國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

格向量及多變量後量子密碼系統實作

An efficient ASIC implementation of Lattice-based and Multivariate Post-Quantum Cryptography

施傑仁

Jie-Ren Shih

指導教授：鄭振牟 博士

Advisor: Chen-Mou Cheng, Ph.D.

中華民國 101 年 5 月

May, 2012

國立臺灣大學碩士學位論文
口試委員會審定書

格向量及多變量後量子密碼系統實作

An efficient ASIC implement of Lattice-based and Multivariate Post-Quantum Cryptography

本論文係施傑仁君（學號 R99921028）在國立臺灣大學電機工程學系完成之碩士學位論文，於民國 101 年 06 月 21 日承下列考試委員審查通過及口試及格，特此證明。

口試委員： 鄭振牟 ＿＿＿＿＿＿＿＿＿＿（簽名）
（指導教授）

楊柏因 ＿＿＿＿＿＿＿ [illegible] ＿＿＿＿＿＿＿

陳君明 ＿＿＿＿＿＿＿ ＿＿＿＿＿＿＿

＿＿＿＿＿＿＿ ＿＿＿＿＿＿＿

＿＿＿＿＿＿＿ ＿＿＿＿＿＿＿

系主任 顏嗣鈞 ＿＿＿＿＿＿＿＿＿＿（簽名）

誌謝

在台大的研究生生活裡，我首先要感謝鄭振牟博士以及楊柏因博士擔任我碩士生涯的指導教授。 感謝兩位教授在研究的路上給予的支援及協助，並提供讓學生能專心做研究的環境，幫我們解決各式各樣的難題。

接著我要感謝和我一起完成這份研究的胡永波同學。 永波兄總是能提供我很多我不知道的硬體知識，讓大學對硬體接觸不深的我對設計能夠有更進一步的瞭解。 感謝 Access Lab 的同學們每兩週和我們一起討論，在系統的設計上給予我許多的協助。

接下來我要感謝快速密碼實驗室一起奮鬥的夥伴： 感謝陳大師兄明興和周二師兄彤在我遇到問題時，總是能提供中肯建議與協助。 感謝郭大師博鈞在我旁邊跟我一起度過悶熱的冬天和寒冷的夏天。 永波兄爲我們帶來對岸世界的觀點和文字用法。 有瑋瑋一起互相提醒當兵和畢業的一些注意事項，並和小衿衿帶領大家一起團練增強實驗室戰力。 和阿榜，安安和崧銘在522的團體咪聽一起腦力激盪。 還有每兩週一次的麥當勞外送員，爲我們送達美味的午餐。

最後我要感謝家裡對我的支持，在我需要幫助的時候給我鼓勵以及幫助。

Abstract

In this paper, we present an ASIC implementation of two post-quantum public-key cryptosystems (PKCs), NTRUEncrypt and TTS. It represents a first step toward securing M2M systems using strong, hardware-assisted PKC. In contrast to the conventional wisdom that PKC is too ”expensive” for M2M sensors, it actually can lower the total cost of ownership because of cost savings in provision, deployment, operation, maintenance, and general management. Furthermore, PKC can be more energy-efficient because PKC-based security protocols usually involve less communication than their symmetric-key-based counterparts, and communication is getting relatively more and more expensive compared with computation. More importantly, recent algorithmic advances have brought several new PKCs, NTRUEncrypt and TTS included, that are orders of magnitude more efficient than traditional PKCs such as RSA. It is therefore our primary goal in this paper to demonstrate the feasibility of using hardware-based PKC to provide general data security in M2M applications.

Keywords: *lattice-based Cryptography, Multivariate Cryptography, ASIC, processor-based approach, Bluespec Verilog*

摘要

本篇論文實做一個以處理器架構爲主並且同時支援 NTRU 公鑰加解密系統及 TTS 數位簽章的硬體密碼系統。 我們首先對於兩個系統分別設計出在頻率，面積以及運算圈數的乘積上取得最佳化的特化硬體。 透過設定不同的平行度參數，利用 bluespec 快速建構程式碼並模擬評估以找出佳的參數。 在 TTS 解方程式的計算上，我們使用 Wiedemann 演算法而不是目前普遍使用的高斯消去法。因爲高斯消去法使用的運算單元不容易被其他運算使用,造成面積上的浪費。 而 Wiedemann 運算主要使用矩陣運算，以操作矩陣爲主的運算單元容易被其它運算重覆使用，減少面積的浪費使用。

我們接著設計出一個硬體可以切換計算不同的密碼系統，設計的重點放在兩個系統的運算資源可以互相重復使用，以減少資源上的使用效率不佳。 我們實作了一個計算單元可以分別被 NTRU 或是 TTS 的指令所使用。 並且透過高度的平行化，實踐了一組不同於現在市面的的高位元組處理器，以多個小位元處理器爲主的適合密碼系統運算的硬體。 透過重新載入韌體, 不同參數的密碼系統也可以被執行。

關鍵字: 格基式密碼系統，多變量密碼系統，特殊應用積體電路，處理器, *Bluespec Verilog*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cryptography is the foundation of data security. There are mainly two kinds of cryptography in use today, symmetric-key and public-key cryptography. In the former, the communicating parties are assumed to share one or more secret keys a priori. How they can establish such a shared secret is often referred to as the key-exchange problem. This problem is challenging not only from a technical but also from a managerial point of view, as we will need to manage $O(n^2)$ keys in a network of size n.

Public-key cryptography (PKC), on the other hand, provides an elegant solution to the key-exchange problem. With PKC, key management becomes straightforward, a user can encrypt a short-lived session key using the communicating party's public key and simply send out the encrypted key. PKC ensures that only the holder of the corresponding private key can decrypt and obtain the session key. Furthermore, PKC can provide digital signatures, which, like a person's signature, provides an efficient means of authentication. As a result, PKC proliferates in today's Internet age and permeates many aspects of our daily life from communication to electronic commerce.

However, there is an emerging threat to the prevailing PKCs due to the recent development of quantum computers. The security of RSA, currently the most popular PKC, depends on the difficulty of the integer factorization problem, while that of ECC, the runner-up PKC, depends on the discrete logarithm problem. As Shor has shown, both of them would be solved by large quantum computers in polynomial time [SC97]. Such a threat is more relevant in the M2M context, as these systems tend to operate over a long period of time, and we certainly should take precaution against such a catastrophic attack, even though it might only happen in the distant future.

There are mostly four different kind of approaches compose the post-quantum cryptography: lattice-based cryptography, multivariate cryptography, hash-based signatures and code-based cryptography. In this paper, we focus on NTRUEncrypt as lattice-based cryptography and TTS as multivariate cryptography for candidate in system development.

As networked machines become more popular around our living, information security on these devices becomes an important issue. Traditionally, PKC is regarded as too expensive to deploy in M2M systems. Typical M2M systems only have limited computational power, making deploying strong cryptography on them extremely challenging.

1.1 Previous attempts

There have been numerous proposals how to secure M2M systems from the academic research community [PSW04, ZFZ08] . Most of them use software-based symmetric-key cryptography. For example, TinySec provides link-layer security for sensor networks using software implementation of symmetric-key cryptosystems [KSW04]. In many proposals, more bits will need to be sent over the air for achieving certain level of security, so using hardware accelerators may not necessarily help in these cases [PSW04]. The same functionality would be achieved by PKC in a more communication-efficient way. This is becoming more attractive as computation is getting cheaper in terms of hardware cost and energy consumption, while wireless communication is less so at the same time. As a result, communication is becoming more expensive compared with computation, not to mention the spectrum will become one of the scarcest resources when billions of sensors are deployed and trying to send out their readings over the air. In this case, it is advantageous to use PKC on sensors for the sake of reducing communication cost.

