國立交通大學

資訊科學與工程研究所

碩士論文

應用於多輸入多輸出通道之低複雜度多模式訊號偵測演算法與超大型積體電路實現

Low-Complexity Multi-Mode Signal Detection Algorithm and VLSI Implementation for Multiple-Input Multiple-Output Channels

研 究 生：吳廸優

指導教授：范倫達　博士

中華民國九十七年八月

應用於多輸入多輸出通道之低複雜度多模式訊號偵測演算法與超大型積體電路實現

Low-Complexity Multi-Mode Signal Detection Algorithm and VLSI Implementation for Multiple-Input Multiple-Output Channels

研 究 生：吳廸優　　Student：Di-You Wu

指導教授：范倫達博士　　Advisor：Dr. Lan-Da Van

國 立 交 通 大 學
資訊科學與工程研究所
碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

August 2008

Hsinchu, Taiwan, Republic of China

中華民國 九 十 七 年 八 月

應用於多輸入多輸出通道之低複雜度多模式訊號偵測演算法與超大型積體電路實現

學生：吳廸優　　　　　　　　　　指導教授：范倫達 博士

國立交通大學
資訊科學與工程研究所

摘　　要

在本論文中，我們使用平行消除干擾、群組干擾壓縮及遞迴之技術提出一個應用於多輸入多輸出通道之廣義平行群組遞迴(GPGI)偵測框架，並提出一個低複雜度多模式演算法。所提出之偵測框架可調整三項參數與三種子演算法以達到效能與複雜度的取捨。所提出的框架平台不只包含傳統的 BODF 偵測、群組偵測、遞迴偵測與 B-Chase 偵測演算法，並且衍生出一種新的低運算複雜度且多模式之 GPGI-T1 偵測演算法。在使用 8 個傳送天線與 8 個接收天線及輸入不編碼之 16-QAM 符號下，與 BODF 偵測演算法相比，在最低複雜度的情況下，GPGI-T1 演算法能夠減少 33.9%的複雜度，並且在效能上還勝過 10 dB。在另一個情況下，GPGI-T1 演算法與 B-Chase 演算法相比，能夠降低 36.8%的複雜度而只損失 0.4 dB。最後，根據所提出之 GPGI-T1 演算法，我們使用 TSMC 0.18 um 製程實作出一多模式之多輸入多輸出訊號偵測器，可使用在傳送天線與接收天線各為二或四的情況下，支援 QPSK、16-QAM、64-QAM 之調變。並且，在五個特定應用晶片中，此實作有較好的功率效益。

Low-Complexity Multi-Mode Signal Detection Algorithm and VLSI Implementation for Multiple-Input Multiple-Output Channels

Student：Di-You Wu Advisor：Dr. Lan-Da Van

Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University

ABSTRACT

In this thesis, we use parallel interference cancellation (PIC), group interference suppression (GIS) and iteration techniques to construct a generalized parallel grouped-iterative (GPGI) detection framework and one new low-complexity algorithm for multiple-input multiple-output (MIMO) channels. The proposed detection framework provides three parameters and three sub-algorithms to configure a range of tradeoffs between performance and complexity. The presented framework not only covers the conventional BLAST-ordered decision feedback (BODF), grouped, iterative, and B-Chase detection algorithms, but also derives the GPGI-Type 1 (GPGI-T1) detection algorithm with low computational complexity. In (8,8) system with uncoded 16-QAM inputs, one instance of the GPGI-T1 algorithm not only substantially reduces the complexity by 33.9% but also outperforms the BLAST-ordered decision feedback algorithm by 10 dB. Another instance of the GPGI-T1 algorithm can save complexity by 36.8% at the penalty of 0.4 dB loss compared with the B-Chase detector. Last, according to the proposed GPGI-T1 algorithm, we implement a multi-mode MIMO signal detector in TSMC 0.18 um CMOS process. The resulting implementation can work in (2,2) or (4,4) system, and supports QPSK, 16-QAM, and 64-QAM modulation modes. Importantly, the resulting MIMO detection implementation possesses the comparable power efficiency among five ASIC designs.

誌　　　　謝

首先感謝指導教授 范倫達老師在這兩年多以來的悉心指導與建議，並提供我各方面的協助，使我可以確立並完成我的論文研究，並在我的研究過程中所提出的種種建議，讓我找到方向、抓住目標。在此先對范倫達老師致上由衷的感激。

其次是 VIPLab 的夥伴們，特別感謝彥澤、旭昇學長在研究與使用 EDA 軟體上幫助，以及同窗籐耀、晉豪、宗融與得安在研究與課業上的幫忙，並感謝你們在我的研究生生活中所帶來的溫馨與歡笑。

最後要感謝家人和親友們的關心、支持與鼓勵，尤其是親愛的父親 吳明龍先生與母親 陳季蓉女士，你們讓我可以無後顧之憂的完成學業。最後感謝女友瑋婷，謝謝妳一路陪著我從大學一直到研究所，在我低潮、壓力大的時候，始終陪在我身旁，默默忍受著我的脾氣，支持我，讓我快樂的度過這幾年，謝謝！

Contents

List of Tables

List of Figures

Chapter 5

Chapter 1

Introduction

Multiple-Input-Multiple-Output (MIMO) technology can significantly improve data transmission rate in bandwidth-limited wireless communications without increasing transmission power. Much research [1-2] has shown that the channel capacity increases with the number of antennas. Because of the above benefit, the MIMO technique has been considered in modern high-speed wireless communication standard including wireless LAN [3] and mobile wireless MAN. For the MIMO communication systems, the detection scheme is more complex than that in the SISO communication systems. Since the MIMO communications transmit information at very high data rates, the low computational complexity detection algorithm at the receiver is essentially considered.

In terms of detection performance, the maximum likelihood (ML) detection scheme is an optimum solution at the receiver. However, it is manifest that the detection complexity raises as the number of antennas and the constellation size increases. Therefore, the computational complexity of the ML scheme is too huge for hardware implementation and unsuitable for high-speed communications. The sphere decoding (SD) scheme [4]-[6] searching for the closest lattice point inside the radius bounded sphere achieves the same performance of the ML detection with efficient computational complexity. However, the complexity of the SD algorithm is unstable owing to the variation of the iteration number which is higher at low signal to noise ratio (SNR) environment especially. Hence, the SD algorithm has higher computational complexity at low SNR communication environment due to the larger iteration numbers. On the other hand, the variable throughput of the SD algorithm also affects the system performance. The Bell Laboratories layered space-time (BLAST) wireless communication system [1] uses multi-element antenna arrays at both the transmitter and receiver to achieve high spectral efficiency. This technology is referred to as the diagonal BLAST (D-BLAST). The D-BLAST theoretically approaches the Shannon capacity for multiple transmitters and receivers, but the D-BLAST is complex and impractical. The vertical BLAST (V-BLAST) system [7], [8] is a simplified architecture of the D-BLAST, where the BLAST-ordered decision feedback (BODF) detection algorithm named in [16] (also called successive interference cancellation (SIC) detection algorithm named in [19]) is applied. Although the BODF algorithm has low computational complexity, the poor bit-error rate (BER) performance is incurred. Other efficient implementations of the BODF algorithm [9], [10] aim at low-complexity computation but still possess poor BER performance.

1.1 Motivation

Many researchers currently concentrate on developing detection algorithms in both complexity and performance between the ML and BODF detection algorithms [11], [12]. The above research work divides symbols into two groups by two schemes. The QR-decomposition [11] is partially applied to the channel matrix such that two sub-channels are orthogonal to each other. The second scheme [12] uses group interference suppression (GIS) technique [13] to divide the V-BLAST system into two lower dimensional sub-systems. After group partition, the first-group symbols are detected by the ML detection and the second-group symbols are detected by a suboptimal algorithm after cancelling the interference from the first-group symbols. Although the previous published schemes using the ML and suboptimal detection algorithms can achieve better performance, the high computational complexity is incurred. Thus, we are motivated to devise a MIMO detection algorithm that features the low computational complexity and satisfactory performance. Furthermore, in order to trade off the performance and complexity for different demands, we develop a framework to cover the previous and proposed algorithms.

1.2 Thesis Organization

This thesis is organized as follows. Brief review of the MIMO detection algorithms is described in Chapter 2. In Chapter 3, one generalized parallel grouped-iterative (GPGI) framework has been presented. In the same chapter, how to generate existing algorithms through this framework will be discussed. In Chapter 4, we propose one new low complexity detection algorithm via this framework. We present the complexity analysis, performance simulation and chip implementation results in Chapter 5. Last, the conclusions are presented.

Chapter 2

Review of the MIMO Detection Algorithms

In this chapter, the MIMO system model will be given, and introduce some existing MIMO detection algorithms.

2.1 MIMO System Model

A MIMO system with N transmit antennas and M receive antennas is considered in this thesis as shown in Fig. 2.1. The discrete-time received signal **r** can be written as

$$\mathbf{r} = \mathbf{H}\mathbf{s} + \mathbf{n}, \qquad (1)$$

where **s** denotes the $N\times 1$ vector of the simultaneous transmitted symbols that selects from constellation C, and $|C|$ denotes the constellation size. **H** is the $M\times N$ equivalent channel transfer matrix, **n** is the $M\times 1$ complex white Gaussian noise vector with zero mean and variance of σ_n^2. In this thesis, the elements in **H** are assumed to be independent identically distributed (IID) complex Gaussian random variable with zero mean, where the dimension is under $M \geq N$. It is assumed that the receiver knows channel matrix **H** perfectly. This is shown that the ML detector is an optimum solution for the receiver in which the scheme detects all sub-stream symbols jointly by choosing the symbol vector which maximizes likelihood function. The above treatment is equivalent to the minimum Euclidean distance (MED) function in (2).

$$\mathbf{s} = \arg\min_{i} \left\| \mathbf{r} - \mathbf{H}\mathbf{s}_i \right\|^2, \tag{2}$$

where $\|\mathbf{x}\|$ denotes 2-norm of the vector $\mathbf{x}$ and $\mathbf{s}_i$ denotes i-th candidate choosing from all possible combination of symbols. Note that the number of all combinations is $|C|^N$. Nevertheless, the high computation-complexity ML scheme blocks the VLSI implementation. Several low-complexity detection algorithms [11-19] have been widely studied. Herein, we briefly review the complexity-oriented algorithms as follows.

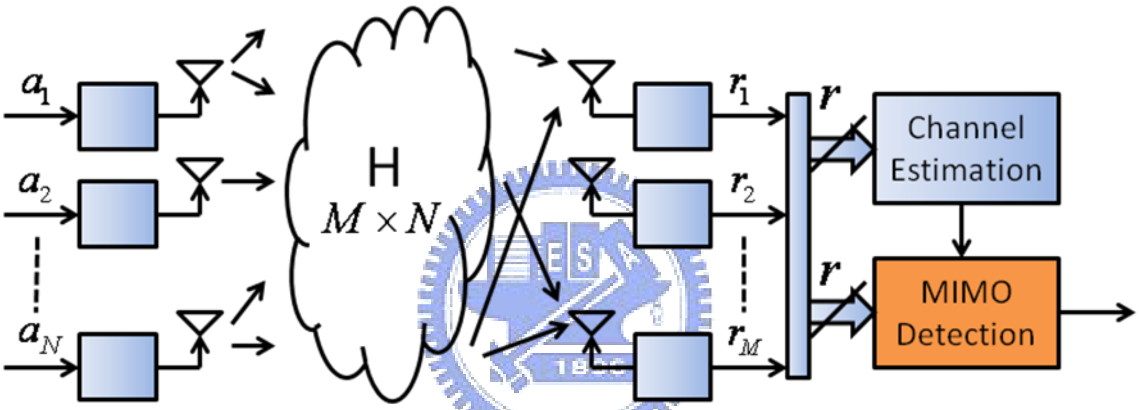

Fig. 2.1: A MIMO system with N transmitters and M receivers.

2.2 Grouped Detection (GD)

The grouped detection algorithm [12] applies the ordering, GIS [13], ML algorithm to the first group symbols, interference canceling (IC), and BODF algorithm to the second group symbols as shown in Fig.2.2. The GIS not only plays the role of dividing symbols into two groups but also suppresses the performance influence of the low SNR signals. After ordering symbols, the ML detection algorithm at the first group is employed to detect higher SNR signals. Because of the property of the ML algorithm, we can detect symbols at the early stage and guarantee the performance without error propagation. The remaining symbols at the second group disturbed by high noise power can be detected by a suboptimal algorithm such as the BODF detection algorithm [7]-[10].

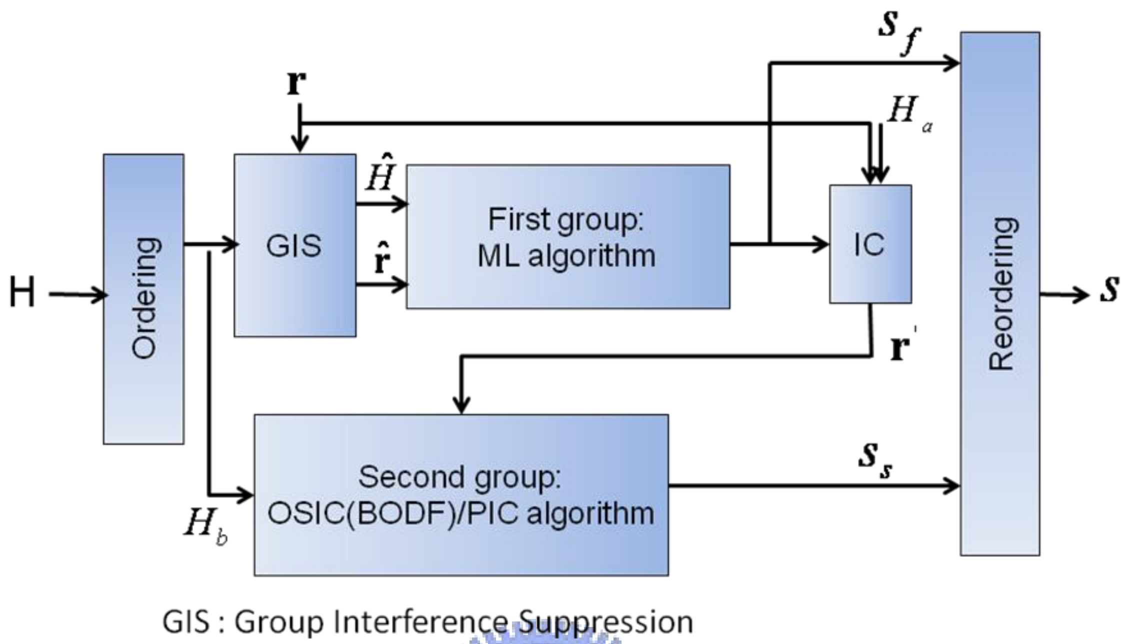

Fig. 2.2: Block diagram of the grouped detection.

