System Design of Cooperative Wireless Networks

Milos Jorgovanovic

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-151
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-151.html

August 14, 2014

System Design of Cooperative Wireless Networks

by

Milos Jorgovanovic

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Borivoje Nikolic, Chair
Professor David Tse
Professor Jasmina Vujic

Fall 2014

System Design of Cooperative Wireless Networks

Abstract

System Design of Cooperative Wireless Networks

by

Milos Jorgovanovic

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Borivoje Nikolic, Chair

With the prediction that the number of wireless devices will reach tens of billions by 2020, wireless networks that exist today will have to be reengineered to support the increase in the number of users and the required capacity. Cooperation among terminals is envisioned as an enabling technique that can benefit from the increased number of connected devices and boost the network capacity beyond what is possible with today's network architectures, which rely on direct source-destination transmissions. While most of the work in this area focuses on designing relaying schemes that can achieve the promised capacity increase, little has been done in terms of implementing practical cooperative systems. To implement cooperation among terminals, a number of practical challenges need to be addressed across several layers of the communication system. This work focuses on the design aspects of physical and MAC communication layers, by suggesting low-complexity signal processing algorithms for multi-device cooperation and designing digital baseband hardware that showcases cooperation between two wireless devices.

We use the quantize-map-and-forward (QMF) relaying scheme, which has been shown to have good theoretical performance with multiple relays. System design of a cooperative communication link with half-duplex QMF relays is presented in three steps. First, we perform a theoretical analysis of the achievable rate and propose a local relay scheduling algorithm that performs close to the optimal relay scheduling algorithm. This local scheduling algorithm, as well as the system design approach in general, is based on the premise that relay terminals can be oblivious to other relays in the network. We show that spectral efficiency scaling with the number of relays achieved with this approach is close to optimal for both slow and fast-fading channel environments. The performance of a physical layer system design procedure for a QMF link is demonstrated by an example in which the spectral efficiency of a direct link is doubled by cooperating with three relays close to the source, as predicted by the theoretical analysis. Finally, we design the main baseband blocks for the three cooperating terminals in hardware and implement them on FPGAs. We show that the complexity of the cooperative receiver's baseband increases by 40% compared to a direct-link receiver.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

I would like to offer my most sincere gratitude to my advisor, Prof. Borivoje Nikolić, for the support and guidance he provided throughout my Ph.D. studies. Bora has been more than just a research advisor, he has been a mentor in every aspect of that word and taught me invaluable career lessons. I am indebted to him for teaching me, among other things, how to use the power of collaboration in a multi-disciplinary research environment and capture the attention of an audience with concise and clear expression of ideas. My deepest gratitude goes to Prof. David Tse, for providing the support and advice throughout my research career at Berkeley. His extraordinary capability to get to the core innovation behind any idea and express it in the simplest possible way remains a model for me for how to structure any research work. Special thanks to Prof. Jasmina Vujić, for persuading me to come to Berkeley and offering her unselfish help and advice at all times. I would also like to thank Prof. Elad Alon, Prof. Jan Rabaey, and Prof. Adam Wolisz for reviewing my research ideas and providing valuable feedback. I am particularly grateful to my colleague Vinayak Nagpal, for spending countless hours discussing details of this project. I would also like to thank I-Hsiang Wang, Matthew Weiner, Kathy Sun and Sameet Ramakrishnan, who worked with me on this project and made much of this research possible and a lot more enjoyable.

My research was funded by the Center for Circuits and Systems (C2S2) and the National Science Foundation (NSF). I would like to express my gratitude to National Instruments Corporation for donating prototyping equipment and providing software support. Special thanks goes to Hugo Andrade, Trung Tran and Sadia Malik from National Instruments, who helped immensely with the design deployment. I would also like to thank the ETF-BAFA alumni organization for funding my first visit to Bay Area, which later resulted in my decision to attend UC Berkeley.

I was based in Berkeley Wireless Research Center (BWRC) for the most part of my career at Berkeley. I would like to thank all faculty directors and staff of BWRC for making it an excellent place to work. It was a privilege to be part of this center.

I would like to extend my gratitude to my dear friends and colleagues, who I owe many thanks for providing companionship and support. I would like to thank all current and past members of the COMIC research group. Special thanks to my senior colleagues Dušan Stepanović, Ružica Jevtić, Zhengya Zhang and Farhana Sheikh, for sharing their experiences and offering life and career advice. I would also like to extend my gratitude to my colleagues from the EECS department: Steven Callender, Mitchel Kline, Richie Przybyla, Rikky Muller, Katerina Papadopoulou, Sharon Xiao, Matthew Spencer, Rachel Hochman, Amanda Pratt, Ranko Sredojević and Gireeja Ranade, for making this PhD a great life experience. I would also like to thank my dear friends Milutin Janjić and Jelena Simjanović, for love and encouragement.

This journey would not be possible without continuing support and love of my family. I would not have been writing this if it wasn't for my parents, Emilija and Nebojša Jorgovanović, and my sister Ana, who made it possible for me to take my schooling to the highest level. My love and gratitude goes to my wife Aleksandra, who has been a tremendous sup-

port and encouraged me to endure through the most difficult times. And finally, I would like to thank the little monster from outer space, for showing up right when I was starting to write Chapter 2, for keeping me awake during days and nights and for erasing all stress with a single smile. She has been the best motivation I could wish for to finish this dissertation.

Chapter 1

Introduction

In the last two decades wireless communications have evolved from the classic broadcast applications (radio and TV) to the mobile telephony and widespread use of the wireless internet. This evolution has had a profound impact on human lives - it made communication *personal*. For example, instead of calling a landline number that was tied to a physical location, one is today calling a cellular number that is tied to a specific person. Development of wireless communication systems made us *connected* virtually everywhere. Instead of accessing the internet from a location with the wired internet connection, each person has access to internet at the touch of their finger no matter where they are.

However, these perks come with a price tag. With the development of wireless communication systems, the wireless data demand was growing in parallel. According to the latest Cisco mobile data traffic forecast [10], this demand causes the mobile data traffic to grow by 50-90% every year. This trend is expected to continue, due to rapid increase in machine-to-machine (M2M) communication. The main task for an army of communication system engineers is to address the data traffic surge and design communication networks that can use the available spectrum more efficiently.

1.1 Motivation

One approach to increase the spectral efficiency that has recently drawn a lot of attention among the communication community is to use cooperation among wireless terminals [23, 44]. In particular, Ozgur et al. show in [44] that the spectral efficiency of an interference-limited ad-hoc network with n source-destination pairs (Figure 1.1a) scales linearly with the number of pairs n in the network, for a very large n. In other words, as the number of node pairs increases, the per-node pair capacity stays constant. They prove that the physical layer scheme that can achieve this scaling is based on hierarchical cooperation among the terminals.

The modern wireless networks, however, are not ad-hoc in nature. For example, the cellular (GSM, CDMA, LTE, LTE-Advanced) and WLAN (802.11b,g,n,ac) standards adopted

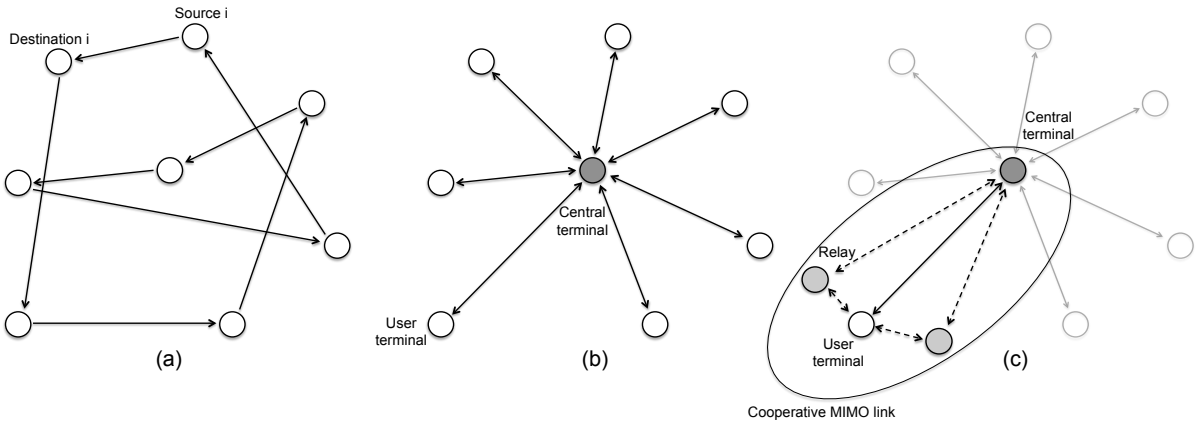

Figure 1.1: Models for (a) ad-hoc network configuration with n source-destination node pairs, (b) star network configuration with one central terminal (shaded) and n user terminals, and (c) cooperative link as a part of star network configuration.

a star network architecture, presented on Figure 1.1b, with one central terminal (base station in cellular and access point in WLAN) and multiple user terminals (mobile stations). According to the star network architecture, transmissions only happen between the central terminal and the user terminals and the spectral efficiency of the network remains constant as the number of users increases. In other words, as we add more users to the network, the per-user capacity decreases.

Recognizing the benefits of cooperation, standards committees have started including different forms of cooperation in LTE and WLAN standards. For example, the latest release of LTE standard includes support for cooperation between base stations called Cooperative Multi Point (CoMP) [11]. There is a lot of research effort in showing that the CoMP can be efficiently implemented in the future versions of the LTE system [21, 35, 43]. Another example is the introduction of relays in LTE standard to improve the coverage in existing LTE cells [17, 42, 43]. While these steps point to the right direction, the benefits in terms of capacity scaling are far from promised by cooperation schemes from [44].

In addition to the cooperation between wireless terminals, another key idea that enables linear increase in spectral efficiency with the number of nodes in [44] is using Multiple-Input Multiple-Output (MIMO) transmissions between the clusters of cooperating nodes. MIMO is a widely recognized technique to improve spectral efficiency of point-to-point communication links, by sending independent data streams across transmit antennas. Point-to-point MIMO has been an integral part of many modern wireless standards, including LTE, LTE-Advanced, HSPA+, 802.11n and WiMAX.

While using point-to-point MIMO enables drastic spectral efficiency increase, it has limitations that have quickly been reached in most of the standards that use this technique. To achieve desired spectral efficiency scaling, the channels between transmit and receive antennas need to be statistically independent. This requirement is satisfied by having enough

separation between antennas at each of the terminals. For example, for 2.4 GHz ISM band the minimum distance between antennas is at least 6 cm and increases for lower carrier frequencies used in cellular. Keeping in mind the size of today's cellular and WiFi devices, it means that most of them can have $1-4$ antennas, which has already been implemented in modern devices. This means that using point-to-point MIMO has already reached its limits.

Authors of [44] suggest that limitations of point-to-point MIMO scheme can be alleviated by cooperation. In particular, they suggest that single-antenna terminals can cooperate to create clusters and thus form much larger arrays of transmit and receive antennas. Because of the MIMO transmissions between these clusters, this scheme can achieve much larger gains in terms of spectral efficiency compared to the point-to-point links, which leads to the linear increase in spectral efficiency with the number of nodes.

The main motivation for this work is to *merge* ideas of cooperation and MIMO (conveniently dubbed *Cooperative MIMO*) with the star network architecture of today's wireless systems. To do so, we focus on an individual communication link between the user terminal and the central terminal in the star network architecture, as shown in Figure 1.1c. The user terminal can cooperate with other wireless devices, called relays, which are assumed not to have any information of their own to send to the central terminal.

The relays could be either implemented as separate type of devices with the sole purpose to relay information for the active network users and thus improve network performance, or summoned among the idle network users willing to share their battery resource. In any of the two scenarios for relay implementation, there are two key features of this approach:

1. The relays are *wireless*, which means they do not require a backhaul connection to the rest of the network infrastructure. This makes them very easy to deploy virtually anywhere: electricity posts, electrical outlets, ceiling, etc.

2. The relays are using the same resource as the source, i.e. there is no separate channel used for relaying. The gain comes solely from spatial multiplexing across multiple antennas.

The desired outcome of this thesis is to determine how the spectral efficiency scales with the number of relays and propose algorithms that can achieve this scaling with reasonable implementation complexity. We support the claim that the complexity is indeed low by demonstrating implementation of the complete cooperative-relaying link baseband in hardware.

1.2 Related Work

Cooperation in wireless networks attracted a great research interest in the past, starting with the first mention of a *relay channel* in the works of Meulen [36] and Cover and El Gamal [12]. After four decades of intensive research, the fundamental limit of a cooperative-relaying link,

its capacity, remains unknown. Even though the capacity has not been computed, a number of cooperative-relaying schemes have been suggested, including amplify-and-forward (AF), decode-and-forward (DF), compress-and-forward (CF) [12], [32], quantize-map-and-forward (QMF) [3, 4]. Avestimehr et al. have shown in [3, 4] that, unlike other schemes, the QMF scheme performs within a constant gap from capacity for an arbitrary number of relays and, in particular, within a single bit from capacity for a single-relay link. It is mainly because of this property that we decided to focus on the implementation of the QMF scheme in this work.

Several papers have been published that propose system design methods of a single-relay QMF link. In [38] and [40] we focus on the channel coding aspect of the QMF scheme and propose a system design procedure for a single-relay link. Similar channel coding principles can be applied to the multi-relay scenario, and we readily follow that line of thought in this work. The authors of [15] present an experimental evaluation of a hybrid DF-QMF scheme without channel coding at the relay. They show that in some scenarios it is beneficial to use DF over QMF. A detailed analysis of a single-relay system design with the DF relaying scheme has been done in [53]. All these papers offer design description and software implementations, but none of them presents the hardware implementation of the relaying link and demonstrates the performance-complexity tradeoff that comes with relaying. To the best of our knowledge, this thesis presents the first hardware implementation of a single-relay QMF relaying link.

There has been a lot of published work that focuses on the complexity issue of a multi-relay scheduling optimization [5, 45, 16, 8, 59]. In [5] it is claimed that the number of optimization parameters, which is exponential with the number of relays (2^N) can be reduced to a small number of non-zero parameters ($N+1$), and the proof is provided for some special network cases. However, these papers do not provide an algorithm to achieve this promised linear-complexity optimization. One of the corollaries of our work is that we have proved that the optimal number of non-zero scheduling parameters is $N+1$ for particular channel conditions and an arbitrary number of relays N. As a result of that analysis, we suggest a very simple relay scheduling algorithm that does not require any optimization and performs close to the optimal relay-scheduling scheme.

One aspect of a multi-relay cooperative-relaying link system design is presented in [52]. The authors argue that the LDPC codes at the relays can be combined into a single factor graph and decoded jointly at the destination. In our work we adopt a similar concept of combining relays' channel codes, but we use the LDPC-LDGM channel coding method that we developed in [38].

1.3 Scope and Organization

This thesis answers three key questions about the implementation of a QMF cooperative-relaying link:

1. How much data rate gain can we achieve by using multiple relays?

2. How to design the physical layer of a QMF cooperative link with multiple relays?

3. What is the complexity of the hardware implementation?

Basic design concepts, such as the QMF scheme and its influence on the achievable rate of communication, channel coding, MIMO detection and the DBLAST space-time scheme are described in Chapter 2. We also present the intuitive explanation of where the cooperative gain comes from in a cooperative-relaying link. Design concepts developed in this chapter will be used throughout this thesis.

The theoretical analysis of a multi-relay cooperative link is presented in Chapter 3. To answer the main question of this chapter, how much data rate gain can be achieved through QMF relaying, we study the problem of relay scheduling. We present a heuristic scheduling scheme that does not require any optimization and performs very close to the optimal scheme. We show on an example that cooperative relaying can improve the data rate of a direct communication link (i.e. the link with no relays) by a factor of 3, once we add five relays and enable cooperation.

To get an idea of how to achieve this promised data-rate gain, in Chapter 4 we show key physical-layer design concepts. We pay special attention to the design of a receiver at the destination, since that is the most processing-heavy terminal in the system. To demonstrate the achievable data rate gain, we present a detailed design procedure of a three-relay cooperative link and show that it can double the data rate of a direct link. Alternatively, we can keep the same data rate as in the direct link, but reduce the transmit power of the source. We show that with a five-relay cooperative link, the transmit power of the source can be reduced by 9 dB compared to the source terminal in a direct link.

In Chapter 5 we present an FPGA implementation of the digital baseband of the three cooperating terminals: the source, the relay and the destination. We demonstrate that the cooperation with a single relay increases the complexity of the baseband receiver at the destination by around 40%. It is interesting to note that one relay increases the data rate of a direct link by 33%. Therefore, to increase the data rate by a third, we pay approximately the same penalty in the hardware complexity of the receiver baseband.

Finally, Chapter 6 presents a summary of the results and provides directions for future work.

Chapter 2

Basic Design Concepts for Cooperative Relaying with QMF

In this chapter we introduce the basic design principles for a cooperative-relaying communication link, which will be used extensively throughout the dissertation.

One of the most important design choices in cooperative relaying is the relaying scheme. In this chapter, we focus on the quantize-map-and-forward (QMF) relaying scheme and its main advantages and disadvantages from the system design point of view. The main benefit of the QMF scheme is that it has been shown to perform close to capacity for an arbitrary number of relays [4]. The main disadvantage is that it requires complex joint decoding of both the source's and the relay's messages.

We argue that this complexity can be reduced by separating the relationships between messages that stem from the MIMO channel and the channel coding [38]. To eliminate the relationship that comes from the MIMO channel, we propose to use diagonal Bell-labs space-time (DBLAST) scheme to coordinate source and relay transmissions. Using the liner MIMO detection scheme enables separate transmitted streams to be received independently, which eliminates any relationship among the transmitted streams that comes from the MIMO channel. The channel coding for QMF requires joint decoding of the source's and the relays' codewords. We show in this section how this joint decoding can be simplified to a standard Tanner graph used for representing linear block codes, which enables using standard decoding architectures.

Lastly, we lay foundation for relay scheduling analysis by presenting a single-relay scheduling scenario. We show the influence of the QMF scheme on the capacity of the cooperative-relaying link, by quantifying the loss that is incurred due to the quantization at the relay.

2.1 Cooperative Relaying

After seeing how the concept of cooperative MIMO fits into the star network architecture from the introduction, we turn our attention to studying a single source-destination link with

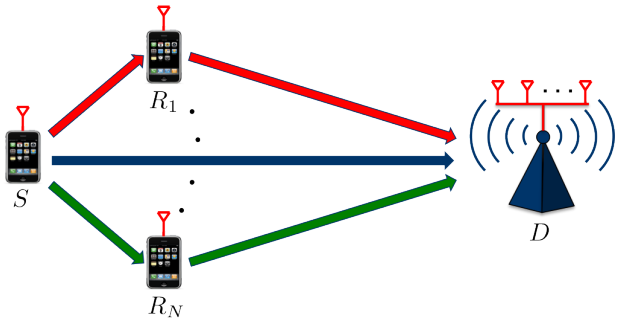

Figure 2.1: Cooperative-relaying link with multiple relays.

relays presented in Figure 2.1. Among the communication theory community, this problem is typically referred to as cooperative relaying. Cooperative relaying is studied from two perspectives, depending on how the relays are used: cooperative diversity and cooperative multiplexing.

Cooperative diversity improves the reliability of transmission and historically has drawn more attention among the research community. Even in the latest release of LTE-Advanced cellular standard, relays are used mainly to provide service to those users who have very bad channel to the base station (cell-edge users) or no connection at all [17]. The idea is to provide alternative connection paths through the relays, which increases the odds that at least one path to the base station will have a satisfying channel quality. An extensive study on cooperative diversity techniques and limits is provided in [32].

Cooperative multiplexing refers to using relays to improve spectral efficiency of a communication link, which is the scenario we focus on in this thesis. To achieve multiplexing gain, a source node in a cooperative link transmits its message with the higher spectral efficiency than what can be supported by a direct source-destination link. The destination would be able to decode the source's message only after it receives additional information (also known as *side information*) about it from the relays.

Diversity and multiplexing gain can be traded off by choosing different cooperation schemes. Fundamental understanding of this tradeoff can be looked up in [64, 14]. Diversity-multiplexing tradeoff (DMT) for a single-relay half-duplex channel has been studied in [46, 39]. This study has shown that a cooperative MIMO link with the relay close to the source can achieve significant multiplexing gain compared to a direct link, and that this gain can be achieved using a QMF cooperation scheme.

An important constraint on the relays is that they need to be able to both receive information from the source and transmit the processed version of what they have received to the destination. Generally, it is very difficult to build full-duplex radios which can receive and transmit at the same time over the same frequency channel. Therefore, we consider the relays to be half-duplex, which means that at any time instance their radios will be set either to receive the message from the source or transmit their message to the destination.

System setup of a cooperative-relaying link is presented in Figure 2.1. We assume an

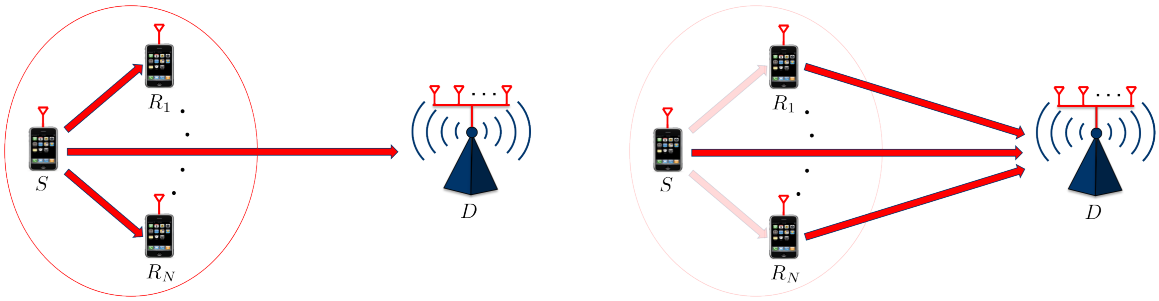

Figure 2.2: Intuition behind the cooperative multiplexing gain: the cooperation phase (left) and the MIMO phase (right).

uplink cooperative wireless link with an information source S, a destination D, and N half-duplex relays $(R_1, R_2, ..., R_N)$ close to the source, available for physical-layer cooperation. Relays are assumed to use time-division-duplexing: they receive the signal from the source for some fraction of total time (listening phase), process this received signal, and then forward it to the destination in the remaining fraction (transmit phase). The source is assumed to continuously transmit data and the destination continuously receives it.

2.1.1 Intuition behind the Cooperative Multiplexing Gain

We argued in the introduction that the MIMO transmission is one of the two essential steps for achieving a higher spectral efficiency. But the multiplexing gain of the cooperative-relaying link does not depend solely on the MIMO component. Figure 2.2 presents an intuitive way of picturing where the multiplexing gain comes from. For simplicity, we assume that all relays in Figure 2.2 listen during the same time fraction f. We also assume that all $T-D$ links have the same signal-to-noise ration (SNR), SNR_{TD}, for $T \in \{S, R_1, R_2, ..., R_N\}$, and that all $S - R_i$ links have the same SNR: $SNR_{SR_i} = SNR_{SD}^{\eta}$. Factor η denotes the proximity gain, i.e. relays' proximity to the source compared to the destination. Now we can picture two distinct phases of cooperative relaying, as presented in Figure 2.2:

1. *The Cooperation Phase.* This is the phase that corresponds to the listening phase of the relays. During this phase the source shares its information with the relays around it, so that they can form their own messages to transmit them during the next phase. The amount of information that the source transmits to the relays and the destination is upper-bounded by the capacity of the broadcast channel. A back-of-the-envelope calculation of this capacity, for large values of SNR_{SD} is:

$$\begin{aligned} C_{COOP} &\approx \log_2(1 + SNR_{SD} + SNR_{SR_1} + ... + SNR_{SR_N}) \\ &\approx \log_2(1 + SNR_{SD} + N \cdot SNR_{SD}^{\eta}) \\ &\approx \eta \log_2(1 + SNR_{SD}) + \log_2 N \end{aligned} \tag{2.1}$$

From this expression we conclude that the *multiplexing gain during the cooperation phase comes from the proximity gain* η, rather than the number of relays N.

2. *The MIMO Phase.* This phase corresponds to the transmit phase of the relays. The transmit antenna array is formed by the single antenna cooperating terminals: the source and the relays that are in the transmit phase. The receive antenna array is entirely placed at the destination. To achieve the full multiplexing gain of this *distributed* MIMO link, we assume the number of receive antennas at the destination is larger or equal than the total number of transmit antennas. The amount of information that the destination receives is bound by the capacity of the MIMO channel, expression (2.2). From this expression we conclude that *the multiplexing gain during the MIMO phase comes from the number of transmitted streams, which is proportional to the number of relays,* N.

$$C_{MIMO} \approx (N+1)\log_2(1+SNR_{SD}) \tag{2.2}$$

To get the maximum multiplexing gain out of this link, we would increase the number of relays until we balance the capacity of the cooperation phase and the capacity of the MIMO phase. Intuitively this balance means that the relays are getting just enough information during the cooperation phase, which they can transmit to the destination during the MIMO phase. This balance is achieved when:

$$N \approx \eta - 1. \tag{2.3}$$

Therefore, increasing the number of relays beyond the proximity gain does not provide the degree-of-freedom gain, but rather just the power gain. We come back to this intuitive conclusion at the end of the next chapter.

2.1.2 Relaying Schemes

The most important design aspect of cooperative relaying is the processing algorithm used at the relays. Two fundamental schemes have been suggested in [12]: decode-and-forward (DF) and compress-and-forward (CF). We summarize the schemes and their achievable data rates over a single-relay channel (presented in Figure 2.3), using results from [12, 41] .

For the half-duplex channel from Figure 2.3, we the source's message $\underline{x}_S$ into two parts: $\underline{x}'_S$ is transmitted during the listening phase of the relay, and $\underline{x}''_S$ is transmitted during the transmitting phase. The relay's received signal $\underline{y}_R$ refers to the listening phase of the relay, and the message $\underline{x}_R$ refers to the transmit phase. The received signal at the destination, $\underline{y}_D$, is also split into two parts: $\underline{y}'_D$ is received during the listening phase of the relay, and $\underline{y}''_D$ is received during the transmit phase. We use f to denote the listening fraction of the relay.

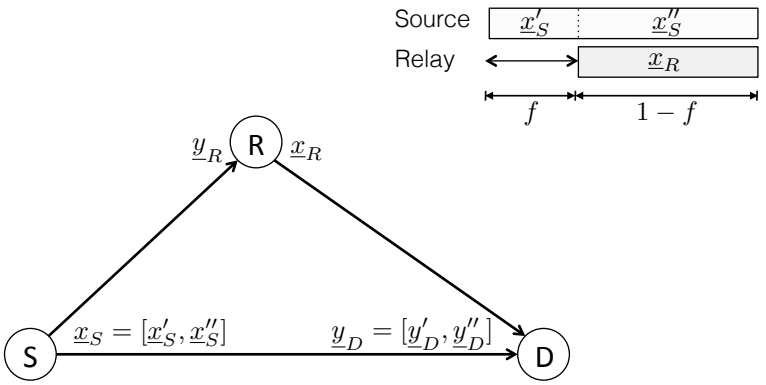

Figure 2.3: Half-duplex relay channel.

Decode-and-Forward

DF scheme requires that the relay decodes the message from the source. This limits the upper bound on the rate between the source and the relay to $I(\underline{x}'_S; \underline{y}_R)$, which eventually limits the overall achievable rate of communication.

After the relay decodes the source's message, it re-encodes it and transmits it to the destination, which can then perform soft combining of the received messages from the source and the relay. For the relay channel presented in Figure 2.3, the upper bound on the achievable rate is given by [12, 41]:

$$R_{DF} = \max_{P(\underline{x}_S, \underline{x}_R)} \min\{fI(\underline{x}'_S; \underline{y}_R) + (1-f)I(\underline{x}''_S; \underline{y}''_D|\underline{x}_R), \\ fI(\underline{x}'_S; \underline{y}'_D) + (1-f)I(\underline{x}''_S, \underline{x}_R; \underline{y}''_D)\} \tag{2.4}$$

Compress-and-Forward

Unlike with the DF scheme, in the CF scheme the relay forwards soft information about the received signal from the source [12]. Therefore there is no need to constraint the relay's listening time because of the decoding. The destination first decodes the message $\underline{x}_R$ it receives from the relay, and then use that decoded message as a side information to decode the original stream from the source. The relay compresses the received signal $\underline{y}_R$ into $\hat{\underline{y}}_R$ using Wyner-Ziv compression method, with rate $I(\hat{\underline{y}}_R; \underline{y}_R|\underline{y}'_D)$ [62]. Compressed message $\hat{\underline{y}}_R$ is then encoded into the new set of transmit symbols, $\underline{x}_R$.

For the destination to be able to decode the message $\underline{x}_R$ it receives from the relay, the rate of compression must not exceed the capacity of the R-D link [12]:

$$fI(\hat{\underline{y}}_R; \underline{y}_R|\underline{y}'_D) \leq (1-f)I(\underline{x}_R; \underline{y}''_D) \tag{2.5}$$

The maximum achievable rate of communication using CF is [12, 41]:

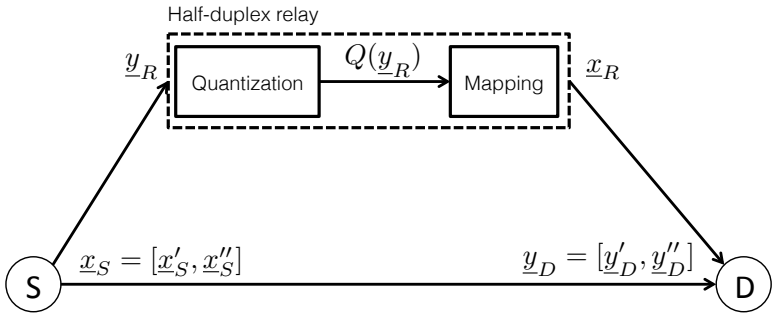

Figure 2.4: Quantize-map-and-forward relaying scheme applied to a half-duplex relay channel. Received signal at the relay is first quantized and then mapped into the new set of symbols.

$$R_{CF} = \max_{P(\underline{x}_S, \underline{x}_R, \hat{\underline{y}}_R, \underline{y}_D)} fI(\underline{x}_S; \hat{\underline{y}}_R, \underline{y}'_D) + (1-f)I(\underline{x}_R; \underline{y}'_D), \tag{2.6}$$

under condition (2.5).

The caveat of the CF scheme is that the relay needs to know the forward channel to the destination, so it can compress the transmit stream in a way the destination can decode. CF scheme has been shown to be optimal in a single-relay scenario, but suboptimal in a multi-relay network [4].

2.1.3 Quantize-Map-and-Forward Relaying Scheme

Recently, the quantize-map-and-forward (QMF) relaying scheme [3, 4] has received attention from the research community, both from the perspective of performance and implementation. The QMF scheme suggests that the relay quantizes the message $\underline{y}_R$ it receives from the source into $Q(\underline{y}_R)$ without performing Wyner-Ziv compression algorithm, and maps the quantized bits into a random codeword $\underline{x}_R$ before forwarding it to the destination, as presented in Figure 2.4. The achievable QMF rate is the same as in case of the CF scheme (2.6), but without the constraint given by (2.5).