Lastly, there have been several attempts in employing software-based PKC to secure inter-sensor communication [WKfC+04, MWS04]. People have demonstrated that it is possible to run PKC on sensors with acceptable performance. We believe that this is the right direction to pursue, and we plan to take it further by hardware acceleration.

Our approach is to provide a foundation for information security using hardware-assisted PKC. Specifically, we plan to design and implement a complete, proof-of-concept PKC-based system. We choose two types of PKCs to support. First, multivariate cryptosystems enjoy the benefit of executing much faster than traditional cryptosystems on the same hardware, making them ideal for securing sensors in M2M systems [DYB+09]. Second, we will include lattice-based cryptosystems such as NTRUEncrypt to provide encryption for key exchanging [BCE+01]. These are also future-proof in the sense that they can defend against the attack by thousand-qubit quantum computers, which might emerge in the next few decade. Based on these primitives, we can implement security protocols and services like multi-way authentication, key exchange/distribution/management, and digital signature.

1.2 Contributions

We implement an efficient hardware, which is processor-based implemented by Bluespec, supports NTRU and TTS. The first design attempt to optimal the time-area-cycle of each cryptosystem separately. The next design, processor-based design, is focused on the reused part of combinational circuit that computes either NTRU or TTS operations.

By using the experiment of the scalable ASIC, we aim to identify the sharing combinational logic for both system. We implement an unit which can compute either NTRU operations or TTS operations at different times. We parallel the unit to fit the width of linear system of equations solver for resource reuses efficiency . It is a processor-based structure which can be decomposed into register file, ALU and control for extensibility of different systems.

For TTS implementation, we choose Wiedemann algorithm rather than Gaussian elimination for solving system of linear equations taking account of the combinational logic reused area in the design. Systolic Gaussian elimination have been seen for an efficient ASIC for solving system of linear equations, but it is hard for other computation to reuse the circuit. On the other hands, Wiedemann uses matrix multiplication to solve the system. It can be used for the matrix remap in the other state.

As the result, we bring in a coprocessor which executes latticed-base and multivariate cryptography, NTRU and TTS. We target the development of novel lightweight public-key signature primitives A large number of parallel 8- and 16-bit processor consist of simple operation is adequate for NTRU and TTS, rather than 32- or 64-bit powerful processor wildly used in present. We believe that a cryptography-friendly processor will be implemented in the future, instead of general purpose processor.

1.3 Organization

The rest of this paper is organized as follows. In Chapter 2, we give the detail of the implemented algorithms, NTRUEncrypt and TTS, respectively. The hardware design of these algorithms is also described at the end of each section. We shows our implementation strategy for designing the ASIC in Chapter 3 and compare the implementation results in Chapter 4. Specifically, we will show and compare side-by-side the results obtained by a high-level synthesis tool, Bluespec Verilog, against that obtained by the more traditional hand-optimal RTL-based design. Finally, we conclude this paper by giving a few future direction of work in Chapter 5.

Chapter 2

Algorithm

Post-quantum cryptography is the research on cryptographic primitives which are not breakable using quantum computers and classical computers [Ber11].

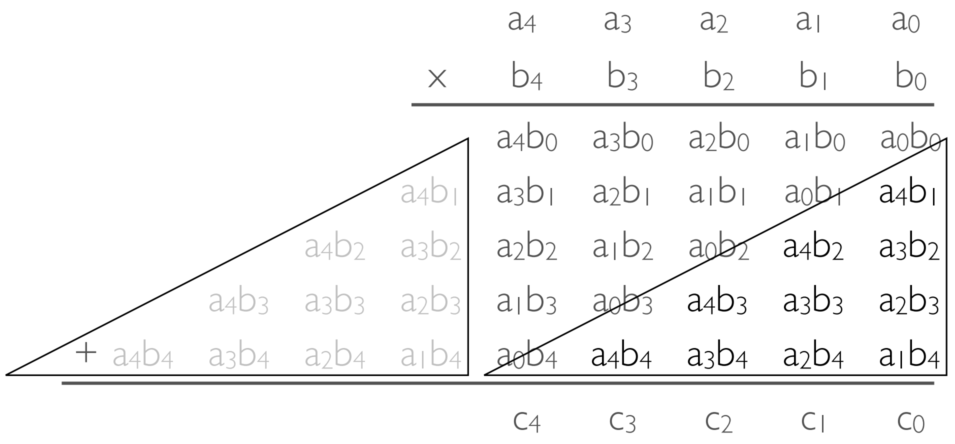
Currently post-quantum cryptography is mostly focused on four different approaches:

- Lattice-based cryptography such as NTRUEncrypt and GGH
- Multivariate cryptography such as Unbalanced Oil and Vinegar and TTS
- Hash-based signatures such as Lamport signatures and Merkle signature scheme
- Code-based cryptography that relies on error-correcting codes, such as McEliece encryption and Niederreiter signatures

In this paper, we focus on NTRUEncrypt and TTS, which we are going to implement for the system. The following sections shows the detail of the algorithm and the present implementation of the system. The system of linear equations solver is also needed for TTS implementation, we compare the pros and cons of Gaussion elimination and Wiedemann algorithm at the end of the chapter.

2.1 NTRUEncrypt Public Key Cryptosystem

NTRUEncrypt is a lattice-based cryptosystem, whose security is based on the hardness of the shortest vector problem in high-dimensional euclidean lattices [HPS98]. The main operations in NTRUEncrypt involve arithmetic in a polynomial ring $R = Z[X]/[X^N - 1]$. The addition in this ring is straightforward polynomial addition, while the multiplication in this ring is convolutional, as shown in Figure [?]. All polynomials in the ring have integral coefficient either modulo P or Q, and their degrees are at most $N-1$, so a typical element can be described as $a = a_0 + a_1X + ...a_{N-1}X^{N-1}$.



a4 a3 a2 a1 a0
× b4 b3 b2 b1 b0
a4b0 a3b0 a2b0 a1b0 a0b0
a4b1 a3b1 a2b1 a1b1 a0b1 a4b1
a4b2 a3b2 a2b2 a1b2 a0b2 a4b2 a3b2
a4b3 a3b3 a2b3 a1b3 a0b3 a4b3 a3b3 a2b3
+ a4b4 a3b4 a2b4 a1b4 a0b4 a4b4 a3b4 a2b4 a1b4
c4 c3 c2 c1 c0

Figure 2.1: Convolutional polynomial multiplication in NTRUEncrypt ring

NTRUEncrypt is parameterized by three parameters, N, P, and Q, which satisfy the following conditions.

- N is a prime number such that the maximal degree for all polynomial in the ring R is $N-1$.
- P and Q are two possible moduli for the coefficients of the polynomials in R, $P \ll Q$, and $\gcd(P, Q) = 1$.

After arithmetic operations in R, the coefficients of the polynomials need to be reduced either modulo P or Q.

2.1.1 Operations

NTRUEncrypt consists of three parts: key generation, encryption, and decryption. In this paper, we only focus on the implementation of encryption and decryption; key generation is often done offline and hence need not be accelerated. Here we focus on the on-line operations that are mostly executed on M2M systems.

Key Generation

A public key h and a private key (f, f_p) are generated as follows.

- Randomly choose a polynomial $f \in R$, with coefficients reduced modulo P.
- Randomly choose a polynomial $g \in R$, with coefficients reduced modulo P.
- Compute the private key f_p as the inverse polynomial of f mod P.
- Compute the private key f_q as the inverse polynomial of f mod Q.
- Compute the public key $h = f_q * g \bmod Q$.

If f is not invertible, then we just choose another f and repeat until we have an invertible f.

