8-9-2006

Hardware/Software Co-Design Architecture and Implementations of MIMO Decoders on FPGA

Cao Liang
University of New Orleans

HARDWARE/SOFTWARE CO-DESIGN ARCHITECTURE AND IMPLEMENTATIONS OF MIMO DECODERS ON FPGA

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
The Department of Electrical Engineering

by

Cao Liang

B.S., UESTC, 2002

August 2006

ACKNOWLEDGEMENT

Many people from different parts of my life have contributed to my success in completing this thesis. First and foremost I would like to express my deepest gratitude to my committee members Dr. Xinming Huang, Dr. Jing Ma and Dr. Vesselin Jilkov for the continuous support and shared knowledge. I would like to offer a special thanks to Dr. Jing Ma and Dr. Xinming Huang for providing me the opportunity to work on this project and for their invaluable inspiration and suggestions which make it possible for me to finish this thesis. I am very grateful to Dr. Xinming Huang for serving as my committee chair and guiding me to the right direction during the long graduation process. I would also thank Dr. Vesselin Jilkov for his insightful ideas and comments to the thesis.

Last, I take this opportunity to thank my parents and wife for their unconditional support and love which gives me unending encouragement and strengths. I dedicate my thesis to them.

TABLE OF CONTENTS

List of Figures

List of Tables

Glossary of Abbreviations

ALU – Arithmetic Logic Unit

ASIC – Application Specific Integrated Circuit

AWGN – Additive White Gaussian Noise

BER – Bit Error Rate

BRAM – Block Random Access Memory

CAD – Computer Aided Design

DSP – Digital Signal Processor

Eb/N0 – Bit Noise Ratio

EDK – Embedded Development Kit

FPGA – Field Programmable Gate Array

FSL – Fast Simplex Link

FSM – Finite State Machine

HW/SW – Hardware/Software

IC – Integrated Circuits

I/O – Input Output

IOB – Input Output Bus

ISE – Integrated Software Environment

MB – MicroBlaze

MIMO – Multiple Input Multiple Output

ML – Maximum Likelihood

NRE – non-recurring engineering

OFDM – Orthogonal Frequency Division Multiplexing

OPB – On-chip Peripheral Bus

PAR – Place and Route

PLB – Processor Local Bus

PLD – Programmable Logic Device

QAM – Quadrature Amplitude Modulation

QoS – Quality of Service

RISC – Reduced Instruction Set Computer

RTL – Register Transfer Level

SISO – Single Input Single Output

SoC – System on a Chip

SoS – System on Silicon

SNR – Signal Noise Ratio

VHDL – Very high speed integrated circuits Hardware Description Language

VLSI – Very Large Scale Integration

XUP – Xilinx University Program

Abstract

During the last years, multiple-input multiple-output (MIMO) technology has attracted great attentions in the area of wireless communications. The hardware implementation of MIMO decoders becomes a challenging task as the complexity of the MIMO system increases. This thesis presents hardware/software co-design architecture and implementations of two typical lattice decoding algorithms, including Agrell and Vardy (AV) algorithm and Viterbo and Boutros (VB) algorithm. Three levels of parallelisms are analyzed for an efficient implementation with the preprocessing part on embedded MicroBlaze soft processor and the decoding part on customized hardware. The decoders for a 4 by 4 MIMO system with 16-QAM modulation scheme are prototyped on a Xilinx XC2VP30 FPGA device. The hardware implementations of the AV and VB decoders show that they support up to 81 Mbps and 37 Mbps data rate respectively. The performances in terms of resource utilizations and BER are also compared between these two decoders.

Chapter 1

Introduction

1.1 Motivations and background

Because of the scarce spectrum bandwidth provided for modern wireless systems, the traditional single-input single-output (SISO) channel can not meet the continuously increasing demands for channel capacity and quality of service (QoS) in wireless communications. By introducing multiple antennas into both transmitter and receiver, MIMO technique has emerged as a key technology for the next generation wireless systems.

MIMO communication system can significantly increase the channel throughput and provide better link reliability [1][2][3] at the same bandwidth and same overall transmit power of the SISO communication system. In MIMO system, multiple data streams are transmitted through different antennas at the same time using the same frequency, which is so called spatial multiplexing. Because of this multipath propagation, each output of the receive antenna is a linear combination of all transmitted streams. Then the extremely high throughput is achieved by separating data streams on the receiver using proper decoding algorithms. In addition, QoS is improved because of spatial diversity advantage, since each receive antenna has a measurement of all transmitted data streams. In general, the spectrum efficiency and the propagation range have been greatly increased due to the spatial multiplexing gain and the diversity gain provided by MIMO technique. Combined with orthogonal frequency division multiplexing (OFDM), MIMO technology has been proposed as part of the IEEE 802.11n High-Throughput standard, which is expected to be ratified in mid 2007 [4].

All of these promising performance improvements resulting from MIMO system are achieved at a cost of increased computational complexity especially in the decoders at the receiver side. In a multiple-antenna channel environment, each transmitted signal is aligned on the modulation constellation points. By multiplying all possible signal sets with the MIMO channel matrix, it generates a set of finite points in a multiple-layered lattice structure. The MIMO decoding problem is essentially to search for the closest lattice point to the received point. The optimal detection strategy for a MIMO decoder is to perform a maximum likelihood (ML) search over all possible points inside a lattice structure and find the best one with smallest Euclidian distance to the received signal. But in reality, this method is not practicable because the corresponding computational complexity grows exponentially with the number of transmit antennas M and the number of bits Q used to represent a symbol, since the detector needs to examine all 2^{MQ} possible lattice points for each received vector. For example, in a MIMO system with 4 transmit and 4 receive antennas using 16-QAM modulation scheme, a total of 65536 candidates have to be examined to find the optimal vector. This method is also referred as exhaustive search. An efficient hardware implementation of the decoders has become a key challenging task in MIMO wireless system design.

There are two typical strategies of MIMO detection algorithms used to accelerate the decoding procedure in an arbitrary lattice structure given by its generation matrix. One is the Pohst strategy proposed in 1981 [5], which examines lattice points lying inside a hyper sphere. The decoding algorithm developed by Viterbo and Boutros [6] is based on this strategy and is so called the VB algorithm in this thesis. Another strategy of lattice detection is suggested by

Schnorr and Euchner in 1994 [7], which performs the point search inside the aforementioned hyper sphere with a zig-zag order in each lattice layer with increasing distances from the received vector. A representative lattice decoding algorithm based on this strategy is introduced by Agrell, Eriksson, Vardy and Zeger [8] and is called the AV algorithm. Both algorithms solve the ML detection problem and are commonly referred as sphere decoding algorithms because they search for the closest lattice point in a hyper sphere. Both algorithms are considered as the most promising solutions for the MIMO decoders. A detailed analysis and efficient architecture for both algorithms will be discussed in Chapter 2 and Chapter 4 respectively.

Due to the complexity and high data dependency involved in the decoding algorithms, the MIMO decoders are traditionally implemented on digital signal processors (DSPs), such as Bell Labs layered space-time (BLAST) system [9][10]. Because of not supporting parallelism, a single DSP based implementations can hardly achieve very high decoding rate for MIMO system, especially as the number of antennas increases. Very large scale integrated circuit (VLSI) architecture of MIMO systems has also been studied recently. It is a challenging task to achieve the real-time performance of the sphere decoder due to the complexity of VLSI implementation. Several hardware implementations have been reported by prototyping the VLSI architecture of the VB algorithm, AV algorithm or their extended versions [11][12]. Most recently, Burg, Bormann, and etc have declared a high processing rate for MIMO decoders on an application specific integrated circuit (ASIC) implementation [13]. But the ASIC implementation is generally refined for a fixed number of antennas and a certain signal constellation. The loss of flexibility becomes a major limitation for ASIC implementations of sphere decoders.

Field programmable gate array (FPGA) devises are also widely used in signal processing field due to their flexible reconfiguration and support of parallelism. Combined with their huge processing capabilities, high data rates are ensured for many computational intense algorithms implemented in FPGAs.

The main advantage provided by FPGAs when compared with DSP implementation is the huge performance gains brought by the opportunity to execute the computationally intensive procedures in parallel of an algorithm. Although the design cycle is longer than DSP implementation, once an efficient architecture is developed and the possible parallelisms are explored, FPGA is able to significantly improve the computation throughput. Another advantage of FPGA is the customizability since the processing capacity is scalable based on the FPGA resources and the applications have the flexibility to be upgraded even during the run-time to keep up with changing standards.

Compared with ASIC applications, the implementing of design changes is much easier and the time-to-market cycle is much shorter in FPGA implementations. Furthermore, for system prototyping the overall cost of an FPGA design is much lower than that of an ASIC design.

In addition, the hardware/software (HW/SW) co-design concept has been adopted by FPGA lately by introducing one or more embedded processors into FPGA design, e.g. the PowPC (PPC) or MicroBlaze^{TM} (MB) on Xilinx FPGAs [14][15] and Nios processors on Altera FPGAs [16]. HW/SW co-design technique generally partitions the computational algorithm into customized hardware to achieve high computational speed and into embedded soft processors to reduce the design complexity. Generally, one or more embedded processors can be instantiated in the FPGA

to execute the processing tasks that are less time critical but highly sequential or considerably complicated for direct circuit implementation. This thesis will explore the FPGA-based HW/SW co-design architecture for both AV and VB lattice decoding algorithms.

1.2 Research Objectives

The main objective of this thesis is to develop efficient HW/SW co-design architecture of sphere decoders for both AV and VB algorithms and prototype it on Xilinx University Program (XUP) Virtex-II Pro developing board [17]. The decoding rates are also examined based on the hardware implementations.

1.3 Contributions

The hardware implementation of MIMO decoders with high decoding rate is a challenging and urgent task in multiple-antenna wireless communications. The main contributions of this thesis are summarized as follow:

The FPGA-based HW/SW co-design architecture for MIMO decoders with complex constellation structure is proposed, which partitions the complicated channel matrix preprocessing including matrix inversion and factorization into soft processor and the iterative decoding procedures into customized hardware modules. This architecture is able to significantly improve the decoding rate, and meanwhile keeps it easy to be implemented in hardware.

Three levels of parallelisms are also explored to accelerate the decoding processing: the concurrent execution of the preprocessing on embedded processor and decoding functions on hardware modules, the parallel execution of the real/imaginary decoding parts of complex constellations, and the parallel execution of multiple states during the closest lattice point search.

Both AV and VB decoders are simulated in ModelSim [18], developed in EDK [19] and implemented on XUP Virtex-II Pro developing board with an XC2VP30 FPGA. The hardware results are also verified by the MATLAB software simulations.

System performances including BER, data rate, and resource utilizations are compared between these two decoding algorithms. To the author's knowledge, the real-time performance of the system prototypes are among the fastest MIMO decoders reported thus far.

1.4 Organization of the thesis

The rest of the thesis is organized as follow. Chapter 2 reviews the principles of MIMO system and two sphere lattice decoding algorithms. The step by step decoding procedure and state based flow charts are given and analyzed for both AV and VB algorithms. Chapter 3 gives the introduction to FPGA structure and hardware/software co-design procedures. Three levels of parallel structures for the complex sphere decoders are explored in Chapter 4. The complex constellation parallelism is derived theoretically and verified by MATLAB simulations. The data flow dependency among the iterative searching procedure is also analyzed to achieve the state level parallelism in this chapter. Comparisons of the experimental results between these two algorithms are presented in Chapter 5, followed by the conclusion Chapter 6.

Chapter 2

MIMO system and sphere decoding algorithms

This chapter gives a brief introduction to MIMO system. Two typical sphere decoding algorithms are described in details and are further analyzed for hardware architecture design.

2.1 Introduction to MIMO system

During recent years, MIMO technology has emerged as a promising solution for the ever increasing demands of higher data rate and better QoS in wireless communications. By applying multiple antennas at both transmitter and receiver sides, MIMO system is able to greatly enhance the spectral efficiency and also provides a better range of coverage than the traditional SISO system. Recent initiatives for standardization of future MIMO systems including UMTS (3GGP Release 7), IEEE 802.11n wireless LAN, and IEEE 802.16 WiMax reflect the importance of MIMO technique.

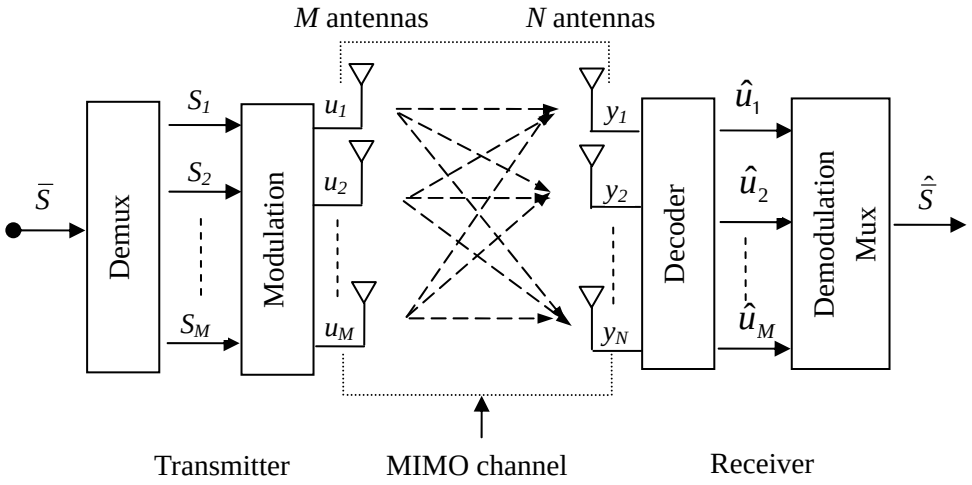

Figure 2.1 Block diagram of MIMO system

Figure 2.1 gives a schematic representation of a MIMO system. The original signal $\overline{S}$ is split into M lower rate data streams, modulated and transmitted simultaneously from each transmit antenna. The receiver, having complete knowledge of the channel, decodes these individual data streams, demodulates them and combines them together so as to recover the original signal. In the wireless MIMO channel, each data stream is transmitted through different data paths to reach different receive antennas. Two performance gains are provided by this multipath propagation method.