The relay can perform either a scalar or a vector quantization. Vector quantization suggests that the relay quantizes the whole received vector $\underline{y}_R$. In this work, we focus exclusively on the scalar quantization, which means that the received symbols in $\underline{y}_R$ are quantized independently. The reason is that the QMF performs well even with the scalar quantization at the relays, which is much simpler to implement [4].

In the QMF scheme, the destination does not need to decode the quantized bits $Q(\underline{y}_R)$ independently, and thus the relay does not need the forward channel knowledge like in the CF scheme. In fact, the destination has to *jointly decode* the message from the source and the side information from the relays. In other words, the destination sees $\{\underline{x}_S, Q(\underline{y}_R)\}$ as one big message which consists of the two parts, and it decodes them both at the same time.

Unlike the CF scheme, QMF is shown to perform within a constant gap from capacity for an arbitrary number of relays [3, 4]. Besides, QMF does not require forward channel knowledge at the relays and thus simplifies the system design and communication overhead. Thus our interest in studying cooperation with multiple half-duplex QMF relays and suggesting and evaluating practical algorithms for implementation of such a communication system.

2.2 MIMO Receiver at the Destination

In Section 2.1 we have presented relaying schemes that describe the fundamental processing step related to the transmit part of a cooperative MIMO system. In this section we focus on the receive part, which consists of a MIMO receiver at the destination. MIMO receiver requires special attention, since it is one of the most complex signal processing blocks of the receiver baseband.

When it comes to choosing the right MIMO receiver architecture, there is a fundamental tradeoff between the performance and the implementation complexity. In this section, we present the two main categories of MIMO receivers, the linear MIMO receivers and the maximum-likelihood (ML) MIMO receivers. The linear detection indicates simple signal processing, since the highest complexity operation involves inverting a matrix of complex numbers. The ML MIMO receivers on the other hand deploy search algorithms that search for those symbols that are most likely to have been transmitted by the transmit antennas. This method yields very good performance at the price of higher complexity of implementation.

For the rest of this section, we will assume that the job of a MIMO detector is to estimate the symbol vector $\underline{x}$, which consists of N_{TX} symbols transmitted by the source and the relays at some time instance. The destination receives the vector of N_{RX} received symbols, $\underline{y}$, which can be represented as:

$$\underline{y} = H \times \underline{x} + \underline{z}, \tag{2.7}$$

where $H = [\underline{h}_1, \underline{h}_2, ..., \underline{h}_{N_{TX}}]$ is an $N_{RX} \times N_{TX}$ matrix of channel coefficients (assuming flat-fading channel conditions), and $\underline{z}$ is an $N_{RX} \times 1$ white noise vector. For the purpose of this section, we will assume that the transmitted symbols in $\underline{x}$ are not correlated i.e. they belong to different messages. A simplified schematic that describes the process of symbol-level MIMO detection is presented in Figure 2.5.

2.2.1 Linear MIMO Receivers

The basic principle of the linear MIMO detectors is to apply a linear filter F, where $F^T = [\underline{f}_1, \underline{f}_2, ..., \underline{f}_{N_{RX}}]$, to the received-symbol vector $\underline{y}$ and get symbol-estimates of the transmitted symbols in $\underline{x}$:

$$\hat{\underline{x}} = F \cdot \underline{y} \tag{2.8}$$

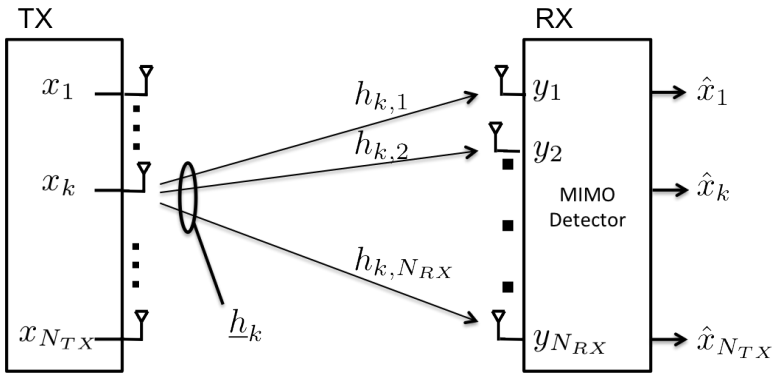

Figure 2.5: Schematic of the symbol-level MIMO detection. Transmitted symbols $x_1, x_2, \ldots, x_{N_{TX}}$ are estimated at the receiver based on the received symbols $y_1, y_2, \ldots, y_{N_{RX}}$ and the knowledge of the MIMO channel H.

The linear filtering essentially means that symbols in $\underline{x}$ are estimated one at the time, independently from the other symbols. This becomes obvious if we rewrite expression (2.8) as:

$$\hat{x}_k = \underline{f}_k^T \cdot \underline{y} \tag{2.9}$$

Different linear detection methods correspond to different choices for the filter matrix F. In the remainder of this section, we present four standard methods for linear MIMO detection: matched filter, decorrelator, MMSE detector, and MMSE-SIC detector.

Matched Filter

First, let's assume that the transmit vector $\underline{x}$ contains only one symbol, x_1. The MIMO channel becomes single-input mutliple-output (SIMO) channel, and the equation (2.7) changes into:

$$\underline{y} = \underline{h_1} \cdot x_1 + \underline{z} \tag{2.10}$$

Let's assume the noise is white and the noise samples across receive antennas are uncorrelated, i.e. $\underline{z} \sim CN(0, \frac{1}{SNR_{IN}} \cdot I)$, where SNR_{IN} denotes the input SNR, defined as the ratio of the transmitted symbol power and the noise power at one receive antenna:

$$SNR_{IN} = \frac{\sigma_{x_1}^2}{\sigma_{z_i}^2}, \forall i \in \{1, 2, ..., N_{RX}\} \tag{2.11}$$

Under this assumption, the optimal method to estimate the transmitted symbol x_1 is to project the received vector $\underline{y}$ onto the channel vector $\underline{h_1}$. This type of SIMO receiver is called the matched filter, and the symbol estimate can be calculated as:

$$\begin{aligned}\hat{x}_1 &= \frac{\underline{h_1}^H}{|\underline{h_1}|^2} \cdot \underline{y} \\ &= x_1 + \frac{\underline{h_1}^H \cdot \underline{z}}{|\underline{h_1}|^2}.\end{aligned} \tag{2.12}$$

The output SNR of the matched filter can be calculated from expression (5.1) as:

$$SNR_{OUT} = |\underline{h_1}|^2 \cdot SNR_{IN} \tag{2.13}$$

Obviously, the SNR gain is determined by the channel coefficients of the SIMO channel vector $\underline{h_1}$. Extending the matched filter to MIMO case would simply mean to apply the set of matched filters to the received vector, to get all transmitted streams:

$$\begin{aligned}\hat{x}_i &= \frac{\underline{h_i}^H}{|\underline{h_i}|^2} \cdot \underline{y} \\ &= x_i + \frac{\underline{h_i}^H \cdot \underline{z}}{|\underline{h_i}|^2}, \quad \forall i, i \in \{1, 2, \ldots, N_{TX}\}\end{aligned} \tag{2.14}$$

While matched filter is an optimal SIMO receiver (when the receive noise $\underline{z}$ is white), it is far from being an optimal MIMO receiver. This is because matched filter works well with the white noise, but does nothing to take into account the inter-stream interference due to the MIMO channel. It can be used as a MIMO receiver only in the very low SNR conditions, when the MIMO channel coefficients are much smaller than the receiver noise samples in $\underline{z}$.

Decorrelator

In terms of the regular MIMO channel given by (2.7), we quickly conclude that the simplest detection method is to simply invert the effect of the MIMO channel. Assuming for a moment that the channel matrix H is square, this means that $F = H^{-1}$ and the vector of symbol estimates becomes:

$$\underline{\hat{x}} = H^{-1} \cdot \underline{y} \tag{2.15}$$

If the channel matrix H is rectangular, we can use pseudo-inverse to invert the effect of the MIMO channel:

$$\underline{\hat{x}} = (H^H H)^{-1} H^H \cdot \underline{y} \tag{2.16}$$

When we expand the received vector $\underline{y}$ using expression (2.7), the vector of symbol estimates becomes:

$$\underline{\hat{x}} = \underline{x} + (H^H H)^{-1} H^H \underline{z} \tag{2.17}$$

Note that this detection method would indeed be optimal, if there was not for the noise part. In other words, if the SNR_{IN} of this MIMO channel is infinite, the channel equation (2.7) becomes deterministic and the decorrelator becomes an optimal way to do symbol detection.

Another important virtue of decorrelator is that it perfectly *cancels interference* of all the other transmitted streams. This is obvious from expression (2.17), since each symbol estimate sees influence from the corresponding transmitted symbol and the filtered receiver noise. Expression (2.17) also shows the weakest point of the decorrelator: the performance in low-SNR regime is completely dominated by the noise term, which gets amplified. This happens because the decorrelator tries to cancel interference from the other streams, while it should be really focusing on the noise that comes from the receiver.

MMSE

The minimum-mean square error (MMSE) method is a linear detection method that minimized the mean square error of the estimate $\underline{\hat{x}}$. It represents a balance between the matched filter, which perform well in low-SNR conditions, and the decorrelator, which performs well in high-SNR conditions. Since the working conditions of most communication systems are somewhere in between these two extremes, the MMSE receiver is almost always the preferred choice to the matched filter and the decorrelator.

To understand the MMSE detection method, we adopt the derivation from [14]. Similarly as we did in the matched filter scenario, we focus on a single stream for now, say x_i. We can rewrite the MIMO channel equation (2.7) as:

$$\underline{y} = \underline{h}_i \cdot x_i + \sum_{\substack{j=1 \\ j \neq i}}^{j=N_{TX}} \underline{h}_j \cdot x_j + \underline{z} \tag{2.18}$$

The goal of linear detection is to estimate the transmitted symbol x_i independently of the other transmitted symbols. As a matter of fact, the influence of other transmitted symbols as well as the receiver noise can be regarded as interference term, $\underline{z_i}$:

$$\underline{y} = \underline{h}_i \cdot x_i + \underline{z}_i, \tag{2.19}$$

where the interference term $\underline{z}_i$ has the covariance matrix:

$$K_{\underline{z}_i} = \sum_{\substack{j=1 \\ j \neq i}}^{j=N_{TX}} \sigma_{x_j}^2 \cdot \underline{h}_j \cdot \underline{h}_j^H + \frac{1}{SNR_{IN}} \cdot I \tag{2.20}$$

We notice a striking resemblance of expression (2.19) to the SIMO channel used in matched filter derivation, (2.10). The only difference is that the noise $\underline{z}_i$ in expression (2.19) is not white. As stated in [14], this observation suggests a natural strategy for the colored noise situation: first whiten the noise, and then simply apply the appropriate matched filter. To whiten the noise $\underline{z}_i$, we apply the whitening filter $K_{\underline{z}_i}^{-1/2}$ to both sides of equation (2.19):

$$\begin{aligned} K_{\underline{z}_i}^{-1/2} \cdot \underline{y} &= K_{\underline{z}_i}^{-1/2} \cdot \underline{h}_i \cdot x_i + K_{\underline{z}_i}^{-1/2} \cdot \underline{z}_i \\ &= K_{\underline{z}_i}^{-1/2} \cdot \underline{h}_i \cdot x_i + \tilde{\underline{z}}, \end{aligned} \tag{2.21}$$

where now noise $\tilde{\underline{z}}$ is white and has the same distribution as the original receiver noise $\underline{z}$: $\tilde{\underline{z}} \sim \underline{z}$. Now it makes sense to apply the matched filter to expression (2.21):

$$\begin{aligned} \frac{(K_{\underline{z}_i}^{-1/2} \cdot \underline{h}_i)^H}{\underline{h}_i^H \cdot K_{\underline{z}_i}^{-1} \cdot \underline{h}_i} K_{\underline{z}_i}^{-1/2} \cdot \underline{y} &= \frac{(K_{\underline{z}_i}^{-1/2} \cdot \underline{h}_i)^H}{\underline{h}_i^H \cdot K_{\underline{z}_i}^{-1} \cdot \underline{h}_i} K_{\underline{z}_i}^{-1/2} \cdot \underline{h}_i \cdot x_i + \frac{(K_{\underline{z}_i}^{-1/2} \cdot \underline{h}_i)^H}{\underline{h}_i^H \cdot K_{\underline{z}_i}^{-1} \cdot \underline{h}_i} \cdot \tilde{\underline{z}} \\ &= x_i + \frac{\underline{h}_i^H \cdot K_{\underline{z}_i}^{-1}}{\underline{h}_i^H \cdot K_{\underline{z}_i}^{-1} \cdot \underline{h}_i} \cdot \underline{z}_i \\ &= x_i + z_i, \end{aligned} \tag{2.22}$$

where

$$\sigma_{z_i}^2 = \frac{1}{\underline{h}_i^H \cdot K_{\underline{z}_i}^{-1} \cdot \underline{h}_i} \tag{2.23}$$

is the noise power in the last line of expression (2.22). The estimated symbol $\hat{x}_i$ can be found by applying the scalar MMSE estimator to expression (2.22):

$$\begin{aligned} \hat{x}_i &= \frac{\underline{h}_i^H \cdot K_{\underline{z}_i}^{-1} \cdot \underline{h}_i}{\underline{h}_i^H \cdot K_{\underline{z}_i}^{-1} \cdot \underline{h}_i + 1} \cdot \frac{\underline{h}_i^H K_{\underline{z}_i}^{-1}}{\underline{h}_i^H \cdot K_{\underline{z}_i}^{-1} \cdot \underline{h}_i} \cdot \underline{y} \\ &= \frac{\underline{h}_i^H K_{\underline{z}_i}^{-1}}{\underline{h}_i^H \cdot K_{\underline{z}_i}^{-1} \cdot \underline{h}_i + 1} \cdot \underline{y} \end{aligned} \tag{2.24}$$

We conclude that the MMSE filter for symbol x_i is finally given by:

$$(\underline{f}_i^{MMSE})^T = \frac{\underline{h}_i^H K_{\underline{z}_i}^{-1}}{\underline{h}_i^H \cdot K_{\underline{z}_i}^{-1} \cdot \underline{h}_i + 1}. \tag{2.25}$$

This expression can be further simplified by using the matrix inversion lemma to a well-known form:

$$(\underline{f}_i^{MMSE})^T = \underline{h}_i^H \cdot (\frac{1}{SNR_{IN}} \cdot I + HH^H)^{-1} \tag{2.26}$$

The output signal to *interference and noise* ratio (SINR) for stream i can be expressed as:

$$\begin{aligned} SINR_i &= \underline{h}_i^H \cdot K_{\underline{z}_i}^{-1} \cdot \underline{h}_i \\ &= \underline{h}_i^H \cdot (\frac{1}{SNR_{IN}} \cdot I + \sum_{\substack{j=1 \\ j \neq i}}^{j=N_{TX}} \sigma_{x_j}^2 \cdot \underline{h}_j \cdot \underline{h}_j^H)^{-1} \cdot \underline{h}_i \end{aligned} \tag{2.27}$$

Studying expressions (2.25) and (2.27), we get that the output SINR can also be written as:

$$SINR_i = \frac{(\underline{f}_i^{MMSE})^T \cdot \underline{h}_i}{1 - (\underline{f}_i^{MMSE})^T \cdot \underline{h}_i} \tag{2.28}$$

By studying the extreme cases, when SNR_{IN} approaches 0 and ∞, we get that the MMSE receiver simplifies to the decorrelator and the matched filter, respectively. This confirms the statement introduced at the beginning of this section — that the MMSE represents a balance between the two extreme scenarios.

MMSE-SIC

So far we were assuming that the transmitted symbols were detected "in place", i.e. all of them at the same time. This eliminated the notion of time from all MIMO detection methods that we so far analyzed. However, the transmitted streams in general need not be detected all at the same time, and we can use the fact that some of the symbols have already been detected and *decoded*, i.e. the receiver already knows what they are.

If we allow the symbols and streams to be detected with some delay, we can use the already detected and decoded streams to cancel interference they impose to the streams that are yet to be detected. This is the logic behind the successive interference cancellation (SIC) type of MIMO receivers. SIC can be applied to any linear MIMO receiver, or even more general, to any MIMO receiver that processes streams one-by-one. In this section we focus on implementation of the SIC technique to the MMSE receiver, which has the best performance among the linear receivers that we have studied so far.

The block diagram of a MMSE-SIC MIMO receiver is presented in Figure 2.6. The first stream is detected first, in presence of interference from all the other streams. "MMSE-SIC receiver 1" block in Figure 2.6 uses the MMSE filter $\underline{f}_1^{MMSE}$ derived in the previous section and provides symbol estimates $\hat{x}_1[m]$. After enough symbols from the first stream have been detected, their estimates are passed to the channel decoder block that ensures the decoded stream is exactly what has been transmitted. During this processing of the first stream, the received symbols $\underline{y}[m]$ are being accumulated in the memory for later processing.

After the first stream has been detected and decoded, it's influence to the received symbol vector $\underline{y}$ can be cancelled, so that the detection of all the other symbols can be improved. In general, after the first $i-1$ streams have been detected and decoded, the remaining received vector $\underline{y}_{i-1}$ becomes:

$$\begin{aligned} \underline{y}_{i-1} &= \underline{y} - \sum_{j=1}^{j=i-1} \underline{h}_j \cdot x_j \\ &= \sum_{j=i}^{j=N_{TX}} \underline{h}_j \cdot x_j + \underline{z} \end{aligned} \tag{2.29}$$

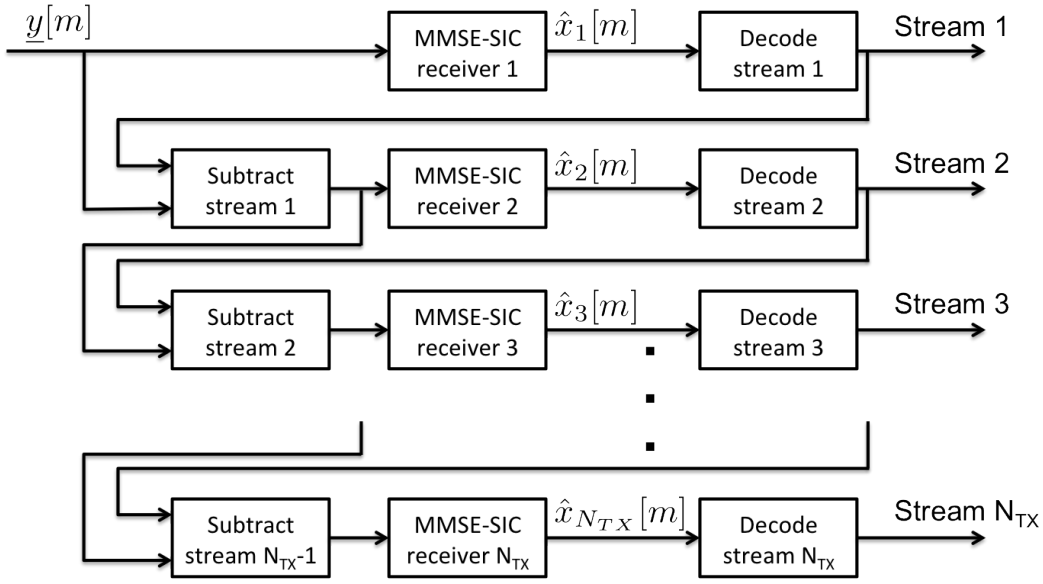

Figure 2.6: Block diagram of an MMSE-SIC MIMO receiver.

In the above equation we assume that the symbols $x_1, x_2, ..., x_{i-1}$ that are being cancelled have been perfectly decoded at the receiver. We see that the detection of the next symbol, x_i, simplifies because the amount of interference from the other streams is now less than in the case of the MMSE detector that we previously discussed. In particular, compared to the covariance matrix of the noise+interference term for estimating symbol x_i using the MMSE detector (2.20), the covariance matrix of the noise+interference term for estimation of the symbol x_i using MMSE-SIC is:

$$K_{\underline{z}_i} = \sum_{j=i+1}^{j=N_{TX}} \sigma_{x_j}^2 \cdot \underline{h}_j \cdot \underline{h}_j^H + \frac{1}{SNR_{IN}} \cdot I \tag{2.30}$$

Using the exact same derivation as in the MMSE scenario, we conclude that the MMSE-SIC filter for stream i, $\underline{f}_i^{MMSE-SIC}$, is:

$$(\underline{f}_i^{MMSE-SIC})^T = \underline{h}_i^H \cdot (\frac{1}{SNR_{IN}} \cdot I + \sum_{j=i}^{j=N_{TX}} \sigma_{x_j}^2 \cdot \underline{h}_j \cdot \underline{h}_j^H)^{-1}, \tag{2.31}$$

and that the output SINR for stream i is:

$$SINR_i = \underline{h}_i^H \cdot (\frac{1}{SNR_{IN}} \cdot I + \sum_{j=i+1}^{j=N_{TX}} \sigma_{x_j}^2 \cdot \underline{h}_j \cdot \underline{h}_j^H)^{-1} \cdot \underline{h}_i, \tag{2.32}$$

and it can be expressed in the same way as the output SINR of an MMSE filter, as:

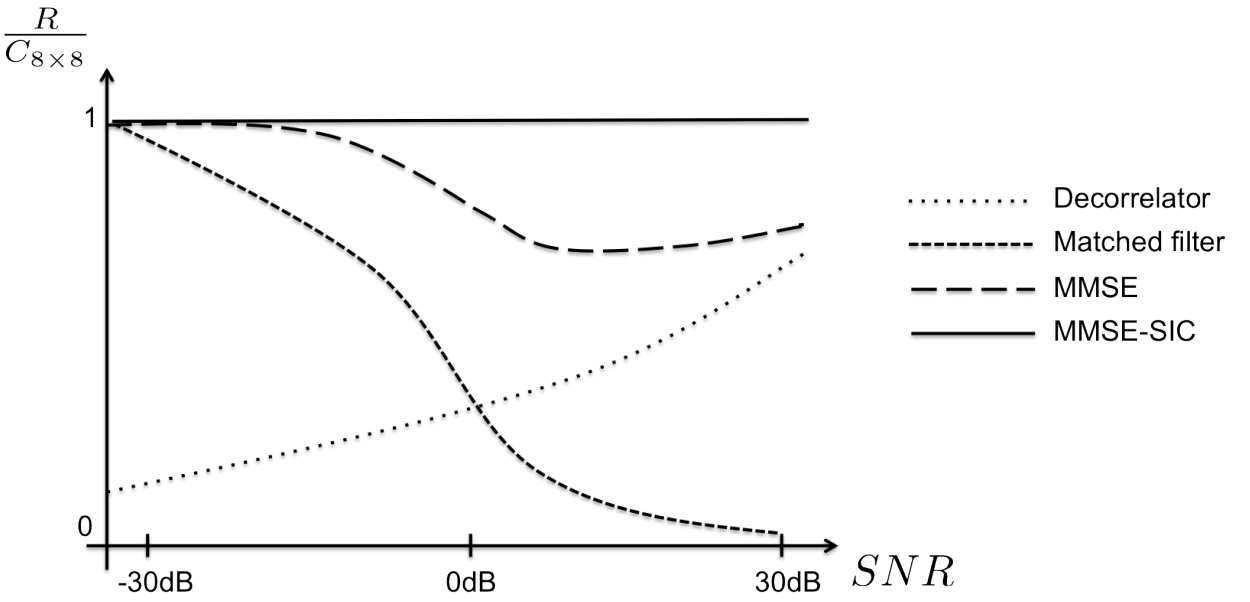

Figure 2.7: Performance of linear MIMO receivers in an 8x8 MIMO system, taken from [14]. The achievable rate with each MIMO receiver has been plotted scaled to the 8x8 MIMO channel capacity.

$$SINR_i = \frac{(\underline{f}_i^{MMSE-SIC})^T \cdot \underline{h}_i}{1 - (\underline{f}_i^{MMSE-SIC})^T \cdot \underline{h}_i} \tag{2.33}$$

Note that the per-stream SINR is strictly lower than in the MMSE scenario (except for the very first stream, when it is equal). This is because part of interference has already been cancelled, and and so the interference power has been reduced prior to detection.

Performance Comparison of the Linear MIMO Receivers

Since the amount of interference when the MMSE-SIC detection method is used is less than when the MMSE method is used, MMSE-SIC receiver performs strictly better than the MMSE, which performs strictly better than the matched filter and the decorrelator. This can be seen in Figure 2.7, taken from [14], which presents the achievable rate of each of the four linear detection schemes over an 8×8 MIMO channel. The matched filter and the MMSE receiver achieve capacity in the very low SNR regime. Decorrelator and the MMSE receiver achieve capacity in the very high SNR regime. MMSE-SIC achieves capacity across all SNR points.

2.2.2 ML MIMO Receivers

Unlike the linear MIMO receivers, the maximum-likelihood (ML) MIMO receivers deploy a very different type of processing to detect the transmitted symbol vector $\underline{x}$. Simply put, they *search* over all possible $\underline{x}$ to find the one that is the most likely to have been transmitted. To maximize the source-coding entropy, the transmitter chooses the transmit vector $\underline{x}$ from the set of equally-likely vectors $\mathcal{S}_{\underline{X}}$: $\underline{x} \in \mathcal{S}_{\underline{X}}$.

By definition, an ML estimate is the most likely choice out of the set of all possible vectors, $\mathcal{S}_{\underline{X}}$, given the observation $\underline{y}$. In other words, it is the vector $\underline{x}$ that maximizes the conditional distribution $f_{\underline{Y}|\underline{X}}(\underline{y}|\underline{x})$:

$$\hat{\underline{x}}^{ML} = \arg\max_{\underline{x}} f_{\underline{Y}|\underline{X}}(\underline{y}|\underline{x}) \tag{2.34}$$

Because the receiver noise $\underline{z}$ in MIMO channel equation (2.7) is white, we can represent the conditional distribution from expression (2.34) as Gaussian and simplify the ML detection to the following expression [54]:

$$\hat{\underline{x}}^{ML} = \arg\min_{\underline{x}} ||\underline{y} - H \cdot \underline{x}||^2 \tag{2.35}$$

To simplify the computation in expression (2.35), channel matrix H is typically represented using QR decomposition: $H = QR$, where Q is an orthonormal matrix ($Q = Q^H = Q^{-1}$) and R is an upper-triangular matrix. Now the ML estimate becomes:

$$\begin{aligned} \hat{\underline{x}}^{ML} &= \arg\min_{\underline{x}} ||Q^H \cdot \left(\underline{y} - R \cdot \underline{x}\right)||^2 \\ &= \arg\min_{\underline{x}} ||Q^H \cdot \underline{y} - R \cdot \underline{x}||^2 \end{aligned} \tag{2.36}$$

Adopting $\tilde{\underline{y}} = Q^H \cdot \underline{y}$, we can write the previous expression in an expanded form as:

$$\begin{bmatrix} \hat{x}_1^{ML} \\ \vdots \\ \hat{x}_{N_{TX}-1}^{ML} \\ \hat{x}_{N_{TX}}^{ML} \end{bmatrix} = \arg\min_{\underline{x}} \left\| \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_{N_{RX}-1} \\ \tilde{y}_{N_{RX}} \end{bmatrix} - \begin{bmatrix} r_{11} & \dots & r_{1N_{TX}-1} & r_{1N_{TX}} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & r_{N_{RX}-1N_{TX}-1} & r_{N_{RX}-1N_{TX}} \\ 0 & \dots & 0 & r_{N_{RX}N_{TX}} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_{N_{TX}-1} \\ x_{N_{TX}} \end{bmatrix} \right\|^2 \tag{2.37}$$

There are two basic approaches to solve this estimation problem: the depth-first and the breadth-first. The depth-first detector, also known as the sphere decoder, searches over multiple *vector* candidates $\underline{x}$ until it finds the most likely candidate. The breadth-first detector, also known in literature as the K-Best detector, searches for most likely transmitted symbols $\hat{x}_{N_{TX}}^{ML}, \hat{x}_{N_{TX}-1}^{ML}, \dots, \hat{x}_1^{ML}$ one by one. A good overview of the ML MIMO detection algorithms is provided in [1] and [13].

To get an intuition about these two ML algorithms, we present in Figure 2.8 a simple scenario of an estimation of three BPSK symbols: $x_1, x_2, x_3 \in \{-1, +1\}$. The depth-first

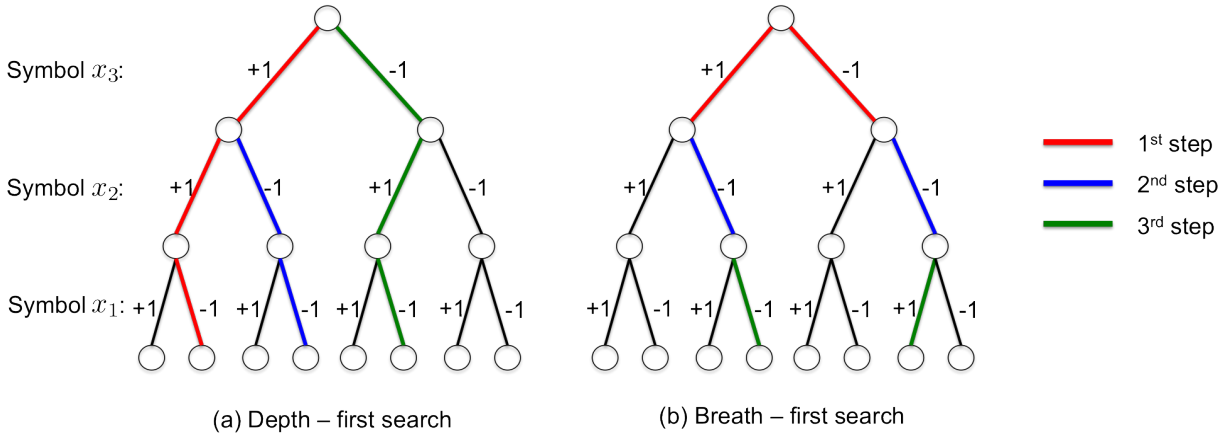

Figure 2.8: ML search algorithms for 3 transmitted BPSK symbols: (a) Depth-first search algorithm (sphere decoder), and (b) Breadth-search algorithm (K-Best detector).

algorithm (sphere decoder) goes all the way through the search tree in each step, calculating the search metric with each pass through the tree. It may not come back all the way to the top level, since some computation from the previous levels can be reused. It may also not have to go all the way down if it concludes that the search metric of the current candidate is already very bad compared to the others. But in general, this algorithm requires significant computation resources and very long delay because it goes through the whole tree. In return, it provides an ML estimate, $\underline{x}^{ML}$.

The breadth-first method (K-best detector) goes through a constant number of nodes in each step. In Figure 2.8b it is assumed that $K = 2$, and so the two candidates with the best search metrics are kept after each level of the tree-search. This means that the amount of computation is constant, and specified by the number of levels in the graph (which corresponding to the number of transmitted symbols) and the parameter K.

In the rest of this section, we explain the basics of these two approaches.