2.3 Iterative Detection (ID)

The iterative detection algorithm detects symbols iteratively was proposed in [14], [15]. The traditional BODF algorithm detecting each symbol once propagates errors owing to the low-diversity symbols and thus greatly constraints the overall system performance. The algorithm detects symbols repeatedly in some specific sequence such that low-diversity symbols are detected by using decisions from high-diversity symbols to retrieve the high diversity gain. Enhancing diversity for all symbols can decrease error propagation. An example is provided in Fig. 2.3 to show the detection flow using the iterative detection [15].

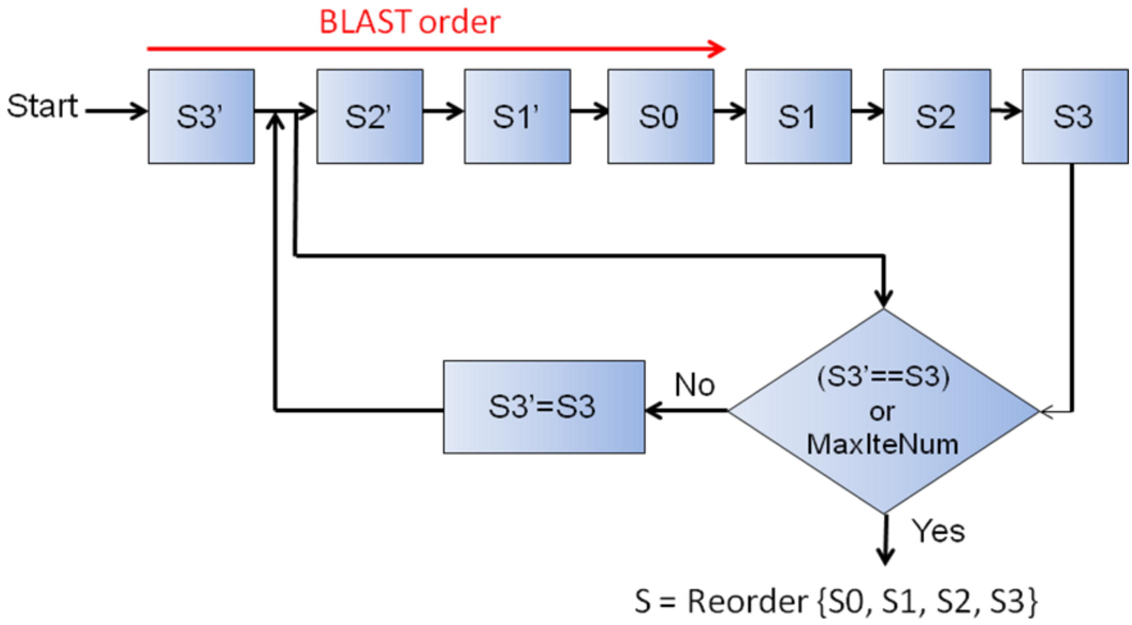

Fig. 2.3: An Example of the iterative detection at 4 transmitted symbols.

2.4 Chase Detection

The Chase detection algorithm [16], [17] which shown in Fig. 2.4 determines which symbol detected first, list length, filter type, and sub-detector algorithm for the MIMO detection application. Many detection algorithms including ML, BODF, parallel [18], B-Chase and S-Chase can be derived from the Chase detection algorithm by adjusting the above four parameters. Table 2.1 shows how to generate the different detection algorithms. The B-Chase detection based on the BODF algorithm provides a tradeoff between the complexity and performance by choosing the list length. When the list length equals the constellation size, the performance of the B-Chase detection is close to that of the ML detection. Although the SD algorithm has better performance than the above Chase detector does, the SD detector shows larger computational complexity in [16]. For example, in [16], at BER=10^{-3}, the SD and B-Chase algorithms respectively own the complexity of 57 RM/b and 18 RM/b, where RM/b represents the required number of real multiplications per detected bit. Hence, in this thesis, we consider the Chase detection algorithm for complexity comparison instead of the SD algorithm.

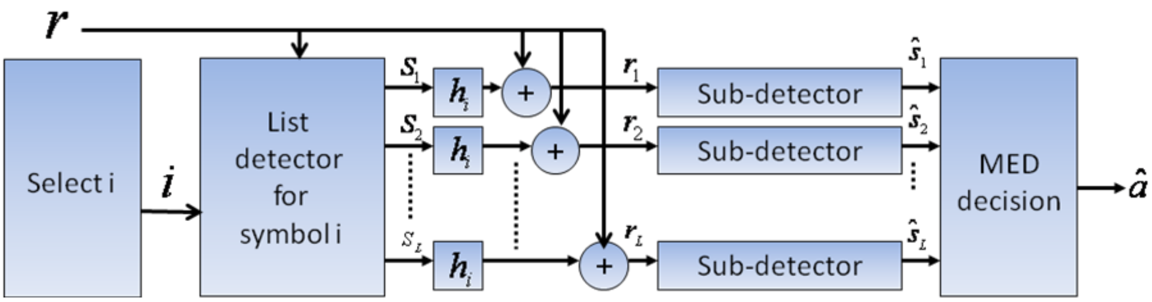

Fig. 2.4: Block diagram of the Chase detection.

Table 2.1: Cases of the Chase detection algorithm

Detector	First-symbol index i	List length L	Filter type	Sub-detector
ML	any	$\|C\|$	ZF*1	ML
BODF [7]-[10]	BLAST order	1	ZF or MMSE*2	BODF
Parallel [18]	Selecting algorithm 1	$\|C\|$	ZF	any
B-Chase [16]	Selecting algorithm 1 or 2	$1 \leqq L \leqq \|C\|$	ZF or MMSE	BODF

*1ZF : Zero forcing

*2MMSE : Minimum mean square error

2.5 GPIC Detection

The generalized parallel interference cancellation (GPIC) detection algorithm [19] is similar to the Chase detection algorithm. The GPIC detection uses two PIC techniques; one is the same as that of the Chase detection algorithm, and another is referred to as a redetection scheme. Fig. 2.5 shows the block diagram of the GPIC detection. For the first PIC technique, the GPIC extends the number of the symbols detected first compared with the Chase detection. In this case, the number of list lengths is the same as the number of all possible combinations of the symbols detected first. Then, the GPIC detection applies the redetection scheme to detect residual symbols, where the redetection scheme uses linear detection (LD) algorithm for lower computational complexity.

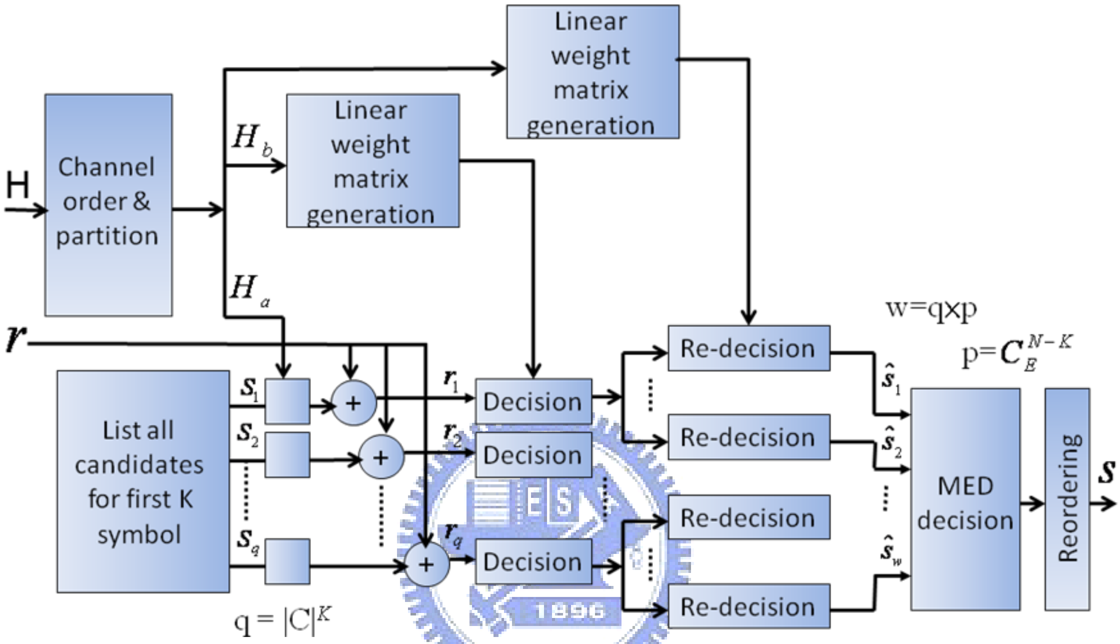

Fig. 2.5: Block diagram of the GPIC detection.

Chapter 3

Generalized Parallel Grouped-Iterative (GPGI) MIMO Detection Framework

In this chapter, we develop the generalized parallel group-iterative (GPGI) framework. Through this framework, we not only generate several previously reported detection algorithms including BODF, GD, ID, B-Chase, GPIC(K,0) detection algorithms, but also propose a new flexible detection algorithm [20]. It is shown in Fig. 3.1 that the GD algorithm outperforms the ID algorithm at high SNR environment. On the other hand, the GD algorithm has weaker performance than the ID does at low SNR environment. We are motivated to take advantages of both algorithms in the following way to attain the low complexity and take into account of the satisfactory performance. Note that each GD and ID algorithm has higher computational complexity than the new one detection algorithm. The proposed GPGI framework can be performed by six steps as shown in Fig. 3.2. We partition all symbols into two groups referred to as group-I and group-II symbols by the GIS scheme and then apply iterative detection to the two group-symbols. In order to further improve performance, we generate more candidates to look for better solution.

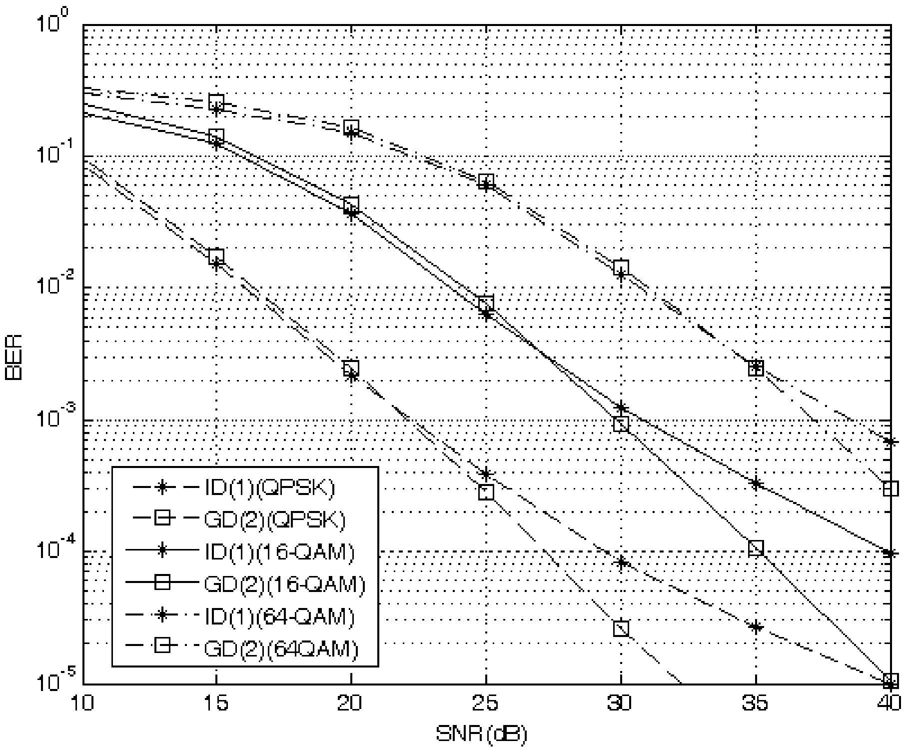

Fig. 3.1: BER performance comparison with GD and ID algorithms in (8,8) MIMO system.

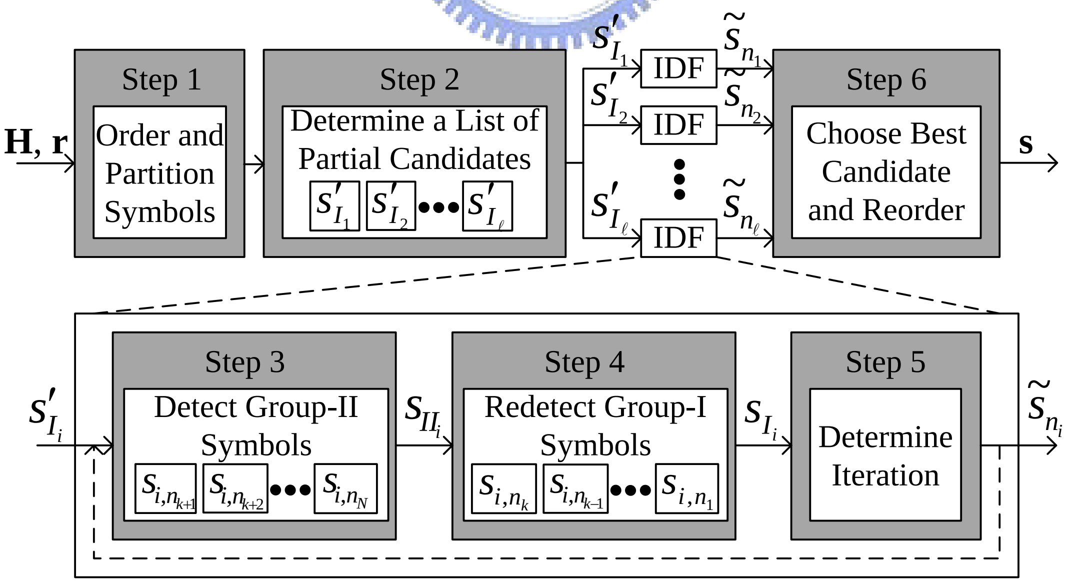

Fig. 3.2: Block diagram of the GPGI framework.

3.1 Steps of GPGI Framework

Each step is illustrated in the following.

Step 1: Order and partition all symbols into two groups. Group I has K symbols $\{s_{n_1}, s_{n_2}, \cdots, s_{n_k}\}$ with the highest order, and the residual (N-K) symbols $\{s_{n_{k+1}}, s_{n_{k+2}}, \cdots, s_{n_N}\}$ are distributed to group II.

Step 2: Determine a list of partial candidates $\{s'_{I_1}, s'_{I_2}, \cdots, s'_{I_\ell}\}$ according to the MED criterion for the group-I symbols, where $\mathbf{s}'_{I_i} = [s'_{i,n_1}\ s'_{i,n_2}\ \cdots s'_{i,n_k}]^T$, where $\mathbf{x}^T$ denotes the transpose of $\mathbf{x}$.