2.1.1 Antenna diversity gain

Fading is generally a cumbersome problem in wireless communications. When the signal power drops significantly, the channel is said to be in a fade, which gives rise to BER and reduces the signal coverage range. In MIMO system, replicas of the same transmitted signal are provided across different antennas at the receiver side through independent fading paths. When one path is in fade, it is unlikely that all other paths are also in deep fade. Hence the more reliable reception is achieved in MIMO system. This QoS gain brought by MIMO system is called antenna diversity gain or spatial diversity gain.

2.1.2 Spatial multiplexing gain

The spatial multiplexing is to transmit the split data streams concurrently through different transmit antennas, which helps to increase the data transmission rate for the same bandwidth and with no additional power expenditure. This performance gain is only possible in MIMO system. The capacity of the wireless channel grows linearly with the number of transmit antennas. Thus the spatial multiplexing can enormously enhance the spectral efficiency, which makes MIMO

technique uniquely important in modern wireless communications with scarce spectrum bandwidth.

2.2 Lattice decoding problem

Design an efficient decoder for MIMO receiver to match with the high transmission rate has become a challenging task, because the high computational complexity is involved especially in high dimensional MIMO systems with complex signal constellation.

Considering a MIMO system with *M* transmit *N* receive antennas, the received vector $\bar{y}$ is given by:

$$\bar{y} = \bar{u}H + \bar{n} \qquad \text{Eq 2.1}$$

where $\bar{u}$ is the $1 \times M$ transmitted signal vector, and $\bar{n}$ is a $1 \times N$ additive white Gaussian noise (AWGN) vector with zero mean and N_0 variance. *H* is the $M \times N$ channel transfer matrix which is assumed to be known by the receiver. Each element in the *H* matrix corresponds to one fading coefficient between one transmit-receive antenna pair. For selected modulation scheme, each element in $\bar{u}$ is represented by one of the constellation points. A lattice structure, denoted as Λ, is generated by multiplying all possible $\bar{u}$ with *H*. The ML lattice decoding problem is to find the closest lattice point to the received point in the lattice structure:

$$\hat{\bar{u}} = \arg \min_{\bar{u} \in \Lambda} \| \bar{y} - \bar{u}H \|^2 \qquad \text{Eq 2.2}$$

where $\hat{\bar{u}}$ is the decoded vector. Thus the ML-based decoding system can be summarized as:

Input: The channel lattice generation matrix *H* and the received vector $\bar{y}$.

Output: A $1 \times M$ vector $\hat{\bar{u}}$ such that $\hat{\bar{u}}H$ is a closest lattice point to $\bar{y}$.

2.3 Sphere lattice decoding algorithms

The optimal lattice decoding method is to examine all possible lattice points in the lattice structure and find the best one with minimum Euclidian distance to the received vector. However, this method is not a solution for practical MIMO decoders because of the extreme complexity involved in this exhaustive search.

Two sphere decoding algorithms are analyzed in this section, named AV and VB algorithms. Both algorithms are ML-based lattice decoding algorithms, which try to enumerate the lattice points inside a sphere, and find the closest one to the received vector. The main difference between these two algorithms is the investigating order inside the lattice structure. The VB algorithm searches from the lower bound to the upper bound in each search layer and examines all possible lattice points falling into a certain sphere in the lattice structure with an initial radius $\sqrt{C}$. Meanwhile the AV algorithm spread out from a nearby lattice point to the received point and terminates once the total distance is greater than the best distance and the search procedure reaches the bottom layer. No initial radius is needed in the AV algorithm, and it does not need to upgrade the lower and upper bounds of each layer using the time consuming square root functions as it needs in the VB algorithm. It is claimed that the AV algorithm is about 2 to 8 times more efficient than the VB algorithm [8] even if a proper initial radius $\sqrt{C}$ is applied to the VB algorithm.

The step by step procedure for each algorithm is presented below. To improve the efficiency of the VB algorithm, a modified version of VB algorithm is adopted in this thesis, which avoids the hardware inefficient square root functions.

2.3.1 VB decoding algorithm

Two lattices are identical if all the lattice points generated by the lattice matrices and given signal set are the same. So basis reduction can be performed on the lattice generation matrix *H* to reduce the complexity of the decoding procedure. In the VB algorithm, Cholesky factorization is applied to the gram matrix $G = HH^T$, and it yields $G = R^T R$, where *R* is an upper triangular matrix. Thus the squared Euclidian distance shown in Eq 2.2 can be rewritten as:

$$\begin{aligned} d^2 &= \| \bar{y} - \bar{u}H \|^2 \\ &= \| (\bar{\rho} - \bar{u})H \|^2 \\ &= (\bar{\rho} - \bar{u})HH^T(\bar{\rho} - \bar{u})^T \\ &= (\bar{\rho} - \bar{u})R^T R(\bar{\rho} - \bar{u})^T \\ &= \| (\bar{\rho} - \bar{u})R^T \|^2 \\ &= \| R(\bar{\rho} - \bar{u})^T \|^2 \end{aligned} \qquad \text{Eq 2.3}$$

where $\bar{\rho} = \bar{y}H^{-1}$. By properly chosen the $\sqrt{C}$, the searching range becomes a sphere with square radius *C* centered at the received point. And thanks for the attributes of the upper triangular matrix, the squared distance can be constrained by the following inequation:

$$d^2(\bar{\rho}, \bar{u}) = \sum_{i=1}^{M} (r_{ii}(\rho_i - u_i) + \sum_{j=i+1}^{M} (r_{ij}(\rho_j - u_j)))^2 \le C \qquad \text{Eq 2.4}$$

Substituting $\xi_i = \rho_i - u_i$, $q_{ii} = r_{ii}^2$ for $i = 1,\ldots,M$ and $q_{ij} = r_{ij} / r_{ii}$ for $i = 1,\ldots,M$, $j = i+1,\ldots,M$ into Eq 2.4:

$$d^2(\bar{\rho}, \bar{u}) = \sum_{i=1}^{M} q_{ii}(\xi_i + \sum_{j=i+1}^{M} (q_{ij}\xi_j))^2 \le C \qquad \text{Eq 2.5}$$

Starting from the bottom row of matrix *R* and working backwards, the upper and lower bounds of the examining lattice point can be determined by the partial distance derived from Eq 2.5:

$$q_{ii}(\xi_i + \sum_{j=i+1}^{M}(q_{ij}\xi_j))^2 \le C \qquad \text{Eq 2.6}$$

where u_i is used to represent the examining index of layer i and L_i denotes the upper bound of u_i. So the initial u_i, which is essentially the lower bound of the examining index, and L_i for each layer is determined by:

$$\begin{aligned} L_i &= \left\lfloor \sqrt{T_i / q_{ii}} + S_i \right\rfloor \\ u_i &= \left\lceil -\sqrt{T_i / q_{ii}} + S_i \right\rceil - 1 \end{aligned} \qquad \text{Eq 2.7}$$

where $\lfloor x \rfloor$ is the smallest integer greater than x, and $\lceil x \rceil$ is the greatest integer smaller than x. And:

$$S_i = \rho_i + \sum_{j=i+1}^{M} q_{ij}\xi_j \qquad \text{Eq 2.8}$$

$$T_{i-1} = T_i - q_{ii}(S_i - u_i)^2 \qquad \text{Eq 2.9}$$

The VB-based decoder searches from the bottom layer to the top layer and scans each lattice index from the lower bound to the upper bound. When the algorithm reaches the top layer without violating the bound constraint, a valid lattice point is found. Then the new distance d_{new} between the valid lattice point and the received point is calculated and compared with the currently best distance d_{best}, which is initialized to be equal to $\sqrt{C}$. If d_{new} is smaller than d_{best}, a closer lattice point is found and stored as the currently best lattice point. The searching radius is upgraded to d_{new}. This process iterates until all the lattice points within the sphere are examined. The flow chart and more details of this VB algorithm are available in [6]. In this thesis, the VB algorithm is partitioned into different states as show in the following step by step demonstration.

Step 1: Preprocessing and initialization:
Transform H into an upper triangular matrix by Cholesky [20] factorization algorithm. Calculate $\bar{\rho} = \bar{y}H^{-1}$. Initialize the sphere radius $\sqrt{C}$ by an adaptive method [21]. Set dimension index $i=M$ and $d_{best}=\sqrt{C}$. Find the upper bound L_M and index u_M.
Step 2: Finite State Machine (FSM)
Upgrade $u_i= u_i +1$.
If $u_i < L_i$ and $i>1$ go to State A;
If $u_i < L_i$ and $i=1$ go to State B;
If $u_i > L_i$ go to State C.
Step 3: State A:
Expand the layer into (i-1)-dimensional sublayer and find the S_i and T_i used to upgrade u_i and L_i. Goes to State D.
Step 4: State B
Upgrade d_{new}.
If $d_{new} < d_{best}$, record the currently best distance and the best point. Set $i=M$. Go to State D.
If $d_{new} > d_{best}$, go to step2.
Step 5: State C
Stop if $i=M$, otherwise move the procedure one step up $i=i+1$, and go to step 2;
Step 6: State D
Upgrade u_i and L_i, and go to step 2.

Figure 2.2 State representation of VB algorithm

As shown in Figure 2.2, basis reduction is performed to reduce the complexity of decoding procedure before the closest point search begins. This is called preprocessing, which involves Cholesky decomposition and matrix inversion. These complex matrix manipulations are difficult and too costly for hardware implementation, and they are not executed frequently in the sphere decoding algorithm. So only the iterative decoding procedures are considered to be implemented on FPGA hardware.

By analyzing the calculations involved in these four states of the searching procedure, it is

clear that State D carries much more computational load than other states. This is because the hardware inefficient square root functions are used to upgrade u_i and L_i. The requirements of the square root functions make the computation load unbalanced among the four states. Furthermore, State D counts for about 40% of the total number of states visited for a vector to be decoded in MATLAB simulation. Thus the State D becomes the dominant computational part in the VB algorithm, which can not take full parallelism advantage of FPGAs. To design an efficient FPGA architecture, the VB algorithm must be modified to avoid the square root calculations. One of the modified versions of the VB algorithm is presented in [22], which suitably matches the FPGA-based design.

The purpose of the square root computations in the original VB algorithm is to find the lower and upper bounds for the examining index in each layer. Because the points in a lattice structure are generated from the transmitted signal vectors, the examining index u_i should also come from the signal constellation points. Besides, in coherent demodulations the modulation scheme is known by the receiver. Thus a new method to determine the searching range for the examining index can be achieved by directly substituting each symbol of the signal constellation into Eq 2.6. Redefine T_i as:

$$T_i = q_{ii}(S_i - u_i)^2 \qquad \text{Eq 2.10}$$

where S_i remains the same as in Eq 2.8. The partial distance at the i^{th} layer can be rewritten as:

$$P_k = q_{ii}(S_i - x_k)^2 + \sum_{j=i+1}^{M} T_j \le C, k = 1,..., K \qquad \text{Eq 2.11}$$

where x_k is a symbol in the signal constellation and K is the total number of symbols. Thus the L_i

and u_i can be upgraded as:

$$L_i = \max(x_k), and \quad P_k \le C$$
$$u_i = x_t - 1, where \quad P_t = \min(P_k) \le C, \quad k = 1,...,K \qquad \text{Eq 2.12}$$

The flow chart of the modified VB algorithm is given below:

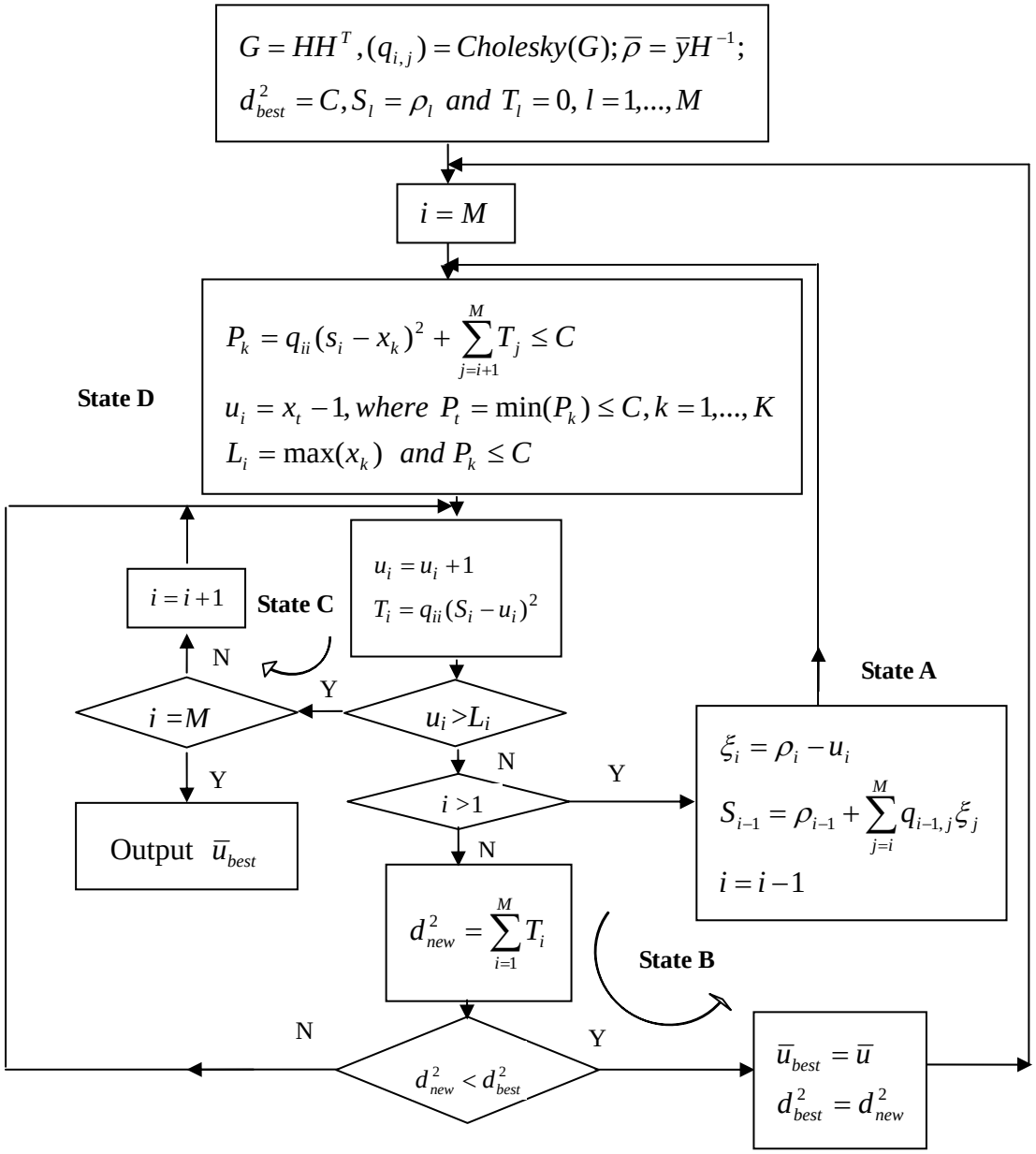

Figure 2.3 Flowchart of modified VB algorithm

The changes of the modified VB algorithm are subtle but significant. Instead of calculating

the examining range by the constraint radius $\sqrt{C}$, the symbols from the signal constellation are directly used to calculate all possible partial distances in the i^{th} layer. And the examining index u_i is restricted to the qualified symbols. There are at least three advantages brought by this method: Firstly, it avoids the square root functions, and the partial distances P_k can be calculated in parallel to achieve an efficient hardware implementation. Secondly, within the signal constellation, the number of the examining points is more likely to be reduced compared with the original method, which makes the received vector to be decoded faster than the original VB algorithm. At last, it guarantees that all the examining points are coming from the signal constellation points which leads to a better BER performance.