Sphere Decoder

The sphere decoder algorithm for finding the closest point in a lattice has been first introduced by Pohst in [47] and then later revised by Fincke and Pohst in [18]. The key idea is to use a parameter, typically called sphere diameter d, which becomes smaller as the search algorithm gets closer to the ML estimate. This parameter will be used to discard those candidates that are *outside* of the sphere diameter.

To measure whether a candidate is inside the search sphere, we define the search metric $s(\underline{x})$ as:

$$s(\underline{x}) = ||\underline{\tilde{y}} - R \cdot \underline{x}||^2, \tag{2.38}$$

The sphere decoding algorithm consists of the three major steps:

1. Initialize sphere diameter d to ∞, so that initially all candidates are within the sphere.
2. Choose a random candidate $\underline{x}^{(1)} \in \mathcal{S}_{\underline{x}}$ and calculate its search metric $s(\underline{x}^{(1)})$:

$$s(\underline{x}^{(1)}) = ||\underline{\tilde{y}} - R \cdot \underline{x}^{(1)}||^2 \tag{2.39}$$

3. If the candidate $\underline{x}^{(1)}$ is inside the sphere with diameter d, i.e. if $s(\underline{x}^{(1)}) < d$, then update the sphere diameter with the new value: $d = s(\underline{x}^{(1)})$. Otherwise, discard this candidate and go back to step 2 and choose another candidate. If there are no more candidates left, select the one that has its search metric equal to the resulting sphere diameter as the most-likely candidate, $\underline{\hat{x}}^{ML}$.

There is a number of flavors of this algorithm that sacrifice some performance (i.e., they may not find the ML estimate every time when run) for a much simpler signal processing. For example, many of them define a partial search metric, $s(\underline{x}_k)$:

$$s(\underline{x}_k) = \left\| \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_{N_{RX}-1} \\ \tilde{y}_{N_{RX}} \end{bmatrix} - \begin{bmatrix} r_{1k} & \dots & r_{1N_{TX}-1} & r_{1N_{TX}} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & r_{N_{RX}-1N_{TX}-1} & r_{N_{RX}-1N_{TX}} \\ 0 & \dots & 0 & r_{N_{RX}N_{TX}} \end{bmatrix} \cdot \begin{bmatrix} x_k \\ \vdots \\ x_{N_{TX}-1} \\ x_{N_{TX}} \end{bmatrix} \right\|^2 \tag{2.40}$$

By using this metric, candidates can be discarded if their partial metric becomes larger than the current sphere diameter d. In other words, if for a candidate $\underline{x}$ there is some $k > 1$ such that $s(\underline{x}_k) > d$, then this candidate can be discarded before the original search metric $s(\underline{x})$ is computed. This is based on the premise that the partial metric $s(\underline{x}_k)$ is always smaller than the original search metric, $s(\underline{x})$. While this is true in most cases, there will be scenarios when the most-likely candidate ends up being discarded before its full search metric is computed.

Note that computation of $s(\underline{x})$ does include computation of all partial metrics: $s(\underline{x}_{N_{TX}})$, $s(\underline{x}_{N_{TX}-1})$, ..., $s(\underline{x}_1) = s(\underline{x})$. This indicates that significant savings in terms of computation can be achieved by using partial search metrics. More detailed complexity analysis of a number of sphere decoding algorithms is provided in [18].

K-Best MIMO Detector

K-best algorithm has first been introduced by Schnorr and Euchner in [50]. It has been later modified in a number of ways, including an implementation friendly version presented in [22].

K-best algorithm is also based on expression (2.37), but instead of going through the vector candidate $\underline{x}$ as a whole, it goes through the scalar values of the transmitted vector $\underline{x}$

one by one. Let's assume that all transmitted symbols x_i, where $i \in \{1, 2, \ldots, N_{TX}\}$, belong to some constellation $\mathcal{X}_x = \{X_1, X_2, \ldots, X_{N_x}\}$ with N_x elements.

The K-best algorithm in general consists of the following steps:

1. Initialize the current tree-level location i to the very top level: $i = N_{TX}$. Calculate the N_x partial metrics:

$$s_j(\underline{x}_{N_{TX}}) = \left\| \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_{N_{RX}-1} \\ \tilde{y}_{N_{RX}} \end{bmatrix} - \begin{bmatrix} r_{1N_{TX}} \\ \vdots \\ r_{N_{RX}N_{TX}} \end{bmatrix} \cdot X_j \right\|^2, \tag{2.41}$$

   for all $j \in \{1, 2, \ldots, N_x\}$. Then proceed to step 3.

2. Calculate all possible partial metrics at the current level $i < N_{TX}$, given the K-best partial candidates from the level above, $i+1$. If we denote the k^{th} candidate out of the inherited set of K candidates as: $\underline{x}_{i+1}^k = [x_{i+1}^k x_{i+2}^k \ldots x_{N_{TX}}^k]^T$, then the partial metrics of all N_x paths that lead from the candidate $\underline{x}_{i+1}^k$ are:

$$s_j^k(\underline{x}_i) = \left\| \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_{N_{RX}-1} \\ \tilde{y}_{N_{RX}} \end{bmatrix} - \begin{bmatrix} r_{1i} & \ldots & r_{1N_{TX}-1} & r_{1N_{TX}} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & r_{N_{RX}-1N_{TX}-1} & r_{N_{RX}-1N_{TX}} \\ 0 & \ldots & 0 & r_{N_{RX}N_{TX}} \end{bmatrix} \cdot \begin{bmatrix} X_j \\ x_{i+1}^k \\ \vdots \\ x_{N_{TX}-1}^k \\ x_{N_{TX}}^k \end{bmatrix} \right\|^2, \tag{2.42}$$

   for all $j \in \{1, 2, \ldots, N_x\}$. Similarly we calculate the N_x partial metrics for each of the candidates $\underline{x}_{i+1}^k$, $k \in \{1, 2, \ldots, K\}$, for the total of $K \cdot N_x$ partial metrics. Note that the calculation of $s_j^k(\underline{x}_i)$ can be greatly simplified by using the precomputed partial metric $s^k(\underline{x}_{i+1})$ of the candidate $\underline{x}_{i+1}^k$ from the previous step.

3. Choose the candidates with the K smallest partial metrics and proceed to the next step.

4. Go one level down in the tree search: $i = i - 1$ and repeat steps 2-4. If $i = 1$, choose the candidate vector $\underline{x}$ with the smallest metric and declare it the most-likely estimate, $\hat{\underline{x}}^{ML}$.

There are many flavors of the K-best algorithm, but the principle is practically the same: perform a tree search starting from the top and keep the K best candidates at each level. Obviously this algorithm has a similar performance issue as the sphere decoder algorithm based on partial sums - it may not result in the ML estimate being found. The performance

of the K-best algorithm depends on the parameter K being large enough to keep the ML estimate among the candidates at each level.

This conclusion leads to a very simple trade-off between the performance and the implementation complexity. By choosing larger K, we make it more likely to end up with the correct ML estimate. Larger K also means that we need to compute more partial metrics at each level, thus investing more into computational resources. In general there is a very complex relationship among the number of transmit symbols N_{TX}, the number of symbols in a transmit constellation $\mathcal{X}_x$, N_x, the parameter K and the probability of not finding the correct ML estimate. Typically the parameter K is determined empirically, and may change when any of the parameters N_{TX} or N_x changes.

2.3 Diagonal Bell-Labs Space-Time Scheme

Another important design decision is the space-time scheme, i.e. arranging the transmissions of source and relay messages in time. As we see in this section, choice of the space-time scheme also influences the choice of a MIMO detector architecture. We compare the two standard space-time schemes, the vertical Bell-Labs space-time (VBLAST) scheme and the diagonal Bell-Labs space-time (DBLAST) scheme, for their usability in cooperative relaying.

2.3.1 DBLAST vs. VBLAST

We assume the source transmits its messages in communication frames $F_1, F_2, ...$ The most obvious choice for the relays is to listen for some time at the beginning of each frame, quickly process the received information according to the QMF scheme, and then transmit their messages to the destination during the remainder of the frame. The single-relay version of this space-time scheme is presented in the top part of Figure 2.9. It is typically referred to in the literature as the vertical Bell-Labs space-time (VBLAST) scheme, [14]. The term "vertical" comes from the fact that both the source and the relay terminals transmit messages related to the same information in the *same* frame. In Figure 2.9 this means that the same-color messages are being transmitted during the same communication frame.

Another option is to delay the transmission of the relays' messages by an integer number of frames compared to the corresponding transmission from the source. The relays listen to the source's message, process the received signal using the QMF scheme, and then delay their transmission by some number of frames. A single-relay version of this space-time scheme is presented in the bottom part of Figure 2.9. We assume that the single relay would delay its transmission by exactly one frame relative to the source's transmission. This scheme is referred to as the diagonal Bell-Labs space-time (DBLAST) scheme, [20], [14]. The term "diagonal" comes from the fact that the two cooperating terminals arrange their transmissions in the diagonal fashion in Figure 2.9, i.e. the same color messages from the source and relay terminals are being transmitted during the two consecutive communication frames.

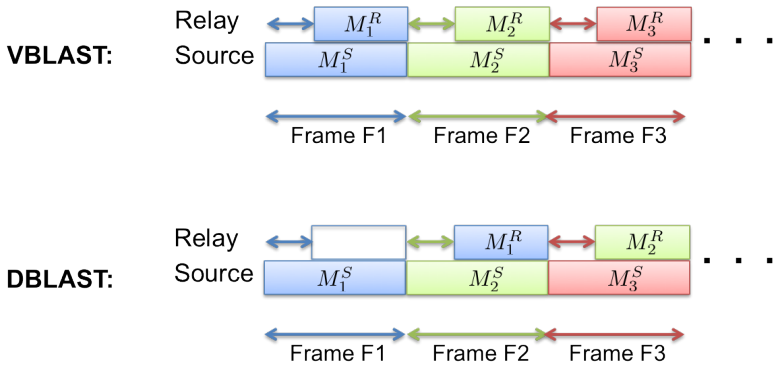

Figure 2.9: VBLAST and DBLAST space-time schemes applied on the single-relay cooperative-relaying link.

To decide on which space-time scheme to use, we analyze the advantages and disadvantages of each in the following lines.

The advantages of using VBLAST over DBLAST:

- Looking at the two diagrams in Figure 2.9, we immediately conclude that the DBLAST scheme introduces a single-frame delay compared to VBLAST. In other words, the destination needs to wait an additional communication frame to receive the relay's message. Not only does it have to wait for the relay's transmission, but it also needs to store the source's message during this one-frame delay for joint processing of the two same-color messages. This latency and memory requirement becomes more pronounced for larger number of relays, and scales linearly with the number of relays in the system. The VBLAST scheme does not require any frame memory, since all information related to the source's message is contained within the same frame. It also does not introduce multi-frame latency, like DBLAST.

- With VBLAST, the information that is being transmitted in message M_1 from the source is completely contained in frame F_1. This means that the transmission can stop after any number of frames, and continue whenever source has more data to transmit. DBLAST requires the first frame to contain only the source's transmission, which means that the last frame in the sequence would need to be designed differently from the other ones. It would either not contain transmission from the source, or have that source's transmission changed compared to the previous ones. In both scenarios, a small penalty in performance would be taken.

The advantages of using DBLAST over VBLAST:

- With VBLAST, the relays transmit their messages immediately after they receive it. That means that every relay will be scheduled to first listen for some time, and then transmit for the remaining portion of the frame. This causality condition turns out to be a huge limitation in terms of scheduling multiple relays to maximize the achievable data rate over the cooperative link. This limitation is removed with the DBLAST scheme, because the relays delay their transmissions by an integer number of frames and therefore can arrange the listening portions anywhere during the frame without the causality problem.

- With the VBLAST scheme, there is always some "dead time" due to QMF processing, when the relay is neither listening nor transmitting. This corresponds to the processing phase (2) in Figure 2.4. This loss is again removed with the DBLAST scheme. Because the relays transmit their messages with a few frame delay, they have enough time to process the received signal and the processing phase (2) effectively disappears from the time diagram.

- The DBLAST scheme arranges messages such that all messages received at the destination during one frame correspond to different original information. This can be confirmed in Figure 2.9 with different color messages being received at the destination during every frame. Having this situation is beneficial from the receiver's standpoint, because the received same-color streams are uncorrelated in space (i.e. through the MIMO channel) and therefore same-color streams are correlated only in a sense that they originate from the same (source's) message. This is not the case with VBLAST, where the streams are correlated both in space (through the MIMO channel) and time (through channel coding) domains.

- In terms of the MIMO detection at the destination, the DBLAST scheme enables linear MIMO detection with successive interference cancellation (SIC), which has been shown to achieve the MIMO capacity [14]. This is possible exactly for the reason that transmitted streams over a single communication frame are independent in space domain. The VBLAST scheme would ideally require either the joint MIMO detection and channel decoding, or the suboptimal MIMO detection, such as the MMSE detector, which treats streams as uncorrelated in space domain. Joint MIMO detection and channel decoding has been suggested in the number of publications for standard point-to-point transmission, yet it is still forbiddingly complex for practical implementations [26, 6, 30, 37].

- Lastly, one of the most important benefits of DBLAST is that it can theoretically support an arbitrary number of relays, without reducing the influence of the relays that were in the system before the new ones were added. If the SIC is done right (and it will be done right, with the good system design), all the interference that the additional relays create can be cancelled and the quality of the links from the original relays will not be compromised. This is not the case with VBLAST, unless the joint

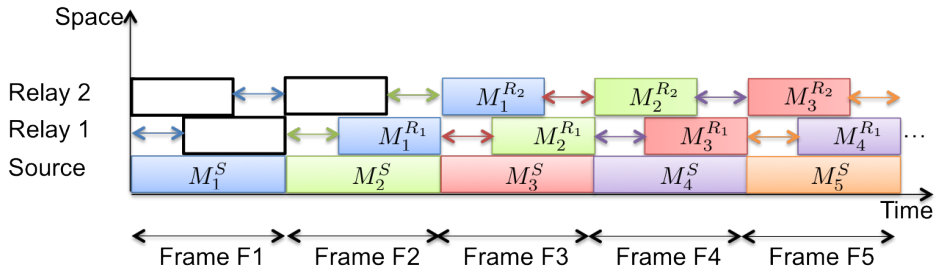

Figure 2.10: DBLAST space-time coding for a two-relay cooperative link.

MIMO detection and channel coding is performed. Any additional relays will reduce the quality of detection of *all the incoming streams*, including the source stream. This eventually may limit the number of relays that should be used with the VBLAST scheme.

Weighing the above pros and cons of the two schemes, we conclude that any application which does not have hard constraint on the transmission latency would benefit from using the DBLAST scheme over the VBLAST scheme. Because most commercial applications fall into this category, we choose to design the QMF cooperative-relaying link using the DBLAST scheme.

2.3.2 DBLAST Scheme with Multiple Relays and Linear MIMO Receiver

According to the DBLAST scheme, all relays listen according to their schedule during the source's transmission, and then delay their transmission by an integer number of frames. In particular, relay R_i listens to the source's message M_k^S during frame F_k, and transmits its message $M_k^{R_i}$ during frame F_{k+i}. The two-relay scenario is presented in Figure 2.10.

To detect all spatial streams at the destination and make use of the diagonal structure of the same-color messages, we use the MMSE-SIC MIMO detector explained in the previous section. As we have seen, the main property of any linear detection scheme is that it treats the incoming streams independently, turning the multiple-access channel at the destination into a series of parallel channels. This has been shown in expressions (2.32) and (2.27) in case of the point-to-point MIMO, in which we showed that each of the symbol estimates has its own SINR. We quickly revisit this computation in the context of the cooperative-relaying link, so we can show the benefit of using the MMSE-SIC receiver in conjunction with the DBLAST space-time scheme.

Let's assume a Gaussian network model for a cooperative link from Figure 2.1, with the channels between the terminals described as scalars $h_{T_1T_2}$ and vectors $\underline{h}_{T_1D}$, for $T_1, T_2 \in \{S, R_1, ..., R_N\}$, due to multiple antennas at the destination. The SNR values of the equiv-

alent parallel channels with the DBLAST scheme are computed using the MMSE-SIC processing as:

$$\begin{aligned} SNR_{SD} &= |\underline{h}_{SD}|^2 = \underline{h}_{SD}^H \underline{h}_{SD}; \\ SINR_{R_iD} &= \underline{h}_{R_iD}^H (I + \underline{h}_{SD}\underline{h}_{SD}^H + \sum_{k=1}^{k=i} \underline{h}_{R_kD}\underline{h}_{R_kD}^H)^{-1} \underline{h}_{R_iD}. \end{aligned} \tag{2.43}$$

As a comparison, if we use the MMSE detector to receive incoming streams arranged using the VBLAST scheme, the equivalent SNR values of the parallel channels become:

$$\begin{aligned} SNR_{SD} &= \underline{h}_{SD}^H (I + \underline{h}_{SD}\underline{h}_{SD}^H + \sum_{k=1}^{k=N} \underline{h}_{R_kD}\underline{h}_{R_kD}^H)^{-1} \underline{h}_{SD}; \\ SINR_{R_iD} &= \underline{h}_{R_iD}^H (I + \underline{h}_{SD}\underline{h}_{SD}^H + \sum_{k=1}^{k=N} \underline{h}_{R_kD}\underline{h}_{R_kD}^H)^{-1} \underline{h}_{R_iD}. \end{aligned} \tag{2.44}$$

Studying expressions above, we conclude that the SNR over all parallel channels with the VBLAST scheme and the MMSE detection reduces as we add more relays. On the other hand, increasing the number of relays does not influence the SNR of the parallel channels from the source and the other relays with the DBLAST scheme and the MMSE-SIC detection in expression (2.43). This means that by adding more relays to the network that is scheduled according to the DBLAST space-time scheme, we do not compromise the performance of those relays that were already in the network.

2.3.3 Destination Processing with DBLAST

Destination processing of each DBLAST frame consists of the following three steps:

1. MIMO detection of the source's and the relays' streams related to information from frame i (same-color messages M_i^T related to frame F_i in Figure 2.10) using the MMSE-SIC detector;
2. Joint decoding of the detected streams because they all contain information that originates from the same source's message, M_i^S;
3. Cancellation of the decoded streams from the received signal in frames $i+1, ..., i+N$. This step corresponds to the "SIC" portion of the MMSE-SIC scheme.

After the frame has been decoded, we can cancel the influence of the decoded streams to the other streams received at the same time. Figure 2.11 shows the received signal "before and after" the cancellation of a decoded message M_1, denoted in blue. If the blue messages in Figure 2.11 are decoded correctly, the received signal at the destination after the cancellation will be the same as if there was no blue streams to begin with. The processing scheme continues further to message M_2 (the green streams), then message M_3 (the red streams), etc.

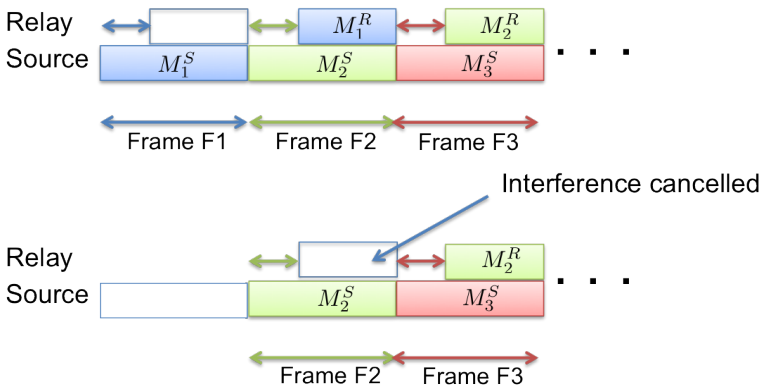

Figure 2.11: Received signal at the destination before (top) and after (bottom) detection, decoding and cancellation of the frame F_1 messages (the blue streams).

2.4 Channel Coding for QMF Relaying

Unlike with the DF and CF relaying schemes, QMF relays do not transmit messages that can be independently decoded by the destination. Because the messages from the source and the relays all contain the same information, which has been originally transmitted in the source's message M_i^S, the destination needs to decode these messages *jointly*. To optimize this process of joint decoding and ensure the maximum reliability of decoding the source's message, channel codes used at the cooperating terminals (the source and the relays) also need to be designed *jointly*.

The destination receives the following messages related to the information transmitted by the source: 1) message M_i^S from the source in frame F_i, and 2) messages $\{M_{i+1}^{R_1}, M_{i+2}^{R_2}, \ldots, M_{i+N}^{R_N}\}$ from relays $R_1, R_2, \ldots, R_N$, which arrive in frames $F_{i+1}, F_{i+2}, \ldots, F_{i+N}$. The key part of designing the channel codes to be used in these messages is to understand the relationship between relays' messages $\{M_{i+k}^{R_k}\}$ and the source's message M_i^S.

Channel coding for the QMF cooperative-relaying link has been suggested in [40, 38, 41]. In this section we summarize the main conclusions from this work related to the influence of the QMF scheme to channel coding and the joint decoding at the destination. To show the concept we use a single-relay example with AWGN channels, as it was done in [38].

2.4.1 The Influence of QMF on Channel Coding

Because we consider only the binary channel coding, we focus on the binary signaling scenario. Conclusions from this section can be later extended to the general QAM signaling using bit-interleaved coded-modulation (BICM), introduced in [7].

Let's assume that the source uses code $\mathcal{C}_S$ to encode its information bit-stream $\underline{b}_I = [b_I[1], b_I[2], \ldots, b_I[N_I]]$ into a codeword $\underline{b}_S = [b_S[1], b_S[2], \ldots, b_S[N_S]]$. These bits are modulated into BPSK symbols $\underline{x}_S = [x_S[1], x_S[2], \ldots, x_S[N_S]]$, which are transmitted to the relay

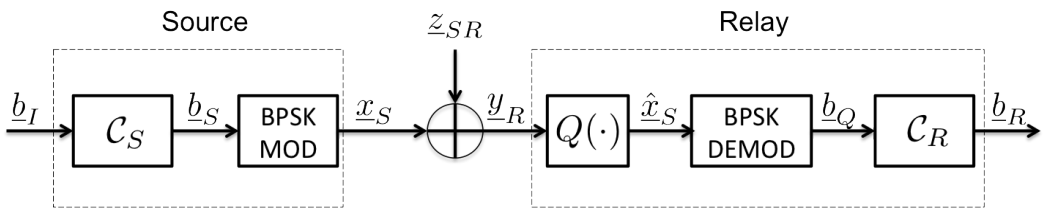

Figure 2.12: Signal processing at the source and the relay for BPSK signaling and AWGN channel.

through an AWGN channel, as shown in Figure 2.12. The received symbols at the relay R are:

$$y_R[i] = x_S[i] + z_{SR}[i], \quad \forall i \in \{1, 2, \ldots, N_f\}, \tag{2.45}$$

where $N_f = f \cdot N_S$ and f refers to the listening fraction of the relay R. For simplicity we will assume that f is a multiple of $1/N_S$ and therefore N_f is an integer.

According to the QMF scheme, the quantizer at the relay quantizes this received symbol to reject the noise, and then maps it into a new codeword. Because symbols $\underline{x}_S$ transmitted by the source are discrete and each contains only one bit (BPSK modulation), it is enough to quantize each of the received symbols $y_R[i]$ into one bit as well [38]. This means that the quantization at the relay essentially becomes BPSK symbol detection:

$$\hat{x}_S[i] = \left\{ \begin{array}{ll} +1, & y_R[i] > 0, \\ -1, & y_R[i] < 0 \end{array} \right\}, \forall i \in \{1, 2, \ldots, N_f\}. \tag{2.46}$$

The probability that the relay has detected the received symbol wrongly, i.e. that $\hat{x}_S[i] \neq x_S[i]$ is [34]:

$$p_{err} = \frac{1}{2} \operatorname{erfc}\left(\sqrt{\frac{SNR_{SR}}{2}}\right) \tag{2.47}$$

The relay demodulates detected symbols $\hat{\underline{x}}_S = [\hat{x}_S[1], \hat{x}_S[2], \ldots, \hat{x}_S[N_f]]$ into bits $\underline{b}^Q = [b_Q[1], b_Q[2], \ldots, b_Q[N_f]]$, and then maps them (or rather, encodes them) into a new codeword using its own code $\mathcal{C}_R$. Let's denote the relay's codeword as $\underline{b}_R = [b_R[1], b_R[2], \ldots, b_R[N_R]]$. These bits are then modulated into BPSK symbols $\underline{x}_R = [x_R[1], x_R[2], \ldots, x_R[N_R]]$, which are transmitted to the destination during the transmit phase of the relay. Because the relay listens to the first N_f symbols from the source, it will transmit to the destination during the remaining $N_R = N_S - N_f$ symbol slots.

Assuming both $\mathcal{C}_S$ and $\mathcal{C}_R$ are linear block codes, we present the relationship between the source and the relay codewords in Figure 2.13. The source's code $\mathcal{C}_S$ and the relay's code $\mathcal{C}_R$ are presented through deterministic relationships, which include check sums among the information bits and the coded bits.

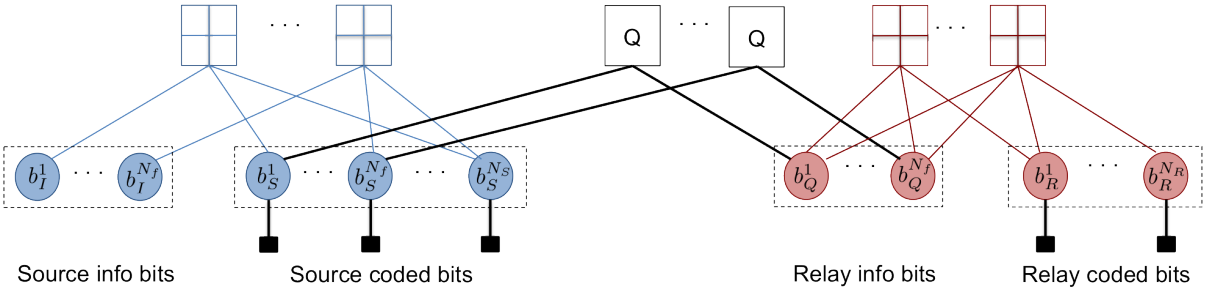

Figure 2.13: Relationship between the relay and the source codewords after the QMF processing.

Based on the probabilistic nature of the relay channel, we form the relationship between the source and the relay code through probabilistic *Q-nodes*. These nodes connect the first N_f encoded bits of the source to the N_f information bits of the relay. The connections are $1-1$, because of the scalar quantization and the BPSK modulation. As a matter of fact, Q-nodes would still connect single bits of the source and the relay codewords even if higher QAM modulation was used [38].

Unlike the deterministic check sums, which denote that the connected bits sum up to 0, the Q-nodes state that the connected bits sum up to 0 *with some probability*. Specifically, the Q-nodes in Figure 2.13 state that any two bits $b_S[i]$ and $b_Q[i]$ connected to that node should be equal (sum up to 0) with probability $1-p_{err}$, where p_{err} is calculated in (2.47).

Having these additional connections between the source and the relay codewords suggests that codes $\mathcal{C}_S$ and $\mathcal{C}_R$ should be designed jointly, taking into account the position of the Q-nodes. We have shown that using the low-density parity-check (LDPC) code at the source and the low-density generator-matrix (LDGM) code at the relay is a good choice for a single-relay scenario [38].

2.4.2 Joint Decoding with Message Passing

When the LDPC and LDGM codes are used at the source and the relay, respectively, it is advised to use the message-passing algorithm to perform joint decoding of both codewords [38]. The message-passing algorithm suggests that each node sends a message to each of the nodes it's connected to, then those nodes send the message back, and so on until successful decoding is declared. A typical form of message passing, called belief-propagation, is when the message reflects the probability of each of the receiving nodes to have the value 0 or 1 [31]. This message is typically represented through the natural logarithm of the ratio of the two probabilities, called log-likelihood ratio (LLR):

$$LLR[i] = \log \frac{P(b[i]=0)}{P(b[i]=1)} \tag{2.48}$$

The joint graph from Figure 2.13, also called the Tanner graph, contains four types of nodes:

1. The observation nodes (solid squares), which provide initialization messages based on the channel values and the received symbols.
2. The variable nodes (circles), which represent the information and the encoded bits. After the last iteration, values of the variable bits represent the outputs of the message-passing algorithm;
3. The check nodes (crossed squares), which represent deterministic relationships among the variable nodes, specified by the LDPC and LDGM generator matrices;
4. The Q-nodes, which represent probabilistic relationships among the variable nodes.

More details about each of the above categories, except the Q-nodes, can be found in [34]. Here we focus on calculating the LLR messages that a Q-node sends to the variable bits it is connected to.

A Q-node has only two connections - to a source node $b_S[i]$ and a relay node $b_Q[i]$. Let's assume that the relay node sends a message $LLR_Q[i]$ to a Q-node, where:

$$LLR_Q[i] = \log \frac{P(b_Q[i] = 0)}{P(b_Q[i] = 1)}, \tag{2.49}$$

and that the probability of error on $S - R$ channel is p_{err}. We can now compute the probabilities of the source node $b_S[i]$ taking values 0 and 1, as:

$$\begin{aligned} P(b_S[i] = 0) &= (1 - p_{err})P(b_Q[i] = 0) + p_{err}P(b_Q[i] = 1) \\ P(b_S[i] = 1) &= (1 - p_{err})P(b_Q[i] = 1) + p_{err}P(b_Q[i] = 0) \end{aligned} \tag{2.50}$$

We conclude that the Q-node message has exactly the same form as the message from a check node with three connections [34]. The message from a Q-node to the source node $b_S[i]$ can now be calculated as:

$$\tanh\left(\frac{LLR_S[i]}{2}\right) = \tanh\left(\frac{LLR_Q[i]}{2}\right) \cdot \tanh\left(\frac{LLR_{err}[i]}{2}\right), \tag{2.51}$$

where

$$LLR_{err}[i] = \log \frac{(1 - p_{err})}{p_{err}} \tag{2.52}$$

is the LLR message corresponding to the probabilistic relationship stemming from the $S-R$ channel.

Therefore, a Q-node can be conveniently represented as a check node attached to an observation node which always sends a constant message $LLR_{err}[i]$, as shown in Figure 2.14.

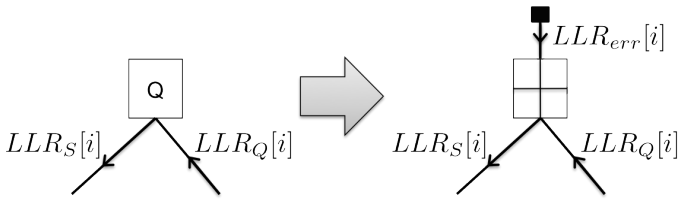

Figure 2.14: Equivalent representation of a Q-node with a check node and an observation node.

It is easy to show that the same relationship holds if the source node $b_S[i]$ is sending a message through the Q-node to the relay node $b_Q[i]$.

This powerful result means that the standard Tanner-graph decoding algorithms can be also used to design a joint LDPC-LDGM decoder for QMF cooperative relaying, because there is no special node introduced. The complexity of decoding increases compared to the source LDPC code by the number of variable and check nodes from the relay, as well as the check nodes coming from transformed Q-nodes.