Step 3: Cancel the interference of $\mathbf{r}$ from the K symbols for each $\mathbf{s}'_{I_i}$ to derive $\mathbf{r}'_i$, and detect the remaining (N-K) symbols $\mathbf{s}_{II_i} = [s_{i,n_{k+1}}\ s_{i,n_{k+2}}\ \cdots s_{i,n_N}]^T$.

Step 4: Cancel the interference of $\mathbf{r}$ from the (N-K) symbols for each $\mathbf{s}_{II_i}$ to derive $\mathbf{r}''_i$, and redetect the K symbols $\mathbf{s}_{I_i} = [s_{i,n_k}\ s_{i,n_{k-1}}\ \cdots s_{i,n_1}]^T$.

Step 5: Determine whether the iterative operation is activated by detection algorithm. If iteration is triggered, the GPGI framework will update the parameter values. When there is no iteration, we combine $\mathbf{s}_{I_i}$ and $\mathbf{s}_{II_i}$ into the i-th candidate $\tilde{\mathbf{s}}_{n_i}$.

Step 6: Choose the best hard decision $\tilde{\mathbf{s}}$ among the candidates $\{\tilde{\mathbf{s}}_{n_1}, \tilde{\mathbf{s}}_{n_2}, \cdots, \tilde{\mathbf{s}}_{n_\ell}\}$ by the MED criterion, and then reorder $\tilde{\mathbf{s}}$ into $\mathbf{s}$.

3.2 The Properties of GPGI Framework

We treat steps 3~5 as an iterative decision feedback (IDF) block that detects two group symbols repeatedly. The operations of steps 1~3 are regarded as the GD algorithm. We generate more candidates at step 2 and process each IDF in parallel. Due to three features of parallel, grouped and iterative, we name as the generalized parallel grouped-iterative (GPGI) detection framework. In order to configure different detection algorithms in the GPGI framework, three parameters and three sub-algorithms are defined in the following.

- K: Number of symbols in group I whose range is $1 \le K < N$.
- ℓ: List length whose value is $1 \le \ell \le |C|^K$.
- $I_{\max}$: Maximum number of iterations whose number is $I_{\max} \ge 0$.
- sa_1, sa_2, and sa_3: Sub-detection algorithms used in step 2, 3, and 4, respectively.

As shown in Table 3.1, while (K, ℓ, $I_{\max}$)= ($1 \le K < N$, 1, 0) and (sa_1, sa_2, sa_3)=(BODF, BODF, Identity), the framework can generate the BODF algorithm in [7-10]. Note that identity means that we bypass the operations at this stage and feed the symbols directly to the next step. When identity used at step 4, we assign $\{s_{i,n_1}, s_{i,n_2}, \cdots, s_{i,n_k}\} = \{s'_{i,n_1}, s'_{i,n_2}, \cdots, s'_{i,n_k}\}$. While ($K$, ℓ, $I_{\max}$)= ($1<K<N$, 1, 0) and (sa_1, sa_2, sa_3)=(ML, BODF, Identity), the framework can reduce to the GD algorithm in [12]. While (K, ℓ, $I_{\max}$)= (N-1, 1, $I_{\max} \ge 1$) and (sa_1, sa_2, sa_3)=(BODF, BODF, BODF), the framework can generate the ID algorithm in [15]. While (K, ℓ, $I_{\max}$)= (1, $1 \le \ell < |C|$, 0) and (sa_1, sa_2, sa_3)=(BODF, BODF, Identity), the framework can generate the B-Chase algorithm in [16]. While (K, ℓ, $I_{\max}$)= ($1 \le K < N$, $|C|^K$, 0) and (sa_1, sa_2, sa_3)=(ML, LD, Identity), the framework can reduce to the GPIC(K,0) algorithm in [19]. The generalized parallel interference cancellation (GPIC) algorithm [19] can be regarded as an extended type of the B-Chase algorithm which differs from the partition of the number of symbols and sa_2. Hence, this framework can cover many conventional detection algorithms. Furthermore, one new proposed algorithm listed in the last row of Table 3.1 will be illustrated in the next chapter.

Table 3.1: Cases of the GPGI framework for MIMO detection

Detector	The number of symbols in group I : K	Sub-Algorithm used in Step 2: sa_1	List length ℓ	Sub-Algorithms used in Step 3, Step 4: sa_2, sa_3	Iteration Determination in Step 5 (MaxIteNum= I_{max})
BODF [7]-[10]	$1 \leq K < N$	BODF	1	(BODF, Identity)	No (I_{max}=0)
Grouped [12]	$1 < K < N$	ML (ZF-GIS)	1	(BODF, Identity)	No (I_{max}=0)
Iterative [15]	$K=(N-1)$	BODF	1	(BODF, BODF)	Yes($1 \leq I_{max}$)*1
B-Chase [16]	K=1	BODF	$1 \leq \ell < \lvert C \rvert$	(BODF, Identity)	No (I_{max}=0)
		ML	$\ell = \lvert C \rvert$		
GPIC(K,0) [19]	$1 \leq K < N$	ML	$\lvert C \rvert^K$	(LD, Identity)	No (I_{max}=0)
GPGI-T1	$1 \leq K < N$	B-Chase (ZF-GIS)	$1 \leq \ell \leq \lvert C \rvert$	(SQRDF, SQRDF/ Identity)	Flexible*2

*1If ($s'_{i,n_1} == s_{i,n_1}$ or IteNum (I) = = I_{max}), end; else set $s'_{i,n_1} = s_{i,n_1}$ and iterate.

*2Yes($1 \leq I_{max}$) or No. If ($\{s'_{i,n_1}, s'_{i,n_2}, \cdots, s'_{i,n_k}\} == \{s_{i,n_1}, s_{i,n_2}, \cdots, s_{i,n_k}\}$ or I = = I_{max}), end;

else set $\{s'_{i,n_1}, s'_{i,n_2}, \cdots, s'_{i,n_k}\} = \{s_{i,n_1}, s_{i,n_2}, \cdots, s_{i,n_k}\}$ and iterate.

Chapter 4

New Type GPGI-Based Detection Algorithm

In this chapter, we explore the above framework by configuring three parameters as well as three sub-algorithms and then propose one new detection algorithm called GPGI-T1. After investigating the configuration parameters including K, ℓ, I_{max} and three sub-algorithms including sa_1, sa_2, sa_3, the GPGI framework can further optimize the complexity and performance. In the proposed algorithm, the ML sub-algorithm used at step 2 of the GD detection algorithm is replaced by the B-Chase sub-algorithm, where the performance of the B-Chase detection is close to that of the ML algorithm with low computational complexity. For low computational complexity and sub-algorithm regularity, we use the sorted QR decision feedback (SQRDF) algorithm [21] as sa_2 and sa_3 instead of the zero-forcing BODF sub-algorithm used in GD. Next, we give a wide range of parameters K, ℓ, and I_{max} to trade off the complexity and performance.

4.1 Implementation of GPGI-Type1

Each detailed design step implementation of the GPGI-T1 detection algorithm is summarized in Figs. 4.1, 4.2, and 4.3. Each corresponding design step is described in the following.

Step 1: At the first step, we select K symbols with higher SNR to detect first by near-optimal algorithm such that error propagation can be alleviated. We resort the columns of channel matrix by 2-norm of the column.

$$p_i = \| \mathbf{h}_{:,i} \|^2 \quad \text{for } i = 1, 2, \ldots, N. \tag{3}$$

Where $\mathbf{h}_{:,i}$ is the i-th column of $\mathbf{H}$. According to the value of each p_i, we can sort the values and obtain (4)

$$p_{n_1} \geq p_{n_2} \geq \ldots \geq p_{n_N} , \tag{4}$$

where $\{n_1, n_2, \ldots, n_N\}$ denotes the detection order index. After permuting all symbols $\mathbf{s}$, the channel matrix $\mathbf{H}$, and identity matrix I_N, we can recast the system function as follows.

$$\mathbf{r} = \tilde{\mathbf{H}}\tilde{\mathbf{s}} + \mathbf{n} . \tag{5}$$

Where $\tilde{\mathbf{H}} = \mathbf{H\Pi} = [\mathbf{h}_{n_1}\ \mathbf{h}_{n_2} \cdots \mathbf{h}_{n_N}]$ and $\tilde{\mathbf{s}} = \Pi^T \mathbf{s} = [s_{n_1}\ s_{n_2} \cdots s_{n_N}]^T$, and $\mathbf{\Pi} = [\mathbf{e}_{n_1}\ \mathbf{e}_{n_2} \cdots \mathbf{e}_{n_N}]$. According to the values of K, $\tilde{\mathbf{s}}$ can be separated to two group symbols $\mathbf{s}_I = [s_{n_1}\ s_{n_2} \cdots s_{n_k}]^T$ and $\mathbf{s}_{II} = [s_{n_{k+1}}\ s_{n_{k+2}} \cdots s_{n_N}]^T$, and simultaneously $\tilde{\mathbf{H}}$ can be considered as two sub-channels $\mathbf{H}'$ and $\mathbf{H}''$, where $\mathbf{H}' = [\mathbf{h}_{n_1}\ \mathbf{h}_{n_2} \cdots \mathbf{h}_{n_k}]$ and $\mathbf{H}'' = [\mathbf{h}_{n_{k+1}}\ \mathbf{h}_{n_{k+2}} \cdots \mathbf{h}_{n_N}]$.

Step 2: After symbol partition as shown in lines 1~5 of Fig. 4.1, we still cannot detect the corresponding symbols because they interfere with each other. In order to solve this problem and achieve lower complexity, we apply the GIS technique to channel matrix instead of the QR-decomposition. Then, we can divide original system into two lower dimensional sub-systems. In Fig. 4.2, we modify the ZF-GIS computation [13] to generate one sub-system used in Fig. 4.1 with lower complexity. Without loss of the generality, we illustrate the computation in (4,6) MIMO system, where (x,y) denotes

x=N and y=M and set K=N/2 at this step. In this case, the ordered channel matrix $\tilde{\mathbf{H}}$ can be written as

$$\tilde{\mathbf{H}} = [\mathbf{h}_{n_1}\ \mathbf{h}_{n_2}\ \mathbf{h}_{n_3}\ \mathbf{h}_{n_4}] = \left[\mathbf{H}'\ \mathbf{H}''\right]$$

$$= \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \\ h_{51} & h_{52} & h_{53} & h_{54} \\ h_{61} & h_{62} & h_{63} & h_{64} \end{bmatrix}. \tag{6}$$

In the proposed detection algorithm, we employ the matrix $\mathbf{H}_\mathrm{b}$ to obtain a left null matrix $\mathbf{Z}$ of $\mathbf{H}''$, where $\mathbf{H}_\mathrm{b}$ is an $(N-K)\times(N-K)$ square matrix on the bottom of $\mathbf{H}''$ and $\mathbf{Z}$ is an $(M-N+K)\times M$ matrix. $\mathbf{Z}$ and $\mathbf{H}_\mathrm{b}$ can be respectively expressed in (7) and (8).

$$\mathbf{Z} = \begin{bmatrix} 1 & 0 & 0 & 0 & \mathrm{x}_{1,1} & \mathrm{x}_{1,2} \\ 0 & 1 & 0 & 0 & \mathrm{x}_{2,1} & \mathrm{x}_{2,2} \\ 0 & 0 & 1 & 0 & \mathrm{x}_{3,1} & \mathrm{x}_{3,2} \\ 0 & 0 & 0 & 1 & \mathrm{x}_{4,1} & \mathrm{x}_{4,2} \end{bmatrix}, \tag{7}$$

and

$$\mathbf{H}_b = \begin{bmatrix} h_{53} & h_{54} \\ h_{63} & h_{64} \end{bmatrix}. \tag{8}$$

We define $\mathbf{x}_i$=[$x_{i,1}$ $x_{i,2}$ …$x_{i,(N\text{-}K)}$], and $\mathbf{x}_\mathrm{i}$ can be calculated via the following matrix computation.

$$\mathbf{x}_i^T = -(\mathbf{H}_b^T)^{-1}\mathbf{h}''^{\mathrm{T}}_{i,:} \quad \text{for } i = 1, 2, \ldots, (M\text{-}N\text{+}K), \tag{9}$$

where $\mathbf{h}''_{i,:}$ denotes the i-th row of $\mathbf{H}''$. In this way, we can retrieve the left null matrix $\mathbf{Z}$ and then apply the Gram-Schmidt orthogonalization [22] to $\mathbf{Z}$ to obtain a row-orthogonal matrix $\mathbf{L}$. Then, $\mathbf{L}$ is multiplied on both sides of (5) and we can derive the following equation as

$$\hat{\mathbf{r}} = \hat{\mathbf{H}}\mathbf{s}'_I + \hat{\mathbf{n}}, \tag{10}$$

where $\hat{\mathbf{n}} = \mathbf{L}\tilde{\mathbf{n}}$ and $\hat{\mathbf{H}} = \mathbf{L}\mathbf{H}'$ with dimension of $(M - N + K) \times K$. After the ZF-GIS operation, we use the B-Chase detection algorithm [16] as sa_1 to detect the sub-system in (10) for choosing better ℓ candidates, where ℓ ranges from 1 to $|C|$. Then, we can derive an ordered list of partial candidates $\{\mathbf{s}'_{I_1}, \mathbf{s}'_{I_2}, \cdots, \mathbf{s}'_{I_\ell}\}$ of the ℓ candidates by the MED criterion in this sub-system.