Although an efficient searching procedure can be conducted by the improved VB algorithm, the choice of the initial sphere radius $\sqrt{C}$ is very crucial to the decoding speed. If the radius is too small, the search will fail and no lattice point within the sphere will be found. However large $\sqrt{C}$ means more points will be examined during the searching procedure and longer time will be spent for one vector to be decoded. Thus the decoding rate will be greatly degraded if the radius is too large. An adaptive method to calculate the radius $\sqrt{C}$ is discussed in Chapter 5.

2.3.2 AV decoding algorithm

Instead of examining the lattice points from the lower bound to the upper bound in each layer, the AV algorithm starts from the Babai point [23], and spreads out within the distance between the Babai point and the received point.

The lattice generation matrix H is decomposed into an $M \times M$ lower triangular matrix R and an $M \times N$ orthonormal matrix Q using KZ reduction or LLL reduction algorithms [24],

where $H = R^{-1}Q$. Then Eq 2.1 can be rewritten as:

$$\begin{aligned} \bar{y} &= \bar{u}R^{-1}Q + \bar{n} \\ \bar{y}Q^T &= \bar{u}R^{-1} + \bar{n}Q^T \end{aligned} \qquad \text{Eq 2.13}$$

The AV decoding problem is formulated as:

$$\hat{\bar{u}} = \arg\min_{\bar{u}\in\Lambda} \| \bar{y}Q^T - \bar{u}R^{-1} \|^2 \qquad \text{Eq 2.14}$$

The index u_i is calculated and examined in the order shown in Eq 2.15 and Eq 2.16:

$$\bar{e}_i = \bar{y}Q^T R \qquad \text{Eq 2.15}$$

$$u_i = \{[e_{ii}], [e_{ii}] \pm 1, [e_{ii}] \pm 2, \ldots\} \qquad \text{Eq 2.16}$$

where $[x]$ finds the closest integer to x. The orthogonal distance in the i^{th} layer is given by:

$$d = (e_{ii} - u_i) / r_{ii} \qquad \text{Eq 2.17}$$

where r_{ii} is the i^{th} diagonal element in R. The partial distance d_{new} in the i^{th} layer is upgraded by Eq 2.18 if it is smaller than the currently best distance d_{best}:

$$d_{new}^2 = \sum_{j=i}^{M} ((e_{jj} - u_j) / r_{jj})^2 = \sum_{j=i}^{M} d^2 \qquad \text{Eq 2.18}$$

And the closest point search expands to the (i-1) dimensional sublayer by:

$$e_{i-1,j} = e_{ij} - dr_{ij}\ ,\ j = 1, \ldots, i-1 \qquad \text{Eq 2.19}$$

On the other hand, if the partial distance is greater than d_{best}, the search procedure steps 1 layer back, and the examining index in this layer is upgraded in the order shown in Eq 2.16. This zig-zag order leads to a nondecreasing distance to the received vector $\bar{y}$ in each layer. The searching procedure terminates when it moves down to the bottom layer without finding a closer lattice point than the currently saved best one. More details are available in [8]. The iterative decoding procedures are divided into three states as shown in Figure 2.4.

Step 1: Preprocessing and initialization:
Transform H into a lower triangular matrix by QR [25] decomposition algorithm. Initialize dimension index $i=M$. Find the bounded index u_M and the orthogonal distance d from $\bar{y}$ to the layer with index u_M. Set $d_{best}=\infty$.
Step 2: Finite State Machine (FSM)
Upgrade d_{new} using d.
If $d_{new} < d_{best}$ and $i>1$ go to State A;
If $d_{new} < d_{best}$ and $i=1$ go to State B;
If $d_{new} > d_{best}$ go to State C.
Step 3: State A:
Expand the layer into (i-1)-dimensional sublayer and find d and bounded u_i. Go to step 2.
Step 4: State B
Record the currently best distance and the lattice point. Set i=2, and find d and bounded u_i. Go to step 2.
Step 5: State C
Stop if $i=M$, otherwise move the procedure one step up $i=i+1$, and find d and bounded u_i. Go to step 2;

Figure. 2.4 State representation of AV algorithm

Similar to the VB algorithm, the matrix preprocessing involves QR decomposition and matrix inversion, which is only needed to be upgraded once in a signal frame length (typically 10 ms). Thus only the iterative decoding functions are considered to be implemented on FPGA. By analyzing the three states involved in the AV algorithm, it is clear that the computational load is well balanced among these states, and no complicated calculations are introduced into the decoding procedure. Furthermore, based on the data flow dependency among these states, the state level parallelism can be explored for the AV algorithm to design an efficient FPGA

architecture.

The flow chart of the AV algorithm is also given in Figure 2.5:

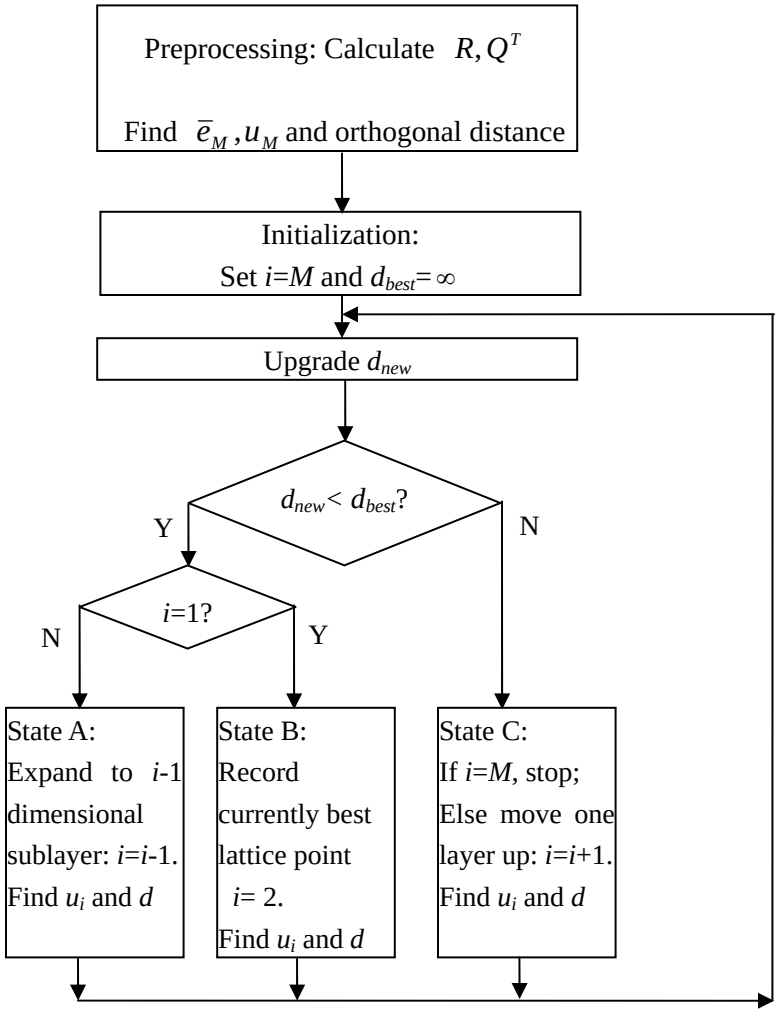

Figure 2.5 Flowchart of AV algorithm

Compared with the VB algorithm, the advantages of using the investigating order in Eq 2.16 are obvious. Firstly, it does not need to calculate and upgrade the upper and lower bounds in each layer which involves intensive computations in the VB algorithm. Secondly, by using this nondecreasing distance order to search inside each lattice layer, the chance of finding the correct layer early is maximized. Furthermore, by spreading the searching procedure from the Babai

point, no initial radius $\sqrt{C}$ is needed in the AV algorithm. Literature results show that the AV-based decoder is about 2 to 8 times faster than the VB-based decoder [8]. Efficient hardware/software co-design architecture is designed for both algorithms in Chapter 4, where the experimental results from our designs also agree with this claim accordingly.

Chapter 3

Hardware/software co-design and developing board

3.1 Introduction to hardware/software co-design

The term of hardware/software co-design surfaced in the early 90s when rapid reduction in the size of integrated circuits (ICs) made it possible to have embedded processor(s) on the same hardware silicon or chip, which is so called System on Silicon (SoS) or System on a Chip (SoC). It has moved from an emerging discipline to a mainstream technology [26], and is applied in a vast number of areas. The principle of HW/SW co-design technique is to partition the applications into the embedded processors and the customized hardware modules. The goal of this architecture is to enhance the system performance while reducing the design effort and costs, which is achieved by the benefits from HW/SW co-designs. The advantages of using processors lie in the following aspects: First, software is more flexible and cheaper than hardware, which allows late design changes and simplified debugging opportunities [35]. Furthermore, the available software libraries make many computationally complicated functions easy to be implemented inside the microprocessors. Finally, the possibility of reusing software by porting it to other processors, reduces the time-to-market cycle and the design effort. On the other hand, the hardware is always used to implement the computationally intensive tasks, which can extraordinarily improve system performance by the parallel executions. With a flexible and high speed interface between the hardware and software cores, this architecture can greatly speed up the applications by the high speed hardware implementation and can also maintain the flexibility and programmability of the microprocessor.

ASICs and FPGAs are both widely used in the HW/SW co-design systems. ASICs provide a dedicated hardware solution, which can usually lead to the best hardware performance at the expense of long design cycle and high non-recurring engineering (NRE) charges. On the other hand, FPGAs provide more flexibility and shorter design time than ASICs at a higher cost per unit device. It is a perfect platform for system prototyping and low-volume products. During past a few years, traditional FPGAs have combined with embedded microprocessors and related peripherals to form complete SoCs. Examples of this hybrid technology can be found in Xilinx Virtex-II Pro, Virtex-4 devices and Altera Stratix FPGAs. In this thesis, Xilinx Virtex-II Pro developing board is chosen to be the test bench for the MIMO decoders. The Xilinx FPGA-based HW/SW co-design architecture is given below:

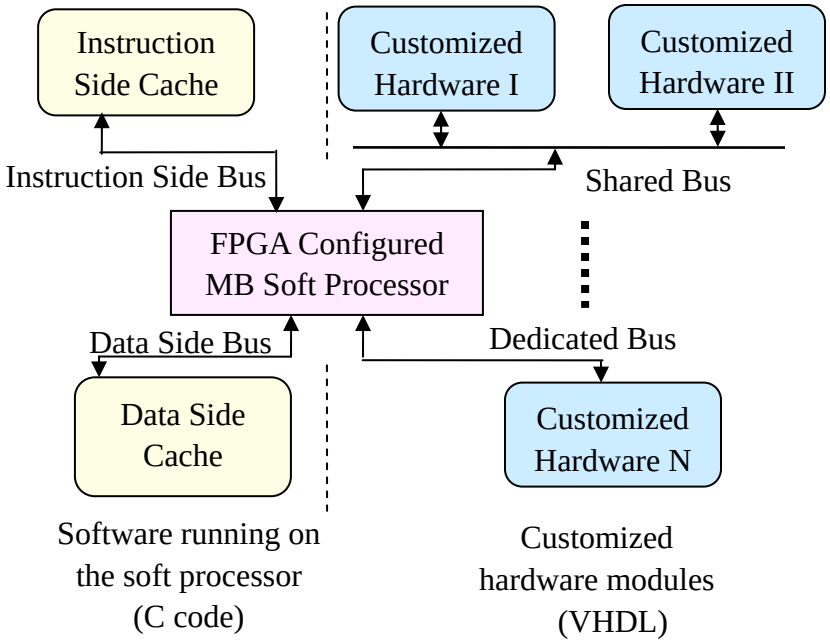

Figure 3.1 Xilinx FPGA-based hardware/software co-design architecture

The Xilinx MicroBlaze soft core, which is licensed as part of Xilinx EDK, is a 3-stage pipeline 32-bit RISC processor with 32 general purpose registers, ALU and a rich instruction set optimized for embedded applications. It is implemented by general logic primitives within the FPGA logics [15]. The MicroBlaze solution is designed to be flexible, giving the user control of a number of features such as the cache size, interfaces, and execution units. The configurability allows the user to trade-off features for size, in order to achieve the necessary performance for the target application at the lowest possible cost. Multiple customized hardware modules can be attached to the MB for hardware functions through interfaces such as on-chip peripheral bus (OPB), fast simplex link (FSL) or processor local bus (PLB). The software functions are executed inside the MB soft core with the instructions and data provided by instruction side cache and data side cache. C code and VHDL code are used to describe the software and hardware designs respectively. By properly partitioning of an application, the HW/SW co-design architecture is able to greatly enhance the computational performance of the whole system meanwhile keeps the flexibility of complicated manipulations inside the MB executions.

