AN ABSTRACT OF THE DISSERTATION OF

Qingwei Li for the degree of Doctor of Philosophy in

Electrical and Computer Engineering presented on January 4, 2008.

Title: Efficient VLSI Architectures for MIMO and Cryptography Systems

Abstract approved: _____

Zhongfeng Wang

Multiple-input multiple-output (MIMO) communication systems have recently been considered as one of the most significant technology breakthroughs for modern wireless communications, due to the higher spectral efficiency and improved link reliability. The sphere decoding algorithm (SDA) has been widely used for maximum likelihood (ML) detection in MIMO systems. It is of great interest to develop low-complexity and high-speed VLSI architectures for the MIMO sphere decoders.

The first part of this dissertation is focused on the low-complexity and high-speed sphere decoder design for the MIMO systems. It includes the algorithms simplification, and transformations, hardware optimization and architecture development. Specifically, we propose the layered reordered K-Best sphere decoding algorithm and dynamic K-best sphere decoding algorithm, which can significantly improve the detection performance or reduce the hardware complexity. We also present the efficient K-Best sorting architecture, which greatly simplifies the sorting operation of the K-Best SDA. In addition, we introduce the early-pruning K-Best SD scheme, which eliminates the unlikely

candidate at early decoding stages, thus saves computational complexity and power consumptions. For the conventional sphere decoder design, we develop the parallel and pipeline interleaved sphere decoder architecture, which considerably increases the decoding throughput with negligible extra complexity. Finally, we design the efficient radius and list updating units for the list sphere decoder, which increases the speed of obtaining the new radius and reduces the complexity for generating the new candidate list.

The wireless communication technologies are widely used for the benefits of portability and flexibility. However, the wireless security is extremely important to protect the private and sensitive information since the communication medium, the airwave, is shared and open to the public. Cryptography is the most standard and efficient way for information protection.

The second part of this thesis is thus dedicated to the high-speed and efficient architecture design for the cryptography systems including ECC and Tate pairing. We propose an efficient fast architecture for the ECC in Lopez-Dahab projective coordinates. Compared with the conventional point operation implementations, the point addition and doubling operations can be significantly accelerated with reasonable hardware overhead by applying parallel processing and hardware reusing. Moreover, we develop a complexity reduction scheme and an overlapped processing architecture for the Tate pairing in characteristic three. The proposed architecture can achieve over 2 times speedup compared with conventional sequential implementations for the Duursma-Lee and Kwon-BGOS algorithms.

Efficient VLSI Architectures for MIMO and Cryptography Systems

by

Qingwei Li

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented January 4, 2008
Commencement June 2008

Doctor of Philosophy dissertation of Qingwei Li presented on January 4, 2008.

APPROVED:

Major Professor, representing Electrical and Computer Engineering

---

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Qingwei Li, Author

ACKNOWLEDGEMENTS

First and foremost, I would like to express the sincere gratitude to my respected advisor, Dr. Zhongfeng Wang for his invaluable advice, incessant guidance, continuous encouragement, and financial support (through the National Science Foundation and National Aeronautics and Space Administration) throughout the course of my study and research at Oregon State University. His ample knowledge, rigorous working attitude, honest personality and eagerness for new technology are always my model to follow in my future study and work.

As well, I would like to thank all the people of School of EECS for providing such an excellent education and research environment. My special thanks go to the members of my Ph.D. program committee – Dr. Albrecht Jander, Dr. Huaping Liu, Roger Traylor, and Dr. William Warnes for their advice and help on my Ph.D. program.

I would also like to thank all my friends and colleagues at Oregon State University for their friendships and support during my past study life, especially our group members, Dr. Zhiqiang Cui, Jinjin He, and Lupin Chen for many useful discussions and help.

Finally, I would like to express my deepest appreciation to my family: my uncle and aunt in New York City, my parents and my girlfriend in Wuhan, to whom this thesis is dedicated, for their constant encouragement, support and unconditional love.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

Figure Page

LIST OF FIGURES (Continued)

LIST OF TABLES

Table Page

Efficient VLSI Architectures for MIMO and Cryptography Systems

1 INTRODUCTION

1.1 MIMO Systems

Multiple-input multiple-output (MIMO) communication systems [1][7][18] have recently been considered as one of the most significant technology breakthroughs for modern wireless communications, due to the higher spectral efficiency and improved link reliability they can provide. MIMO techniques have been proposed as extensions to current wireless communication standards such as IEEE 802.11n and are part of the emerging standards such as IEEE 802.16. Therefore, the research in the MIMO systems is very attractive and useful for contemporary wireless communication industry.

1.1.1 MIMO System Model

It has been well studied in [17] that a multi-antenna array can be employed to obtain independent fading signals from a rich scattering multi-path channel, and the receiver can achieve processing gain by applying optimum ratio combining (ORC). This concept was extended in [1] by employing multi-antenna arrays at both ends of the communication link, thereby exciting independent paths between each of the transmit and receive elements.

Consider a symbol synchronized and uncoded MIMO system with *M* transmit antennas and *N* receive antennas. The baseband equivalent model for such MIMO system is

$$\tilde{\mathbf{y}} = \tilde{\mathbf{H}}\,\tilde{\mathbf{s}} + \tilde{\mathbf{n}}\ , \tag{1.1}$$

where $\tilde{\mathbf{s}} = [\tilde{s}_1\ \tilde{s}_2 \ldots \tilde{s}_M]^T$ is the *M* dimensional transmit signal vector, in which each component is independently drawn from a complex constellation such as QAM. Let $\tilde{\mathbf{y}}=[\tilde{y}_1\ \tilde{y}_2 \ldots \tilde{y}_N]^T$ denote the received symbol vector, and $\tilde{\mathbf{n}} = [\tilde{n}_1\ \tilde{n}_2 \ldots \tilde{n}_N]^T$ stands for an independent identical distributed (i.i.d.) complex zero-mean Gaussian noise vector with variance σ^2 per dimension. Moreover, assume a Rayleigh fading channel is represented by the $N \times M$ channel matrix $\tilde{\mathbf{H}}$, whose elements $\tilde{h}_{ij}$ represent the complex transfer function from the *j*-th transmit antenna to the *i*-th receive antenna, and are all i.i.d. complex zero-mean Gaussian variables with the variance of 0.5 per dimension. The channel matrix is assumed to be perfectly known to the receiver, and $M = N$ is assumed in this work.

The complex matrix equation (1.1) can be transformed to its real matrix representation

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}\ , \tag{1.2}$$

i.e.,
$$\begin{bmatrix} \mathrm{Re}(\tilde{\mathbf{y}}) \\ \mathrm{Im}(\tilde{\mathbf{y}}) \end{bmatrix} = \begin{bmatrix} \mathrm{Re}(\tilde{\mathbf{H}}) & -\mathrm{Im}(\tilde{\mathbf{H}}) \\ \mathrm{Im}(\tilde{\mathbf{H}}) & \mathrm{Re}(\tilde{\mathbf{H}}) \end{bmatrix} \begin{bmatrix} \mathrm{Re}(\tilde{\mathbf{s}}) \\ \mathrm{Im}(\tilde{\mathbf{s}}) \end{bmatrix} + \begin{bmatrix} \mathrm{Re}(\tilde{\mathbf{n}}) \\ \mathrm{Im}(\tilde{\mathbf{n}}) \end{bmatrix}, \tag{1.3}$$

where Re(·) and Im(·) denote the real and imaginary part, respectively. Since the element of $\tilde{\mathbf{H}}$ are assumed to be i.i.d. Gaussian, **H** has a full rank of $2M$.

The information theoretical capacity of the (*M, N*) MIMO channel is given by:

$$C = \log_2 \det[\mathbf{I}_N + \frac{\rho}{M}\widetilde{\mathbf{H}}\widetilde{\mathbf{H}}^H] \qquad \text{bits/s/Hz} \qquad (1.4)$$

In the above equation, ρ is the average signal-to-noise ratio (SNR) at each receive antenna, "det" means determinant, $\mathbf{I}_N$ is the identity matrix and $\widetilde{\mathbf{H}}^H$ means transpose conjugate. This equation assumes that the transmitter does not have any knowledge of the channel response, and hence distribute its power equally among the *M* antennas.

Such MIMO channel corresponds to the creation of multiple paths between the transmit and receive antennas. The relative power gains of each of these parallel channel are given by the eigenvalues λ_i of the channel covariance matrix $\widetilde{\mathbf{H}}\widetilde{\mathbf{H}}^H$. It is the creation of these parallel channels that gives rise to the high capacities of MIMO systems. Since all these 'spatial channels' are capable of supporting independent data streams, the overall capacity (suppose $N = M$) can therefore also be calculated as the sum of the classical Shannon capacities ($\log_2(1+SNR)$) of each spatial channel (modified by their individual channel gain) as:

$$C = \sum_{i=1}^{N} \log_2(1 + \frac{\rho}{N}\lambda_i) \qquad \text{bits/s/Hz,} \qquad (1.5)$$

which can be considered as linearly proportional to the antenna number *N*. Comparing with the capacity formula in [1] for optimum ratio combining or receive diversity

$$C = \log_2[1 + \rho \cdot \chi^2_{2N}] \qquad \text{bits/s/Hz,} \qquad (1.6)$$

where χ^2_{2N} denote a chi-square variant with 2*N* degrees of freedom, which is determined by the random channel matrix $\widetilde{\mathbf{H}}$, the advantage of MIMO system in spectrum efficiency is clearly demonstrated.

1.1.2 MIMO System Architecture

There are two types of MIMO signaling designed for different priorities such as high data-rate or high reliability under severe channel conditions.

1) MIMO with space-time coding (the signals transmitted from individual antennas are correlated/coded) for higher communication reliability.
2) MIMO with spatial multiplexing (the signals transmitted from individual antennas are independent from each other) for higher data rate.

1.1.3 MIMO System Detection Methods

For the detection of MIMO systems, we assume the receiver has acquired perfect information of the channel matrix $\widetilde{\mathbf{H}}$ (e.g., through a preceding training phase or inserting pilots signal and applying channel estimation). Algorithms used to separate the parallel data streams corresponding to the *M* transmit antennas can be divided into the following four categories:

1). *Zero-Forcing (ZF)* method is a suboptimal linear method based on finding the inverse of the channel matrix,

$$\mathbf{s}_{ZF} = (\mathbf{H}^{\mathrm{H}}\mathbf{H})^{-1}\mathbf{H}^{H} \cdot \mathbf{y} \tag{1.7}$$

and then slice the result onto the signal constellations. The detection method is simple. However, its performance is rather poor due to the noise and interference from other antennas.

2) *Minimum-Mean-Square-Error (MMSE)* is another suboptimal linear method which is similar to zero-forcing. However, in this method, noise term has been taken into account:

$$\mathbf{s}_{MMSE} = (\frac{\mathbf{I}_{2N}}{SNR} + \mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H \cdot \mathbf{y} \tag{1.8}$$

It has intermediate complexity, but requires an accurate estimate of the noise level present in the system, which is normally hard to obtain in a practical system.

3) *Ordered Successive Interference Cancellation (OSIC)* decoder such as the V-BLAST algorithm is an iterative application of zero-forcing or MMSE, effectively implementing iterative interference cancellation. It shows better performance, but suffers from error propagation and is still suboptimal. It has five main steps: 1. Ordering--choosing the best channel, 2. Nulling--using ZF or MMSE, 3. Slicing--making a symbol decision, 4. Cancelling--subtracting the detected symbol, and 5. Iteration--going to the first step to detect the next symbol.

4) *Maximum Likelihood (ML)* detection, which solves

$$\mathbf{s}_{ML} = \arg\min_{\mathbf{s}\in\Lambda} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \tag{1.9}$$

where Λ is the lattice defined by having each entry of the 2N dimensional vector $\mathbf{s}$ be taken from the signal constellation, is always the optimum detection method and minimizes the bit-error-rate (BER). The ML detection can be conducted via two approaches. A straightforward approach to solve

equation 1.9 is an exhaustive search. Unfortunately, the corresponding computational complexity grows exponentially with the transmission antenna numbers and constellation sizes. For example, in a 4x4 system with 16-QAM modulation, 65536 candidate symbols have to be considered for each received vector. A better approch is the sphere decoding method, which will be regarded as a new method for MIMO detection.

5) *Sphere Decoding (SD)* is a reduced complexity algorithm which implements the ML detection for MIMO system while avoids the unmanageable complexity of exhaustive search. The main idea is to reduce the search range from the whole finite lattice space to the lattice within a hypersphere so as to find out the ML solution for the MIMO system. Mainly it can be categorized into hard-decision sphere decoding and soft-decision sphere decoding. Moreover, depending on the search method among the constellation tree, it can be catagorized into depth-first search (regular sphere decoding) and breadth-first search (K-Best sphere decoding). Both of them are applied to real hardware implementations, and will be discussed later in the details.

Nowadays, the sphere decoding algorithm has been widely used for maximum likelihood detection in MIMO systems. However, conventional SDA is very complex for hardware implementations, and the throughputs of current SDA designs are generally below the requirement of next generation high-speed wireless communications.

The first part of this research is focused on the low-complexity and high-speed VLSI architecture of sphere decoder designs which intends to achieve the ML detection for the MIMO wireless systems. It includes the contents from Chapter 2 to Chapter 6, where we propose the layer-reordering SDA, efficient sorting architecure for K-Best SDA, early-pruning scheme for K-Best SDA, parallel and pipeline interleaved SD, and efficient radius and list update units design for list sphere decoders.

1.2 Cryptography

The wireless communication technologies, to which MIMO system belongs, are widely used today by the business organizations, governments, militaries, and civil residents, because they can offer many benefits such as the portability, flexibility, increased productivity and lower installation and maintenance costs. Wireless technologies cover a broad range of different capabilities oriented toward different uses and needs. For instance, the wireless LAN devices allow users to move their computers from place to place within the office or home without the need for wires and without losing network connectivity. Less wiring means greater flexibility, increased efficiency and reduced wiring costs. Bluetooth functionality also eliminates cables for printer and other peripheral device connections. The handheld devices such as PDA and cellular phones allow remote users to exchange voice information and access to the network service such as wireless email and web browsing.

However, the risks are inherent in any wireless technology for the reason that the technology's underlying communication medium, the airwave, is shared and

open to the public, including the intruders and eavesdroppers. Therefore, the security of the wireless communication is extremely important to protect the private the sensitive information.

Cryptography is the most standard and efficient way to protect the securities. It can be used to protect the confidentiality, integrity, authentication, and non-repudiation. There are two major categories of cryptography schemes, i.e., symmetric key cryptography and asymmetric key cryptography.

1.2.1 Symmetric Key Cryptography

The basic encryption/decryption scheme of symmetric key cryptography is shown in Figure 1.1 [44][55]. In Figure 1.1, plaintext is the original form of the message that sender wants to send to the recipient. Ciphertext is the encrypted form of the original message which can be transmitted in an insecure channel such as wireless media. The sender and the recipient use the same secret key for the encryption and decryption function. Therefore, it is named symmetric key cryptography.

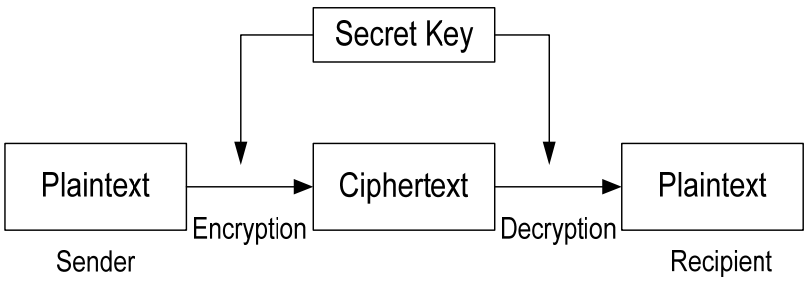

Figure 0.1. Symmetric key encryption / decryption scheme.

In symmetric key cryptography, the receiver and sender must share the same private key, which needs to be pre-distributed safely. Such scheme requires extra key distribution and considerable management cost which is not as convenient as the asymmetric key cryptography.

1.2.2 Asymmetric Key Cryptography

The basic encryption / decryption scheme of the asymmetric key cryptography (also known as public key cryptography) is shown in Figure 1.2 [44][55]. The sender uses recipient's public key for encryption. The recipient can decrypt the ciphertext using his own private key. In symmetric key cryptography, each pair of sender and recipient share a secret key, whereas in public key cryptography, only the sender's public key is broadcasted to the public, and multiple senders can use the same public key for encryption and transfer data to the same recipient.

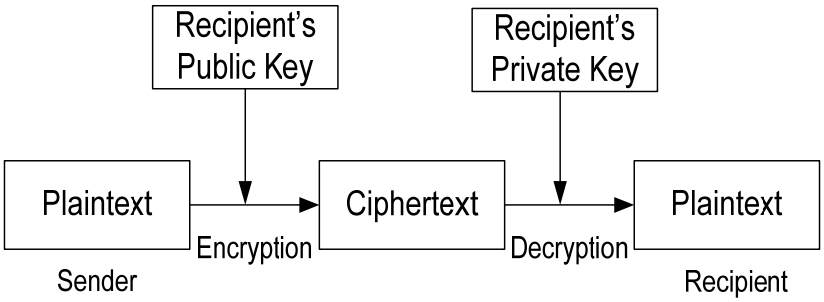

Figure 0.2. Asymmetric key encryption / decryption scheme.

Public key cryptography is easy for key distribution and key management. A well-known public-key cryptography algorithm is RSA, which was first introduced

by Rivest, Shamir and Adleman in 1977 [45]. The security of RSA is based on hardness of integer factorization problem. The RSA is commonly used in today's security systems.

1.2.3 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is an efficient substitution for RSA. It was originally proposed by Victor Miller at IBM [46] and Neal Koblitz from the University of Washington [47] independently. The security of ECC is based on the hardness of solving the elliptic curve discrete logarithm problem (ECDLP). Comparing with the sub-exponential time it takes to solve the integer factorization problem, it takes fully exponential time for today's best algorithm to solve ECDLP. Compared with RSA, ECC has much smaller key length yet still provides the same security level. Smaller key length results in faster computation, lower power consumption, and lower memory / storage usage. Table 1.1 [55] shows the equivalent key sizes of ECC and RSA [48]. Currently, 1024-bit RSA is standard, and it is projected that its size will increase to 2048 bits after 2010. Such large key size will severely affect the cost of RSA implementation; therefore, ECC becomes a long-term trend which will substitute RSA.

TABLE 0.1 EQUIVALENT KEY SIZES BETWEEN ECC AND RSA

ECC	RSA	Protection Lifetime
163	1024	until 2010
283	3072	until 2030
409	7680	beyond 2030

1.2.4 Tate Pairing

Identity based cryptography (IBC) schemes [64, 65] have recently opened a new territory for public key cryptography. Using the identity based cryptography scheme, a sender can derive the public key of a receiver without receiving the receiver's certificate issued by a certificate authority (CA). The public key can be directly derived from the identity of the receiver such as the email address or IP address. The pairing over the elliptic curve is used to construct the identity based cryptography schemes. It is a mapping from two points on the elliptic curve to another multiplicative group. It has special properties of bilinearity. Currently, the most commonly used pairing methods are Tate pairing [66] and Weil paring [80]. Weil pairing was originally used to attack public key cryptosystems and later was used for pairing based cryptosystems. It can be computed using either Miller algorithm [71] or modified Miller's algorithms [75, 77].

Tate pairing is more efficient than Weil pairing because it requires only one iteration of Miller's algorithm instead of two for Weil pairing. Also, it is more than two times faster than Weil pairing. Currently, Tate pairing is the most popular method which is used in many identity based cryptography schemes [64, 65].

The best method of Tate pairing calculation before 2002 was presented by Miller in [71]. In 2002, Galbraith [74] and Barreto [75] greatly simplified the pairing computation by introducing the triple-and-add BLKS algorithm in characteristic three. The BLKS algorithm was further modified and developed as the Duursma-Lee algorithm [69] and the Kwon-BGOS algorithm [70].

The second part of this thesis is focused on the high-speed and efficient architecture for the cryptography systems.

Due to the advantages of ECC over RSA, it is necessary to develop the high-speed ECC architecture for hardware implementations. The implementation of ECC mainly relies on the operations at three levels: the scalar multiplication, the point addition / doubling, and the finite field modulo arithmetic. The projective coordinate [50][51][52] is more widely used for point operation because it avoids the costly field inversion operation.

In Chapter 7, we introduce an efficient fast architecture for the Lopez-Dahab projective coordinates [49]. By applying parallel processing and hardware reusing, the point addition and doubling operations can be significantly accelerated with reasonable hardware overhead compared with the conventional point operation implementations.

Prior implementations of the Tate paring are mainly in software domain [67]. These implementations can only run at low speed due to the high complexity. In order to boost the speed of IBC to practical level, efficient and high-speed hardware implementations of Tate paring need to be explored.

In Chapter 8, through exploring the intrinsic property of the Duursma-Lee algorithm, we propose complexity-reducing schemes and an overlapped processing architecture. Compared with conventional sequential implementations [68], the proposed architecture can achieve over 2 times speedup. The proposed method can be also applied to the Kwon-BGOS algorithm, and similar speedup can be obtained.

1.3 Summary of Contributions

The main contributions in this dissertation are summarized as follows:

1.3.1 Improved K-Best Sphere Decoding Algorithms

The Sphere Decoding Algorithm (SDA) has been used for Maximum Likelihood (ML) detection in MIMO systems. However, regular SDAs have a disadvantage that the computational complexity varies with different signals and channels. Hence the detection throughput is non-fixed, which is not desirable for real time detection and hardware implementations. For this reason, the K-Best sphere decoding algorithm is introduced in [5] [6]. Instead of doing depth-first search, the K-Best SDA uses breadth-first search. At each search layer, only the best K candidates are kept for the next level search. The K-Best SDA requires less computational complexity, has fixed throughput, and is suitable for pipelined hardware implementation.

In Chapter 2 and [16], we first applied the layer reordering method (sorted QR decomposition) to the K-Best SDA. Hence, we can achieve the same performance with a smaller K than usual and thus reduce complexity. We then introduced the dynamic K-Best SDA, which can also reduce complexity by applying different K values at each layer. We pointed out that such a dynamic K-Best SDA can be combined with the layer reordering method mentioned above to obtain more complexity savings.

Simulation results show that by applying sorted QR decomposition for the channel matrix, and/or introducing dynamic K values for different layers, our improved algorithms can achieve about 30% complexity reduction for 4x4 64QAM

MIMO systems over the traditional K-Best SDA without introducing extra computational complexity.

1.3.2 Reduced Complexity K-Best Sphere Decoder Scheme and Sorting Architecture

This part of work is also conducted based on the K-Best sphere decoder design. The K-Best SDA requires less computational complexity, has fixed throughput, and is suitable for pipelined hardware implementation. Most of the K-Best SDA computational complexity lies in the path extension and the sorting operations (choosing K Best paths among $K \cdot Mc$ paths). Moreover, the sorting part is more computation intensive when K is large. Therefore, for hardware implementation, it is critical to reduce the sorting complexity.

Our contributions in Chapter 3 and [22] are as follows: 1) Introduced a reduced complexity K-Best SDA based on SE strategy. In our decoder design, only partial path extension needs to be done. Simulations show that for 4x4 64QAM system, we can save 25% path cost computation and 27% sorting operations with almost no performance loss. 2) By exploiting the natural partial sorted results coming from the SE method, we derived a sorting architecture which applied rank order filters (Batcher's merge sort algorithm). Such sorting architecture exploits the natural partial order from SE enumeration, and can significantly reduce the sorting complexity (around 50%) comparing with bubble sorting algorithm, which is a significant contribution to the K-Best SDA implementation for MIMO systems.

The improved sphere decoding algorithms discussed in Chapter 2 can be used to reduce the decoder complexity, i.e., to achieve the same performance, a smaller

K value can be used. Moreover, they can be combined with the sorting architecture to further reduce the computational complexity. We have provided the simulation results showing these three methods can be combined together to achieve the same detection performance as regular K-Best SDA with much smaller K values. Also, a comprehensive complexity analysis has been presented [31] to demonstrate that even regardless of the memory access time and area savings, our proposed sphere decoding algorithm and sorting architecture can achieve a total complexity saving of 68%.

1.3.3 Parallel and Pipeline Interleaved Sphere Decoder Architecture

The SDA is very complex for hardware implementation. To the best of our knowledge, the sphere decoder designs published in the literature have lower throughput than 180Mb/s, which is below the requirement of next generation high-rate wireless communication systems (over 200Mb/s). Therefore, efficient high-speed architectures for sphere decoder implementation are really desirable.