Encryption

The ciphertext e is computed from public key h, a random polynomial $r \in R$, and the message $m \in R$.

$$e = P \times r * h + m \bmod Q$$

Decryption

The decryption procedure in three step:

- Compute $a = f * e \bmod Q$

- Shift the coefficients of a to the range $[-Q/2, Q/2]$ and then modulo P
- Compute $d = f_p * a \bmod P$

How it works

It is easy to see why decryption works. By appropriate rewriting, we can see that we are performing the following computation.

$$\begin{aligned}
a &= f * e && \mathrm{mod} Q \\
&= f * (h * r + m) && \mathrm{mod} Q \\
&= f * (P \times f_q * g * r + m) && \mathrm{mod} Q \\
&= P \times r * g + f * m && \mathrm{mod} Q \\
d &= f_p * a && \mathrm{mod} P \\
&= P \times r * g * f_p + f * m * f_p && \mathrm{mod} P \\
&= m && \mathrm{mod} P
\end{aligned}$$

Instead of choosing the coefficient of a in $[0, Q-1]$, they are mapped to the interval $[-Q/2, Q/2]$ so that the original message can be recovered.

2.1.2 Previous attempts

We have found several software and hardware implementations of NTRUEncrypt in the literature; here we only discuss hardware implementations. O'Rourke presented a hardware design for accelerating NTRUEncrypt's core operation of polynomial multiplication in his master thesis [O'R02]. It has a gate count of at least 1483 gates, but the design is not optimized for low-cost, and there is no power consumption report. Bailey presented an FPGA design that targets low-cost systems [BCE$^+$01]. The design uses approximately 60000 gates on a Xilinx Virtex 1000 EFG860 FPGA. There are also implementations that are geared toward RFID (radio-frequency identication) and low-end sensors [ABF$^+$08]. We note that although there have been several attempts that target compact implementation, which is quite suitable for use in M2M systems, our design in this paper not only target lower gate count but also circuit reuse to support signature schemes such as TTS, which we will describe in the next section.

2.2 Multivariate Cryptography

Multivariate Public Key Cryptography is a kind of Public key cryptosystems whose trap door one-way function takes the form of a multivariate polynomial map over a finite field [DYB$^+$09]. In such cryptosystem, the public key is given by a set of polynomials:

$$P = (p_1(w1, ..., w_n), ..., p_m(w_1, ..., w_n)),$$

where p_i is a non-linear polynomial in $w = (w_1, ..., w_n)$: with all coefficients and variables in $\mathbb{K} = \mathbb{F}_p$.

$$p_k(W) := \sum_i P_{ik} w_i + \sum_i Q_{ik} w_i + \sum_{i>j} R_{ijk} w_i w_j$$

The evaluation of these polynomials at any given value corresponds to either encryption or verification procedure. A generic way to construct trapdoors for multivariate cryptosystems is to compose two affine maps before and after a quadratic polynomial map with a special structure. In this case, the trapdoor information consists of the affine maps as well as the central map, with which it is easy to invert the composed public map that otherwise looks random. The one-way function difficulty is based on inverting a multivariate quadratic map, which is equivalent to solving a set of quadratic equations over a finite field.

Definition 1 (Multivariate Quadratic Problem). *Solve the system $p_1(x) = p_2(x) = ... = p_m(x) = 0$, where each p_i is a quadratic in $x = (x_1, \ldots, x_n)$. All coefficients and variables are in $\mathbb{K} = \mathbb{F}_p$.*

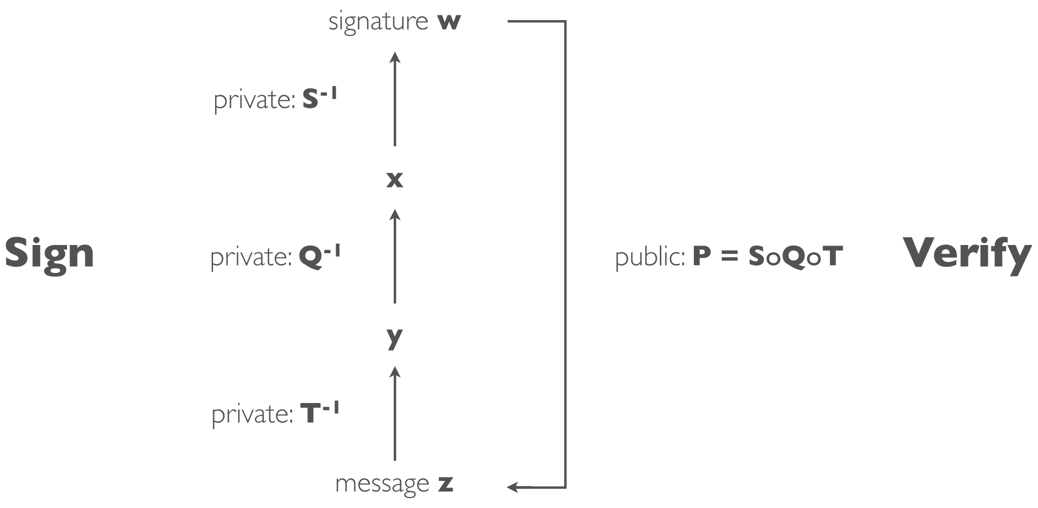


Figure 2.2: Generic structure of multivariate PKCs

Multivariate Quadratic Problem has been proved as an NP-complete problem. The structure of the central map of a multivariate PKC maps elements from $\mathbb{K}^n$ to $\mathbb{K}^m$, while the affine transforms map elements from $\mathbb{K}^n$ to $\mathbb{K}^n$ and $\mathbb{K}^m$ to $\mathbb{K}^m$. The design of central maps mainly can be categorized into two classes. One class consists of small-field schemes, including rather conservative schemes such as Unbalanced Oil and Vinegar (UOV) as well as more aggressively designed proposals such as Rainbow or TTS [DS05]. The other class is big-field schemes, such as Hidden Field Equations (HFE) and Matsumoto-Imai (MIA).

2.2.1 General structure

As shown in Figure 2.2, extant MPKCs almost always hide the private map Q via composition with two affine maps T, S. So, $P = T \circ Q \circ S : K_n \to K_m$, or

$$\mathcal{P} : \mathbf{w} \in K^n \overset{S}{\mapsto} \mathbf{x} = \mathbf{M}_S\mathbf{w} + \mathbf{c}_S \overset{Q}{\mapsto} \mathbf{y} \overset{T}{\mapsto} \mathbf{z} = \mathbf{M}_T\mathbf{y} + \mathbf{c}_T \in K^m$$

In any scheme, the central map Q belongs to quadratic maps whose inverse can be computed relatively easily. The maps S and T are affine and of full rank. The key of a multivariate PKC is the designing of the central map.

To sign a block, one computes $x = T^{-1}(w), y = Q^{-1}(x), z = S^{-1}(y)$. To verify a signature, one simply computes $z = P(w)$. Public key contains only the composed quadratic multivariate polynomials. Private key contains the detail of S and T, as well as the central map Q. Different multivariate PKC schemes usually differ only in Q.

2.2.2 TTS and related schemes

In the design of TTS, it uses UOV scheme as the main operation to construct the central map. By wrapping the input multiple times using UOV, we can construct multi-layer nonlinear map. Such a construction is known as the Rainbow scheme. TTS is a special case of Rainbow, whose central maps is constructed by using two layers of UOV and has a particular sparse form.

Oil and Vinegar schemes

Suppose n is an integer and $o = n - v, v < n$. The variables $x_1, ..., x_v$ are termed vinegar variables and $x_{v+1}, ..., x_n$ oil variables.