3.2 FPGA structure

There are three typical types of ICs, programmable logic devices (PLDs), ASICs and FPGAs. With predetermined architecture by manufacturer, PLDs can provide great programmable flexibility for engineers to perform a variety of different functions. But these devices contain a relatively limited number of logic gates which is only suitable for some simple and small applications. On the other hand, ASICs contain millions of logic gates, which can be use to implement extremely large and complex systems and to offer optimized hardware

implementations. But the designing of ASIC systems is too time consuming and costly. And once the final design is finished, it can not be modified to adapt to any application changes.

FPGAs stand in the middle of PLDs and ASICs, which contain programmable logic blocks along with configurable interconnections between them [27]. Their functionality can be reconfigured in the way like PLDs, and they can contain millions of logic gates to provide a performance lean more toward to that of ASICs. The cost of an FPGA design is much lower than that of an ASIC. Meanwhile the design changes, even at the run-time, are implemented much easier in FPGAs. Thus, FPGA is considered as an ideal platform to perform the computationally intensive operations involved in the sphere decoding algorithms for reasons of performance, cost and reconfigurability. Xilinx and Altera are the two major manufacturers in FPGA market. Each company has a variety of FPGA series, which suit for different design requirements. Figure 3.2 shows the structure of Xilinx FPGAs. The Xilinx FPGA device is organized as an array of logic elements, named slices, and programmable routing resources used to provide connections between the slices, FPGA I/O pins, on-chip block memory and other resources [28]. Each slice contains two logic cells, which can be performed as a 16-bit register or conducts the functions of basic logic gates such as AND, OR, XOR, and etc. By properly combining these basic logic elements, certain circuits can be generated to perform more complex combinatorial functions. Multipliers are inherently slow if implemented by connecting a large number of programmable logic blocks together. Xilinx FPGAs incorporate special hardwired embedded multipliers to improve the design performance.

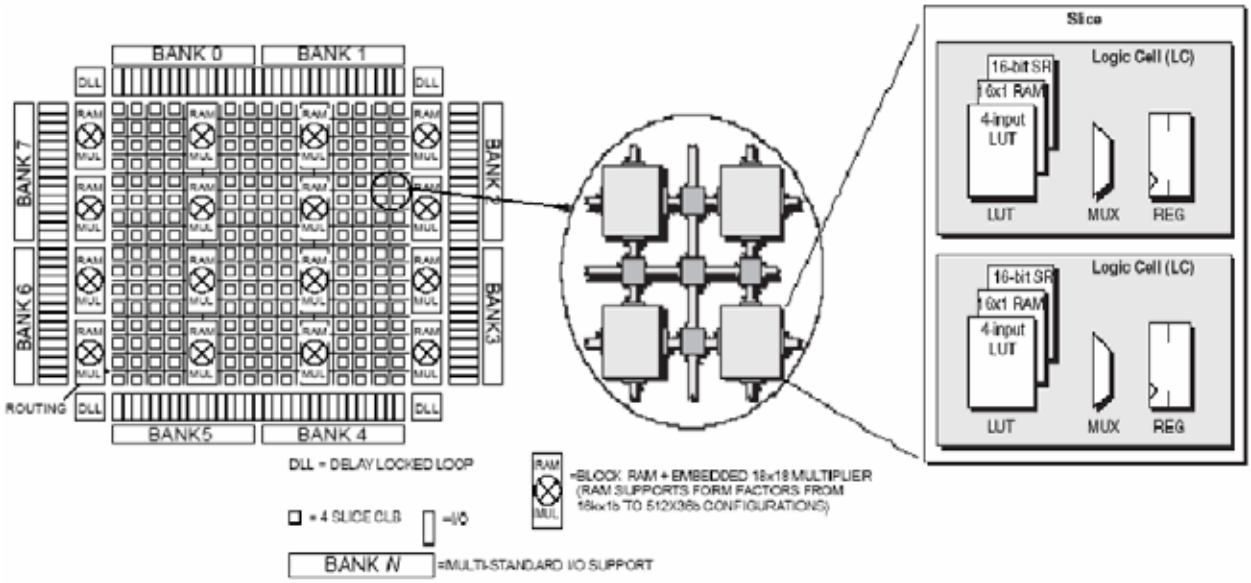

Figure 3.2 Xilinx FPGA structure [28]

The Xilinx Virtex-II Pro XC2VP30 FPGA is chosen as the test bench device for this thesis research. The available resources of particular interest to our design are listed in Table 3.1

FPGA	XC2VP30
Number of slices	13969
Number of External IOBs	896
Number of embedded 18×18 MULT	136
Size of BRAMs	2448 Kb

Table 3.1 Xilinx XC2VP30 FPGA features

3.3 Design flow and developing board

As the HW/SW co-design architecture became more and more sophisticated, a lot of commercialized computer aided design (CAD) tools have emerged into this field to facilitate the co-design procedure. Figure 3.3 shows the HW/SW co-design flow involved in the

implementation and evaluation of the MIMO lattice decoders.

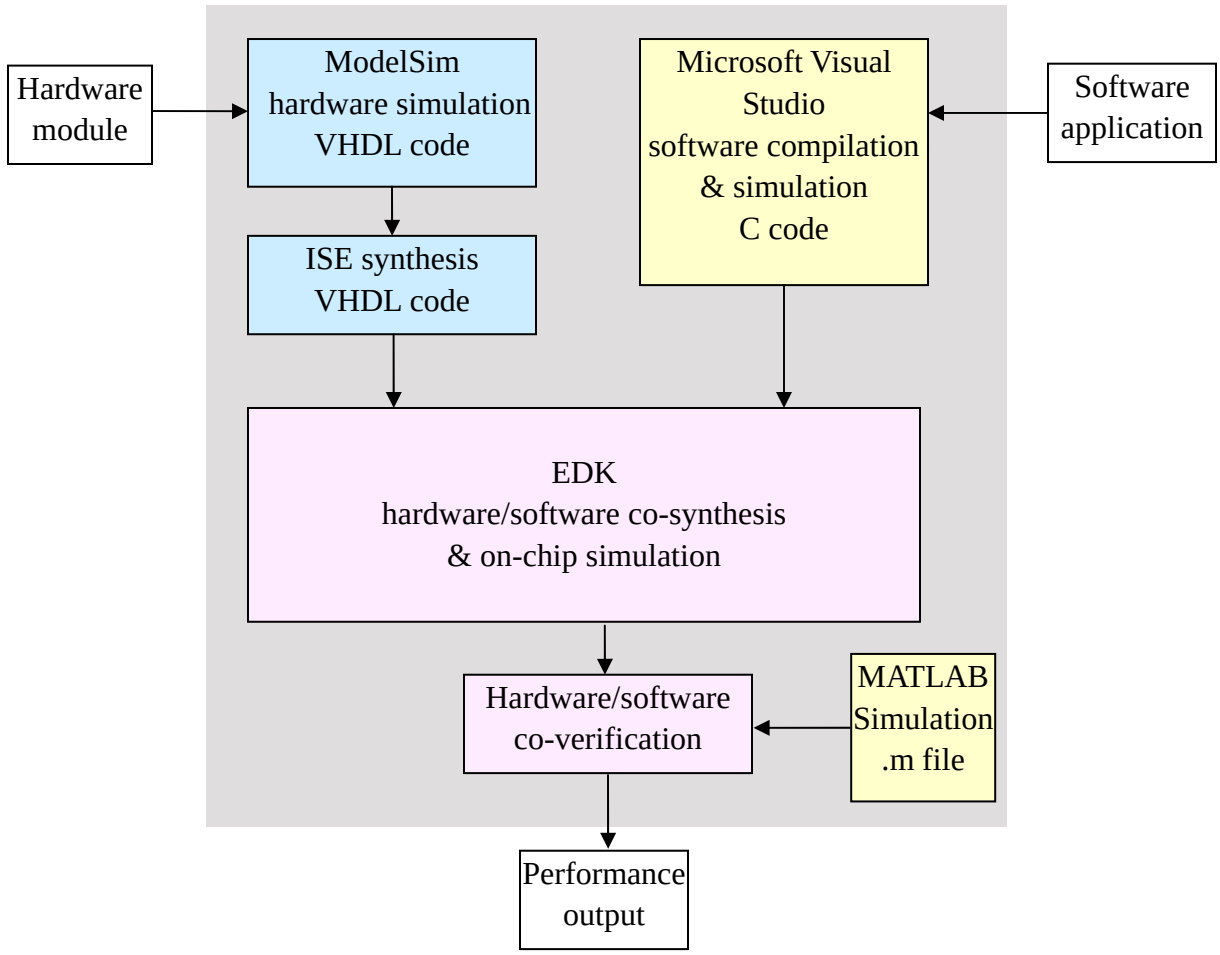

Figure 3.3 Design flow for FPGA-based HW/SW co-design systems

In HW/SW co-designs, the application is partitioned into soft core and hardware modules. The software applications are developed and simulated in C language by Microsoft Visual Studio [29]. The hardware modules are first described in VHDL language, and simulated in Mentor Graphics ModelSim [30] at RTL level. Then they are synthesized in Xilinx ISE [31] software environment and translated into electronic hardware circuits to be mapped on chosen FPGA. Xilinx EDK [32] tool is used to merge the hardware and software designs together, and it is a comprehensive suit of integrated development environment to ease and facilitate the HW/SW

co-design process. As shown in Figure 3.4, the system window is used to design the SoC architecture, where versatile hardware modules can be attached to the embedded processor with certain address allocation and I/O assignments. After that the software application is developed in the Application window along with the interface between the software and hardware modules. The on-chip simulations for the whole system can also be conducted within the EDK tool.

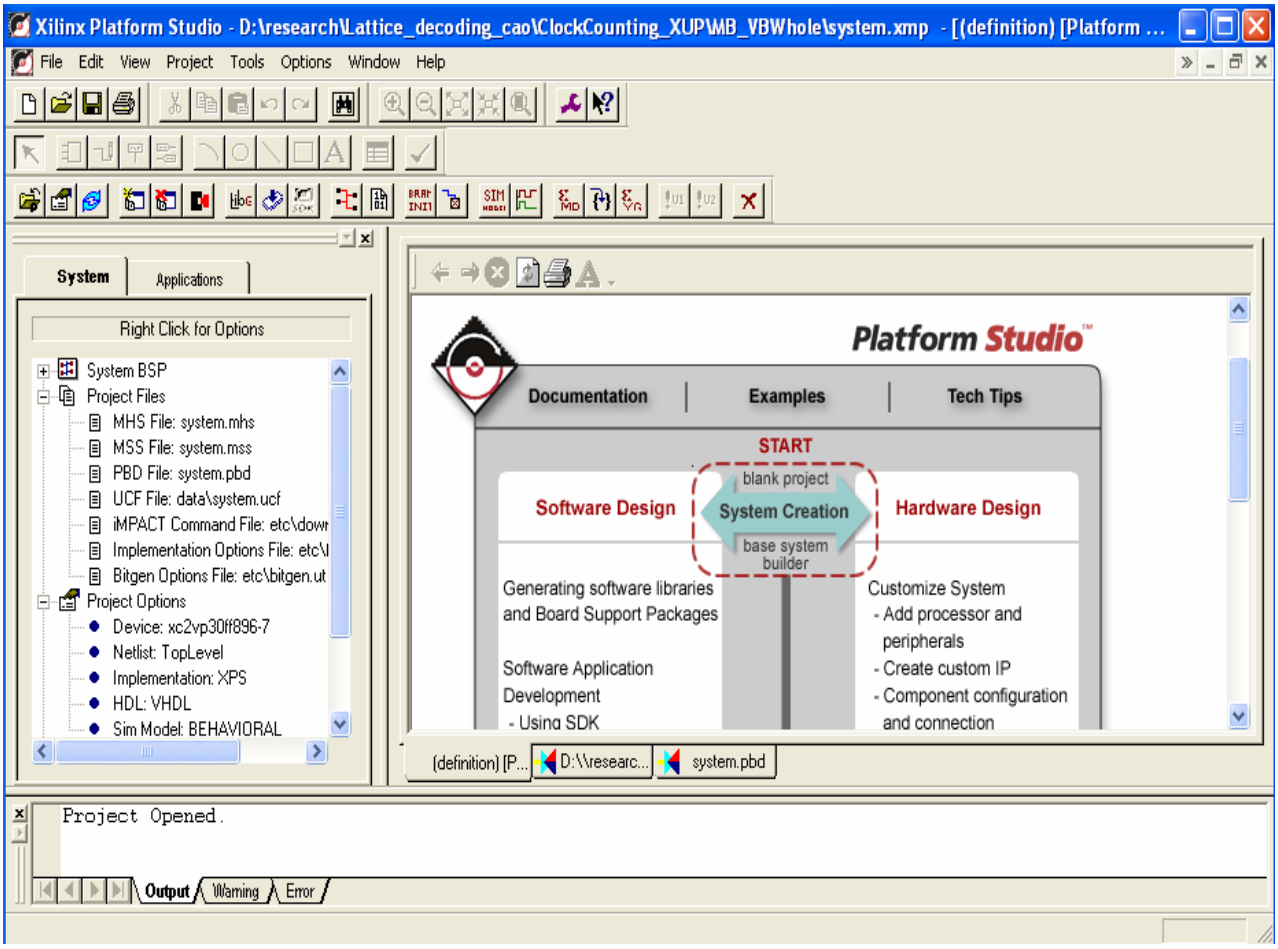

Figure 3.4 Xilinx EDK workplace

At last, the results from the HW/SW co-design are verified with the MATLAB software simulations. In the MIMO lattice decoder designs, the performances of decoding rate, FPGA resource utilization and BER are evaluated based on these simulations and verifications.