Steps 3, 4, and 5: For convenience of illustration, the operations at steps 3, 4 and 5 are concurrently described. We just describe the operation of the *i*-th iterative decision feedback (IDF). At step 3 and 4 of the proposed work, we apply the SQRDF algorithm as sa_2 and sa_3 to detect two sub-systems in (11) and (12).

$$\mathbf{r}'_i = \mathbf{r} - \mathbf{H}'\mathbf{s}'_{I_i} = \mathbf{H}''\mathbf{s}_{II_i} + \mathbf{n}', \tag{11}$$

$$\mathbf{r}''_i = \mathbf{r} - \mathbf{H}''\mathbf{s}_{II_i} = \mathbf{H}'\mathbf{s}_{I_i} + \mathbf{n}''. \tag{12}$$

The SQRDF algorithm can be divided into two parts: sorted QR decomposition (SQRD) and decision feedback (DF) whose pseudo code is listed in Fig. 4.3. Both parts can be computed using the algorithm in [17] with slightly modification. After the SQRD operation on $\mathbf{H}''$, we can derive $\mathbf{H}''\mathbf{\Pi}'' = \mathbf{Q}''\mathbf{R}''$, where $\mathbf{Q}''$, $\mathbf{R}''$, $\mathbf{\Pi}''$ denote the unitary matrix, upper triangular matrix with positive and real diagonal elements, and permutation matrix, respectively. Next, we can obtain the vector $\mathbf{d}''$ which contains the reciprocal of the diagonal elements of $\mathbf{R}''$. After multiplying $\mathbf{Q}''^*$ on both sides of (11), the system can be changed to

$$\mathbf{y}''_i = \mathbf{Q}''^*\mathbf{r}'_i = \mathbf{Q}''^*\mathbf{r} - \mathbf{Q}''^*\mathbf{H}'\mathbf{s}'_{I_i} = \mathbf{R}''\bar{\mathbf{s}}_{II_i} + v'', \tag{13}$$

where $\bar{\mathbf{s}}_{II_i} = \mathbf{\Pi}''^*\mathbf{s}_{II_i} = [\bar{s}_{i,n_{k+1}}\ \bar{s}_{i,n_{k+2}} \cdots \bar{s}_{i,n_N}]^T$, and where x^* denotes the conjugate transpose of x. $\bar{\mathbf{s}}_{II_i}$ obtained from the DF operation in Fig. 4.3 can be expressed in

(14).

$$\bar{s}_{i,n_b} = \mathbf{quan}\left(\left(\mathbf{y}''_{i,b-k} - \sum_{j=b-k+1}^{N-k} \mathbf{R}''_{b-k,j}\bar{s}_{i,n_{j+k}}\right) d''_{b-k,b-k}\right), \text{ for } b=N, N\text{-}1, \ldots, K\text{+}1. \quad (14)$$

Where **quan**(x) denotes the quantization function which quantizes the value x to the nearest constellation point. The symbols $\mathbf{s}_{II_i}$ can be obtained by reordering $\bar{\mathbf{s}}_{II_i}$.

Similarly, at step 4, we can obtain following equations in (15) and (16).

$$\mathbf{y}'_i = \mathbf{Q}'^{*}\mathbf{r}''_i = \mathbf{Q}'^{*}\mathbf{r} - \mathbf{Q}'^{*}\mathbf{H}''\mathbf{s}_{II_i} = \mathbf{R}'\bar{\mathbf{s}}_{I_i} + v'. \quad (15)$$

$$\bar{s}_{i,n_c} = \mathbf{quan}\left(\left(\mathbf{y}'_{i,k-c+1} - \sum_{j=k-c+2}^{k} \mathbf{R}'_{k-c+1,j}\bar{s}_{i,n_{k-j+1}}\right) d'_{k-c+1,k-c+1}\right), \text{ for } c=1, 2, \ldots, \text{K}. \quad (16)$$

Where $\mathbf{H}'\mathbf{\Pi}' = \mathbf{Q}'\mathbf{R}'$ and $\bar{\mathbf{s}}_{I_i} = \mathbf{\Pi}'^{*}\mathbf{s}_{I_i} = [\bar{s}_{i,n_k} \ \bar{s}_{i,n_{k-1}} \cdots \bar{s}_{i,n_1}]^T$. The maximum iterative number of $I_{\max}$ affects the computational complexity. The initial iterative number I is set to zero. When executing step 4 once, I is increased by one. If $\mathbf{s}'_{I_i}$ equals $\mathbf{s}_{I_i}$ or I equals $I_{\max}$, we obtain the candidate $\tilde{\mathbf{s}}_{n_i} = [\mathbf{s}_{I_i} \ \mathbf{s}_{II_i}]^T$. Otherwise, let $\mathbf{s}'_{I_i} = \mathbf{s}_{I_i}$ and repeat steps 3 and 4. Note that if $I_{\max}$=0, there is no need to deal with the sub-system of (12), and the operation of (15) and (16) can be skipped.

Step 6: At the last step, we choose the final hard decision $\tilde{\mathbf{s}}$ according to the MED criterion among the candidates $\{\tilde{\mathbf{s}}_{n_1}, \tilde{\mathbf{s}}_{n_2}, \cdots, \tilde{\mathbf{s}}_{n_\ell}\}$. The MED of the i-th candidate is obtained by $\varepsilon_i = \|\mathbf{r} - \tilde{\mathbf{H}}\tilde{\mathbf{s}}_i\|^2$. According to the permutation matrix Π at step 1, we rank the detected symbols $\tilde{\mathbf{s}}$ to obtain the final symbols **s**.

In terms of algorithm flexibility, when K=1, the GPGI-T1 algorithm can performs as the combination of the Chase and ID algorithms. When ℓ=1, the GPGI-T1 algorithm reduces to the combination of the GD and ID algorithms. When $I_{\max}$=0, the GPGI-T1 algorithm reduces to the combination of the Chase and GD algorithms.

FUNCTION: **GPGI-T1 Detection Algorithm**
INPUT: ($\mathbf{H}$, $\mathbf{r}$, M, N, C, K, ℓ, I_{max}) OUTPUT: ($\mathbf{s}$)

---

for i = 1 to N, $p_i = \|\mathbf{h}_{:,i}\|^2$ end
$\mathbf{\Pi}$ = $N \times N$ permutation matrix that sorted by p_i from $\mathbf{I}_N$
$\tilde{\mathbf{H}} = \mathbf{H\Pi}$
$\mathbf{H}'$ = first K columns of $\tilde{\mathbf{H}}$
$\mathbf{H}''$ = last (N-K) columns of $\tilde{\mathbf{H}}$
$[\hat{\mathbf{H}}, \hat{\mathbf{r}}]$ = ZF-GIS($\mathbf{H}', \mathbf{H}'', \mathbf{r}, K$)
$[\mathbf{s}'_{I_1}, \mathbf{s}'_{I_2}, \cdots, \mathbf{s}'_{I_\ell}]$ = B-Chase($\hat{\mathbf{H}}, \hat{\mathbf{r}}, C, \ell$)
$[\mathbf{Q}'', \mathbf{R}'', \mathbf{d}'', \mathbf{\Pi}'']$ = SQRD($\mathbf{H}'', \mathbf{p}, \mathbf{\Pi}$)
$\mathbf{u}'' = \mathbf{Q}'' * \mathbf{r}$
$\mathbf{v}'' = \mathbf{Q}'' * \mathbf{H}'$
$[\mathbf{Q}', \mathbf{R}', \mathbf{d}', \mathbf{\Pi}']$ = SQRD($\mathbf{H}', \mathbf{p}, \mathbf{\Pi}$)
$\mathbf{u}' = \mathbf{Q}' * \mathbf{r}$
$\mathbf{v}' = \mathbf{Q}' * \mathbf{H}''$
$E_{min} = \infty$
for i = 1 to ℓ,
  $I = 0$
  while ($I < I_{max}$) & ($\mathbf{s}'_{I_i} \neq \mathbf{s}_{I_i}$),
    if $I \neq 0$, $\mathbf{s}'_{I_i} = \mathbf{s}_{I_i}$ end
    $\mathbf{y}'' = \mathbf{u}'' - \mathbf{v}''\mathbf{s}'_{I_i}$
    $\mathbf{s}_{II_i}$ = DF($\mathbf{R}'', \mathbf{d}'', \mathbf{y}'', \mathbf{\Pi}''$, N-K)
    $\mathbf{y}' = \mathbf{u}' - \mathbf{v}'\mathbf{s}_{II_i}$
    $\mathbf{s}_{I_i}$ = DF($\mathbf{R}', \mathbf{d}', \mathbf{y}', \mathbf{\Pi}'$, K)
    $I = I + 1$
  End
  $\mathbf{e}_i = \mathbf{r} - \mathbf{H}'\mathbf{s}_{I_i} - \mathbf{H}''\mathbf{s}_{II_i}$
  $\varepsilon_i = 0$
  for j = 1 to M,
    if $\varepsilon_i < E_{min}$, $\varepsilon_i = \varepsilon_i + |e'_{i,j}|^2$ end
  end
  if $\varepsilon_i < E_{min}$, $E_{min} = \varepsilon_i$
    $\tilde{\mathbf{s}} = [\mathbf{s}_{I_i}\ \mathbf{s}_{II_i}]^T$
  end
end
$\mathbf{s} = \mathbf{\Pi}\tilde{\mathbf{s}}$

Fig. 4.1: Processing pseudo code for the implementation of the GPGI-T1 detection algorithm for $I_{max} \geqq 1$.

FUNCTION: **ZF-GIS**
INPUT: ($\mathbf{H}', \mathbf{H}''$, $\mathbf{r}$, K) OUTPUT: ($\hat{\mathbf{H}}, \hat{\mathbf{r}}$)

1. $\mathbf{H}_b$ = (N-K)×(N-K) matrix at the bottom of $\mathbf{H}''$
2. $\mathbf{W} = (\mathbf{H}_b^T)^{-1}$
3. for i = 1 to (M-N+K), $\mathbf{x}_i^T = -\mathbf{W}\mathbf{h}''^T_{i,:}$ end
4. $\mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \ldots \quad \mathbf{x}_{M\text{-}N+K}]^T$
5. $\mathbf{Z} = [\mathbf{I}_{M\text{-}N+K} \;\vdots\; \mathbf{X}]$
6. $\mathbf{L}$ = GSO($\mathbf{Z}$)
7. $\hat{\mathbf{H}} = \mathbf{L}\mathbf{H}'$
8. $\hat{\mathbf{r}} = \mathbf{L}\mathbf{r}$

Fig. 4.2: Processing pseudo code for the proposed modified GIS implementation.

FUNCTION: **DF**
INPUT: ($\mathbf{R}, \mathbf{y}, \mathbf{d}, \mathbf{\Pi}, J$) OUTPUT: ($\mathbf{s}$)

1. for i = 1 to J
2. $\quad t = \sum_{j=1}^{i-1} r_{i,j}\bar{s}_j$
3. $\quad \bar{s}_i = \mathbf{quan}((y_i - t)d_i)$
4. end
5. $\mathbf{s} = \mathbf{\Pi}\bar{\mathbf{s}}$

Fig. 4.3: Processing pseudo code for the proposed DF implementation that modified from [17].

4.2 Reducing Complexity Highlight

There are two schemes to lower the computational complexity. First, we reduce complexity by reusing tentative computations.

- Observing (13) and (15), we can reuse tentative calculations for parallel and iterative computing such that we just compute the SQRD function on $\mathbf{H}'$ and $\mathbf{H}''$, $\mathbf{Q}''^*\mathbf{r}$, $\mathbf{Q}''^*\mathbf{H}'$, $\mathbf{Q}'^*\mathbf{r}$, and $\mathbf{Q}'^*\mathbf{H}''$ once.
- Observing (3), the 2-norm of each column of **H** can be reused in the computations of the SQRD function.

Second, we reduce complexity by avoiding unnecessary computations.

- When the input symbols are the same as that at the last iteration, we can skip the calculations in the following iterations. That mean we do not need to reach the maximum iteration number $I_{\max}$ in each IDF.
- We use pruning and threshold-tightening strategy given in [16] to generate a threshold $E_{\min}$ which records the last time MED values of other candidates. When the Euclidean distance is greater than $E_{\min}$ during the process, the computation can be terminated.

Using the above two schemes, we can alleviate the computational complexity, where the complexity analysis of the GPGI-T1 algorithm will be debated in detail in the following chapter.

Chapter 5

Complexity Analysis, Simulation Results, and Implementation

This chapter demonstrates the complexity and performance of the GPGI-T1 detection algorithm and shows the comparison results with the existing detection schemes including the BODF, GD, ID, B-Chase algorithms and GPIC(K,0) detection algorithm. We use the GPGI-T1(K, ℓ, $I_{\max}$) to denote the GPGI-T1 algorithm with K symbols distributed to group I, list length ℓ and maximum iteration $I_{\max}$. Moreover, the B-Chase(ℓ) denotes the ZF B-Chase algorithm with list length ℓ, and the GPIC(K,E) denotes the GPIC algorithm with K symbols in group I and E error symbols in group II.

5.1 Complexity Analysis

In Table 5.1, we summarize the number of complex multiplications, complex divisions and square roots required by the GPGI-T1 algorithm. The GPGI-T1 algorithm includes the order and partition symbols (OPS), GIS, B-Chase used in sub-system, precomputation1 (PC1), precomputation2 (PC2) and the combination of DF and MED (DF&MED) of the design steps. PC1(1) and PC1(2) correspond to the operations of lines 8 and 9-10 of Fig. 4.1, respectively. Similarly, PC2(1) and PC2(2) represent the operations of lines 11 and 12-13 of Fig. 4.1, respectively. Although the computations of division and square root are more complex than those of multiplications, the number of divisions and square roots is much less than the number of multiplications in the GPGI-T1 and others algorithms. Therefore, the complexity is measured by the sum of complex multiplications, divisions and square roots in the worst case. The multiplication of a number and a constellation point can be implemented by scaled integers [23] such that we can reduce the number of multiplications. For simplicity, we assume that the number of transmitters is an even integer and K=N/2 in the GPGI-T1 algorithm, and the channel matrix changes during every symbol period. That means we process all computations at each symbol period. The comparisons of the worst-case computational complexity of the GPGI-T1 algorithm, B-Chase scheme, and GPIC(1,0) algorithm are tabulated in Table 5.2. When M=N, the complexity order of the GPGI-T1, B-Chase and GPIC algorithms are $O(47/24N^3)$, $O(11/3N^3)$ and $O(4N^3)$ respectively. Herein, we do not formulate the complexity of the GD and ID algorithms since both algorithms require more computational complexity than the B-Chase detection, where the complexity of the GD algorithm is exponential time of the number of first-group symbols and the complexity of the ID algorithm almost doubles that of the BODF algorithm (B-Chase(1)) mentioned in [15]. It is emphasized again that since the SD detector shows larger computational complexity as addressed in [16], for example, at BER=10^{-3}, the SD and B-Chase algorithms respectively own the complexity of 57 RM/b and 18 RM/b, we only consider the Chase detection algorithm for complexity comparison instead of the SD algorithm.