Take the map $Q : \mathbb{K}^n \to \mathbb{K}^m$ with form $y = Q(x) = (q_1(x), ..., q_o(x))$, where

$$q_l(x) = \sum_{i=1}^{v} \sum_{j=i}^{n} \alpha_{ij}^{l} x_i x_j, l = 1...o$$

The original Oil and Vinegar scheme has $m = o = v = n/2$. When $o < v$, it becomes the Unbalance Oil and Vinegar signature scheme. If we have a UOV structure, then the quadratic part of each component q_i in the central map from x to y, when expressed as a symmetric matrix, looks like

$$M_i := \left[\begin{array}{ccc|ccc} \alpha_{11}^{(i)} & \cdots & \alpha_{1v}^{(i)} & \alpha_{1,v+1,}^{(i)} & \cdots & \alpha_{1n}^{(i)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{v1}^{(i)} & \cdots & \alpha_{vv}^{(i)} & \alpha_{v,v+1,}^{(i)} & \cdots & \alpha_{vn}^{(i)} \\ \hline \alpha_{v+1,1,}^{(i)} & \cdots & \alpha_{v+1,v,}^{(i)} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1}^{(i)} & \cdots & \alpha_{nv}^{(i)} & 0 & \cdots & 0 \end{array}\right] \text{ or for short, } \left[\begin{array}{c|c} * & * \\ \hline * & 0 \end{array}\right]$$

We can easily figure out that the quadratic terms of the system is the combination of some instead of all variables by looking at the shape of the matrix. These variables are called the vinegar part. By fixing the values of these variables, the quadratic terms of the system will be eliminated. We can then solve the resulting system of linear equations to find the values of the rest variables.

Rainbow scheme

By stacking several layers of UOV together for an invertible central map, we arrive at Rainbow-type constructions.

For $0 < v_1 < v_2 < \cdots < v_{u+1} = n$,

$$S_l := \{1, 2, \ldots, v_l\}$$

$$O_l := \{v_l + 1, \ldots, v_{l+1}\}$$

$$o_l := v_{l+1} - v_l = |O_l|$$

$$Q : X = (x_1, \ldots, x_n) \mapsto Y = (y_{v_1+1}, \ldots, y_n)$$

where each $y_k := q_k(X)$, with following form if $v_l < k \leq v_{l+1}$,

$$q_k = \sum_{i \leq j \leq v_l} \alpha_{ij}^{(k)} x_i x_j + \sum_{i \leq v_l < j < v_{l+1}} \alpha_{ij}^{(k)} x_i x_j + \sum_{i < v_{l+1}} \beta_i^{(k)} x_i.$$

Given all y_i with $v_l < i \leq v_{l+1}$, and all x_j with $j \leq v_l$, we can compute

$x_{v_l+1}, \ldots, x_{v_{l+1}}$ via elimination.

For historical reason, a Rainbow signature scheme is said to be a TTS scheme if the coefficients of Q are sparse.

2.2.3 Previous attempts

There are plenty of works trying to efficiently implement multivariate PKCs. People have implemented TTS on low-cost smartcards [YCC04] and researched into minimized multivariate PKC on low-resource embedded systems [YCCC06]. On modern x86 CPUs, Chen *et al.* uses SSE instructions to implement multivariate PKCs [CCC+09]. Recently, a parallel hardware implementation of the Rainbow signature scheme [BCB+08] and time-area optimized hardware implementation of multivariate signature schemes using systolic arrays [BERW08] are presented. Last but not least, Tang *et al.* tried to minimize cycle counts for Rainbow signature schemes [TYD+11].

2.3 Solving systems of linear equations

The bottleneck operation is the decryption process of TTS and the related schemes is solving a system of linear equations. There are several algorithms that can efficiently solve such problems, all of which have time complexity $o(n^3)$ for general cases.

2.3.1 Systolic Gaussian elimination

Gaussian elimination is perhaps the most well-known algorithm to solve a linear system of equations.

In addition to solving, Gaussian elimination can also be used to find the rank of a matrix, to calculate the determinant of a matrix, and to calculate the inverse of an invertible square matrix.

Algorithm 1 Gaussian elimination

```
for $k = 1 \to m$ do
    Find pivot for column k:
    $i_{max}$ = argmax ( $i = k \to m$ , abs(A[i, k]))
    if A[$i_{max}$, k] = 0 then return Error
    end if
    swap rows(k, $i_{max}$)
    Do for all rows below pivot:
    for $i = k + 1 \to m$ do
        Do for all remaining elements in current row:
        for $j = k + 1 \to n$ do A[i, j] := A[i, j] - A[k, j] * (A[i, k] / A[k, k])
        end for
        Fill lower triangular matrix with zeros:
        A[i, k] := 0
    end for
end for
```

Systolic Gaussian elimination is a fully pipelined parallel hardware design optimized for solving systems of linear equations. It is widely used for multivariate PKC for inverting the central map, which is regarded as optimal in terms of trade-offs between speed and area [BERW08].

Systolic Gaussian elimination is the space-time minimal ASIC implementation for solving linear systems of equations, in which an architecture of simple processors are used as systolic cells connected in a triangular network. Systolic implementation can reduce the combinational area in the total design. At the same time, the timing delay can be minimized via pipelining.

2.3.2 The Wiedemann algorithm

The Wiedemann algorithm is a randomized algorithm for solving systems of linear equations over a finite field. It is usually used for solving a sparse matrix because there it only involves $O(n^2w)$ operations for a sparse matrix, where w is the number of nonzero coefficients in each equation.

Definition 2 (Linear recurrence)**.** *A sequence $a = (a_i)_{i \in \mathbb{N}}$ is said to be linear recur-*

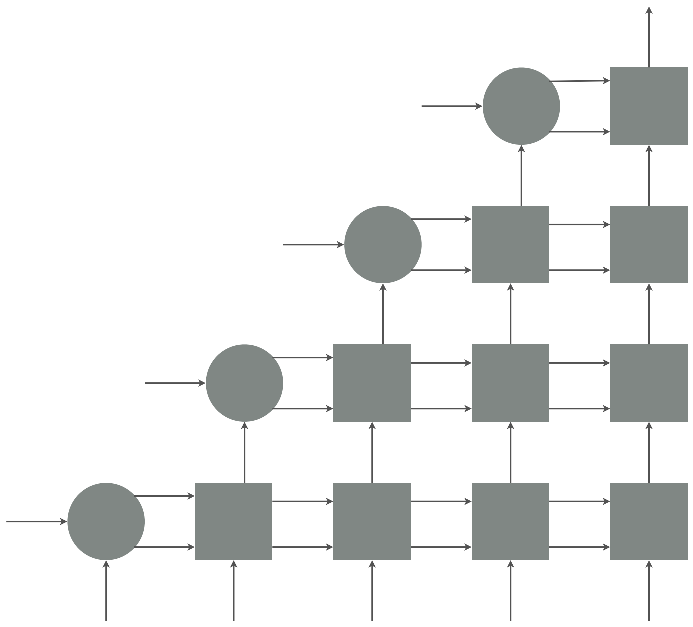


Figure 2.3: Circuit diagram for systolic Gaussian elimination

rent over $\mathbb{F}$ if there exist $n \in \mathbb{N}$ and $f_0, \ldots, f_n \in \mathbb{F}$ with $f_n \neq 0$ such that

$$\sum_{0 \leq j \leq n} f_j a_{i+j} = 0, \forall i \in \mathbb{N}.$$

The polynomial $f(x) = \sum_{0 \leq j \leq n} f_j x^j \in F[x]$ of degree n is called a characteristic polynomial of a.

Definition 3 (Minimal polynomials)**.** *$f(x) = \sum_{0 \leq j \leq n} f_j x^j$ is called the minimal polynomial of the sequence a if f is a characteristic polynomial of a of the least degree.*

The Berlekamp-Massey algorithm finds the shortest linear feedback shift register (LFSR) that produces a given binary sequence. The algorithm will also find the minimal polynomial of a linear recurrent sequence in the field. It is used in the Wiedemann algorithm to find the minimal polynomial of a sequence.