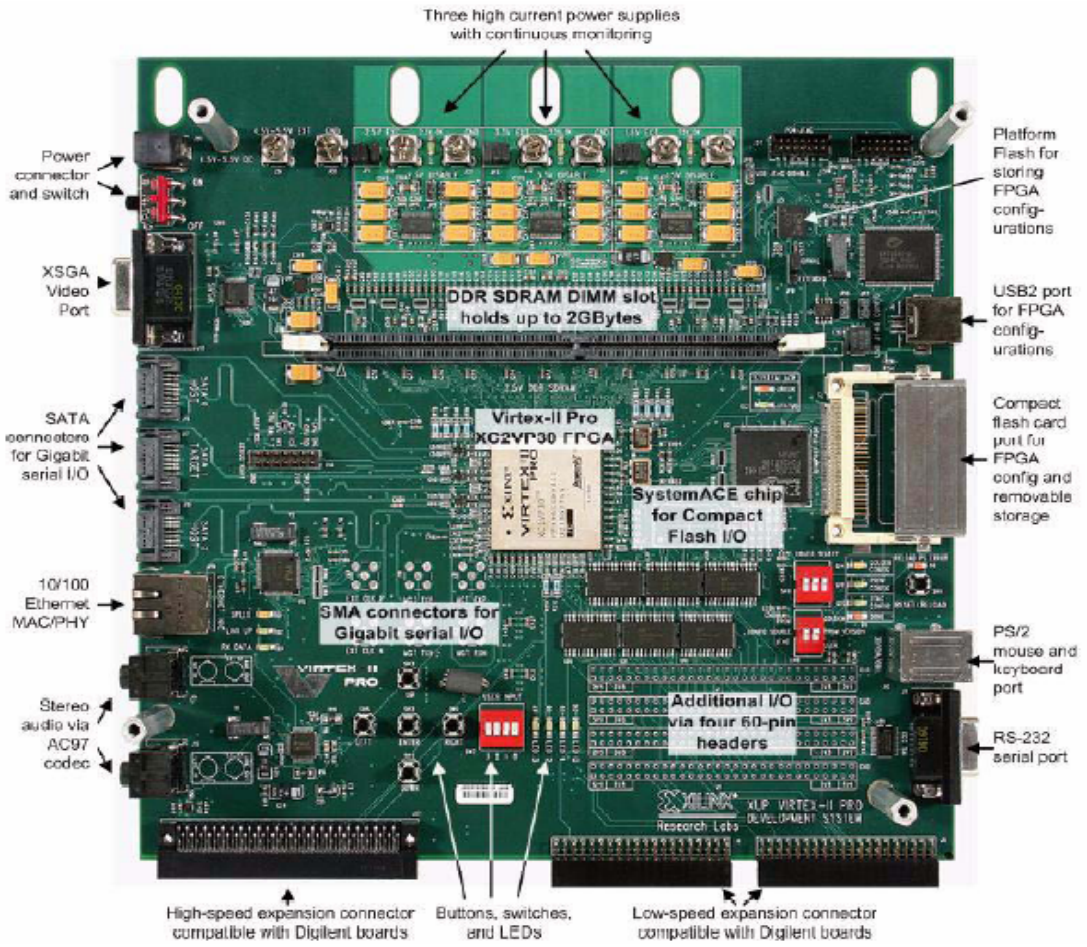

Figure 3.4 XUP Virtex-II Pro developing board [17]

Xilinx XUP Virtex-II Pro developing board is chosen as the target board for the implementations of lattice decoders. It consists of a high performance XC2VP30 FPGA surrounded by a comprehensive collection of peripheral components that can be used to create a complex system. The FPGA device can be configured by several methods including external parallel port interface, USB cable or internal CompactFlash storage media [33]. One or more MB soft cores can be implemented by configuring a group of logic cells inside the FPGA chip. Efficient architecture of aforementioned two sphere lattice decoders is explored in the next chapter for the FPGA-based HW/SW co-design.

Chapter 4

Hardware/software architecture for MIMO Sphere decoders

This chapter presents HW/SW co-design architecture for two MIMO sphere decoding algorithms. A coarse-gain partition of both algorithms is discussed first. Then three levels of parallel structures are explored to generate an efficient HW/SW architecture to improve the system performance. The co-design architecture is prototyped on the Xilinx Virtex-II Pro developing board.

4.1 Partitioning of the sphere decoding algorithms

In our AV and VB sphere decoding algorithms, the basis reduction is generally performed to reduce the complexity of the decoding procedure, which transforms the lattice generation matrix into a triangular matrix. This process is called preprocessing that involves matrix factorization operations such as QR or Cholesky decompositions and matrix inversions. These complex matrix manipulations are difficult and not efficient to be mapped directly onto FPGAs. Fortunately, in MIMO wireless communications, the preprocessing stage of the sphere decoders does not need to be updated frequently. Generally, the channel matrix H can be assumed to be static during the transmission of one frame length of data signals. Therefore the preprocessing of lattice generation matrix only needs to be executed once for a certain time period (typically 10 ms). The MB soft processor on Virtex-II Pro FPGA running at about 100 MHz is able to complete the preprocessing task timely. For this reason, the basis reduction operations for both AV and VB algorithms are partitioned into the Xilinx MB soft processor.

Unlike the preprocessing stage of the decoding algorithms, the actual closest lattice point

searching procedure needs to be executed for every received signal vector. This iterative procedure is very time consuming if implemented sequentially in soft processor. Furthermore the decoding rate of the sphere decoder is determined by the average completion time of the searching procedure. Thus it is desirable to put the sphere searching procedure directly into FPGA hardware circuits. The gate level implementations on FPGA can greatly accelerate the computationally intensive functions involved in the decoding procedure. Because of supporting parallelism, some of the decoding calculations can be executed concurrently if no data flow dependency lies among them. Once an efficient architecture is explored, the FPGA-based implementation can extraordinarily speed up the decoding procedure of the sphere decoders. In addition to the performance gain, FPGA device also provides the flexibility to reconfigure the system to accommodate any changes of the MIMO systems including number of antennas, modulation schemes, and etc.

Based on the analysis above, a straightforward coarse-gain partition is applied to the AV and VB decoding algorithms to build up the HW/SW co-design architecture, which includes a soft processor, customized hardware module(s), and a shared peripheral bus between them for data communication. As shown is Figure 3.1, the channel matrix preprocessing is implemented in the embedded MB soft core, and the decoding procedure is mapped onto FPGA hardware modules. The preprocessing results (the upper or lower triangular matrix) are transferred from the MB to the hardware modules periodically through the OPB interface, which is synchronous to MB processor and can achieve a high data transfer rate. Subsequently, three levels of parallel structures are explored to further accelerate the decoding speed of this co-design architecture

while maintaining the same BER performance.

4.2 Processor level parallelism

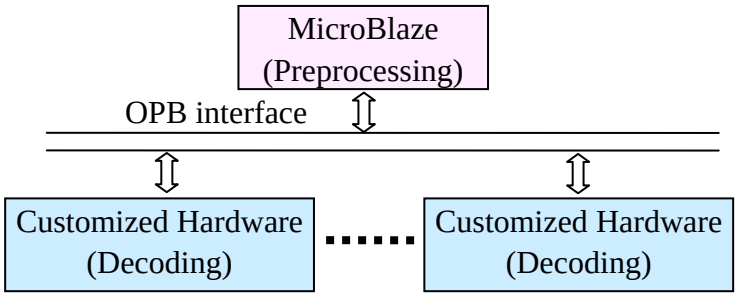

Figure 4.1 Architecture of the processor level parallelism

The HW/SW co-design architecture for both AV and VB algorithms is illustrated in Figure 4.1. The preprocessing part including matrix factorization and inversion is programmed on MB soft processor for both algorithms. The lattice searching procedure is implemented in customized hardware modules to improve the decoding rate. The OPB interface is used to transfer data between them. The processor level parallelism refers to the concurrent execution of the preprocessing stage and the decoding procedure. And multiple decoding blocks can be mapped onto the FPGA. This is feasible because all signals received within the same frame length are decoded using the same triangular matrix decomposed by the preprocessing unit. Thus multiple received vectors can be decoded simultaneously. The maximum number of the customized modules that can be implemented is determined by the size of the hardware core and the available resources of chosen FPGA.

To evaluate the performance gain brought by the processor level parallelism, t_p is defined as

the execution time of the preprocessing part, and t_d is defined as the average execution time to decode one received vector. Then the system architecture with 1 MB processor and N customized hardware modules can achieve a speedup factor of:

$$S_f = \frac{t_p + t_d L}{\max(t_p, t_d L / N)} \qquad \text{Eq 4.1}$$

where L is the number of received vector per frame length. There exists a slight overhead before starting a new searching procedure in hardware modules, because the received vectors need to be manipulated for different hardware modules based on the basis reduction schemes in MB soft processor. This overhead is not significant because only the multiplication of a $1 \times N$ vector with an $N \times M$ orthogonal matrix is involved. Furthermore, instead of execution in MB soft core, if this multiplication is partitioned into each hardware module, this kind overhead can be almost ignored because of the parallel executions in FPGA. Based on the EDK synthesis results shown in Chapter 5, the XC2VP30 FPGA can accommodate 4 hardware decoders for the AV algorithm and 3 hardware decoders for the VB algorithm in parallel to accelerate the decoding speed.

4.3 Complex constellation parallelism

In wireless communications, the transmitted signals are often modulated using the complex quadrature amplitude modulation scheme. If this is the case, the MIMO channel matrix H becomes an $M \times N$ complex matrix, and the received signal $\bar{y}$ and the transmitted signal $\bar{u}$ becomes $1 \times N$ and $1 \times M$ complex vectors respectively. In this subsection, the MIMO systems with the same number of transmit and receive antennas (M=N) are considered to achieve the complex constellation parallelism.

4.3.1 Conventional 2M-dimensional search method

Generally, the MIMO system can be described as:

$$\bar{y} = \bar{u}H + \bar{n} \qquad \text{Eq 4.2}$$

The sphere decoders search the closest lattice point in a specified order as discussed in chapter 2. But in the complex constellation case, the admissible order in each searching layer can not be easily specified because the examining index in each layer becomes a complex number. A solution for this problem is to linearly transform Eq 4.2 into [35]:

$$\begin{bmatrix} \Re\{\bar{y}\} \\ \Im\{\bar{y}\} \end{bmatrix}^T = \begin{bmatrix} \Re\{\bar{u}\} \\ \Im\{\bar{u}\} \end{bmatrix}^T \begin{bmatrix} \Re\{H\} & \Im\{H\} \\ -\Im\{H\} & \Re\{H\} \end{bmatrix} + \begin{bmatrix} \Re\{\bar{n}\} \\ \Im\{\bar{n}\} \end{bmatrix}^T \qquad \text{Eq 4.3}$$

where $\Re\{x\}$ and $\Im\{x\}$ represent the real part and imaginary part of a complex number x. Thus the M-dimensional complex decoding problem can be solved via decomposing the complex matrix into an equivalent 2M-dimensional (2M-D) real matrix as shown in Eq 4.3.

Thus the inputs of the sphere decoder become a 2M-dimensional channel matrix and a 1×2M received vector. While this method provides a possible approach to the complex MIMO systems, the searching space is greatly enlarged by the doubled dimension. 2M layers are involved in the decomposed channel matrix, thus significantly increases the number of iterations during the closest lattice point searching procedure. Figure 4.2 shows the number of states visited as the dimension of MIMO channel matrix grows for the AV algorithm. Number of states visited is increased greatly when the dimension of MIMO system doubles. And as the number of states visited is an important factor of the decoding rate, the data throughput of the MIMO decoder will be considerably degraded if the dimension of the system is doubled. In addition, decomposing

the complex lattice into a 2*M*-dimensional real lattice ignores the orthogonal feature of real and imaginary parts of a complex system and the symmetry of the QAM constellation, thus reducing the possibility of parallel implementations on FPGA. Consequently, the conventional 2*M*-D method is not suitable for FPGA implementations.

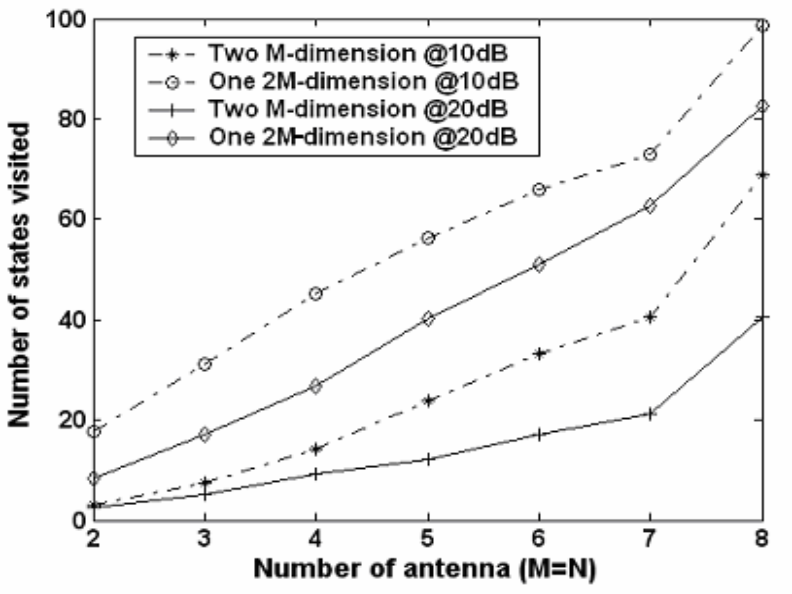

Figure 4.2 Number of states visited vs. number of antennas

An efficient two *M*-dimensional (two *M*-D) transform method is mathematically derived for an alternative solution to the complex MIMO decoders. And the simulation results are presented to verify its performance.

4.3.2 Two M-dimensional search method

The idea of the two *M*-D transform method is to decompose the *M*-dimensional complex MIMO decoding problem into two independent *M*-dimensional real problems. Each decomposed subsystem only contains the real part or imaginary part of the transmitted signals. Hence the complex lattice decoding problem is solved by decoding the real part and imaginary part of the transmitted signal separately.