2.4.3 LDPC-LDGM Code Design

The LDPC code design relies on the density evolution (DE) method for measuring average performance of chosen degree profiles for LDPC codes, developed by Richardson and Urbanke [49, 9]. DE measures the average decoding probability of a chosen degree profile for a given SNR, assuming infinite code blocklength. Once the decoding probability is sufficiently small at SNR of interest (typically SNR is chosen to be the Shannon limit, or very close to it), the same degree profile is adopted for the finite-length code blocklength. The methodology suggested in [49, 9] can be as well applied to the cooperative-relaying scenario, when the joint code design is required. This procedure has been proposed in [41, 38] for a single-relay scenario, and we readily accept that methodology in this work.

The joint LDPC-LDGM code design turns into finding a good set of degree profiles (λ_S, ρ_S) and (λ_Q, ρ_Q) for the LDPC and LDGM codes at the source and the relay, respectively. Node degree denotes the number of edges incident upon that node. λ_S and ρ_S are polynomials which represent the degree distributions of variable and check nodes at the source:

$$\begin{aligned} \lambda_S &= \sum_{i=2}^{\infty} \lambda_{S,i} x^{i-1} \\ \rho_S &= \sum_{i=2}^{\infty} \rho_{S,i} x^{i-1}, \end{aligned} \tag{2.53}$$

where $\lambda_{S,i}$ and $\rho_{S,i}$ denote the fraction of edges of the LDPC code that connect to the degree i variable node and the degree i check node, respectively. λ_Q and ρ_Q are defined in the same manner, for edges that connect variable nodes b_Q^i with the LDGM check nodes, shown in Figure 2.13. According to the equivalent representation of the Q-nodes, the equivalent check

nodes all have degrees 3 because they are connected to one variable node from the source, $b_S[i]$, one variable node from the relay, $b_Q[i]$, and a dummy observation node.

In general, code design relies on performing a search over many possible options for degree profiles. Using Gaussian approximation for DE computation proposed in [9] significantly reduces the complexity of computation, and this method is typically used in every search. Some heuristics for finding good degree profiles (λ_S, ρ_S) and (λ_Q, ρ_Q) are provided in [41, 38], but the search space is still huge and it typically takes a long time to find codes that perform close to Shannon limits.

2.5 Relay Scheduling for a Single-Relay Channel

We have previously mentioned that the relays are time-division duplexed. This brings up an immediate question of when the relays should be scheduled to operate in the receiver mode (*"listen"*) and when they should be scheduled to operate in the transmit mode (*"transmit"*). In this section, we answer that question for a single-relay link. Relay scheduling for a multi-relay link is a much more complex problem and is addressed in the following chapter.

2.5.1 The Influence of QMF on the Source-Destination Information Flow

Quantization at the relay inevitably introduces some loss in the overall information flow between the source and the destination. This loss comes from the fact that the relay does not forward the original received signal, but rather the quantized version of it. Depending on the message being received at the relay, we distinguish two scenarios in this analysis:

1. The message has Gaussian distribution. This scenario corresponds to the source transmitting a Gaussian signal to the relay.
2. The message that the source transmits is a discrete signal, in particular it belongs to a QAM constellation.

In our theoretical computations, we will exclusively use the first scenario. The results for the Gaussian signalling have been presented in other works, such as [51]. For completeness, we present the detailed analysis for both scenarios below.

Gaussian Input Signals

We assume that the relay receives the signal from the source with some Gaussian noise z:

$$y_R = x_S + z. \tag{2.54}$$

If the relay quantizes the noise at or above the noise level, the quantized signal can be expressed as:

$$\begin{aligned} Q(y_R) &= Q(x_S + z) \\ &= x_S + z + z_Q, \end{aligned} \tag{2.55}$$

where $z_Q \sim CN(0, \Delta)$ is the quantization noise, which we assume has Gaussian distribution.

The penalty for the relays that are listening ($S - R$ links) is that the effective noise on the $S - R$ channel becomes $z_Q + z$ instead of z. Equivalently, the variance of the noise has increased to $1 + \Delta$, and the effective SNR of these links has to be divided by this factor:

$$SNR_{SR,eff} = \frac{SNR_{SR}}{1 + \Delta} \tag{2.56}$$

The loss that comes from relays' transmissions ($R - D$ links) comes from the fact that the relay does not transmit the original information it receives, y_R, but rather the quantized version of it, $Q(y_R)$. The loss of information is the mutual information between these two terms:

$$\begin{aligned} I_{loss} &= I(y_R; Q(y_R)|x_S) \\ &= I(z; z + z_Q) \end{aligned} \tag{2.57}$$

This last term denotes the standard equation for Gaussian input signal $z \sim CN(0, 1)$ that is passing through a Gaussian channel with noise $z_Q \sim CN(0, \Delta)$. Therefore, the loss can be computed as:

$$\begin{aligned} I_{loss} &= \log(1 + \frac{1}{\Delta}) \\ &= \log(\frac{1 + \Delta}{\Delta}) \end{aligned} \tag{2.58}$$

With quantization at the noise level, suggested by the original work on QMF, [3], variance of the quantization noise z_Q becomes equal to the variance of the receiver noise z, i.e. $\Delta = 1$. Therefore, for the quantization at the noise level, the loss of information on the R-D link becomes:

$$I_{loss} = 1, \tag{2.59}$$

and the effective SNR on the $S - R$ link should be reduced by a factor of 2:

$$SNR_{SR,eff} = \frac{SNR_{SR}}{2} \tag{2.60}$$

QAM Input Signals

Instead of analyzing higher constellations, we use the bit-interleaved coded-modulation (BICM) technique introduced in [7]. This method enables analyzing each bit of the QAM

symbol as transmitted through an independent Gaussian channel, with channel SNR depending on the bit position within the QAM symbol.

Therefore, we focus only on the quantization of the BPSK symbol at this point, and extend the result to any QAM constellation using BICM. After the quantization of a BPSK signal, the quantized version of the received signal is:

$$\begin{aligned} Q(y_R) &= Q(x_S + z) \\ &= x_S + z + z_Q, \end{aligned} \tag{2.61}$$

where z_Q has a distribution such that:

$$z + z_Q \sim \begin{cases} 0, & -zx_S < 1, \\ -2x_S, & \text{else.} \end{cases} \tag{2.62}$$

In other words, after quantization, the channel between the transmitted symbol, x_S, and the quantized version of the received symbol, $Q(y_R)$, can be described as binary symmetric channel with error probability $p = \frac{1}{2}\operatorname{erfc}(\sqrt{\frac{SNR}{2}})$. The capacity of this link is $1 - H(p)$, therefore the incurred information loss is exactly $H(p)$ per every bit. Note that for a multi-bit QAM symbol this loss changes depending on a bit-position within the QAM symbol.

The loss that stems from relay transmitting the quantized version of a received signal instead of the true received signal can be calculated in a similar way as in (2.57). Difference is in distribution of $z + z_Q$, which is now given through (2.62). We can express the loss as:

$$\begin{aligned} I_{loss} &= I(y_R; Q(y_R)|x_S) \\ &= I(z; z + z_Q) \\ &= H(z) - H(z|z + z_Q) \\ &= H(z + z_Q) - H(z + z_Q|z) \\ &= H(p) - 0. \end{aligned} \tag{2.63}$$

Therefore, the loss due to relay's transmission is also $H(p)$ and is very small for large SNR on $S - R$ links.

2.5.2 Finding the Optimal Listening Time for a Single-Relay Cooperative Link

The cooperative-relaying link with the half-duplex relay R and the two parallel channels between the source and relay and the destination (according to the MMSE-SIC MIMO detection) is presented in Figure 2.15. We denote the SNR of the links between the source and the relay, the source and the destination and the relay and the destination as SNR_{SR}, SNR_{SD} and SNR_{RD}, respectively.

Relay scheduling in this scenario simply means finding a single scheduling parameter: listening time of the relay R, which maximizes the achievable rate between the source and the destination. To find this maximum rate, we apply the min-cut max-flow theorem based

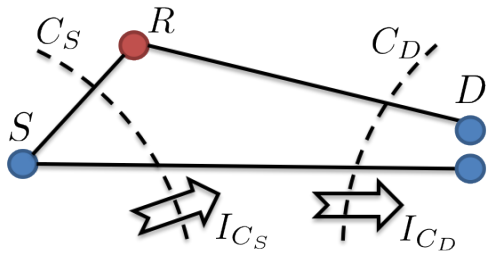

Figure 2.15: Single-relay cooperative link and its two cuts.

on the Ford-Fulkerson algorithm [19], which was also used earlier in Section 2.1.2. The network presented in Figure 2.15 has only two cuts: C_S and C_D. To maximize the minimum of the information flows through these two cuts, we need to choose the listening parameter f that equalizes the information flows through the two cuts, i.e. $I_{C_S} = I_{C_D}$.

The information flow through the cut C_S can be calculated by adding the information flow during the listening and the transmitting phases of the relay:

$$I_{C_S} = f \times \log(1 + SNR_{SD} + \frac{SNR_{SR}}{2}) + (1 - f) \times \log(1 + SNR_{SD}) \tag{2.64}$$

Similarly, we express the information flow through the cut C_D:

$$I_{C_D} = f \times \log(1 + SNR_{SD}) + (1 - f) \times [\log(1 + SNR_{SD}) + \log(1 + SNR_{RD}) - 1] \tag{2.65}$$

The effect of quantization at the relay is quantified though the losses calculated in Section 2.5.1:

- The reduction of SNR_{SR} by the factor of 2 during the listening phase of the relay is due to quantization at the noise level. This means that the receiver noise at the relays is effectively doubled.
- The "-1" term in equation (2.65) comes from the fact that the relay does not transmit the originally received signal, but rather its quantized version.

After setting the two information flows in (2.64) and (2.65) to be equal, we get that the optimal listening time for the relay R is:

$$f = \frac{\log(1 + SNR_{RD})}{\log(1 + SNR_{RD}) + [\log(1 + SNR_{SD} + \frac{SNR_{SR}}{2}) - \log(1 + SNR_{SD}) + 1]}. \tag{2.66}$$

2.6 Summary

In this chapter we introduced the basic design concepts for cooperative relaying. The QMF relaying scheme is chosen because it offers the best performance in the multi-relay scenario. We choose the DBLAST space-time scheme and the MMSE-SIC MIMO detection to separate the influence of the channel coding and the MIMO channel. The QMF scheme requires design of special channel codes, and we briefly explain the procedure used in [41] to design the LDPC-LDGM codes, which have been shown to perform very well with QMF. We also provide an introduction into the problem of relay scheduling, by analyzing the single-relay scheduling optimization.

Chapter 3

Relay Scheduling and Interference Management in a Multi-Relay QMF Cooperative Link

The main goal of this chapter is to provide an answer to the following question: *"How much can we gain by using a multi-relay cooperative link versus a direct link?"* The gain is measured through an increase in the achievable data rate between the source and the destination. Ozgur et al. have shown in [44] that the capacity of an ad-hoc network that deploys cooperative MIMO scheme increases linearly with the number of terminals. We demonstrate a similar behavior in the case of the cooperative-relaying link. The achievable data rate grows approximately linearly as we add more relays, and significant data rate gain can be achieved over a direct source-destination link.

The maximum rate of communication of a cooperative-relaying link with multiple half-duplex relays depends on the optimal listen/transmit fractions of the relays. The complexity of this relay-scheduling problem grows exponentially with the number of relays N. We show that with the suggested system design choices, the general network can be approximated with a diamond network, which further motivates our analysis of relay scheduling for a diamond network. We suggest a local scheduling algorithm, a linear-complexity alternative to an optimal relay scheduling algorithm. We prove that this simple algorithm achieves optimal performance of a diamond network, if the channel conditions enable the local schedules to be arranged such that there are no relays that listen at the same time.

Another important issue to consider in a multiple-relay cooperative system is interference between relays. Even though signals from other relays can be considered as side information, we constrain the analysis to a single-hop relaying scenario and choose to treat these signals as interference. We present a simple algorithm for inter-relay interference cancellation, which relies on the DBLAST space-time scheme. Restricted to linear encoding schemes at the source and the relays, we show that the achievable QMF rate of this algorithm applied to the general network is the same as the achievable QMF rate of the same network without relay-to-relay links. We simulate typical channel conditions with these two networks, and

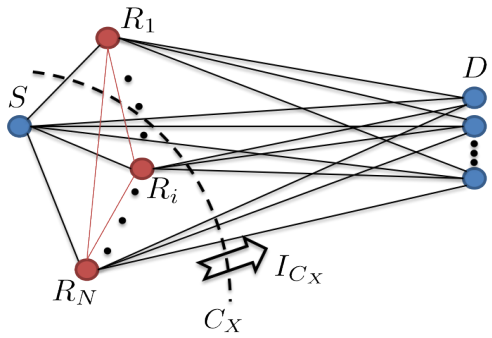

Figure 3.1: General cooperative network model.

demonstrate that the performance of the proposed algorithm is within 5% of the optimal performance for the general network.

3.1 Model Description

The cooperative MIMO link consists of a single-antenna source S, N half-duplex single-antenna relays $R_1, ..., R_N$, and a multiple-antenna destination D (Figure 3.1). Let $\mathcal{R}$ denote a set of indices of all relays in the system, i.e. $\mathcal{R} = \{1, ..., N\}$. In the most general scenario, each node can hear every other node in the network, as presented in Figure 3.1. Each bubble presents an antenna. We assume a Gaussian network model, with the channels between the terminals described as scalars $h_{T_1T_2}$ and vectors $\underline{h}_{T_1D}$, for $T_1, T_2 \in \{S, R_1, ..., R_N\}$.

We assume that communication between source and destination occurs in frames, similar to an actual communication system. Let f_i be a fraction of the frame length for which relay R_i is in the listening phase. With N relays in the system, the frame can be partitioned into 2^N time fractions as:

$$1 = \lambda_\emptyset + \sum_{i \in \mathcal{R}} \lambda_{\{i\}} + \sum_{i,j \in \mathcal{R}} \lambda_{\{i,j\}} + \ldots + \lambda_{\mathcal{R}}, \tag{3.1}$$

where $\lambda_{\mathcal{S}}$ denotes the time fraction that corresponds to the network state $\mathcal{S}$, and $\mathcal{S}$ denotes the set of indices of the relays that are in the listening phase. The scheduling problem involves finding the optimal vector Λ of these time fractions, $\Lambda = [\lambda_\emptyset, \lambda_{\{1\}}, \ldots, \lambda_{\{N\}}, \lambda_{\{1,2\}}, \ldots, \lambda_{\{1,2,3\}}, \ldots, \lambda_{\mathcal{R}}]$, under condition (3.1).

Let $\mathcal{R}_{C_X}$ and $\mathcal{L}_{C_X}$ further denote the sets of indices of relay nodes on the right and the left side of the cut C_X, respectively, such that $D \in \mathcal{R}_{C_X}$ and $S \in \mathcal{L}_{C_X}$. Given the network state $\mathcal{S}$, let $\mathcal{S}^R_{C_X}$ and $\mathcal{T}^R_{C_X}$ be the sets of indices of relay nodes on the right side of the cut C_X that are in listening and transmitting phase, respectively: $\mathcal{R}_{C_X} = \mathcal{S}^R_{C_X} \cup \mathcal{T}^R_{C_X}$. Analogously, $\mathcal{L}_{C_X} = \mathcal{S}^L_{C_X} \cup \mathcal{T}^L_{C_X}$.

The QMF achievable rate of the cooperative wireless network from Figure 3.1 is calculated in two steps. First, we identify all the cuts, $C_1, C_2, ..., C_{2^N}$, in the network. Then we calculate

the QMF information flow through each cut, I_{C_X}, and find the minimum one to be the achievable QMF rate:

$$R_{QMF} = \min_{C_X}\{I_{C_X}\} \tag{3.2}$$

It has been shown in the previous chapter that quantization at the noise level at the relays has twofold negative effect on the QMF information flow, I_{C_X}, compared to the cut-set bound: a) the SNR on $T_i - R_i$ links, where $T_i \in \{S, R_1, ..., R_N\}$, is reduced by a factor of 2; b) for each relay with index $i \in \mathcal{L}_{C_X}$, there is an information loss that equals f_i.

3.2 Relay Scheduling for the General Network

3.2.1 Slow-Fading Regime

For the purpose of the analysis in this section, we assume that the channel changes much slower than the frame duration so that the scheduling procedure can be performed for every channel instance. The optimal relay scheduling for the general network model from Figure 3.1 is found by maximizing the achievable rate:

$$\Lambda^* = \arg \max_{\substack{\Lambda \\ \sum_{\mathcal{S}} \lambda_{\mathcal{S}}=1}} \min_{C_X}\{I_{C_X}(\Lambda)\}, \tag{3.3}$$

where $I_{C_X}(\Lambda)$ is the QMF information flow through the cut C_X, given a scheduling vector Λ that satisfies condition (3.1). This optimization is done over 2^N scheduling parameters $\lambda_{\mathcal{S}}$, so its complexity grows exponentially with the number of relays N. Furthermore, the number of cuts also grows exponentially with the number of relays, which means that the complexity of the optimization function, $\min_{C_X}\{I_{C_X}\}$, also increases with the number of relays.

With a Gaussian network model, this information flow can be expressed as a summation over information flows of all possible network states $\mathcal{S} \subset \mathcal{R}$:

$$I_{C_X}(\Lambda) = \sum_{\mathcal{S}\subset\mathcal{R}} \lambda_{\mathcal{S}} \log\det(I + H_{\mathcal{S}} H_{\mathcal{S}}^H) - \sum_{i\in\mathcal{L}_{C_X}} f_i. \tag{3.4}$$

$\mathcal{S}$ represents the set of indices of all relays that are listening, i.e. $\mathcal{S} = \mathcal{S}_{C_X}^R \cup \mathcal{S}_{C_X}^L$. The logarithm term is the MIMO capacity through the cut C_X when the network is in this state. All logarithms are base 2. MIMO capacity is calculated using channel matrix $H_{\mathcal{S}}$:

$$H_{\mathcal{S}} = \begin{bmatrix} \underline{h}_{SD} & H_{\mathcal{T}_{C_X}^L D} \\ \frac{\underline{h}_{S\mathcal{S}_{C_X}^R}}{\sqrt{2}} & \frac{H_{\mathcal{T}_{C_X}^L \mathcal{S}_{C_X}^R}}{\sqrt{2}} \end{bmatrix}, \tag{3.5}$$

where $\underline{h}_{S\mathcal{S}_{C_X}^R}$ is a vector consisting of the channel coefficients between source S and relays with indices in $\mathcal{S}_{C_X}^R$, $H_{\mathcal{T}_{C_X}^L D}$ is a matrix whose columns are channel vectors between the relays

with indices in $\mathcal{T}^L_{C_X}$ and the destination D, and $H_{\mathcal{T}^L_{C_X}\mathcal{S}^R_{C_X}}$ is a matrix of channel coefficients $h_{R_iR_j}$ where $i \in \mathcal{T}^L_{C_X}$, and $j \in \mathcal{S}^R_{C_X}$. $\sqrt{2}$ factors are due to the quantization loss at the relay R_j, where $j \in \mathcal{S}^R_{C_X}$.

3.2.2 Fast-Fading Regime

The relay scheduling analysis of a slow-fading scenario assumes that channel values do not change across multiple communication frames. In practical applications, channel values may change more rapidly and adapting relay scheduling to instantaneous channel values may not be possible. Therefore, for practical purposes it is desirable to choose fixed relay scheduling given some long-term channel statistics.

If we assume that the channel distribution is known and that the channel is changing much faster than the frame duration, the optimal scheduling can be calculated using ergodic capacity:

$$\Lambda^* = \arg \max_{\substack{\Lambda \\ \sum_{\mathcal{S}} \lambda_{\mathcal{S}}=1}} \min_{C_X} \{I^{ERG}_{C_X}(\Lambda)\}, \tag{3.6}$$

where ergodic capacity can be computed using channel statistics:

$$\begin{aligned} I^{ERG}_{C_X}(\Lambda) &= E[\{I_{C_X}(\Lambda)\}] \\ &= \sum_{\mathcal{S}\subset\mathcal{R}} \lambda_{\mathcal{S}} E[\log\det(I + H_{\mathcal{S}}H^H_{\mathcal{S}})] - \sum_{i\in\mathcal{L}_{C_X}} f_i. \end{aligned} \tag{3.7}$$

To perform this optimization, the scheduler would need to have information about the channel distribution. In general, the channel distribution changes with physical conditions and it may be very difficult and costly to track it exactly. This makes the optimal scheduling scheme very difficult to implement in the fast-fading channel conditions.

3.3 Simplification of the General to the Diamond Network: Interference Cancellation Algorithm

Computation of the QMF information flow in (3.4) appears to be a very complex operation, with lots of matrix manipulations. This is caused by the following two factors: the relay-to-relay communication and the multiple-access channel at the destination.

In the remainder of this section, we focus our attention exclusively to the relay-to-relay communication and present an algorithm for cancellation of these signals. Elimination of the relay-to-relay communication links simplifies the general network to the no-interference one, presented in Figure 3.2. Applying DBLAST and MMSE-SIC detection turns the multiple-access channel at the destination into a series of equivalent parallel channels, which effectively simplifies the no-interference network to the diamond network.

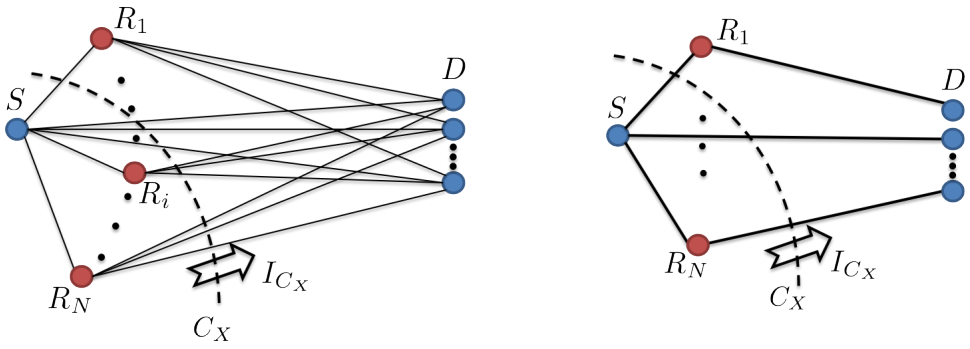

Figure 3.2: No-interference network model (left), and a diamond network model (right).

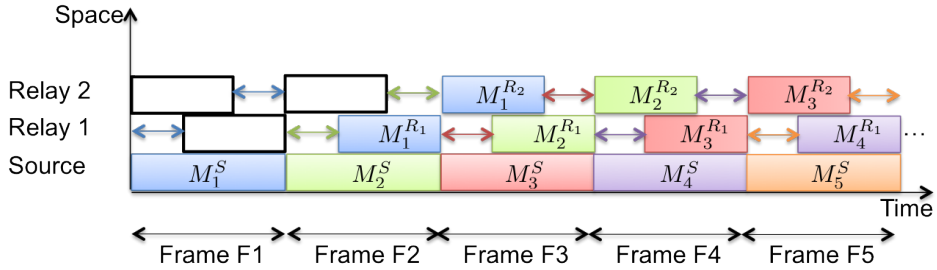

Figure 3.3: DBLAST space-time coding for a two-relay cooperative link.

3.3.1 Relay-to-Relay Communication

The QMF scheme described in Chapter 2 assumes that a relay receives only information from the source. In the general network scenario with multiple relays, a relay will receive a superposition of signals from the source and all the relays that are in the transmit phase. An easy way to picture this is to return to the DBLAST space-time diagram, repeated in Figure 3.3. On this diagram, relays R_1 and R_2 are scheduled to listen at different time during the frame, according to Theorem 1. This means that in any frame F_i, where $i > 1$, relay R_2 listens not only to the source but also to the transmission from relay R_1. Same holds for relay R_1 in frames F_j, where $j > 2$.

Estimating the source's message in the presence of (strong) signals from other relays would be detrimental for the performance of the system. In the rest of this section, we first explain the influence of the inter-relay communication, then we suggest a way to cancel it at the destination and establish the same type of probabilistic relationship between source's and relays' messages as described in Chapter 2. This enables implementation of already developed joint-decoding techniques from [38].

3.3.2 The Effect of Inter-Relay Interference

In the general network configuration, the very first frame will experience no inter-relay interference since the relays are still not transmitting (Figure 3.3). The received signal at relay R_i can be expressed as:

$$y_{R_i}^1 = h_{SR_i} x_S^1 + z, \tag{3.8}$$

or after the applied matched filter at the relay:

$$\hat{y}_{R_i}^1 = x_S^1 + \hat{z}, \tag{3.9}$$

where x_T^i denotes the transmitted symbol by terminal T in frame F_i, y_T^i denotes the received symbol at terminal T in frame F_i, and z is noise at the receiver. The quantized version of the received signal can be expressed as:

$$Q(\hat{y}_{R_i}^1) = x_S^1 + \hat{z} + z_Q, \tag{3.10}$$

where z_Q is the quantization noise. For this frame, the relationship between the original source symbol, x_S^1, and the quantized versions of this symbol at relay R_i, $Q(\hat{y}_{R_i}^1)$, is the same as it would be for a network without the relay-to-relay links. We denote this relationship between the source's message M_1^S and the relays' messages $M_1^{R_i}$ in Figure 3.4 as a "Q" relationship. This relationship changes for subsequent frames because there is interference between relays. Figure 3.4 also presents relationships between messages from frame F_3 and the previous frames, assuming the same two-relay system. The signals that relay R_i receives during frame F_3 and then forwards to the destination are respectively:

$$\begin{aligned} y_{R_i}^3 &= h_{SR_i} x_S^3 + h_{R_j R_i} x_{R_j}^{3-j} + z; \\ Q(\hat{y}_{R_i}^3) &= x_S^3 + x_{interf} + \hat{z} + z_Q. \end{aligned} \tag{3.11}$$

The relationship between x_S^3 and $Q(\hat{y}_{R_i}^3)$ is represented in Figure 3.4 as an "I" relationship. It suggests that additional information about messages M_1 and M_2 can be inferred, given some knowledge about the message M_3.

3.3.3 Using Inter-Relay Communication as Side Information

If the destination decides to use this additional information about messages M_1 and M_2 contained in message M_3, all three messages (M_1, M_2 and M_3) would have to be decoded jointly. In the general network configuration with N relays, $N+1$ messages would have to be decoded jointly. This approach would have two adverse effects on the system parameters:

1. It would additionally increase the complexity of the decoding algorithm at the destination. Instead of jointly decoding the three parts of message M_1 ($M_1^S, M_1^{R_1}$ and $M_1^{R_2}$), the decoder would need to jointly decode parts of all three messages: M_1, M_2, M_3. In

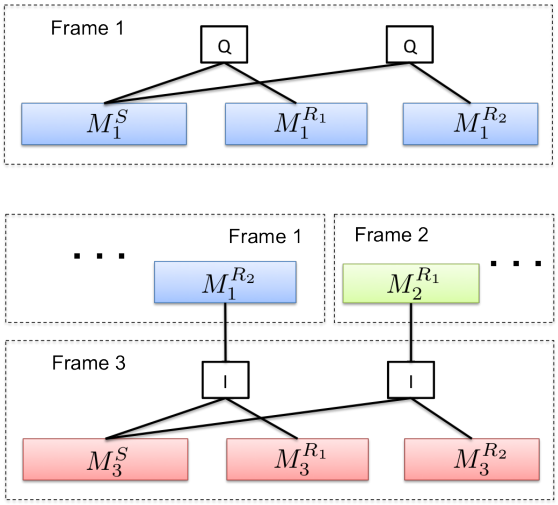

Figure 3.4: Decoding of frames 1 and 3 for the two-relay general network.

a general case with N relays, the complexity of the joint decoder would increase by a factor of $N + 1$.

2. It would lead to a very high decoding latency. By using the inter-relay communication, the destination would need to wait for all messages where that communication has propagated. In a general case with N relays, the destination would need to wait N extra frames before it can start the joint decoding.

Because of these adverse effects on a system performance, we opt not to use the inter-relay communication as side information. This leaves us with the option to treat these signals as interference and cancel them.

3.3.4 Cancellation of Inter-Relay Interference

The interference from previous messages M_1 and M_2 can be cancelled before the joint decoding of the message M_3 starts, since according to DBLAST these frames have already been decoded (Figure 3.3). In a general scenario with N relays, the term x_{interf} contains interference from N previous messages that have all been decoded. Cancellation is always possible for any linear encoding scheme at the relays, such as the one that was introduced in [38]. Effectively, (3.11) simplifies to (3.10), and relationship "I" simplifies to "Q".

After inter-relay interference cancellation is performed, the "I" relationship in any frame F_i simplifies to the "Q" relationship, as if we had no interference in the first place. When applied to the general network from Figure 3.1, this algorithm can achieve the same rate as an equivalent network without relay-to-relay links, i.e. the no-interference network from Figure 3.2. The loss incurred by the suggested interference cancellation algorithm is exactly the difference between the QMF rates of the general and the no-interference networks. The

benefit is that we maintained the same linear complexity and decoding latency as if there was no interference, and can use the channel coding and decoding techniques using linear block codes, as described in Chapter 2.

3.4 Relay Scheduling for the Diamond Network

In Section 3.3, we showed that the general network can be simplified to the diamond network by canceling inter-relay interference and using MMSE-SIC MIMO detection at the destination. In this section, we demonstrate the analysis of the relay scheduling procedure for the diamond network and show that in some channel conditions the complex optimization given by (3.3) can be simplified to analytic expressions.

3.4.1 Slow-Fading Regime

For the diamond network presented in Figure 3.2, we assume that channels between the terminals are represented through the SNR values SNR_{ST} for $T \in \{R_1, ..., R_N, D\}$ and $SINR_{TD}$ for $T \in \{R_1, ..., R_N\}$.

Let us first assume that the diamond network has only one relay, R_i. This is a scenario that has been analyzed in the previous chapter, where we found that the optimal listening time of the relay R_i is:

$$f_i = \frac{\log(1 + SINR_{R_iD})}{\log(1 + SINR_{R_iD}) + [\log(1 + SNR_{SD} + \frac{SNR_{SR_i}}{2}) - \log(1 + SNR_{SD}) + 1]} \tag{3.12}$$

For the diamond network with N relays, the complexity of optimization is still exponential with the number of relays. Even though [5] suggests that the number of non-zero scheduling parameters is only $N + 1$, there is no systematic way to identify these parameters and calculate the schedules but to perform an optimization over all 2^N parameters. This is not practical for three reasons: 1) the computation may be intense and possibly infeasible in real-time for a large number of relays and fast-fading channels, 2) there needs to be a node that collects all channel information in the network, performs the optimization (3.3) and sends the schedules to all the relays in the network, which may induce huge communication overhead, and 3) an optimal scheduling may require some relays to switch between Rx/Tx mode multiple times per frame. Since each switching instance has some time loss associated with it, it may contribute to a non-negligible loss, depending on the frame duration. It is more practical to make this switch happen only once per frame. Before we propose an actual scheme that satisfies these conditions, we prove the following theorem.

