國立交通大學

電子工程學系 電子研究所碩士班

碩 士 論 文

可重組化 RSA 密碼系統之設計與實作

Design and Implementation of Reconfigurable RSA Cryptosystems

研究生:陳允律

指導教授:張錫嘉

中華民國九十五年七月

國立交通大學

論文口試委員會審定書

本 校 電子工程學系 電子研究所 陳允律 君

所提論文：可重組化RSA密碼系統之設計與實作

合於碩士資格標準，業經本委員會評審認可。

口試委員：張錫嘉　　劉志尉

蔡維昌

指導教授：張錫嘉

所　　長：陳紹基

系 主 任：李鎮宜

中 華 民 國 95 年 7 月 17 日

可重組化 RSA 密碼系統之設計與實作

Design and Implementation of Reconfigurable RSA Cryptosystems

學生：陳允律 Student : Yun-Lu Chen

指導教授：張錫嘉 Advisor : Hsie-Chia Chang

國立交通大學

電子工程學系 電子研究所碩士班

碩士論文

A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics

College of Electrical and Computer Engineering

National Chiao Tung University

In Partial Fulfillment of the Requirements

for the Degree of Master

In

Electronics Engineering

July 2006

Hsinchu, Taiwan, R.O.C.

中華民國九十五年七月

可重組化 RSA 密碼系統之設計與實作

學生：陳允律 **指導教授：張錫嘉**

國立交通大學

電子工程學系 電子研究所碩士班

摘 要

本論文提出了在 RSA **密碼系統上可重組化的實作方法。**RSA **在許多安全性的應用上常被使用，像是智慧型卡片。而一造安全性的層級不同，**RSA **模數的區塊大小也不盡相同。在大多數的智慧型卡片的應用上，**RSA **密碼系統通常被設計在** 512/1024/2048/4096 **位元模數區塊大小的加解密下工作。為了能工作在大多數的智慧型卡片的應用上，本篇論文中的演算法是在可重組化和縮減面積下由蒙哥馬利模數乘法演算法改進而來，為了縮減面積，在架構上也利用五個記憶體區塊來減小龐大的暫存器數量。在這樣的設計之下可利用** Xilinx Vertex2 XC2V8000 **的五個記憶體區塊和** 6783 **個** slices **來達到** 512 **模數下** 99kb/s，**1024 模數下** 26kb/s，**2048 模數下** 6.8kb/s，**4096 模數下** 1.7kb/s。

Design and Implementation of Reconfigurable RSA Cryptosystems

Student: Yun-Lu Chen Advisor: Hsie-Chia Chang

Department of Electronics Engineering & Institute of Electronics
National Chiao Tung University

ABSTRACT

This thesis introduces a reconfigurable approach to the hardware implementation of the RSA cryptosystem. RSA is a well used algorithm in many security applications, such as smart card. The modulus block sizes are different for different level of security. For the majority applications of smart card, the RSA cryptosystem would be designed to work with 512/1024/2048/4096-bit RSA encryption/decryption. The algorithm of this thesis is modified by the Montgomery modular multiplication algorithm to be reconfigurable and reduce area. In order to reduce area, there are five memory blocks to substitute the large amount of registers. As such the design can used five memory blocks and 6783 slices to achieve the baud rate of 99kb/s for 512-bit modulus, 29kb/s for 1024-bit modulus, 6.8kb/s for 2048-bit modulus and 1.7kb/s for 4096-bit with Xilinx Vertex2 XC2V8000.

誌　　謝

要完成一本論文，除了作者本身的努力與付出，旁人的幫助也是不可或缺的。 在研究生的學涯裡我的指導教授張錫嘉博士無論是在研究方面、工作方面，亦或是生活方面，無不給予作者極大的幫助，研究方面像嚴師，工作方面像前輩，生活方面像朋友，這就是老師給我的感覺，真的是非常感謝老師。再來要感謝的就是林建青學長，每當我遇到問題時，學長總是不厭其煩的幫我解答，讓我在研究的路上，能更加的順暢。

此外，當然也要感謝我的家人，無論是在精神上或是物質上給予的支持，使我可以無後顧之憂完成學業，最後也要感謝 OASIS 和 SUN 實驗室的同學、學弟們，沒有你們，這段時光不會如此值得回憶，僅以此篇論文與你們共同分享這份喜悅。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As time goes on, the internet has become as popular as possible. The communication between people and people is originally need face-to-face, but can carry on the remote communication with some tools now. The convenient means of communication are depended on the prosperity of the internet. People depend on the internet sending message that may be secret or important. If the hacker gets the message which has no protector, the secret is not secret. Informant security is more and more important in the internet age.

Cryptography [1] is a knowledge to research how to avoid that the hacker gets the real message. As the increase of the computer performance, there are much difference between conventional and modern cryptography. For the general information communication model that is shown in Fig. 1.1, there are the sender, receiver and hacker. Before beginning, we define some terms. The original intelligible message that the sender sends is known as the plaintext. The coded message that has scrambled by cryptosystem is called the ciphertext. Reversibility is the fundamental need for the cryptosystem. Generally speaking, the cryptosystem supply following functions to depend on the application.

Confidentiality : It is to avoid that the hacker gets plaintext.

Authentication : It is to ensure that the receiver can verify the source of message.

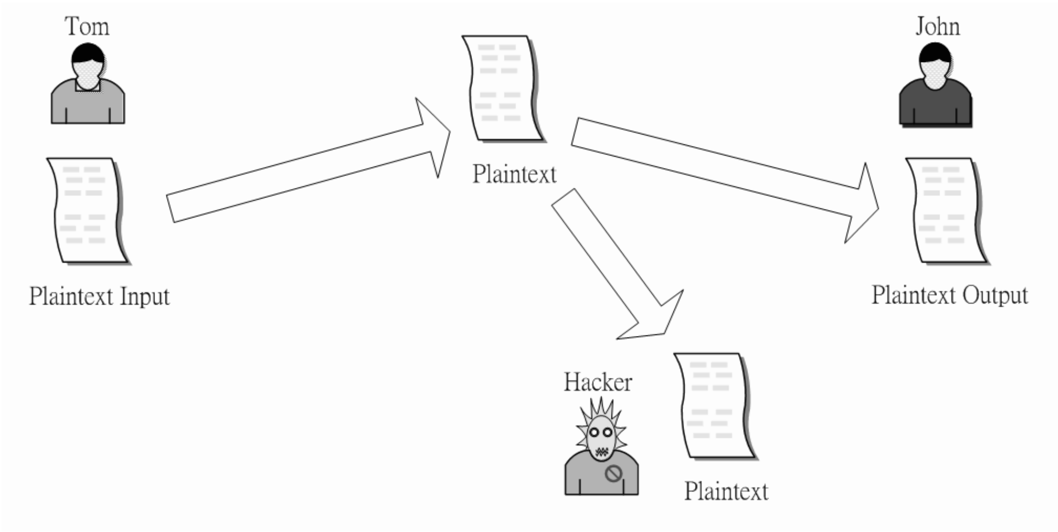

Figure 1.1: Information Communication Model

.

Integrity : It is to know that the message has been modified in transmission or not.

Nonrepudiation : It can prevent either sender or receiver from denying a transmitted message.

There are two kinds of commonly used encryption/decryption ways:

1. Conventional: Secret-key encryption/decryption

2. Modern: Public-key encryption/decryption

The conventional cryptography usually focus on the confidentiality of the information, but the modern cryptography also considers that the authentication, integrity and nonrepudiation of the information are more important for the application of the commerce.

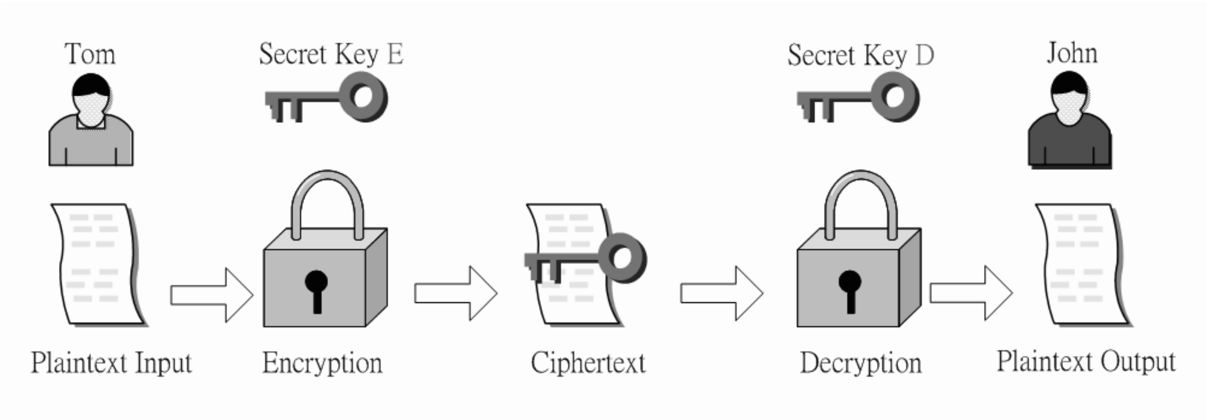

Figure 1.2: Secret-key Cryptosystem Model

.