In Chapter 4 and [26], we first proposed a parallel sphere decoding scheme. In this method, the whole constellation tree is divided into two sub-trees, and the two processing engines (PE) can conduct depth-first search in parallel and update the new radius. Thus the decoding throughput is significantly improved. Considering the parallel architecture needs to double the hardware cost, we further introduced the pipeline interleaved SD architecture. For this architecture, by exploiting the similarity and interleaving the data streams for both processing engines, only one PE is needed with some small interleave control logics. The new

sphere decoder has almost the same hardware cost as conventional SD with 44% improvement of the throughput.

1.3.4 Early-Pruning K-Best Sphere Decoder

The sphere decoding algorithm has been used for maximum likelihood detection in MIMO systems, and the K-Best sphere decoding algorithm is proposed for MIMO detections for its fixed complexity and throughput. However, to achieve near-ML performance, the *K* needs to be sufficiently large, which leads to large computational complexity and power consumption in path expansion, sorting, and path updating.

In Chapter 5 and [37], we introduced some dynamic early-pruning schemes, which will eliminate the survival candidates with relatively large partial Euclidian distances (PEDs) at early stages. These candidates are unlikely to become the ML solution when the tree searching reaches the final layer. Therefore, such early pruning can save computation and power consumption without sacrificing the performance. Our simulation results show that for the 4x4 64QAM MIMO system, by applying the proposed schemes, about 55% computational complexity can be reduced with almost no performance degradation.

1.3.5 Efficient Radius and List Updating Units Design for List Sphere Decoders

The sphere decoder (SD) has been utilized for maximum likelihood (ML) detection in MIMO systems. In order to improve system performance, the SD is usually combined with the error correction codes where soft decoding is utilized. The list sphere decoder (LSD) was introduced to generate a candidate list, which

can provide the soft information to the outer decoder. Unlike the conventional sphere decoder, the LSD has the candidate list updating and new radius generation units, which causes extra complexity and latency.

In Chapter 6 and [42], we present a novel radius updating architecture, which can obtain the new radius much faster than the conventional method. Furthermore, we propose an efficient candidate list updating scheme, which can significantly save the complexity (without affecting the decoding speed) of updating the candidate list used to compute the soft information.

1.3.6 Fast Point Operation Architecture for Elliptic Curve Cryptography

The ECC has higher security strength per bit over RSA, which can offer potential reduction in storage space, bandwidth and power consumptions. Hence, the high-speed ECC architecture for hardware implementations becomes necessary, especially for the scenarios where high speed communications are required. The implementation of ECC mainly relies on the operations at three levels: the scalar multiplication, the point addition / doubling, and the finite field modulo arithmetic. The projective coordinate is more widely used for point operation because it avoids the costly field inversion operation.

In Chapter 7 and [61], we introduced an efficient fast architecture for the ECC based on Lopez-Dahab projective coordinate. By applying parallel processing and hardware reusing, the point addition and doubling operations can be significantly accelerated compared with the conventional point operation

implementations. Analysis shows that, with reasonable hardware overhead, our architecture can achieve a speedup of 2.5 times for the point addition operation in Lopez-Dahab projective coordinate and 1.8 times for the point doubling operation, which facilitates the design of high-speed ECC systems.

1.3.7 Efficient Architecture for the Tate Pairing in Characteristic Three

Due to the high complexity of the Tate pairing operation, prior implementations of the Tate paring are mainly in software domain and very few previous efforts have been devoted to hardware implementation. These implementations can only run at low speed due to the high algorithm complexity. In order to boost the speed of IBC to practical level, efficient and high-speed hardware implementations of Tate paring need to be explored.

In Chapter 8 and [81], we proposed complexity-reducing schemes and an overlapped processing architecture. Without introducing extra hardware complexity, compared with conventional sequential implementations, the proposed architecture can achieve over 2 times speedup, which is a big improvement for the Tate pairing implementation. The proposed method can be also applied to the Kwon-BGOS algorithm, and similar speedup can be obtained.

2 IMPROVED K-BEST SPHERE DECODING ALGORITHMS FOR MIMO SYSTEMS

Multiple-input multiple-output (MIMO) systems have attracted considerable research attentions in the wireless communication area recently. It has been shown in [1] that extraordinary spectral efficiency near Shannon limit can be achieved in MIMO systems. However, to achieve optimal maximum-likelihood (ML) detection, the computational complexity becomes huge when higher modulation constellations are applied, and it increases exponentially with antenna numbers. Therefore, the sphere decoding algorithm (SDA) has been introduced in [2-4] to drastically reduce detection complexity for MIMO systems. The sphere decoder can be regarded as a depth-first tree search approach with pruning. The SDAs for MIMO system have two types of searching strategies, i.e., the Fincke-Phost (FP) method proposed in [2][3] and the Schnorr-Euchner (SE) strategy introduced in [4]. The second method has less computational complexity by re-ordering the constellation searching at each layer.

Regular SDAs have a disadvantage that the computational complexity varies with different signals and channels. Hence the detection throughput is non-fixed, which is not desirable for real time detection and hardware implementation. To resolve this issue, the K-Best sphere decoding algorithm was introduced in [5] [6]. Instead of doing depth-first search, the K-Best SDA uses breadth-first search. At each search layer, only the best K candidates are kept for the next level search. The K-Best SDA has fixed complexity and throughput, and is suitable for pipelined hardware implementation. The drawbacks of the K-Best SDA are 1) it generally

has performance degradation as the ML solution cannot be guaranteed by keeping the K best candidates during each layer's search unless K is sufficiently large. 2) the sorting operations (choosing K Best paths among $K \cdot Mc$ paths, Mc is the constellation size) account for the major complexity of the K-Best SDA, especially when K is large.

Our contributions in this Chapter include: 1. Apply the layer reordering method (sorted QR decomposition) to the K-Best SDA. Hence, we can achieve the same performance with a smaller K than usual and thus reduce complexity. 2. Introduce the dynamic K-Best SDA, which can also reduce complexity by applying different K values at each layer. Such a dynamic K-Best SDA can be combined with the layer reordering method mentioned above to obtain more complexity savings.

2.1 Sphere Decoding Algorithms

Based on the system model above, the set $\{\mathbf{Hs}\}$ can be considered as the lattice $\Lambda(\mathbf{H})$ generated by $\mathbf{H}$. If the received vector $\mathbf{y}$ is considered as a perturbed lattice point due to the Gaussian noise $\mathbf{n}$, the maximum-likelihood MIMO detection is to find the closest lattice point $\mathbf{s}_{\mathrm{ML}}$ for a given lattice $\Lambda(\mathbf{H})$, i.e.,

$$\mathbf{s}_{\mathrm{ML}} = \arg \min_{\mathbf{s} \in \Omega} \left\| \mathbf{y} - \mathbf{Hs} \right\|^2, \tag{2.1}$$

where Ω is the set of real entries in the constellation, e.g., $\Omega = \{\pm 1, \pm 3, \pm 5, \pm 7\}$ for 64-QAM. Also let M_c denote the one dimensional constellation size (here M_c=8).

2.1.1 The Sphere Decoding Algorithm

Equation (2.1) can be re-written as:

$$\mathbf{s}_{\mathrm{ML}} = \arg\min_{\mathbf{s}\in\Omega} (\mathbf{s}-\hat{\mathbf{s}})^T \mathbf{H}^T \mathbf{H} (\mathbf{s}-\hat{\mathbf{s}}) = \arg\min_{\mathbf{s}\in\Omega} \bar{\mathbf{s}}^T \mathbf{R}^T \mathbf{R} \bar{\mathbf{s}}, \tag{2.2}$$

where $\mathbf{R}$ is the upper triangular matrix with non-negative diagonal element such that $\mathbf{R}^T\mathbf{R} = \mathbf{H}^T\mathbf{H}$ ($\mathbf{R}$ can be obtained by applying QR decomposition to $\mathbf{H}$), $\hat{\mathbf{s}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$ is the zero-forcing (ZF) solution of $\mathbf{s}$. $\bar{\mathbf{s}} = \mathbf{s} - \hat{\mathbf{s}}$ is the distance from signal candidate to ZF solution. The sphere decoder avoids an exhaustive search by examining only the lattice points falling inside a hyper-sphere $\bar{\mathbf{s}}^T\mathbf{R}^T\mathbf{R}\bar{\mathbf{s}} < r^2$, with the radius r large enough to contain the ML solution. Due to the triangular shape of $\mathbf{R}$, (2.2) can be written as an iterative, monotonically increasing form:

$$\mathbf{s}_{\mathrm{ML}} = \arg\min_{\mathbf{s}\in\Omega} \bar{\mathbf{s}}^T \mathbf{R}^T \mathbf{R} \bar{\mathbf{s}} = \arg\min_{\mathbf{s}\in\Omega} \sum_{i=M}^{1} \left| \sum_{j=i}^{M} r_{ij} (\hat{s}_j - s_j) \right|^2, \tag{2.3}$$

where r_{ij} are the elements of upper triangle matrix $\mathbf{R}$. We define the branch cost function associated with nodes in the i-th layer as

$$e_i(\mathbf{s}^i) = \left| \sum_{j=i}^{M} r_{ij}(\hat{s}_j - s_j) \right|^2 = \left| \sum_{j=i}^{M} r_{ij}\hat{s}_j - \sum_{j=i+1}^{M} r_{ij}s_j - r_{ii}s_i \right|^2 = \left| b_i(\mathbf{s}^i) - r_{ii}s_i \right|^2, \tag{2.4}$$

where $\mathbf{s}^i$ is the partial vector of $\mathbf{s}$ with $s_1 = 0, s_2 = 0, \ldots, s_{i-1} = 0$. $\rho_i(\mathbf{s}^i) = \sum_{j=i}^{M} r_{ij}\hat{s}_j$,

$$b_i(\mathbf{s}^i) = \rho_i(\mathbf{s}^i) - \sum_{j=i+1}^{M} r_{ij}s_j, \tag{2.5}$$

and

$$T_k(\mathbf{s}^i) = \sum_{i=M}^{k} e_i(\mathbf{s}^i) \tag{2.6}$$

is the partial sum of $e_i(\mathbf{s}^i)$ (i.e., partial Euclidean distance (PED)) which is non-decreasing. The decoding process can be regarded as descending down in a tree in which each node has M_c branches. If a PED exceeds r^2, the entire branch and all its descendents are pruned.

2.1.2 SE Enumeration

The basic principle of Schnorr-Euchner (SE) strategy was introduced in [4]. When the sphere decoder descends in the tree, for each partial vector, it examines each possible child symbol/node in the constellation. It has been shown in [13] that enumerating these symbols in an ascending order according to their distance to the Babai point will expedite the tree search. Such enumeration ensures that if a node does not obey the sphere constraint, the following nodes will not satisfy the constraint either, and can all be pruned.

In [14], a look up table is suggested to implement SE enumeration to avoid sorting branch cost functions. It is efficient and has been used in our K-Best SDA.

2.1.3 The K-Best Sphere Decoding Algorithm with SE strategy

The search in the tree can also be conducted in a breadth-first approach. Instead of expanding every node at each layer, we only keep K nodes, which have the smallest accumulated PEDs. Finally, we will reach K leaves with smallest PEDs. Each leaf's path corresponds to a signal vector s. The decoder regards the s with the smallest PED as the detection result. After our modification, the mathematical description of K-Best SDA is as following:

1. **Preprocessing**: compute $\mathbf{H}^{-1}$, QR decomposition **H=QR**

2. **SE enumeration:** $i=M$, enumerate each $\mathbf{s}^M$ among the constellation using the look up table in [14].
3. **Path expansion-1:** calculate the branch cost $e_M(\mathbf{s}^M)$ for each $\mathbf{s}^M$ according to (2.4). Totally M_c branch costs obtained. Based on (2.3), for $i=M$, PED $T_M(\mathbf{s}^M) = e_M(\mathbf{s}^M)$.
4. **Find K partial vectors**: Sort the M_c PED and find the smallest K partial vector $\mathbf{s}_k^M$
5. **Survival path update**: update $\rho_i(\mathbf{s}^i)$ and $b_i(\mathbf{s}^i)$ in (2.5)
6. **Path expansion-2***: i=i-1*. For each surviving partial vector $\mathbf{s}_k^{i+1}$ from the last layer, $(1 \le k \le K)$, enumerate $s_{k,i}$ among the constellation using look up table, $s_{k,i}$ is the i-th element of $\mathbf{s}_k$. Calculate the branch cost $e_i(\mathbf{s}_k^i)$ for each $s_{k,i}$. Compute PEDs $T_i(\mathbf{s}_k^i) = T_{i+1}(\mathbf{s}_k^{i+1}) + e_i(\mathbf{s}_k^i)$.
7. **Sorting**: Sort the KM_c PEDs. Select K partial vectors $\mathbf{s}_k^i$ which have the smallest PEDs among the KM_c.
8. **Path update**: update $b_i(\mathbf{s}^i)$ and go to step 6.
9. **Check termination condition**: if $i=1$, output the vector $\mathbf{s}$ with smallest cost

2.2 Layer Reordered K-Best SDA

The K-Best SDA has constant throughput and is desirable for pipelined hardware implementations. However, it is sub-optimal compared with ML detection, and has performance loss in general. Before introducing our layer reordered K-Best SDA, let us analyze the reason that causes such performance

degradation. Assume we have two candidate symbols $\mathbf{s}_1$ and $\mathbf{s}_2$, both are *M*-dimensional vectors. For the above MIMO model, the total cost functions are $T_1(\mathbf{s}_1) = \sum_{i=M}^{1} e_i(\mathbf{s}_1)$ and $T_1(\mathbf{s}_2) = \sum_{i=M}^{1} e_i(\mathbf{s}_2)$, respectively. Suppose $\mathbf{s}_1$ is the ML solution, then we have $T_1(\mathbf{s}_1) < T_1(\mathbf{s}_2)$. The K-Best SDA should select $\mathbf{s}_1$ as the candidate instead of choosing $\mathbf{s}_2$. However, the K-Best SDA is making decision based on PEDs $T_i(\mathbf{s}_1^i)$ and $T_i(\mathbf{s}_2^i)$, $i = M, M-1, \cdots, 1$. If at some early stage *i*, $T_i(\mathbf{s}_1^i)$ is not among the K smallest PED (although the total sum of $e_i(\mathbf{s}_1^i)$ is minimum, its partial sum is not always minimum), and candidate $\mathbf{s}_1^i$ will be discarded. In other words, even though we select the K-Best PED at early layers, the excluded PEDs are still possible to become the minimum PED at final layer after accumulating the cost metrics of the remaining stages. Thus the errors at early layer will propagate and make the decoder miss the ML solution.

Normally, to obtain near-ML performance, a large K value is used for sphere decoding, and this will introduce large complexity including the PEDs computation, sorting, and path updating. Our approach here is to introduce some schemes which can significantly improve the detection performance even using smaller K values; therefore, the complexity of the whole sphere decoder can be reduced by avoiding using large K values.

To improve the K-Best SDA performance for small K, we intend to reduce the possibility that the SDA excludes the ML solution at early stages. One approach is reordering the layer. The idea is to permute the columns of channel matrix **H**. Therefore, the order of the elements of vector **s** to be decoded by the sphere

decoder is altered accordingly. In this way, the PEDs of different vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ have been re-distributed, while maintaining the total cost (i.e., $T_1(\mathbf{s}_1)$ and $T_1(\mathbf{s}_2)$ remains the same) [16]. Hence the decoding at early stages has been changed. If we can find such reordering schemes that reduce the possibility of missing ML solution at early stage, the performance can be improved.

Supposing some layer reordering can re-distribute the PEDs such that the differences of PEDs of vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ ($\left|T_i(\mathbf{s}_1^i) - T_i(\mathbf{s}_2^i)\right|$) at early layer are enlarged, we can claim the K-Best decision at layer *i* is more reliable than the decision before reordering. The reason is that if $T_i(\mathbf{s}_1^i) < T_i(\mathbf{s}_2^i)$ and the difference is enlarged, it is less likely that after accumulating the cost metrics of the remaining layers, $T_1(\mathbf{s}_1) > T_1(\mathbf{s}_2)$ (the less likely the remaining cost can change the early order). Hence, the K-Best candidates at early layers are more likely to be the real K-Best solutions. And such reordering approach may improve the detection performance.

From (2.6) the difference between PEDs is the partial sum of the difference between the branch cost function $e_i(\mathbf{s})$. Hence, increasing the difference of $e_i(\mathbf{s})$ is a good approach. Notice from (2.4), if by reordering the layer we can put larger r_{ii} for early layers (*i* is large) and smaller r_{ii} for lower layers, the difference of PEDs at early layers are increased, thus SDA performance can be improved.

In [7] a sorted QR decomposition method was introduced. The idea is to find the permutation of **H** that minimizes each $|r_{ii}|$ with *i* running from 1 to *M*. Therefore it intends to maximize diagonal elements $\left|r_{jj}\right|$ in the succeeding step $j > i$. For details, readers can refer [7].

Wubben [7] only applied this method to decode layered space time codes by using successive cancelling. Such reordering method can be combined with QR decomposition. It introduces negligible extra complexity. In our research, we found such sorted-QR decomposition method can also be applied to sphere decoding. Our approach is to apply this method to the K-Best SDA. We perform the reordering when decomposing H, and after decoding, we can permute the detected vector s reversely to recover the original order. It should be noticed that the extra computation complexity for the new method is negligible compared with the traditional K-Best SDA.

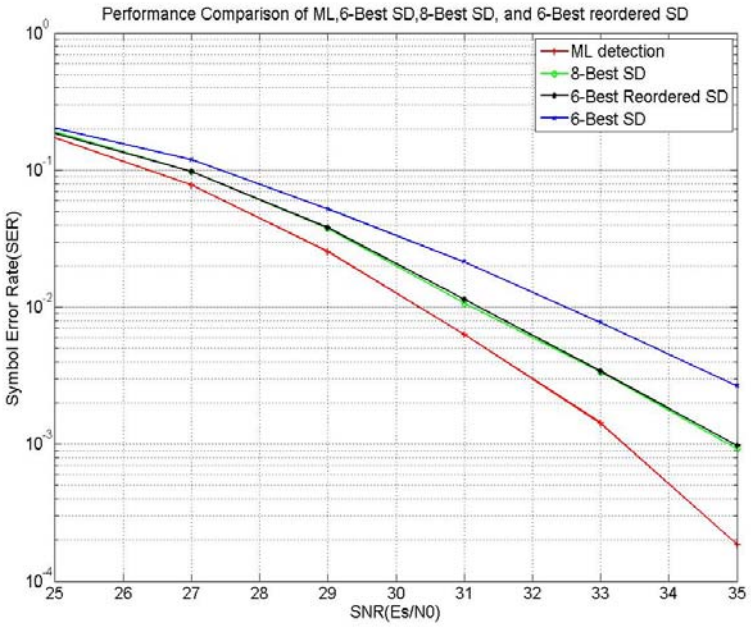

Figure 2.1. Performance comparisons of ML, 6-Best, 8-Best and 6-Best reordered SD (N=M=4, 64QAM).

Figure 2.1 shows some simulation results by applying the layer reordered K-Best SDA. The MIMO system used 4x4 antennas, the constellation is 64QAM. By

decoupling the complex constellations, the real model used is an 8x8 8PAM MIMO system. Figure 2.1 compares the performance (symbol error rate) of the ML detection, the normal K-best SDA (K=8 and K=6), and the layer reordered K-Best SDA (K=6) at different SNRs (E_s / N_0). We can see the traditional 8-Best SDA (at SNR=32dB) has about 1dB performance loss compared with ML detection, and is 1.8dB better than the 6-best SDA. By applying our reordering scheme, the performance of our re-ordered 6-Best SDA has almost the same performance as the conventional 8-Best SDA, which means it brings us about 1.8dB performance gain.

Therefore, we can use 6-Best re-ordered SDA to replace the normal 8-Best SDA. According to the algorithm in Section 2.1.3, for each surviving candidate, we only need to compute the first 6 PEDs among its 8 child nodes. And we need to sort out the 6 survivors with the smallest PEDs out of 6x6=36 candidates compared to sorting 8 out of 8x8=64. Afterwards, the path update effort is also reduced from 8 to 6. The only extra complexity is after decoding, we need to permute the detected vector s reversely to recover the original order, which is negligible. Hence, with the same performance, the path expansion and path updating complexity can be reduced by around 25%, and the sorting complexity was reduced by 60% (for bubble sort, 35+34+33+32+31+30=195, 63+62+61+60+59+58+57+ 56=476, 195/476=40.96%).

Such layer reordered K-Best SDA can be applied to any MIMO K-Best SDA with better performance and negligible complexity.

2.3 Dynamic K-Best SDA

Based on the discussion in section 2.2, the approach to improve the K-Best SDA performance for small K values is to reduce the possibility of excluding ML solution at early stages. A useful method is to change the K value (dynamic K) at different decoding layers.

The idea is, at the early stages, to use larger K values to ensure the ML solution is included in the K-Best candidates. The reason is that at the early stage i (i is large), there are $i-1$ layers left. Therefore, the partial Euclidean distance has another $i-1$ branch cost metrics to accumulate before reaching the final total cost. It is more likely to miss the ML solution at early layers. Increasing K here can reduce such possibilities. As the decoder descends in the tree (searching lower layers), the PED is close to the final result. Hence it is less likely to miss the ML solution in the K-Best candidates. As a result, we can reduce the K value at later stage to reduce complexity while maintaining performance.

There is not a fixed law regarding how to dynamically adapt K values at different layers. They are determined by extensive simulations. For our simulations, we use 4x4 64QAM MIMO systems. After constellation decoupling, the resulting system is an 8x8 8PAM system. Here we use $\mathbf{K} = [8\ 9\ 8\ 7\ 6\ 5\ 4\ 3]$ at different layers, from first layer to the last layer. The simulation result is shown in Fig. 2.2.

From the simulation result shown in Figure 2.2, it can be seen that applying dynamic K-Best SDA can obtain much better performance than original 6-Best SDA (about 2dB improvement). The result is even better than the layer-reordered 6-Best SDA, and regular 8-Best SDA. Therefore, such dynamic K-Best scheme can

be used to replace the original 8-Best SDA design with much less complexity and better performance.

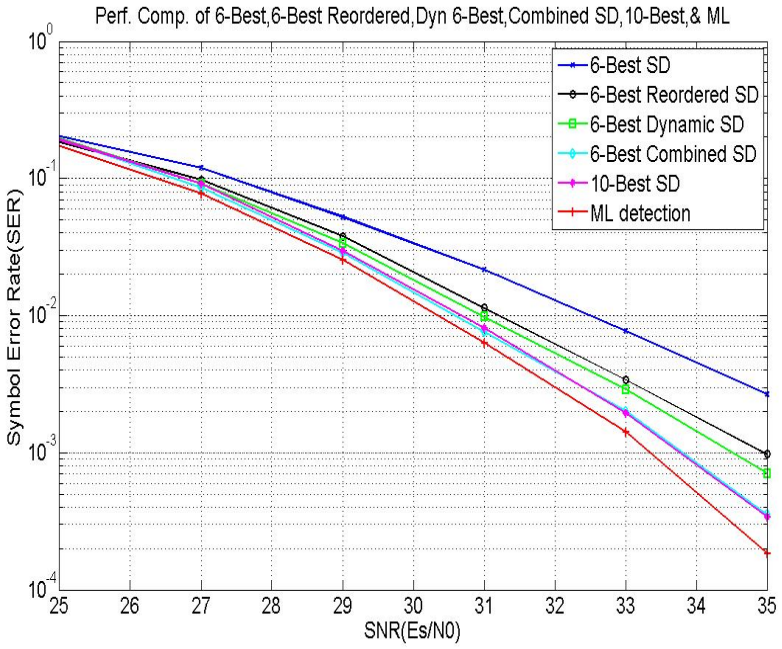

Figure 2.2. Performance comparisons for 6-Best, 6-Best reordered, dynamic K-Best, combined dynamic reordered, 10-Best SD and ML (N=M=4, 64QAM).

Moreover, the dynamic K-Best can be combined together with the reordered K-Best SDA to obtain even better performance. In the simulation results shown in Figure 2.2, it can be noticed that the combined dynamic-6 & reordered SDA can achieve almost the same performance as the regular 10-Best SDA, which is already very close to the ML detection. However, the complexity is much less than the normal 10-Best (see Section 3.4 for detailed complexity analysis). This result has enlighten us that for large complexity MIMO systems it is possible to apply such combined sphere decoding algorithm to considerably reduce the computational complexity while maintaining the detection performance.