Algorithm 2 Berlekamp-Massey

```
Require: $(a_0, ..., a_{2d-1}$, a linear recurrent sequence whose minimal polynomial is of
   degree $\leq d$
Ensure: $f(x) = \sum_{0 \leq j \leq n} f_j x^j$ that $f(x)$ is the minimal polynomial of $a$
   Let $\Lambda(x) = 1, K(x) = 0, l = 1, \delta = 1$.
   for $j$ from 1 to 2d do
       Set $\gamma = \Lambda_0 a_{j-1} + ... + \Lambda_n a_{j-(n+1)}$
       if $\gamma = 0$ then
           $K(x) = xK(x)$
       else
           $\Theta(x) = \Lambda(x)$
           $\Lambda(x) = \Lambda(x) - \frac{\gamma}{\delta} xK(x)$
           if $2l < j$ then
               $K(x) = \Theta(x)$
               $\delta = \gamma$
               $l = j - l$
           else
               $K(x) = xK(x)$
           end if
       end if
   end for
```

The Wiedemann algorithm

Let $\mathbb{F}$ be a finite field. Given $A \in \mathbb{F}^{n \times n}$ nonsingular and $b \in \mathbb{F}^n$, we need to find a $y \in \mathbb{F}^n$ such that $Ay = b$.

The idea of the Wiedemann algorithm is to consider the sequence $a = (A^i b), i \in \mathbb{N}$. Let f be the minimal polynomial of a. According to the definition of minimal polynomial, we have

$$f(A) \cdot b = 0 = \sum_{0 \leq j \leq d} f_j A^j \cdot b = f_0 b + \sum_{1 \leq j \leq d} f_j A^j \cdot b.$$

Therefore,

$$-f_0 b = \sum_{1 \leq j \leq d} f_j A^j \cdot b.$$

Hence,

$$A \cdot y = b = -f_0^{-1} \sum_{1 \leq j \leq d} f_j A^j \cdot b,$$

and therefore,

$$y = -f_0^{-1} \sum_{1 \leq j \leq d} f_j A^{j-1} \cdot b.$$

Algorithm 3 Wiedemann

```
Require: A non-singular matrix $A \in \mathbb{F}^{n \times n}$ and a vector $b \in \mathbb{F}^n$
Ensure: $y = A^{-1}b, y \in \mathbb{F}^n$
  Generate recurrent sequence $a = (A^i b)_{i \in \mathbb{N}}$
  Using $a$ as input, compute minimal polynomial by Berlekamp-Massey
  $h = -\frac{m-m_0}{m_0 x}$, compute $y = h(A)b$
```

Chapter 3

Implementation

In this section, we describe the tools and the strategy used for our ASIC design. We use Bluespec SystemVerilog to generate parameterized designs. Finally, we present our processor-based design that supports several post-quantum PKCs.

3.1 Bluespec

Bluespec SystemVerilog is an EDA tool set for ASIC and FPGA design [Nik04]. It is a hardwaredescription language that compiles into either Verilog RTL or a cycle-accurate C simulation program. It provides high productivity by a radically different approach to high-level synthesize [Nik04].

3.1.1 Bluespec basics

Bluespec SystemVerilog uses the model of hardware structure similar to that used by Verilog. It supports modules and interfaces. Interfaces are used for communication between modules, and module hierarchies can be obtained by static elaboration. Modularity helps improve productivity and allows rapid design-verification cycles. Like SystemC, it provides a software model and hence is friendly to software engineers. It also generates more efficient RTL than most SystemC implementations.

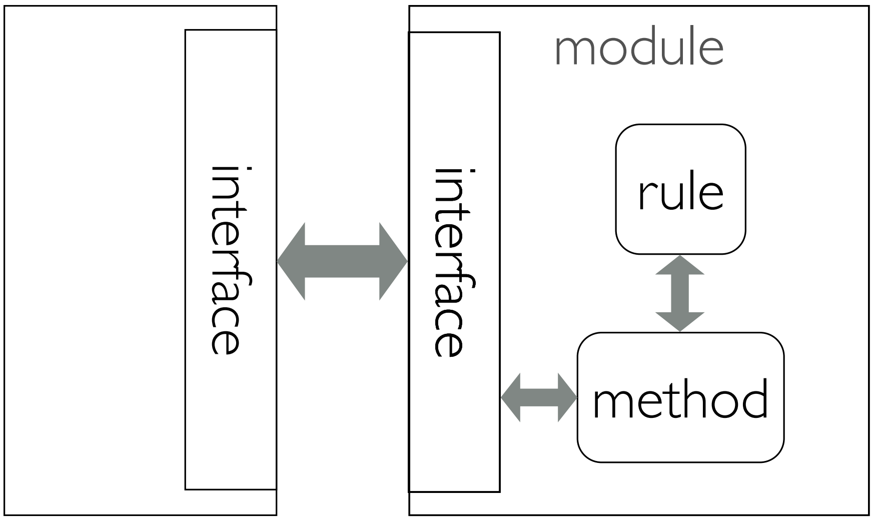


Figure 3.1: Modules, interfaces, and methods in Bluespec SystemVerilog

3.1.2 Modules and interfaces

In Bluespec SystmeVerilog, a module's behavior is specified using a collection of rules. A module's interface is specified with a collection of methods. A rule in one module may invoke methods in other modules. Although rules may span multiple modules, they may be composed in a modular way with conditions and actions that are localized and encapsulated within each module. External users of a module only need to deal with the specified input and output. With such a design, a divide-and-conquer style can be easily supported.

3.1.3 Atomic actions and rules

Rules and actions atomicity are a powerful tool for ensuring correctness. The module can be understood as some sequential composition of rule firings. Each module's designer can locally encapsulate the correctness conditions for use of its interface and be assured that it cannot be used incorrectly from any context. The compiler will check the conditions statically to ensure the correctness of the design.

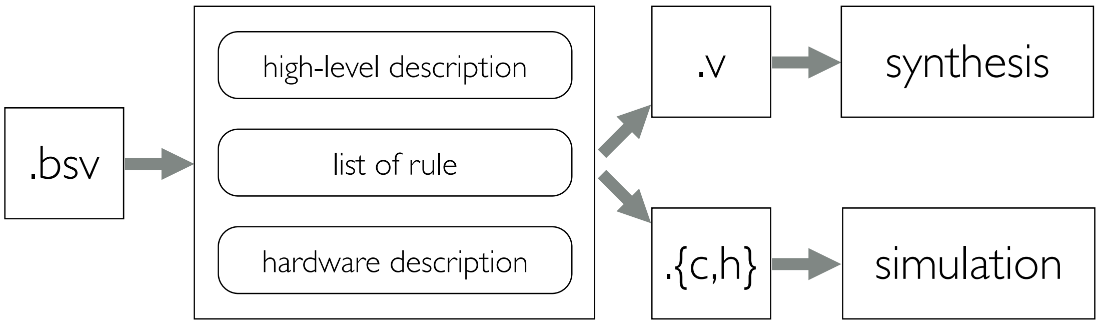


Figure 3.2: Bluespec SystemVerilog compiler structure

3.1.4 Fast compilation and simulation

The compiler of Bluespec SystemVerilog statically verifies user-specified assertions to ensure that the whole design is correct. It also supports abstract data types to preserve representation invariants, enhanced overloading, and bit-width constraints.

The Bluespec compiler generates both datapath and control hardware to implement rule behavior in parallel hardware. The core of the generated control hardware is a dynamic scheduler that governs the rule firings. The generated RTL can be sent to standard netlist synthesis and physical design tools. The timing and area are competitive with hand-written RTL for most design.

It can also compile into C code for fast simulation. Both kinds of output are cycle accurate to each others and can dump to standard VCD files. The C-based simulation is faster than Verilog simulation because it can exploit rule semantics to optimize the execution.