1.1 Secret-key Cryptosystem

In conventional cryptography, there is a key pair E,D. Usually, the key E and the key D are the same, or it is easy to find one if another is known. A secret-key cryptosystem model is shown in Fig. 1.2. The sender uses the key E to encrypt the plaintext, and the receiver uses the key D to decrypt the ciphertext. This method is known as secret-key or symmetric cryptosystem. All secret-key cryptosystems are based on substitution and transposition. Substitution is mean mapping from one field to anther and transposition is mean replace the element of the message. Because of that the hardware of substitution and transposition is very easy, secret-key cryptosystem is much faster than public-key cryptosystem in general. There are many algorithms proposed for the secret-key cryptosystem, such as DES(Data Encryption Standard) and AES(Advanced Encryption Standard).

A secret-key cryptosystem has several drawback listed below:

- How to manage the keys when the number of key pairs is large?
  - If there are n subscribers in a network, then everyone have to hold $n-1$ keys to communicate with others. For example, If there are 1000 sub-

scribers in a network, then the number of key pairs is 499×500. It is too hard to manage such many key pairs.

- How to get the key pair between the sender and receiver?
  - A secure channel is needed between the sender and receiver who have never meet before, but how to get this channel? This problem is called the key distribution problem.
- How to ensure the Authentication and Nonrepudiation?
  - In symmetric cryptosystem, the sender and receiver both have the same key for encryption and decryption. Thus the sender may deny sending the message that has sent before because it is impossible to ensure that the message is not make by the receiver.

1.2 Public-key Cryptosystem

The concept of public-key cryptography evolved from an attempt to solve the problems of symmetric encryption. Public-key algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic.

- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

In addition, some algorithms also exhibit the following characteristic.

- Either of the two related keys can be used for encryption, with the other used for decryption.

The equations can be written as follow:

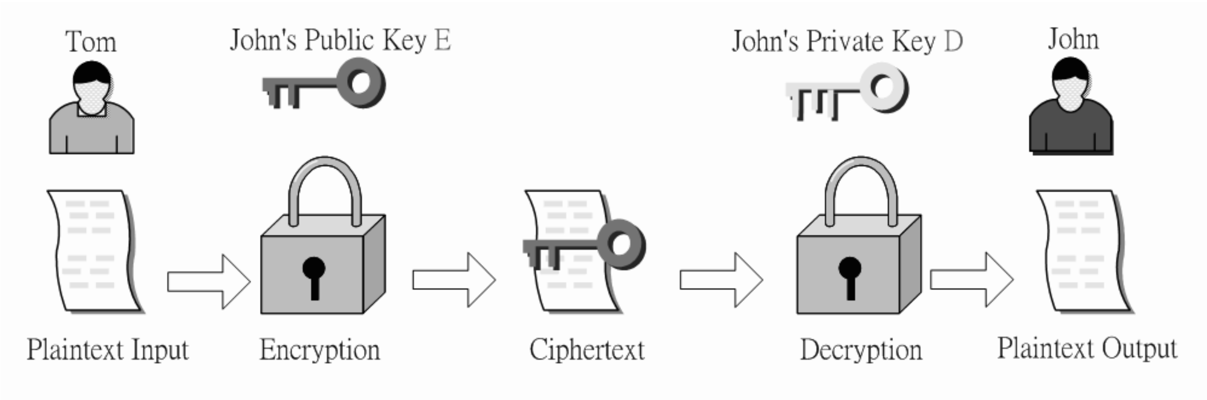

Figure 1.3: Public-key Cryptosystem Model

.

$$C = EC(E_J, M)$$

$$M = DC(D_J, C)$$

A secure communication using a public-key cryptosystem is shown in Fig. 1.3, and the essential steps are the following:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.

2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. Each user maintains a collection of public keys obtained from others.

3. If Tom wishes to send a confidential message to John, Tom encrypts the message using John's public key.

4. When John receives the message, he decrypts it using his private key. No other recipient can decrypt the message because only John knows John's private key.

In this approach, all participants can access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As

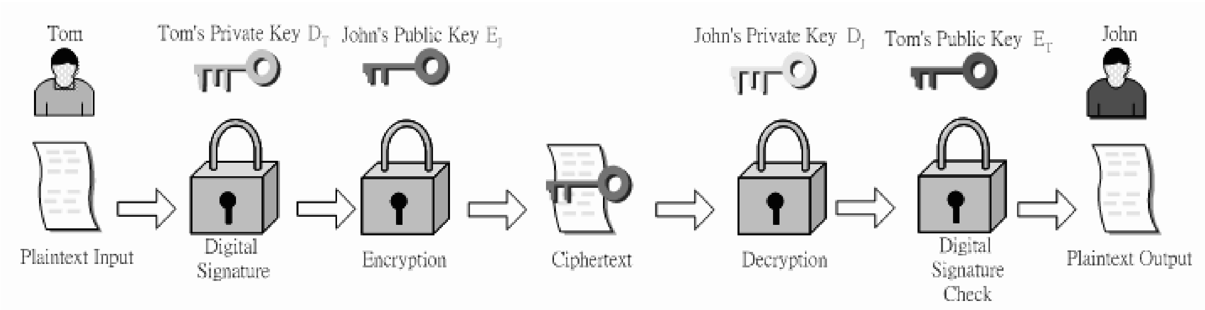

Figure 1.4: Public-key Cryptosystem Model with Digital Signature

.

long as a system controls its private key, its incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

It is important to emphasize that the encryption process just described does not provide confidentiality. That is, the message begin sent is safe from alteration but not from eavesdropping.

The public-key cryptosystem model with digital signature is shown in Fig. 1.4. It is impossible to provide both the authentication function and confidentiality by a double use of the public-key scheme.

$$S = EC(D_T, M)$$
$$C = EC(E_J, S)$$
$$S = DC(D_J, C)$$
$$M = DC(E_T, S)$$

In this case, we begin as before by encrypting a message, using the sender's private key. This provides the digital signature. Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, confidentiality is provided. The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

Depending on the application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function. In broad terms, we can classify the use of public-key cryptosystems into three categories:

- Encryption/decryption : The sender encrypts a message with the recipient's public key.
- Digital signature : The sender signs a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- Key exchange : Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key of one or both parties.

Chapter 2

RSA Cryptosystem

2.1 Mathematics Foundation

2.1.1 Number Theory

In modern cryptosystem, the encryption processes are mathematical operations that turn the input numerical values into output numerical values. The must important mathematical tool is number theory, especially the theory of congruences.

Congruences

One of the most basic and useful in number theory is modular arithmetic, or congruences. Let a, b, n be integers with $n \neq 0$. If a and b differ by a multiple of n, a is congruent to b mod n.

$$a \equiv b \pmod{n}$$

It can be rewritten as

$$a \equiv b + nk$$

for some integer k

Primitive Roots

In general, when p is a prime, a primitive root mod p is a number whose powers yield every nonzero class mod p. There are $\phi(p-1)$ primitive roots mod p. Let g be a primitive root for the prime p.

- If i is an integer, then $g^i \equiv 1$ (mod p) if and only if $i \equiv 0$ (mod p-1).
- If j and k are integers, then $g^j \equiv g^k$ (mod p) if and only if $j \equiv k$ (mod p-1).

Fermat's Theorem

Fermat's theorem states the follows:If p is prime and a is a positive integer not divisible by p, then

$$a^{p-1} \equiv 1 \ mod \ p \tag{2.1}$$

We know that if all of the elements of Z_p, where Z_p is the set of integers $\{0, 1, ..., p-1\}$, are multiplied by a, modulo p, the result consists of all of the elements of Z_p in some sequence. Furthermore, $a \times 0 \equiv 0$ mod p. Therefore, the $(p-1)$ numbers $\{a$ mod p, $2a$ mod p,...,$(p-1)a$ mod $p\}$ are just the numbers $\{1, 2, ..., (p-1)\}$ in some order. Multiplying the numbers in both sets and taking the result mod p yields

$$\begin{aligned} 1 \times 2 \times ... \times (p-1) &\equiv (a \ mod \ p) \times (2a \ mod \ p) \times ... \times ((p-1)a \ mod \ p) \\ (p-1)! \ mod \ p &\equiv (p-1)!a^{p-1} \end{aligned}$$

We can cancel the $(p-1)!$ term because it is relatively prime to p. This yields Equation 2.1.