Eq 4.3 can be rewritten into the following two equations by multiplying out the matrix H and vector $\bar{u}$:

$$\Re\{\bar{y}\} = \Re\{\bar{u}\}\Re\{H\} - \Im\{\bar{u}\}\Im\{H\} + \Re\{\bar{n}\} \quad \text{Eq 4.4}$$

$$\Im\{\bar{y}\} = \Re\{\bar{u}\}\Im\{H\} + \Im\{\bar{u}\}\Re\{H\} + \Im\{\bar{n}\} \quad \text{Eq 4.5}$$

By calculating Eq4.4 + Eq4.5* $\Re^{-1}\{H\} * \Im\{H\}$, we can get:

$$\begin{aligned}\Re\{\bar{y}\} + \Im\{\bar{y}\}\Re^{-1}\{H\}\Im\{H\} = \Re\{\bar{u}\}(\Re\{H\} + \Im\{H\}\Re^{-1}\{H\}\Im\{H\}) \\ + \Re\{\bar{n}\} + \Im\{\bar{n}\}\Re^{-1}\{H\}\Im\{H\}\end{aligned} \quad \text{Eq 4.6}$$

Similarly, by solving Eq4.4* $\Re^{-1}\{H\} * \Im\{H\} -$ Eq4.5, we have:

$$\begin{aligned}\Im\{\bar{y}\} - \Re\{\bar{y}\}\Re^{-1}\{H\}\Im\{H\} = \Im\{\bar{u}\}(\Re\{H\} + \Im\{H\}\Re^{-1}\{H\}\Im\{H\}) \\ + \Im\{\bar{n}\} - \Re\{\bar{n}\}\Re^{-1}\{H\}\Im\{H\}\end{aligned} \quad \text{Eq 4.7}$$

To simplify these equations, we denote the following terms:

$$\begin{aligned}\bar{y}_1 &= \Re\{\bar{y}\} + \Im\{\bar{y}\}\Re^{-1}\{H\}\Im\{H\} \\ \bar{y}_2 &= \Im\{\bar{y}\} - \Re\{\bar{y}\}\Re^{-1}\{H\}\Im\{H\} \\ S &= \Re\{H\} + \Im\{H\}\Re^{-1}\{H\}\Im\{H\} \\ \bar{n}_1 &= \Re\{\bar{n}\} + \Im\{\bar{n}\}\Re^{-1}\{H\}\Im\{H\} \\ \bar{n}_2 &= \Im\{\bar{n}\} - \Re\{\bar{n}\}\Re^{-1}\{H\}\Im\{H\}\end{aligned} \quad \text{Eq 4.8}$$

By substituting Eq 4.8 into Eq 4.6 and Eq 4.7, we get:

$$\bar{y}_1 = \Re\{\bar{u}\}S + \bar{n}_1 \quad \text{Eq 4.9}$$

$$\bar{y}_2 = \Im\{\bar{u}\}S + \bar{n}_2 \quad \text{Eq 4.10}$$

Thus the complex lattice decoding problem of Eq 4.2 can be decomposed into two independent M-dimensional real decoding problems shown in Eq 4.9 and Eq 4.10. This transformation is well suited for FPGA-based implementations because the real and imaginary parts of the transmitted signal can be decoded in parallel, which is named the complex

constellation parallelism in the sphere decoder architecture. Since the number of states visited for each decoding part remains the same as an *M*-dimensional real system, this method can significantly improves the decoding rate for a MIMO system with complex modulation scheme. Of cause there exist some overheads to achieve this performance gain, because the matrix *S*, vectors $\bar{y}_1$ and $\bar{y}_2$ in Eq 4.8 need to be calculated before the preprocessing and lattice searching procedure starts. But by partitioning the calculation of *S* into MB soft core, and $\bar{y}_1$ and $\bar{y}_2$ into FPGA hardware modules, this parallelism is still able to meet the preprocessing time requirement and greatly improve the throughput of the sphere decoder. The architecture of the complex constellation parallelism is illustrated in Figure 4.3, as the R-decoder and I-decoder denote for the real part and the imaginary part decoders respectively.

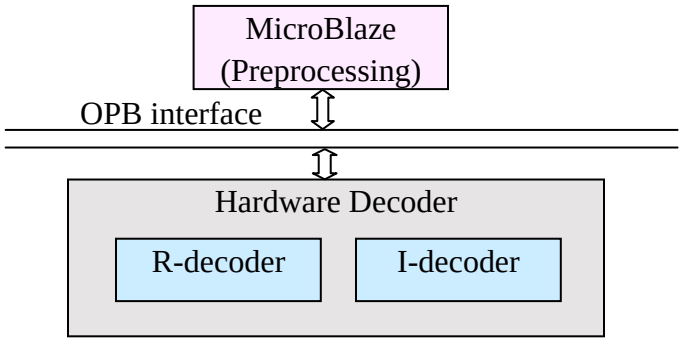

Figure 4.3 Architecture of complex constellation parallelism

The transformed matrix *S* is calculated in MB soft processor. Then it is decomposed in the preprocessing stage to generate the triangular matrix. The results are transferred to hardware decoder along with an intermediate matrix $\Re^{-1}\{H\}\Im\{H\}$, which is used to calculate $\bar{y}_1$ and $\bar{y}_2$ as in Eq 4.8. At last the vectors $\bar{y}_1$ and $\bar{y}_2$ are decoded concurrently in the R-decoder and I-decoder to recover the real and imaginary parts of the transmitted signal.

4.3.3 Simulation validation

After mathematically deriving the two *M*-D transform method for complex MIMO decoders, the MATLAB simulations are carried out to validate its BER performance with the conventional 2*M*-D method. 4×4 MIMO decoders with 16-QAM modulation scheme are simulated for both AV and VB algorithms. The BER performances are compared between the aforementioned two decoding transformations at different SNRs. The results are presented in Figure 4.4.

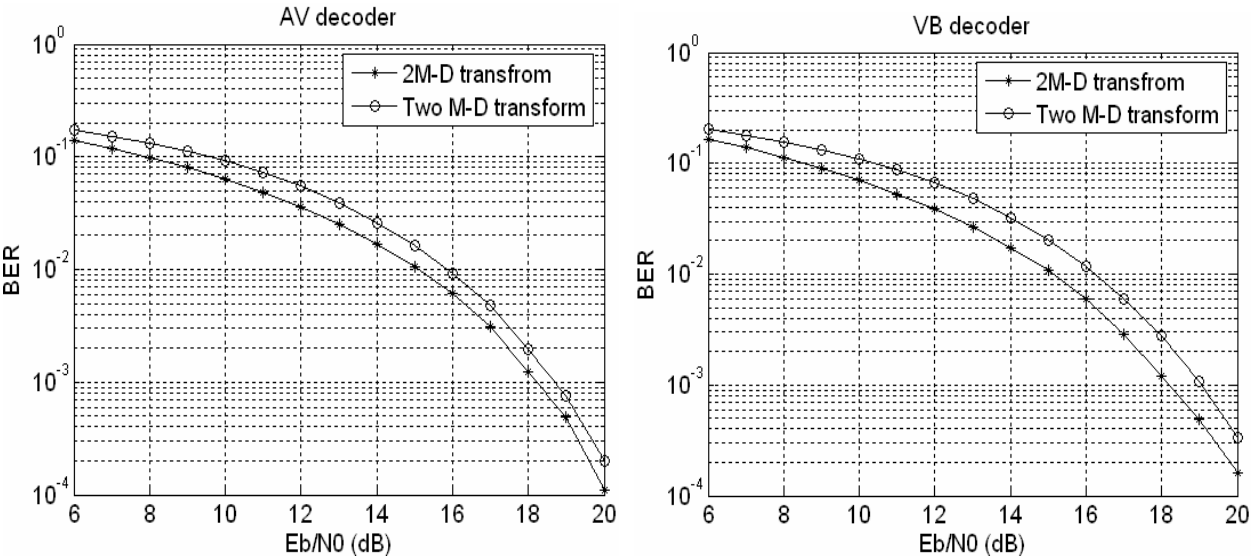

Figure 4.4 Comparisons of BER performances of two transformation methods

According to the simulation results, the two *M*-D transform method results in a slight degradation in BER performance for both AV and VB algorithms. This is because some distortions have been involved by this two *M*-D method to improve the decoding efficiency. In the basis reduction and 2*M*-D transformations, the original channel matrix and the transformed matrix are identical because one can be obtained from the other through scaling, rotation, and reflection [8]. And the BER performance will not be degraded by these kinds of transforms because these lattice generation matrices are able to generate the same lattice points to decode by

the sphere decoding algorithms if the same signal constellation is used. Unlike these identical transforms, the two *M*-D transform reduces the points in the lattice structure from 2^{MQ} to $2*2^{MQ/2}$ by Eq 4.9 and 4.10, where *M* and *Q* denote for the number of transmit antennas and the number of bits used to represent a symbol in the signal constellation. Although only basic transforms are used, there are still some distortions involved by this nonlinear two *M*-D transformation. And Figure 4.4 shows that the VB decoder suffers a bigger distortion than the AV decoder. But even in the VB decoder, the two *M*-D method brings only about 1 dB degradation in BER performance compared with the 2*M*-D method. Furthermore, by considering the reduction of the number of states visited, as shown in Figure 4.5, it is clear that the proposed two *M*-D method can considerably accelerate the decoding procedure and meanwhile keeps the BER performance within an acceptable degradation range.

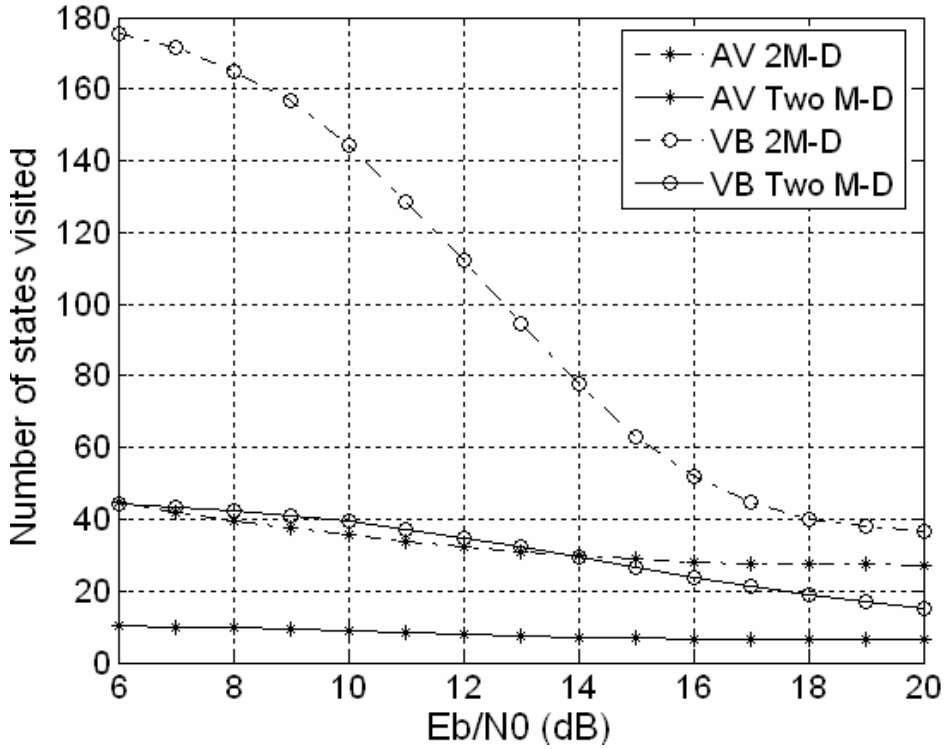

Figure 4.5 Comparisons of number of states visited of two transformation methods

4.4 State level parallelism

To further improve the decoding speed, a state level parallelism is developed based on the data flow dependency analysis between all the states shown in Figure 2.2 and 2.4. A finite state machine (FSM) is designed to control the transitions among these states. This state level parallelism is implemented in the FPGA hardware modules.

4.4.1 Data flow dependency analysis

There are three states involved in the decoding procedure as described in the step by step demonstration in Figure 2.4 for the AV algorithm. Figure 4.6 shows the data dependency between all possible state transition pairs. State A is dependent on both states B and C if the search procedure switches to A from B or C, because the orthogonal distance d calculated in B or C is used to upgrade the layer index u_i in State A. Similarly, State A is self dependent. State B has no dependency on A because these two states work on different search dimensions if B follows A during the decoding procedure. For the same reason State C is not dependent on any other states that could jump to C. Based on the data flow dependency, the possibility of the parallelism among these three states is found as: A || B, A || C, B || C, C || C. These conditions are used to implement the state level parallelism for the AV algorithm.

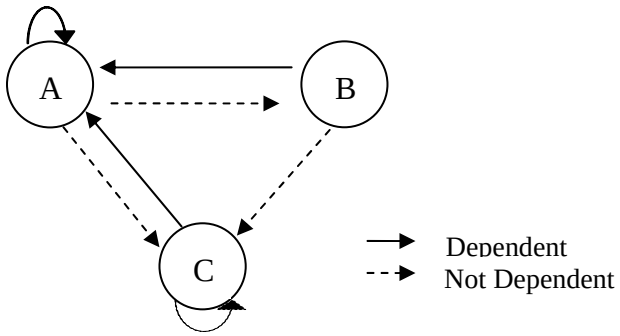

Figure 4.6 Data flow dependency graph of AV algorithm

Similar to the AV algorithm, the data flow dependency for the VB algorithm can be drawn in Figure 4.7. Because of high data dependency among the decoding procedure, only State C can be executed simultaneously with states B or C in this algorithm, which is summarized as: B||C, C||C.

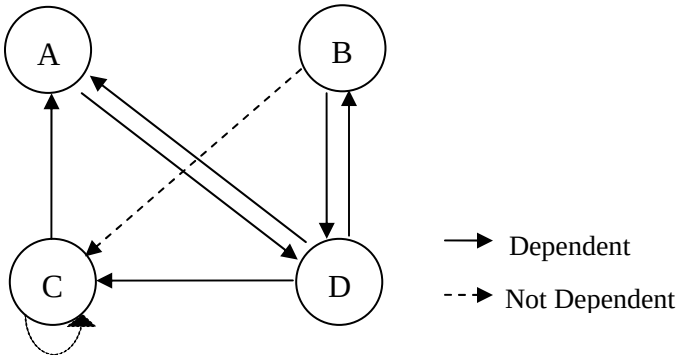

Figure 4.7 Data flow dependency graph of VB algorithm

4.4.2 Hardware architecture of state level parallelism

Depending on the data flow dependency analysis above, multiple states can be executed concurrently in the hardware decoding modules. As an example, Figure 4.8 presents the hardware architecture of state level parallelism for the AV decoder. An FSM is designed as a central unit to control the state transitions and synchronization between them. For a 4-antenna MIMO system, five hardware state modules are created, with one for each state and other two duplicated C states. This is because up to three State Cs can be executed in parallel based on the data flow analysis and experimental results. For instance, when the current state is B, FSM enables not only State B but also all three State C models. The results from these modules can be directly used if this State B is followed by multiple C states. A data buffer unit is used to temporally store the data during the decoding procedure.