Table 5.1: Computational complexity of the GPGI-T1 algorithm

	Belong to	Multiplications	Divisions	Square root
OPS	Step 1	MN		
GIS	Step 2	$1/6M^3+1/2M^2N-1/2MN^2+MNK-1/2MK^2+1/6N^3-N^2K+3/2NK^2-2/3K^3+3/2M^2-MN+3/2MK-N^2+3/2NK-1/2K^2+1/3M-1/6N+1/6K$	$1/2N^2-NK+1/2K^2+M-3/2N+3/2K$	$M-N+K$
B-Chase ($\ell=\lvert C\rvert$)	Step 2	$3MK^2-3NK^2+11/3K^3+2MK-2NK+5K^2+1/3K+2K\lvert C\rvert$	$1/2K^2+3/2K$	$2K$
PC1 (1)	Step 3	$MN^2-2MNK+MK^2+1/2N^2-NK+1/2K^2-1/2N+1/2K$	$N-K$	$N-K$
(2)		$MNK-MK^2+MN-MK$	0	0
PC2* (1)	Step 4	$MK^2+1/2K^2-1/2K$	K	K
(2)		$MNK-MK^2+MK$	0	0
DF&MED*	Step 3, 4, 6	$(NI_{max}+M)\ell$	0	0
Total GPGI-T1	All steps	$1/6M^3+1/2M^2N+1/2MN^2+MNK+5/2MK^2+1/6N^3-N^2K-3/2NK^2+3K^3+3/2M^2+MN+7/2MK-1/2N^2-3/2NK+11/2K^2+1/3M-2/3N+1/2K+2K\lvert C\rvert+(NI_{max}+M)\ell$	$1/2N^2-NK+K^2+M-1/2N+3K$	$M+3K$

*When I_{max} equals zero, the computational complexity of **PC2** is equal to zero and the multiplication complexity of **DF&MED is** changed to $(M+N-K)\ell$.

Table 5.2: Complexity comparison among the proposed GPGI-T1 and conventional algorithms

Algorithm	Multiplications/Divisions/Square roots
B-Chase	$3MN^2+2/3N^3+2MN+7/2N^2+23/6N+2N\ell$ [*1]
GPIC(1,0)	$4MN^2-4MN+N^2+3/2M-2N+1+MN\lvert C\rvert$
GPGI-T1	$1/6M^3+1/2M^2N+13/8MN^2-1/3N^3+3/2M^2+11/4MN+3/8N^2+7/3M+25/12N+N\lvert C\rvert+(NI_{max}+M)\ell$ [*2]

[*1] When $1<\ell<|C|$, the additional computation complexity of $1/6N^3+3/2N^2+4/3N$ is needed.

[*2] In this case, N is an even integer and $K=N/2$.

5.2 Simulation Results

On the other hand, we show simulation results to sustain the performance of the GPGI-T1 detection algorithm. The simulation environment is assumed Rayleigh flat-fading channel and no correlation between sub-channels. The performance measurement targets at the SNR that reaches BER=10^{-3}. Fig. 5.1 shows the performance of the GPGI-T1 algorithm with different K and ℓ in (8,8) system with 16-QAM inputs. We can find that the performance with larger K is better than that with smaller K under the same ℓ. In this case, the complexity of the GPGI-T1 algorithm with K=6 approximately doubles with K=2 under the same ℓ, and the range of performance of the GPGI-T1 algorithm with K=6 is narrow. In order to trade off the complexity and performance, we prefer to choose K in the range from 2 to N/2. Fig. 5.2 shows the performance of the GPGI-T1 algorithm with different $I_{\max}$ and ℓ in (8,8) system with QPSK inputs. The performance of the GPGI-T1 algorithm can be improved by increasing $I_{\max}$ under the same ℓ. We set K=N/2 and suitable value of $I_{\max}$ in the GPGI-T1 algorithm to compare with the existing detection algorithms. Figs. 5.3-5.10 show the performance in (4,4), (6,6), (8,8), and (4,6) systems. Figs. 5.3, 5.5, 5.7, and 5.9 use the constellation of QPSK, and Figs. 5.4, 5.6, 5.8, and 5.10 use the constellation of 16-QAM.

From the comparison results, we can find out the BER performance of the GPGI-T1 algorithm can be significantly enhanced by slightly increasing the list length ℓ. For example, GPGI-T1(2,2,1) outperforms GPGI-T1(2,1,1) by 3.3 dB and 3 dB with respect to QPSK and 16-QAM inputs in (4,4) system, and just increases complexity 3.3% and 2.8%. The better performance can be obtained with longer list length; GPGI-T1(2,16,1) outperforms GPGI-T1(2,1,1) by 5 dB with 16-QAM inputs. On the other hand, better BER performance can be obtained by increasing $I_{\max}$ under the same list length. For example, in (8,8) system with QPSK inputs, GPGI-T1(4,4,3) outperforms GPGI-T1(4,4,0) and GPGI-T1(4,4,1) by 1.3 dB and 0.3 dB, respectively. In summary, the computational complexity and BER performance of the GPGI-T1 algorithm depends on these parameters given above. The smaller K, ℓ, $I_{\max}$, and simplified sub-algorithm achieve low complexity. Otherwise, the higher K, ℓ, $I_{\max}$, and better sub-algorithm attain better performance.

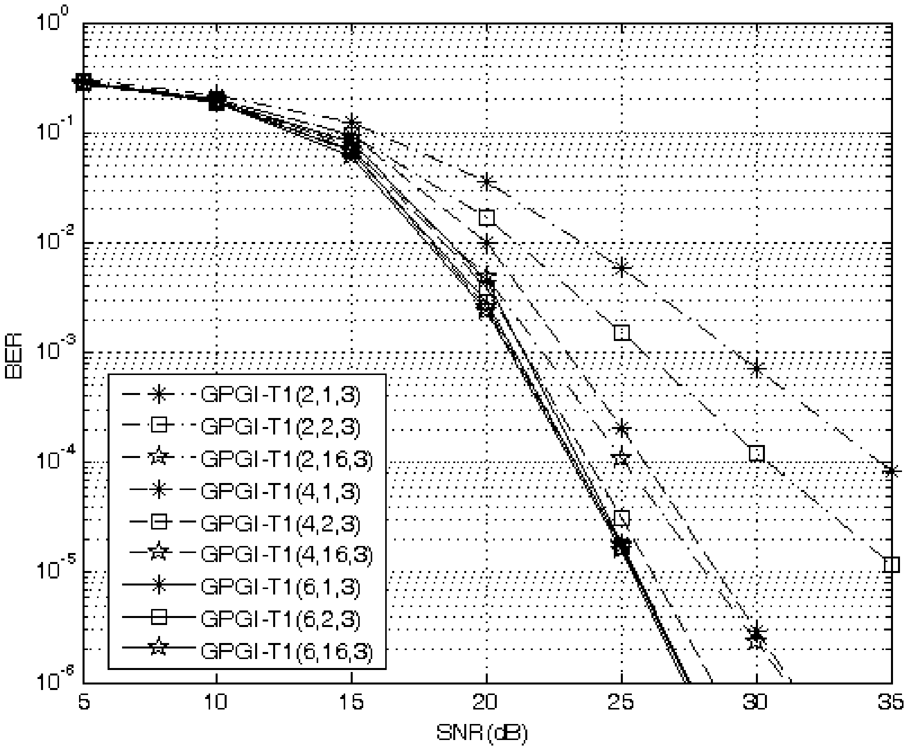

Fig. 5.1: BER performance of the GPGI-T1 algorithm with different K and ℓ in (8,8) MIMO system with 16-QAM inputs.

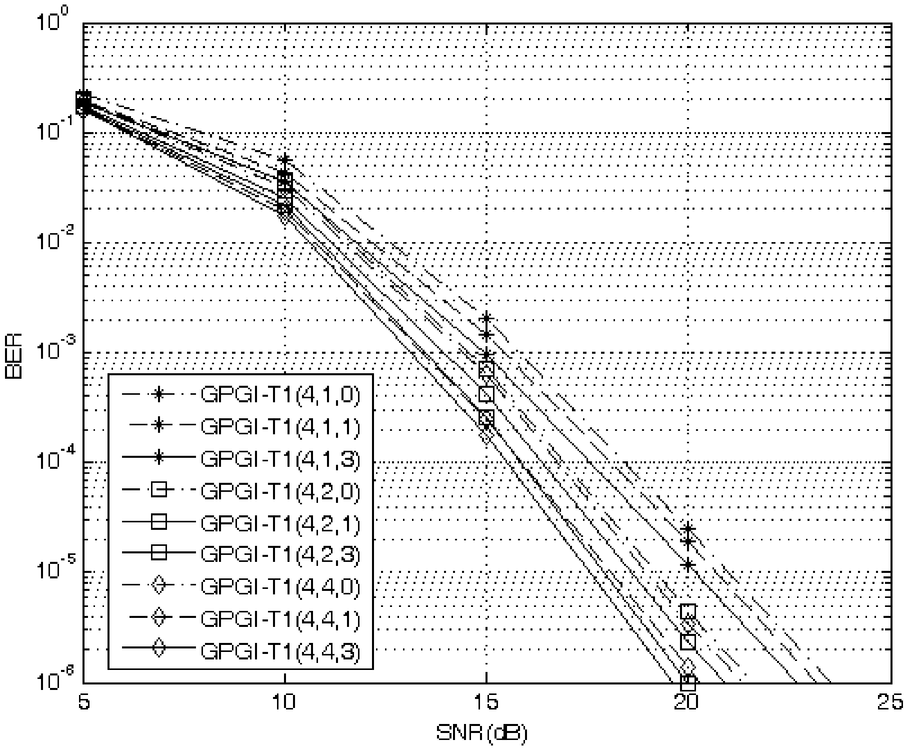

Fig. 5.2: BER performance of the GPGI-T1 algorithm with different I_{max} and ℓ in (8,8) MIMO system with QPSK inputs.

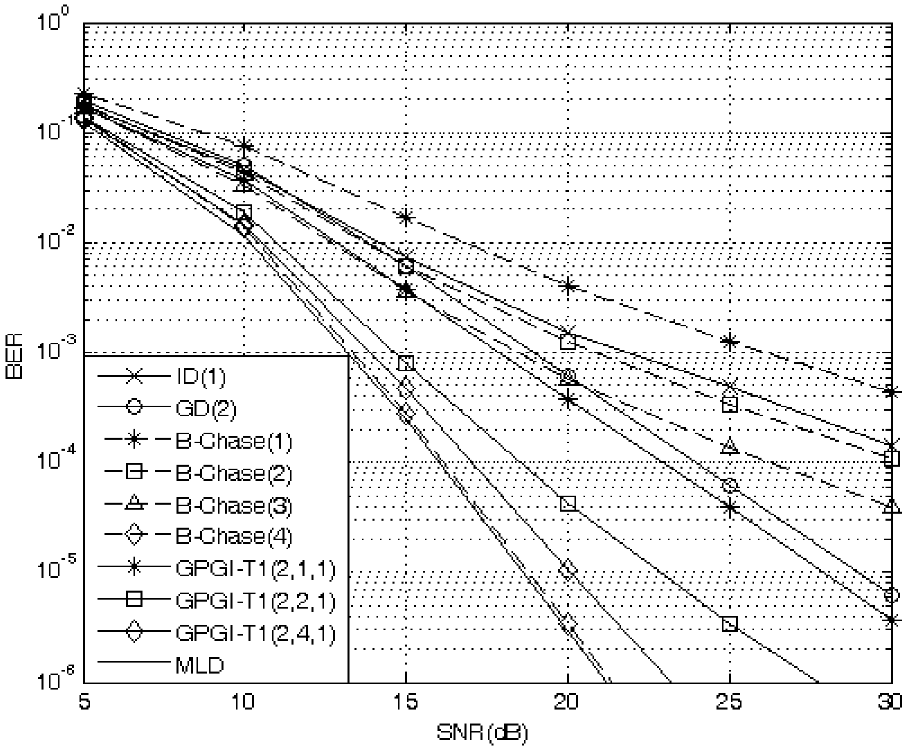

Fig. 5.3: BER performance of the GPGI-T1 algorithm and conventional algorithms in (4,4) MIMO system with QPSK inputs.

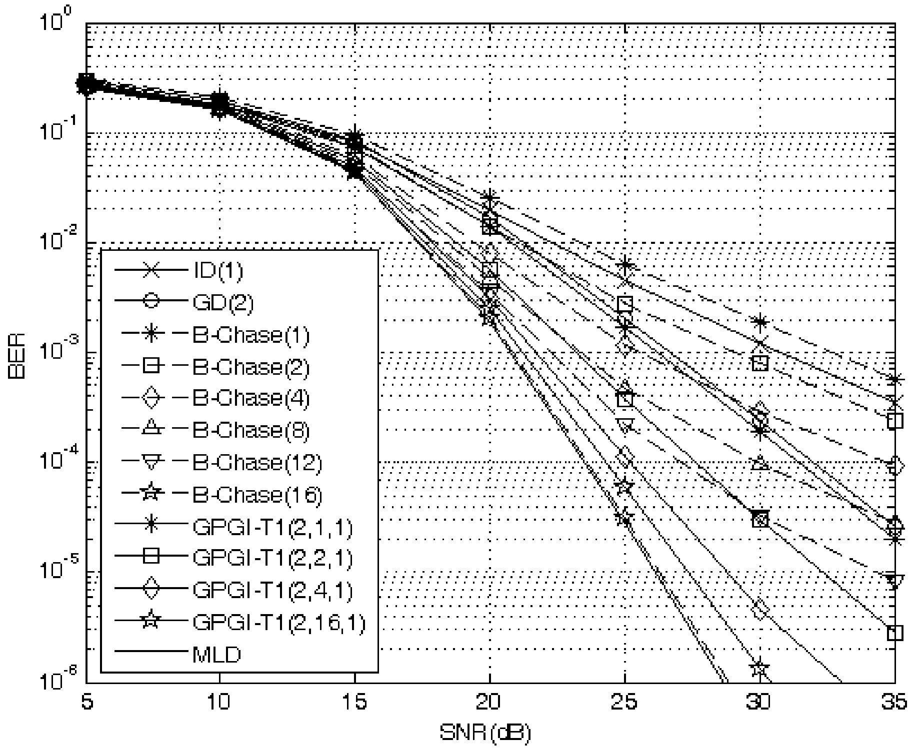

Fig. 5.4: BER performance of the GPGI-T1 algorithm and conventional algorithms in (4,4) MIMO system with 16-QAM inputs.