According to the above analysis and simulation results, this novel dynamic K-Best method can obtain better performance or decrease computational complexity for hard decision sphere decoding. Moreover, for $\mathbf{K} = [8\ 9\ 8\ 7\ 6\ 5\ 4\ 3]$, the K value is changing at each layer. Such irregularity may be not desirable for hardware implementations. In the real hardware design, we can use $\mathbf{K} = [8\ 8\ 8\ 6\ 6\ 6\ 4\ 4]$, which has almost the same performance but more regularity.

2.4 Conclusions

We have introduced some improved K-Best sphere decoding algorithms, which include layer reordered K-Best SDA, dynamic K-Best SDA, and the combined K-Best SDA. All these algorithms can improve the detection performance, or reduce the computational complexity compared with the traditional K-Best SDA. Among these algorithms, the reordered K-Best SDA is most promising for its negligible extra complexity and flexibility to combine with any other K-Best sphere decoding algorithms.

3 REDUCED COMPLEXITY K-BEST SPHERE DEOCOER SCHEME AND SORTING ARCHITECTURE

The Sphere Decoding Algorithm (SDA) has been used for achieving maximum likelihood (ML) detection for today's Multiple-Input Multiple-Output (MIMO) systems. Regular SDAs have a disadvantage that the computational complexity varies with different signal constellations and channels. Hence the detection throughput is non-fixed, which is not desirable for real time detection and hardware implementations. To resolve this issue, the K-Best sphere decoding algorithm was introduced in [5] [6]. Instead of doing depth-first search, the K-Best SDA uses breadth-first search. At each search layer, only the best K candidates are kept for the next level search. The K-Best SDA requires less computational complexity, has fixed throughput, and is suitable for pipelined hardware implementation. Most of the K-Best SDA computational complexity lies in the path extension and the sorting operations (choosing K Best paths among $K \cdot Mc$ paths). Moreover, the sorting part is more computational intensive when K is large. Therefore, for hardware implementation, it is critical to reduce the sorting complexity. The basic SE SDA architecture was introduced in [4][9]. In [12], the SDA complexity can be reduced by applying a probabilistic search approach and error-performance-oriented fast stopping criterion.

Our contributions in this Chapter are: 1. Introduce a reduced complexity K-Best SDA based on SE strategy. In our decoder, only partial path extension needs to be done. Simulation showed when applying layer reordering, our SDA has almost the same performance as original K-Best SDA, while saving about 25%

complexity. 2. Derive a sorting architecture which applied rank order filters (Batcher's merge sort algorithm). Such sorting architecture exploits the natural partial order from SE enumeration, and can significantly reduce the sorting complexity (around 50%) comparing with bubble sorting algorithm

3.1 Reduced Complexity K-Best SDA

The K-Best SE SDA described above has constant throughput, fixed data path, and is desirable for hardware implementation. However, the complexity is high (need to expand K paths to KM_c paths at each layer and select K best candidates out of KM_c PEDs). There are some approaches to reduce such complexity.

First, for $K>M_c$, when performing the path expansion for each node, it is always necessary to fully expand one path at layer i to M_c paths at layer i-1. For this case, no path expansion complexity can be reduced.

Second, if $K \le M_c$, it is not necessary to fully expand a path at last layer to M_c paths at current layer. Here, only expansion to the first SE enumerated K paths is sufficient. The reason is that after SE enumeration, the branch costs and the PEDs of the last $M_c - K$ paths are already larger than the first K paths. Therefore, none of them will become one of the K survival paths after the path expansion and sorting. In such cases, the path expansion complexity can be reduced to K from M_c for each node, and the total sorting complexity can be reduced to sort K smallest PEDs out of K^2 instead of KM_c.

Moreover, for $K \le M_c$, more complexity is possible to be reduced. To expand one path to K paths is the sufficient condition to obtain the K smallest PEDs after

sorting. However, it is not always necessary. Suppose the final K best paths at layer i-1 have the distribution $j_1, j_2, ..., j_K$, where j_m is the number of survival paths (among the total K survival paths) expanded from the m-th candidate $\mathbf{s}_m^i$ of the previous layer i, and $K = j_1 + j_2 + ... + j_K$.

Let $j_{\max} = \max(j_m)$, $1 \le m \le K$. Obviously we have $j_{\max} \le K$. For such case it is sufficient that we expand each path from the last layer into $j_{\max}$ SE enumerated paths (the same reason here, any later path than $j_{\max}$ which has larger PED cannot be among the K survival paths). In this way, we can further reduce the path expansion complexity from K to $j_{\max}$, and the sorting complexity from K out of K^2 to K out of $K \cdot j_{\max}$.

However, here $j_{\max}$ is not constant, varied with different channel and signals, and is unknown to us. To reduce complexity, we can only use some constant J (less than K) as a guess for $j_{\max}$. If J is too small, it might be less than $j_{\max}$ and may introduce performance degradation. On the other hand, if J is too large, not much complexity can be saved. Following are some discussion on this method:

1. There is no fixed law to select proper J value here. One way is empirical by trying different values with simulation.
2. Dynamic values J_i can be used for decoding at layer i. According to the discussing in [16], a good approach is to make J_i large for bigger i (early layers) and use smaller J_i for later layers. This can minimize the probability of missing ML solution at early stages, and reduce performance loss. Using dynamic J_i can further reduce the complexity.

3. The disadvantage of using of dynamic J_i at each layer is that it will break the regularity which normal K-Best SDA has at each stage. This makes it more difficult for hardware implementation.

Figure 3.1 shows the simulation result by using the complexity reduction method discussed in this section. The MIMO system used 4x4 antennas, and the constellation is 64QAM. By decoupling the complex constellations, the real model used is an 8x8 8PAM MIMO system. We used the sorted QR decomposition here, which was introduced for decoding layered space-time codes in [7].

The simulation result compares the performance (symbol error rate) of the ML detection, the normal 8-Best SDA, reordered 8-Best SDA, the reduced reordered 8-Best SDA (*J=6*), and reduced reordered 8-Best SDA (*J=4*) at different SNRs (E_s / N_0) .As stated above, we applied the reordered QR decomposition to K-Best SDA, and it has been shown the reordered K-Best SDA has better performance than the normal K-Best SDA. Here we use the result of reordered 8-Best SDA comparing with normal 8-best (the dashed line). From the result it can be seen that there is almost no performance difference between original reordered 8-Best SDA and the modified 8-Best SDA (*J=6*) (dashed dot line in green). Therefore, by applying our strategy discussed above, 25% path cost computation complexity and 27% sort operation (8 out of 48 comparing with 8 out of 64) can be saved. If we let *J=4*, simulation tells the complexity can further be reduced to less than 50%. However, there is about 0.3dB performance degradation for such small *J*.

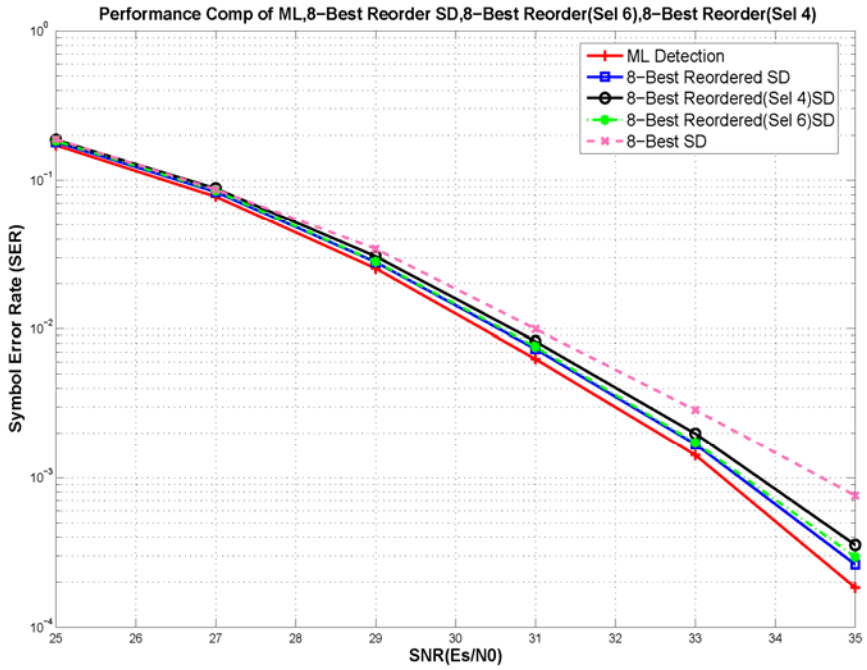

Figure 3.1. Performance comparison of ML, 8-Best, 8-Best reordered, 8-Best reordered SD (*J*=6*)*, and 8-Best reordered SD (*J*=4*)* (*N*=*M*=4, 64QAM).

3.2 Improved K-Best Sorting Architecture

In this section, an efficient sorting architecture has been introduced to K-Best SDA which can save about 50% sorting efforts.

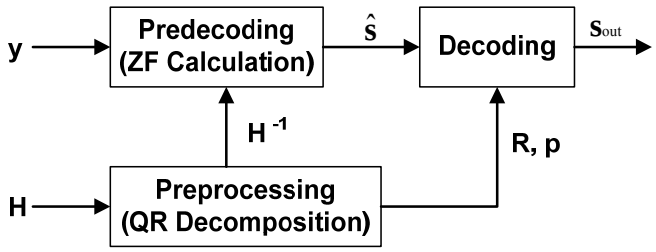

Figure 3.2. Block diagram of K-Best lattice decoder.

The block diagram of a K-Best SDA, consisting of a preprocessing unit, a pre-decoding unit, and a decoding unit, is shown in Figure 3.2. The preprocessing unit is used for the sorted QR decomposition and computing the inverse of **H** (this pre-computation only needs to be done once if **H** does not change). Pre-decoding unit is to compute the ZF solution $\hat{\mathbf{s}}$. **p** is the permutation vector generated by preprocessing unit. After decoding, $\mathbf{s}_{out}$ needs to be permuted reversely to recover its original order. Decoding module has an *M* stage pipelined K-Best decoding structure, whose detail is shown in Figure 3.3.

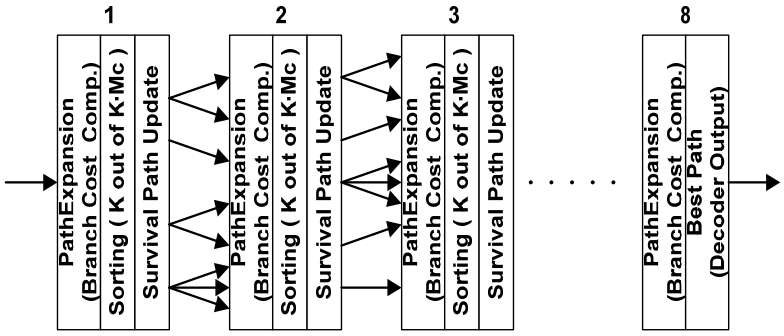

Figure 3.3. An *M* stage decoding module of a K-Best SE SDA *(M=K=8)*.

Based on the K-Best SE decoding algorithm described in Section 3.3, the detail of the decoding unit is shown in Figure 3.3. It has *M* decoding stages, which can be implemented in a pipelined fashion, and for each stage there are 3 sub-modules: path expansion, sorting, and survival path update, corresponding to the step 6, 7, 8 in the algorithm. For the last stage, if only the best path is needed for hard decision, its structure is the same as in Figure 3.3. If the best *K* paths need to be outputted for obtaining soft decoding information, the last stage is the same as the middle stages.

For the K-Best SE decoder at each stage, the sorting operation sub-module accounts for the major complexity (selecting K paths with smallest PEDs out of KM_c is computational intensive and takes lots of comparisons and swaps operations, which is time-consuming). Hence, reducing the sorting complexity is crucial in reducing the complexity of K-Best SDA.

In [6], the bubble sort algorithm is applied to conduct the sorting. However, we found out that when applying the SE method to the K-Best SDA, by exploiting the natural partial orders coming with SE enumeration, a smarter sorting architecture can be adopted to considerably reduce the sorting complexity.

Let $T_i^1, T_i^2, ..., T_i^K$ denote the K smallest PEDs from layer i. After SE enumeration (here it can be done using a lookup table [14] instead of doing sorting) and path expansion (each path expanded to M_c paths), we have KM_c partial Euclidean distances $T_{i-1}^{1,1}, T_{i-1}^{1,2}, ..., T_{i-1}^{1,Mc}, ..., T_{i-1}^{K,1}, T_{i-1}^{K,2}, ..., T_{i-1}^{K,Mc}$ at layer i-1, where $T_{i-1}^{m,n}$ stands for the PED of the n-th path expanded from the m-th path at layer i. The sorting operation is to select K smallest PEDs. First, it is not necessary to do fully sorting. Partial sorting which finds out the K smallest is sufficient. Moreover, based on the SE enumeration we know $T_{i-1}^{m,1} < T_{i-1}^{m,2} < ... < T_{i-1}^{m,Mc}$ for each $1 \le m \le Mc$, which means the KM_c PEDs have been partially ordered in each group (group size M_c, K groups). Exploiting such property, instead of using the partial bubble sorting, we can use the modified rank order filter (Batcher's merge sort algorithm) [15] as the architecture for the K-Best sorting at each stage, which can significantly reduce the sorting complexity.

Similarly, we take the 4x4 64QAM MIMO system used in the previous section as an example. After real decoupling, it becomes an 8x8 8PAM system. When 8-Best SE decoding is used, we have *K=Mc=8*.

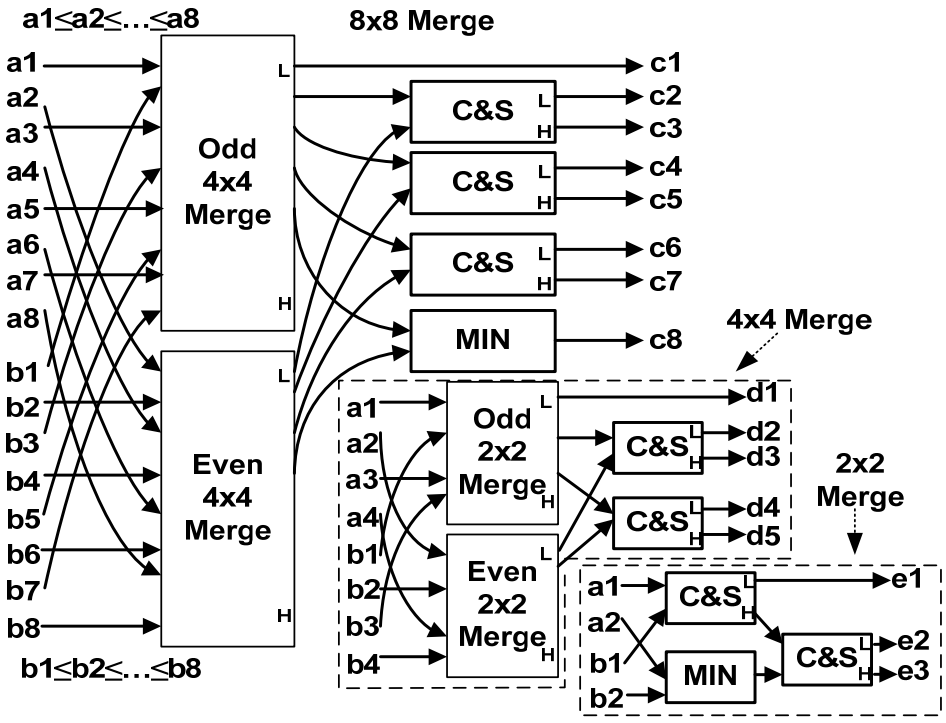

Figure 3.4. Modified architecture of 8x8 merge sorting.
(4x4 &2x2 merge sort modules are given at right bottom corner)

Figure 3.4 shows the modified 8x8 merge sort architectures, which takes in two partial sorted arrays (each has 8 entries) and outputs the minimum 8 entries. The 4x4 & 2x2 merge-sort modules are also shown at the right bottom corner. Such architecture is exactly what we need in the SE K-Best sphere decoder; for each survival path was expanded to 8 paths with ordered PED after path expansion. Then we can apply the sorting architecture in Figure 3.4 to complete the sorting

job. It should be mentioned that we only show example architecture of 8x8, 4x4 &2x2 merge. Actually this architecture can be easily modified to 3x3, 5x5, 6x6 merge etc. For instance, the 3x3 merge can be developed from the 4x4 merge by removing the unused C&S modules related to a4 and b4.

The modified merge sort architecture in Figure 3.4 will greatly reduce the sorting complexity (refer to the results in Table 3.1), and can be used for sorting 8 smallest out of 64 PEDs in the 8-Best SDA. At first, use PEDs $T_{i-1}^{1,1}, T_{i-1}^{1,2}, ..., T_{i-1}^{1,8}$ and $T_{i-1}^{2,1}, T_{i-1}^{2,2}, ..., T_{i-1}^{2,8}$ as the input to the 8x8 merge sort (it has been modified to discard the remaining 8 larger outputs because they will not be used later). Then the sorted smallest 8 PEDs can be combined with another 8 PEDs $T_{i-1}^{3,1}, T_{i-1}^{3,2}, ..., T_{i-1}^{3,8}$ as the merge sort input, and by doing this iteratively, the final 8 smallest PEDs can be obtained (the merge sorting can also be done in a fully parallel manner, but the complexity is the same).

TABLE 3.1 SORTING COMPLEXITY COMPARISON (C&S)

	8-Best (8 out of 64)	8-Best (*J*=6) (8 out of 48)
Bubble sort	63+62+…+56=476	47+46+…+40=348
Proposed sort	20*7=140	16*4+20*3=124

Table 3.1 compares the sorting complexity between bubble sort and our modified merged sort (the numbers stand for average times of the compare and swap operations needed. A C&S unit can be implemented with a comparator and a multiplexer). For instance, for the sorting of 8-best SDA 4x4 64QAM, using bubble

sort needs 476 C&S operations. By using our architecture, 7 stages of merge sort are needed. For each modified 8x8 merge sort, 20 C&S is used (a 8x8 merge sort need two 4x4 merge sort plus 4 extra C&S. The 4x4 merge sort has 4 or 5 outputs and needs two 2x2 merge unit plus 2 C&S. Each 2x2 merge unit has 3 C&S. So totally (2*3+2)*2+4=20 C&S are used). Therefore, by using our modified sorting architecture, 70% complexity can be saved. The third column stands for a modified K-Best SD algorithm; *J* denotes the number of child nodes to be calculated for each node. Here *J*=6 means for each node, we only compute the PED of the first 6 child nodes of the SE enumeration. The simulation result in Figure 3.5 shows this method has almost the same performance as regular 8-Best SDA while having less complexity. For this case, we need to use 6x6 merge sort with 8 outputs sorting. The 6x6 unit contains two 3x3 units plus 4 extra C&S. In this way, the sorting is further reduced to 124 at each stage, i.e., almost 74% sorting complexity has been decreased compared with the original 8-Best SDA.

3.3 The combination of the Layer Reordered K-Best SDA and Merge Sorting

In the above discussions, we introduced the layer reordered K-Best SDA and Dynamic K-Best SDA, and we showed that these two methods can be combined together to further increase the detection performance or reduce the decoder complexity, i.e., to achieve the same performance, a smaller K value can be used.. In Section 3.2, we proposed an improved sorting architecture, which can save about 50% of the sorting efforts. Certainly, this sorting scheme can be applied to the

combined K-Best SDA to achieve a significant total complexity savings for the complete K-Best sphere decoder design.

3.4 Comprehensive Complexity Analysis

In this work, we take the 4x4 64QAM combined 6-Best SDA as an example, to analyze the complexity savings. As the results shown in Figure 2.2, by applying the layered reordering and using dynamic K values at different layers ($\mathbf{K} = [8\ 8\ 8\ 6\ 6\ 6\ 4\ 4]$), the combined 6-Best SDA has almost the same performance as the regular 10-Best SDA. The total complexity of the SDA comes from three major operations: path expansion, sorting, and survival path update.

1. Path Expansion: As for the regular 10-Best SDA, at the top layer only the PEDs of the 8 nodes are calculated; at the lower layer, for each survival candidate, the PEDs of its 8 child nodes need to be computed. Therefore, 10*8=80 PEDs are computed at each layer. Totally, 8+8*8+ (10*8)*6=553 PED calculations are needed. Each PED calculation consists of one multiplication, two additions and one squaring (if Burg's approximation [10] is used, the square operation can be replaced by a MAX). So totally 553 multiplications and 1106 additions are needed. For the dynamic 6-Best SDA, totally 8+8*8+8*8+8*8+6*6+6*6+6*4+4*4=312 PED calculation are used, which is 312/553=56.4% of 10-Best.
2. Path Updating: for each survival path, we need to update $b_i(\mathbf{s}^i)$ according to (2.5) ($\rho_i(\mathbf{s}^i)$ can be pre-computed), which is used by the computation of PEDs at lower layers. From (2.5), $b_i(\mathbf{s}^i)$ is a partial sum which needs *M-i*

multiplications (for 64QAM, s_j can only be ±1, ±3, ±5, and ±7, hence the multiplication can be replace by shift and add), and additions. For normal 10-Best, totally 8+10*6=68 path are updated, i.e., 8*1+10*2+10*3+10*4+10*5+10*6+10*7=278 multiplication and additions. As for combined dynamic 6-Best, 8+8+8+6+6+6+4=46 path are updated, total 8+8*2+8*3+6*4+6*5+6*6+4*7=166 multiplications and additions, which saves (1-166/278)=40.28%.

3. Sorting: As for the regular 10-Best SDA, each stage we need to sort 10 smallest PEDs out of 10*8=80 (the top layer is just 8 candidates, no sorting, and the 2nd layer is sorting 10 out of 8*8=64). Totally (63+62+…+54)+(79+78+… +70)*6=5045 comparisons and swaps. However, for the dynamic 6-Best with merge sorting, the top stage needs no sorting. The 2nd and 3rd stage is to sort 8 out of 64, so 20*7*2=280 C&S when using our architecture. The 4th stage is to sort 6 out of 48, needs 14*7=98 C&S. The 5th and 6th stage is to sort 6 out of 36 candidates, 14*5=70 C&S are used. The 7th stage is to sort 4 out of 24, 8*5=40 C&S are needed. The final stage is to sort 4 out of 16, so needs 8*3=24 C&S. Totally 280*2+98+70*2+40+24=862 C&S units. The saving is 1-862/5045=83%.

The overall complexity results are shown in Table 3.2.

From the comparison, even regardless the memory access and area savings, our proposed sphere decoding algorithm and sorting architecture can achieve a total complexity saving of 68% (here we estimate the complexity of a multiplication by 1, 3, 5 or 7 as 2 additions, a MAX or C&S unit as 1.3 additions).

TABLE 3.2 TOTAL COMPLEXITY COMPARISON

	Addition	Multiplication	MAX	C&S
Nor. 10-Best	1384	831	553	5045
Dynamic 6-Best + merge sort	790	478	312	862
Savings	43%	42.48%	43.6%	83%

3.5 Conclusions

In this Chapter, we have introduced a reduced complexity K-Best SDA which can be used for $K \le M_c$ cases. By selecting the J value less than K, the total decoder complexity can be reduced. In addition, the modified merge sort architecture is presented and applied to the sorting of K-Best SDA at each stage. Such architecture can be used for the sorting of any K-Best SE lattice decoder while significantly reducing the sort complexity.

Moreover, the simulation results show that these three methods can be combined together to achieve the same detection performance as regular K-Best SDA with much smaller K values. Therefore, when this efficient sorting method is applied, significant complexity reductions can be realized. Hence, a comprehensive complexity analysis has been presented to demonstrate that even regardless the memory access time and area savings, our proposed sphere decoding algorithm and sorting architecture can achieve a total complexity saving of 68%.

4 NEW PARALLEL AND PIPELINE INTERLEAVED SPHERE DECODER ARCHITECTURE

The sphere decoding algorithm [2][3][4] is a key algorithm to achieve the optimal ML performance for MIMO systems. The basic principle of SDA is to avoid the exponentially complex exhaustive search in the signal constellations, by applying a sphere constraint (only the constellation points within the sphere would be considered) and transform the ML detection problem into a tree search and pruning process. Regular SDA conducts a depth-first search in the tree while the K-Best lattice decoding algorithm [5], a variant of SDA, does a breadth-first tree search. The latter approach, however, has performance degradation unless K is sufficiently large. In this work, our discussion is focused on the regular SDA.

SDA is very complex for hardware implementation. To the best of our knowledge, the sphere decoder designs published in the literature have lower throughput than 180Mb/s, which is below the requirement of next generation high-rate wireless communication systems (over 200Mb/s). Therefore, efficient high-speed architectures for sphere decoder implementation are really desirable.