Euler's Totient Function

Before presenting Euler's theorem, we need to introduce an important quantity in number theory, referred to as Euler's totient function and written $\phi(n)$, where

$\phi(n)$ is the number of positive integers less than n and relatively prime to n. It should be clear that for a prime number p,

$$\phi(p) = p - 1$$

There are two prime numbers p and q, with $p \neq q$. Then, for $n = pq$,

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p-1) \times (q-1) \tag{2.2}$$

Euler's Theorem

Euler's theorem states that for every a and n that are relatively prime:

$$a^{\phi(n)} \equiv 1 \ mod \ n \tag{2.3}$$

Equation 2.3 is true if n is prime, because in that case $\phi(n) = (n-1)$ and Fermat's theorem holds. However, it also holds for any integer n. Recall that $\phi(n)$ is the number of positive integers less than n that are relatively prime to n. Consider the set of such integers, labeled as follows:

$$R = \{x_1, x_2, ..., x_{\phi(n)}\}$$

Now multiply each element by a, modulo n:

$$S = \{(ax_1 \ mod \ n), (ax_2 \ mod \ n), ..., (ax_{\phi(n)} \ mod \ n)\}$$

The set S is a permutation of R, by the following line of reasoning:

1. Because a and x_i are relatively prime to n, ax_i must also be relatively prime to n. Thus, all the elements of S are integers less than n that are relatively prime to n.

2. There are no duplicates in S. If $ax_i \ mod \ n = ax_j \ mod \ n$, then $x_i = x_j$.

Therefore,

$$\begin{aligned}
\prod_{i=1}^{\phi(n)} (ax_i \ mod \ n) &= \prod_{i=1}^{\phi(n)} x_i \\
\prod_{i=1}^{\phi(n)} ax_i &\equiv \prod_{i=1}^{\phi(n)} x_i(\ mod \ n) \\
a^{\phi(n)} \times \prod_{i=1}^{\phi(n)} x_i &\equiv \prod_{i=1}^{\phi(n)} x_i(\ mod \ n) \\
a^{\phi(n)} &\equiv 1 \ mod \ n
\end{aligned}$$

An alternative form of the theorem is also useful:

$$a^{k\phi(n)+1} \equiv a \ mod \ n \tag{2.4}$$

2.1.2 Montgomery's Method

In 1985, P. L. Montgomery introduced an efficient algorithm for computing $R = a \cdot b \ mod \ n$ where a, b, and n are k-bit binary numbers. The algorithm is particularly suitable for implementation on general-purpose computers which are capable of performing fast arithmetic modulo a power of 2. The Montgomery reduction algorithm computes the resulting k-bit number R without performing a division by the modulus n. Via an ingenious representation of the residue class modulo n, this algorithm replaces division by n operation with division by a power of 2. This operation is easily accomplished on a computer since the numbers are represented in binary form. Assuming the modulus n is a k-bit number, i.e., $2^{k-1} < n < 2^k$, let r be 2^k. The Montgomery reduction algorithm requires that r and n be relatively prime, i.e., $\gcd(r, n) = \gcd(2^k, n) = 1$. This requirement is satisfied if n is odd. The basic idea of the Montgomery reduction algorithm was showed as following.

Given an integer $a < n$, we define it's n-residue with respect to r as

$$\overline{a} \equiv a \cdot r \ mod \ n$$

It is straightforward to show that the set

$$\{i \cdot r \ mod \ n | 0 \leq i \leq n-1\}$$

is a complete residue system, i.e., it contains all numbers between 0 and $n-1$. Thus, there is a one-to-one correspondence between the numbers in the range 0 and $n-1$ and the numbers in the above set. The Montgomery reduction algorithm exploits this property by introducing a much faster multiplication routine which computes the n-residue of the product of the two integers whose n-residues are given. Given two n-residues $\overline{a}$ and $\overline{b}$, the Montgomery product is defined as the n-residue

$$\overline{R} \equiv \overline{a} \cdot \overline{b} \cdot r^{-1} \ mod \ n$$

where r^{-1} is the inverse of r modulo n, i.e., it is the number with the property

$$r^{-1} \cdot r \equiv 1 \ mod \ n.$$

The resulting number $\overline{R}$ is indeed the n-residue of the product

$$R \equiv a \cdot b \ mod \ n$$

since

$$\begin{aligned} \overline{R} &\equiv \overline{a} \cdot \overline{b} \cdot r^{-1} \ mod \ n \\ &\equiv a \cdot r \cdot b \cdot r \cdot r^{-1} \ mod \ n \\ &\equiv a \cdot b \cdot r \ mod \ n \end{aligned}$$

2.2 RSA Algorithm

The pioneering paper by Diffie and Hellman introduced a new approach to cryptography and, in effect, challenged cryptologists to come up with a cryptography algorithm that met the requirements for public-key systems. One of the first of the responses to the challenge was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published [2] in 1978. The Rivesr-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.

2.2.1 RSA Scheme

The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and $n-1$ for some n which is usually between 512 and 4096. The flow of RSA is showed as following:

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate N and $\phi(N)$	$N = p \times q, \phi(N) = (p-1)(q-1)$
Select integer E	$\gcd(\phi(N), E) = 1; 1 < E < \phi(N)$
Calculate D	$D \equiv E^{-1}\ mod\ \phi(N)$
Public key	$KU = \{E, N\}$
Private key	$KR = \{D, N\}$

Encryption

Plaintext M	$M < N$
Ciphertext C	$C = M^E\ mod\ N$

Decryption

Ciphertext C $\quad C$

Plaintext M $\quad M = C^D \ mod \ N = M^{DE} \ mod \ N = M \ mod \ N$

Let p and q be two distinct large random primes. The modulus N is the product of these two primes: $N = pq$. Euler's totient function of N is given by

$$\phi(N) = (p-1)(q-1)$$

Now, select a number $1 < E < \phi(N)$ such that

$$\gcd(\phi(N), E) = 1,$$

and compute D with

$$D \equiv E^{-1} \ mod \ \phi(N).$$

Here, $\{E, N\}$ is the public key and $\{D, N\}$ is the private key. The value of D and the prime numbers p and q are kept secret. Encryption is performed by computing

$$C = M^E \ mod \ N,$$

where M is the plaintext such that $0 \leq M < N$. The number C is the ciphertext from which the plaintext M can be computed using

$$M = C^D \ mod \ N.$$

The correctness of the RSA algorithm follow from Euler's theorem: Let N and a be positive, relatively prime integers. Then

$$a^{\phi(N)} \equiv 1 \ mod \ N.$$

Since ED is equal to $1 \ mod \ \phi(N)$, It meet that ED is equal to $1 + k\phi(N)$ for some integer k.

$$
\begin{aligned}
C^D &\equiv (M^E)^D \ mod \ N \\
&\equiv M^{ED} \ mod \ N \\
&\equiv M^{1+k\phi(N)} \ mod \ N \\
&\equiv M \times (M^{\phi(N)})^k \ mod \ N \\
&\equiv M \times 1 \ mod \ N
\end{aligned}
$$

2.2.2 Modular Exponentiation

In RSA cryptosystem, the modular exponentiation is the basic operation for encryption, decryption or signing. The simple and direct way to compute $M^E \mod N$ is to multiply M sequentially for E times. Since all the operands in RSA operation (M, N, E, D) are typically large than 512 bits and it is too hard to store the result that was computed M^E. It is need to find some efficient methods. There are two common algorithms which can be used: the L-R binary method and the R-L binary method.

L-R Algorithm

$$
\begin{aligned}
M^E \ mod \ N &\equiv M^{(e_{n-1} \times 2^{n-1} + \ldots + e_1 \times 2^1 + e_0 \times 2^0)} \ mod \ N \\
&\equiv (M^{e_0} \times (M^{e_1} \times (\ldots \times (M^{e_{n-2}} \times (M^{e_{n-1}} \ mod \ N)^2 \ mod \ N)^2 \ldots)^2 \ mod \ N)^2 \ ) mod \ N
\end{aligned}
$$

In the L-R algorithm [3] , the square and performed sequentially. It does mean that both the square and multiply operations can be performed in the same single hardware multiplier, thus saving on area.

$$LREM\{M, E\}$$

$$\{$$

$$S = 1$$

$$for\ (i = n - 1\ downto\ 0)$$

$$\{$$

$$S = S^2 mod N$$

$$S = S \times M^{e_i} mod N$$

$$\}$$

$$return\ S$$

$$\}$$

R-L Algorithm

$$\begin{aligned} M^E\ mod\ N &\equiv M^{(e_{n-1}\times 2^{n-1}+...+e_1\times 2^1+e_0\times 2^0)}\ mod\ N \\ &\equiv M^{e_{n-1}\times 2^{n-1}} \times (M^{e_{n-2}\times 2^{n-2}} \times (... \times (M^{e_1\times 2^1} \times (M^{e_0\times 2^0}\ mod\ N)...)\ mod\ N)\ mod\ N \end{aligned}$$

In the R-L algorithm, the square and multiply operations are independent, and may be performed in parallel. Thus, 50% less clock cycles are required to complete the exponentiation. However, two physical hardware multipliers are required to achieve this speed up.