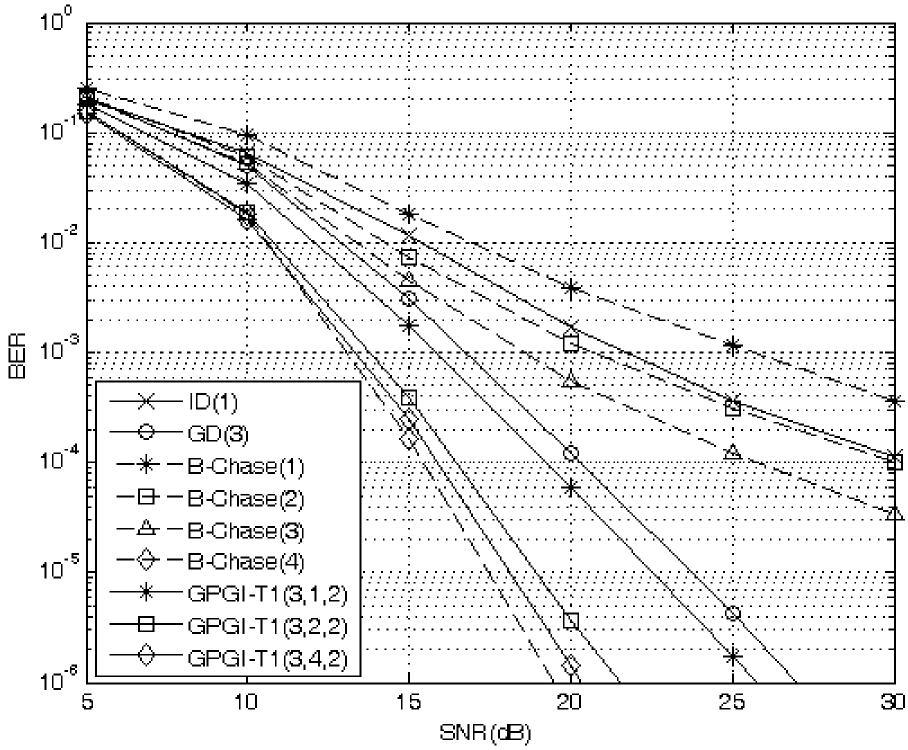

Fig. 5.5: BER performance of the GPGI-T1 algorithm and conventional algorithms in (6,6) MIMO system with QPSK inputs.

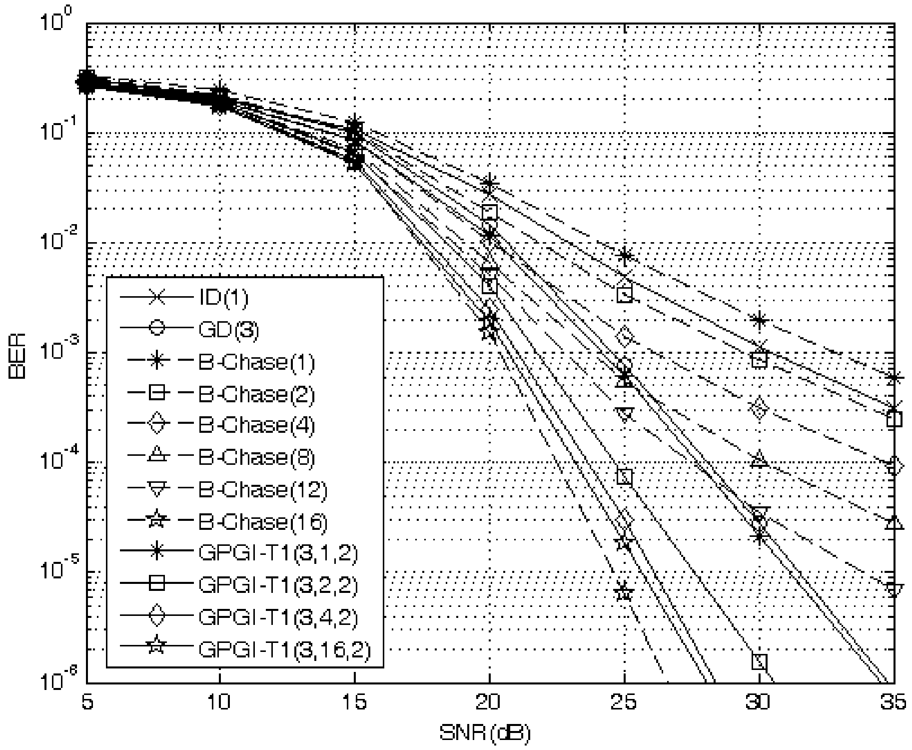

Fig. 5.6: BER performance of the GPGI-T1 algorithm and conventional algorithms in (6,6) MIMO system with 16-QAM inputs.

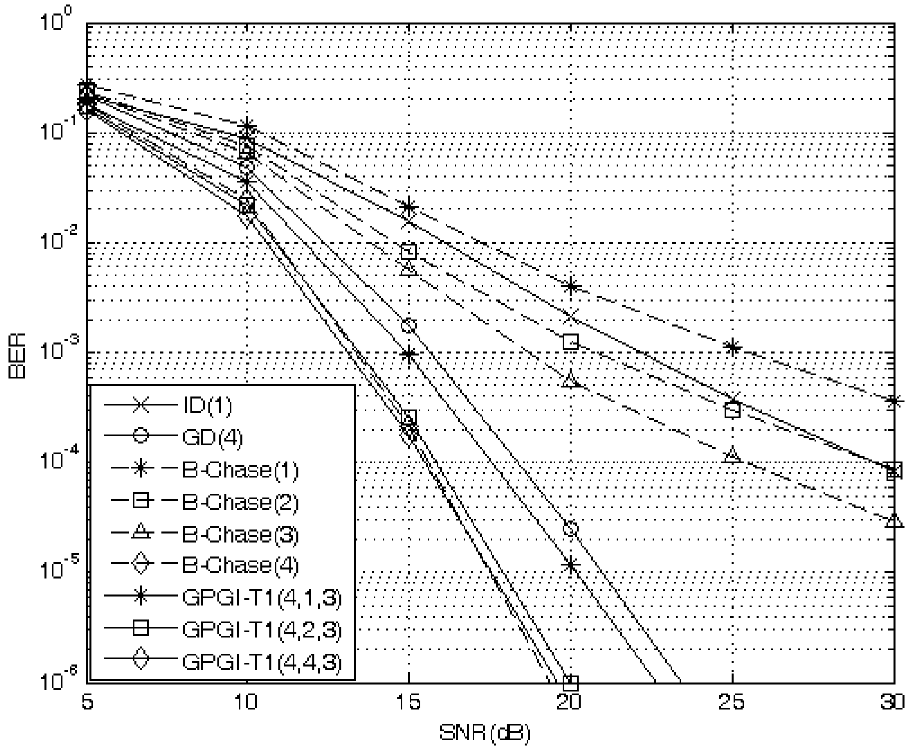

Fig. 5.7: BER performance of the GPGI-T1 algorithm and conventional algorithms in (8,8) MIMO system with QPSK inputs.

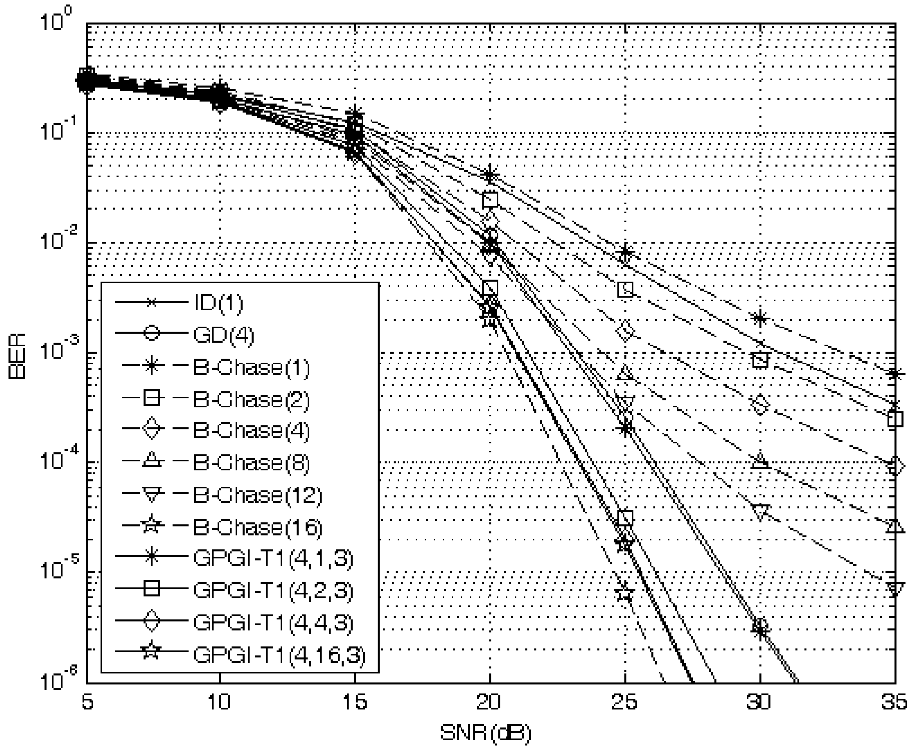

Fig. 5.8: BER performance of the GPGI-T1 algorithm and conventional algorithms in (8,8) MIMO system with 16-QAM inputs.

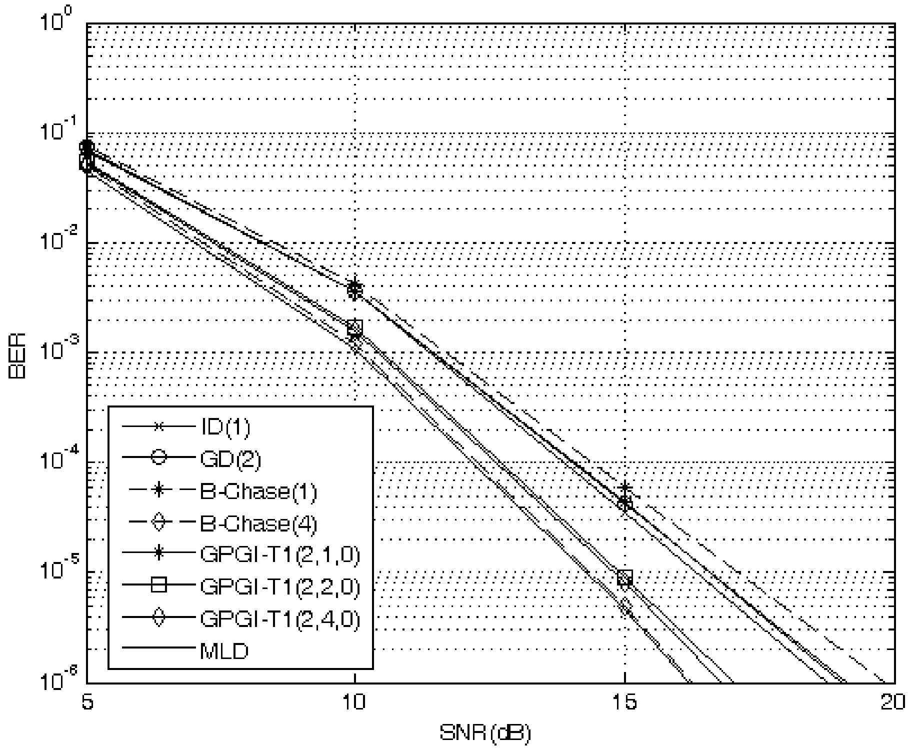

Fig. 5.9: BER performance of the GPGI-T1 algorithm and conventional algorithms in (4,6) MIMO system with QPSK inputs.

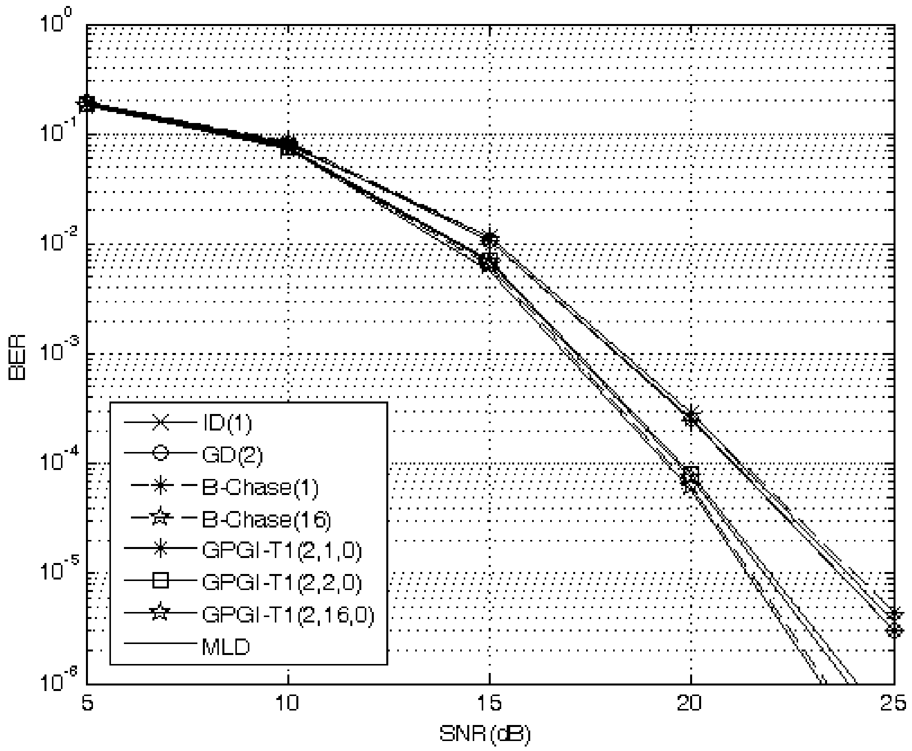

Fig. 5.10: BER performance of the GPGI-T1 algorithm and conventional algorithms in (4,6) MIMO system with 16-QAM inputs.

5.3 Complexity-Performance Tradeoff

Next, we show the complexity-performance tradeoff of the GPGI-T1, B-Chase, and GPIC(1,0) algorithms in Figs. 5.11 and 5.12. In (8,8) system, GPGI-T1(4,1,3) not only reduces the complexity of 38.1% and 33.9% but also gains 9.5 dB and 10 dB compared with the BODF algorithm (B-Chase(1)) with respect to QPSK and 16-QAM inputs, respectively. GPGI-T1(4,16,3), GPGI-T1(4,4,3), and GPGI-T1(4,2,3) reduce the complexity of 21.5%, 36.8%, and 39.3% while falling 0.3 dB, 0.4 dB, and 0.8 dB short of the B-Chase(16) algorithm with 16-QAM inputs respectively. In other configurations with M=N, the comparison of complexity and performance has behavior similar to that of the above analysis trend. On the other hand, in (4,6) system, the GPGI-T1 algorithm has comparable performance and less computational complexity. We do not present the comparison with GD and ID here since both algorithms require more computational complexity compared with the corresponding cases of the GPGI-T1 algorithm, and have poor BER performance. Therefore, from the complexity and performance analysis, the GPGI-T1 algorithm attains better complexity-performance tradeoff at the slight penalty of BER performance degradation compared with the B-Chase and GPIC(1,0) detection algorithms. Moreover, the GPGI-T1 algorithm has the lowest complexity in all cases under M=N and provides adjustable performance for different user's requirements.