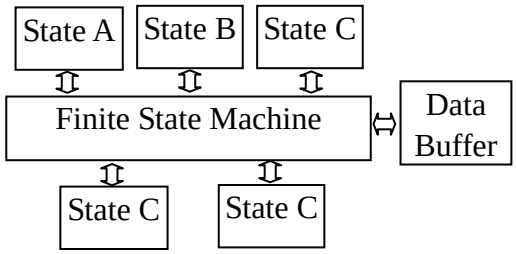

Figure 4.8 Architecture of state level parallelism for AV decoder

The state parallel structure for the VB decoder is similar to that of the AV decoder. There are six state modules for a 4-antenna MIMO system compared with Figure 4.8, with an extra State D involved in the VB algorithm.

To see the performance gain brought by this state level parallelism, Figure 4.9(a) shows a typical searching procedure in the AV decoding algorithm, where the state transition is executed sequentially. Three hardware modules are generated and 7 time cycles are needed to finish the searching procedure. In the state parallel structure, five hardware modules are created to speed up the decoding procedure as shown in Figure 4.9(b). All possible states after the current one are enabled if they are allowed to execute in parallel. For example, if current state is A, modules A, B and C are all enabled because the searching procedure could jump to either B or C after State A, and these two states are not dependent on State A. At the end of the processing in State A, the results from either B or C will be accepted based on the state transition conditions. The next state of the selected state is determined and the search procedure continues. By using this structure, the searching procedure shown in this example can be finished in 4 time cycles. Thus 3 time cycles are saved compared with the sequential implementation.

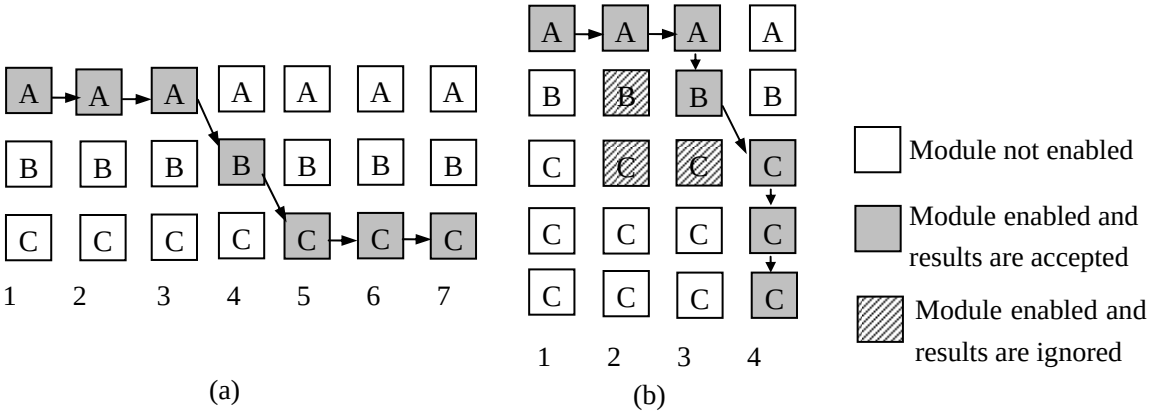

Figure 4.9 An example of state level parallelism for AV decoder
(a) Sequential implementation (b) Parallel implementation

This performance gain is achieved at a cost of more FPGA resource utilization. But this overhead is trivial when compared with the available slices on the XC2VP30 FPGA device. The experimental results are given in the next chapter.

Chapter 5

Performance and results

A 4×4 MIMO system with 16-QAM modulation scheme is chosen as the target application. Both AV and VB sphere decoders are developed in Xilinx Embedded Development Kit (EDK) and prototyped on XUP Virtex-II Pro developing board with an XC2VP30 FPGA. The resource utilization and the maximum frequency of the decoders are obtained from the synthesis results by Xilinx ISE tool. The bit accurate models for both algorithms are simulated in MATLAB, and are used to verify the results from the hardware/software co-design implementations. The two algorithms are also implemented on TI DSP processor for decoding rate comparisons.

5.1 MicroBlaze soft core prototype performance

As discussed in Chapter 4, the preprocessing part of the sphere decoding algorithms is partitioned into MB soft core, which includes QR or Cholesky factorization, matrix inversion and transpose, and other operations. In order to apply the constellation parallelism for complex MIMO systems, the real/imaginary (R/I) decomposition is also calculated in MB soft processor, which includes matrix inversion and multiplication. From the synthesis results, the MicroBlaze soft processor can operate at 100 MHz when prototyped on the XC2VP30 FPGA. The soft core timing performance for the 4×4 MIMO system is presented in Table 5.1.

It is demonstrated from the results that the R/I decomposition and preprocessing parts of the sphere decoders can be completed within 0.8 ms. A MIMO system with 8 transmit and 8 receive antennas has also been examined. The results show that the MB executions can be completed within 3.8 ms for both algorithms, which is still much less than the typical frame length (10 ms)

of a MIMO system. These results prove the feasibility of the processor level parallelism for both AV and VB sphere decoders.

	AV algorithm	VB algorithm
Device	Xilinx MicroBlaze	
Frequency	100 MHz	
R/I decomposition	25,253 cycles	
Factorization	36,277 cycles (QR)	4,621 cycles (Cholesky)
Inversion	12,287 cycles	
Transpose	1,129 cycles	
Total Time	78,675 cycles 0.787 ms	57,801cycles 0.578 ms

Table 5.1 MicroBlaze soft core timing performance

5.2 Hardware synthesis performance

Based on the HW/SW co-design architecture described in Chapter 4, the closest lattice point searching procedure is mapped on FPGA as customized hardware modules to speed up the decoding rate. Because the constellation and state level parallelisms are applied in the sphere decoding architecture, it is meaningful to examine the overheads of FPGA resource utilization caused by these two optimizations. The real and imaginary parts of the transmitted vector $\bar{u}$ are decoded separately after the two *M*-D transformation. Table 5.2 presents the number of slices and embedded multipliers used in single real part or imaginary part decoders for both algorithms. R-I decomposition is used to calculate $\bar{y}_1$ or $\bar{y}_2$ in Eq 4.8, and two extra C states are used to

achieve the state level parallelism. Compared with total slices used by the decoder, the slice overhead caused by these parallel structures are trivial. Although 8 extra embedded multipliers are involved in the proposed parallel architecture, it is still acceptable when compared with the total number of available embedded multipliers provided by XC2VP30 FPGA.

	AV decoder		VB decoder	
	Slices	Embedded Mult	Slices	Embedded Mult
R-I decomposition	245	4	245	4
Two extra C states	170	4	128	4
Single decoder	1705	22	2419	31

Table 5.2 Resource utilization of real or imaginary part decoder

From the post place-and-route (PAR) report, the FPGA resource utilization and the maximum frequencies are compared between the AV and VB algorithms for the 4×4 complex MIMO system as shown in Table 5.3:

	AV algorithm	VB algorithm
Target FPGA	XC2VP30	
Slices	3410 out of 13696	4838 of 13696
External IOBs	205 out of 896	227 out of 896
Embedded Multipliers	44 our of 136	62 our of 136
Maximum Frequency	251 MHz	257 MHz

Table 5.3 FPGA resource utilization for AV and VB sphere decoders

The results show that the VB-based decoder uses more embedded multipliers and more FPGA slices than the AV-based decoder because the VB algorithm involves more complicated calculations.

5.3 Decoding rate performance

The data rate for the MIMO sphere decoders with M transmit and M receive antennas is determined by:

$$R = \frac{f * b_{\dim} * M}{n_{state} * C_{state}} \quad \text{(Mbps)} \qquad \text{Eq 5.1}$$

where f is the synthesized system frequency in MHz, $b_{\dim}$ is the number of bits per dimension which is determined by the modulation scheme. Because 16-QAM signal constellation is chosen in our designs, 4 bits per dimension are transmitted in each symbol. And n_{state} is the average number of states visited required to decode a received vector at a certain SNR. With the state level parallelism applied to the sphere decoder architecture, two or more states executed at the same time are counted as one state visit. C_{state} is the average number of clock cycles per state visit, which is calculated as:

$$C_{state} = \sum (P_{i,state} * C_{i,state}) \qquad \text{Eq 5.2}$$

where $P_{i,state}$ denotes the visiting statistic percentage of state i, which is obtained from high-level simulation. $C_{i,state}$ is the clock cycles used by state i captured from the FPGA RTL level simulation.

In section 2.3.1, it is discussed that the initial radius $\sqrt{C}$ is very crucial to the performance of VB-based sphere decoders. An adaptive method is adopted to calculate this squared initial

radius as shown in Eq 5.3:

$$C = round(d^2(\bar{y}, ST(\bar{y} * H^{-1}) * H) + \alpha) \qquad \text{Eq 5.3}$$

where $d^2(\bar{x}, \bar{y})$ denotes the squared Euclidian distance between two vectors $\bar{x}$ and $\bar{y}$. $ST(\bar{x})$ is a function that transfers each element in vector $\bar{x}$ into its closest signal constellation point. And α is an adaptive positive number based on experience, which is chosen to be 0.2 for our 4×4 MIMO system with 16-QAM modulation. The idea here is to find a distance from the received point to a nearby lattice point, and then use this distance as the initial radius in the VB sphere decoding algorithm. Because in MIMO wireless communications, the channel is considered to be static during a short length of time, the squared radius *C* only needs to be upgraded once in a signal frame length. An adaptive number α is added to this *C* in order to keep the BER performance. The simulation results of number of states visited for different initial radiuses are compared in Figure 5.1 for the VB decoder.

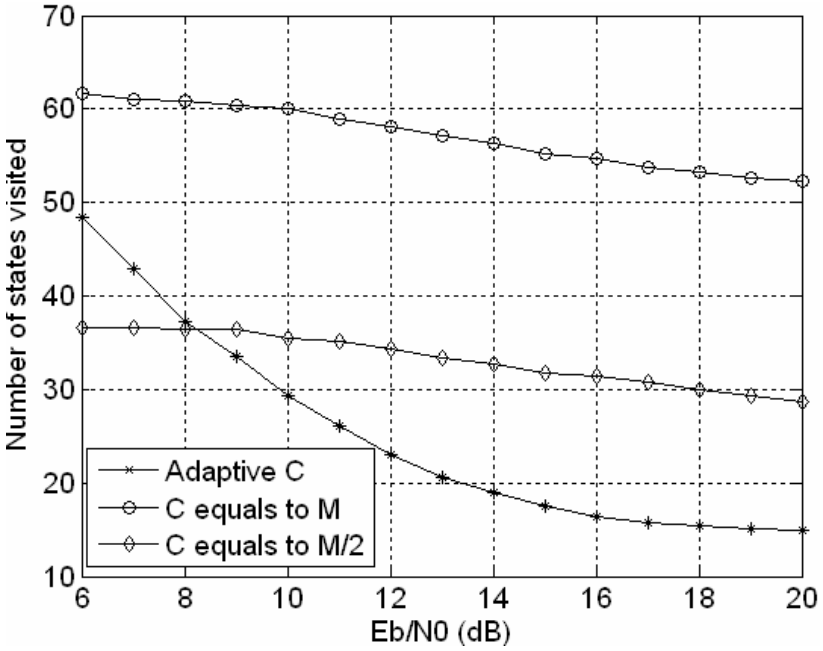

Figure 5.1 Number of states visited for different initial radiuses in VB decoder

Figure 5.1 shows that when the Eb/N0 (bit-noise ratio) is greater than 8 dB, the number of states visited is significantly reduced by the adaptive method compared with other two fixed *C* cases. And from BER simulations, all these three methods can achieve almost the same BER performances. So it is concluded that the adaptive method is able to improve the decoding rate without violating the BER performance.

Based on above analysis, the statistic information for the number of states visited and average clock cycles at 20 dB Eb/N0 is given in Table 5.4. The time for data exchange and decision making of FSM module is considered inside each state. The simulation results show that the average processing time for one state visit is 8.5 clock cycles for the AV algorithm and 7.6 clock cycles for the VB algorithm. The average number of states visited for one decoding procedure is 5.8 and 14.5 for the AV and VB decoders respectively.

	AV algorithm		VB algorithm	
	Clock cycles	Percentage	Clock cycles	Percentage
State A	11	50%	6	30%
State B	6	5%	6	10%
State C	6	45%	6	20%
State D	--	--	10	40%
Average	8.5 cycles/state 5.8 states visited		7.6 cycles/state 14.5 states visited	

Table 5.4 Average state visit statistics for AV and VB decoders at 20 dB Eb/N0

These two algorithms are also implemented on DSP in comparison with the proposed FPGA-based sphere decoders. A fixed point TI TMS320C6201 DSP device is chosen as the benchmark platform, which operates at a fixed frequency 200 MHz. Because DSP does not support parallel executions, the 2*M*-D transformation is applied to decode the complex signals in the MIMO system. Based on these parameters, the decoding rate of the sphere decoders of different hardware implementations at 20 dB Eb/N0 is given in Table 5.5. The FPGA results show that the AV-based decoder is more than 2 times faster than the VB-based decoder, which matches the theoretical analysis as in [8].

	FPGA		DSP	
	AV decoder	VB decoder	AV decoder	VB decoder
Platform	Xilinx XC2VP30		TI TMS320C6201	
f (MHz)	251	257	200	
C_{state} (cycles)	8.5	7.6	94	474
n_{state}	5.8	14.5	15.5	33.2
b_{dim}	4			
M	4			
R (Mbps)	81.5	37.3	2.2	0.2

Table 5.5 Comparisons of decoding rate for different implementations at 20 dB Eb/N0

The DSP results show that the AV and VB decoders can only reach about 2.2 Mbps and 0.2 Mbps respectively for our MIMO system at 20dB Eb/N0. Comparing with FPGA-based

performances, the key reason for the low decoding rates in DSP implementations is the lack of parallel executions and high instruction overhead. As shown in the above table, DSP-based VB decoder takes 474 clock cycles in average for a state visit, which is only 7.6 clock cycles in the corresponding FPGA implementation. The number of states visited of the DSP implementation is larger than that of the FPGA implementation because DSP does not support the complex constellation and state level parallelisms designed in Chapter 4. Similarly, the DSP-based decoder can not implement the current execution of preprocessing and decoding parts. As shown in Figure 5.2, the decoding rate of the FPGA-based sphere decoders is about 37 times and 187 times faster than the DSP-based implementations of AV and VB algorithms respectively at 20 dB Eb/N0.