$RLEM\{M, E\}$

$\{$

$S = 1, P = M$

$for\ (i = 0\ to\ n - 1)$

$\{$

$S = S \times P^{e_i} modN, P = P^2 modN$

$\}$

$return\ S$

$\}$

2.2.3 Modular Multiplication

Original Montgomery Multiplication

Montgomery's Algorithm [4] computes the modular multiplication without trial division. It turns the modular multiplication into iterations of n-bit addition and shifting and reduces the complexity of modular multiplication to constant time operations. This is the key point why Montgomery's algorithm is so popular in hardware implementation. However, the S, B, and N are n-bit integers, each iteration of the above procedure needs to accumulate three n-bit integers and divide the result by 2.

Given $A = (a_{n-1}, a_{n-2}, ..., a_1, a_0)$ and $B = (b_{n-1}, b_{n-2}, ..., b_1, b_0)$ are two n-bit integers, and N is a n-bit prime, where $0 \leq A, B < N$.

$MM\{A, B, N\}$

$\{$

$S = 0$

$for\ (i = 0\ to\ n - 1)$

$\{$

$$t_0 = (S + a_i B)\ mod\ 2$$
$$S = \frac{(S + a_i B + t_0 N)}{2}$$

$\}$

$return\ S$

$\}$

There are two n-bit addition in each iteration. The result of above algorithm is in the range $0 \leq S < N + B$, not in the correct range $0 \leq S < N$. It is needed to subtract S by N and the result can be expressed as

$$S\ or\ S - N \equiv A \times B \times 2^{-n}\ mod\ N.$$

Chen's Modified Montgomery Multiplication

$CMMM\{A, B, N\}$

$\{$

$C = A \times B = c_{2n-1}2^{2n-1} + c_{2n-2}2^{2n-2} + ... + c_1 2^1 + c_0 2^0$

$S = 0$

$for\ (i = 0\ to\ 2n - 1)$

$\{$

$t_0 = (S + c_i)\ mod\ 2$

$S = \dfrac{(S + c_i + t_0 N)}{2}$

$\}$

$return\ S$

$\}$

In order to improve the disadvantage of original Montgomery's algorithm. Chen made a new consideration in 1996. The result of Montgomery's algorithm is in the range $0 \leq S < N + B$. However, if $B = 1$, the range becomes $0 \leq S < N + 1 \leq N$. S will be equal to N only when $A \times B$ is a multiple of N, which is impossible in RSA scheme. In order to achieve this, the multiplication, $A \times B$, can be computed before the loop. For hardware implementation with pipeline, modular operation can work without waiting the final result of $A \times B$. This means that multiplication and modulus can be work in parallel. The disadvantage of Chen's algorithm is the number of iteration that is two times than Montgomery's. However, there is only one n-bit addition in each iteration in Chen's. The result can be expressed as

$$S \equiv A \times B \times 2^{-n}\ mod\ N. \tag{2.5}$$

Yang's Modified Montgomery Multiplication

$YMMM\{A, B, N\}$

$\{$

$C = A \times B = c_{2n-1}2^{2n-1} + c_{2n-2}2^{2n-2} + ... + c_1 2^1 + c_0 2^0 = C_U \times 2^n + C_L;$

$(0 \leq C_U, C_L < 2^n, C_L = c_{n-1}2^{n-1} + c_{n-2}2^{n-2} + ... + c_1 2^1 + c_0 2^0$

$S = 0$

$for\ (i = 0\ to\ n-1)$

$\{$

$$t_0 = (S + c_i)\ mod\ 2$$
$$S = \frac{(S + c_i + t_0 N)}{2}$$

$\}$

$S = S + C_U$

$return\ S$

$\}$

In order to reduce the number of the iterations in Chen's algorithm, Yang [5] proposed a modified algorithm in 1998. This algorithm split the result of $A \times B$ into two parts.

$$\begin{aligned} A \times B \times 2^{-n}\ mod\ n &\equiv C \times 2^{-n}\ mod\ n \\ &\equiv (c_{2n-1}2^{2n-1} + c_{2n-2}2^{2n-2} + ... + c_1 2^1 + c_0 2^0) \times 2^{-n}\ mod\ n \\ &\equiv (C_U \times 2^n + C_L) \times 2^{-n}\ mod\ n \\ &\equiv (C_U + C_L \times 2^{-n})\ mod\ n \end{aligned}$$

C_U has the same weight as S, it means that C_U can be added after the loop. The result of Yang's algorithm is expressed as

$$S \equiv A \times B \times 2^{-n} \ mod \ N, 0 \leq S < 2^n + N < 2^{n+1}.$$

Yang's algorithm reduces half cycles than Chen's, and there is only one n-bit addition in each iteration in Yang's algorithm.

Proposed Modified Montgomery Multiplication

$PMMM\{A, B, N\}$

$\{$

$X = 2B, Y = 2B + N$

$S = 0$

$for \ (i = 0 \ to \ n)$

$\{$

$t_0 = S \ mod \ 2$

$case \ (t_0, a_i)$

$2'b00 : S = S;$

$2'b01 : S = S + X;$

$2'b10 : S = S + N;$

$2'b11 : S = S + Y;$

$endcase$

$$S = \frac{S}{2}$$

$\}$

$return \ S$

$\}$

The proposed algorithm is modified with word-base algorithm[6]. After computing $S + B$, it is need To determine that add N or not. If it turns B to $2B$, the result of

$S + 2B\ mod\ 2$ is equal to $S\ mod\ 2$. Equation. 2.5 can be modified as following:

$$
\begin{aligned}
S &\equiv A \times B \times 2^{-n}\ mod\ N \\
&\equiv A \times 2B \times 2^{-(n+1)}\ mod\ N
\end{aligned}
$$

It shows that the cost is adding one iteration.

The disadvantage of above algorithm is the same with original Montgomery's algorithm. The range of S is not between 0 and N. In order to make proposed algorithm working correct, there have some conditions to observe.

$$
Given\ prime\ N = n_{t-1}2^{t-1} + n_{t-2}2^{t-2} + ... + n_1 2^1 + n_0 2^0, n_{t-1} \neq 0;
$$

$$
and\ \ the\ \ loop\ \ input\ M = m_{k-1}2^{k-1} + m_{k-2}2^{k-2} + ... + k_1 2^1 + k_0 2^0;
$$

1. $M < 2N$

2. $N + 2M < 2^t + 2^{k+1} < 2^n$

3. $N + 2^{-(n-k)}2M < 2^t + 2^{-n-k)}2^{k+1} \leq 2^k$

$\Longrightarrow t = n - 2; k = n - 2;$

First, the input M of each iteration would be less and equal to two times N. Second, there only have n-bit addition. The last means that the output of each iteration cannot large than the maximal value of M of each iteration.

$PEM\{M, E, N\}$

$\{$

$\quad P = PMMM(M, 2^{2n}, N);$

$\quad S = 1;$

$\quad for\ (i = 0\ to\ n - 1)$

$\quad \{$

$\quad\quad if(e_i = 1)$

$\quad\quad\quad S = PMMM(P, S, N);$

$\quad\quad else$

$\quad\quad\quad S = S;$

$\quad\quad endif$

$\quad\quad P = PMMM(P, P, N)$

$\quad \}$

$\quad return\ S$

$\}$

Chapter 3

Proposed Reconfigurable RSA Architecture

3.1 Overview of the Proposed Architecture

In order for the RSA cryptosystem to be considered secure, the key sizes should be long enough. It is impossible that all data of the key transmit in parallel.

3.1.1 I/O Description

Figure. 3.1 show the input signals and output signals of the proposed architecture. The *clk*, *reset*, *enable* and *in* are input signals. The *busy*, *out_valid*, *out_enable* and *in* are output signals. The data bandwidth of the proposed architecture is defined as 16-bits and the total bandwidth of input signals and output signals is 38-bits. The I/O pins are described detail as following:

1. *clk*: This is the input pin of operation clock.

2. *reset*: This is in order to make initialization in the beginning of the cryptosystem work.