Theorem 1: If the listening time of each relay, f_i^*, is chosen according to (3.12), and the condition:

$$f_1^* + f_2^* + \ldots + f_N^* \leq 1 \tag{3.13}$$

holds, then an optimal schedule is given by:

$$\begin{aligned} \lambda_{\{i\}}^* &= f_i^*, \forall i \in \{1, 2, ..., N\} \\ \lambda_{\emptyset}^* &= 1 - \sum_{i=1}^{i=N} f_i^* \\ \lambda_{\mathcal{S}}^* &= 0, \forall \mathcal{S}, |\mathcal{S}| \geq 2. \end{aligned} \tag{3.14}$$

Proof. For the diamond network with N relays, the information flow through an arbitrary cut C_X can be calculated by adding terms that correspond to each of the 2^N network states:

$$\begin{aligned} I_{C_X}(\Lambda) = &\sum_{\substack{\mathcal{S}_{C_X}^R \subset \mathcal{R}_{C_X} \\ |\mathcal{S}_{C_X}^R| \geq 1}} \sum_{\mathcal{S}_{C_X}^L \subset \mathcal{L}_{C_X}} \lambda_{\mathcal{S}_{C_X}^R \cup \mathcal{S}_{C_X}^L} \log(1 + SNR_{SD} + \sum_{i \in \mathcal{S}_{C_X}^R} \frac{SNR_{SR_i}}{2}) \\ &+ \sum_{\mathcal{S}_{C_X}^L \subset \mathcal{L}_{C_X}} \lambda_{\mathcal{S}_{C_X}^L} \log(1 + SNR_{SD}) \\ &+ \sum_{i \in \mathcal{L}_{C_X}} [(1 - f_i) \log(1 + SINR_{R_iD}) - f_i] \end{aligned} \tag{3.15}$$

In the above expression, the first term corresponds to the information flow of the broadcast channel, when there is at least one relay (in addition to the destination D) on the right side of the cut C_X that is listening. The summation goes over all network states that satisfy this broadcast scenario, relative to the cut C_X. The second term denotes the information flow through the direct link, $S - D$, when there are no relays on the right side of the cut C_X that are listening. The last two terms correspond to the information flow through the relay-to-destination $(R - D)$ links. Notice that in both the first and the last term we accounted for the information loss due to quantization at the relays, according to the analysis provided in Section 2.5.

It is convenient to rewrite (3.15) by expressing f_i as a sum of individual scheduling parameters:

$$f_i = \lambda_{\{i\}} + \sum_{j \in \mathcal{R}} \lambda_{\{i,j\}} + \sum_{j,k \in \mathcal{R}} \lambda_{\{i,j,k\}} + \ldots + \lambda_{\mathcal{R}}. \tag{3.16}$$

We can rewrite the first two terms in expression (3.15) using the listening times of the relays on the right side of the cut C_X as:

$$
\begin{aligned}
I_{C_X}(\Lambda) = & \sum_{i \in \mathcal{R}_{C_X}} f_i [\log(1 + SNR_{SD} + \frac{SNR_{SR_i}}{2}) - \log(1 + SNR_{SD})] \\
& + \log(1 + SNR_{SD}) \\
& + \sum_{i \in \mathcal{L}_{C_X}} (1 - f_i) \log(1 + SINR_{R_i D}) - \sum_{i \in \mathcal{L}_{C_X}} f_i \\
& - \sum_{\substack{\mathcal{S}^R_{C_X} \subset \mathcal{R}_{C_X} \\ |\mathcal{S}^R_{C_X}| \geq 2}} \sum_{\mathcal{S}^L_{C_X} \subset \mathcal{L}_{C_X}} \lambda_{\mathcal{S}^R_{C_X} \cup \mathcal{S}^L_{C_X}} \times \Delta_{\mathcal{S}^R_{C_X}},
\end{aligned}
\tag{3.17}
$$

where

$$
\begin{aligned}
\Delta_{\mathcal{S}^R_{C_X}} = & \sum_{i \in \mathcal{S}^R_{C_X}} \log(1 + SNR_{SD} + \frac{SNR_{SR_i}}{2}) \\
& - \log(1 + SNR_{SD} + \sum_{i \in \mathcal{S}^R_{C_X}} \frac{SNR_{SR_i}}{2}) \\
& - (|\mathcal{S}^R_{C_X}| - 1) \log(1 + SNR_{SD}).
\end{aligned}
\tag{3.18}
$$

Notice that in expression (3.17) the first four terms depend only on the total listening fraction of each of the relays, f_i. The last term contains the contribution to the information flow of the scenario when there are two or more relays on the right side of the cut C_X that are in the listening phase.

It is easy to show that the term $\Delta_{\mathcal{S}^R_{C_X}}$ is strictly non-negative for $|\mathcal{S}^R_{C_X}| \geq 2$ and positive values of SNRs because $\prod_{i=1}^{k}(x + y_i) > (x + \sum_{i=1}^{k} y_i)x^{k-1}$, where $x = 1 + SNR_SD$ and $y_i = SNR_{SR_i}/2$. From (3.17) we conclude that given listening fractions f_i for each relay R_i, *overlaps* in listening intervals ($\lambda_{\{i,j\}}$, $\lambda_{\{i,j,k\}}$ etc.) always *decrease* the information flow through an arbitrary cut C_X. Furthermore, since listening fractions f_i^* are chosen according to (3.12), the sum of the first four terms in (3.17) is the same across all cuts. The fifth term is the penalty term if more than one relay is listening at the same time.

Next we show that the scheduling given by (3.14) is optimal. With this scheduling, QMF information flows through all cuts are equal to an optimal achievable QMF rate, R^*_{QMF}. If we keep the same listening fractions f_i^* but introduce overlaps, at least one cut will have its information flow reduced compared to R^*_{QMF}, and the rate of the network will be reduced. If some listening fractions f_i differ from those given by (3.12), we can identify at least one cut C_Y, such that $I_{C_Y} < R^*_{QMF}$. With the new scheduling vector Λ having $f_i > f_i^*$ where $i \in \mathcal{A}$ and $f_j < f_j^*$ where $j \in \mathcal{B}$, and $\mathcal{A} \cup \mathcal{B} \neq \emptyset$, any cut C_Y that satisfies $\mathcal{A} \subset \mathcal{L}_{C_X}$ and $\mathcal{B} \subset \mathcal{R}_{C_X}$ will have information flow $I_{C_Y} < R^*_{QMF}$. Any overlaps would further reduce the information flow through this cut.

We conclude that if the schedule given by (3.14) is feasible, it is optimal. Note that the number of non-zero scheduling parameters does not exceed $N+1$, as predicted by [5]. □

Corrolary 1: The analogous theorem can be stated for optimizing the cut-set bound (instead of the QMF rate).

Theorem 2: If the listening time of each relay, f_i^*, is chosen as

$$f_i^* = \frac{\log(1+SINR_{R_iD})}{\log(1+SINR_{R_iD}) + [\log(1+SNR_{SD}+SNR_{SR_i}) - \log(1+SNR_{SD})]}, \tag{3.19}$$

and the condition:

$$f_1^* + f_2^* + \ldots + f_N^* \leq 1 \tag{3.20}$$

holds, then the schedule that maximizes the cut-set bound is given by:

$$\begin{aligned} \lambda_{\{i\}}^* &= f_i^*, \forall i \in \{1,2,...,N\} \\ \lambda_{\emptyset}^* &= 1 - \sum_{i=1}^{i=N} f_i^* \\ \lambda_{\mathcal{S}}^* &= 0, \forall \mathcal{S}, |\mathcal{S}| \geq 2. \end{aligned} \tag{3.21}$$

Proof. Proof is equivalent to the proof of Theorem 1. The only difference is that the quantization loss at the relays should be removed from the information flow expressions (3.15) and (3.17). The same conclusion still holds: to maximize the cut-set bound of a multi-relay diamond network, relays should be scheduled such that overlaps in listening intervals are avoided. □

3.4.2 Fast-Fading Regime

Just like in the case of the general network, if we assume that the channel distribution is known and that the channel is changing much faster than the frame duration, the optimal scheduling can be calculated using the ergodic capacity:

$$\Lambda^* = \arg \max_{\substack{\Lambda \\ \sum_{\mathcal{S}} \lambda_{\mathcal{S}}=1}} \min_{C_X} I_{C_X}^{ERG}(\Lambda), \tag{3.22}$$

where the ergodic information flow is computed using expression (3.15), and the fact that the operator $E[\cdot]$ is linear:

$$\begin{aligned} I_{C_X}(\Lambda) = & \sum_{\substack{\mathcal{S}^R_{C_X} \subset \mathcal{R}_{C_X} \\ |\mathcal{S}^R_{C_X}| \geq 1}} \sum_{\mathcal{S}^L_{C_X} \subset \mathcal{L}_{C_X}} \lambda_{\mathcal{S}^R_{C_X} \cup \mathcal{S}^L_{C_X}} E[\log(1 + SNR_{SD} + \sum_{i \in \mathcal{S}^R_{C_X}} \frac{SNR_{SR_i}}{2})] \\ & + \sum_{\mathcal{S}^L_{C_X} \subset \mathcal{L}_{C_X}} \lambda_{\mathcal{S}^L_{C_X}} E[\log(1 + SNR_{SD})] \\ & + \sum_{i \in \mathcal{L}_{C_X}} (1 - f_i) E[\log(1 + SINR_{R_iD})] - \sum_{i \in \mathcal{L}_{C_X}} f_i \end{aligned} \tag{3.23}$$

To compute the optimal schedules, the scheduler would need to know the distribution of all the SNR parameters of the network. Because this distribution depends on the distribution of the channel coefficients, we reach the similar conclusion as in the fast-fading analysis of the general network - it would be very difficult to perform this optimization in a practical communication system.

It is interesting to see, however, that a theorem similar to the one that we proved in the previous section holds for the fast-fading channel condition as well:

Theorem 3: If the listening time of each relay, f_i^*, is chosen as:

$$f_i^* = \frac{E[\log(1 + SINR_{R_iD})]}{E[\log(1 + SINR_{R_iD})] + E[\log(1 + SNR_{SD} + \frac{SNR_{SR_i}}{2})] - E[\log(1 + SNR_{SD})] + 1]} \tag{3.24}$$

and the condition:

$$f_1^* + f_2^* + \ldots + f_N^* \leq 1 \tag{3.25}$$

holds, then an optimal schedule for the diamond network and the fast-fading channel conditions is given by:

$$\begin{aligned} \lambda^*_{\{i\}} &= f_i^*, \forall i \in \{1, 2, ..., N\} \\ \lambda^*_{\emptyset} &= 1 - \sum_{i=1}^{i=N} f_i^* \\ \lambda^*_{\mathcal{S}} &= 0, \forall \mathcal{S}, |\mathcal{S}| \geq 2. \end{aligned} \tag{3.26}$$

Proof. The proof of this theorem is similar to the proof of Theorem 1. We are using the fact that the operator $E[\cdot]$ is linear and that the relationships between the scheduling parameters in the information flow expressions are also linear.

We can rewrite expression (3.23) to contain the terms that depend only on the listening times of the relays, f_i's, and one term that reflects the influence of having two or more relays on the right side of the cut C_X that are listening at the same time:

$$
\begin{aligned}
I_{C_X}(\Lambda) = & \sum_{i \in \mathcal{R}_{C_X}} f_i [E[\log(1 + SNR_{SD} + \frac{SNR_{SR_i}}{2})] - E[\log(1 + SNR_{SD})]] \\
& + E[\log(1 + SNR_{SD})] \\
& + \sum_{i \in \mathcal{L}_{C_X}} (1 - f_i) E[\log(1 + SINR_{R_i D})] - \sum_{i \in \mathcal{L}_{C_X}} f_i \\
& - \sum_{\substack{\mathcal{S}^R_{C_X} \subset \mathcal{R}_{C_X} \\ |\mathcal{S}^R_{C_X}| \geq 2}} \sum_{\mathcal{S}^L_{C_X} \subset \mathcal{L}_{C_X}} \lambda_{\mathcal{S}^R_{C_X} \cup \mathcal{S}^L_{C_X}} \times \Delta^{FF}_{\mathcal{S}^R_{C_X}},
\end{aligned}
\tag{3.27}
$$

where

$$
\begin{aligned}
\Delta^{FF}_{\mathcal{S}^R_{C_X}} = & \sum_{i \in \mathcal{S}^R_{C_X}} E[\log(1 + SNR_{SD} + \frac{SNR_{SR_i}}{2})] \\
& - E[\log(1 + SNR_{SD} + \sum_{i \in \mathcal{S}^R_{C_X}} \frac{SNR_{SR_i}}{2})] \\
& - (|\mathcal{S}^R_{C_X}| - 1) E[\log(1 + SNR_{SD})].
\end{aligned}
\tag{3.28}
$$

The term $\Delta^{FF}_{\mathcal{S}^R_{C_X}}$ is strictly non-negative, because $\Delta^{FF}_{\mathcal{S}^R_{C_X}} = E[\Delta_{\mathcal{S}^R_{C_X}}]$ and the term $\Delta_{\mathcal{S}^R_{C_X}}$ has been shown in the proof of Theorem 1 to be non-negative. This means that yet again the overlaps in listening intervals (i.e. positive $\lambda_{\{i,j\}}$, $\lambda_{\{i,j,k\}}$ etc.) *decrease* the information flow through an arbitrary cut C_X. Also, if the listening fractions f_i^* are chosen according to (3.24), the sum of the first four terms in (3.27) is the same across all cuts.

The optimality of the schedule given by (3.26) is proven in exactly the same manner as in Theorem 1. We conclude that if the condition (3.25) is satisfied, the schedule given by (3.26) is optimal. □

3.5 Local Scheduling Scheme

In the previous sections we analyzed the optimal relay scheduling strategy for the general and the diamond networks. We concluded that the complexity of this optimization grows exponentially with the number of relays, and that knowing channel distributions in case of the fast-fading channel conditions may be impossible. In this section we present a simple sub-optimal scheme, the local scheduling scheme, which takes care of both issues and can be applied both to the slow-fading and the fast-fading channel conditions.

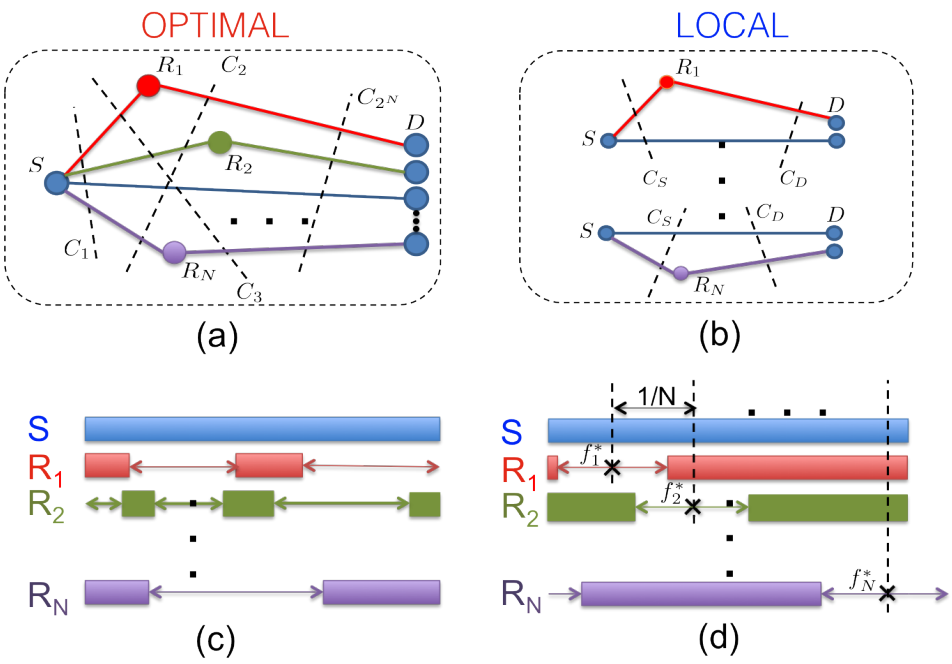

Figure 3.5: Intuition behind the optimal (a) and the local scheduling scheme (b), and an illustration of the relay schedules for the optimal (c) and the local scheduling scheme (d).

3.5.1 Local Scheduling Scheme Based on the Instantaneous Channel Knowledge

Theorem 1 leads to a very simple scheduling scheme that is based on the idea that each relay calculates the total listening fraction f_i as if there were no other relays in the network. Intuitively, in terms of the relay scheduling calculation this is equivalent to splitting the N-relay diamond network to N single-relay sub-networks, as presented in Figure 3.5b.

If each relay R_i thinks that it is the only relay in the network, it will use only the channel state information (CSI) of the surrounding links: $S-D$, $S-R_i$ and R_i-D (thus the name *local scheduling*), to calculate its listening time f_i. This calculation is shown in Section 2.5 and the expression for f_i given by (3.12). We have shown that under condition (3.13), these locally calculated listening fractions represent the optimal solution if the relays are scheduled such that there are no overlaps. If condition (3.13) is not satisfied, there may be some overlaps in listening between the relays, and in general the local scheduling scheme is no longer optimal.

Based on this idea, we propose the following algorithm for the local scheduling scheme:

The local scheduling algorithm based on the instantaneous channel values:

1. For the general and the no-interference network, SNR values needed for f_i computation

are calculated using expressions given by (2.43) from the previous chapter. SNR values for a diamond network are explicitly specified;

2. Listening time of the relay R_i, f_i, is calculated using expression (3.12);

3. If condition (3.13) is satisfied, then go to step 4), otherwise go to step 5);

4. The relays are scheduled such that there are no overlaps in listening: relay R_1 starts listening at the beginning of the frame and listens for duration of f_1, relay R_2 starts listening immediately after R_1 finishes, and so on. Relay R_i is scheduled to listen in interval $[\sum_{j=1}^{j=i-1} f_j, \sum_{j=1}^{j=i} f_j]$;

5. Relay R_i is scheduled to listen in interval $[\frac{2i-1}{2N} - \frac{f_i}{2}, \frac{2i-1}{2N} + \frac{f_i}{2}]$, based on its index number i.

Example results of the optimal and the local scheduling schemes are presented in Figures 3.5c and 3.5d, respectively. In Figure 3.5d, the local schedules are presented as if the above algorithm executed step 5, i.e. as if the condition (3.13) was not satisfied.

3.5.2 Local Scheduling Scheme Based on Known Channel Distributions

The local scheduling scheme presented in the previous section is suitable only for slow-fading channel conditions, when the SNR parameters have the same values over multiple communication frames. In the fast-fading channel conditions, it is not possible to schedule the relays based on the instantaneous channel values. Therefore, we must select a static schedule, which will be applied over a number of fast-changing channel instances. In this section we assume that static schedule is selected using a local scheduling algorithm and a known channel statistics.

The same intuition is still applied — the relays calculate their listening times as if there were no other relays in the network. They do so by using expression (3.24), which can be computed given the channel distributions. If the relays can be scheduled such that there are no overlaps, according to Theorem 3 this scheduling scheme is optimal. Otherwise, if there are overlaps, the scheme is suboptimal. The following algorithm presents the local scheduling algorithm for fast-fading channel conditions given the channel distribution.

The local scheduling algorithm based on known channel distribution:

1. For the general and the no-interference network, distributions of SNR values needed for f_i computation are calculated using expressions given by (2.43) from the previous chapter and the known channel distributions. SNR distributions for a diamond network are explicitly specified;

2. Listening time of the relay R_i, f_i, is calculated using expression (3.24);

3. If condition (3.25) is satisfied, then go to step 4), otherwise go to step 5);

4. Relay R_i is scheduled to listen in interval $[\sum_{j=1}^{j=i-1} f_j, \sum_{j=1}^{j=i} f_j]$;

5. Relay R_i is scheduled to listen in interval $[\frac{2i-1}{2N} - \frac{f_i}{2}, \frac{2i-1}{2N} + \frac{f_i}{2}]$, based on its index number i.

Except for the first two steps, which include calculating the distributions of SNR values and the listening times of the relays, this algorithm is equivalent to the local scheduling algorithm from Section 3.5.1.

3.5.3 Local Scheduling Scheme Based on the Average Channel Knowledge

We concluded in Sections 3.2.2 and 3.4.2 that in practical communication systems channel distributions are not known. Therefore, performing an optimization described in (3.6) or using the local scheduling algorithm with the given channel distributions would not be possible. Typically what is available are the long-term SNR estimates that depend on shadowing effects and change at the rate of seconds. We can use these long-term SNR values (SNR^{AVG}) in the local scheduling scheme described previously for slow-fading channels to apply it in the fast-fading channel conditions scenario with the unknown channel distributions.

Therefore, the proposed heuristics for relay scheduling in fast-fading channel conditions is the same as the local scheduling scheme described in Section 3.5.1. The only difference is in the first two steps, where we use the long-term SNR values instead of the instantaneous ones:

$$\begin{aligned} SNR_{SR_i}^{AVG} &= E[|h_{SR_i}|^2]; \\ SNR_{SD}^{AVG} &= E[|\underline{h}_{SD}|^2] = E[\underline{h}_{SD}^H \underline{h}_{SD}]; \\ SINR_{R_iD}^{AVG} &= E[\underline{h}_{R_iD}^H (I + \underline{h}_{SD}\underline{h}_{SD}^H + \\ &\quad + \sum_{k=1}^{k=i} \underline{h}_{R_kD}\underline{h}_{R_kD}^H)^{-1} \underline{h}_{R_iD}]. \end{aligned} \tag{3.29}$$

The local scheduling time is now calculated as in (3.12), but using the average SNR values instead of the instantaneous ones:

$$f_i = \frac{\log(1 + SINR_{R_iD}^{AVG})}{\log(1 + SINR_{R_iD}^{AVG}) + [\log(1 + SNR_{SD}^{AVG} + \frac{SNR_{SR_i}^{AVG}}{2}) - \log(1 + SNR_{SD}^{AVG}) + 1]} \tag{3.30}$$

The local scheduling algorithm based on the average SNR values:

1. For the general and the no-interference network, average SNR values needed for f_i computation are measured using expressions given by (3.29). Average SNR values for a diamond network are measured as $SNR_{XY}^{AVG} = E[SNR_{XY}]$;

2. Listening time of the relay R_i, f_i, is calculated using expression (3.30);

3. If condition (3.13) is satisfied, then go to step 4), otherwise go to step 5);

4. Relay R_i is scheduled to listen in interval $[\sum_{j=1}^{j=i-1} f_j, \sum_{j=1}^{j=i} f_j]$;

5. Relay R_i is scheduled to listen in interval $[\frac{2i-1}{2N} - \frac{f_i}{2}, \frac{2i-1}{2N} + \frac{f_i}{2}]$, based on its index number i.

We notice that the three local scheduling algorithms presented in this section differ only in how we compute the listening times, f_i. The best method to compute f_i parameters is to use expression (3.12), based on the instantaneous channel values. If the channel is changing faster than the frame duration and there is a good way to estimate the channel distributions, the best method is to use expression (3.24). In case the channel is fast-fading and the channel distributions are unknown, we can use expression (3.30) and the measured average SNR values to calculate the listening time of each relay.

3.5.4 The Benefits of Using the Local Scheduling Scheme

The performance of the local scheduling scheme is suboptimal, unless the condition (3.13) holds. Exactly how much of the performance is lost depends on the channel conditions. Leaving the performance difference aside for now, we can state the following benefits of using the local scheduling scheme over the optimal one:

1. There is no need for a complex optimization required to find the optimal schedules. The local scheduling scheme does not require any optimization, as the listening time f_i can be analytically computed by applying expression (3.12), (3.24) or (3.30) to all N relays. Starting time for each relay can also be analytically computed according to steps 4 or 5 of the above algorithm.

2. There is no need for a global scheduler. The reason why the local scheme is called *local*, is because relay R_i can compute its scheduling parameters using only the local CSI knowledge: $S - R_i$, $R_i - D$ and $S - D$. The amount of overhead used for transmission of CSI and scheduling parameters is minimal, because the local scheme is very-well structured (bottom right part of Figure 3.5). As a comparison, in the optimal scheme the CSI of the whole network needs to be forwarded to a single node (the scheduler) and then the scheduling parameters sent back to the relays.

3. Each relay switches between the listening and the transmitting mode only once during the frame. This is a desired behavior for implementation, due to the time loss that

occurs when the transceiver switches between the receive and the transmit modes. This is not the case with the optimal scheme, where in general relays may be required to switch between listening and transmitting multiple times during the frame, as presented in the bottom left part of Figure 3.5.

As we see in the next chapter, using the local scheduling scheme makes the multi-relay cooperation much easier for implementation. There we show on an example that the performance of the local scheme is less sensitive to calculating the correct listening fractions f_i, as long as the overlaps are avoided according to steps 4 and 5 of the local scheduling algorithm.

3.6 Evaluation of Relay-Scheduling Schemes

In this section we provide performance evaluation of the two relay-scheduling schemes, the optimal and the local, for different network configurations and channel values. We plot separately results for the slow and fast fading channel conditions. Unless otherwise noted, we plot results for maximum of $N = 5$ relays and the setup explained in Section 4.1, with the $N + 1$-antenna destination, so it can ideally receive all the incoming streams (from the source and all N relays). In the second part of this section, we show that a good data rate scaling can be achieved even if the destination has less than $N+1$ receive antennas, i.e. even if it cannot receive all incoming streams with full multiplexing gain.

To evaluate performance of the proposed relaying schemes, we perform Monte-Carlo simulations. All channel coefficients are assumed to have Rayleigh distribution with $\underline{h}_{T_1D} \sim \mathcal{CN}(0, SNR_D \times I)$ and $h_{T_1T_2} \sim \mathcal{CN}(0, SNR_R)$ where $SNR_R = SNR_D^{\eta}$, and the noise at each receive antenna is $z \sim \mathcal{CN}(0, 1)$. The proximity gain η comes from the assumption that relays are located much closer to the source and to each other than to the destination.

3.6.1 Comparison Between Optimal and Local Scheduling Schemes

Slow Fading

In the slow-fading scenario, for every channel instance we applied both the optimal and the local scheduling scheme to both the general and the no-interference networks, then averaged computed QMF rates over many channel instances. This way we get the average performance in slow-fading channel conditions, presented in Figure 3.6.

Based on Figure 3.6, we capture three important conclusions:

1. Achievable QMF rate of a 5-relay system compared to a system without relays is approximately $2\times$ higher for $\eta = 2$ and $3\times$ higher for $\eta = 4$. This means that significant data rate scaling can be achieved with the QMF cooperative-relaying link. Furthermore, we notice that data rate scales *linearly* with the number of relays, for smaller number of relays. This result resembles that one of the ad-hoc network from [44]. It

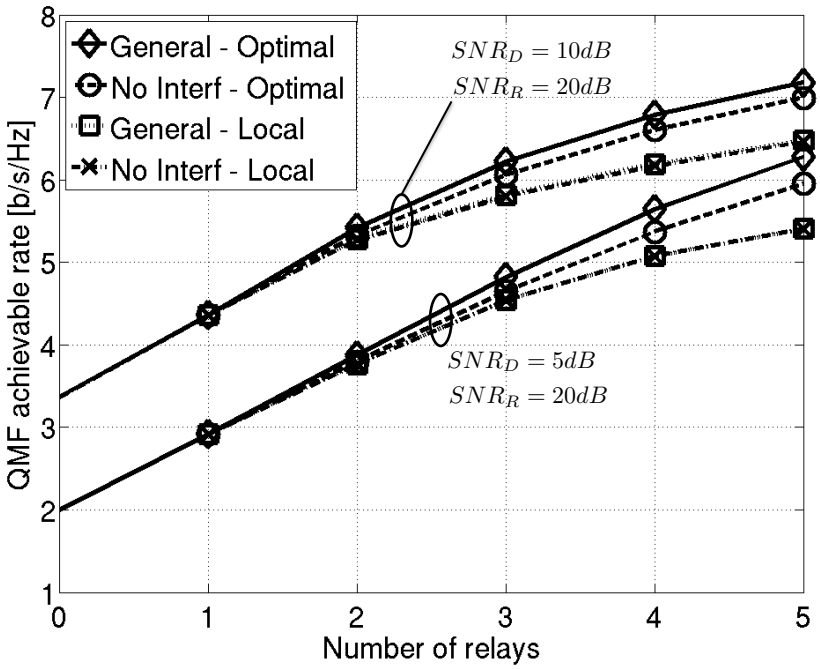

Figure 3.6: QMF achievable rate scaling with the number of relays in slow-fading channel conditions. Network with (General) and without (No Interf) relay-to-relay links, optimal and local relay scheduling.

also follows the intuition introduced in Section 2.1.1, which says that increasing the number of relays beyond the proximity gain value brings in only marginal, power gain. This corresponds to the sub-linear scaling of the QMF achievable rate curve with the number of relays larger than the proximity gain.

2. For typical values of η, the achievable QMF rate of the network without relay-to-relay links is very close to that of the general network. Rate loss of the suggested inter-relay interference cancellation algorithm with 5 relays is around 2.5% and 5%, for $\eta = 2$ and $\eta = 4$, respectively. Given a significant decrease in computation complexity that this algorithm enables, we conclude that it is good to treat relay-to-relay links as interference and cancel them at the destination.

3. The local scheduling scheme performs close to the optimal one, specifically for small number of relays (up to 3). Optimality of the local scheduling scheme is related to the proximity gain: the larger the proximity gain is, the closer to optimal will the local scheme perform. This is because larger η results in smaller listening time of the relays, which makes the condition (3.13) more likely to be satisfied.

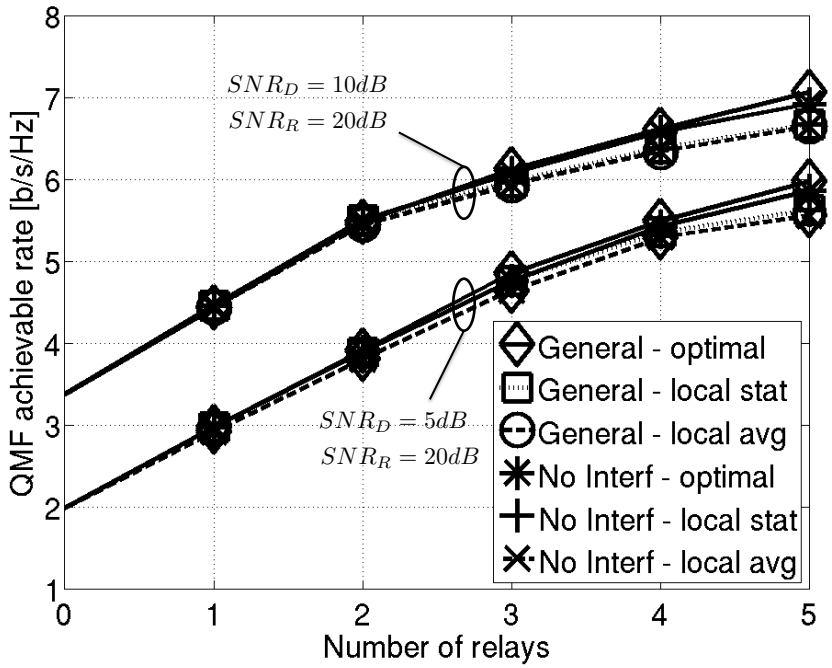

Figure 3.7: QMF achievable rate scaling with the number of relays in fast-fading channel conditions. Network with (General) and without (No Interf) relay-to-relay links, optimal relay scheduling (solid lines), local scheduling with known channel statistics (dotted lines) and local scheduling with known average channel values (dashed lines).