In this Chapter, we first propose a parallel sphere decoding scheme. In this method, the whole constellation tree is divided into two sub-trees, and the two processing engines (PE) can conduct depth-first search in parallel and update the new radius. Thus the decoding throughput is significantly improved. Considering the parallel architecture needs to double the hardware cost, we further introduce the pipeline interleaved SD architecture. For this architecture, by exploiting the similarity and interleaving the data streams for both processing engines, only one

PE is needed plus some small interleave control logics. The new sphere decoder has almost the same hardware cost as conventional SD with 44% improvement of the throughput.

4.1 Conventional Sphere Decoder Architecture

The detailed MIMO system model and SDA can be found in [10]. For convenience in later discussions, some important equations are given as follow, where we adopt the same notations as [10].

The partial Euclidean distance (PEDs) are given by

$$T_i(\mathbf{s}^{(i)}) = T_{i+1}(\mathbf{s}^{(i+1)}) + \left|e_i(\mathbf{s}^{(i)})\right|^2, \tag{4.1}$$

where $i = M_T, M_T - 1, \ldots, 1$ is the layer number in the tree search, and the branch cost $\left|e_i(\mathbf{s}^{(i)})\right|^2$ can be obtained as follows:

$$\left|e_i(\mathbf{s}^{(i)})\right|^2 = \left|\hat{y}_i - \sum_{j=1}^{M_T} R_{ij} s_j\right|^2. \tag{4.2}$$

We can further decompose the equation to separate the part influenced by s_i:

$$\left|e_i(\mathbf{s}^{(i)})\right|^2 = \left|b_{i+1}(\mathbf{s}^{(i+1)}) - R_{ii} s_i\right|^2, \text{ and} \tag{4.3}$$

$$b_{i+1}(\mathbf{s}^{(i+1)}) = \hat{y}_i - \sum_{j=i+1}^{M_T} R_{ij} s_j. \tag{4.4}$$

In the above equations, M_T stands for number of antennas (in this work, we assume the numbers of transmit antennas and receive antennas are equal), matrix **R** comes from QR decomposition of channel matrix, vectors $\hat{y}$ and $\mathbf{s}^{(i)}$ represent

zero forcing solution and partial decoded symbol, respectively. The goal is to find the symbol **s** which minimizes $T_1(\mathbf{s})$. Figure 4.1(a) shows two sphere decoders working in parallel. For each new node $\mathbf{s}^{(i)}$ to be examined, the branch cost computation unit computes $e_i(\mathbf{s}^{(i)})$ based on (4.3). According to [10], the l^∞ norm can be used to approximate l^2 norm with reduced complexity. Eq. (4.1) can be rewritten as:

$$\sqrt{T_i(\mathbf{s}^{(i)})} \approx \mathrm{MAX}(\sqrt{T_{i+1}(\mathbf{s}^{(i+1)})}, \left|e_i(\mathbf{s}^{(i)})\right|), \tag{4.5}$$

where $T_1(s) < r^2$ is the sphere constraint. In Figure 4.1a, the PED update unit computes the MAX using (4.5) and updates the new PED. The result is passed to the depth-first tree search control logic that checks the sphere constraints, updates radius *r*, does tree pruning and determines which node to be examined next. Finally $b_i(\mathbf{s}^{(i)})$ is updated according to (4.4) for succeeding operations. Then it comes the new cycle to evaluate another candidate symbol.

4.2 Parallel Sphere Decoder

According to the architecture discussed above, the sphere decoding is actually an iterative process. Due to the long computation delay associated with the loop, the clock speed is quite limited. Adding pipeline in the loop can reduce the critical path length, but it cannot increase the overall throughput. Hence, we propose to use parallel processing to increase the throughput of sphere decoders.

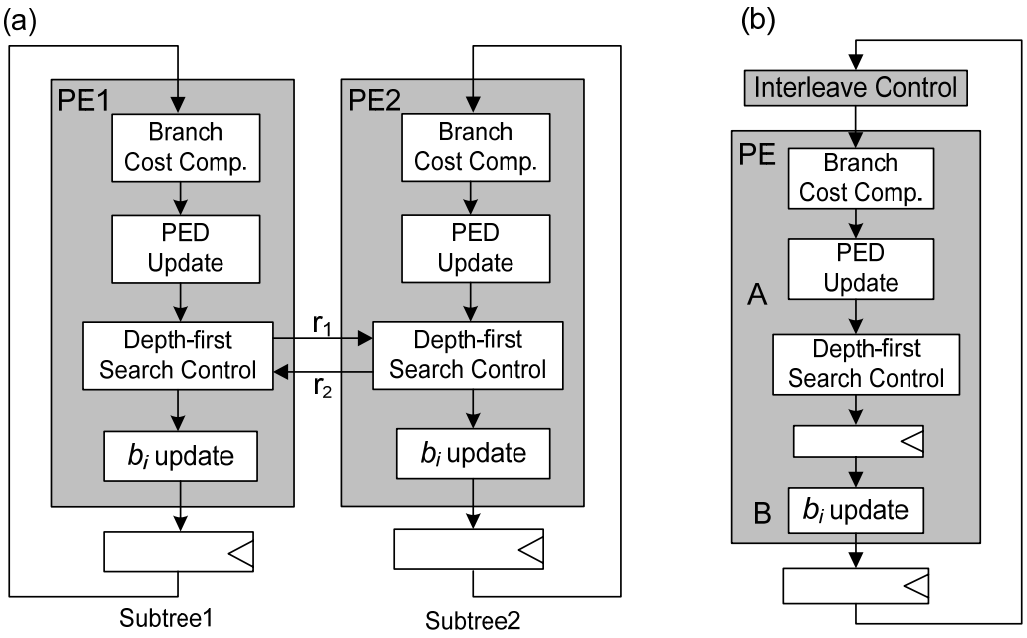

Figure 4.1. (a) Parallel SD architecture, (b) Pipeline interleaved SD architecture.

For instance, for a 4x4 64QAM MIMO system, after real decoupling, the tree structure becomes 8 layers and every node has 8 child nodes. The conventional SD is to search the constellation tree and find out the ML solution. In order to facilitate parallel processing, we can split the constellation tree into multiple sub-trees. For example, we can divide the nodes at the first layer alternately into two groups, e.g., we group them as {-7, -3, 1, 5} and {-5, -1, 3, 7} (according to the constellation values) and all the nodes below the first layer are kept unchanged. Then as shown in Figure 4.2, the whole constellation tree is split evenly into two half-size sub-trees (the shaded nodes form a sub-tree, and the remaining nodes comprise the other sub-tree). In Figure 4.1(a), we can apply the sphere decoding architecture to both sub-

trees to perform sphere decoding in parallel, i.e., PE1 conducts depth-first search within the constellation subtree1, while PE2 searches the subtree2. To expedite the searching process, two sub-trees exchange the current sphere radius r with each other to tighten the sphere more quickly. When a leaf node has been reached, the sphere decoder updates the new radius, and also passes the new radius to the other PE; thus the search complexity can be reduced with the faster shrinking radius. The simulation results are provided in Section 4.4.

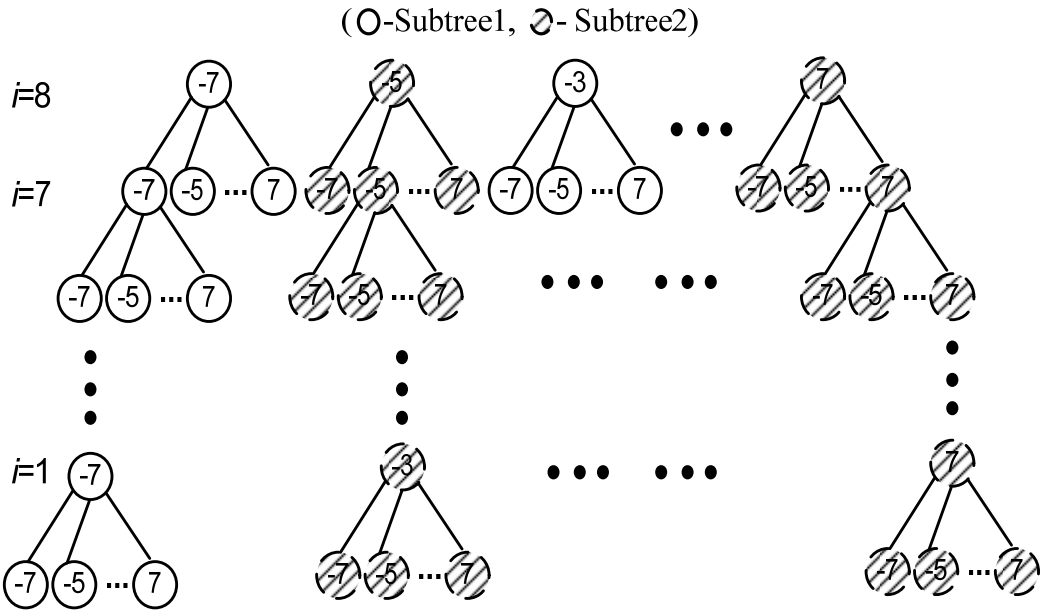

Figure 4.2. Example of tree splitting.

Actually, after Schnorr-Euchner (SE) enumeration, it is a better way to group the first layer nodes according to their indexes in the enumeration. Later in the simulation results we will show such grouping by indexes has better efficiency.

4.3 Pipeline Interleaved Sphere Decoder

The above parallel sphere decoder can increase the decoding speed at the cost of doubling the hardware, which is not efficient. Considering the depth-first search in the two sub-trees are independent (the nodes at the first layer were evenly split), and both of them have exactly the same hardware architecture (both data path and control), we can exploit the pipeline interleaving technique to save hardware. For the detailed principles of pipeline interleaving, readers please refer to [15].

TABLE 4.1 PIPELINE INTERLEAVED DATA PROCESSING SEQUENCE

Clock	**PE_A**	**PE_B**
1	*S*1	*S*2
2	*S*2	*S*1
3	*S*1	*S*2
4	*S*2	*S*1

Figure 4.1(b) shows the architecture of a 2-level pipeline interleaved sphere decoder. It inserts 1-stage pipeline register in the PE, which can reduce the critical path by half in the ideal case. In this way, the clock speed can be roughly doubled due to the shorter critical path. However, after applying pipelining, one loop iteration now takes two clock cycles, which brings no benefit. To facilitate decoding speedup, we can interleave the data patterns from subtree1 and subtree2 into the odd and even cycle of the processing loop. For example as shown in Table 4.1, at an odd clock cycle, the top part of PE (the part above the pipeline register, we name it PE_A, and the bottom part PE_B), processes subtree1 (*S*1), and the bottom part processes subtree2 (*S*2). At the next clock cycle, the data of subtree1 come to

the PE_B and the data of subtree2 loop back to PE_A. In this way, the data of two sub-trees has been interleaved and processes by PE_A and PE_B alternately. In brief, the architecture in Figure 4.1(b) is equivalent to the parallel sphere decoding architecture of Figure 4.1(a). It should be noted that this pipeline interleaved architecture does not introduce extra hardware except pipeline registers and small overhead in control logic.

Hence, by applying such pipeline interleaving technique, the new sphere decoder can achieve the same throughput as the parallel SD, whereas it has only small extra hardware compared with the conventional SD.

4.4 Simulation Results

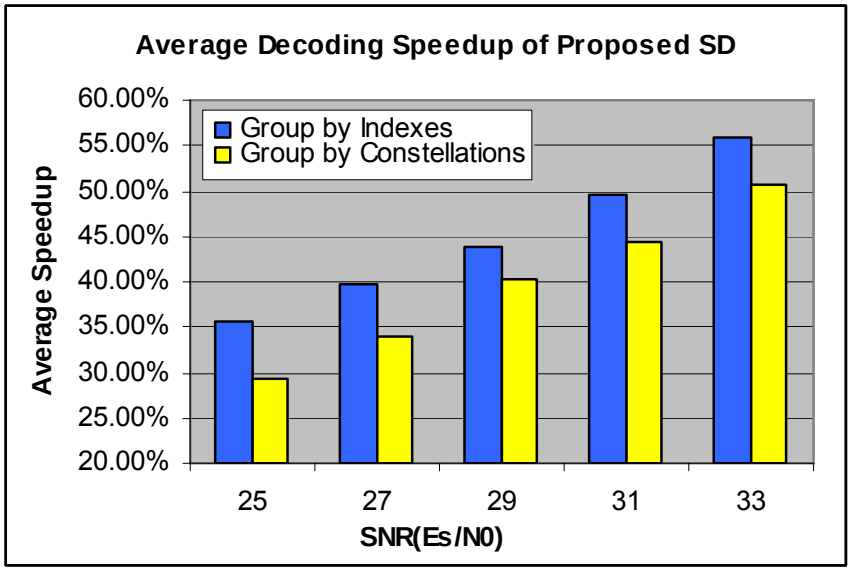

Figure 4.3. Average decoding speedup of proposed sphere decoding architecture (4x4 MIMO system with 64-QAM modulation).

Figure 4.3 shows the simulation results of a 4x4 64-QAM MIMO system. We split the constellation tree in two different ways (according to the 1st layer real constellation value or the node index after SE enumeration), and applied the pipeline interleaving sphere decoding scheme. The results show that the latter tree-splitting method has slightly better performance because it divides the tree more evenly after SE enumerations. On average, our sphere decoder takes 69.54% total computational time of a regular sphere decoder when SNR ($Es/N0$) is 29dB. Therefore, for this example MIMO system, our proposed architecture provides 43.80% decoding speedup with negligible hardware overhead.

4.5 Conclusions

In this Chapter, an efficient pipeline interleaved sphere decoding architecture has been presented. Such a scheme can significantly increase the decoding throughput. For our example 4x4 64-QAM MIMO system, the pipeline interleaved SD can achieve an average speedup of 44% with negligible hardware overhead compared with the conventional sphere decoders.

5 EARLY-PRUNING K-BEST SPHERE DECODER

The sphere decoding algorithm has been used for maximum likelihood detection in MIMO systems, and the K-Best sphere decoding algorithm is proposed for MIMO detections for its fixed complexity and throughput. However, to achieve near-ML performance, the K needs to be sufficiently large, which leads to large computational complexity and power consumption in path expansion, sorting, and path updating. Therefore, it is desirable to reduce the computational complexity and thus the power consumption for K-Best SD designs.

In this Chapter, we introduce some dynamic early-pruning schemes, which will eliminate the survival candidates with relatively large partial Euclidian distances (PEDs) at early stages. These candidates are unlikely to become the ML solution when the tree searching reaches the final layer. Therefore, such early pruning can save computation and power consumption without sacrificing the performance. Our simulation results show that for the 4x4 64QAM MIMO system, by applying the proposed schemes, about 55% computational complexity can be reduced with almost no performance degradation.

The MIMO system model has already been described in Chapter 1, and the SDA, SE enumeration and the K-Best SDA has been introduced in Chapter 2. In Section 5.1, we describe our early-pruning scheme and demonstrate its improvement with simulations. In Section 5.2, we combine our scheme with a threshold-based algorithm [20] and show that more complexity saving can be achieved. Conclusions are drawn in Section 5.3.

5.1 Early Pruning K-Best SD

The original K-Best SDA is to conduct the breadth-first tree search, keep the K paths with smallest PEDs at each layer i until it reaches the bottom layer (i=1), then output the final path with the smallest total cost. If the path corresponding to the ML solution is retained among the K survival paths at each stage, then the final path with smallest cost is the ML solution, and the decoding is correct. Otherwise, the ML solution would be missed at early stages, and decoder error occurs. Hence, to reduce the possibility that the ML solution is pruned at early layers, the K value needs to be sufficiently large (normally≥8). However, this will lead to large computational complexity and power consumption for path expansion, sorting, and path update. In this Chapter, we will introduce a scheme that can reduce such complexity while maintaining the decoding performance.

Supposing the K survival paths at layer i are P_1, P_2,…, P_K. When the sphere decoder descends one layer (i.e., at layer i-1), the candidates paths are P_1, P_2,…, $P_{K\cdot Mc}$ after path expansion and sorting. The corresponding PEDs are denoted as T_1^{i-1}, T_2^{i-1}, …, T_{KMc}^{i-1}, with $T_1^{i-1} \le T_2^{i-1} \le \ldots \le T_{KMc}^{i-1}$. The original K-Best algorithm is to keep the K candidate paths with smallest PEDs among these KMc paths. However, from our observation, under many circumstances, this condition is too loose.

For instance, let us consider two paths P_U and P_V ($1 \le U \le V \le K$) at layer i-1. Suppose their PED difference $\Delta = T_V^{i-1} - T_U^{i-1}$ is large, i.e., the PED T_V^{i-1} of path P_V is much larger than the PED T_U^{i-1} of path P_U. After accumulating the branch cost of

remaining layers, most probably the final cost T_V^1 of path P_V is still larger than the final cost T_U^1 of path P_U. Based on equation (2.6),

$$T_V^1 = T_V^{i-1} + \sum_{j=i-2}^{1} e_V^j , \tag{5.1}$$

$$T_U^1 = T_U^{i-1} + \sum_{j=i-2}^{1} e_U^j . \tag{5.2}$$

Unless

$$\sum_{j=i-2}^{1} e_U^j - \sum_{j=i-2}^{1} e_V^j > \Delta = T_V^{i-1} - T_U^{i-1} , \tag{5.3}$$

the partial sum of the branch costs of the remaining layers will not change the original order of the PEDs). This means the path P_V is unlikely to become the ML solution when the sphere decoder descends to the final layer. Hence, we can prune this path at layer *i*-1 in advance from further processing, thus save the computational effort of the lower layers. And since the eliminated node is unlikely to become the ML solution, such early prune scheme will not affect the decoding performance.

From the above observation, it is possible to set up a criterion to identify such paths that are unlikely to become the ML solution, and prune them at early stages. At layer *i*, the original K-Best SD will retain the *K* paths corresponding to smallest PEDs: T_1^i, T_2^i,…, and T_K^i. Based on the above analysis, if $\Delta = T_K^i - T_1^i$ is large, then path P_K can be pruned. Therefore, we set a bound

$$B = \alpha \cdot T_1^i + (1-\alpha) \cdot T_K^i , \tag{5.4}$$

which is a linear combination of T_1^i and T_K^i. If the PED of a candidate path is larger than B, this candidate path is discarded. Here, α is a value determined by simulations. Such bound, together with the K-Best condition (number of survival paths), can serve as a stricter condition to prune more nodes at early stages.

In Chapter 2, we proposed a layer reordered (LR) K-Best SDA, which can improve the performance of the regular K-Best sphere decoders. In fact, such layer reordering scheme can also be used to reduce the complexity. For example, as for the 4x4 64QAM MIMO system, the 10-Best SD has almost the same (even slighter better) performance as 12-Best regular SD. Therefore, we can use the 10-Best LR-SD to replace the 12-Best regular SD. In this way, 17% complexity can be saved.

In addition, we can combine the early-pruning scheme with the LR-SD in Chapter 2 to achieve more complexity savings. Figure 5.1 shows the simulation results by applying our scheme. In this work, we use the 4x4 MIMO systems (4 transmit and 4 receive antennas). The modulation scheme is 64QAM (By decoupling the complex constellations, the real model used is an 8x8 8PAM MIMO system). In case of hard-output detection, 1000 independent channel realizations (packets) of 1000 uncoded 64QAM symbols are transmitted with 250 symbols from each antenna. Figure 5.1 compares the performance (symbol error rate) of the ML detection, the normal 12-Best SDA, the 10-Best LR SDA, and our 10-Best early-pruning LR SDA (α=1/4 and α=1/3) at different SNRs (E_s / N_0).

From Figure 5.1 we can see the 10-Best layer reordered SD has the same performance as regular 12-Best SD. And after combining the early-pruning (EP) scheme with 10-Best LR-SD, when α=1/4, no performance loss can be observed.

For α=1/3, the 10-Best EP-LR-SD has 0.15dB performance degradation comparing with regular 12-Best SD.

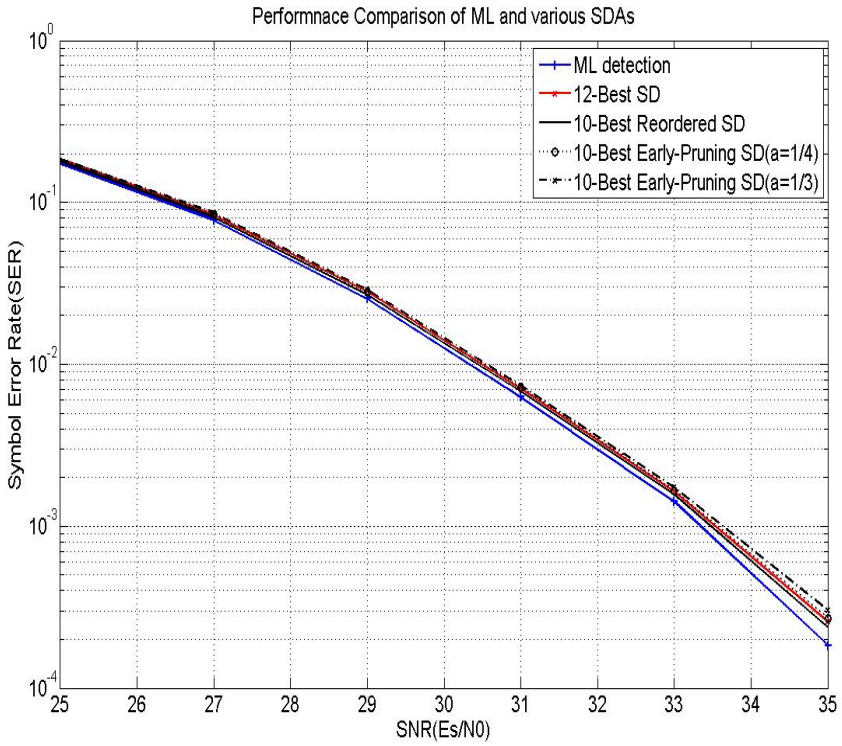

Figure 5.1. Performance comparison of the ML, 12-Best SD, 10-Best LR SD, and 10-Best early-pruning SD (α=1/4 & α=1/3) (*N*=*M*=4, 64QAM).

Figure 5.2 compares the complexity savings of the 10-Best early-pruning layer-reordered SD to the normal 12-Best SD at different SNRs. Compared to α=1/4, the α=1/3 case can achieve more complexity savings (up to 48%). However, the α=1/4 10-Best EP-LR-SD has slightly better performance. Moreover, to save more complexities, we can set the α value as a function α(*i*) of the layer *i*. At later stages, α(*i*) can be bigger (the bound becomes tighter) because from equations (5.1) and (5.2), at later stages fewer remaining branch cost will be added to the PEDs.

Therefore, the possibility of condition (5.3) is smaller. From our simulation, $\alpha = [0.26\ 0.26\ 0.3\ 0.3\ 0.34\ 0.34\ 0.38\ 0.38]$ can be used. Such 10-Best EP-LR SDA can achieve the same saving as α=1/3, while the 0.15dB performance loss can be avoided.

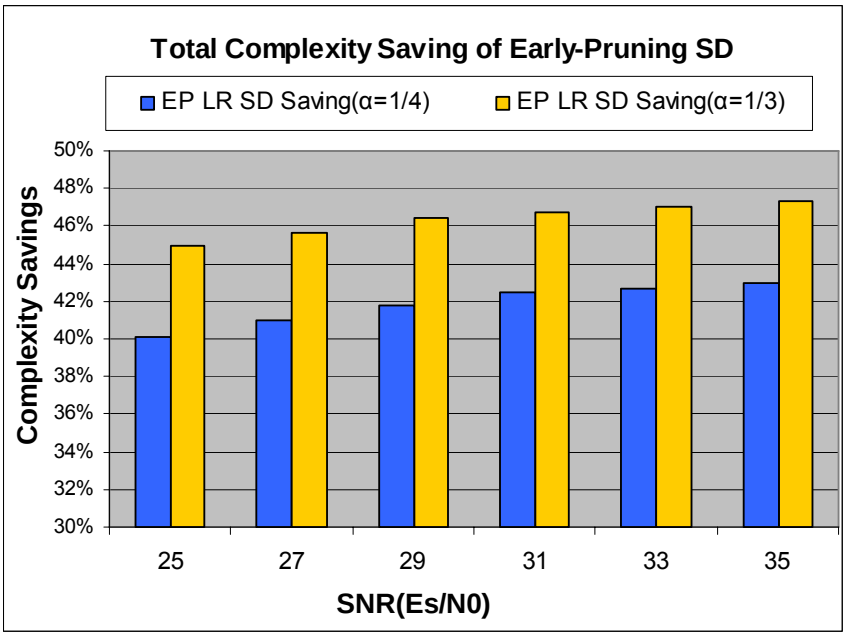

Figure 5.2. Complexity savings of the early-pruning LR 10-Best SD (α=1/4 & α=1/3) compared with regular 12-Best SD (*N*=*M*=4, 64QAM).