3.1.5 Atom-based design

As in designing any complex systems, we analyze and decompose the system into atomic operations. We then design units of combinational circuit, called atoms, for those different operations in a time-area minimized way so as to get a better understanding how far we can go. In addition to being the unit of optimization, atoms can be seen as a basic building block in building our ASIC design.

Under such a design strategy, atoms are the main combinational part of the whole ASIC. After defining the atom, we then increase parallelism by increasing the number of atoms. Often, this can decrease the time-area-cycle product. The intuition behind is that there are inevitably some fixed cost such as registers that one needs to include no matter at all parallelism levels, and these parts consume energy during the whole course of the computation. Therefore, we can reduce total energy consumption if we can finish the computation earlier via, e.g., increasing parallelism.

By using Bluesoec SystemVerilog, we can code once and generate many ASIC designs with different parameters. It helps us have a quick look at the trend of the time-area-cycle product and find the best trade-off point, a process usually referred to as architectural exploration.

3.1.6 Array-based design

Compared with atom-based design, array-based design can be seen as the alternative in which we increase parallelism inside an atom. In contrast to atom-based design, we first fill the system with a fixed number of atoms, called an array, and then increase the computational power of each atom. The idea is still to get the job done sooner and decrease the total number of cycles needed to finish the computation, thereby reducing the total energy consumption. However, the critical path may increase, resulting a prolonged clock cycle and hence an increase in time-area-cycle product. We hope to find an optimal point before the margin return begins to diminish.

3.2 Processor-based design

Eventually, we plan to have a programmable cryptographic processor that can support various cryptosystem via firmware. It can be separated into three part, combinational logic, memory, and control logic. Control logic will decide which function to

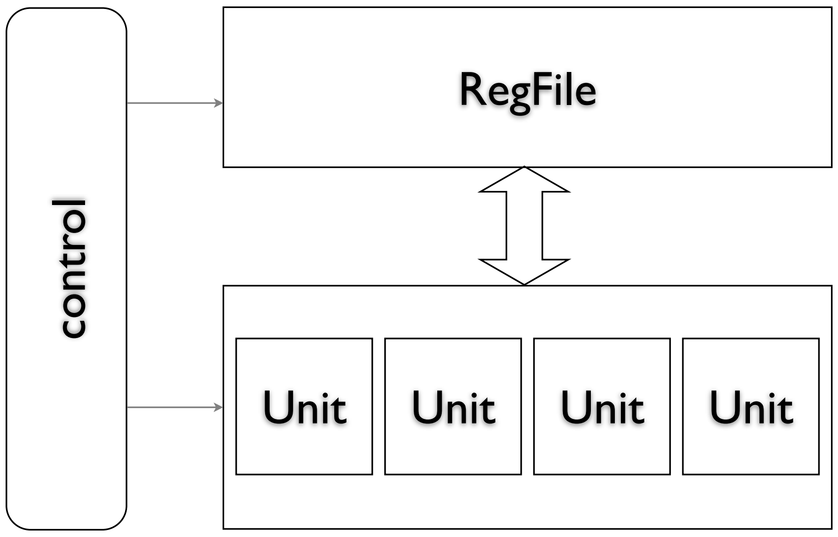


Figure 3.3: Processor-based design

perform and move the data from memory to combinational logic at the appropriate times. With different functions, it would reconfigure the control logic appropriately.

We have implemented a preliminary processor that supports both NTRUEncrypt and TTS schemes. Our strategy works as follows. First, we find the part in each atom-based ASIC that can be shared and combine them into a combinational logic that can compute the basic operations of both PKCs. We then parallelize the atoms to form arrays of appropriate sizes that are suitable for both systems. Finally, we increase each atom's computational power until we find the best trade-off point in terms of time-area-cycle product.

Chapter 4

Experiment Result

We present the implementation results in this section. First, we show the implementation results of ASIC designs of NTRUEncrypt and TTS, respectively. We then move to the processor-based design that supports both cryptosystems and compare the time-area-cycle product with the combined ASIC design. All results are obtained by synthesis with Synopsys Design Compiler at 90nm process.

4.0.1 NTRUEncrypt ASIC

In NTRUEncrypt ASIC design, we compare the results of two approaches, atom-based and array-based designs. Atom-based design scale in one dimension (degree of parallelism), while array-based design adds an additional dimension of computational power. We compare the performance of timing, area, and total cycle count under different parameter settings.

4.0.2 Atom-based approach

The bottleneck operation in encryption and decryption of NTRUEncrypt is the convolutional polynomial multiplication. We decompose the resources used to compute polynomial multiplication into sequential and combinational parts. The sequential part cannot be parallelized, such as registers needed for temporary storage and con-

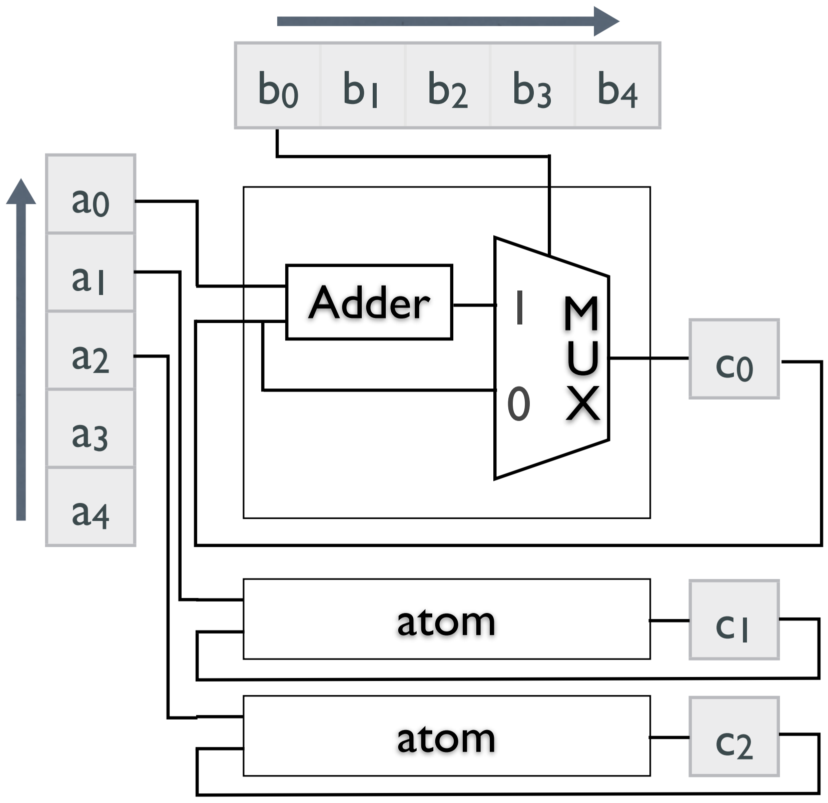


Figure 4.1: Atom-based design diagram

trol. The combinational part is the part that actually carries out the computation. It can be parallelized to reduce the total cycle count, but it increases the circuit size, as well as might increase the length of the critical path.

The resulting circuit of the atom-based design is shown in Figure 4.1. We note that the modulo circuitry is not shown here for brevity. Also, the mux controls whether the registers will be updated with new values.

An atom can compute a term of polynomial multiplication in the product given the two input polynomial in N cycles. The whole polynomial multiplication needs N^2/P cycles if P atoms are used. We are interested in the total energy consumption of the encryption and decryption processes, so we use time-area-cycle product as a figure of merit when comparing the resulting performance under different parameter settings.

Figure 4.2 and Table 4.1 show that, while the number of atoms increases, the time-area-cycle product decreases rapidly.