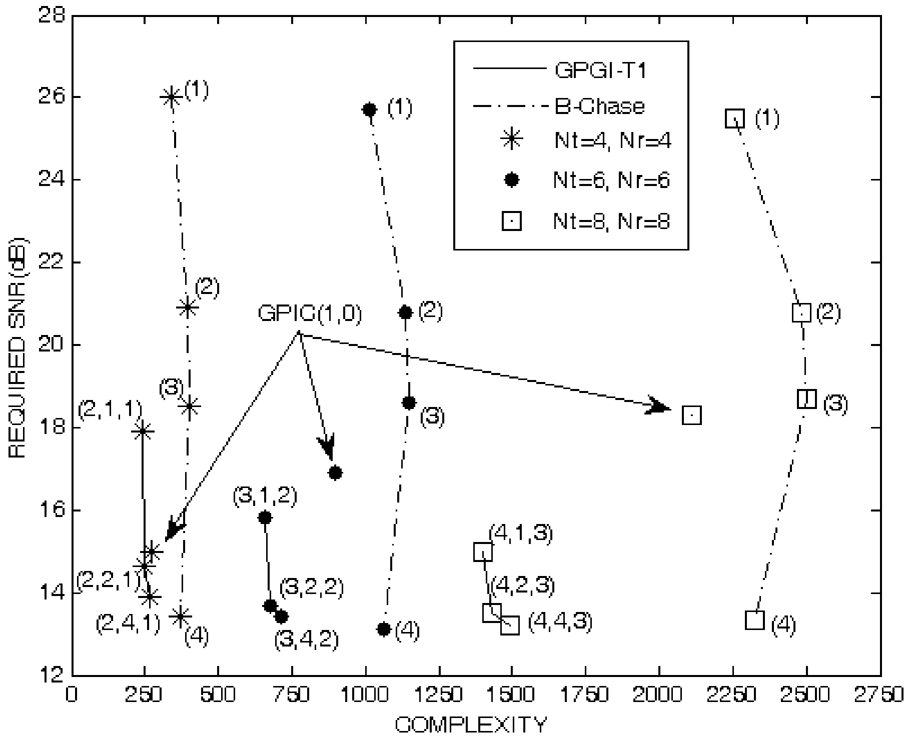

Fig. 5.11: Complexity-performance trade-off of the GPGI-T1, B-Chase and GPIC(1,0) algorithms with QPSK inputs.

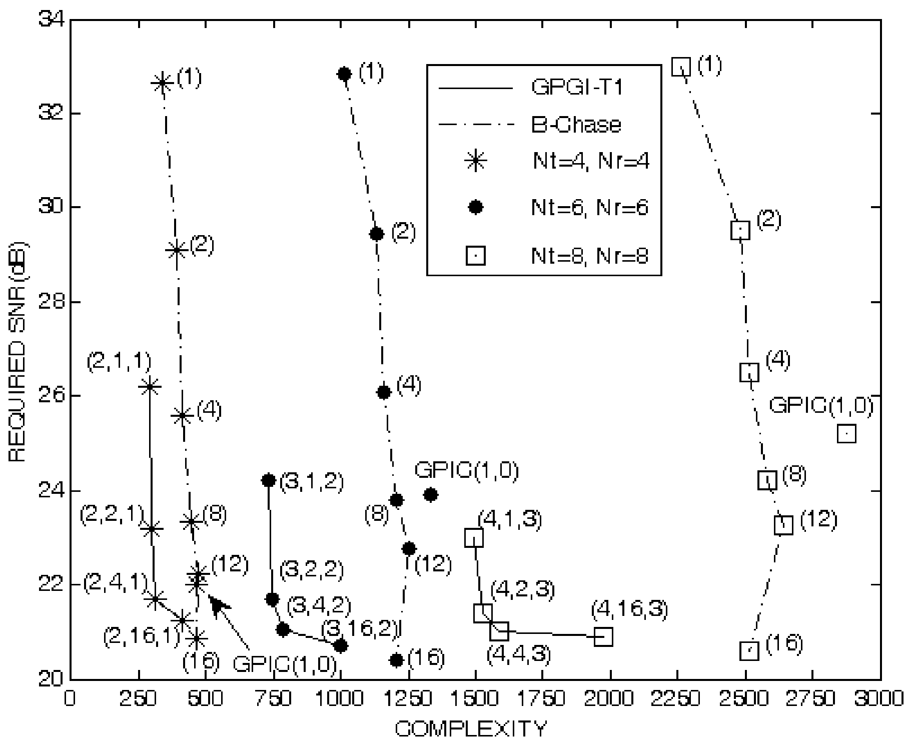

Fig. 5.12: Complexity-performance trade-off of the GPGI-T1, B-Chase and GPIC(1,0) algorithms with 16-QAM inputs.

5.4 VLSI Implementation

In this section, we begin to show that how implement a multi-mode MIMO detector using the proposed GPGI-T1 algorithm. We replace the block diagram of the GPGI framework in Fig. 3.2 to that of our proposed GPGI-T1 algorithm in Fig. 5.13.

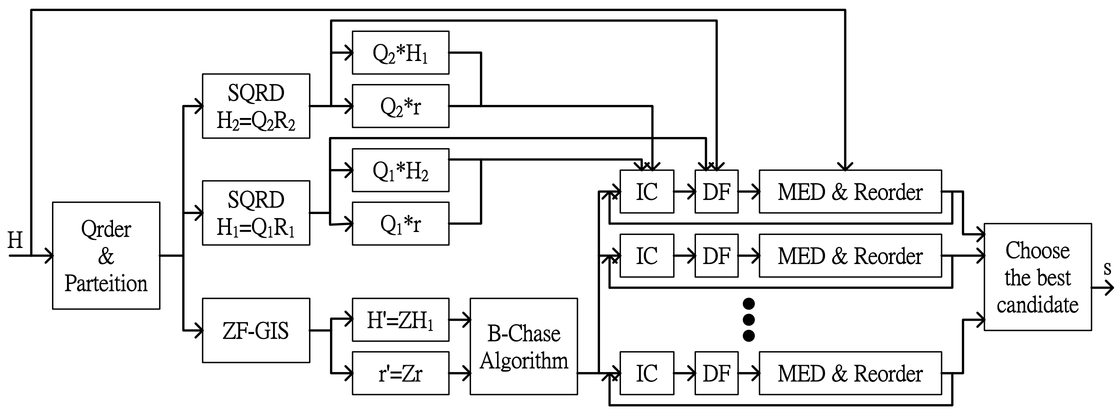

Fig. 5.13: Block diagram of the GPGI-T1 algorithm.

Besides, the channel matrix **H** is the same for each frame. It means that we just compute the variable that only related to **H** once each frame. So, we divide the detection flow of the GPGI-T1 algorithm into two parts, preprocessing part and decision part. The preprocessing part computes just once when the channel matrix **H** is unchanged, and the decision part operates for each symbol period. In this thesis, we just implement the decision part, where the maximum iteration I_{max} is equal to zero. The block diagram of the GPGI-T1 implementation is shown in Fig. 5.14.

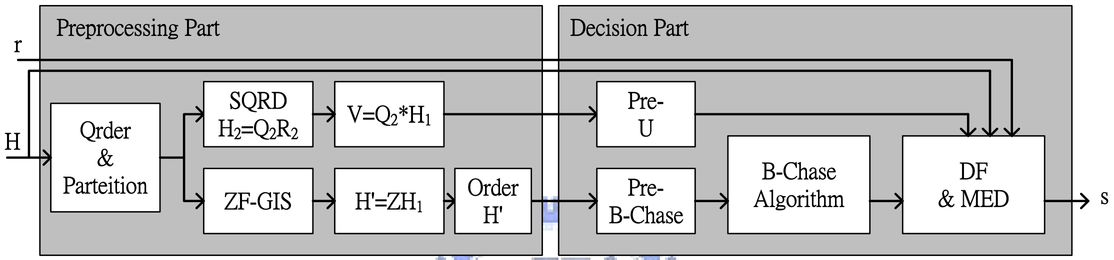

Fig. 5.14: Block diagram of the implementation of the GPGI-T1 algorithm.

Moreover, we would like to design a multi-mode GPGI-T1 detector which can work in many practical conditions including (2,2) and (4,4) MIMO system with QPSK, 16-QAM, and 64-QAM inputs. We design the MIMO detector with power-aware feature in (4,4) MIMO system. Before designing hardware architecture, we simulate the BER performance of the floating-point GPGI-T1 algorithm and the modified fixed-point GPGI-T1 algorithm in (4,4) MIMO system with 64-QAM inputs, the critical mode in our implementation, as shown in Fig. 5.15. The modified GPGI-T1 algorithm changes the MED function from 2-norm to 1-norm. We can find that GPGI-T1(2,8,0) is a setting candidate for the trade-off of the BER performance and computational complexity. Therefore, we set that the maximal list length equals eight in the multi-mode GPGI-T1 detector, and the maximal input word length equals ten bits. Table 5.3 illustrates how to attain multi-mode BER performance by adjusting the parameter ℓ.

The pipeline architecture of the multi-mode GPGI-T1 detector is depicted in Fig. 5.16. The two-group input buffers are used to store inputs and process data simultaneously. The Pre-U and Pre-B-Chase parts implemented by multiply Accumulate (MAC) unit process the reused variables for B-Chase and DF&MED parts. The B-Chase part is divided to four stages. The former two stages are in charged of parallel search and Euclidean distance calculation. The latter two stages play the role of sorting network implemented by Bitonic sort. The DF&MED part is divided to four stages including interference cancellation (IC), decision feedback 1 (DF1), decision feedback 2 (DF2), and minimum Euclidean distance (MED). The IC stage cancels the interference from the two symbols obtained by B-Chase. The DF1 and DF2 stages decide another two symbols and calculate the temporary variable for Euclidean distance calculation. The MED stage calculates the final Euclidean distance and stores the symbols with MED after comparison.

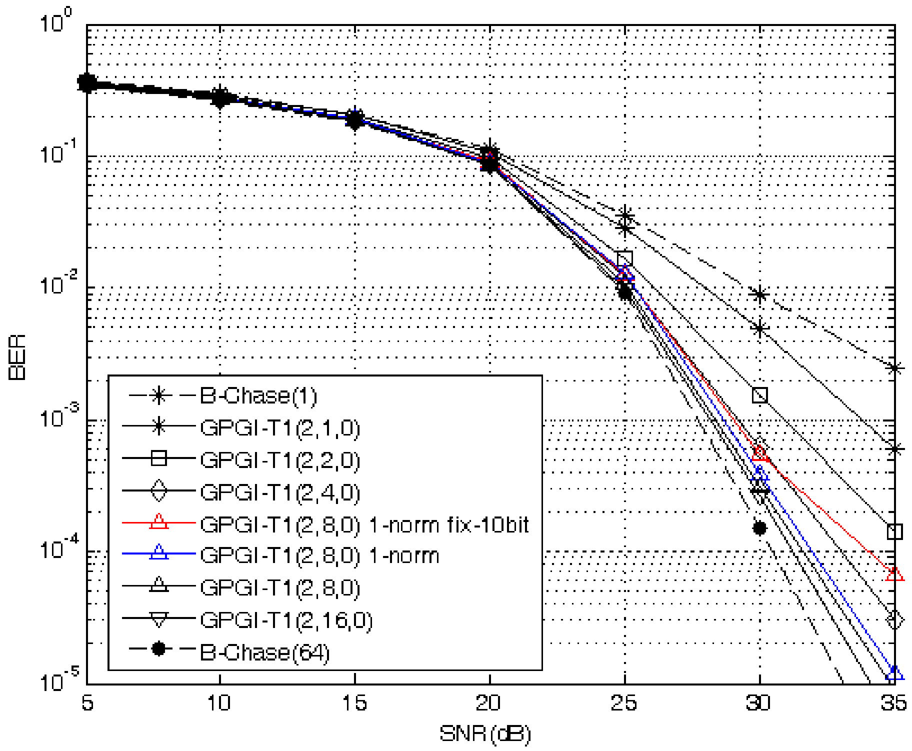

Fig. 5.15: BER performance of the GPGI-T1 algorithm and B-Chase algorithm in (4,4) MIMO system with 64-QAM inputs.

Table 5.3: Performance selection by choosing different list length

Antenna	4x4					
Modulation	QPSK		16-QAM		64-QAM	
BER Performance	Close to optimal	Close to optimal	Close to optimal	Close to optimal / Sub-optimal	Close to optimal / Sub-optimal	Close to optimal / Sub-optimal
List length ℓ	2	1	4	1	8	1

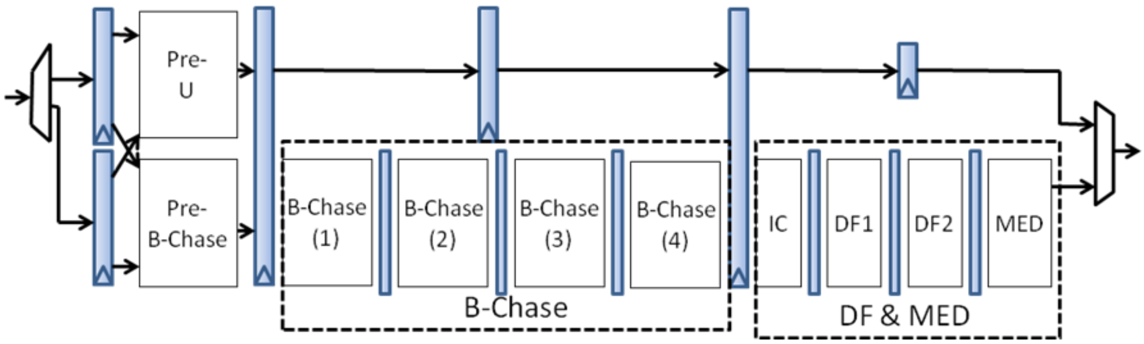

Fig. 5.16: Pipeline architecture of the multi-mode GPGI-T1 detector.

Concerning the chip implementation, the cell-based design flow with Artisan standard cell library is adopted and the multi-mode GPGI-T1 detector has been implemented in TSMC 0.18-um CMOS process. The Synopsys Design Compiler is used to synthesize the RTL design of the proposed detector and Cadence SOC Encounter is adopted for placement and routing (P&R). The Synopsys PrimePower is used to analyze the power consumption. The active chip layout area of the proposed multi-mode GPGI-T1 detector as shown in Fig. 5.17 is 1.41 mm $\times$ 1.39 mm. Table 5.4 summarizes the chip characteristics of the multi-mode GPGI-T1 detector. Table 5.5 summarizes the supplied modes and the respective power consumption of our chip design. It can work in nine modes, where three and six modes belong to (2,2) and (4,4) systems, respectively. The multi-mode functions of the GPGI-T1 detector has been proved by post-layout simulation verification as shown in Fig 5.18.