5.2 Combined Method with threshold-based SDA

In [20], a threshold-based K-Best SDA was introduced. At each layer, a threshold

$$B = C - \beta_i \cdot (9 - i) \cdot C / 8 \tag{5.5}$$

was used to eliminate candidate nodes with large PEDs, where i stands for the i-th decoding layer, running from 8 to 1, C is the Euclidean distance between the

received signal and the estimation of zero-forcing solution multiplied by the MIMO channel matrix **H.** At each decoding layer, the candidates whose PEDs are bigger than *B* will be discarded. β is the coefficient determined by simulation. This method can also save much complexity especially in high SNR regions.

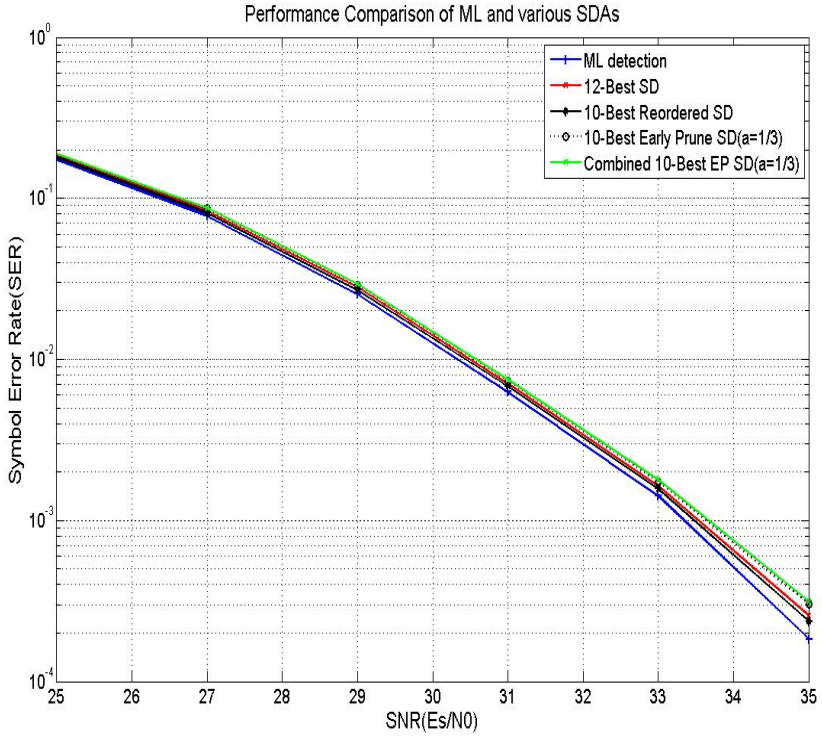

Figure 5.3. Performance comparison of the ML, 12-Best SD, 10-Best LR SD, 10-Best early-pruning SD (α=1/3) & combined EP SD (N=M=4, 64QAM).

Since the early-pruning scheme we introduced in Section 5.1 is dynamic (based on the distribution of the PEDs of the survival paths), it is reasonable that we combine the early-pruned SDA with the SDA in [20]. Here, we can set the combined bound as

$$B = \min\{\alpha \cdot T_1^i + (1-\alpha)\cdot T_K^i, C - \beta\cdot(9-i)\cdot C/8\}. \qquad (5.6)$$

In this way, more computational complexity or power consumption can be saved without sacrificing the performance. The α value can be set as 1/3 or 1/4 as in Section IV, and the β value can be set as 1.

Figure 5.3 shows the simulation result of the ML, 12-Best SDA, 10-Best EP-LR SDA (α=1/3), and the combined 10-Best EP-LR SDA (α=1/3, β=1) for the 4x4 64 QAM system. Compared with the 12-Best SDA, there is almost no performance (less than 0.2db) loss for the combined methods.

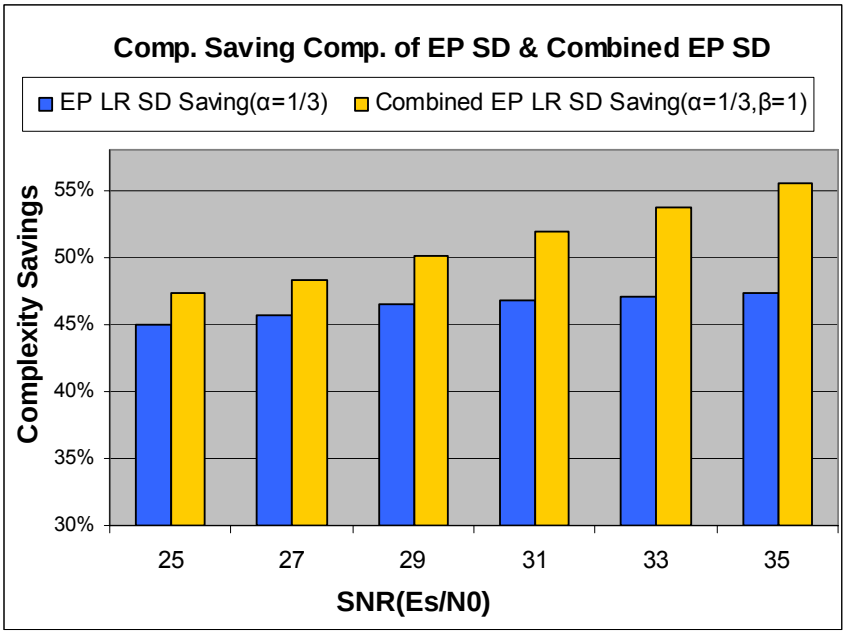

Figure 5.4. Complexity savings comparison of the early-pruning LR 10-Best SD (α=1/3) & combined EP-LR 10-Best SD (α=1/3, β=1) (N=M=4, 64QAM).

In Figure 5.4, the complexity saving of the combined method has been shown. It can be seen that more complexity saving (up to 55.5%) can be achieved with the combined method, especially at high SNR regions.

5.3 Conclusions

In this Chapter, by exploiting the intrinsic properties of the K-Best breadth-first searching algorithm, we have introduced an efficient early-pruning scheme. For our example 4x4 64QAM MIMO system, when such early-pruning scheme is combined with the layer reordered SDA, about 47% complexity savings can be achieved without losing detection performance. This early-pruning scheme can be further combined with the threshold-based K-Best SDA, thus for the same system, the total of 55% computational complexity/power consumption can be reduced compared to the original K-Best SDA.

6 EFFICIENT RADIUS AND LIST UPDATING UNITS DESIGN FOR LIST SPHERE DECODERS

The sphere decoder (SD) has been utilized for maximum likelihood (ML) detection in MIMO systems. In order to improve system performance, the SD is usually combined with the error correction codes where soft decoding is utilized. The SD needs to provide soft information in order to combine with the channel code decoder which implements soft decoding Therefore, the conventional SD, which finds the ML hard decision symbols, needs to be modified. In [8], the list sphere decoder (LSD) was proposed for iterative decoding schemes. By conducting search in the sphere, the LSD generates a candidate list instead of only finding the ML solution. The soft extrinsic information is computed based on the list of candidates, and then delivered the soft-decoder. Therefore, the key of LSD is to find out a candidate list by taking the search and pruning approach.

Differing from the regular SD, the LSD needs to generate a candidate list other than only the ML solution. This will introduce extra latency in finding out the sphere radius, and extra complexity for updating the candidate list. The contributions of this Chapter include the follows: 1) we propose an efficient architecture, which can compute the new sphere radius in one clock cycle, thus reduce the decoding latency; 2) we present a low complexity list updating unit, which can greatly save the complexity for updating the candidate list.

This Chapter is organized as follows. In Section 6.1, we briefly review the sphere decoding algorithm and list sphere decoder. Then in Section 6.2, we describe our fast radius updating architecture and show its improvement. In Section

6.3, the new list updating scheme is illustrated. The complexity analysis has also been provided. Finally, conclusions are drawn in Section 6.4.

6.1 List Sphere Decoder

6.1.1 Conventional Sphere Decoding Algorithm

Based on the MIMO system model described in Section 1.1, an approach to obtain the ML detection is the sphere decoding. It only examines the point inside a hyper-sphere around **y**, i.e.,

$$d(\mathbf{s}) = \left\|\mathbf{y} - \mathbf{Hs}\right\|^2 \le r^2 = C \ , \tag{6.1}$$

which is regarded as the sphere constraints (SC). By applying QR decomposition to H, the right hand side of equation (2.1) can be transformed into a sum of non-deceasing terms for each layer. Thus the search in the sphere is further transformed into a tree search. Once the partial Euclidean distance (PED) exceeds C, its children nodes can all be pruned. The sphere radius (C is the square of the sphere radius. For the convenience of expression, in this work, we call *C* radius) is updated every time a new solution is found. As a result, the complexity of finding ML solution can be significantly reduced.

6.1.2 List Sphere Decoder

To increase the system performance, the sphere decoder can be combined with the error correction code which uses soft decoding. An iterative processing scheme for Turbo code and LDPC code, has been applied to the MIMO systems [8][23]. The block diagram is shown in Figure 6.1. For this iterative decoding

scheme, the MIMO detector needs to generate the soft reliability information (log likelihood ratio) for feeding into the soft-input-soft-output (SISO) decoder.

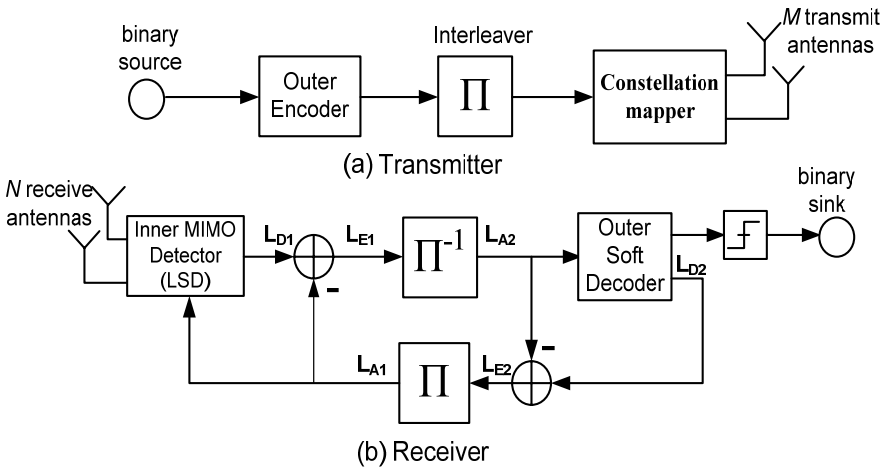

Figure 6.1. MIMO transmission and iterative receiver model.

At the receiver side, the inner MIMO detector (sphere decoder) uses the received symbol vector **y** and *a-priori* (intrinsic) information L_{A1} from the soft decoder, computes the extrinsic information L_{E1} for each bit according to the approximation equation (12) in [8]. Then L_{E1} is de-interleaved and becomes the intrinsic information L_{A2} for the SISO decoder. Based on L_{A2}, the SISO decoder then calculates the extrinsic information L_{E2}, which will become the intrinsic information of inner MIMO detector after being interleaved. Thus, one iteration is completed. To facilitate such iterative decoding, the conventional SD needs to be modified for soft decoding. The list sphere decoder (LSD) was introduced in [8]. Instead of searching the ML solution within the hyper-sphere, the LSD finds out a list of most likely symbol vectors (including the ML solution) which has the

smallest Euclidean distances ($\|\mathbf{y}-\mathbf{Hs}\|^2$), and computes the soft bit reliability information based on the candidates list.

6.2 Fast Radius Updating Architecture

For the list sphere decoder, suppose it has a candidate list $\boldsymbol{L}$ of size N. Each candidate is denoted as $L_i(1 \le i \le N)$, and the corresponding ED is $ED_{L(i)}$ with $ED_{L(i)} \le ED_{L(j)}$ when $1 \le i \le j \le N$. The original LSD decoding flow is illustrated in Figure 6.2(a). At first, the LSD conducts depth-first search at upper layers. When the tree search reaches the leaf nodes and finds out K candidates: $A_i, 1 \le i \le K$, two operation are needed. A) *Candidate List Update*: the new candidates will be inserted into the candidate list, and the original candidates with the biggest EDs will be deleted; B) *New Radius Update*: the radius will shrink from $ED_{L(N)}$ to the new biggest ED in the list. For the original design, the new radius is updated after the new list has been generated (it picks up the biggest ED in the new list and updates radius C). However, this will introduce more latency because only after the new radius has been computed, the depth-first tree search can resume (the depth-first search engine needs the new C to check each PED). In this Chapter, we propose a new LSD decoding flow. In the new flow shown in Figure 6.2(b), the new radius is computed in advance. Therefore, the depth-first search will continue after one clock cycle. And the list update operation can be conducted in parallel. In this way, the DFS and LU can overlap, thus save the latency introduced by the list updating unit.

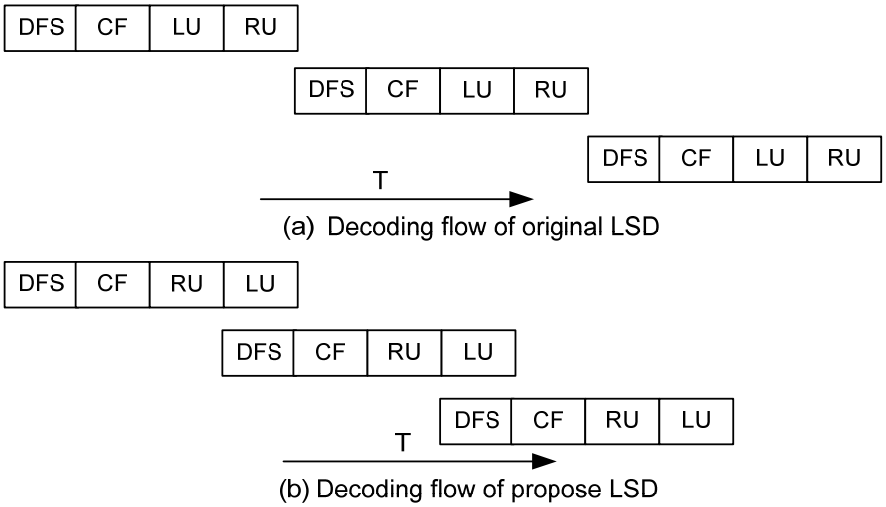

Figure 6.2. Decoding flows of LSD (DFS—depth first search, CF—candidates found, LU—list update, RU—radius update).

To elaborate the new radius update unit, first let us consider the simplest case K=1, i.e., in each search the LSD found one new candidate A. The architecture to compute new C is presented in Figure 6.3.

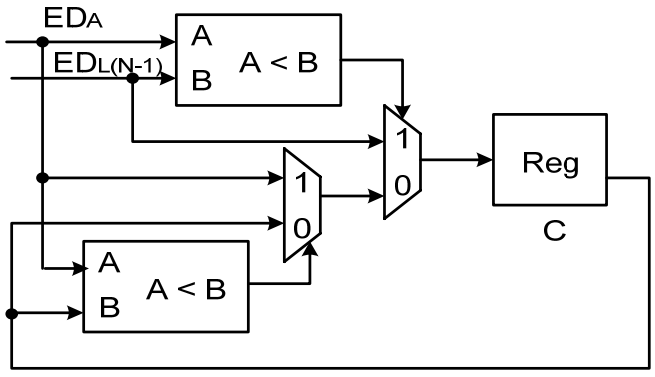

Figure 6.3. Radius update unit for K=1.

The original radius is $ED_{L(N)}$. When the new ED of candidate A has been found, it needs to compare with original C. If ED_A is bigger, the candidate will be pruned, and the radius C remains. If ED_A is smaller, the bigger value between ED_A and $ED_{L(N-1)}$ will become the new radius.

However, for normal SD decoder design (such as [10]), when search reaches the bottom level, all the leaves under the same parent node are computed in parallel to increase the decoding speed. Therefore, more than one new candidate will be found at the CF phase. For instance, for a 4x4 16QAM MIMO system, 4 new candidates will be generated at the same time (K=4). Since SE enumeration is used (refer to [14], a look-up table can be used to avoid sorting), the EDs of the new candidates A are in ascending order. The radius update unit needs to compute the new radius quickly for such scenarios.

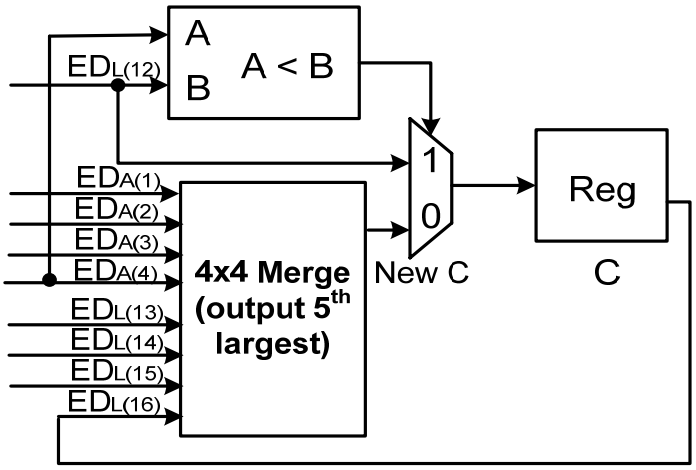

Figure 6.4. Radius update unit for K=4, N=16.

Figure 6.4 shows an example fast radius update architecture for K=4, N=16. For this situation, the original list size is 16 with $ED_{L(1)} \leq ED_{L(2)} \leq \ldots \leq ED_{L(16)}$. Now the 4 newly computed EDs are $ED_{A(1)} \leq ED_{A(2)} \leq ED_{A(3)} \leq ED_{A(4)}$. Therefore, the new radius should be the 5th largest number in the total 16+4=20 numbers. The straightforward method is to insert the list A into list L, and select the 5th largest number. This has large complexity and long latency. The architecture in Figure 6.4 illustrates an efficient way to find out the new radius. Such architecture exploits the existing order in list $\boldsymbol{L}$ and $\boldsymbol{A}$, and uses the fastest way to find out the new radius. First, we need to compare $ED_{A(4)}$ and $ED_{L(12)}$. If $ED_{A(4)} < ED_{L(12)}$, then it means all the new candidates will be inserted into the candidate list (L_{13}, L_{14}, L_{15}, L_{16} will be deleted). And the new radius is $ED_{L(12)}$. Otherwise, the list $\boldsymbol{A}$ and the largest 4 Euclidean distances $ED_{L(13)}$, $ED_{L(14)}$, $ED_{L(15)}$, and $ED_{L(12)}$ (original C) will be sorted by an efficient 4x4 merge sort unit, and the 5th largest value will be output as the new C (this value is guaranteed to be the 5th largest among the whole 20 numbers when $ED_{L(12)} > ED_{A(4)}$).

Figure 6.5 shows the 4x4 and 2x2 merge sort unit used for the new radius update architecture. The dotted line part is just used for illustration and can be omitted in real implementation. C&S stands for compare and swap. The total hardware for the 4x4 merge unit is 7 C&S (each has a comparator and a multiplex). And the total hardware for the radius update unit is 8 comparators and 8 MUXs with the critical path of 3 C&S units.

The above discussion provides an example of designing fast radius update unit. For different MIMO systems with other K and N values, the merge sort unit can be simply modified to compute the new radius in one clock cycle. In addition,

this 4x4 merge unit can be slightly modified to be re-used for the list updating, which will be discussed in the following section.

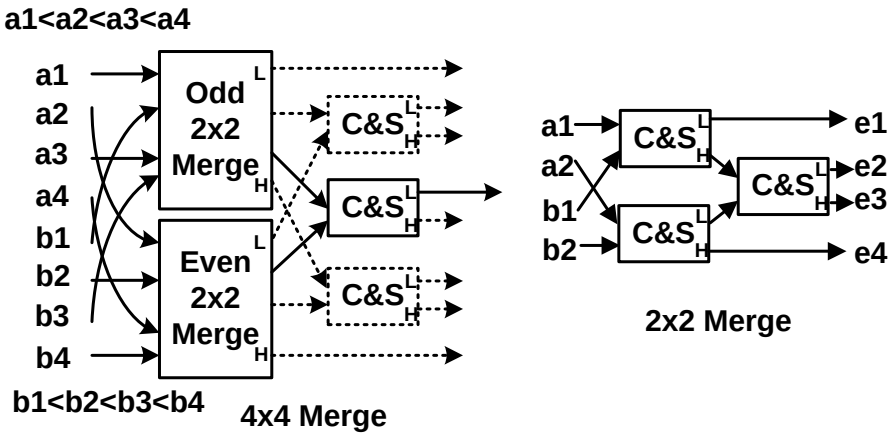

Figure 6.5. The 4x4 & 2x2 merge sort unit: C&S—compare & swap.

6.3 Efficient List Updating

After the new radius has been computed, the depth-first tree search can resume. And the candidate list needs to be updated in parallel. The new list will be used by the next radius and list updating units. Finally, when the whole tree has been traversed, the final list will be needed by the soft information generation unit to compute the soft bit reliability information.

In [24], a tree-type comparator (TTC) array was proposed to update the candidate list. Compared with the fully parallel comparator (FPC) array, the TTC architecture can significantly save the complexity for 4x4 16QAM MIMO systems. In this work, based on the same merge architecture, we can create an efficient

candidate list updating scheme, which will update the candidate list with smaller complexity.

Figure 6.6 shows the list updating unit architecture for N=16, K=4 case. Here, the inputs are the Euclidean distances and history paths coming from list $\boldsymbol{L}$ (16 entries) and $\boldsymbol{A}$ (4 entries). After updating, the LU unit will generate the new paths for the 16 candidates with the smallest EDs. The original 4x4 merge unit needs to be slightly modified since after the comparison of the Euclidean distance, both the ED values and the corresponding candidate paths need to be swapped. Hence, the comparator remains, but the extra MUXs for candidate paths need to be added.

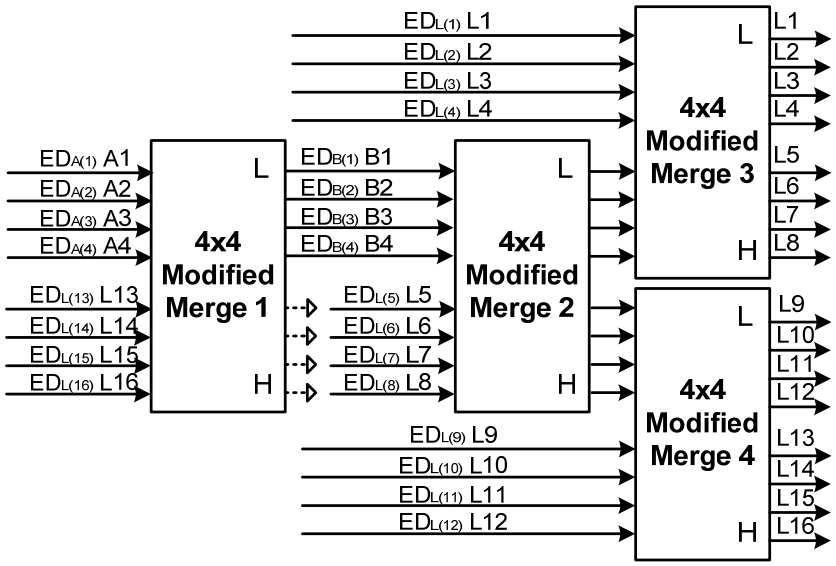

Figure 6.6. List updating architecture for N=16, K=4.

This architecture works as follows: first, the new entries from $\boldsymbol{A}$ are merged with the biggest 4 entries (L_{13},L_{14},L_{15},L_{16}) in the list $\boldsymbol{L}$. The four largest entries after

merge are discarded, and the other 4 smallest outputs B_1,B_2,B_3,B_4 will be further compared to the middle 4 entries (L_5,L_6,L_7,L_8) in the remaining list $\boldsymbol{L}$ of size 12 (the binary search principle is applied here: start merge from the ½ point). Afterwards, the smaller 4 outputs are further merged with the L_1, L_2, L_3, L_4, thus generate the smallest 8 entries for the new list. Similarly, the bigger 4 outputs are merged with L_9,L_{10},L_{11},L_{12}, which become the other 8 entries for the new list.

The total latency for this list updating architecture is 9 comparison operations. An interesting point is: the modified merge unit **1** has the same inputs as the merge unit used for the radius updating unit discussed in the previous section. Therefore, we can reuse the 4x4 merge unit in the radius updating for both radius updating and list updating (the 4x4 merge unit in the RU needs to be changed to the 4x4 modified unit in Figure 6.6, which can swap both the ED and candidate path). In this way, the list updating latency can be further reduced to 6 comparison operations.