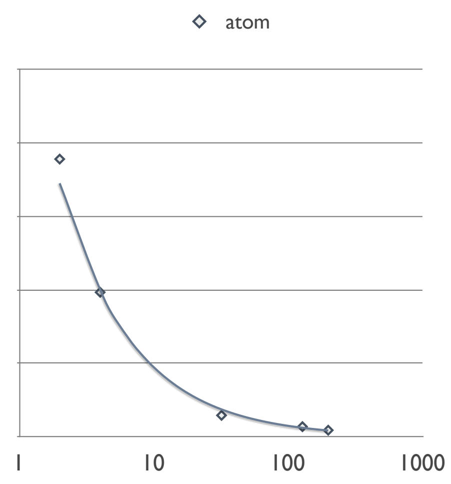


Figure 4.2: Performance under various parameter settings for atom-based design

#atom	Area	Timing	#cycle	Time-area-cycle product
2	275224	4.12	200	226,784,576
4	276908	4.22	101	118,023,728
32	303685	4.12	14	17,516,551
128	401246	4.14	5	8,305,792
199	432121	4.15	3	5,379,906

Table 4.1: Time-area-cycle products for atom-based design

This is because the combinational part only occupies a small percentage of the total area, and as a result, a higher degree of parallelism helps reduce idle circuitry and hence lower total energy consumption. Also, increasing the number of atoms does not affect the timing of the whole design because it does not increase the length of the critical path. We conclude that one should use an as high degree of parallelism as possible to have a better time-area-cycle product in atom-based design.

4.0.3 Array-based design

In atom-based design, more atoms will lead to better time-area-cycle product. So in array-based design, we first use a maximal number of atoms and try to increase the computational power of each atom. We are interested to know whether this would improve the time-area-cycle figure of merit. Indeed, as we add P terms in a cycle, the total number of cycles needed will reduce to $1/P$. However, timing gets worse

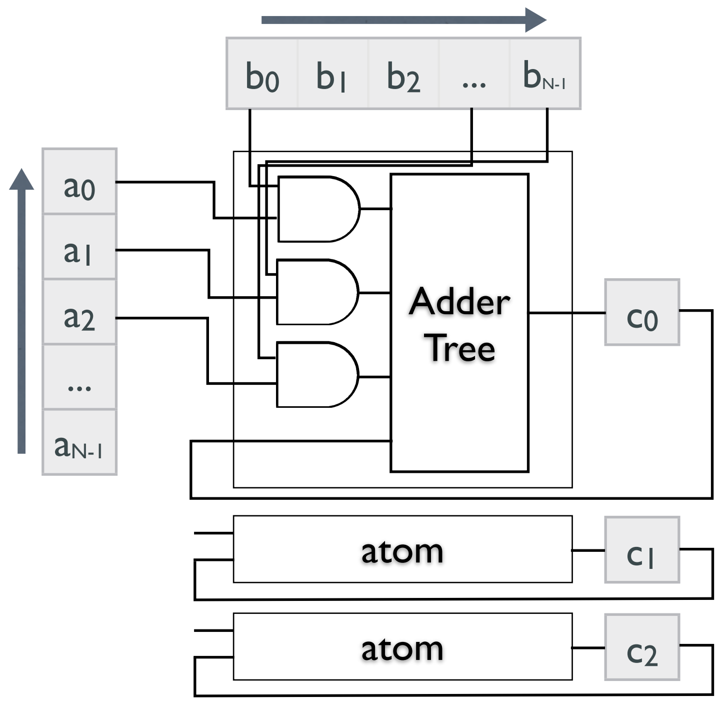


Figure 4.3: Array based implementation circuit

because we increase the height of the adder tree at the output of the circuit. There should be an optimal point where adding increasing the computational power will no longer lead to an improved time-area-cycle product.

With the help from Bluespec SystemVerilog, we generate four designs corresponding to adding 1, 3, 7, and 15 terms in a single cycle and summarize the results in Figure 4.4 and Table 4.2.

#atom	Area	Timing	#cycle	Time-area-cycle product
1	579721	4.27	1	2475409
3	1230304	6.83	1/3	2800992
7	2342892	19.78	1/7	3608054
15	4658949	14.42	1/15	4478803

Table 4.2: Time-area-cycle products for array-based design

As we have expected, the combinational part grows as the computational power of each atom increases, and the length of the critical path also grows. This makes the product increases as the cycle count decreases. Overall, as the per-atom computational power increases, the length of the critical path grows more rapidly than

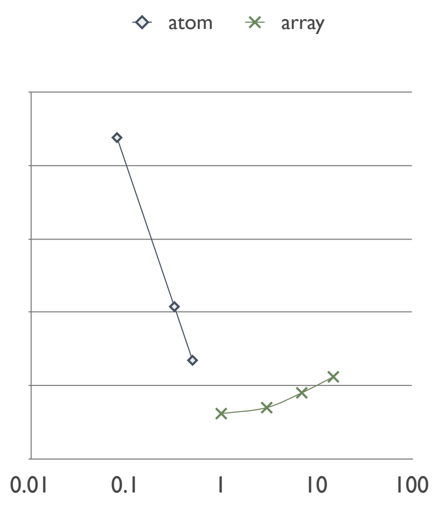


Figure 4.4: Performance under various parameter settings for array-based design

the reduction of cycle count, so it does not help the time-area-cycle product by increasing the computational power.

Finally, we compare the performance of atom-base and array-based designs and find that the optimal trade-off point for an NTRUEncrypt ASIC design appears at the point where there are a maximal number of atoms with each atom having minimal computational power of a single-layer adder.

4.1 TTS ASIC

As evaluation of a quadratic map can be computed using linear-system evaluation as a basic building block, the bottleneck computation of TTS is linear-system evaluation and solving. In this section, we first present the ASIC implementation of Gaussian elimination, the state-of-the-art linear system solver, as well as our attempt in using the Wiedemann algorithm to solve linear systems. To support both encryption and decryption, we use matrix-vector multiplication to maximize the degree of resource reuse. Finally, as in previous section, we will discuss the performance under various parameter settings such as parallelism in terms of time-area-cycle product and find the optimal trade-off point in the design space.

Linear system solvers

Systolic Gaussian elimination has been wildly used for solving systems of linear equations over the reals or a finite field. It is an optimal design in terms of time-area product. However, it requires a dedicated combinational circuit that cannot be easily reused for other computations.

We investigate the Wiedemann algorithm for solving linear systems of equations. With Wiedemann, linear-system solving can be decomposed into matrix-vector multiplication, which is ideal for reuse in other parts of the computation of a multivariate PKC. However, such a reuse comes at a price that the computational complexity is twice as high as Gaussian elimination. We are interested in finding out how much we can save by using a reuse-friendly algorithm for linear-system solving.

Matrix-based design

The design of a scalable matrix-vector multiplication is similar to NTRUEncrypt's array-based design. We use 20 atoms, which is dictated by the number of variables in the target TTS algorithm. We then scale the number of the multipliers in each atom to control the degree of parallelism.

We generate five designs corresponding to five different levels of computational power capable of computing the inner product of two vectors of dimensions 1, 2, 4, 5, and 10. The inputs are multiplied in parallel before sent to the final adder tree for summation. The critical path consists of that of the multiplier as well as the final adder tree.

As shown in Figure 4.5 and Table 4.3, the result is quite different compared with that of NTRUEncrypt.