Table 5.6 provides a comprehensive comparison of the relevant ASIC implementations for MIMO detection. In [23] and [24], the BER performance of the implementation algorithms is optimal or close to optimal, respectively, but the power consumption is large. An implementation of the BODF algorithm by square root method [9] which shows low computational complexity but poor BER performance was proposed in [25]. The above chip design [25] including preprocessing part has better power efficiency than the SD implementation [23], [24], where the power efficiency is defined as the ratio of the normalized throughput to the normalized power. Our implementation of the GPGI-T1 algorithm has best power efficiency compared with other implementation designs. For example, in (4,4) MIMO system with 16QAM inputs, our design shows seven times the power efficiency of the implementation in [24]. Furthermore, our design possesses low-complexity computation and multi-mode implementation with better power efficiency compared with other reference designs.

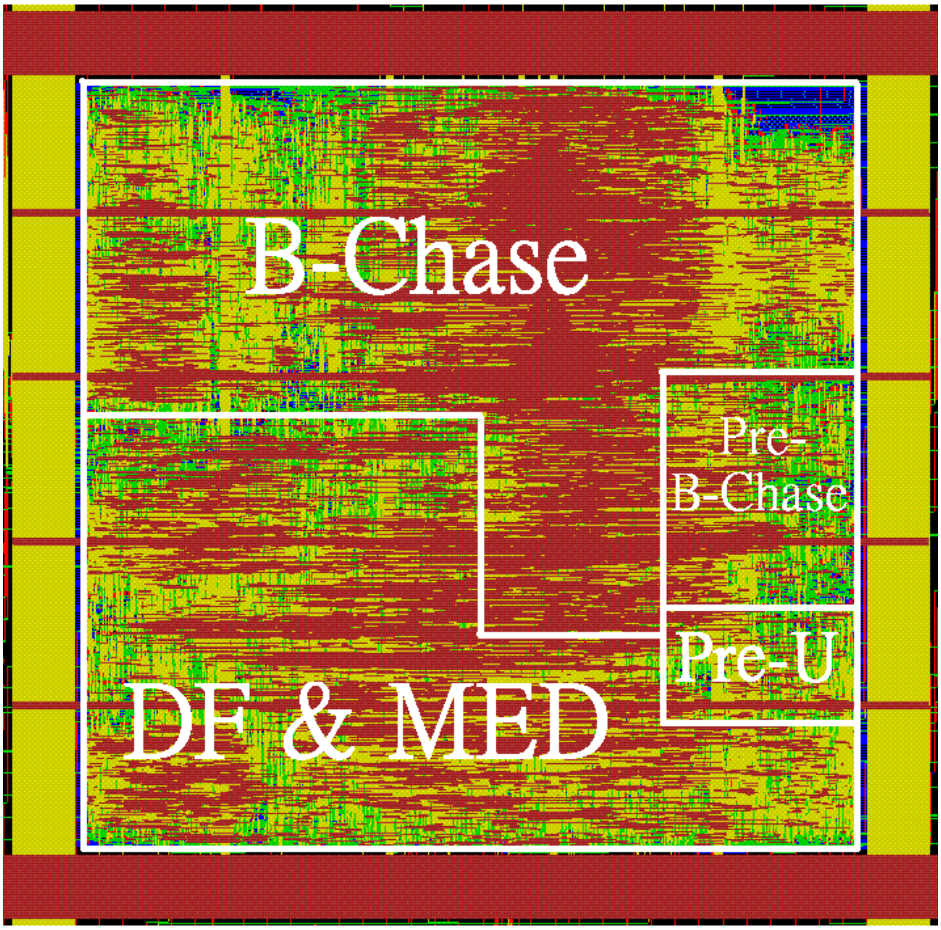

Fig. 5.17: Chip layout of the multi-mode GPGI-T1 detector.

Table 5.4: Chip characteristics of the multi-mode GPGI-T1 detector

Power Supply	1.8 V
Max. Clock	100 MHz
Max. Power	177 mW
Gate Count	141 K
Active Chip Area	1.41 mm × 1.39 mm
Process Technology	TSMC 0.18 um CMOS

Table 5.5: Supplied modes of the GPGI-T1 chip implementation

Antenna	2×2			4×4		
Modulation	QPSK	16-QAM	64-QAM	QPSK	16-QAM	64-QAM
BER Performance	Close to optimal	Close to optimal	Close to optimal	Close to optimal / Sub-optimal	Close to optimal / Sub-optimal	Close to optimal / Sub-optimal
Throughput	50Mbps	100Mbps	150Mbps	100Mbps	200Mbps	300Mbps
Power (mW)	79	95	126	118/116	137/130	177/161

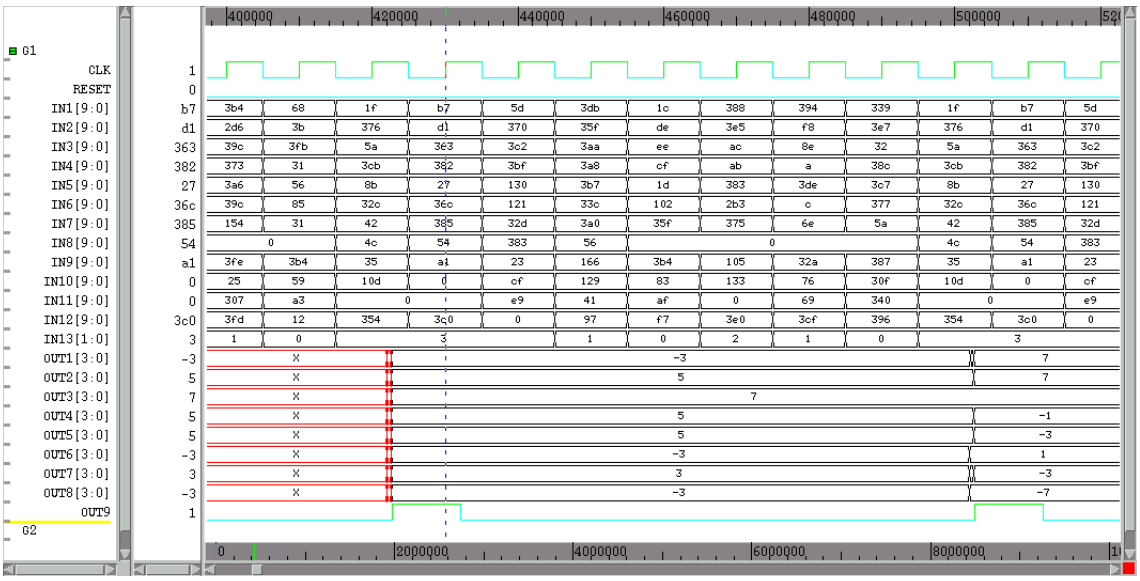

Fig. 5.18: Post-layout simulation of the multi-mode GPGI-T1 detector.

Table 5.6: Comparison of ASIC implementation for MIMO detection

	IEEE JSSC [23] Design 1	IEEE JSSC [23] Design 2	IEEE JSAC [24]	JVLSI Signal Processing [25] *	Proposed Work	
Antenna	4x4	4x4	4x4	4x4	4x4	
Modulation	16-QAM	16-QAM	16-QAM	QPSK	QPSK/16-QAM /64-QAM	
Detector	SD	SD	K-best SD	BODF	GPGI-T1	
BER performance	Optimal	Close to optimal	Close to optimal	Sub-optimal	Close to optimal / Sub-optimal	
Technology	0.25 um	0.25 um	0.35 um	0.35 um	0.18 um	
Cate Count	117 K +preproc.	50 K +preproc.	91 K +preproc.	190 K	141 K +preproc.	
Max. Clock	51 MHz	71 MHz	100 MHz	80 MHz	100 MHz	
Throughput	73 Mbps @20 dB	169 Mbps @20 dB	53.3 Mbps	128 Mbps	100/200/300 Mbps	
Power	360 mW @2.5 V	N/A	626 mW @2.8 V	608 mW @2.7 V	Close to optimal	Sub-optimal
					118/137 /177 mW	116/130 /161 mW
Power Efficiency	0.391 Mbps/mW	N/A	0.206 Mbps/Mw	0.474 Mbps/mW	0.847 /1.46/1.69 Mbps/mW	0.862 /1.54/1.86 Mbps/mW

*Note that the implementation includes the preprocessing part. If the preprocessing part is removed, the power consumption will decrease greatly.

Chapter 6

Conclusion and Future Work

In this thesis, the GPGI framework that generates many MIMO detection algorithms has been presented. Based on the GPGI framework, we propose the GPGI-T1 detection algorithm that trades off the complexity and performance by modifying the number of symbols detected first, the list length and the numbers of maximum iterations. The GPGI-T1 detection algorithm significantly reduces the multiplication complexity and has comparable BER performance compared with the existing detection algorithms. For example, in (8,8) system with 16-QAM inputs, GPGI-T1(4,1,3) can reduce the multiplication complexity by 33.9% and outperform 10 dB compared with the BODF detection at low complexity end. At high performance end, GPGI-T1(4,16,3) and GPGI-T1(4,2,3) can reduce the multiplication complexity by 21.5% and 39.3% at the penalty of 0.3 dB and 0.8 dB loss compared with the B-Chase(16) detection, respectively. With the features of low complexity, satisfactory BER performance and parallel processing, the GPGI-T1 algorithm is suitable for modern high-speed communication systems. According to the proposed GPGI-T1 algorithm, we implement a multi-mode MIMO detector using TSMC 0.18um process CMOS. The resulting implementation supports QPSK, 16-QAM, and 64-QAM modulation modes, and can work in nine modes, where three and six modes belong to (2,2) and (4,4) systems, respectively. Importantly, the resulting MIMO detection implementation possesses the comparable power efficiency among five

ASIC designs.

Bibliography

[1] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Tech. J.*, vol. 1, no. 2, pp. 41-59, Autumn 1996.

[2] I. E. Telatar, "Capacity of multi-antenna Gaussian channels," *Eur. Trans. Telecommum.*, vol.10, no. 6, pp. 585-595, Nov. 1999.

[3] A. van Zelst and T. C. W. Schenk, "Implementation of a MIMO OFDM-based wireless LAN system," *IEEE Trans. Signal Processing*, vol. 52, no. 2, pp. 483-494, Feb. 2004.

[4] E. Viterbo and J. Boutros, "A univrsal lattice decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, Jul. 1999.

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

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

[7] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. ISSSE*, Sep.-Oct. 1998, pp. 295–300.

[8] G. D. Golden, C. J. Foschini, R. A. Valenzuela, and P. W. Wolniansky, "Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture," *Electron. Letters.*, vol. 35, no. 1, pp. 14–16, Jan. 1999.

[9] B. Hassibi, "An efficient square-root algorithm for BLAST," in *Proc. IEEE Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Jun. 2000, vol. 2, pp. 737-740.

[10] Jacob Benesty, Yiteng Huang, and Jingdong Chen, "A fast recursive algorithm for optimum sequential signal detection in a BLAST system," *IEEE Trans. Signal Processing*, vol.51, no.7, pp. 1722-1730, July 2003.

[11] W. Choi, R. Negi, J. Cioffi, "Combined ML and DFE decoding for the V-BLAST system," in *Proc. IEEE Conf. Commun. (ICC)*, Jun. 2000, vol. 3, pp. 1243-1248.

[12] L. Yang, M. Chen, S. Cheng, and H. Wang, "Combined maximum likelihood and ordered successive interference cancellation grouped detection algorithm for multistream MIMO," in *Proc. IEEE Int. Sym. Spread Spectrum Tech. and Appl.*, Aug.-Sep. 2004, pp. 250-254.

[13] Vahid Tarokh, Ayman Naguib, Nambi Seshadri, and A. Robert Calderbank, "Combined array processing and space-time coding" *IEEE Trans. Inform. Theory*, vol. 45, no. 4, pp. 1121-1128, May 1999.

[14] Cong Shen, Hairuo Zhang, Lin Dai, and Shidong Zhou, "Detection algorithm improving V-BLAST performace over error propagation" *Electronic Letters*, vol. 39, no. 13, pp. 1107-1108, June 2003.

[15] D. Li, L. Cai and H. Yang, "New iterative detection algorithm for V-BLAST," in *Proc. IEEE Vehicular Technol. Conf. (VTC)*, Sep. 2004, vol. 4, pp.2444-2448.

[16] D. W. Waters and J. R. Barry, “The Chase family of detection algorithms for multiple-input multiple-output channels,” *IEEE Trans. Signal Processing*, vol.56, no.2, pp. 739-747, Feb. 2008.

[17] D. W. Waters and J. R. Barry, “The sorted-QR chase detector for multiple-input multiple-output channels,” in *Proc. IEEE Wireless Commum. and Networking Conf. (WCNC)*, Mar. 2005, vol. 1, pp. 538-543.

[18] Y. Li and Z. Luo, “Parallel detection for V-BLAST system,” in *Proc. IEEE Conf. Commun. (ICC)*, May. 2002, vol. 1, pp.340-344.

[19] Z. Luo, M. Zhao, S. Liu, and Y. Lin, “Generalized parallel interference cancellation with near-optimal detection performance,” *IEEE Trans. Signal Processing*, vol.56, no.1, pp. 304-312, Jan. 2008.

[20] D. Y. Wu and L. D. Van, "A grouped-iterative framework for MIMO detection," in *Proc. IEEE Vehicular Technol. Conf. (VTC)*, Sep. 2008, accepted, Calgary, Canada.

[21] D. Wübben, R. Böhnke, J. Rinas, V. Kühn and K. Kammeyer, “Efficient algorithm for decoding layered space-time codes,” *Electronic Letters*, vol.37, no. 22, pp.1348-1350, Oct 25, 2001.

[22] G.H. Golub and C.F. Van Loan., *Matrix Computations*, third edition, Baltimore, MD: Johns Hopkins University Press, 1996.

[23] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, “VLSI implementation of MIMO detection using the sphere decoding algorithm,” *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp.1566–1577, Jul. 2005.

[24] Z. Guo and P. Nilsson, “Algorithm and implementation of the K-Best sphere decoding for MIMO detection,” *IEEE J. Selected Areas in Communication*, vol. 24, no. 3, pp.491–503, Mar. 2006.

[25] Z. Guo and P. Nilsson, “A VLSI architecture of the square root algorithm for V-BLAST detection,” *J. VLSI Signal Processing*, vol. 44, pp.219–230, 2006.

Autobiography

吳廸優，高雄人，生於公元 1984 年 1 月 30 日。先後畢業於高雄縣立登發國民小學、高雄縣立文山中學、國立鳳山高級中學及國立成功大學數學系。2006 年 9 月進入國立交通大學資訊學院資訊科學與工程研究所碩士班就讀。其研究興趣為無線通訊系統、VISI 訊號處理、SOC 平台之軟硬體整合設計與系統分析，碩士論文題目為應用於多輸入多輸出通道之低複雜度多模式訊號偵測演算法與超大型積體電路實現。