3. *enable*: This is the control signal to indicate that the data of the *in* signal is valid.

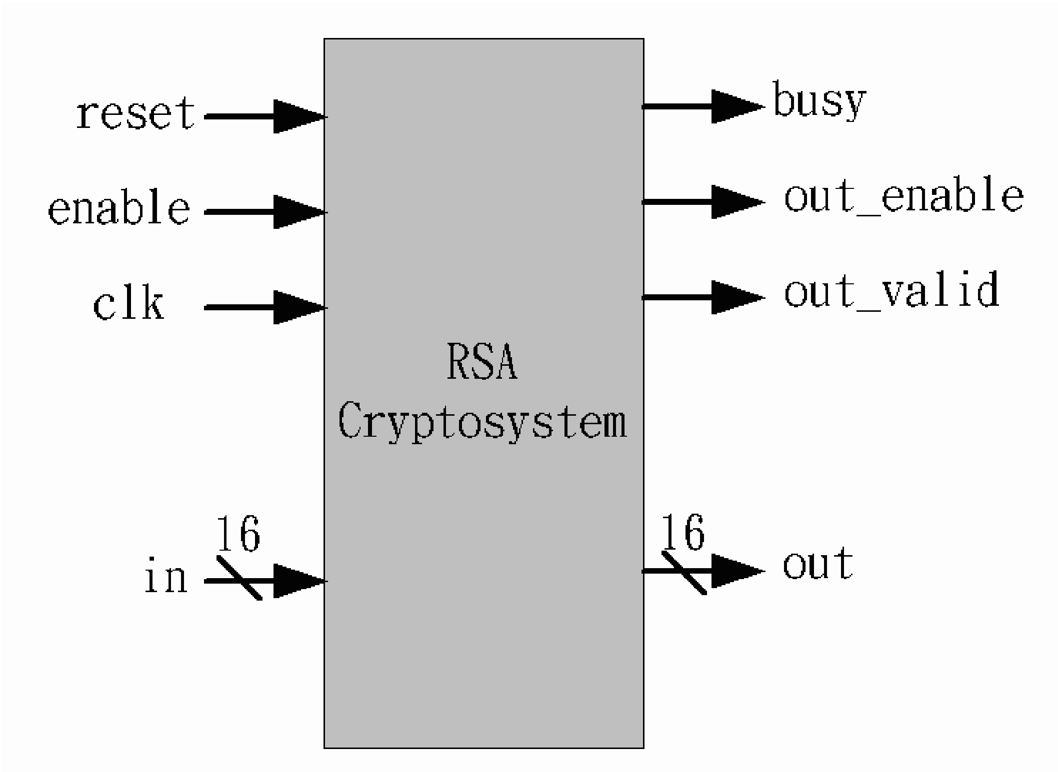

Figure 3.1: I/O pins of the proposed RSA cryptosystem

.

4. *in*: This is the data input. There may carry exponent E or D, modulus N, pre-compute coefficient $R^2 modN$ and operation mode type on the *in* signal.

5. *busy*: When the *busy* signal is high, it means the RSA cryptosystem is busy for another process.

6. *out_enable*: When the *out_enable* is high, it meets that the correct output data has been stored in the memory and ready for loading.

7. *out_valid*: When the *out_valid* is high, there is the correct result in the *out* signal.

8. *out*: The *out* shows the correct result of the encryption/decryption of RSA.

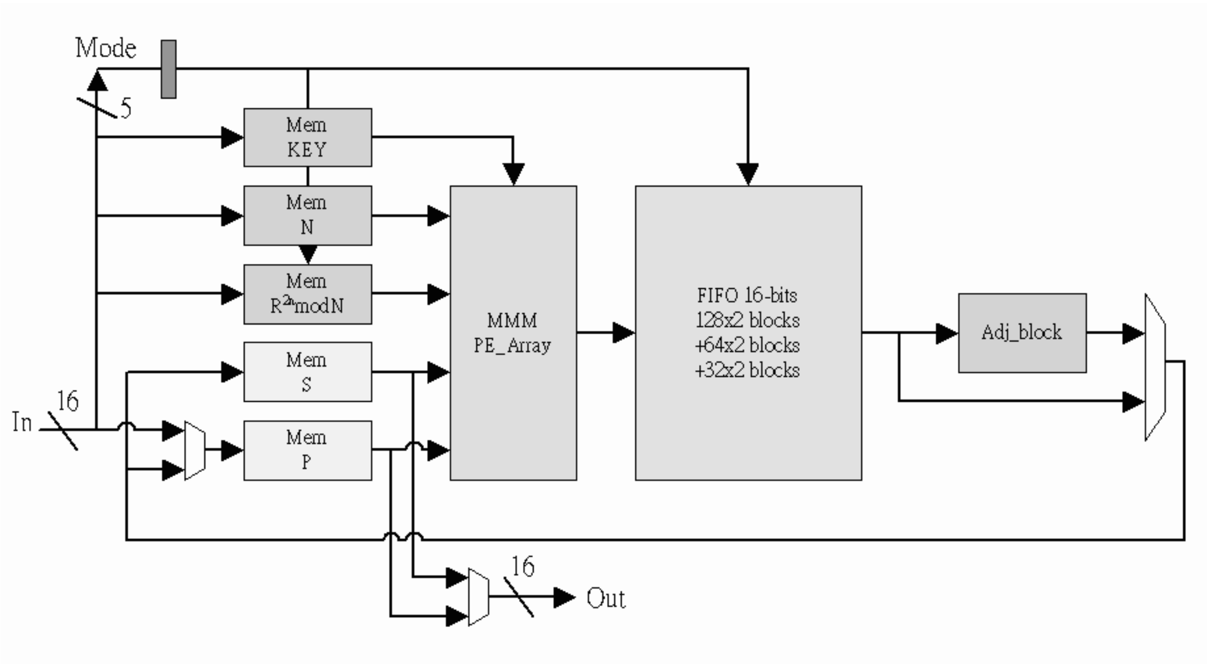

Figure 3.2: Block diagram of proposed RSA cryptosystem

3.1.2 Main Modules Description

In order for the less bandwidth, the storage of data and the redundant cycles are needed. There are five memory block, processing element array, adjustment unit and 4096-bits FIFO, illustrated in figure. 3.2. The main modules are described detail as following:

1. *Memory Block*:

   There are five memory block for using. The exponent E or D is stored in the "key" memory. The modulus N is stored in the "N" memory and the pre-compute coefficient $2^{2n} modN$ is stored in the "R^2 mod N" memory. The temporary values M^{2^i} and M^j are stored in the "S" and "P" memories.

2. *PE Array*:

   The processing element array is composed of 16 level processing elements. Each processing element has two-stage pipeline, as shown in the figure. 3.3.

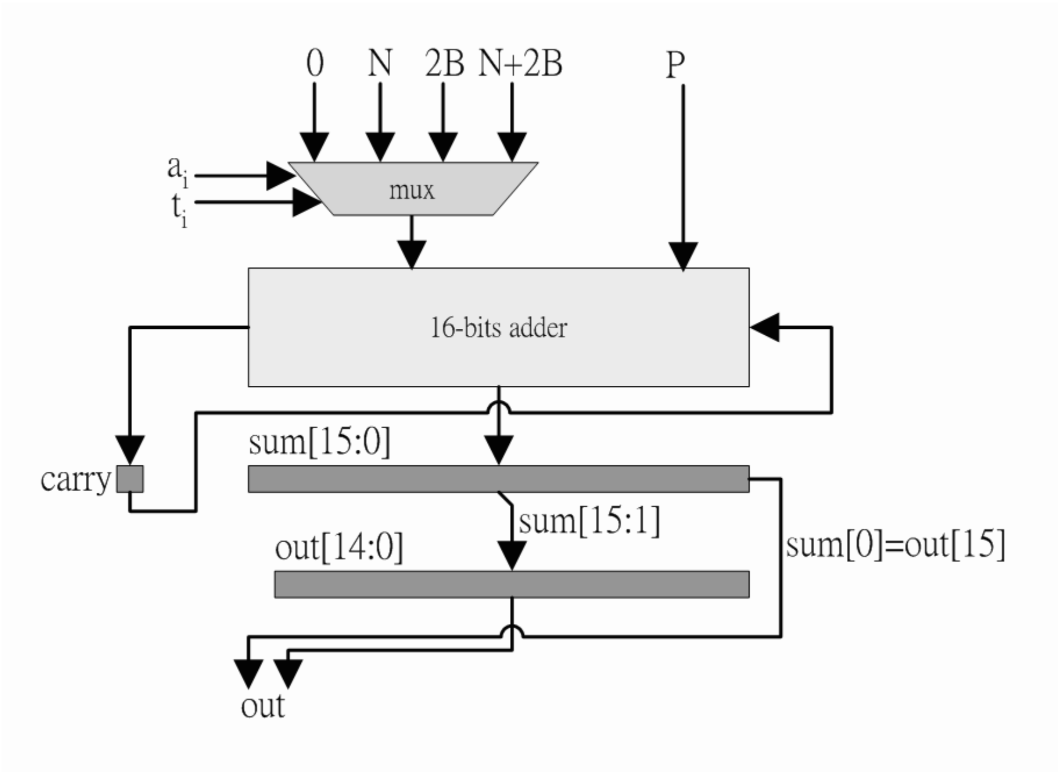

Figure 3.3: Processing Element used in figure. 3.4
.