Suppose ED has the word-length of W, the history path has S (for 4x4 16QAM, S=16) bits. Therefore, each 4x4 modified merge unit has 9 comparators, $9W$+$9S$ 2-to-1 MUXs for both EDs and history paths. Considering the four merge units has the same structure, it is possible to reuse one merge unit four times instead of using 4 merge units, thus the total area is further reduced to 9 comparators and $9W$+$9S$ 2-to-1 MUXs plus the extra reuse logic: 8(W+S) 4-to-1 MUXs. For this reuse scheme, the total radius updating and list updating procedure will take 4 clock cycles (1 clock cycle for radius updating and 3 clock cycles for list updating). Based on the decoding flow in Figure 6.2, the LU is in parallel with the depth-first search; therefore such latency will not affect the decoding speed.

TABLE 6.1 COMPARISON OF DIFFERENT LIST UPDATING SCHEMES

	# of Comp.	# of 3-to-1 MUXs	# of 2-to-1 MUXs	# of 4-to-1 MUXs
FPC	64	1856	-	-
TTC	16	232	704	-
Merge	9	-	288	256

Table 6.1 compares the complexity of FPC, TTC, and our merge list updating scheme for the example case (4x4 16QAM MIMO, N=16, K=4, W=16, S=16). From the result it can be seen that our proposed merge scheme can save about 45% complexity compared with TTC.

6.4 Conclusions

In this Chapter, we have introduced some new schemes for list sphere decoder. We have first suggested changing the decoding flow to do the radius updating before list updating. This will avoid the latency introduced by the radius updating unit. By exploiting the intrinsic order of the candidate list and the SE enumeration, we proposed a fast list updating architecture based on the merging of two partially ordered arrays. For the example 4x4 16QAM N=16 MIMO system, such radius updating unit can compute the new radius in one clock cycle with the critical path of 3 comparison operations. In addition, we have presented an efficient list updating architecture based on the merge sort and binary insertion. For the same system, by reusing the merge unit in radius updating, this list updating can

achieve a 45% complexity saving compared with TTC without affecting the decoding speed.

7 FAST POINT OPERATION ARCHITECTURE FOR ELLIPTIC CURVE CRYPTOGRAPHY

Public key cryptography has been widely used today for information security and E-commerce. A well-known public key cryptography scheme is RSA, which was first proposed by Riverst, Shamir and Adleman in 1978 [45]. The security of RSA is based on the difficulty of the integer factorization problem. In 1985, Elliptic curve cryptography (ECC) was introduced by Victor Miller [46] and Neal Koblitz [47] independently. The security of ECC is based on the hardness of solving the elliptic curve discrete logarithm problem (ECDLP). Comparing with the sub-exponential time it takes to solve the integer factorization problem, it takes fully exponential time for today's best algorithm to solve ECDLP. Therefore, ECC delivers much higher security strength per bit than RSA. A typical example is: a 160-bit key ECC has equivalent level of security to a 1024-bit key RSA [48]. For this reason, ECC offers potential reduction in storage space, bandwidth and power consumptions, which is very desirable for the security applications in the constraint devices such as cell phones, PDAs, and smart cards.

Due to the advantages of ECC over RSA, a lot of papers have been published on the software or hardware ECC implementations, among which the latter one provides much higher processing speed and is more suitable for real-time applications. The implementation of ECC mainly relies on the operations at three levels: the scalar multiplication, the point addition / doubling, and the finite field modulo arithmetic. The ECC system based on $GF(2^n)$ is widely utilized for its simple field arithmetic and efficient scalar multiplication algorithms. Two different

coordinates: the affine coordinate and the projective coordinate can be used for the ECC where the curve is defined over $GF(2^n)$. It was shown in [50][51][52] that the projective coordinate is more desirable for hardware implementation because it avoids the costly field inversion operation.

However, the conventional point addition and point doubling algorithms conduct the operations in sequential and takes many steps (e.g., 14 steps for point doubling and 22 steps for point addition). In this work, we introduce an efficient fast architecture for the Lopez-Dahab projective coordinates [49]. By applying parallel processing and reusing the field multipliers, the point addition and doubling operations can be significantly accelerated, with reasonable hardware overhead.

The Chapter is organized as follows: in Section 7.1 the mathematic background of elliptic curve cryptography and arithmetic hierarchy are reviewed. Then Section 7.2 describes the projective coordinate and presents its advantages for the hardware implementation. In Section 7.3, the fast point operation architecture is proposed, and the speedup analysis is given. Finally, the conclusions are drawn in Section 7.4

7.1 Elliptic Curve Cryptography Arithmetic

7.1.1 Elliptic Curves

In this work, we will focus on the elliptic curves defined over $GF(2^n)$. The mathematic foundation of ECC is based on the Weierstrass equation for a non-super singular elliptic curve. The equation in affine coordinate is given as:

$$y^2 + xy = x^3 + ax^2 + b, \tag{7.1}$$

where $a, b \in GF(2^n)$ and $b \neq 0$. The set of points $E(GF(2^n))$ includes all points (x, y), where $x, y \in GF(2^n)$, and satisfy equation (7.1), together with the infinity point O. The set $GF(2^n)$ forms an additive abelian group, which is based on the following definitions with $P = (x_1, y_1)$, $Q = (x_2, y_2)$, and $P \neq \pm Q$:

Identity: $P + O = O + P = P$.

Negation: for $P \neq O, -P = (x_1, x_1 + y_1)$

Point addition: $P + Q = R = (x_3, y_3)$ where $x_3, y_3 \in GF(2^n)$,

$$\text{and } x_3 = \lambda^2 + \lambda + x_1 + x_2 + a, \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \qquad (7.2)$$

$$\text{with } \lambda = (y_1 + y_2)/(x_1 + x_2).$$

Point doubling: $2P = R = (x_3, y_3)$, where $x_3, y_3 \in GF(2^n)$,

$$\text{and } x_3 = \lambda^2 + \lambda + a = x_1^2 + b / x_1^2, \quad y_3 = x_1^2 + \lambda x_3 + x_3 \qquad (7.3)$$

$$\text{with } \lambda = x_1 + y_1 / x_1.$$

The major task of ECC is to compute the scalar multiplication kP, where $k = k_l k_{l-1} \ldots k_1 k_0$ is a positive integer and P is a point on $E(GF(2^n))$. The computation of kP is performed by applying the "double and add" method:

$$kP = \sum_{j=0}^{l-1} k_j 2^j P = 2(\cdots 2(2k_{l-1}P + k_{l-2}P) + \cdots) + k_0 P \qquad (7.4)$$

This method requires l doublings and w_k-1 additions, where w_k is the weight of binary representation of k. As can be seen from the definition of point addition and point doubling, each add operation takes 6 finite field additions, 2 finite field

multiplications, 1 square, and 1 inversion, and each doubling operation requires 8 finite field additions, 2 multiplications, 1 square, and 1 inversion.

7.1.2 ECC Arithmetic Hierarchy

The elliptic curve operations are performed at three different layers. As discussed above, the top layer computation of ECC is the scalar multiplication, which is based on the point addition and point doubling operations.

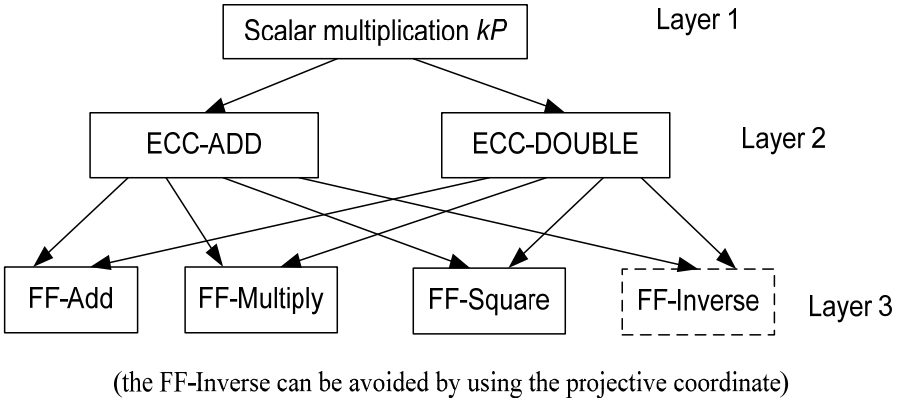

Figure 7.1. ECC arithmetic hierarchy.

In the middle layer is the point addition and point doubling, which are denoted as ECC-ADD and ECC-DOUBLE respectively. At the bottom layer is the finite field operations, which include finite field multiplication, finite field addition, finite field squaring, and finite field inverse operations.

The complete ECC arithmetic hierarchy is illustrated in Figure 7.1.

7.2 Projective Coordinate based point arithmetic

7.2.1 Projective Coordinate

According to the definition of point operations on elliptic curves based on the affine coordinate, we observe that both the point addition and the point doubling need a finite field inversion, which is very costly for hardware implementation. In many finite field arithmetic implementations, the cost-ratio of inversion to multiplication is more than 8. Therefore, it is desirable if the inverse operation can be avoided. An alternative method to implement the point arithmetic is to use the projective coordinate.

A projective plane of the fixed exponential integers (α, β) over $GF(2^n)$ is defined by creating an equivalence relation of the triples $(x, y, z) \sim (X, Y, Z)$ if there exists $\lambda \in GF(2^n)$, and $\lambda \neq 0$ such that we have $(x, y, z) = (\lambda^{\alpha} X, \lambda^{\beta} Y, \lambda Z)$. Every point ($x$, y) on the affine coordinate can be mapped to the projective plane with $\phi : (x, y) \rightarrow (x, y, 1)$. From the above definition, every equivalent class of the triples on the projective plane $(X, Y, Z), Z \neq 0$ can be mapped back to the affine point by $x = X / Z^{\alpha}$ and $y = Y / Z^{\beta}$. Currently, there are three popular projective coordinates applied to the ECC system, which are: a) the Homogenous projective coordinate [51] with α =1 and β =1; 2) the Jacobian coordinate and arithmetic [52] with α =2 and β =3; and 3) the Lopez-Dahab projective coordinate with α =1 and β =2. In the third case, $x = X / Z$ and $y = Y / Z^2$. The computation cost comparison of the point arithmetic based on all the three coordinates is shown in Table 7.1.

TABLE 7.1 COMPARISON OF THE COMPUTATION COST OF POINT OPERATION ON DIFFERENT PROJECTIVE COORDINATES

		Multiplications	Squares
Homogeneous	Add	13	1
	Double	7	5
Jacobian	Add	11	4
	Double	5	5
Lopez-Dahab	Add	10	4
	Double	5	5

For practical implementations, the cost-ratio of the $GF(2^n)$ multiplier to the $GF(2^n)$ squaring unit is over 7. Therefore, from the comparison of Table 7.1, the point arithmetic based on the Lopez-Dahab coordinate is the most efficient for implementations. This work is focused on the L-D point arithmetic.

7.2.2 Lopez-Dahab point arithmetic

In the Lopez-Dahab projective coordinate, the point (X, Y, Z) $(Z \neq 0)$ is corresponding to the point $(X/Z, Y/Z^2)$ in the affine coordinate, and the elliptic curve equation is transformed into the following form:

$$Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4. \tag{7.5}$$

The point addition formula that does not involve the inversion operation can be derived by converting the point to affine projective as $x = X/Z$ and $y = Y/Z^2$ at first, then adding the affine points with equation (7.2), and finally clearing the denominators. Similarly, the L-D point doubling equation can be derived by converting to affined projective, substituting into equation (7.3) and clearing the denominators.

It should be noted that when using the "double and add" method for the scalar multiplication *kP*, the point *P* is never modified in all the point addition operations. Therefore, the coordinate of *P* can be further reduced as $P = (X_1, Y_1, 1)$ to simplify the computation. As for two distinct points $P = (X_1, Y_1, 1)$ and $Q = (X_0, Y_0, Z_0)$ on the elliptic curve, the result $R = (X_2, Y_2, Z_2) = P + Q$ is computed by the following steps [49]:

$$\begin{array}{lll} 1.\ A \leftarrow X_1 Z_0 & 9.\ C \leftarrow A + C & 17.\ G \leftarrow Y_1 E \\ 2.\ B \leftarrow X_0 + A & 10.\ B \leftarrow B \cdot C & 18.\ F \leftarrow B + F \\ 3.\ A \leftarrow Z_0^2 & 11.\ E \leftarrow A^2\ (Z_2) & 19.\ G \leftarrow B + G \\ 4.\ C \leftarrow aA & 12.\ A \leftarrow A \cdot D & 20.\ F \leftarrow A \cdot F \\ 5.\ A \leftarrow Y_1 A & 13.\ B \leftarrow A + B & 21.\ G \leftarrow E \cdot G \\ 6.\ D \leftarrow Y_0 + A & 14.\ D \leftarrow D^2 & 22.\ D \leftarrow F + G\ (Y_2) \\ 7.\ A \leftarrow Z_0 B & 15.\ B \leftarrow B + D\ (X_2) & \\ 8.\ B \leftarrow B^2 & 16.\ F \leftarrow X_1 E & \end{array} \tag{7.6}$$

and $R = 2Q$ is computed as ($c = b^{1/2}$ is pre-computed, b is the coefficient in (7.5)):

$$\begin{array}{llll} 1.\ A \leftarrow Z_1^2 & 6.\ C \leftarrow C^2 & 11.\ D \leftarrow B + D \\ 2.\ B \leftarrow c \cdot A & 7.\ C \leftarrow B + C\ (X_2) & 12.\ D \leftarrow C \cdot D \\ 3.\ B \leftarrow B^2 & 8.\ D \leftarrow Y_1^2 & 13.\ B \leftarrow A \cdot B \\ 4.\ C \leftarrow X_1^2 & 9.\ E \leftarrow a \cdot A & 14.\ D \leftarrow B + D\ (Y_2) \\ 5.\ A \leftarrow A \cdot C\ (Z_2) & 10.\ D \leftarrow D + E & \end{array} \tag{7.7}$$

7.3 Fast Point Operation Architecture

For most of the existing point arithmetic implementations based on Lopez-Dahab projective coordinate (e.g., [53]), the algorithms discussed above were implemented in a sequential way. The advantage is that the number of finite field arithmetic modules can be reduced to minimum (for example, only one adder, one multiplier and one squaring unit are needed for point addition and doubling). However, such designs introduce long latency for performing the point operations

(the latency of the point doubling is $5T_M+5T_S+4T_A$, and the latency of the point addition is $10T_M+4T_S+8T_A$, where T_M, T_S, T_A denote the latency of the finite field multiplier, squaring unit, and adder respectively), which is not desirable for the applications where high-speed ECC implementation is required.

7.3.1 Fast point doubling architecture

A popular approach to increase the processing speed (reduce the latency) is to apply the parallel processing technique. By introducing more processing units which can operate in parallel, the results can be obtained much faster.

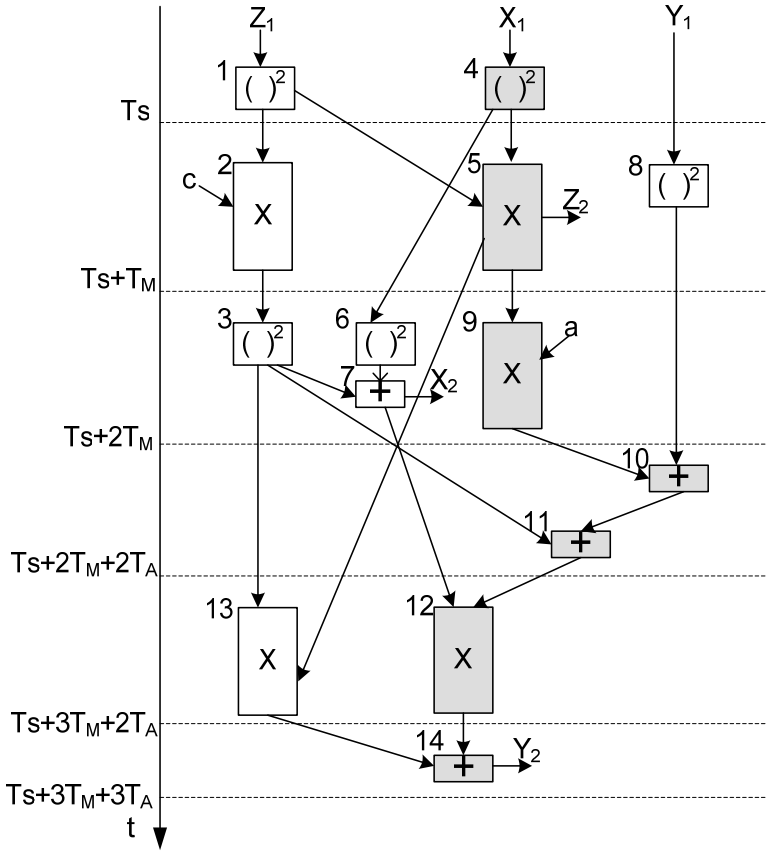

Figure 7.2. Parallel architecture for L-D point doubling.

In Figure 7.2, the parallel architecture of the Lopez-Dahab point doubling algorithm is shown. The number associated with each module corresponds to the step in (7.7). The critical path is indicated in grey color. We can see the total latency required to compute Y_2 is $3T_M+T_S+3T_A$. Assume the timing cost ratio of T_M to T_A is r_1 (usually around 15), T_S to T_A is r_2 (around 2), the total latency ratio of the serial architecture of point doubling to the corresponding parallel architecture is $(5r_1+5r_2+4)/(3r1+r_2+3)=1.78$ (when $r_1=15$ and $r_2=2$), which is the speedup we have achieved by applying the parallel architecture.

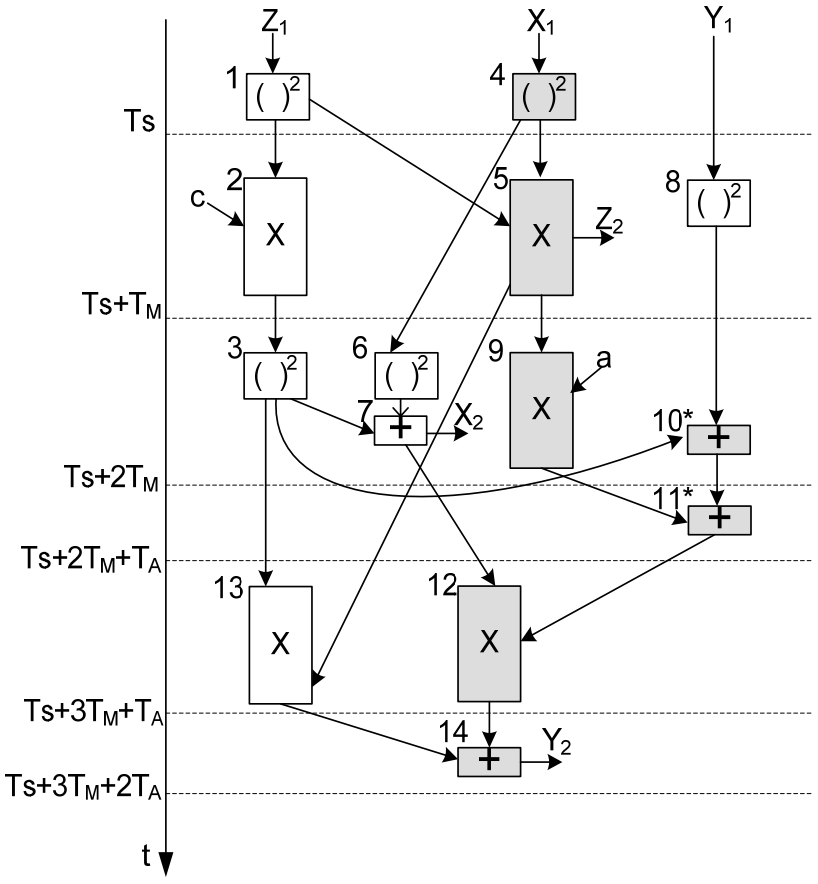

Figure 7.3. Modified parallel architecture for L-D point doubling.

Figure 7.3 shows the modified architecture of the parallel architecture for point doubling. In this architecture, the Steps 10 and 11 are modified, which is indicated as Steps 10* and 11*. This modification does not change the final result of Y_2 because the output of Step 11* is the same as Step 11, which equals the sum of the output of Steps 3, 8, and 9. However, by changing the order of these additions, Step 10* can be overlapped with multiplier 9, thus the critical path length becomes $3T_M+T_S+2T_A$, which is reduced by the computation delay of one adder, T_A. And the total speedup becomes 1.81 times.

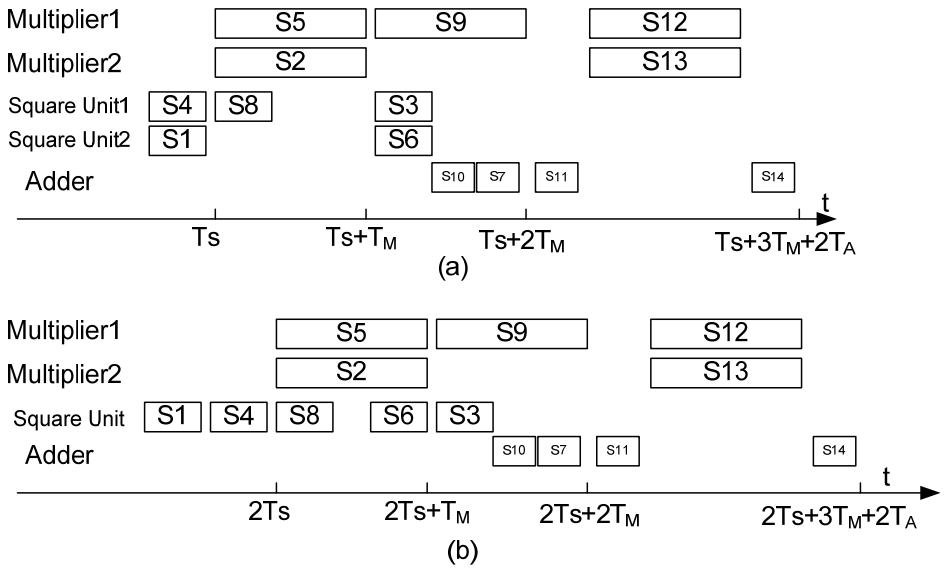

Figure 7.4. Timing schedule of the L-P point doubling.

Based on the above architecture, totally 5 multipliers, 5 squaring units, and 4 adders are used for the fully parallel architecture. However, this is not necessary. From Figure 7.3 we can see that at the same time instance, at most two multipliers are working in parallel. Similarly, at most two squaring units are needed at the

same time. If the point doubling implementation has been pipelined and well scheduled, we can use the schedule shown in Figure 7.4(a) to minimize the number of finite field arithmetic units, thereby reduce the hardware cost to two multipliers, two squaring units and one adder.

Moreover, by using the schedule shown in Figure 7.4(b), we can save another squaring unit. In this case, the latency is increased by T_S, which becomes $3T_M+2T_S+3T_A$, and the overall extra hardware is only one $GF(2^n)$ multiplier.

7.3.2 Fast point addition architecture

Figure 7.5 shows the parallel architecture for the Lopez-Dahab point addition arithmetic. Similarly, the number associated with each module corresponds to the step in equation (7.6). The critical path is indicated in grey color (note the Steps 19 and 21 can be regarded as part of the critical path instead of Steps 18 and 20). The total latency to calculate Y_2 is $4T_M+6T_A$. Using the same notation as the point doubling discussed above, the total latency ratio of the serial architecture of point addition over the parallel point addition architecture is $(10r_1+4r_2+8)/ (4r_1+6) =2.52$ (when $r_1=15$ and $r_2=2$), which is the speedup achieved by using the parallel architecture for point addition.

From Figure 7.5, we can see 4 multipliers working in parallel are needed for the point addition operation, and only one squaring unit is necessary. In addition, since Step 18 and Step 19 are executed in parallel, we need two adders for this architecture. However, we can delay the addition in Step 19 by T_A, which means Step 19 starts after Step 18 completes. In this way, only one finite field adder is needed. Thus, the total latency is increased by T_A. Considering the $GF(2^n)$ adder is

a number of XOR gates in parallel, whose latency is very small, the re-scheduling discussed above becomes meaningful.