Fast Fading

In the fast-fading scenario, we compute the schedules using one of the three methods: optimal scheduling with known channel statistics, local scheduling with known channel statistics and local scheduling with known average SNR values. Then we use the expressions (3.7) and (3.2) to compute the QMF rate. Results for fast-fading channel conditions are presented in Figure 3.7.

Based on Figure 3.7, we conclude that the fast-fading scenario exhibits the same properties as the slow-fading scenario: 1) the QMF rate scales linearly with the number of relays and about $2-3\times$ data-rate gain can be achieved with relaying, 2) it pays off to cancel inter-relay interference, since the loss in performance is negligible, and 3) the local scheme performs very close to the optimal one even when *only average channel values are available*. This is of particular importance, because it shows that in the fast-fading channel conditions it is enough to track just the average channel values instead of the channel distributions.

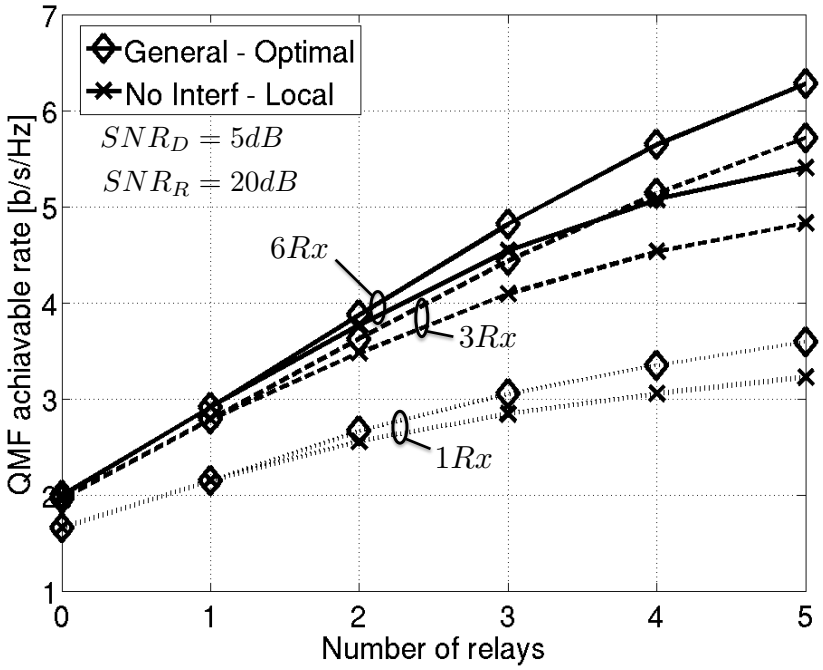

Figure 3.8: QMF achievable rate scaling with the number of relays in slow-fading channel conditions, for 6 (solid lines), 3 (dashed lines) and 1 (dotted lines) receive-antenna configurations.

3.6.2 Reducing the Number of Antennas at the Destination

Many would argue that having $N+1$ antennas at the destination is not practical for a large number of relays. In this section, we show that significant performance gain can be achieved even with smaller number of antennas. The reason for this is that relays are half-duplex and statistically it is not likely to have the situation that all relays are transmitting at the same time. Therefore, the destination can still achieve full multiplexing gain with less than $N+1$ antennas.

In Figures 3.8 and 3.9, we show data rate scaling when the destination has less than $N+1$ receive antennas, for slow and fast fading channel conditions, respectively. SNR values have been scaled such that the receive power gain is the same for all configurations, for fair comparison. We conclude that most of the multiplexing gain is captured even with 3 receive antennas, whereas 1 receive antenna captures only the power gain that comes from increasing number of transmit antennas. The difference in performance between the 3-antenna and the 6-antenna cases is not only because of the reduction in multiplexing gain, but also due to the reduction in diversity gain, even though the SNR values have been scaled for the same power gain.

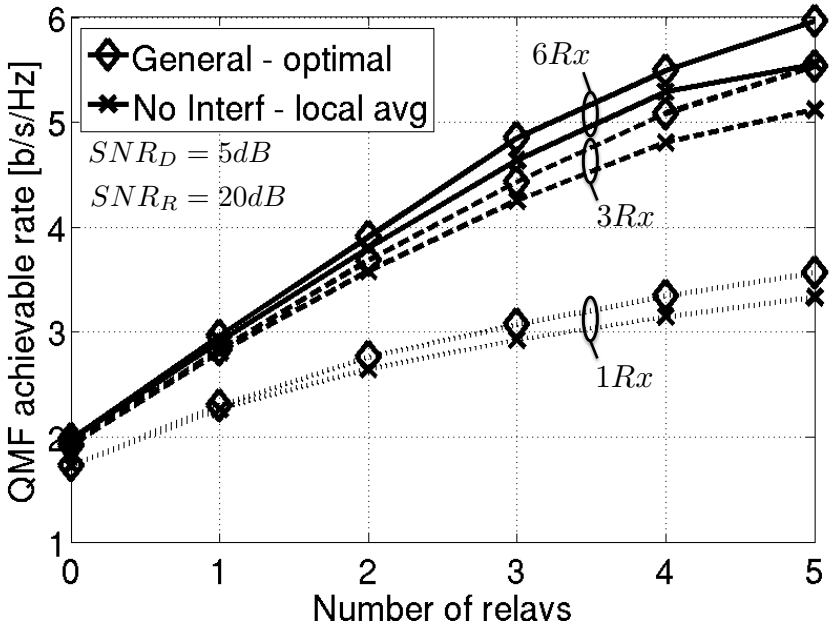

Figure 3.9: QMF achievable rate scaling with the number of relays in fast-fading channel conditions, for 6 (solid lines), 3 (dashed lines) and 1 (dotted lines) receive-antenna configurations.

3.7 Summary

Solving the relay scheduling problem is the key part of answering the question of how much data rate increase can be achieved with the QMF relaying. We demonstrate that the relays can be scheduled such that the spectral efficiency can be doubled with three relays and tripled with five relays. Calculating optimal relay schedules involves an optimization, with the complexity that grows exponentially with the number of relays. This is alleviated by using heuristics in the form of the local relay scheduling scheme and a simple rule to avoid overlaps in listening among different relays. The local scheduling scheme requires no optimization at all and is optimal if the relays can be scheduled such that no two relays listen at the same time. Otherwise, it performs close to the optimal scheme for smaller number of relays. The analysis and simulations are repeated for both the slow- and fast-fading channel conditions and different number of receive antennas at the destination. In all scenarios, the same conclusion has been reached: the local scheduling scheme performs very well and is much less complex for implementation than the optimal scheme.

Chapter 4

Physical-Layer Design of a QMF Cooperative-Relaying Link

In the previous chapter we showed that a significant data-rate gain can be achieved by cooperating with multiple relays. Those results were obtained through simulations of the theoretically achievable rate. In this chapter we show how to design a QMF cooperative-relaying link with multiple relays at the *bit-level*. In other words, how to ensure that the bit stream generated at the source is properly received and decoded at the destination, using advanced signal processing algorithms at the terminals.

In Chapter 3 we demonstrated that the general network, the one that includes relay-to-relay communication, can be simplified to the no-interference network, which does not take these links into consideration. The cancellation of relay-to-relay communication is enabled by using QMF and DBLAST. This result motivates us to focus on designing a multi-relay no-interference network, a simpler version compared to the general network case. One way to extend these design methods to the general network is presented at the end of this chapter.

We start this chapter by explaining main physical-layer signal processing blocks at each of the terminals: bit-interleaving and deinterleaving, modulation and demodulation, quantization at the relays, channel coding and decoding using principles and codes described in [38], and MIMO detection based on the MMSE-SIC algorithm presented in Chapter 2. To demonstrate that this system design procedure can achieve the data-rate gains promised by the theoretical analysis, we showcase a design example of a three-relay link. For given channel conditions, the theoretical analysis promises about $2\times$ multiplexing gain compared to the scenario without the relays. We present the design procedure that confirms the $2\times$ multiplexing gain, supporting this claim with the bit-level simulations of the cooperative-relaying link.

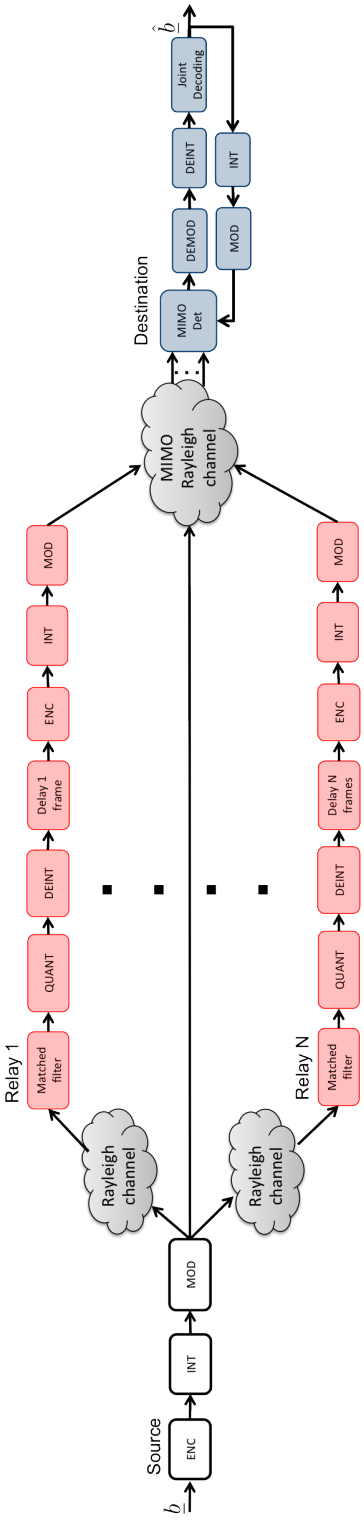

Figure 4.1: Block diagram of a multi-relay cooperative link.

4.1 System Description

A diagram of a multi-relay communication link presenting baseband signal processing blocks for all three types of terminals is shown in Figure 4.1. We focus on the signal processing blocks that come after the time and frequency synchronization and the channel estimation, which we assume have been done perfectly.

Note that the relays are half-duplex, so they need to be able to both receive signal from the source and transmit processed message to the destination. That is why they have both the receive and transmit baseband chains. The source transmits continuously and the destination receives continuously and therefore we focus only on their transmit and receive processing chains, respectively.

The DBLAST space-time scheme is reflected in delay blocks at each of the relays. Relay R_1 delays its transmission by one frame delay, relay R_2 by two frame delays, etc. Successive interference cancellation at the destination is reflected in the feedback path that feeds the decoded bits back to the MMSE-SIC MIMO detector for interference cancellation.

4.1.1 Notation

We denote the transmitted QAM symbols during frame l and symbol time n at the source and the relays with $x_S^l[n]$ and $x_{R_i}^l[n]$, respectively. We use the same notation for channel coefficients introduced in the previous chapter: we assume a flat-fading channel model, with the channels between the terminals described as scalars $h_{SR_i}^l[n]$ and vectors $\underline{h}_{TD}^l[n]$, for $T \in \{S, R_1, ..., R_N\}$ and n and l denoting the symbol time and the frame index, respectively.

The received symbols at the relays that are in the listening phase and at the destination can now be represented as:

$$\begin{aligned} y_{R_i}^l[n] &= h_{SR_i}^l[n] x_S^l[n] + z_{SR_i}^l[n] \\ \underline{y}_D^l[n] &= \underline{h}_{SD}^l[n] x_S^l[n] + \sum_{i \in \mathcal{T}_n} \underline{h}_{R_i D}^l[n] x_{R_i}^l[n] + \underline{z}_D^l[n], \end{aligned} \tag{4.1}$$

where z_{SR_i} and $\underline{z}_D$ denote the receive noise at relay R_i and destination D, respectively. $\mathcal{T}_n \subset \mathcal{R}$ is the set of relays that are in the transmitting phase during symbol time n.

4.1.2 Packet Structure and Effective Data Rate

In Chapter 2 we presented the DBLAST space-time coding scheme as if the frames were to be sent infinitely. In practical communication systems, data transfer occurs in packets that may contain many, but finite number of DBLAST frames. This means that the last N frames in a packet cannot be used for cooperation with N relays, because there is no time for all the relays to transmit their messages.

The solution is either to reduce the number of relays that the source cooperates with (and therefore reduce it's spectral efficiency and the relaying scheme), or to not use the relays at all and transmit the last N frames through a direct channel between the source and the

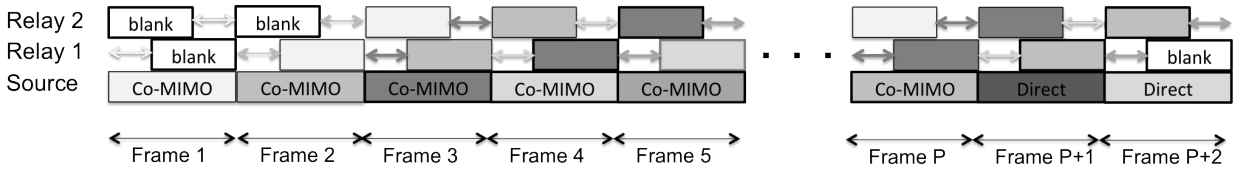

Figure 4.2: Packet structure with DBLAST and two relays.

destination. Because it would not be practical to change the design of a cooperative-relaying link for these last N frames, we decide to use the direct transmission between the source and the destination.

A typical DBLAST packet that contains P cooperative frames is presented in Figure 4.2, for the cooperative network with 2 relays. The first P frames are cooperative frames, and follow the DBLAST scheme from Figure 2.10. These P frames are designed for spectral efficiency of a cooperative-relaying link, R_{comimo}. The last 2 frames use direct communication between the source and the destination and are designed for spectral efficiency R_{direct}. The effective spectral efficiency is averaged over the whole packet and is calculated as:

$$R_{eff} = \frac{P}{P+N} \times R_{comimo} + \frac{N}{P+N} \times R_{direct} \tag{4.2}$$

4.2 Processing at the Source

Processing at the source uses the bit-interleaved coded-modulation (BICM) technique proposed in [7] and consists of channel coding using low-density parity-check (LDPC) code, bit-interleaving and QAM modulation, as shown in Figure 4.1. Before we explain the design of each of these blocks, we introduce the frame structure.

4.2.1 Frame Structure

As shown in Figure 4.2, the source transmits its information in packets, which consist of frames. We assume that each source frame contains a fixed number of N_S QAM symbols and each QAM symbol consists of C_S bits. Let N_S be equal to the blocklength of the LDPC code used at the source, so that the length of the frame corresponds to the length of a single codeword and the frame consists of C_S codewords. Data contained in a source frame is visualized in Figure 4.3.

4.2.2 LDPC Encoding

Let's denote the information bits at the source with the set of matrices $\{B_I^l[m,n]\}$, where $m \in \{1,\ldots,C_S\}$ denotes which codeword within the frame the bit belongs to, $n \in \{1,\ldots,N_I\}$ the position of the bit within the information word and $l \in \{1,\ldots,P\}$ the frame index.

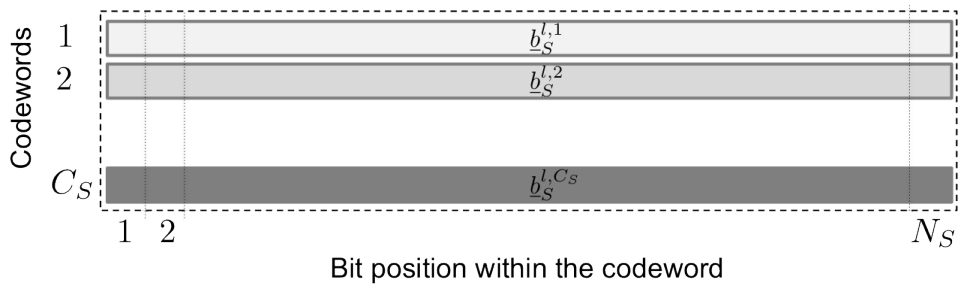

Figure 4.3: Data structure of a source frame l: Bit-matrix B_S^l contains C_S codewords, each N_S bits long. This data gets modulated into N_S QAM symbols, where each QAM symbol contains C_S bits.

The matrix of information bits in frame l is organized into information words $\underline{b}_I^{l,m}[n]$ as: $B_I^l = [\underline{b}_I^{l,1}, \ldots, \underline{b}_I^{l,C_S}]^T$. Each information word is encoded using the LDPC encoder with the $N_I \times N_S$ generator matrix G_S:

$$(\underline{b}_S^{l,m})^T = (\underline{b}_I^{l,m})^T \cdot G_S, \quad (\forall m, l) : m \in \{1, \ldots, C_S\}, l \in \{1, \ldots, P\} \tag{4.3}$$

We denote bits in a source frame l using a bit-matrix B_S^l as shown in Figure 4.3. Bit-matrix in each frame, B_S^l, can be represented through source codewords as: $B_S^l = [\underline{b}_S^{l,1}, \ldots, \underline{b}_S^{l,C_S}]^T$.

Based on the results on channel code design for cooperative relaying [41], it is reasonable to assume that the LDPC code is systematic, meaning that the first N_I bits of the source codeword $\underline{b}_S^{l,m}$ are equal to the information bits $\underline{b}_I^{l,m}$. In other words, the generator matrix G_S has a form:

$$G_S = [I | G'_S], \tag{4.4}$$

where a $N_I \times (N_S - N_I)$ matrix G'_S represents a non-systematic part of G_S.

4.2.3 Bit-Interleaving

Interleaving operation Π_S shuffles the encoded bits before the modulation, thus ensuring that bits from the same codeword do not all take the same bit-position within a QAM symbol. This is one of the key features of BICM [7]. The interleaving operation is performed per frame: the bit-matrix $B_S^l[m, n]$ is reshuffled into a new matrix $B_{S,INT}^l[m, n]$ through frame-by-frame multiplications with the $(C_S N_S) \times (C_S N_S)$ permutation matrix Π_S:

$$[(\underline{b}_{S,INT}^{l,1})^T, \ldots, (\underline{b}_{S,INT}^{l,C_S})^T] = \Pi_S \cdot [(\underline{b}_S^{l,1})^T, \ldots, (\underline{b}_S^{l,C_S})^T] \tag{4.5}$$

After interleaving, the bit-matrix of frame l is: $B^l_{S,INT} = [\underline{b}^{l,1}_{S,INT}, \ldots, \underline{b}^{l,C_S}_{S,INT}]^T$, with each row of this new bit-matrix containing approximately an equal number of bits from every source codeword. This technique ensures that when modulated and then demodulated, bits that belong to the original source codewords see as many equivalent binary-input AWGN (BIAWGN) channels as available [7].

4.2.4 QAM Modulation

At the modulation block, we use a standard QAM modulator that modulates bits of a bit-matrix $B^l_{S,INT}[m,n]$ into a sequence of QAM symbols $\{x^l_S[n]\}$, where $n \in \{1,\ldots,N_S\}$ and l is the frame index. The modulator takes bits from $B^l_{S,INT}$ column-by-column and modulates them into the QAM symbols $\{x^l_S[n]\}$:

$$x^l_S[n] = \mathrm{mod}(B^l_{S,INT}[1,n],\ldots,B^l_{S,INT}[C_S,n]), \tag{4.6}$$

where symbols $x^l_S[n]$ take one of the 2^{C_S} values in the set of all 2^{C_S}-QAM symbols, $\mathcal{X}_C$: $x^l_S[n] \in \mathcal{X}_S$. These symbols are then broadcasted one-by-one through the medium, and received at the destination and at the relays that are in the listening phase.

4.3 Processing at the Relay

Relay processing includes the receive chain that is active during the listening phase, a delay element to implement the DBLAST scheme, and the transmit chain that is active during the transmit phase, as shown in Figure 4.1. The receive chain consists of the matched filter, which extracts available energy from the fading channel, quantizer and deinterleaver. The quantized bits are delayed by an integer number of frames and passed through a transmit chain that resembles that of the source, but uses different channel code and a different interleaver.

Before we explain each of the processing blocks, we summarize the relay-scheduling algorithm that determines which part of the relays' processing chain is active during each symbol time n.

4.3.1 Relay Scheduling

Relay scheduling for both slow and fast fading channel conditions has been analyzed in the previous chapter. The algorithm below summarizes the local scheduling scheme that includes both fading scenarios and is adapted to the discrete nature of symbol transmission:

1. If the channel stays constant across at least $N+1$ communication frames, calculate the SNR values of all links of the no-interference network using the instantaneous channel values:

$$\begin{aligned} SNR_{SD} &= |\underline{h}_{SD}|^2 = \underline{h}_{SD}^H \underline{h}_{SD}; \\ SNR_{SR_i} &= |h_{SR_i}|^2; \\ SINR_{R_iD} &= \underline{h}_{R_iD}^H (I + \underline{h}_{SD}\underline{h}_{SD}^H + \sum_{k=1}^{k=i} \underline{h}_{R_kD}\underline{h}_{R_kD}^H)^{-1} \underline{h}_{R_iD}, \end{aligned} \tag{4.7}$$

where symbol and frame indices have been omitted because the channel remains constant. Otherwise calculate the average SNR values:

$$\begin{aligned} SNR_{SD} &= E[|\underline{h}_{SD}^l[n]|^2] = E[(\underline{h}_{SD}^l[n])^H \underline{h}_{SD}^l[n]]; \\ SNR_{SR_i} &= E[|h_{SR_i}^l[n]|^2]; \\ SINR_{R_iD} &= E[(\underline{h}_{R_iD}^l[n])^H (I + \underline{h}_{SD}^l[n](\underline{h}_{SD}^l[n])^H + \sum_{k=1}^{k=i} \underline{h}_{R_kD}^l[n](\underline{h}_{R_kD}^l[n])^H)^{-1} \underline{h}_{R_iD}^l[n]], \end{aligned} \tag{4.8}$$

where expectation is taken over all symbol times $n \in \{1, 2, \ldots, N_S\}$ and frame indices $l \in \{1, 2, \ldots, P\}$.

2. Calculate the listening time f_i for each relay R_i as if there was no other relays in the network:

$$f_i = \frac{\log(1 + SINR_{R_iD})}{\log(1 + SINR_{R_iD}) + [\log(1 + SNR_{SD} + \frac{SNR_{SR_i}}{2}) - \log(1 + SNR_{SD}) + 1]}, \tag{4.9}$$

with SNR values from step 1. Round these values to the nearest multiple of $\frac{1}{N_S}$:

$$\tilde{f}_i = \frac{[f_i \cdot N_S]}{N_S}, \tag{4.10}$$

so that relay R_i listens to the source for $N_{Q_i} = \tilde{f}_i \cdot N_S$ symbols and transmits for $N_{R_i} = N_S - N_{Q_i}$ symbols. $[\cdot]$ denotes rounding to the nearest integer.

3. If the condition

$$\tilde{f}_1 + \tilde{f}_2 + \ldots + \tilde{f}_N \leq 1 \tag{4.11}$$

holds, then go to step 4, otherwise go to step 5;

4. Relay R_i is scheduled to listen to the transmitted symbols with indices $n \in \mathcal{L}_i$, where:

$$\mathcal{L}_i = [\sum_{j=1}^{j=i-1} \tilde{f}_j \cdot N_S + 1, \sum_{j=1}^{j=i} \tilde{f}_j \cdot N_S] \tag{4.12}$$

5. According to the continuous local scheduling scheme, relay R_i should be scheduled to listen in the interval $[\frac{2i-1}{2N} - \frac{f_i}{2}, \frac{2i-1}{2N} + \frac{f_i}{2}]$, based on its index number i. We round this interval to the nearest multiple of $\frac{1}{N_S}$, so that the relay R_i should listen to the symbol indices $n \in \mathcal{L}_i$, where:

$$\mathcal{L}_i = \left[\left[\left(\frac{2i-1}{2N} - \frac{f_i}{2}\right) \cdot N_S\right] + 1, \left[\left(\frac{2i-1}{2N} - \frac{f_i}{2}\right) \cdot N_S\right] + \tilde{f}_i \cdot N_S\right] \tag{4.13}$$

As we concluded in the previous chapter, the local scheduling scheme is implementation friendly for several reasons: 1) it does not require complex optimization procedure, 2) it schedules the relays to switch between listening and transmitting phase only once during each frame, and 3) it allows distributed calculation of scheduling parameters (f_i can be calculated at the relays).

4.3.2 Matched Filter

In Chapter 2 we concluded that the matched filter is the optimal receiver for a SIMO channel. Since SISO channel between the source and the relay is a special case of SIMO, the relay deploys a matched filter to maximize the output SNR of the received symbols. For each symbol within a frame, a matched filter computes the received estimate of the transmitted QAM symbol as:

$$\hat{y}^l_{SR_i}[n] = \frac{h^l_{SR_i}[n]^H \cdot y^l_{R_i}[n]}{|h^l_{SR_i}[n]|^2} \quad = x^l_S[n] + \hat{z}^l_{SR_i}[n], \tag{4.14}$$

where the variance of $\hat{z}^l_{SR_i}[n]$ can be computed as:

$$\sigma^2_{\hat{z}^l_{SR_i}[n]} = \frac{\sigma^2_{z_{SR_i}}}{|h^l_{SR_i}[n]|^2} \tag{4.15}$$

4.3.3 Quantization at the Relay

The original QMF scheme [4] states that the relays should quantize the received signal at the noise level, or that quantization levels can be determined through optimization, as suggested by [51]. When a discrete constellation such as QAM is used, however, there is practically no need to quantize each of the received symbols into more bits than what it has been used at the transmitter, C_S. This approach has been justified in [38], by using bit-interleaved coded modulation (BICM) and a scalar quantizer at the relay.

Quantization using Hard Demodulation

When the received symbol $\hat{y}^l_{SR_i}[n]$ is quantized into exactly C_S bits, as suggested by [38], quantization simplifies to hard demodulation of the received symbols. Hard demodulation

consists of two steps: 1) finding the QAM symbol $\hat{x}^l_{SR_i}[n]$ from the constellation $\mathcal{X}_S$ that is closest to the received symbol $\hat{y}^l_{SR_i}[n]$, and 2) demapping the demodulated QAM symbol $\hat{x}^l_{SR_i}[n]$ back into bits:

$$\hat{x}^l_{SR_i}[n] = \arg\min_{x \in \mathcal{X}_S} \left\{ |\hat{y}^l_{SR_i}[n] - x|^2 \right\} \tag{4.16}$$

$$\{B^l_{Q_i,INT}[1,n], \ldots, B^l_{Q_i,INT}[C_S,n]\} = \text{demod}(\hat{x}^l_{SR_i}[n]), \tag{4.17}$$

Quantization using Soft-Bit Demodulation

If we were to follow procedure described in [38] exactly, we would demodulate the received symbols $\hat{y}^l_{SR_i}[n]$ into the soft-bit values (LLRs) first, and then quantize these into an arbitrary number of bits. This procedure is advantageous when the received symbols are quantized into more than C_S bits, because it enables easier deinterleaving and subsequent encoding at the relays.

Soft-bit demodulator calculates the log likelihood ratios (LLRs) for each of the bits in received symbol estimate $\hat{y}^l_{SR_i}[n]$ as:

$$LLR^l_{SR_i}[m,n] = \log \frac{P_{B|Y,H}(B^l[m,n]=0|\hat{y}^l_{SR_i}[n], h^l_{SR_i}[n])}{P_{B|Y,H}(B^l[m,n]=1|\hat{y}^l_{SR_i}[n], h^l_{SR_i}[n])}, \tag{4.18}$$

where $m \in \{1, \ldots, C_S\}$ and $n \in \mathcal{L}_i$ is the symbol index when the relay listens to the source. Because this operation takes place for every symbol time $n \in \mathcal{L}_i$ and for every frame index l, we can omit the symbol and frame indices in the following discussion.

Calculation of probability of bit b_m taking values 0 and 1 depends on the QAM constellation $\mathcal{X}_S$. The optimal way to calculate the numerator and the denominator in (4.18) is to use the Bayes formula and then sum over all QAM symbols within $\mathcal{X}_S$ that have $b_m = 0$ and $b_m = 1$, respectively:

$$\begin{aligned} LLR_{SR_i}[m] &= \log \frac{\frac{f_{\hat{Y}_{SR_i}|b_m}(\hat{y}_{SR_i}|b_m=0)\cdot P(b_m=0)}{f_{\hat{Y}_{SR_i}}(\hat{y}_{SR_i})}}{\frac{f_{\hat{Y}_{SR_i}|b_m}(\hat{y}_{SR_i}|b_m=1)\cdot P(b_m=1)}{f_{\hat{Y}_{SR_i}}(\hat{y}_{SR_i})}} \\ &= \log \frac{f_{\hat{Y}_{SR_i}|b_m}(\hat{y}_{SR_i}|b_m=0)}{f_{\hat{Y}_{SR_i}|b_m}(\hat{y}_{SR_i}|b_m=1)} \\ &= \log \frac{\sum_{x_S \in \mathcal{X}_S, b_m=0} f_{\hat{Y}_{SR_i}|X}(\hat{y}_{SR_i}|x) \cdot P(x=x_S)}{\sum_{x_S \in \mathcal{X}_S, b_m=1} f_{\hat{Y}_{SR_i}|X}(\hat{y}_{SR_i}|x) \cdot P(x=x_S)} \end{aligned} \tag{4.19}$$

We also assume that the source-coding block (not presented in Figure 4.1) has made each bit equally likely to take values 0 and 1, i.e. $P(b_m = 0) = P(b_m = 1) = 0.5$, and that

due to random interleaving at the source, Π_S, each symbol $x \in \mathcal{X}_S$ is equally likely to be transmitted. Assuming noise $z^l_{SR_i}[n]$ is white, expression (4.19) becomes:

$$LLR_{SR_i}[m] = \log \frac{\sum_{x\in\mathcal{X}_S, b_m=0} e^{-\frac{1}{2\sigma^2_{\hat{z}_{SR_i}}}|\hat{y}_{SR_i}-x|^2}}{\sum_{x\in\mathcal{X}_S, b_m=1} e^{-\frac{1}{2\sigma^2_{\hat{z}_{SR_i}}}|\hat{y}_{SR_i}-x|^2}} \tag{4.20}$$

This equation can be further simplified by using the so-called max-log approximation, by keeping only the most dominant exponential terms in the nominator and the denominator sums:

$$LLR_{SR_i}[m] = \log \frac{\max_{x\in\mathcal{X}_S, b_m=0} \left\{ e^{-\frac{1}{2\sigma^2_{\hat{z}_{SR_i}}}|\hat{y}_{SR_i}-x|^2} \right\}}{\max_{x\in\mathcal{X}_S, b_m=1} \left\{ e^{-\frac{1}{2\sigma^2_{\hat{z}_{SR_i}}}|\hat{y}_{SR_i}-x|^2} \right\}}. \tag{4.21}$$

This enables convenient cancellation of the exponentials and the logarithm:

$$\begin{aligned} LLR_{SR_i}[m] &= \max_{x\in\mathcal{X}_S, b_m=0} \left\{ -\frac{1}{2\sigma^2_{\hat{z}_{SR_i}}}|\hat{y}_{SR_i}-x|^2 \right\} - \max_{x\in\mathcal{X}_S, b_m=1} \left\{ -\frac{1}{2\sigma^2_{\hat{z}_{SR_i}}}|\hat{y}_{SR_i}-x|^2 \right\} \\ &= \frac{1}{2\sigma^2_{\hat{z}_{SR_i}}} \left(\min_{x\in\mathcal{X}_S, b_m=0} \left\{ |\hat{y}_{SR_i}-x|^2 \right\} - \min_{x\in\mathcal{X}_S, b_m=1} \left\{ |\hat{y}_{SR_i}-x|^2 \right\} \right) \end{aligned} \tag{4.22}$$

Note that the max-log algorithm is much simpler to implement than to sum over all possible QAM symbols in (4.20), specially for larger QAM constellations. Expression (4.22) reduces to finding the two closest QAM symbols to the received symbol $\hat{y}_{SR_i}$, one of which has $b_m = 0$ and the other one $b_m = 1$ [29].