Recall that the performance improvement of array-based NTRUEncrypt design decreases as the computational power of each atom increases. However, for matrix-based TTS design, the optimal trade-off point happens when each atom has a medium level of computational power. It is because the delay of the multiplier

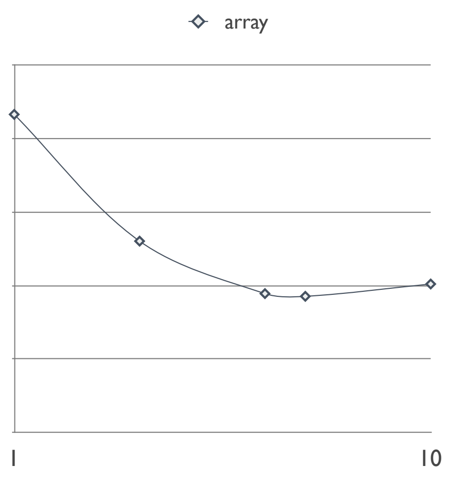


Figure 4.5: Performance under various parameter settings for matrix-based design

Dimension	Area	Timing	#cycle	Time-area-cycle product
1	222013	9.07	21	60616125
2	254413	9.45	11	36470858
4	330818	10.31	6	26473853
5	363404	11.26	5	25955876
10	564705	14.22	3	28300857

Table 4.3: Time-area-cycle product for matrix-based design

is larger than that of the final adder tree, so it makes sense to fill up the gap by adding more partial products in each atom. If we use Wallace adder tree in our atom design, this ratio might change, so will the optimal trade-off point.

4.1.1 Processor-based design

As shown in Figure 4.6, we first classify circuits into combinational logic, register, and control logic like in a typical processor design. We use the standard general-

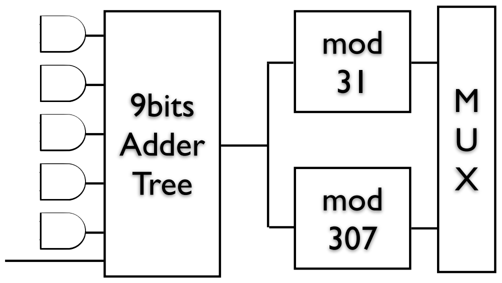


Figure 4.6: Processor-base design that supports both NTRUEncrypt and TTS

purpose register file construction to implement the registers, while control logic is implemented as a finite-state machine. We then combine NTRUEncrypt and TTS ASIC designs and reuse as much as possible the common components. For combinational logic, we implement a datapath unit that can either compute a multiplication and an addition module 31, or six additions module Q. We duplicate 20 such datapath units; this is the maximal degree of parallelism allowed by the target TTS scheme.

Table **??** shows that by lightly increasing the combinational logic and therefore sacrificing some timing performance, we can obtain a circuit that can efficiently support two cryptosystems.

	Combinational area	Timing
NTRUEncrypt	2814	7.16
TTS	2076	7.16
Processor-based	3633	7.97

Table 4.4: Combinational logic comparing

The resulting circuit is about 18% smaller than the two ASIC designs combined while delivering a satisfactory performance in terms of power consumption.

4.2 Verilog versus Bluespec Verilog

Finally, we evaluate the quality of the RTL code generated by Bluespec SystemVerilog by comparing its performance to hand-optimized RTL code. From Table 4.5, we can see that the RTL code generated by Bluespec SystemVerilog is in general only 2 to 3 times worse than hand-optimized RTL code for the same designs.

	Bluespec SystemVerilog	Hand-optimized RTL
NTRUEncrypt	579721	176389
TTS	338245	172000

Table 4.5: Bluespec generated vs. hand-optimized RTL code

Therefore, we conclude that Bluespec SystemVerilog is suitable for early design and automatic architectural exploration. After preliminary studies find an optimal architecture, experienced RTL designer can then come in and further optimize the design by hand.

Chapter 5

Conclusion

In this paper, we report our experience implementing a scalable ASIC design for both NTRUEncrypt and TTS. We also use Bluespec SystemVerilog to generate parametrized designs and find the optimal trade-off points under different parameter settings. By analyzing the timing-area-cycle products of these designs, the optimal degree of parallelism can be determined from an architectural design viewpoint.

We also present a combined design that supports both NTRUEncrypt and TTS. It has a similar design to a processor. In such a processor-based design, not only registers but also combinational circuits are shared among the different supported cryptosystems for maximal resource reuse. The resulting design is more similar to a multi-core 8-bit processor than a single-core 16- or 32-bit processor, which is not as straightforward without the knowledge learned from our effort of architectural exploration.

After combining NTRUEncrypt and TTS into a single system, the logical next step we can try is to add support for further more cryptosystems. Our goal is to secure M2M systems with strong cryptography by means of hardware acceleration. Also, to support a larger number of cryptosystems, a compiler is needed to generate the complex firmware. At the end, we hope that we will be able to demonstrate the feasibility of using hardware-based PKC to provide general data security in M2M applications.

Bibliography

[ABF$^+$08] Ali Can Atici, Lejla Batina, Junfeng Fan, Ingrid Verbauwhede, and Siddika Berna Örs. Low-cost implementations of ntru for pervasive security. In *ASAP*, pages 79–84, 2008.

[BCB$^+$08] Sundar Balasubramanian, Harold W. Carter, Andrey Bogdanov, Andy Rupp, and Jintai Ding. Fast multivariate signature generation in hardware: The case of rainbow. In *ASAP*, pages 25–30, 2008.

[BCE$^+$01] Daniel V. Bailey, Daniel Coffin, Adam J. Elbirt, Joseph H. Silverman, and Adam D. Woodbury. Ntru in constrained devices. In *CHES*, pages 262–272, 2001.

[Ber11] Daniel J. Bernstein. Post-quantum cryptography. In *Encyclopedia of Cryptography and Security (2nd Ed.)*, pages 949–950, 2011.

[BERW08] Andrey Bogdanov, Thomas Eisenbarth, Andy Rupp, and Christopher Wolf. Time-area optimized public-key engines: -cryptosystems as replacement for elliptic curves?. In *CHES*, pages 45–61, 2008.

[CCC$^+$09] Anna Inn-Tung Chen, Ming-Shing Chen, Tien-Ren Chen, Chen-Mou Cheng, Jintai Ding, Eric Li-Hsiang Kuo, Frost Yu-Shuang Lee, and Bo-Yin Yang. Sse implementation of multivariate pkcs on modern x86 cpus. In *CHES*, pages 33–48, 2009.

[DS05] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *ACNS*, pages 164–175, 2005.

[DYB$^+$09] Jintai Ding, Bo-Yin Yang, Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. *Multivariate Public Key Cryptography*, pages 193–241. Springer Berlin Heidelberg, 2009.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. pages 267–288, 1998.

[KSW04] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: a link layer security architecture for wireless sensor networks. pages 162–175, 2004.

[MWS04] David J. Malan, Matt Welsh, and Michael D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. pages 71–80, 2004.

[Nik04] Rishiyur S. Nikhil. Bluespec system verilog: efficient, correct rtl from high level specifications. In *MEMOCODE*, pages 69–70, 2004.

[O'R02] Colleen Marie O'Rourke. Efficient ntru implementations. 2002.

[PSW04] Adrian Perrig, John A. Stankovic, and David Wagner. Security in wireless sensor networks. pages 53–57, 2004.

[SC97] Peter W. Shor and Siam J. Comput C. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. pages 1484–1509, 1997.

[TYD$^+$11] Shaohua Tang, Haibo Yi, Jintai Ding, Huan Chen 0004, and Guomin Chen. High-speed hardware implementation of rainbow signature on fpgas. In *PQCrypto*, pages 228–243, 2011.

[WKfC^{+}04] Ronald J. Watro, Derrick Kong, Sue fen Cuti, Charles Gardiner, Charles Lynn, and Peter Kruus. Tinypk: securing sensor networks with public key technology. In *SASN*, pages 59–64, 2004.

[YCC04] Bo-Yin Yang, Jiun-Ming Chen, and Yen-Hung Chen. Tts: High-speed signatures on a low-cost smart card. In *CHES*, pages 371–385, 2004.

[YCCC06] Bo-Yin Yang, Chen-Mou Cheng, Bor-Rong Chen, and Jiun-Ming Chen. Implementing minimized multivariate pkc on low-resource embedded systems. In *SPC*, pages 73–88, 2006.

[ZFZ08] Yun Zhou, Yuguang Fang, and Yanchao Zhang. Securing wireless sensor networks: A survey. pages 6–28, 2008.