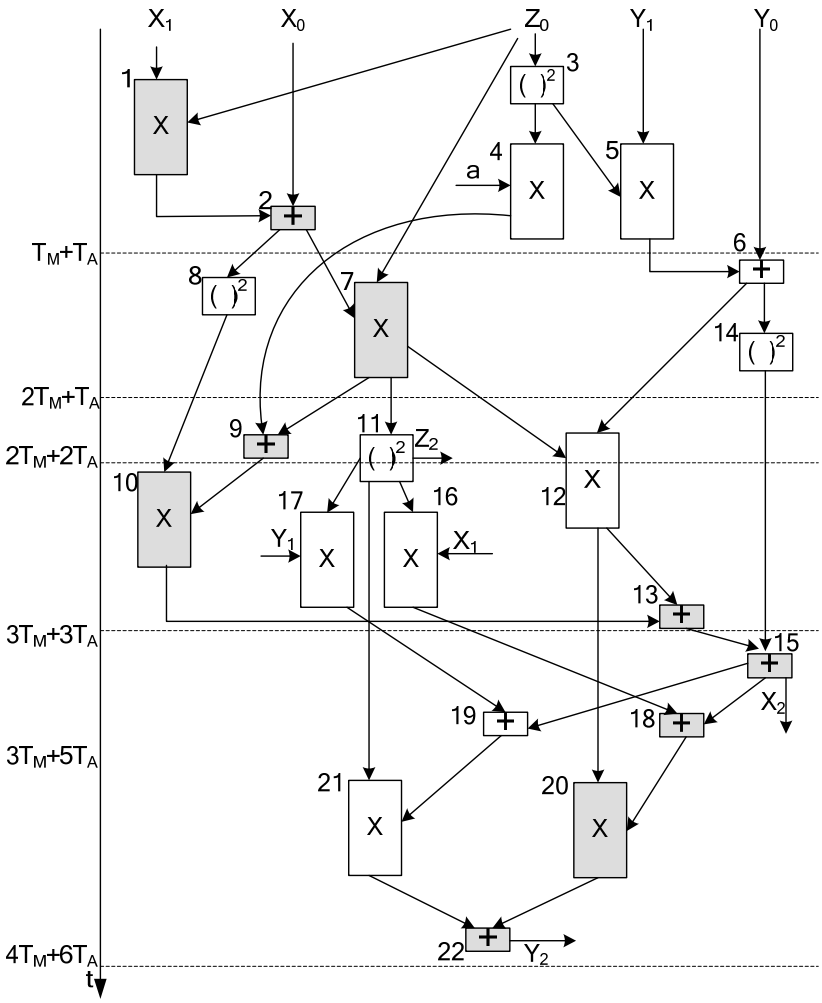

Figure 7.5. Parallel architecture for L-D point addition.

For an efficient L-D coordinate based point addition design, we can reduce the number of multipliers from 4 to 2. The method is to advance the multiplication of Step 1 by $T_M - T_S$, which makes Step 1 complete before Steps 4 and 5 begin.

Also, we need to delay the multiplications of Steps 16 and 17 so that they start after the completion of Step 10. For such a modified architecture, at most two multipliers are working in parallel at any time instance. Thereby the total number of multipliers is reduced to two. And the total latency becomes $6T_M+4T_A$, which offers another trade-off between the area and speed. When r_1=15 and r_2=2, the speedup achieved is 1.77. In this case, the major hardware overhead is one $GF(2^n)$ multiplier compared with the sequential implementation. The schedule for this modified scheme is shown in Figure 7.6.

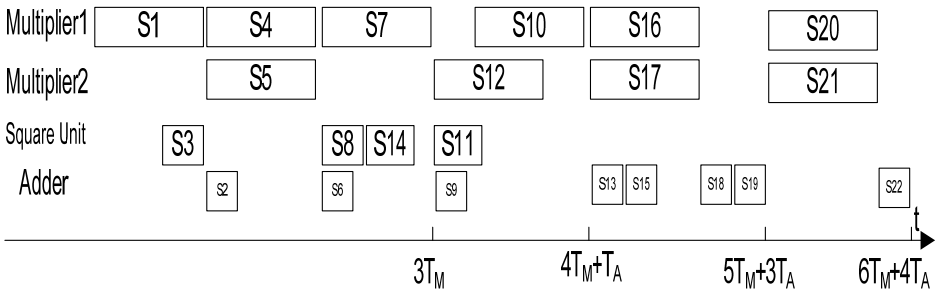

Figure 7.6. Timing schedule of the modified L-P point addition.

7.4 Conclusions

In this work, a fast architecture for the ECC point arithmetic (point doubling and point addition) based on the Lopez-Dahab projective coordinate is presented. This proposed architecture can significantly speed up the ECC computation with reasonable hardware overhead, which is essential for the applications where high-speed systems are required.

8 EFFICIENT ARCHITECURE FOR THE TATE PAIRING IN CHARACTERISTIC THREE

Identity-based cryptography (IBC) is a public-key cipher introduced by Shamir [62] in 1984. In IBC, the public key is derived from user identity (an arbitrary string) instead of from a relationship with private information as in conventional schemes, such as RSA. The corresponding private key is created by binding the user identity with a trusted third party, called private key generator's secret key. This system allows any party to encrypt messages or verify signatures with no prior knowledge on the public keys of other parties. This is extremely useful in cases where pre-distribution of authenticated keys is inconvenient or infeasible due to technical restraints. Moreover, such scheme presents a rich set of functional and security characteristics which are difficult or impossible to realize by other ciphers. Modern implementations of IBC include Cock's quadratic residues scheme [63], Bohen/ Franklin [64] and Sakai's [65] pairing schemes. The pairing schemes are based on bilinear mapping over elliptic curves and are faster for practical implementation than the quadratic residues scheme. There exist two types of bilinear mapping: the Weil pairing and Tate pairing. Among these two algorithms, the Tate pairing has lower computational cost. Nevertheless, its complexity is still very high. Normally a pairing operation takes much longer (about 5-10 times as long) to compute than the point multiplication in elliptic curve cryptography. Prior implementations of the Tate paring are mainly in software domain [67]. These implementations can only run at low speed due to the high complexity. In order to employ IBC in practical applications, hardware

implementations must be employed. A few previous efforts have been devoted to hardware implementation of Tate pairing. The notable ones include the FPGA hardware accelerator for Tate pairing over $GF(3^m)$ presented in [68]. In order to boost the speed of IBC to practical level, efficient and high-speed hardware implementations of Tate paring need to be developed.

The best method of Tate pairing calculation before 2002 was presented by Miller in [71]. In 2002, Galbraith [74] and Barreto [75] greatly simplified the pairing computation by introducing the triple-and-add BLKS algorithm in characteristic three. The BLKS algorithm was further modified and developed as the Duursma-Lee algorithm [69] and the Kwon-BGOS algorithm [70]. Through exploring the intrinsic property of the Duursma-Lee algorithm, we propose complexity-reducing schemes and an overlapped processing technique. Compared with conventional sequential implementations, the proposed architecture can achieve over 2 times speedup. The proposed method can be also applied to the Kwon-BGOS algorithm, and similar speedup can be obtained.

This Chapter is organized as follows: In Section 8.1, the Tate pairing and the Duursma-Lee & Kwon-BGOS algorithms are introduced. Section 8.2 presents the overlapping technique and its application to the Tate pairing algorithm. The processing gain analysis is also provided in this section. At the end, conclusions are drawn in Section 8.3.

8.1 Tate Pairing Algorithms

In this section, a brief introduction of the Tate pairing in characteristic three and the Duursma-Lee & Kwon-BGOS algorithms will be given.

8.1.1 Tata Pairing

Let $E(GF(q))$ be an elliptic curve defined over finite field $GF(q)$, where $q=3^m$ and m is a prime number. Let l be a positive integer coprime to q. In practice, l is usually picked as a large prime such that l divide $\#E(GF(q))$ and l^2 does not divide $\# E(GF(q))$. Here $\#E$ stands for the number of rational points on the elliptic curve. Let k be the smallest integer which satisfies $l \mid q^k-1$. Actually k is the embedding degree of the curve with respect to l. Let $GF(q^k)$ be the smallest extension field of $GF(q)$ which contains the l-th root of unity. Also, let $E[l](GF(q))$ denote the subgroup of $E(GF(q))$ of all points of order dividing l (l-torsion) and similarly for the degree k extension of $GF(q)$. Tate pairing of order l is defined in terms of rational functions over the points of an elliptic curve evaluated in a divisor. It is a bilinear mapping between $E[l](GF(3^m))$ and $E[l](GF(3^{km}))$ to the element of the multiplicative subgroup of $GF(3^{km})$, i.e., $GF(3^{km})^*$. Such a bilinear mapping can be denoted by:

$$E[l](GF(3^m)) \times E[l](GF(3^{km})) \rightarrow GF(3^{km})^* . \tag{8.1}$$

It is only defined up to l^{th} power of unity. The quotient group $GF(3^{km})^* / (GF(3^{km})^*)^l$ is isomorphic to the elements of order l in $GF(3^{km})^*$ by raise them to the power $(3^{km}-1)/ l$. Now consider the following super-singular elliptic curve on a finite field of characteristic 3:

$$E(GF(3^m)) : y^2 = x^3 - x + b, \quad b = \pm 1. \tag{8.2}$$

Similar to [68], we set the embedding degree k to 6. Consider $P=(x_1,y_1)$ and $Q=(x_2,y_2) \in E[l](GF(3^m))$, i.e., $x_1, y_1, x_2, y_2 \in GF(3^m)$. The pairing is efficiently computed in practice by considering the point $\phi(Q) \in E[l](GF(3^{6m}))$, where ϕ is a distortion map of (8.2). The distortion map is defined as

$$\phi(Q) = \phi((x_2, y_2)) = (\rho - x_2, \sigma y_2), \qquad (8.3)$$

where $\rho, \sigma \in GF(3^{6m})$ such that $\rho^3 - \rho - b=0$ and $\sigma^2+1=0$. And $GF(q^6)$ is the quadratic extension $GF(q^6) = GF(q^3)[\sigma] / [\sigma^2+1]$, $GF(q^3)= GF(q)[\rho]/ [\rho^3 - \rho - b]$.

8.1.2 Duursma-Lee & Kwon-BGOS algorithms

Duursma and Lee introduced in [69] a faster Tate pairing algorithm using a group of order $l = q^3 + 1 = 3^{3m} +1$ instead of an order that divides $q^3 + 1$. In this case, a closed formula for the mapping of equation (8.1) was found. The pseudo codes for the Duursma-Lee algorithm are shown in Algorithm 1.

Algorithm 1. Duursma-Lee algorithm for calculating Tate pairing in characteristic three

Input : point $P=(x_1,y_1)$, $Q=(x_2,y_2)$

Output: $f = f_P(\phi(Q)) \in GF(q^6)^* / GF(q^3)^*$

1. **begin**
2. $f \leftarrow 1$
3. **for** i =1 **to** m **do**
4. $x_1 \leftarrow x_1^3$
5. $y_1 \leftarrow y_1^3$
6. $\mu \leftarrow x_1 + x_2 + b$
7. $\lambda \leftarrow -y_1 y_2 \sigma - \mu^2$
8. $g \leftarrow \lambda - \mu\rho - \rho^2$
9. $f \leftarrow f \cdot g$
10. $x_2 \leftarrow x_2^{1/3}$
11. $y_2 \leftarrow y_2^{1/3}$
12. **end for**
13. **return** f
14. **end**

Another widely used fast Tate pairing algorithm is the Kwon-BGOS algorithm. The details of this algorithm are given in Algorithm 2.

Algorithm 2. Kwon-BGOS algorithm for calculating Tate pairing in characteristic three

Input : point $P=(x_1,y_1)$, $Q=(x_2,y_2)$

Output: $f = f_P(\phi(Q)) \in GF(q^6)^* / GF(q^3)^*$

1. **begin**
2. $f \leftarrow 1$
3. $x_2 \leftarrow x_2^3$
4. $y_2 \leftarrow y_2^3$
5. $d \leftarrow mb$ **mod 3**
6. **for** i =1 **to** m **do**
7. $x_1 \leftarrow x_1^9$
8. $y_1 \leftarrow y_1^9$
9. $\mu \leftarrow x_1 + x_2 + d$
10. $\lambda \leftarrow -y_1 y_2 \sigma - \mu^2$
11. $g \leftarrow \lambda - \mu\rho - \rho^2$
12. $f \leftarrow f^3 \cdot g$
13. $y_2 \leftarrow -y_2$
14. $d \leftarrow (d-b) \bmod 3$
15. **end for**
16. **return** f
17. **end**

As discussed in [69][70], to obtain the compatible result with that for the BKLS algorithm [75], the output of algorithm 1 and 2 should be powered to $(3^{6m}-1)/ l=3^{3m}+1$.

8.2 Efficient Tate Pairing Architecture

It can be observed from Algorithm 1 and 2 that both the Duursma-Lee and Kwon-BGOS algorithms for Tate pairing algorithms are very complicated. The major computations in these algorithms are finite field addition/subtraction,

multiplication, cubing and cube root. The speed can be achieved by these algorithms is limited by the *m* iterative loops.

8.2.1 Efficient arithmetic over finite fields of characteristic 3

In $GF(3^m)$, using polynomial basis representation for field elements leads to faster implementation of finite field operations than using normal basis representation [68]. An element in $GF(3)$ can be represented by a digit in the set {-1, 0,1}, which can be encoded into two bits with the most significant bit as the sign bit (10 for -1, 00 for 0, 01 for 1). The operations in $GF(3)$ can be easily implemented by simple combinational logic, or 4-input look-up-tables. Moreover, the finite field $GF(3^m)$ is isomorphic to $GF(3)/f(x)$, where $f(x)$ is a irreducible polynomial of degree *m* over GF(3). Therefore, each element of $GF(3^m)$ can be represented by an *m*-dimension vector over $GF(3)$, i.e., $2m$ bits binary vector.

The addition/subtraction over $GF(3^m)$ can be realized digit-wise with *m* adders in $GF(3)$, which can be completed within one clock cycle. As mentioned above, the sign of element in $GF(3)$ can be inverted by swapping the bits. The architecture of GF(3) adder/subtracter is show in Figure 8.1. $GF(3^m)$ adder is *m* such adder in parallel.

The GF(3) adder logic is defined as: $t = (a_1 | b_2) \wedge (a_2 | b_1)$, $c_1 = (a_2 | b_2) \wedge t$, $c_2 = (a_1 | b_1) \wedge t$.

Using polynomial basis representation, the cubing in $GF(3^m)$ can be implemented by inserting one zero between each adjacent coefficient then applying modulo reduction by f(x) to the corresponding polynomial. This process can be also

completed in one clock cycle. As for the 9th power (Steps 7&8 of algorithm two), the operation can be done with two cubing units.

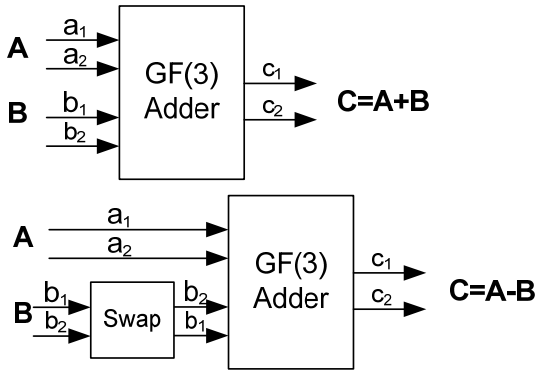

Figure 8.1. *GF*(3) adder/subtracter unit.

The cubic root function can be realized using the method introduced by Barreto [72], as long as that the field polynomial $f(x) = x^m + x^w - 1$ satisfies $m \equiv w \pmod 3$. In this case, only bit addition and shift are involved.

The most complicated units in the Tate pairing algorithms are the multiplications, which includes the multiplications in $GF(3^m)$ and $GF(3^{6m})$ (Steps 7, 8, 9 in Algorithm 1, and steps 10, 11, 12 in Algorithm 2). The multiplication of two elements *A, B* in $GF(3^m)$ can be performed in a digit-serial way: all bits in operand A are processed in parallel, while operand *B* is split into $\lceil m / D \rceil$ groups (each group has *D* digits) and processed serially. Using this digit-serial computation, each multiplication takes $\lceil m / D \rceil$ clock cycles, and the critical path consists of one *m* by *D* digits finite field multiplier. The latency of this multiplier is $t_m = \lceil m / D \rceil \cdot t_{mD}$, where t_{mD} is the propagation delay of the m by D digit finite field multiplier, and is

also the critical path of the whole Tate pairing module. The multiplication in $GF(3^{6m})$ can be decomposed into operations in $GF(3^m)$ by using composite field arithmetic. According to Karatsuba's work [73], a $GF(3^{6m})$ multiplier can be implemented by 56 adders and 18 $GF(3^m)$ multipliers connected in parallel. Hence, the latency for this unit is $t_{6m} = t_m + t_a$, where t_a stands for the latency introduced by those adder arrays. The block diagram is shown in Figure 8.2.

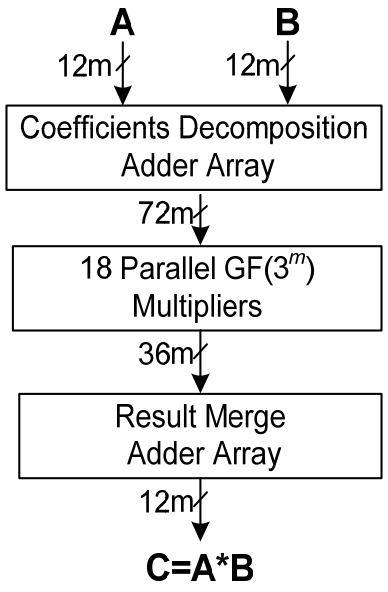

Figure 8.2. Block diagram of the $GF(3^{6m})$ multiplier.

Modulo reduction is required in Steps 5 and 14 of Algorithm 2. Considering $16 \equiv 1 \pmod 3$, Step 5 operation can be implemented in an efficient way as shown in Figure 8.3(a):

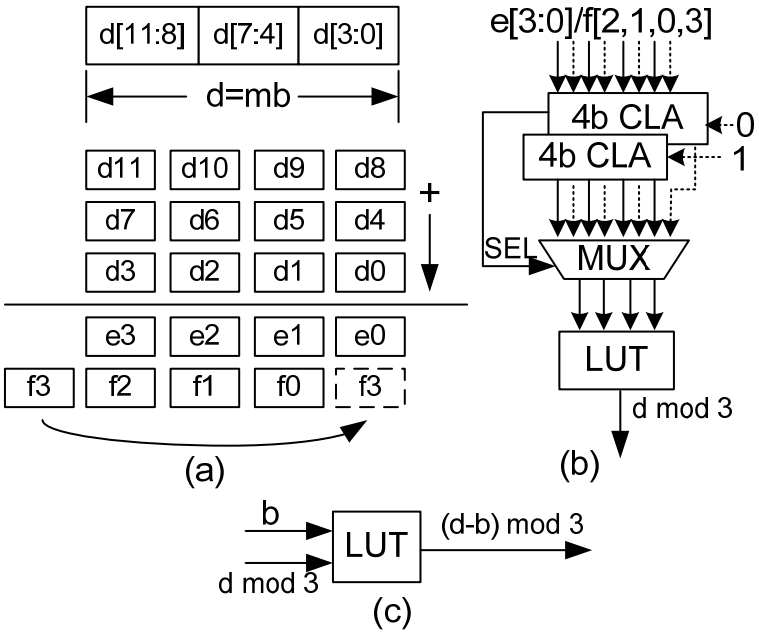

Figure 8.3. Fast mod 3 architecture.

Assume d has 12 bits and they are divided into groups of 4 bits: d[11:8], d[7:4] and d[3:0] (Other length of d can follow similar approach. For instance, if d has 15 bits, it can be grouped into d[14:12], d[11:8], d[7:4] and d[3:0]). Then $d=d$[11:8]*256+d[7:4]*16+d[3:0]. Hence, $d \equiv d$[11:8]+d[7:4]+ d[3:0] (mod 3). This addition can be implemented by 3-to-2 compressor shown in Figure 3(a). Each digit in the results e and f, has 4 bits. Since 16*$f3 \equiv f3$ (mod 3), the bit $f3$ can be moved to the least significant digit (LSD) position. Therefore, $d \equiv e$[3:0]+f[2,1,0,3] (mod 3). To speed up the addition, we use two parallel 4-bit carry look-ahead adders (CLA) to carry out the addition as shown in Figure 8.3(b). If $e+f$ has no carry out, the result of the top CLA is selected by the MUX. Otherwise, the result of the bottom CLA ($e+f+1$) is selected. The 4-digit output from the CLA is then sent to a 4-input look-up-table to obtain the $(d \bmod 3) \in GF(3)$ result, which is represented by 2 bits.

The modulo reduction in Step 14 can be implemented in an easier way. Since $d \in \{-1,0,1\}$ and $b = \pm 1$, a 3-input look-up-table can be used for this operation, as shown in Figure 8.3(c), e.g., when $d = -1, b = 1$, it outputs $(d - b) \bmod 3 = 1$.

8.2.2 Algorithmic simplifications

Following the Duursma-Lee algorithm directly, the implementation of Step 7 needs two $GF(3^m)$ multipliers, one $GF(3^{6m})$ multiplier, and one $GF(3^{6m})$ adder. In addition, Step 8 needs another $GF(3^{6m})$ multiplier and two $GF(3^{6m})$ adders. These computations have very high hardware complexity.

It can be derived that Steps 7 and 8 can be combined into one step by using an approach similar to that proposed in [76]:

$$g \leftarrow -y_1 y_2 \sigma - \mu^2 - \mu\rho - \rho^2 \ . \tag{8.4}$$

Using the property of the distortion mapping: $\rho, \sigma \in GF(3^{6m})$ such that $\rho^3 - \rho - b=0$ and $\sigma^2+1=0$, and the field extension $GF(3^{6m}) = GF(3^{3m})\ [\sigma] \ / \ [\sigma^2+1]$, $GF(3^{3m}) = GF(3^m)\ [\rho] / \ [\rho^3 - \rho - b]$, the polynomial basis of $GF(3^{6m})$ can be expressed as

$$\{\beta^0, \beta^1, \beta^2, \beta^3, \beta^4, \beta^5\} = \{1, \sigma, \rho, \sigma\rho, \rho^2, \sigma\rho^2\} . \tag{8.5}$$

By using this basis, (8.4) can be re-written as

$$g \leftarrow -y_1 y_2 \sigma - \mu^2 - \mu\rho - \rho^2 = (-\mu^2)\beta^0 + (-y_1 y_2)\beta^1 + (-\mu)\beta^2 + (0)\beta^3 + (-1)\beta^4 + (0)\beta^5 \tag{8.6}$$

Accordingly, Steps 7 and 8 can be merged into a single step, which only requires two $GF(3^m)$ multiplications. Compared with the conventional implementation, this modified approach saved two $GF(3^{6m})$ multiplier (which equals 18 $GF(3^m)$ multipliers and 56 $GF(3^m)$ adders), and two $GF(3^{6m})$ adders (which is equivalent to 6 $GF(3^{6m})$ adders each).

8.2.3 Fast Tate pairing architecture

It can be observed from Algorithm 1, the Duursma-Lee algorithm consists of the initialization phase (Steps 1-3) and the m-iteration loop phase (Steps 4-11). The loop phase accounts for the major part of the overall complexity. Speeding up the loop phase is critical to achieve high speed implementation of the Tate paring. In previous software and hardware co-processor design [68], the computations in the loop are carried out serially, as shown in Figure 8.4(a). The serial computation can only achieve very limited throughput. In order to increase the throughput to next higher level, novel schemes need to be developed to achieve parallel computation.

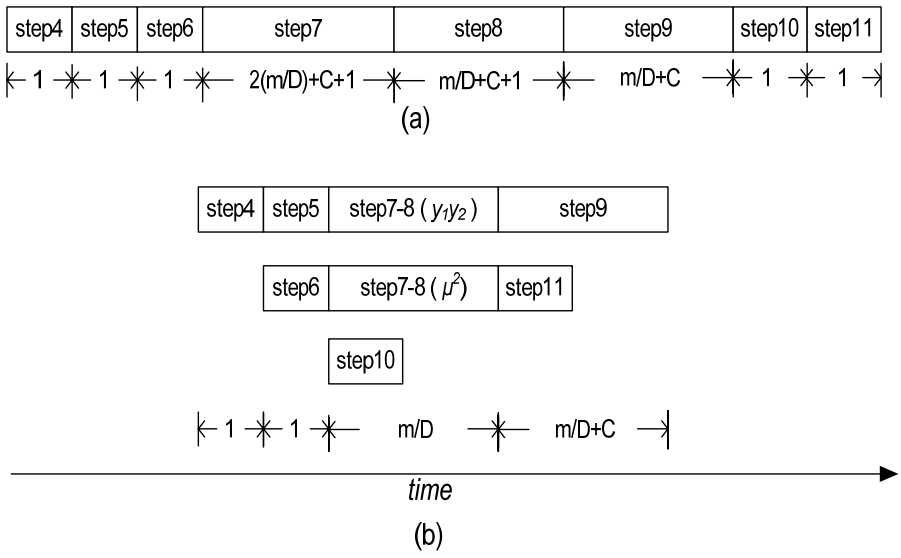

Figure 8.4. (a). Conventional processing scheme for the Duursma-Lee algorithm. (b) Overlapped processing scheme.