According to [38], it is enough to quantize each LLR value $LLR^l_{SR_i}[m,n]$ with a scalar 1-bit quantizer. Because LLR values are real numbers, 1-bit scalar quantization becomes a simple "sign" operation:

$$B^l_{Q_i,INT}[m,n] = \text{sign}(LLR^l_{SR_i}[m,n])) \tag{4.23}$$

Notice that performing soft-bit demodulation and 1-bit scalar quantization of the LLR values is equivalent to performing hard demodulation, but requires more complex signal processing due to soft-bit computation.

4.3.4 Deinterleaver

The deinterleaver block at the receiver cancels the effect of the interleaver block at the transmitter. It takes the quantized bits $B^l_{Q_i,INT} = [\underline{b}^{l,1}_{Q_i,INT}, \ldots, \underline{b}^{l,C_S}_{Q_i,INT}]^T$ from the quantizer and

reshuffles them back to the pre-interleaver format, presented in Figure 4.3. The deinterleaved bit-matrix $B^l_{Q_i} = [\underline{b}^{l,1}_{Q_i}, \ldots, \underline{b}^{l,C_S}_{Q_i}]^T$, is computed as:

$$[(\underline{b}^{l,1}_{Q_i})^T, \ldots, (\underline{b}^{l,C_S}_{Q_i})^T] = \Pi_S^{-1} \cdot [(\underline{b}^{l,1}_{Q_i,INT})^T, \ldots, (\underline{b}^{l,C_S}_{Q_i,INT})^T] \tag{4.24}$$

As a result of this operation, rows of the bit-matrix $B^l_{Q_i}$ will contain soft-bit information of the original source codewords, so they can be encoded again without worrying about mixing bits from different source codewords.

4.3.5 LDGM Encoding

The transmit part of the relays' processing chain re-encodes the quantized bit-matrix $B^l_{Q_i}$ row-by-row, using a low-density generator matrix (LDGM) code with a $N_{Q_i} \times N_{R_i}$ generator matrix G_{R_i}:

$$(\underline{b}^{l,m}_{R_i})^T = (\underline{b}^{l,m}_{Q_i})^T \cdot G_{R_i}, \quad (\forall m, l) : m \in \{1, \ldots, C_S\}, l \in \{1, \ldots, P\} \tag{4.25}$$

Design of the LDGM code for a single-relay scenario has been done in [38]. Same code design process can be extended in a straightforward manner to a multi-relay no-interference network, by having each relay use an LDGM code with the rate that matches its listening schedule. Because the relay R_i acquires $\tilde{f}_i N_S$ quantized bits that get encoded into $(1-\tilde{f}_i)N_S$ coded bits, the code rate of its LDGM code is:

$$r_i^{LDGM} = \frac{\tilde{f}_i}{1-\tilde{f}_i} = \frac{N_S - N_{R_i}}{N_{R_i}} \tag{4.26}$$

To achieve the optimal performance, a joint code design for the source and all the relays should be performed. For a cooperative system with a large number of relays, there is a huge number of combinations for channel codes to be used, depending on the channel conditions and the corresponding listening schedules at the relays. For practical purposes, in this work we take the suboptimal approach for choosing the channel codes: we design an LDGM code for each relay separately, using principles from [38]. This idea follows the one introduced in Chapter 3 about the local scheduling scheme: relays should be treated as if there was no other relays around.

4.3.6 Interleaving and Modulation

Just as in the case of the source transmit chain, these encoded bits are interleaved into a new set of matrices $\{B^l_{R_i,INT}\}$ through frame-by-frame multiplications with the random $(C_S N_{R_i}) \times (C_S N_{R_i})$ permutation matrix Π_{R_i}:

$$[(\underline{b}^{l,1}_{R_i,INT})^T, \ldots, (\underline{b}^{l,C_S}_{R_i,INT})^T] = \Pi_{R_i} \cdot [(\underline{b}^{l,1}_{R_i})^T, \ldots, (\underline{b}^{l,C_S}_{R_i})^T] \tag{4.27}$$

We modulate the interleaved bits to a new set of QAM symbols, $x^l_{R_i}[n], \forall n \in \mathcal{T}_i$, to be transmitted to the destination during the transmit phase. $\mathcal{T}_i = \{1, \ldots, N_S\} \setminus \mathcal{L}_i$ is the set

of symbol indices within the frame when the relay R_i is transmitting. During the listening phase, relay R_i does not transmit and therefore $x^l_{R_i}[n] = 0, \forall n \in \mathcal{L}_i$

We assume that the relays use the same QAM constellation as the source (C_S bits per QAM symbol), so the blocklength of the LDGM code at relay R_i, N_{R_i}, depends only on its listening fraction: $N_{R_i} = (1 - \tilde{f}_i)N_S$. Note that $\tilde{f}_i$ has been chosen in Section 4.3.1 such that N_{R_i} is an integer.

4.4 Processing at the Destination

Signal processing at the destination includes the receive chain that supports DBLAST space-time scheme, MMSE-SIC MIMO detection, soft-bit demodulation, deinterleaving and joint decoding. Because of the successive interference cancellation, decoded bits need to be re-interleaved (with the interleaver used at the corresponding transmitting terminal) and modulated to QAM symbols, whose transmission created the interference that is being cancelled. That explains the feedback path in the destination block diagram, presented in Figure 4.1.

To detect all transmitted streams at the destination, we apply the MMSE-SIC MIMO detector, which has been studied in Chapter 2. This detector detects transmitted streams from the source and the transmitting relays one by one, according to the DBLAST scheme. The blocks that follow the MMSE-SIC detector — demodulator, deinterleaver and joint decoder, as well as the interleaver and modulator in the feedback path, are run stream-by-stream, until all $N + 1$ transmitted streams are decoded.

Because the DBLAST scheme includes latency of i frames for relay R_i, the destination needs to store values received up to N frames before the current frame so it can detect messages coming from all N relays. In particular, in order to decode the message M^S_l the destination needs to store received symbols that contain information about all messages related to that one: $\{M^S_l, M^{R_1}_l, \ldots, M^{R_N}_l\}$, which have been transmitted in frames $F_l, F_{l+1}, \ldots, F_{l+N}$. Instead of storing the $N + 1$ frames worth of the received symbol vectors, $\{\underline{y}^l_D, \underline{y}^{l+1}_D, \ldots, \underline{y}^{l+N}_D\}$, we choose to store the $N + 1$ frames of the "residual" received vectors $\{\tilde{\underline{y}}^l_D, \tilde{\underline{y}}^{l+1}_D, \ldots, \tilde{\underline{y}}^{l+N}_D\}$, which are calculated by successively subtracting interference of previously decoded streams. Before the messages $\{M^S_l, M^{R_1}_l, \ldots, M^{R_N}_l\}$ are decoded, these residual received vectors will be equal to:

$$\tilde{\underline{y}}^{l+N-i}_D[n] = \underline{y}^{l+N-i}_D[n] - \sum_{k=1}^{i} \underline{h}^{l+N-i}_{R_{N+1-k}D}[n] \cdot \hat{x}^{l+N-i}_{R_{N+1-k}}[n], \tag{4.28}$$

for $n \in \{1, ..., N_S\}$ and $i \in \{1, \ldots, N\}$. $\hat{x}^l_{R_iD}[n]$ represents a reconstructed QAM symbol transmitted by relay R_i during frame l and symbol time n. To avoid storing large amount of decoded bits and old channel values, it is more practical to update the residual received vectors $\{\tilde{\underline{y}}^l_D[n]\}$ as each stream gets detected and decoded. Also, from this point on we assume that channel coefficients $\underline{h}^l_{R_iD}[n]$ take value zero when the relay R_i does not transmit, i.e.:

$$\underline{h}^l_{R_iD}[n] = 0, \quad \forall n \in \mathcal{L}_i \tag{4.29}$$

According to the DBLAST scheme, messages related to the first frame, $M_1^S, M_1^{R_1}, ..., M_1^{R_N}$, do not experience any interference from the "upper layer" streams, so the residual received vectors can be initialized to the original received vectors:

$$\tilde{\underline{y}}^l_D[n] = \underline{y}^l_D[n], \forall (n,l) : n \in \{1, ..., N_S\}, l \in \{1, ..., N+1\} \tag{4.30}$$

Destination processing of the messages $\{M_l^S, M_l^{R_1}, ..., M_l^{R_N}\}$ happens after frame $l+N$ has been received and consists of the following steps: MIMO detection, demodulation, deinterleaving, joint decoding, interleaving, modulation and successive interference cancellation.

4.4.1 MIMO Detection

Streams coming from the source and the relays are detected using the MMSE-SIC filters that assume all interference from the upper layer relays' streams has been cancelled (or non-existing), as suggested by the DBLAST scheme. To detect streams from the source and the relay R_i, the destination applies the MMSE-SIC filters defined in (2.31). In the cooperative-relaying notation we use in this chapter, these filters are:

$$\begin{aligned}
(\underline{g}^l_S[n])^T &= (\underline{h}^l_{SD}[n])^H \left(I + \underline{h}^l_{SD}[n](\underline{h}^l_{SD}[n])^H\right)^{-1} \\
(\underline{g}^l_{R_i}[n])^T &= (\underline{h}^l_{R_iD}[n])^H \left(I + \underline{h}^l_{SD}[n](\underline{h}^l_{SD}[n])^H + \sum_{k=1}^{i} \underline{h}^l_{R_kD}[n](\underline{h}^l_{R_kD}[n])^H\right)^{-1}
\end{aligned} \tag{4.31}$$

The received symbol estimate $\hat{y}^l_{SD}[n]$ of the source's message M_l^S is calculated as:

$$\hat{y}^l_{SD}[n] = (\underline{g}^l_S[n])^T \cdot \tilde{\underline{y}}^l_D[n], \quad \forall n \in \{1, 2, \ldots, N_S\} \tag{4.32}$$

Similarly, the received symbol estimate $\hat{y}^{l+i}_{R_iD}[n]$ of the message $M_l^{R_i}$ from relay R_i is calculated as:

$$\hat{y}^{l+i}_{R_iD}[n] = \begin{cases} (\underline{g}^{l+i}_{R_i}[n])^T \cdot \tilde{\underline{y}}^{l+i}_D[n], & \forall n \in \mathcal{T}_i \\ 0, & \forall n \in \mathcal{L}_i \end{cases} \tag{4.33}$$

Note that zeroing received symbol estimates during the listening phases of the relays is in line with the notation from (4.29). The received symbol estimates $\hat{y}^l_{SD}[n]$ and $\hat{y}^{l+i}_{R_iD}[n]$ are computed for all symbols $n \in \{1, ..., N_S\} = \mathcal{L}_i \cup \mathcal{T}_i$ and all the relays $i \in \{1, ..., N\}$.

4.4.2 Demodulation and Deinterleaving

Demodulation and deinterleaving follow the same procedure as in case of the relays' receiver chain described in Section 4.3. The received symbol estimates $\{\hat{y}^l_{SD}[n], \hat{y}^{l+1}_{R_iD}[n], ..., \hat{y}^{l+N}_{R_ND}[n]\}$, where $n \in \{1, 2, \ldots, N_S\}$ are demodulated using the soft-bit QAM demodulator that computes LLR values of the bits consisting QAM symbols using any of the methods described in Section 4.3.3, for example max-log:

$$\begin{aligned} LLR^l_{SD,INT}[m,n] &= \frac{SINR^l_{SD}[n]}{2}\left(\min_{x\in\mathcal{X}_S, b_m=0}\left\{|\hat{y}^l_{SD}[n]-x|^2\right\} - \min_{x\in\mathcal{X}_S, b_m=1}\left\{|\hat{y}^l_{SD}[n]-x|^2\right\}\right) \\ LLR^l_{R_iD,INT}[m,n] &= \frac{SINR^l_{R_iD}[n]}{2}\left(\min_{x\in\mathcal{X}_S, b_m=0}\left\{|\hat{y}^l_{R_iD}[n]-x|^2\right\} - \min_{x\in\mathcal{X}_S, b_m=1}\left\{|\hat{y}^l_{R_iD}[n]-x|^2\right\}\right), \end{aligned} \tag{4.34}$$

where the SINR values are computed by applying expression (2.32) on each stream.

Demodulated LLR values are then deinterleaved by applying deinterleavers $\Pi_S^{-1}, \Pi_{R_1}^{-1}, ..., \Pi_{R_N}^{-1}$ to the data blocks $LLR^l_{SD,INT}, LLR^{l+1}_{R_iD,INT}, \ldots, LLR^{l+N}_{R_ND,INT}$ respectively, in exactly the same fashion as described in Section 4.3.4. Deinterleaved LLR values are denoted as LLR^l_{SD}, $LLR^{l+1}_{R_iD}, \ldots, LLR^{l+N}_{R_ND}$.

4.4.3 Joint Decoding

After detection, demodulation and deinterleaving, LLR estimates of the messages $M_l^S, M_l^{R_1}$, $\ldots, M_l^{R_N}$ are passed to the joint decoder. As we explained in Chapter 2, one of the key properties of the QMF scheme is that a message from the source is decoded using the relays' messages as side information. This relationship between the source and the relay codewords is described by connecting the two codewords through the Q-nodes. Structure of the joint Tanner graphs, which connect the Tanner graphs of each of the terminals is presented in Figure 4.4, for the cases of the single-relay and the two-relay communication links. The Q-nodes are represented as equivalent combination of check nodes and observation nodes, as suggested in Chapter 2.

Note that the joint Tanner graph from Figure 4.4 can be extended in a similar manner to the scenario with more than two relays: information bits of the LDGM code at relay R_i would be connected through the equivalent Q-nodes to the bits of the source codeword that correspond to the time indices $n \in \mathcal{L}_i$ when the relay R_i listens. This is possible because each relay listens to the source only, therefore there are no direct connections between the nodes that belong to different LDGM codes. The LDPC Tanner graph at the source does not include information variable bits, because they are equivalent to the first N_I coded variable bits due to LDPC code being systematic.

The nature of the Q-nodes is to introduce ambiguity that comes from the transmission of the source bits, $\underline{b}^l_S[n]$, through the source-relay channel, before they are received and quantized at the relays. This probabilistic relationship is captured in the message transmitted by the dummy observation nodes, denoted as $LLR^{l+i}_{ERR_i}[n]$, and it depends on the SNR of

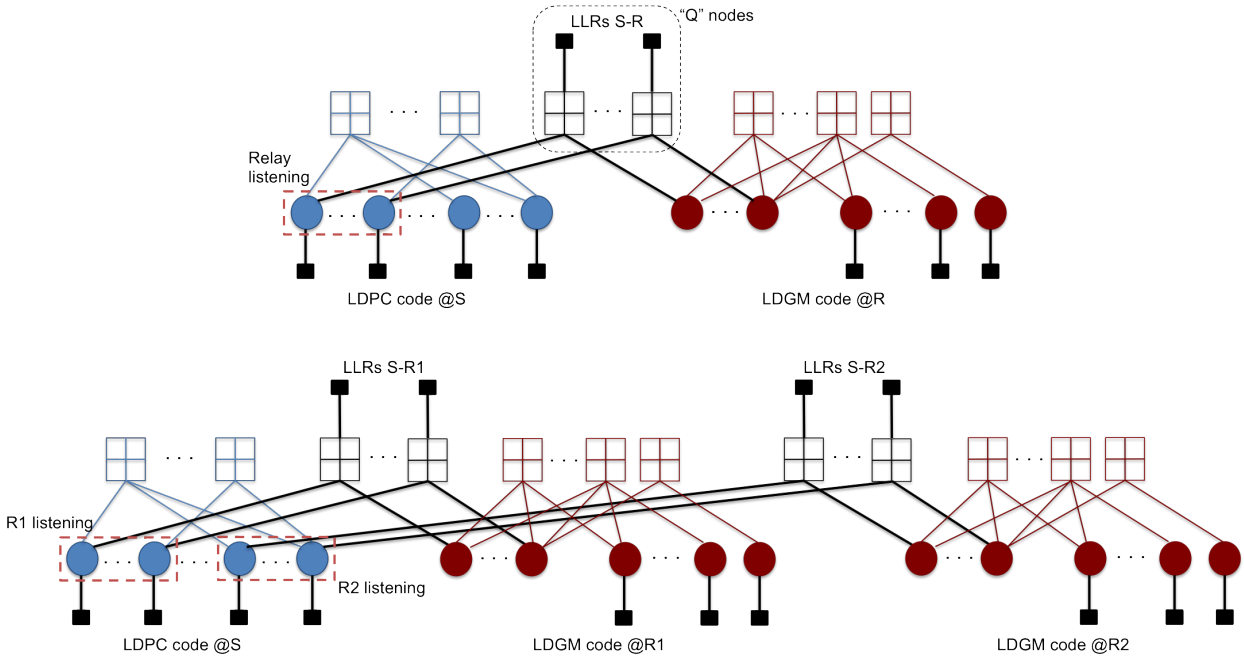

Figure 4.4: Tanner graph structure for cooperative relaying system with one (top) and two (bottom) relays and schedules that correspond to Figure 4.2.

the equivalent BIAWGN channel between the source and the relay on bit position i, after BICM has been applied [38]. Calculating this equivalent BIAWGN SNR in real time for every source-relay channel instance can be very challenging, specially if the channel is changing quickly. A much easier approach is to precompute the $LLR^{l+i}_{ERR_i}[n]$ messages based on the symbol-SNR calculated after matched filtering at the relays.

In particular, for any QAM constellation and any bit position within a symbol, we can find an empirical relationship between the output SNR on each source-relay link and the probability p_{err} of having received bits at the relay flipped. These values for p_{err} can then be stored at the destination to be used during the joint decoding step for computation of $LLR^{l+i}_{ERR_i}[n]$ messages according to expression (2.52). Some example values for the bit-flipping probabilities of the $S-R$ link are given in Table 4.1.

Once all LLR messages for all the observation nodes in the joint Tanner graph have been computed, a standard message passing algorithm can be applied and the source and the relays' codewords can be jointly decoded. Detailed description of the message passing algorithm can be found in [34]. Here we provide a short version, to show how it applies to the joint decoder.

The messages are passed along the edges of the Tanner graph. Let's denote a message from the node n_i to the node n_j, connected to the node n_i, as $m(n_i, n_j)$. We denote the source variable nodes of the joint Tanner graph as $b_S[n]$, and the information and coded

Table 4.1: Bit-flipping probabilities for different SNR values, constellations and bit positions.

SNR	Constellation	Position	Stored P_{err}
5	QPSK	1	0.075
5	16-QAM	1	0.158
5	16-QAM	2	0.29
5	64-QAM	1	0.197
5	64-QAM	2	0.318
5	64-QAM	3	0.464
20	QPSK	1	$< 10^{-3}$
20	16-QAM	1	$< 10^{-3}$
20	16-QAM	2	$< 10^{-3}$
20	64-QAM	1	0.028
20	64-QAM	2	0.034
20	64-QAM	3	0.059

variable nodes of relay R_i as $b_{Q_i}[n]$ and $b_{R_i}[n]$, respectively. The decoding algorithm iterates over the joint Tanner graphs presented in Figure 4.4 and consists of the following steps:

1. Initialization of variable nodes. Each variable node connected to an observation node initializes the value of all of it's messages to the message it receives from the observation node:

$$\begin{aligned} m(b_S[n], \cdot) &= LLR^{l}_{SD}[n], \quad \forall n \in \{1, 2, \ldots, N_S\} \\ m(b_{R_i}[n], \cdot) &= LLR^{l+1}_{R_iD}[n], \quad \forall n \in \mathcal{T}_i \end{aligned} \tag{4.35}$$

   Relays' information bits do not have the observation nodes, and their messages are initialized to zeros:

$$m(b_{Q_i}[n], \cdot) = 0, \quad \forall n \in \mathcal{L}_i \tag{4.36}$$

   The iteration counter is set to 1, denoting that the first iteration begins.

2. Computation of the check-node messages. Each check node n_C connected to P other nodes, $n_1, n_2, \ldots, n_P$ (variable or observation nodes in this case) calculates its messages $m(n_C, n_i)$ to each of the connected nodes n_i in the following manner:

$$\tanh\left(\frac{m(n_C, n_i)}{2}\right) = \prod_{\substack{j=1 \\ j \neq i}}^{j=P} \tanh\left(\frac{m(n_j, n_C)}{2}\right), \quad \forall i \in \{1, 2, \ldots, P\}, \tag{4.37}$$

   where $m(n_j, n_C)$ denotes the message sent from the node n_j to the node n_C in the previous iteration. If the maximum number of iterations has been reached, the algorithm proceeds to step 4. Otherwise, it proceeds to step 3. An alternative is to have

an "early termination" block, that checks if all the check-sums are satisfied, given the current soft-bit values. If they are satisfied, the algorithms declares that the correct codeword has been found and proceeds to step 4.

3. Computation of the variable-node messages. Each variable node n_V connected to P other nodes, $n_1, n_2, \ldots, n_P$ (check nodes, as well as the observation nodes, if any) calculates its messages $m(n_V, n_i)$ to each of the connected nodes n_i in the following manner:

$$m(n_V, n_i) = \sum_{\substack{j=1 \\ j \neq i}}^{j=P} m(n_j, n_V), \quad \forall i \in \{1, 2, \ldots, P\}, \tag{4.38}$$

where $m(n_j, n_V)$ denotes the message previously sent from the node n_j to the node n_V. Note that the observation nodes that receive messages from the variable nodes simply disregard them, since they always send out the same message, according to (4.35). After this step, the iteration counter is incremented and the algorithm goes back to step 2.

4. Computation of the soft-bit estimates of the decoded bits. This step resembles step 3), but instead of calculating messages sent to the other nodes, each variable node calculates its final soft-bit estimate. Variable node n_V connected to P other nodes, $n_1, n_2, \ldots, n_P$ (check nodes, as well as the observation nodes, if any) calculates its final value LLR_{n_V} as:

$$LLR_{n_V} = \sum_{j=1}^{j=P} m(n_j, n_V), \quad \forall i \in \{1, 2, \ldots, P\} \tag{4.39}$$

These LLR values are then turned to the hard bit-estimates to get the codewords that were transmitted by the source and the relays, $\{\hat{\underline{b}}_S^{l,m}, \hat{\underline{b}}_{R_1}^{l+1,m}, \ldots, \hat{\underline{b}}_{R_N}^{l+N,m}\}$.

4.4.4 Interleaving and Modulation

The purpose of re-interleaving and re-modulating the decoded bits is to be able to cancel the interference that these symbols caused to other transmitted symbols due to the MIMO channel.

Bit-matrices of the decoded bits $\{\hat{B}_S^l, \hat{B}_{R_1}^{l+1}, \ldots, \hat{B}_{R_N}^{l+N}\}$ are interleaved into a new set of bit-matrices $\{\hat{B}_{S,INT}^l, \hat{B}_{R_1,INT}^{l+1}, \ldots, \hat{B}_{R_N,INT}^{l+N}\}$ through frame-by-frame multiplications with the permutation matrices $\Pi_S, \Pi_{R_1}, \ldots, \Pi_{R_N}$:

$$\begin{aligned} &[(\underline{b}_{S,INT}^{l,1})^T, \ldots, (\underline{b}_{S,INT}^{l,C_S})^T] = \Pi_S \cdot [(\underline{b}_S^{l,1})^T, \ldots, (\underline{b}_S^{l,C_S})^T] \\ &[(\underline{b}_{R_i,INT}^{l+i,1})^T, \ldots, (\underline{b}_{R_i,INT}^{l+i,C_S})^T] = \Pi_{R_i} \cdot [(\underline{b}_{R_i}^{l+i,1})^T, \ldots, (\underline{b}_{R_i}^{l+i,C_S})^T], \end{aligned} \tag{4.40}$$

where $\hat{B}_T^l = [\hat{\underline{b}}_T^{l,1}, \ldots, \hat{\underline{b}}_T^{l,C_S}]^T$ and $\hat{B}_{T,INT}^l = [\hat{\underline{b}}_{T,INT}^{l,1}, \ldots, \hat{\underline{b}}_{T,INT}^{l,C_S}]^T$, for $T \in \{S, R_1, \ldots, R_N\}$. We modulate the interleaved bits to a new set of QAM symbols, $\{\underline{\hat{x}}_S^l, \underline{\hat{x}}_{R_1}^{l+1}, \ldots, \underline{\hat{x}}_{R_N}^{l+N}\}$:

$$\begin{aligned} \hat{x}_S^l[n] &= \text{mod}(\hat{B}_{S,INT}^l[1,n], \ldots, \hat{B}_{S,INT}^l[C_S,n]), (\forall n) n \in \{1,2,\ldots,N_S\}, \\ \hat{x}_{R_i}^{l+i}[n] &= \begin{cases} \text{mod}(\hat{B}_{R_i,INT}^{l+i}[1,n], \ldots, \hat{B}_{R_i,INT}^{l+i}[C_S,n]), & (\forall n) n \in \mathcal{T}_i \\ 0, & (\forall n) n \in \mathcal{L}_i \end{cases} \end{aligned} \tag{4.41}$$

These symbols are passed back to the MMSE-SIC block, that performs the cancellation of interference they caused to other symbols in frames $F_{l+1}, \ldots, F_{l+N}$.

4.4.5 Successive Interference Cancellation

After the messages $M_l^S, M_l^{R_1}, ..., M_l^{R_N}$ have been decoded, the residual received vectors at the destination should be updated by canceling this known interference.

The residual received signals $\tilde{\underline{y}}_D^{l+1}[n], ..., \tilde{\underline{y}}_D^{l+N}[n]$ get updated by subtracting the decoded messages multiplied by the corresponding channel values:

$$\tilde{\underline{y}}_D^{l+i}[n] = \tilde{\underline{y}}_D^{l+i}[n] - h_{R_iD}^{l+i}[n] \cdot \hat{x}_{R_i}^{l+i}[n], \tag{4.42}$$

Operation (4.42) is performed for all symbols $n \in \{1, ..., N_S\}$, and all relays $i \in \{1, ..., N\}$.

4.5 System Design Example

The goal of this section is to showcase the physical-layer system design of a cooperative-relaying link that can double the rate of a direct source-destination link. This demonstration is performed in two steps. First, we design a direct link and measure the spectral efficiency that can be achieved under the imposed channel conditions and system constraints. After that, we design the cooperative relaying link that doubles the spectral efficiency of a direct system, by deploying design principles described in the previous sections.

We consider a cooperative network with up to $N = 5$ relays. If all N relays are transmitting, the destination would need to have $N+1$ antennas to extract the full multiplexing gain of the MIMO channel. This is the reason why we assume that the destination has $N_{RX} = 6$ receive antennas.

Data is transmitted in packets that consist of frames, as described in Section 4.1. Each data frame is N_S symbols long and consists of $N_S \times C_S$ coded bits, where N_S is the LDPC blocklength and C_S is the number of bits per transmitted QAM symbol. As our design constraints, we consider $N_S \approx 2000$ and $C_S \in \{1, 2, 4, 6\}$.

We assume all channels have block-fading Rayleigh distribution, with 20 independent channel instances per data frame. That means that the channels change approximately every 100 transmitted symbols. We consider the scenario in which the relays are close to the source, with average SNR of 20dB: $h_{SR_i} \sim \mathcal{CN}(0, SNR_{SR})$, $SNR_{SR} = 20dB$. The

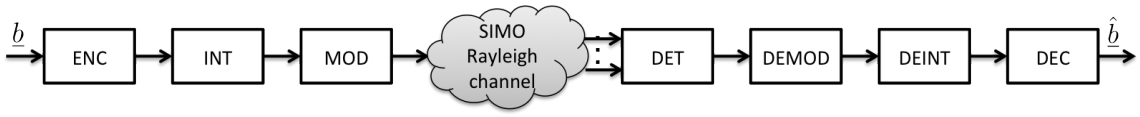

Figure 4.5: Direct-link block diagram.

destination is farther away, and we assume the average SNR between the source and the relays terminals and the destination terminal is 6dB (including the power gain that comes from 6 receive antennas): $\underline{h}_{SD}, \underline{h}_{R_iD} \sim \mathcal{CN}(\underline{0}, \frac{SNR_{SD}}{N_{RX}} \times I)$, $SNR_{SD} = 6dB$. Target BER for the physical layer is 10^{-3}.

4.5.1 Point-to-Point System Design

To establish the baseline link performance, we first design a source-destination link without relays. Compared to the cooperative-relaying block diagram from Figure 4.1, the transmitter at the source stays the same and the receiver at the destination simplifies to a SIMO receiver. We use the matched filter receiver instead of a MIMO detector, simplify the joint decoder to a standard LDPC decoder and eliminate the interference cancellation loop, as shown in Figure 4.5.

Each data frame consists of C_S codewords obtained by using one of the LDPC codes from 802.11n standard [27], interleaved using a block interleaver and then modulated into a series of QAM symbols. The receiver first detects the received signal using a matched filter and then does the inverse operation from the transmitter: demodulates, deinterleaves and finally decodes the transmitted bits.

The ergodic capacity that can be achieved with this channel distribution is approximately 2.25b/s/Hz. Due to a non-Gaussian signaling, a finite code blocklength and the block-fading environment with 20 fades per codeword, the actual spectral efficiency will be lower than the calculated ergodic capacity. The maximum spectral efficiency we measure using the LDPC codes from 802.11n standard and different constellations is 1.5b/s/Hz. It is achieved with the QPSK constellation and the LDPC code with rate $r = 3/4$ and blocklength $N_S = 1944$ from [27]. The targeted BER is measured at the SNR of 5.5dB.

