the Montgomery algorithm based binary.

To calculate the product of A*B mod M, where:

\[ A = \sum_{i=0}^{n-1} a_i \times 2^i \]

\[ B = \sum_{i=0}^{n-1} b_i \times 2^i \]

\[ M = \sum_{i=0}^{n-1} m_i \times 2^i \]

\( a_i, b_i, m_i \in \{0, 1\} \)

Input: A, B, M

Output: R = AB \mod M = MM(A,B)

\( R_0 = 0 \)

For \( i = 0 \) to \( n \) do

begin
  \( q_i = (R_i + a_i \times b_0) \mod 2; \)
  \( R_{i+1} = (R_i + a_i \times B + q_i \times N) / 2 \)

\]
IV. SYSTEM SIMULATION AND VERIFICATION

This design by the method of top-down is divided to several modules, such as, control module, modular multiplication module, memory module and so on. The sub-modules will be simulated first, and then all modules will be simulated together, till the function of entire circuit is right.

A. The simulation result of RSA

The simulation result of RSA is depicted in Fig. 2.

The simulation which does not consider the delay of the circuit mainly verifies the function of the circuit whether to meet the design requirements.

This design mainly uses Model Corporation’s ModelSim SE 6.0 to carry on the function simulation. This simulator supports Verilog, VHDL and the mixed-simulation. The simulator can take the simulation step by step and view the current value of any variables. You can view the Dataflow window of a module of input and output of continuous changes. The ModelSim is one of the widespread used simulators at present.

The test data used in the simulation is as follows:

- Modulus: 1024'h0ac66f597f338ca1;
- Plaintext: 1024'h0072418cccccccc3;
- Key: 1024'h0000000000000007;
- Constant A: 1024'h0a16c6d0ac51a104;
- Constant C: 1024'h0985c1bc96ceba58;
- Ciphertext: 1024'h052dc2c78533d116.

The result is right and is consistent with the encryption function.

B. The High-level synthesis

High-level synthesis, sometimes referred to as electronic system level synthesis, algorithmic synthesis, or behavioral synthesis, is an automated design process that interprets an algorithmic description of a desired behavior and creates hardware that implements that behavior. The starting point of a high-level synthesis flow is System C code. The code is analyzed, architecturally constrained, and scheduled to create a register transfer level hardware design language, which is then in turn commonly synthesized to the gate level by the use of a logic synthesis tool. The goal of HLS is to let hardware designers efficiently build and verify hardware, by giving them better control over optimization of their design architecture, and through the nature of allowing the designer to describe the design at a higher level of tools while the tool does the RTL implementation. Verification of the RTL is an important part of the process. Hardware design can be created at a variety of levels of abstraction. The commonly used levels of abstraction are gate level, register transfer level, and algorithm level.

The synthesis diagram of the RSA is depicted in Fig. 3.

In the diagram, the modules which color is yellow are the all modules in RSA. We can see the control module, memory module, modular multiplication module, and so on. We can also see the pins of the RSA.

C. The synthesis report of RSA

We can receive the circuit parameters from the synthesis report of the RSA. The report can give us the total resources that RSA circuit takes. The report is depicted in Fig. 4.
ACKNOWLEDGMENT

Thank the teachers and schoolmates in the laboratory. They give me much help in the study.

REFERENCES