There are 16-bits ripple adder and 5 × 16-bits registers in each processing element. It takes 32 clock cycles to pass the PE array.

3. $224 * 16$-*bits FIFO*:

   There are only 16-stages PE array. If the proposed RSA cryptosystem work with 4096-bits RSA encryption/decryption, it is not enough to store the temporary data. In order to make the proposed RSA cryptosystem work regular with 4096-bits RSA encryption/decryption, the registers are needed to store $4096 - 32 * 16 = 224 * 16$ date.

4. *Adjustment Adder*:

   In order to get the correct result, the output of the PE array would be subtracted by modulus N with adjustment adder in the finish of encryp-

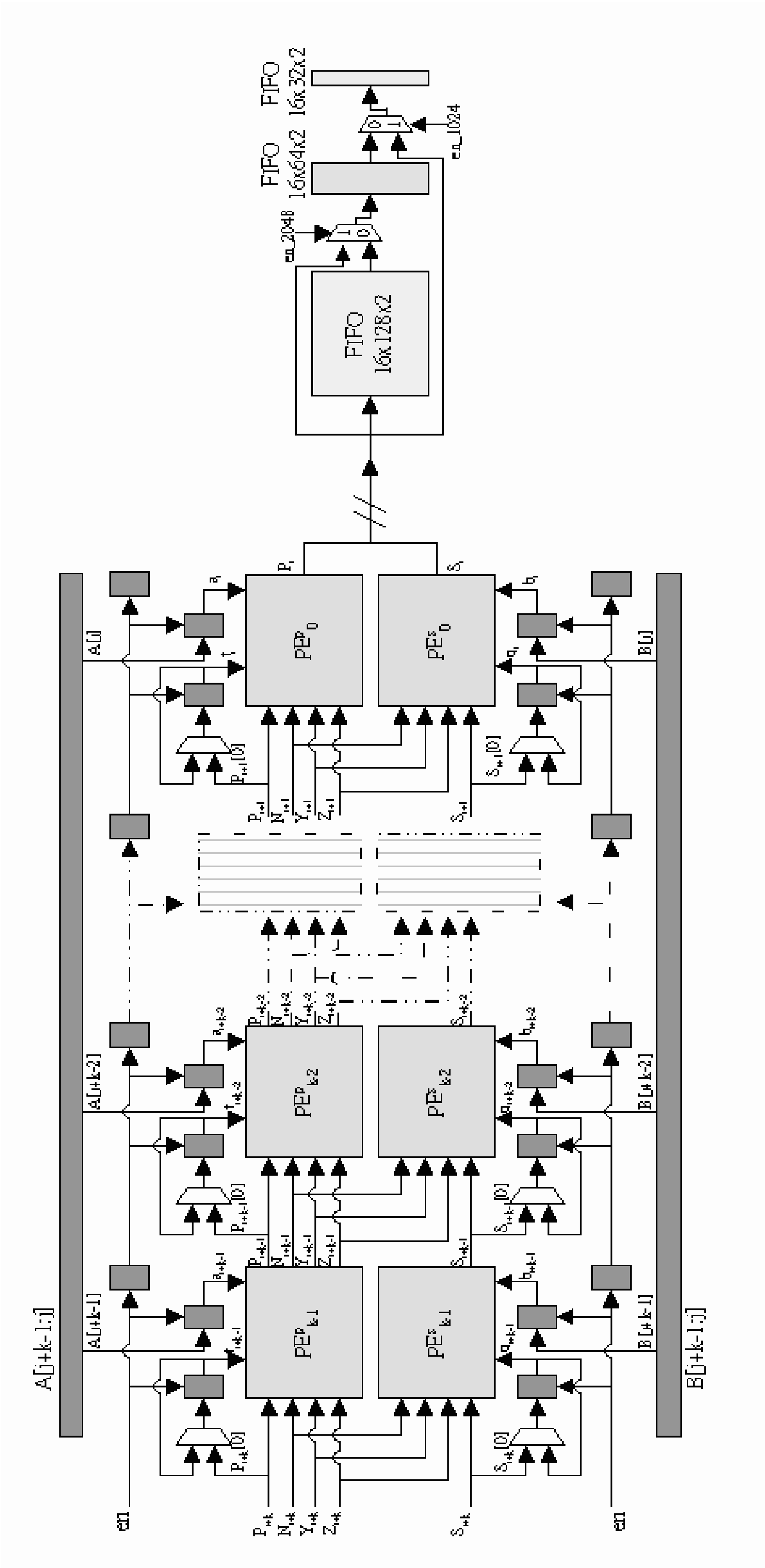

Figure 3.4: Processing Element Array and FIFO

tion/decryption.

5. *Complete Detector*:

   It can compute the key length k of E or D and decide the iteration number of Montgomery modular exponentiation.

3.2 Operation Mode Description

When the *enable* signal turns from low to high, as shown in figure. 3.5, there is the message of the *in* signal with reference to *mode type*. Table. 3.1 indicates that there have three type of operation and four variety of block length. Three type of operation, which are *encryption/decryption*, *configuration* and *result*, depend on the *in[5:4]* signal. *Encryption* is the action to convert plaintext to ciphertext, and *decryption* is the action to convert ciphertext to plaintext. *Configuration* is the action to store the key into the memory. *Result* is the action to transmit the data of the memory to the *out* signal.

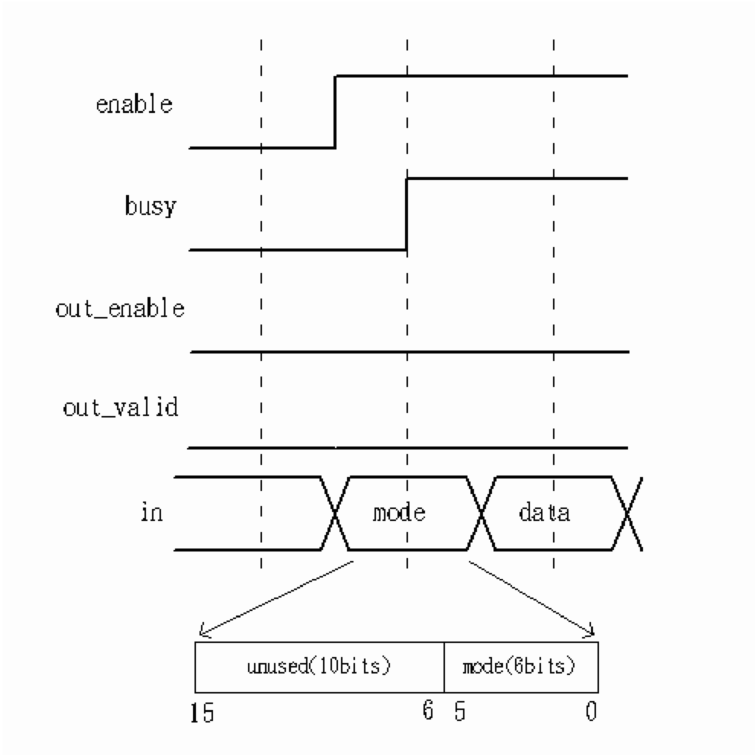

Figure 3.5: Load mode type

.

When the *out_enable* signal and *out_valid* signal are both idle and the *enable* signal is high, it means that there have no useful plaintext or ciphertext in the memory and it can work at the *Configuration* or *encryption/decryption* mode.