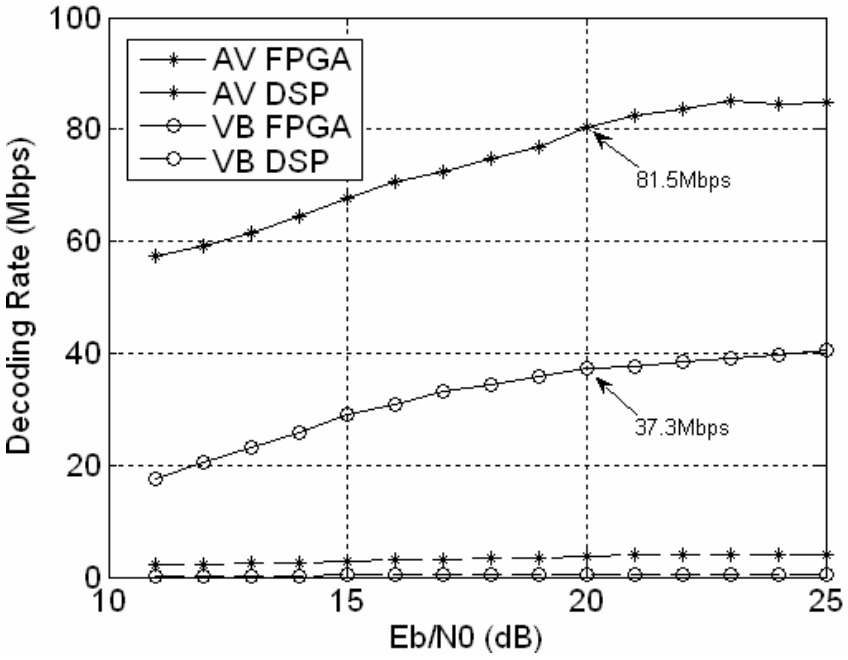

Figure 5.2 Comparisons of decoding rates for different sphere decoders

5.4 BER performance

The BER performance is evaluated for different lattice dimensions, different lattice

generation matrices with Gaussian distribution and different SNRs. Figure 5.3 shows the BER performance for the VB-based sphere decoders. The simulation results show that the 8-antenna MIMO system can achieve a better BER performance than that of the 4-antenna system. Theoretically this is because that if there are more antennas involved in a MIMO system, more antenna diversity gains is obtained to improve the BER performance. From Figure 5.3, it also shows that the fixed point FPGA-based sphere decoders in this thesis match the BER performances from the floating point software simulations. This result proves that in our hardware implementations, the number of bits used to present a fixed point and the scaling number to convert the floating point to a fixed point are suitably chosen which can keep the precision within the examined range of SNRs.

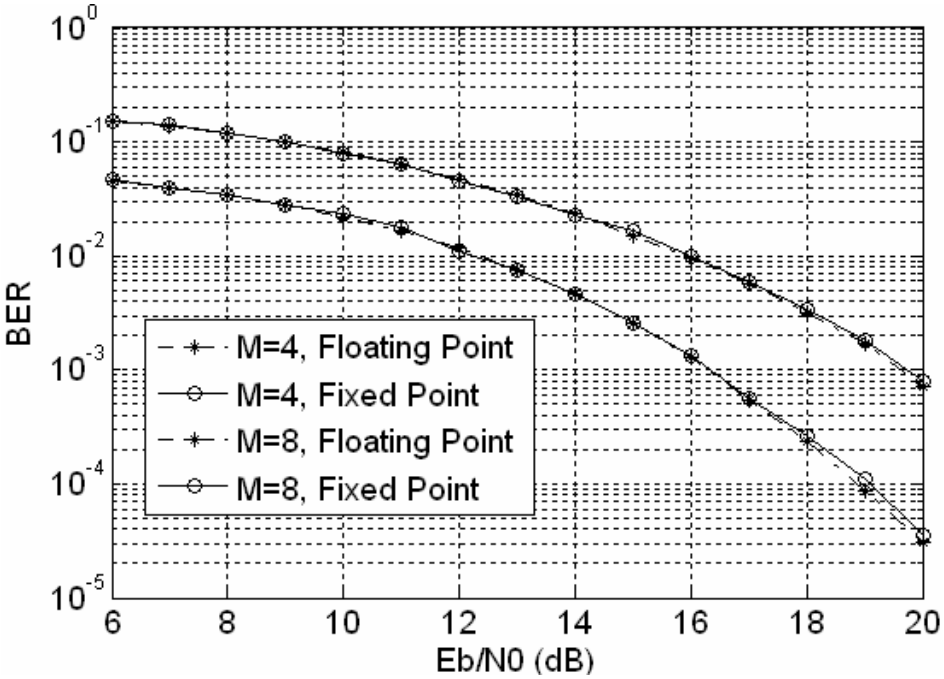

Figure 5.3 BER performances for VB sphere decoders

The BER performance of the AV-based sphere decoder is close to that of the VB-based

decoder as shown in Figure 5.4.

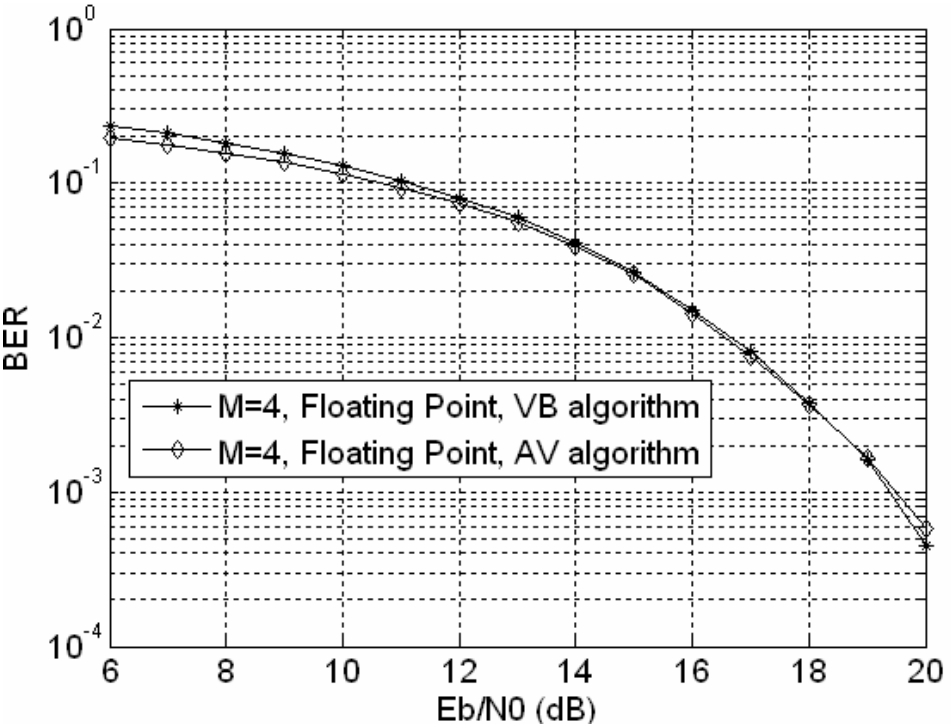

Figure 5.4 Comparison of BER performances for two sphere decoders

Chapter 6

Conclusions

Hardware/software co-design architecture for two typical MIMO lattice decoding algorithms has been designed and implemented in this thesis. The closest lattice point searching procedure is partitioned into the FPGA-based hardware modules. And a MicroBlaze soft core is used for the channel matrix preprocessing and R/I decomposition. Three levels of parallel structures are designed in this co-design architecture to improve the decoding rate. The overheads involved in these parallel structures are also analyzed. The proposed architecture is prototyped on the Xilinx XUP Virtex-II Pro developing board with an XC2VP30 FPGA. The experimental results show that the AV and VB based decoders can reach up to 81.5 Mbps and 37.3 Mbps decoding rate respectively at 20 dB Eb/N0 for a 4×4 MIMO system with 16-QAM modulation, which are among the fastest MIMO decoders to the author's knowledge. They are about 37 and 187 times faster than their respective implementations in a DSP processor. The BER performance of the experimental prototype matches with the software simulation results.

The implementation results show that our FPGA-based HW/SW co-design architecture is a promising solution to design efficient MIMO decoders to match with the high transmission rate in MIMO systems. This thesis research also provides author an excellent opportunity to learn the procedures involved in the HW/SW co-design and to get invaluable experiences on real issue designs.

References:

[1] G. J. Foschini and M. Gans, “On the limits of wireless communication in a fading environment when using multiple antennas,” *Wireless Personal Commun.*, vol. 6, pp. 311–335, Mar. 1998.

[2] B. M. Hochwald and S. ten Brink, “Achieving near-capacity on a multiple-antenna channel,” IEEE Trans. Commun., vol. 51, no. 3, pp. 389–399, Mar. 2003.

[3] D. Gesbert, M. Shafi, D. Shiu, P. J. Smith and A. Naguib, “From Theory to Practice: An Overview of MIMO Space–Time Coded Wireless Systems,” IEEE Journal on Selected Areas in Communications, vol. 21, no. 3, pp. 281-302, Apr. 2003

[4] http://en.wikipedia.org/wiki/Multiple-input_multiple-output_communications

[5] M. Pohst, “On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications,” SIGSAM Bull., vol. 15, no. 1, pp. 37–44, Feb. 1981.

[6] E. Viterbo and J. Boutros, “A universal lattice decoder for fading channels,” IEEE Trans. Inf. Theory, vol. 45, no. 5, pp. 1639–1642, Jul. 1999.

[7] C. P. Schnorr and M. Euchner, “Lattice basis reduction: improved practical algorithms and solving subset sum problems,” Math. Programming, vol. 66, no. 2, pp. 181–191, Sep. 1994.

[8] E.Agrell, T.Eriksson, A.Vardy and K Zeger, “Closet point search in lattice”, IEEE Trans. on Inf. Theory, pp 2201-2214, Aug. 2002

[9] Adjoudani, E. Beck, A. Burg, and ect. “ Prototype experience for MIMO BLAST over third-generation wireless system”, IEEE Journal of Selected Areas in Communications, Vol,.21, No.3, pp 440-451, April 2003

[10] M. Guillaud, A.Burg, M.Rupp, E.Beck, and S. Das, “Rapid prototyping design of a 4x4 BLAST-over- UMTS system”, 35th Asilomar Conference on Signals, Systems and Computers, pp1256-1260, Pacific Grove, CA, USA, November 4-7, 2001

[11] Z. Guo and P. Nilsson. ”A VLSI Architecture of the Soft-Output Schnorr-Euchner Decoder for MIMO Systems”, accepted in IEEE International Midwest Symposium on Circuits and Systems (MID-WEST), Cincinnati, USA, Aug. 2005.

[12] Z. Guo, P. Nilsson and V. Ovall. ”A Low-Complexity High-Throughput Soft-Output MIMO Decoder”, The 14th IST Mobile & Wireless Communications Summit, Dresden, Germany, Jun. 2005.

[13] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, “VLSI Implementation of MIMO Detection Using the Sphere Decoding Algorithm,” IEEE Journal of Solid-State Circuits, vol. 40, no. 7, pp. 1566-1577, Jul. 2005.

[14] http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex4/capabilities/powerpc.htm

[15] http://www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=micro_blaze

[16] http://www.altera.com/technology/embedded/emb-index.html

[17] http://www.xilinx.com/univ/xupv2p.html

[18] http://www.model.com/

[19] http://www.xilinx.com/bvdocs/ipcenter/data_sheet/EDK_Sell_Sheet.pdf

[20] G. Strang, Linear Algebra and Its Applications, 3rd ed. San Diego, CA: Harcourt Brace Jovanovich, 1988.

[21] J. Ma, C. Liang and X. Huang, “A comparison of lattice decoding algorithms in hardware implementation”, Proc. SPIE, Vol. 5819, pp. 200-210, June 2005.

[22] Jing Ma, Xinming Huang, Swapna Kura, “A high data rate universal lattice decoder on FPGA”, Proc. SPIE, Vo1. 5819, pp 211-221, June 2005.

[23] L. Babai, “On Lovász’ lattice reduction and the nearest lattice point problem,” *Combinatorica*, vol. 6, no. 1, pp. 1–13, 1986.

[24] H. Cohen, “A course in Computational Algebraic Number Theory”, Berlin, Germany: Spring-Verlag, pp.102-104, 1993.

[25] W. W. Hager, “Applied Numerical Linear Algebra”, Englewood Cliffs, NJ: Prentice-Hall, pp. 208-236, 1988.

[26] Wolf. W, “A decade of hardware/software codesign”, IEEE Computer Society, vol. 36, pp. 38-43, April 2003.

[27] Clive Maxfield**, "**The Design Warrior's Guide to FPGAS**",** ISBN: 0-7506-7604-3, pp1-10, 2004

[28] Chris H. Dick, "Design and Implementation of High-Performance FPGA Signal Processing Data paths for Software Defined Radios", Xilinx Inc.

[29] http://msdn.microsoft.com/vstudio/

[30] http://www.model.com/products/60/se.asp

[31] http://www.xilinx.com/ise/logic_design_prod/foundation.htm

[32] http://www.xilinx.com/ise/embedded/edk_pstudio.htm

[33] http://www.xilinx.com/univ/XUPV2P/Documentation/XUPV2P_User_Guide.pdf

[34] M.O.Damen, H.E.Gamal, and G.Caire, "On maximum-likelihood detection and the search for the closest lattice point", IEEE Trans. on Inf. Theory, pp 2389-2402, Oct. 2003

[35] http://ls12-www.cs.uni-dortmund.de/~niemann/codesign/codesign.html

Vita

Cao Liang was born in 1979, in Sichuan, China. He received the B.S. degree in Electrical Engineering from University of Electronic Science and Technology of China in 2002. After the graduation, he worked in Sifang Information & Technology Co., Ltd as a software engineer for more than one year. In fall 2004, he began his graduate studies in the EE department at University of New Orleans.

His research is mainly focused on embeded computer systems for high performance computation and wireless communications.