We propose to schedule the computations in the Duursma-Lee algorithm in an over-lapped manner. Our proposed scheduling scheme is illustrated in Figure 8.4(b). The major modifications made in our scheme are as follows:

(a) As discussed earlier, Step 7 and Step 8 are merged as Step 7-8 to reduce complexity and increase speed.

(b) Since there is no data dependency between Step 5 and 6, they can be carried out simultaneously. Accordingly, one clock cycle processing time can be saved. Step 5 only needs the result from Step 4. Hence Step 5 can start right after Step 4. In addition, the cubing unit can be shared by Step 4 and 5 in a time-multiplexed way. Hence, only one cubing unit is needed.

(c) Steps 10 and 11 are independent of Step 9 and the merged Step 7-8. In addition, Step 7-8 and Step 9 involve the finite field multiplication, which usually takes much longer time than the cubic root operation (simplified as the finite field addition and shifting). For this reason, Step 10 and 11 can be carried out simultaneously with Step 7-8 and 9, respectively. Similarly, Step 10 and Step 11 are carried out serially. Hence one cube root unit can be shared by these two steps.

8.2.4 Speed analysis and comparison

Table 8.1 and 8.2 shows the number of clock cycles needed for each iteration of Duursma-Lee algorithm by using sequential processing and our proposed overlapped processing, respectively. *C* stands for the extra clock cycles introduced by the extra adders in the Karatsuba decomposition of multiplication in $GF(3^{6m})$ (normally when $D>=5$, $C=1$.When $D<=4$ $C=2$). From Table 8.2, it can be observed that the sequential implementation takes $7+3C+4*\lceil m/D \rceil$ cycles per iteration. Assume $m=97$ and $D=8$, which are typical values for IBC. The sequential processor

took 62 clock cycles per iteration, and it takes 97*62=6014 clock cycles in total. Comparatively, using our proposed overlapped scheme, $2*(1+\lceil m/D \rceil)+C = 29$ cycles are needed per iteration and only 97*29=2813 cycles are required in total. Therefore, our proposed overlapped scheme can achieve a speed up of 6014/2813=2.13. In addition, no extra hardware is introduced by our proposed scheme.

TABLE 8.1 NUMBER OF CLOCK CYCLES FOR ONE ITERATION OF THE DUURSMA-LEE ALGORITHM (SEQUENTIAL PROCESSING)

Step	Operation	Clock cycles
4	$x_1 \leftarrow x_1^3$	1
5	$y_1 \leftarrow y_1^3$	1
6	$\mu \leftarrow x_1 + x_2 + b$	1
7	$\lambda \leftarrow -y_1 y_2 \sigma - \mu^2$	$2*\lceil m/D \rceil+C+1$
8	$g \leftarrow \lambda - \mu\rho - \rho^2$	$\lceil m/D \rceil+C+1$
9	$f \leftarrow f \cdot g$	$\lceil m/D \rceil+C$
10	$x_2 \leftarrow x_2^{1/3}$	1
11	$y_2 \leftarrow y_2^{1/3}$	1

TABLE 8.2. NUMBER OF CLOCK CYCLES FOR ONE ITERATION OF THE DUURSMA-LEE ALGORITHM (OVERLAPPING PROCESSING)

Step	Operation	Clock cycles
4	$x_1 \leftarrow x_1^3$	1
5, 6	$y_1 \leftarrow y_1^3$, $\mu \leftarrow x_1 + x_2 + b$	1
7-8, 10	$g \leftarrow -y_1 y_2 \sigma - \mu^2 - \mu\rho - \rho^2, x_2 \leftarrow x_2^{1/3}$	$\lceil m/D \rceil$
9, 11	$f \leftarrow f \cdot g$, $y_2 \leftarrow y_2^{1/3}$	$\lceil m/D \rceil+C$

Similarly the overlapped processing scheme can be applied to the Kwon-BGOS algorithm as shown in Figure 8.5. It can be derived that similar speedup can be also achieved in this case without introducing extra hardware requirement.

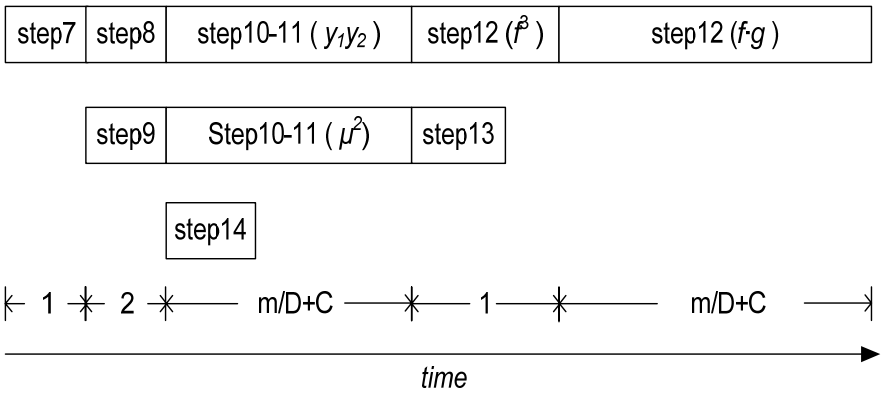

Figure 8.5. Overlapped processing scheme for the Kwon-BGOS algorithm.

8.3 Conclusions

In this Chapter, efficient computation architectures and an overlapped processing scheme have been proposed for the Tate pairing over finite field of characteristic three. The proposed schemes can significantly accelerate the modern Tate pairing algorithms such as Duursma-Lee and Kwon-BLGS algorithms without introducing hardware overhead, which are very useful for the dedicated hardware implementation of the Identity Based Cryptography.

9 CONLUSIONS

In this research, we have proposed high-speed low-complexity solutions to address various VLSI architecture issues for the MIMO and cryptography systems; mainly focus on the MIMO sphere decoder designs (including K-Best SD, conventional SD and list sphere decoder), elliptic curve cryptography and Tate pairing. The main contributions are summarized as follows.

For the K-Best sphere decoder designs for MIMO systems, we have three main contributions.

First, we apply the sorted QR decomposition and developed the layer reordering K-Best SDA. Moreover, we introduce the dynamic K-Best SDA, which applies different K values at each decoding layer. Both methods can significantly improve the detection performance or reduce the design complexity. Simulation results showed that for our example 4x4 64QAM MIMO system, about 30% complexity reduction can be achieved by utilizing these methods.

Second, by exploiting the natural partial sorted results coming from the SE enumeration, we derive an efficient sorting architecture based on Batcher's merge sort algorithm. Compared with the bubble sorting algorithm, our sorting architecture saves around 50% sorting complexity, which effectively solves the bottleneck of the K-Best SDA.

Third, in Chapter 5 we present the early-pruning K-Best sphere decoder scheme. Without sacrificing detection performance, about 55% power consumption or computational complexity can be saved by eliminating the survival candidates

with relatively large partial Euclidian distances, which are unlikely to become the ML solution when the tree searching reaches the bottom layer.

Meanwhile, for the conventional sphere decoder, we first propose a parallel SD architecture which splits the constellation tree into two sub-trees. Thereby the two processing engines can conduct depth-first search and update new radius in parallel. Moreover, by exploiting the similarity and interleaving the data streams for both processing engines, we introduce the pipeline interleaved SD architecure, in which only one PE is needed with very small interleave control logics. Simulations show that the new architecture achieves an average throughput speedup of 44% with negligible hardware overhead compared with the conventional sphere decoders.

In Chapter 6, we introduce some new schemes for list sphere decoder. We first change the decoding flow to do the radius updating before list updating which will avoid the latency introduced by the radius updating unit. We also proposed a fast list updating architecture based on the merging of two partially ordered arrays which can compute the new radius in one clock cycle. Finally, we have presented an efficient candidate list updating architecture based on the merge sort and binary insertion. By reusing the merge unit in radius updating, this architecture can achieve a 45% complexity saving compared with tree-type comparator without affecting the decoding speed.

For the elliptic curve cryptography, in Chapter 7, a fast point operation architecture on the Lopez-Dahab projective coordinate is introduced. By applying parallel processing and hardware reusing, compared with the conventional point operation implementations, our architecture can achieve a speedup of 2.5 times for the point addition operation and 1.8 times for the point doubling operation with

reasonable hardware overhead, which facilitates the design of high-speed ECC systems.

Finally in Chapter 8, efficient computation architectures and an overlapped processing scheme are proposed for the Tate pairing over finite field of characteristic three. Compared with conventional sequential implementations, the proposed architecture can significantly accelerate (achieve over 2 times speedup) the modern Tate pairing algorithms such as Duursma-Lee and Kwon-BLGS algorithms without introducing extra hardware complexity, which are very useful for the dedicated high-speed hardware implementation of the identity based cryptography.

BIBLIOGRAPHY

[1] G. J. Foschini, "Layered space-time architecture for wireless communication in fading environment when using multiple antennas," *Bell Labs. Tech. Journal*, vol.2, Autumn, 1996.

[2] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Comput.,* vol. 44, pp. 463-471, April 1985.

[3] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. on Inform. Theory*, vol. 45, no. 5, pp. 1639-1642, July 1999.

[4] C. P. Schnorr and M. Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems", *Mathematical Programming*, vol. 66, no. 2, pp. 181-191, September 1994.

[5] J. Anderson and S. Mohan, "Sequential coding algorithms: a survey and cost analysis", *IEEE Trans. on Comm*. no. 2, vol. 32, pp. 169-176, Feb 1984.

[6] K. W. Wong, C. Y. Tsui, R. S. K. Cheng, and W. H. Mow, "A VLSI archetecture of a K-best lattice decoding algorithm for MIMO channels," *IEEE ISCAS 02,* vol. 3, pp. 273-276, 2002.

[7] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K. D. Kammeyer, "Efficient algorithm for decoding layered space-time codes," *Electronics letters*. no. 22, vol. 37, pp. 1348-1540, Oct. 2001.

[8] B. Hochward, and S. Brink, "Achieving near-capacity on a multiple antenna channel," *IEEE Trans. Inform. Theory*, vol. 51, no. 8, pp. 389-399, Mar. 2003.

[9] Zhan Guo, and P. Nilsson, "A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems," *Proc. IEEE CAS Symposium on Emerging Tech.*, pp. 65-68, June 2004.

[10] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE Jounal of Solid State Circuits*, Nov. 2004.

[11] B. Widdup, G. Woodward, and G. Knagge, "A highly-parallel VLSI architecture for a list sphere detector," *IEEE Inter. Conf. on Comm,*, vol. 5, pp. 2720-2725, June 2004.

[12] Wanlun, Zhao and G. B. Giannakis, "Reduced Complexity Closest Point Decoding Algorithms for Random Lattices," *IEEE Transactions on Wireless Communications,* vol. 5, issue 1, pp. 101-111, 2006.

[13] A. M. Chan and I. Lee, "A new reduced complexity sphere decoder for multiple antenna systems", *Proceedings of IEEE International Communications Conference*, April 2002.

[14] A. Wiesel, X. Mestre, A. Pages, and J. R. Fonollosa, "Efficient implementation of sphere demodulation", *IEEE Workshop on Sign. Proc. Advan. in Wireless Comm*., pp. 36-40, 2003.

[15] K. Parhi, *VLSI Digital Signal Processing Systems Design and Implementation,* John Wiley & Sons, 1999.

[16] Qingwei Li and Zhongfeng Wang, "Improved K-Best sphere decoding algorithms for MIMO systems.," *Proc. IEEE International Symposium on Circuits and Systems,* Kos, Greece, 2006.

[17] J. D. Parsons, *Mobile Radio Propagation Channel*, 2nd Edition, Wiley, 2000.

[18] D. Gesbert, M. Shafi, D. Shiu, P. Smith and A. Naguib, "From theory to practice: an overview of MIMO space-time coded wireless systems," *IEEE Journal on Selected Areas in Commun.*, vol. 21, pp. 281-302, April, 2003.

[19] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closet point search in lattices", *IEEE Trans. Inf. Theory,* vol. 48, no. 8, pp. 2201-2214, Aug. 2002.

[20] Jin Jie, Chi-ying Tsui and Wai-Ho Mow, "A threshold-based algorithm and VLSI architecture of a K-best lattice decoder for MIMO systems", *Proc. IEEE ISCAS 2005*, pp. 3359-3362, vol. 4, May 2005.

[21] Zhan Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection", *IEEE Journal on Selected Area in Communications*, vol 24, issue 3, pp. 491-503, March 2006.

[22] Qingwei Li and Zhongfeng Wang, "An improved K-Best sphere decoding architecture for MIMO systems." *Proc. 40th IEEE Asilomar Conf. on Signals, Systems, and Computers*, Asilomar, CA, 2006.

[23] B. Hassibi and H. Vikalo, "on the sphere-decoding algorithm i. expected complexity", *IEEE Trans. on Signal Processing*, vol. 53, pp. 2806-2818, Aug 2005.

[24] Y. Zhang, J. Tang and K.K. Parhi, "Low-Complexity List Updating Circuits for List Sphere Decoders", *Proc. 2006 IEEE Workshop on Signal Processing Systems*, pp. 28-33, Banff, Canada, Oct. 2006.

[25] D. Garrett, L. Davis, S. ten Brink, and B. Hochwarld, "APP processing for high performance MIMO systems,", *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 271-274, San Jose, CA, Sep. 2003.

[26] Qingwei Li and Zhongfeng Wang, "New sphere decoding architecture for MIMO Systems." *Proc. IEEE 13th NASA Symposium on VLSI Designs,* Post Falls, Idaho, June 2007.

[27] A. Burg, N. Felber, and W. Fichtner, "A 50 mbps 4x4 maximum likelihood decoder for multiple-input multiple-output systems with QPSK modulation," *Proc. 10th IEEE Int. Conf. Electron., Circuits and Systems, (ICECS),* pp. 322-325, Dec. 2003.

[28] G. Rekaya and J.-C. Belfiore, "On the complexity of ML lattice decoders for the decoding of linear full rate space-time codes," *Proc. IEEE International Symp. on Info. Theory,* pp. 206-208. June 2003.

[29] S. Baro, J. Hagenauer, and M. Witzke, "Iterative detection of MIMO transmission using a list-sequential (liss) detector," *Proc. IEEE Int. Conf. Commun.* pp. 2653-2657, 2003.

[30] Y. L. de Jong and T. J. Willink, "Iterative tree search detection for MIMO wireless systems," *Proc. IEEE 56th Veh. Tech. Conf.* pp. 1041-1045, Sept. 2002.

[31] Qingwei Li and Zhongfeng Wang, "Reduced Complexity K-Best Sphere Decoder Design for MIMO Systems." under review at *Journal on Circuits, Systems, and Signal Processing* (will be accepted after minor revision).

[32] D. L. Ruyet, T. Bertozzi and B. Ozbek, "Breadth first algorithms for APP detectors over MIMO channels," *Proc. IEEE Int. Conf. Commun.*, pp. 926-930, June 2004.

[33] B. Hassibi, "An efficient square-root algoruthm for BLAST," *Online Bell lab report*, http://mars.bell-labs.com/

[34] P. A. Bengough and S. J. Simmons, "Sorting-based VLSI architecture for the M-algorithm and T-algorithm trellis decoders," *IEEE Trans. on Commun.*, vol. 43, pp. 514-522, 1995.

[35] M. O. Damen, H. E. Gamal, and N. C. Beaulieu, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. on Info. Theory,* Vol. 49, no. 10, pp. 2372-2388, Oct. 2003.

[36] D. Garrett, L. Davis, S. ten Brink, B. Hochwald, and G. Knagge, "Silicon complexity for maximum likelyhood MIMO detection using spherical decoding," *IEEE Jounal on Solid-State Circuits,* vol. 39, no. 9, pp. 154401552, Sept. 2004.

[37] Qingwei Li and Zhongfeng Wang, "Early-Pruning K-Best sphere decoder for MIMO systems." *Proc. IEEE Workshop on Signal Processing Systems SiPS 2007*, Shanghai, Oct. 2007.

[38] A. Paulraj, R.Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications*, New York, Cambridge Univ. Press, 2003.

[39] J. Jalden and B. Ottersten, "An exponential lower bound on the expected complexity of sphere decoding," *Proc. IEEE ICASSP*, vol. 4, pp. 393-396, May 2004.

[40] M. O. Damen, H. El Gamal, and G. Caire, "On maximum likelihood detection and the search for the closest lattice point," *IEEE Trans. on Info. Theory,* vol. 49, no. 10, pp. 2389-2402, Oct. 2003.

[41] D. Garrett, G. Woodward, L. Davis, G. Knagge, and C. Nocol, "A 28.8 Mb/s 4x4 MIMO 3G high-speed downlink packet access receiver with normalized least mean square equalization," *IEEE ISSCC Dig. Tech. Papers,* vol. 1, p. 420, Feb. 2004.

[42] Qingwei Li and Zhongfeng Wang, "Efficient Radius and List Updating Units Design for List Sphere Decoders." submitted to *IEEE International Symposium on Circuits and Systems ISCAS 2008*, Seattle, May 2008.

[43] M. Borgmann and H. Bölcskei, "Efficient matrix inversion for linear MIMO-OFDM receivers," *Proc. 38th IEEE Asilomar Conf. Signals Syst, Comput.* , Pacific Grove, CA, Nov. 2004.

[44] Lincoln D. Stein, *Web security*, Addison-Wesley, <assachusetts, 1997.

[45] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM 21* 1978, pp. 120-126.

[46] Victor S. Miller, "Use of elliptic curves in cryptography", *in Advances in Cryptology CRYPTO'85*, pp. 417-426, New York, Springer-Verlag, 1986.

[47] Neal Koblitz, "Elliptic cure cryptosystems," *Mathematics of Computation*, vol. 48, no. 188, pp. 203-209, 1987.

[48] V. Gupta, S. Gupta, S. Chang and D. Stabila, "Performance analysis of elliptic curve cryptography for SSL", *3rd ACM workshop on wireless security*, pp. 87-94, 2002.

[49] J. Lopez and R. Dahab, "Improved algorithm for elliptic curve arithmetic in $GF(2^n)$." *Selected Areas in Cryptography-SAC'98, LNCS 1556*, pp. 201-212, 1999.

[50] IEEE Standard P1363-2000, "IEEE standard specifications for public-key cryptography", Aug, 2000.

[51] G. Agnew, R. Mullin, and S. Vanstone, "On the development of a fast elliptic curve processor chip", *Advances in Cryptology CRYPTO'91,* pp. 482-487, New York, Springer-Verlag, 1991.

[52] D. V. Chudnovsky and G. V. Chudnovsky, "Sequences of numbers generated by addition in formal groups and new primality and factorization tests." *Advances in Applied Mathematics,* vol.7, no. 4, pp. 385-434, May 1986.

[53] Y. B. Wang, X.J. Dong, and Z.G. Tian, "FPGA based design of elliptic curve cryptography coprocessor," *IEEE 3rd International Conference on Natural Computation, ICNC* 2007.

[54] M. Morales-Sandoval and C. Feregrino-Uribe, "On the hardware design of an elliptic curve cryptosystem", *Proceedings of the 5th Mexican International Conference in Computer Science (ENC'04)*, pp. 64-70, 2004.

[55] Jian Huang, "FPGA implementations of elliptic curve cryptography and Tate pairing over binary field," *M.S. Thesis*, Univ. of North Texas, 2007.

[56] D. Hankerson, L. Lopez, and A. Menezes, "Software implementation of elliptic curve cryptography over binary fields", *Proc. Of the 2nd International Workshop on Cryptographic Hardware and Embedded Systems, CHES' 2000, vol. 1965 LNCS*, pp.1-24 August 2000, Springer-Verlag.

[57] K. Lauter, "The advantages of elliptic curve cryptography for wireless security," *IEEE Wireless Communications,* pp. 62-67, 2004.

[58] A. Satoh and K. Takano. "A scalable dual-field elliptic curve cryptographic processor", *IEEE Transactions on Computers,* vol. 52, no. 4, pp. 449-460, April 2003.

[59] A. Weimerskirch, D. Stebila, and S.C. Shantz, "Generic GF(2^m) arithmetic in software and its application to ECC." *The 8th Australasian Conference on Information Security and Privacy (ACISP 2003)*, vol. 2727 LNCS, pp. 79-92, 2003.

[60] Julio Lopez and Ricardo Dahab, "An overview of elliptic curve cryptography", *Technical report,* Institute of Computing, State University of Campinas, Brazil, May 2000.

[61] Qingwei Li and Zhongfeng Wang, "Fast point operation architecture for elliptic curve cryptography." submitted to *GLSVLSI* 2008. Florida, May 2008.

[62] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes." *In Advances in Cryptology (CRYPTO)*, Springer-Verlag, LNCS 196 pp. 47-53, 1985.

[63] C. Cocks, "An identity-based encryption scheme based on quadratic residues." *Cryptography and Coding*, LNCS, 2260. pp. 360-363, 2001.

[64] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing." *Advances in Cryptology (CRYPTO) LNCS* 2139, pp. 213-229, 2001.

[65] R. Sakai, K. Ohgishi and M. Kasahara, "Cryptosystems Based on Pairings." *Symposium on Cryptography and Information Security (SCIS)*, 2000.

[66] G. Grey, M. Muller, and H. Ruck, "The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems." *IEEE Trans. on Inform. Theory*, no 5, pp. 1717-1718, 1999.

[67] R. Granger, D. Page, and M. Stam, "Hardware and software normal basis arithmetic for pairing-based cryptography in characteristic three", *IEEE Trans. on Computers.* vol. 54, no. 7, pp.852-860. July 2006.

[68] P. Grabher and D. Page, "Hardware acceleration of the Tate pairing in characteristic three", *Cryptographic Hardware and Embedded Systems (CHES)*, LNCS, 3659, pp. 398-411, 2005.

[69] I. Duursma and H. Lee. "Tate pairing implementation for hyperelliptic curves $y^2=x^p-x+d$." *Advances in Cryptology –Asiacrypt*, LNCS, 2894, pp.111-123, 2003.

[70] S. Kwon. "Efficient Tate pairing computation for supersingular elliptic curves over binary fields." *Cryptology ePrint Archive http://eprint.iacr.org/2004/303*, 2004.

[71] V. Miller, "Short programs for functions on curves," *unpublished manuscript, 1986.*

[72] P.S.L.M. Barreto. "A note on efficient computation of cube roots in characteristic three," *Cryptology ePrint Archive http://eprint.iacr.org/2004/ 3035*, 2004.

[73] A. Karatsuba and Y. Ofman. "Multiplication of multidigit numbers on automata." *Sov. Phys. Dokl(English translation)*, vol. 7, no. 7, pp.595-596, 1963.

[74] S. Galbraith, K. Harrison and D. Soldera. "Implementing the Tate pairing." *Algorithm Number Theory Symposium- ANTS V*, vol. 2369 of LNCS, pp. 324-337. Springer-Verlag 2002.

[75] P.S.L.M. Barreto, H.Y. Kim, B. Lynn and M. Scott. "Efficient implementation of pairing based cryptosystems." *Advances in Cryptology CRYPTO' 2002* vol. 2442 of LNCS, pp. 354-368. Springer-Verlag 2002.

[76] T. Kerlins, W. Marnane, E. Popovici, and P. Barreto. "Efficient hardware for the Tate pairing calculation in characteristic 3." *Cryptographic Hradware and Embedded Systems – CHES*, LNCS, 3659:412-426, 2005.

[77] P.S.L.M. Barreto, S. Galbraith, C. Eigeartaigh, and M. Scott. "Efficient pairing computation on supersingular abelian varieties." *Cryptology ePrint Archive http://eprint.iacr.org/2004/375.pdf*, 2004.

[78] G. Bertoni, J. Guajardo, S. Kumar, G. Orlando, C. Paar, and T. Wollinger. "Efficient GF(pm) arithmetic architectures for cryptographic applications." *Topics in Cryptology - CT RSA*, LNCS, 2612:158–175, 2003.

[79] D. Page and N. Smart. "Hardware implementation of finite fields of characteristic 4." *Cryptographic Hardware and Embedded Systems - CHES*, LNCS, 2523:529–539, 2002.

[80] A Menezes, S. Vanstone, and T. Okamoto, "Reducing elliptic curve logarithms to logarithms in a finite field," *Proc. of the 33rd ACM Symposium on Theory of Computing*., pp. 80-89, 1991.

[81] Qingwei Li and Zhongfeng Wang, "Efficient architecture for the tate pairing in characteristic three," submitted to *IEEE Transaction on VLSI*.