Table 3.1: Mode Type

	4096 bits (in[3:0]=4'b1000)	2048 bits (in[3:0]=4'b0100)	1024 bits (in[3:0]=4'b0010)	512 bits (in[3:0]=4'b0001)
En/Decryption (in[5:4]=2'b00)	$(001000)_2$	$(000100)_2$	$(000010)_2$	$(000001)_2$
Configuration (in[5:4]=2'b01)	$(011000)_2$	$(010100)_2$	$(010010)_2$	$(010001)_2$
Result (in[5:4]=2'b10)	$(101000)_2$	$(100100)_2$	$(100010)_2$	$(100001)_2$

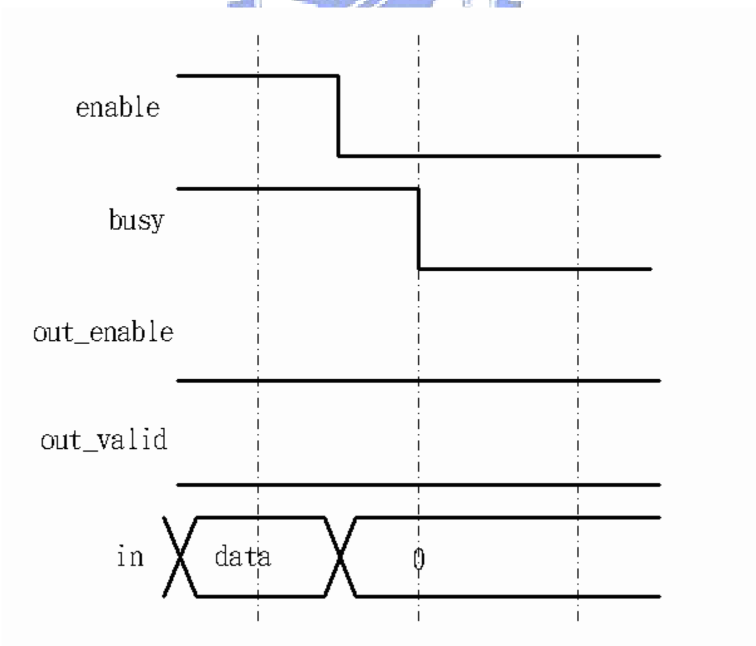

Figure 3.6: Configuration completion

.

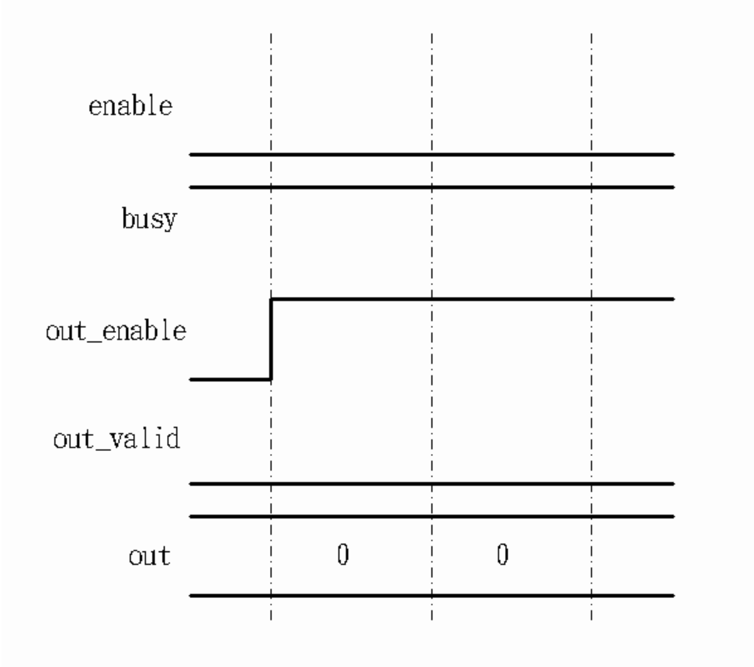

Figure 3.7: Correct output data stored

Figure. 3.6 shows the state of *configuration* completion. When the operation mode type is *Configuration*, the *busy* signal turns from high to low after the *enable* signal turning from high to low. In this case, it is ready for the *encryption/decryption* mode. The *busy* signal is still high after the *enable* signal turning from high to low, if the RSA cryptosystem works at the *encryption/decryption* mode.

Figure. 3.7 shows that the *out_enable* signal is asserted and the *out_valid* signal is idle. That meets that the correct output data has been stored in the memory and ready for loading. The correct output data is ready for reading until the *enable* has been asserted. The *Result* mode indicates transmitting the correct output data form the memory block to the *output* signal, as shown in figure. 3.8. If the data of the *output* signal is correct, the *out_valid* signal will be asserted. When the *Result* mode is complete, as shown in figure. 3.9, the *busy*, *out_valid* and *out_enable* signals will turn from high to low.

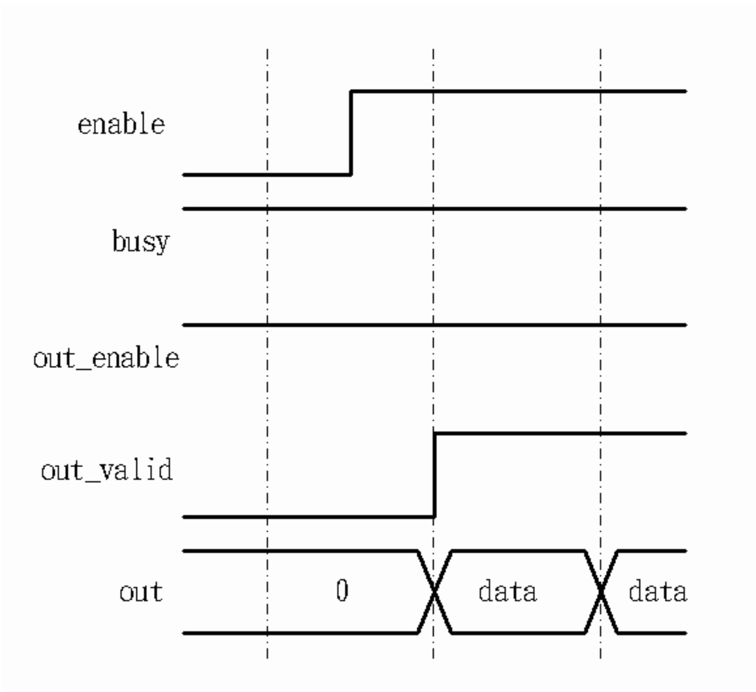

Figure 3.8: Read correct output data

.

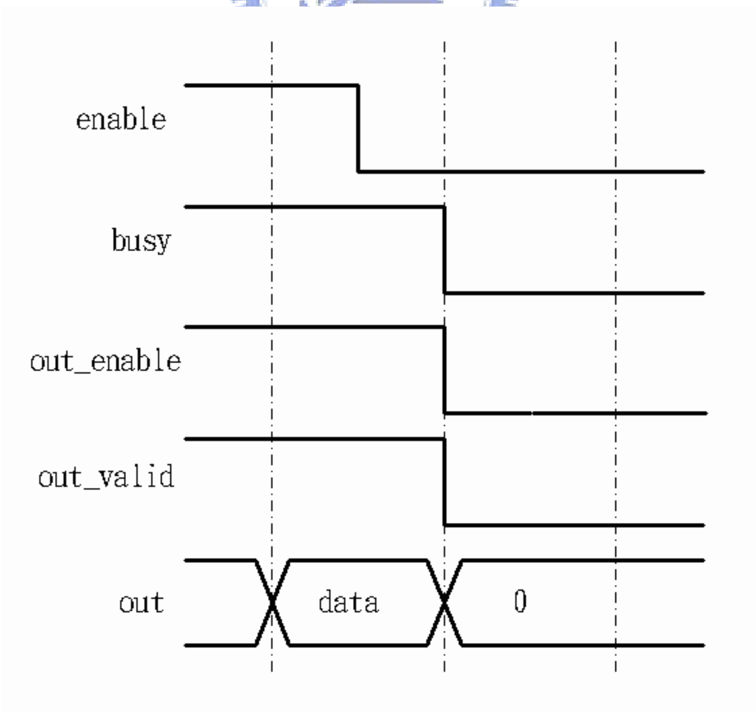

Figure 3.9: Encryption/Decryption finish

.

Chapter 4

Implementation

4.1 Implementation with cell base design

The synthesized result is given bellow. The cycle time is set to 2.7ns and the synthesis standard library is UMC 0.18μm technology. The operation time for encryption/decryption depends on the efficient key length of E_{1024}. For example, if E_{1024}=$(0...010001)_2$=17, the efficient key length of E_{1024} is 5, and if E_{1024}=$(010...01)_2$ =$2^{1022}+1$, the efficient key length of E_{1024} is 1023. Number of clock cycles for each modular product is given by

$$n \times (n+4),$$

n is encryption/decryption block size.

The operation time for encryption/decryption is given by

$$2.7 \times k \times \frac{n}{16} \times (\frac{n}{16}+4)\ (ns),$$

k is the efficient key length of E or D.

The data throughput rate is given by

$$\frac{10^9 \times n}{2.7 \times k \times \frac{n}{16} \times (\frac{n}{16}+4)}\ (bit/sec),$$

Table 4.1: The synthesized results with cell base design

Design	Cell base design			
Length (bits) (key and text)	512	1024	2048	4096
Technology	UMC 0.18μm			
Clock frequency	370 MHz			
Gate count	175.8k			
Baud (kb/s)	314	83	21	5.4

For example, if n=512 and k=17, the data throughput rate is

$$\frac{10^9 \times 512}{2.7 \times 17 \times \frac{512}{16} \times (\frac{512}{16} + 4)}$$
$$= 9.7 \times 10^6 \ (bit/sec)$$
$$= 9.23 \ (Mb/s).$$

If n=512 and k=512, the data throughput rate is

$$\frac{10^9 \times 512}{2.7 \times 512 \times \frac{512}{16} \times (\frac{512}{16} + 4)}$$
$$= 321.5 \times 10^3 \ (bit/sec)$$
$$= 314 \ (kb/s).$$

The proposed RSA cryptosystem was implemented using Verilog. The logic synthesis is performed with Synopsys Design Aalyzer using UMC 0.18μm CMOS standard-cell technologies.

Table 4.2: The verification results on FPGA

Design	FPGA			
Length (bits) (key and text)	512	1024	2048	4096
FPGA card	iProve xc2v8000			
Clock frequency	116.7 MHz			
Number of RAMB16s	5			
Number of slices	6783			
Total equivalent gate count	445,596			
Baud (kb/s)	99	26	6.8	1.7

The clock frequency is set to 370MHz and the evaluate gate count is 175.8k. The detail value is shown as table. 4.1.

4.2 Implementation on FPGA

The verification is given by an integrated FPGA system. The proposed RSA cryptosystem was implemented onto the Xilinx Virtex2: XC2V8000. Table. 4.2 shows that the synthesis frequency is set to 116.7Mhz and the number of slices is 6783.

Chapter 5

Conclusion

An implementation of proposed reconfigurable RSA cryptosystem has been presented in this paper. It is feasible to complete the 512, 1024, 2048 and 4096-bit RSA encryption/decryption with the proposed reconfigurable RSA cryptosystem on FPGA. This uses thirty-four 16-bit ripple adders, two 4096-bit FIFO and five 256*16-bit memory block. There are only 38 bits I/O that are 19 bits input and 19 bits output. The baud rate is 99kb/s for 512-bit, 29kb/s for 1024-bit, 6.8kb/s for 2048-bit and 1.7kb/s for 4096-bit RSA encryption/decryption. The number of slices is 6783.

Table. 5.1 shows the comparison with other 1024-bit RSA implementations with cell base design. The gate count without registers of the proposed architecture is smaller than [7], but the baud rate of the proposed architecture is higher than [7]. Table. 5.3 shows the comparison with other 1024-bit RSA implementations on FPGA. Fournaris [8] uses 1024-stages PE to implement the RSA cryptosystem. The word length of each PE is one bit. Nibouche [9] implements the multipliers of the RSA cryptosystem with the systolic array. McIvor's [10] another approach uses CRT(Chinese Remainder Theorem) to speed up. Tang [11] implement the RSA cryptosystem with radix-2^{17}.

As the table. 5.3 shows, the proposed RSA cryptosystem is small and slow,

Table 5.1: Comparison with other 1024-bits implementations with cell base design

Authors	Proposed	Su [12]	Cho [7]	Mukaida [13]
Technology (μm)	.18	.18	.18	.18
Methodology	Reconfigurable	Coprocessor	not Montgomery	CRT and radix2^{32}
Clock Frequency (MHz)	370 (synthesis result)	83 (measurement result)	40 (synthesis result)	200 (synthesis result)
Total gate count	175.8k	120k	230k	965k
⋆Combinational gate count	37.1k	34k	156k	755k
⋆Register gate count	138.7k	86k	74k	210k
Baud Rate (kb/s)	83	67	78.8	5000

⋆ is indicated that the value is estimated.

but it can work with 512/1024/2048/4096-bit RSA encryption/decryption. For the application of smart cards, it is a nice choice to reduce area.

Table 5.2: Comparison with other 512-bits implementations on FPGA

Authors	Proposed	Shao [14]	Blum [15]	McIvor [10]	Tang [11]
Platform	XC2V8000	XC2V	XC2V4000	XC2V6000	XC2V3000
Methodology	reconfigurable	Booth	N.A.	CRT	radix2^{17}
Clock Frequency (MHz)	116.7	100	95	116	99
Total number of slices	6783	N.A.	4440	13910	8235
⋆Number of slices (w/o registers)	1673	N.A.	1840	N.A.	5163
⋆Number of slices (registers)	5110	N.A.	2600	N.A.	3072
Baud Rate (kb/s)	99	355	86	887	1680

⋆ is indicated that the value is estimated.

Table 5.3: Comparison with other 1024-bits implementations on FPGA

Authors	Proposed	Fournaris [8]	Nibouche [9]	McIvor [10]	Tang [11]
Platform	XC2V8000	N.A.	XC2V8000	XC2V6000	XC2V3000
Methodology	reconfigurable	n-stage PE	systolic	CRT	radix2^{17}
Clock Frequency (MHz)	116.7	129	78	97	90
Total number of slices	6783	7873	19900	26136	14334
⋆Number of slices (w/o registers)	1673	2241	N.A.	N.A.	8190
⋆Number of slices (registers)	5110	5632	N.A.	N.A.	6144
Baud Rate (kb/s)	26	119	148	376	429

⋆ is indicated that the value is estimated.

Bibliography

[1] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 2002.

[2] R. L. Rivest, A. Shamir, and L. M. Adelman, "A method for obtaining digital signatures and public key cryptosystems," *Communication of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[3] K. Koc, "High-speed rsa implementation," tech. rep., RSA Laboratories, 1994.

[4] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519–521, 1985.

[5] C. C. Yang, T. S. Chang, and C. W. Jen, "A new rsa cryptosystem hardware design based on montgomerys algorithm," in *IEEE Trans. on Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 45, pp. 908–913, July 1998.

[6] C. H. Wang, C. P. Su, C. T. Huang, and C. W. Wu, "A word-based rsa crypto-processor with enhanced pipeline performance," pp. 218–221, Aug 2004.

[7] K. S. Cho, J. H. Ryu, and J. D. Cho, "High-speed modular multiplication algorithm for rsa cryptosystem," *IEEE Industrial Electronics Society (IECON)*, vol. 1, no. 6, pp. 479–483, 2001.

[8] A. P. Fournaris and O. Koufopavlou, "A new rsa encryption architecture and hardware implementation based on optimized montgomery multiplication," in

proceedings of 2005 IEEE International Symposium on Circuits and Systems (ISCAS 2005), vol. 5, pp. 4645–4648, May 2005.

[9] O. Nibouche, M. Nibouche, A. Bouridane, and A. Belatreche, "Fast architectures for fpga-based implementation of rsa encryption algorithm," in *Proceedings of the 2003 IEEE International Conference on Field Programmable Technology (FPT)*, pp. 271–278, Dec 2004.

[10] C. McIvor, M. McLoone, and J. V. McCanny, "Modified montgomery modular multiplication and rsa exponentiation techniques," in *IEE Proceedings Computers and Digital Techniques*, vol. 151, pp. 402–408, Nov 2004.

[11] S. H. Tang, K. S. Tsui, and P. H. W. Leong, "Modular exponentiation using parallel multipliers," in *Proceedings of the 2003 IEEE International Conference on Field Programmable Technology (FPT)*, pp. 52–59, Dec 2003.

[12] C. P. Su, C. H. Wang, K. L. Cheng, C. T. Huang, and C. W. Wu, "Design and test of a scalable security processor," in *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC)*, vol. 1, pp. 372–375, Jan 2005.

[13] K. Mukaida, M. Takenaka, N. Torii, and S. Masui, "Design of high-speed and area-efficient montgomery modular multiplier for rsa algorithm," *IEEE Symp. VLSI Circuits*, pp. 320–323, Jun 2004.

[14] J. Shao and S. Kort, "A high speed 512-bit modular multiplier for a rsa chip," *IEEE international Northeast Workshop on Circuits and Systems (NEWCAS)*, pp. 237–240, Jun 2004.

[15] T. Blum and C. Paar, "Montgomery modular exponentiation on reconfigurable hardware," in *Proc. 14th IEEE Symposium on Computer Arithmetic (ARITH-14)*, pp. 70–77, 1999.

作者簡介

姓名: 陳允律

出生: 台北縣

學歷 ： 安坑國小　竹林中學　師大附中

85.9 ~ 89.6 國立中央大學電機工程學系

92.9 ~ 95.7 國立交通大學 電子研究所 (oasis lab)