4.5.2 Multi-Relay System Design

Since the direct link achieves the spectral efficiency of 1.5b/s/Hz, the goal of this section is to design a cooperative link that doubles this rate and achieve the spectral efficiency of 3b/s/Hz. This corresponds to the cooperative-multiplexing gain of 2. Design procedure of the cooperative-relaying link consists of the following steps:

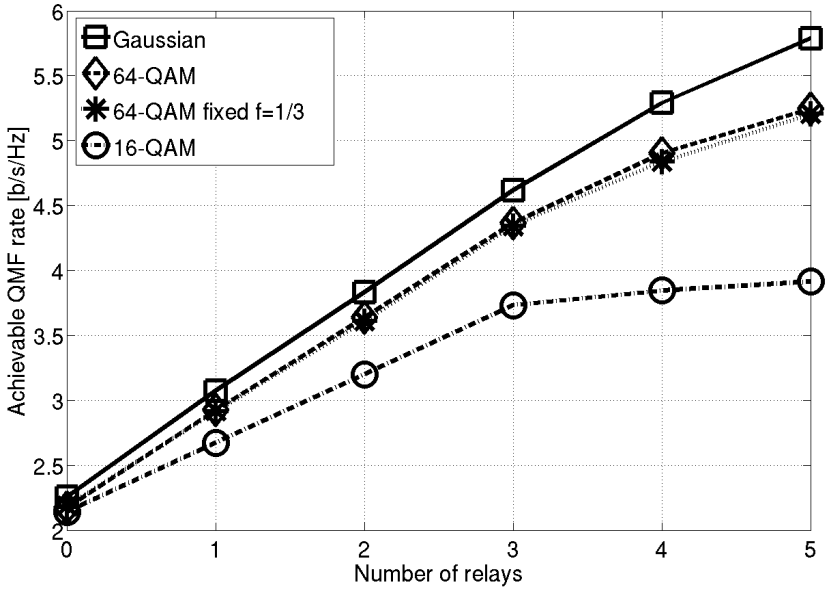

Figure 4.6: Achievable QMF rate scaling with the number of relays for local scheduling scheme and different signaling methods: Gaussian (solid), 64-QAM (dashed) and 16-QAM (dot-dashed). The achievable QMF rate with the 64-QAM signaling and the same listening time of each relay ($f_i = \frac{1}{3}$) is presented in dotted line.

Calculate theoretically achievable rates

To find the theoretically achievable spectral efficiency with multiple relays, we assume Gaussian signaling and deploy the same type of analysis as described in Chapter 3. Since the channels are changing significantly faster than the frame duration, we use the ergodic capacity as a reference point. Theoretically achievable spectral efficiency is presented in Figure 4.6 in solid. We conclude that the rate of a direct link can be doubled (from 2.25b/s/Hz to 4.5b/s/Hz) with 3 cooperative relays.

Schedule the relays and choose constellation size

The next design step is to choose the transmit constellation. According to the design constraints, we can use any of the standard QAM constellations: BPSK, QPSK, 16-QAM or 64-QAM. To choose the best option, in Figure 4.6 we plot the achievable QMF rate for the two constellations that can deliver required spectral efficiency of 3b/s/Hz: 16-QAM and 64-QAM.

Choosing the 64-QAM signaling promises much better performance than 16-QAM, and can theoretically achieve 3b/s/Hz with 2 relays. The reason for 64-QAM performing much better than 16-QAM is that it can achieve higher spectral efficiency (6b/s/Hz with 64-QAM

versus 4b/s/Hz with 16-QAM) and use the proximity gain between the source and the relays more efficiently. This is important, since the proximity gain between the source and the relays is limited by the spectral efficiency of 4b/s/Hz that the 16-QAM constellation can achieve.

In Chapter 3 we have shown that the local scheduling scheme performs very well in the fast-fading channel conditions, even when only average channel values are known. Because of its great performance and low complexity of implementation, we choose the local scheduling scheme to schedule all relays in the showcased cooperative system.

When we calculate the actual listening fractions of the relays with 64-QAM signaling, we get that $f_i \in (0.27, 0.34), \quad \forall i \in \{1, 2, \ldots, N\}$. Having each relay listen for a different amount of time is not very practical, because a different LDGM code would need to be designed for each relay. Since all listening fractions lie close together, we choose a practical solution to schedule all relays to listen for the same amount of time: $f_i = \frac{1}{3}, \quad \forall i \in \{1, 2, \ldots, N\}$. Based on the plot from Figure 4.6, the performance loss is negligible compared to the original local scheduling scheme.

Choose channel codes

Spectral efficiency of 3b/s/Hz can be achieved by using the 64-QAM constellation with the rate $r^{LDPC} = \frac{1}{2}$ LDPC code at the source. The relays are listening for $f = \frac{1}{3}$ of the frame length, and so the code rate of the LDGM codes used at the relays will be:

$$r^{LDGM} = \frac{N_S - N_R}{N_R} = \frac{1}{2} \tag{4.43}$$

An optimal approach to designing channel codes would be to design the LDPC and all LDGM codes jointly, as discussed in Chapter 2 and [41]. Designing up to six (one LDPC and up to five LDGM) codes jointly by searching for good degree distributions, would require a search over all possible degree distributions. We choose to avoid this complex search by following the idea from Chapter 3 of designing the relays independently of each other, i.e. as if there was no other relays in the network.

LDPC and LDGM channel codes can be designed independently for each relay, using the algorithm described in [41]. Degree profiles (λ_S, ρ_S) and (λ_Q, ρ_Q), defined in Chapter 2, must satisfy the following requirements:

$$\begin{aligned} r^{LDPC} &= 1 - \frac{\int_0^1 \rho_S(x)dx}{\int_0^1 \lambda_S(x)dx} = \frac{1}{2} \\ r^{LDGM} &= 1 - \frac{\int_0^1 \rho_Q(x)dx}{\int_0^1 \lambda_Q(x)dx} = \frac{1}{2} \end{aligned} \tag{4.44}$$

Applying the heuristics suggested in [41, 38], such as choosing regular LDPC and LDGM code profiles and limiting maximum degree to 8, we came up with the following LDPC-LDGM profiles:

$$\begin{aligned} \lambda_S(x) &= x^3, \quad \rho_S(x) = x^6; \\ \lambda_Q(x) &= x^6, \quad \rho_Q(x) = x^3, \end{aligned} \tag{4.45}$$

that is, the LDPC code at the source is chosen to be a (3,6) regular code, and the LDGM code at the relay is chosen to be a (6,3) regular code.

Once we designed the LDGM code that works well with the source's LDPC code, we can use that same LDGM code for all relays in the system because all of them are scheduled to listen for the same amount of time, $f_i = \frac{1}{3}$. Since the relays use the scalar bit-to-bit quantizer, we choose the relays' constellation to be the same as the one that the source uses, i.e. 64-QAM.

Measure BER rate with the chosen system parameters

By deploying all of the aforementioned design procedures, we simulate the system presented in Figure 4.1. The BER curves of a cooperative system with 2, 3 and 4 relays are plotted in Figure 4.7. We see that the cooperative relaying link with 3 relays and the spectral efficiency of 3b/s/Hz achieves BER of 10^{-3} at about 5.5dB of average SNR, which is the same SNR needed to operate the direct link with 1.5b/s/Hz spectral efficiency. Note that this result matches the calculation from Section 4.5.2, where we calculated that the cooperative link would need 3 relays to double the spectral efficiency of the direct link under the given channel conditions.

Choose number of DBLAST frames within a packet

In this section we explore how the BER performance and the effective spectral efficiency of the cooperative-relaying link changes with the number of DBLAST frames within a packet. In the simulations results presented in Figure 4.7 we assume that each DBLAST packet has 20 cooperative frames. The effective data rate with 20 cooperative frames within a packet and a three-relay system is calculated using expression (4.2) as:

$$R_{eff} = \frac{20}{23} \times 3\text{b/s/Hz} + \frac{3}{23} \times 1.5\text{b/s/Hz} = 2.8\text{b/s/Hz}. \tag{4.46}$$

We demonstrate that the effective data rate could be increased by increasing the number of DBLAST frames within a packet with virtually no sacrifice in the BER performance. In Figure 4.8 we plot the BER performance of a cooperative link with three relays that we just designed, for a different number of cooperative frames within a packet. Having only one cooperative frame improves the BER performance by 0.2dB, but reduces the effective data rate significantly, to only 1.875b/s/Hz. Having more cooperative frames within a packet is always better, since the effective data rate approaches 3b/s/Hz and the BER penalty is negligible. Maximum value for P is determined by the amount of time the source has for communication with the destination, or the amount of data that it has to transmit.

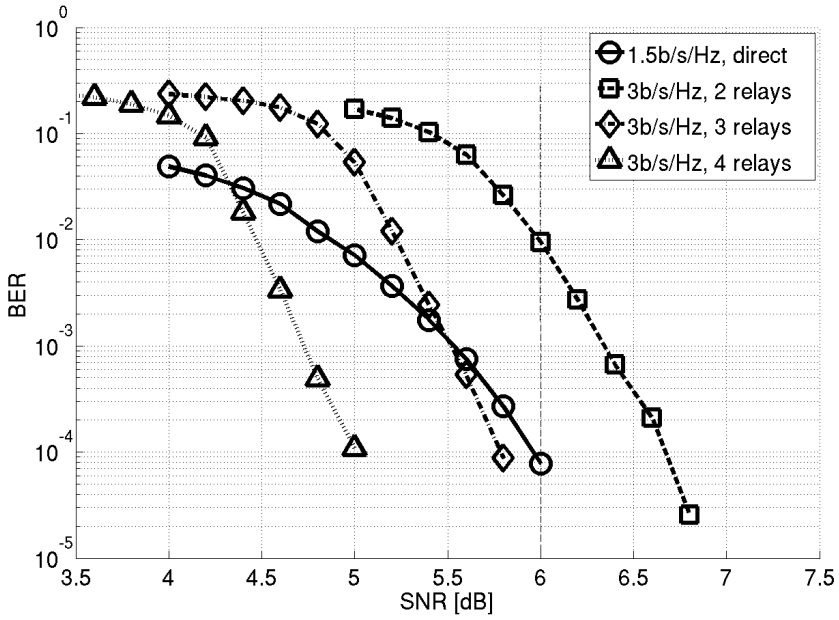

Figure 4.7: Simulated BER curves for the designed links: the direct source-destination link with the spectral efficiency of 1.5 b/s/Hz and the cooperative-relaying link with three relays and the spectral efficiency of 3 b/s/Hz achieve target BER of 10^{-3} at the same SNR.

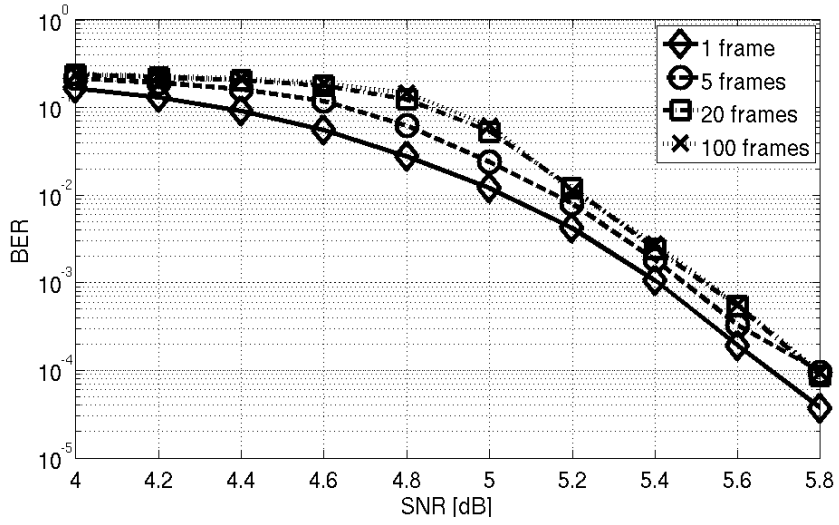

Figure 4.8: Three-Relay link performance with different number of frames per packet.

Cooperative Multiplexing Gain Expressed as SNR Gain

The cooperative-relaying gain does not necessarily need to be measured through the spectral efficiency increase. We use the same design example to present the gain measured by how much we can reduce the transmit power at the terminals and still achieve the same spectral efficiency as with the direct link.

In Figure 4.9 we plot the BER curves for the cooperative-relaying link that we just designed, with different number of relays and the spectral efficiency of 3 b/s/Hz. All relays are still set to listen for the same amount of time, $f_i = \frac{1}{3}$ and all the other design parameters stay the same. We also repeat the design procedure for a direct link, this time with spectral efficiency of 3 b/s/Hz. The best BER curve that achieves this spectral efficiency is obtained by using the LDPC code from 802.11n standard with $r = \frac{3}{4}$ and $N_S = 1944$ and the 16-QAM constellation.

From Figure 4.9 we conclude that the required SNR_{SD} to achieve 3 b/s/Hz efficiency with the direct link is around 12.7 dB. To achieve the same efficiency with the cooperative-relaying link with one, two and five relays, the required SNR_{SD} is 9.2 dB, 6.2 dB and 3.8 dB, respectively. Each of the first two relays reduces the required SNR by about 3 dB, which effectively halves the transmit power at the terminals. By adding five relays, the transmit power of the source and the relays can be reduced by almost 9 dB compared to the direct link. That means almost an order of magnitude power savings for the transmitter at the source!

4.6 Relay-to-Relay Interference Cancellation Techniques

In this section, we provide an idea on how to design physical layer of a general network, the one that does include the relay-to-relay links, based on the design procedure for the no-interference network that we discussed so far. We studied relay-to-relay communication under the QMF and DBLAST framework in Chapter 3. We have shown that using relay-to-relay communication increases the achievable QMF rate by 2-5% compared to a link without the relay-to-relay links. On the other hand, to treat messages transmitted over the relay-to-relay links as useful information drastically increases the complexity of implementation. In particular, a huge problem is that the relays do not receive just the signal from the source, but also from all the other relays that are in the transmit phase.

Because relay-to-relay communication marginally increases the information gain and drastically increases the implementation complexity, it may be instructive to treat it as *interference*, rather than useful information. This idea fits well into the DBLAST framework, since that interference originates from the previous frames that have already been decoded at the destination and therefore can be cancelled. In Chapter 3 and [28], it is argued that the relay-to-relay interference can be captured perfectly at the relays, due to quantization at the noise level. After the interference cancellation is performed, the general

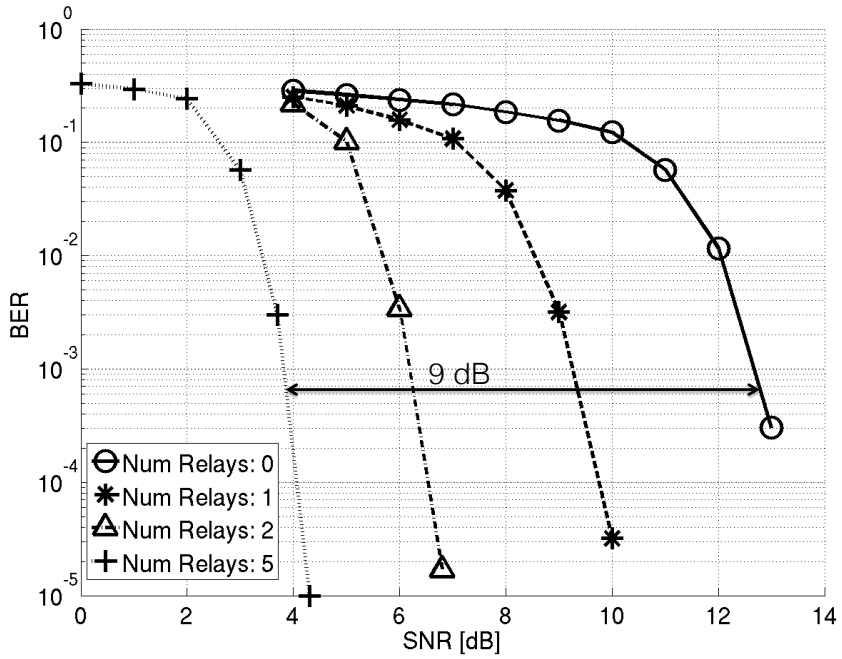

Figure 4.9: BER curves for a direct link and a cooperative-relaying link with different number of relays and the same spectral efficiency of 3 b/s/Hz.

network with relay-to-relay links simplifies to exactly the no-interference network model that we used throughout this chapter.

4.6.1 Relay-to-Relay Interference with QAM Signaling

Canceling the relay-to-relay interference is, however, not as straightforward when terminals use QAM modulation. According to the quantization procedures presented in Section 4.3.3, the relays are quantizing the received symbols into exactly C_S bits, i.e. as many bits as it takes to represent the transmitted symbol at the source. This intuitively makes sense because information that the relays receive and are supposed to forward to the destination is discrete, and the number of quantization bits reflects that. In the general network scenario, information that the relays receive is *continuous* because it is a mix of many transmitted symbols (from the source and the transmitting relays) that all pass through different physical channels. That means that quantizing the received symbol $\underline{y}^l_{R_i}[n]$ into C_S bits will not capture this information perfectly.

In particular, if we denote the channel coefficient between relays R_j and R_i with $h^l_{R_jR_i}[n]$, where $i, j \in \mathcal{R}$, $l \in \{1, 2, \ldots, P\}$ and $n \in \mathcal{T}_j \cap \mathcal{L}_i$, then the received signal at relay R_i changes from the one given by (4.1), to:

$$y^l_{R_i}[n] = h^l_{SR_i}[n]x^l_S[n] + \sum_{\substack{j\in\mathcal{R}\\ j\neq i}} h^l_{R_jR_i}[n]x^l_{R_j}[n] + z^l_{SR_i}[n], \quad \forall n \in \mathcal{L}_i. \tag{4.47}$$

We can apply the same matched filter receiver at the relay R_i, as described in (4.14). Now the received symbol $\hat{y}^l_{SR_i}[n]$ becomes:

$$\begin{aligned}\hat{y}^l_{SR_i}[n] &= \frac{h^l_{SR_i}[n]^H \cdot y^l_{R_i}[n]}{|h^l_{SR_i}[n]|^2} \\ &= x^l_S[n] + \sum_{\substack{j\in\mathcal{R}\\ j\neq i}} \frac{h^l_{SR_i}[n]^H \cdot h^l_{R_jR_i}[n]}{|h^l_{SR_i}[n]|^2} x^l_{R_j}[n] + \hat{z}^l_{SR_i}[n] \\ &= x^l_S[n] + x^l_{interf,i}[n] + \hat{z}^l_{SR_i}[n], \quad \forall n \in \mathcal{L}_i.\end{aligned} \tag{4.48}$$

Because each relay has only one antenna, there is no good way to perform a reliable estimation of the signal from the source. According to the scheme introduced in Chapter 3, relays should not try to cancel this interference from other relays, but rather quantize it and forward it to the destination. However, there are two problems with this approach:

1. The dynamic range problem. If any of the coefficients $h^l_{R_jR_i}[n]$ is significantly larger than $h^l_{SR_i}[n]$, then signal $\hat{y}^l_{SR_i}[n]$ would need to be quantized into much more than just C_S bits, to properly capture interference term that dominates the sum. Because $h^l_{SR_i}[n]$ can get arbitrarily small, the quantizer would likely saturate in some instances.

2. The accuracy problem. Depending on which accuracy is required for interference cancellation at the destination, the received signal $\hat{y}^l_{SR_i}[n]$ would need to be quantized finer than C_S bits because the interference term $x^l_{interf,i}[n]$ is, unlike the source's symbol $x^l_S[n]$, a continuous signal.

In the next section, we offer a special type of relay quantization that can solve the first problem above and adjust to any desired accuracy of quantization.

4.6.2 Modified Quantization at the Relay: Modulo-QMF

The problem with the dynamic range of the received symbols at the relays given by (4.48) is that the interference term, $x^l_{interf,i}[n]$, can take unpredictably high values due to division of the two channel coefficients. When this situation occurs, the received symbol $\hat{y}^l_{SR_i}[n]$ would need to be quantized into unpredictably many quantization bits. To make sure that the interference term is properly captured, there are two obvious solutions: 1) a dynamic quantization scheme, which changes the number of quantization bits according to the current channel values, or 2) a back-off quantization scheme, which always assumes the worst case scenario and quantizes the received symbols into a number of quantization bits that is statistically enough to represent the interference term.

Intuitively, neither of the above two choices is very practical. Dynamic change of quantization at the relay could introduce huge communication overhead, because all changes in the quantization scheme must be reported to the destination. Because quantization is at the symbol level, different receive symbols would be quantized to different number of bits, depending on the instantaneous channel conditions. Also, changing the quantization scheme also changes the relay scheduling, LDGM code, etc. The second scheme would simply introduce too much loss whenever the interference term is not large.

The scheme that we present does not rely on any of the above two suggestions. The idea is similar to the Tomlinson-Haroshima precoding for channels with known symbol interference [57, 24], and uses the fact that the interference term $x^l_{interf,i}[n]$ is known at the receiver (the destination), and that the transmitted signal $x^l_S[n]$ is discrete and belongs to a specific QAM constellation.

To remove the accuracy problem for a moment, we assume that the relay can quantize any received symbol $\hat{y}^l_{SR_i}[n]$ very accurately, but only below the signal level of a source symbol $x^l_S[n]$. As a matter of fact, we propose that the relay does not quantize at all above the signal level of $x^l_S[n]$. Instead, it performs modulo operation on the received symbol $\hat{y}^l_{SR_i}[n]$, which shifts the result $\tilde{y}^l_{SR_i}[n]$ to the range of QAM constellation used at the source, as shown in Figure 4.10. We call this quantization *Modulo-QMF*, because it essentially consists of the two steps: modulo operation and then the quantization.

$$\tilde{y}^l_{SR_i}[n] = \mathrm{mod}_{C_S}(\hat{y}^l_{SR_i}[n]) \tag{4.49}$$

The result of the modulo operation, $\tilde{y}^l_{SR_i}[n]$, is quantized into C_S+C_Q bits, where $C_Q \geq 2$ to capture at least one additional bit per each dimension, beyond what is occupied by the source bits. The quantized bits are then taken through the relay's transmit chain, in the same way as explained in Section 4.3.

4.6.3 Cancellation of Relay-to-Relay Interference

The destination does an inverse operation to the modulo operation performed at the relays. At the time when the bits that belong to the source symbol $x^l_S[n]$ are to be decoded, the destination already knows the interference term $x^l_{interf,i}[n]$, because it comes from the messages that have already been decoded. In addition to that, the destination received soft-bit estimates of the term $\tilde{y}^l_{SR_i}[n]$. Based on these two values, it can properly cancel the interference term $x^l_{interf,i}[n]$ and end up with the same relationship as if there was no interference between the relays:

$$\mathrm{mod}_{C_S}\left(\tilde{y}^l_{SR_i}[n] - x^l_{interf,i}[n]\right) = x^l_S[n] + \tilde{z}^l_{SR_i}[n], \tag{4.50}$$

where

$$\tilde{z}^l_{SR_i}[n] = \mathrm{mod}_{C_S}(\hat{z}^l_{SR_i}[n]) \tag{4.51}$$

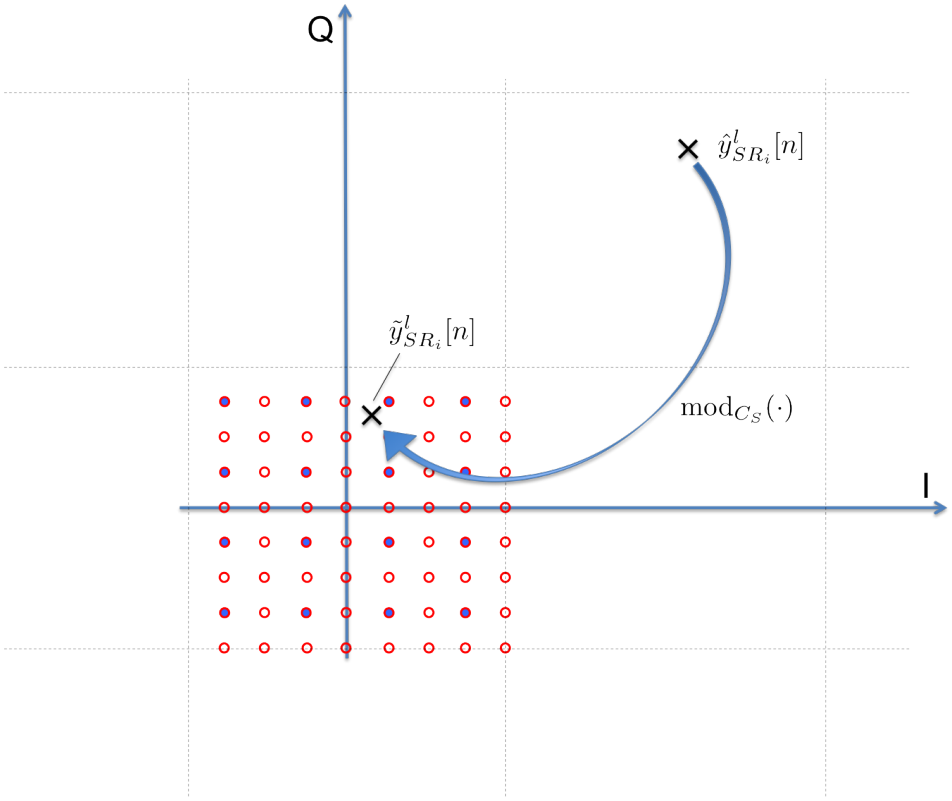

Figure 4.10: $\mathrm{mod}_{C_S}(\cdot)$ operation presented on the 16-QAM example and with $C_Q = 1$. The received symbol $\hat{y}^l_{SR_i}[n]$ is first shifted by modulo operation to fall into the the quantization range, and then quantized using 6 bits (instead of regular 4 bits for 16-QAM).The blue circles represent the 16-QAM constellation points, and the red-lined circles — the quantization levels.

This means that after the destination cancels the interference from other relays, it can proceed with the "standard" Q-node operation that we discussed previously in this chapter. In terms of the joint Tanner graph used for joint decoding, it looks the same as the one presented in Figure 4.4, but with "modified" Q-nodes. The modification of Q-nodes is presented in Figure 4.11. To cancel the interference from other relays, the source's signal $x^l_S[n]$ would have to be recreated at the Q-node, as well as the quantized version of the received signal at the relay, $\tilde{y}^l_{SR_i}[n]$.

To receive updates for source variable nodes, the interference is subtracted from the recreated quantized received symbol at the relay, and then the result is demodulated and passed through the "standard" Q-nodes to calculate the source bits' LLRs, as presented on the left part of Figure 4.11. The updates for relay variable nodes are calculated in the opposite fashion — the source symbol is recreated by modulating the source variable bits, then the interference is added and the resulting received symbol is demodulated and

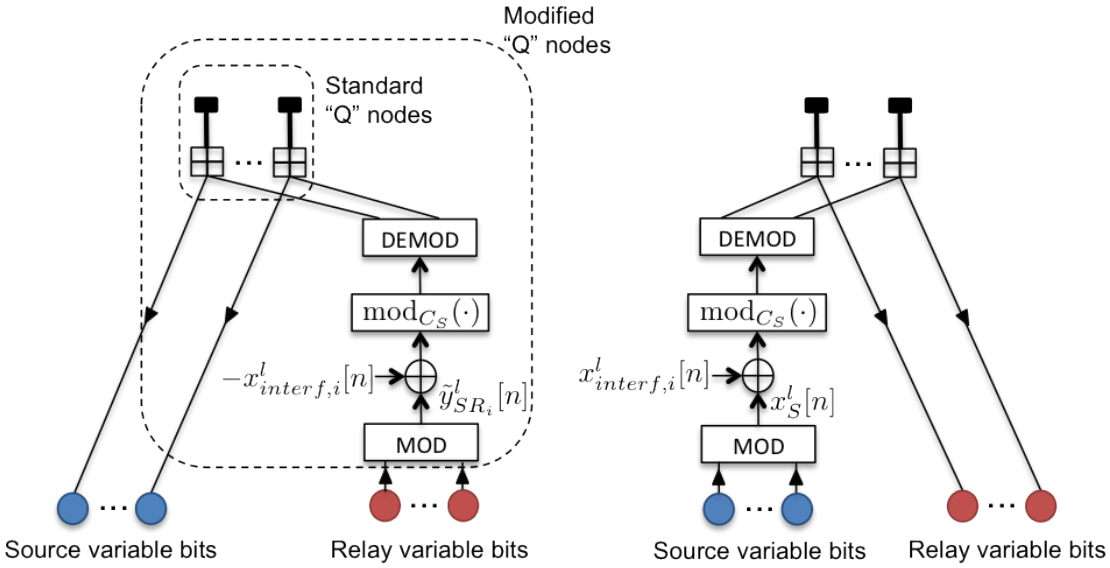

Figure 4.11: Computation of the source variable bit updates (left) and the relay variable bit updates (right) in a joint factor graph with the relay-to-relay interference cancellation.

passed through the "standard" Q-nodes to calculate the relay bits' LLRs, as presented on the right part of Figure 4.11. The modulation and demodulation inside the modified Q-nodes require either the symbol-level interleaving or joint decoding of multiple codewords with bit-interleaving.

4.7 Summary

In this chapter we show the physical-layer system design procedure for a multi-relay cooperative link. The intuition we developed for the relay scheduling analysis, to treat relays independently from each other has been applied to the system design as well. We demonstrate the benefits of such design approach by showcasing a design procedure on an example cooperative link that targets to double the spectral efficiency of a direct link. It is showed that the proposed signal processing algorithms provide approximately the same spectral efficiency increase as predicted by the theoretical analysis. In the end we present an idea for including the relay-to-relay interference cancellation scheme suggested in Chapter 3 into the joint decoding framework. This is enabled by using the Modulo-QMF scheme at the relays, instead of the regular QMF scheme.

Chapter 5

Hardware Implementation of a Single-Relay QMF Cooperative Link

The goal of this chapter is to showcase hardware implementation of a single-relay cooperative link, based on the signal processing algorithms developed in Chapter 4. The hardware implementation of the digital baseband is done on the National Instruments PXIe platform for communication system development. Terminals are implemented on field-programmable gate array (FPGA) chips, and the over-the-air channels between the terminals are emulated in software as baseband-equivalent channels. We implement all three terminals of a single-relay link — the source, the relay and the destination, each on a separate FPGA, as shown in Figure 5.1. We also implement a direct source-destination link for complexity and performance comparison, shown in Figure 5.2.

We describe the hardware architecture of all signal processing blocks presented in Chapter 4, with the special focus on the MMSE-SIC MIMO detector and the joint LDPC-LDGM decoder because these two blocks are by far the most complex and the most resource consuming. The implementation of the MMSE-SIC detector for the cooperative-relaying receiver with DBLAST is based on the square-root algorithm, which enables efficient computation of the MMSE-SIC filter coefficients [25, 29]. The implementation of the joint LDPC-LDGM decoder is a serial implementation of the belief-propagation algorithm [31]. The same architecture can be used on any Tanner-graph based code, so the same decoder can be used both as an LDPC decoder for a direct-link receiver and an LDPC-LDGM decoder for a cooperative-relaying receiver.

We express the difference in implementation complexity between the direct link and the single-relay cooperative link through the basic FPGA resources: logic slices, multiply-accumulate (MAC) units and block RAM (BRAM) blocks. We show that the destination receiver in the single-relay link is about 40% more complex than the destination receiver in the direct link. This result compares only the back-end baseband complexity, i.e. it does not take into account synchronization, channel estimation and equalization. The measured spectral efficiency of the cooperative-relaying link is 33% better than the spectral efficiency of the direct link, which means that the spectral efficiency increase and the receiver complexity

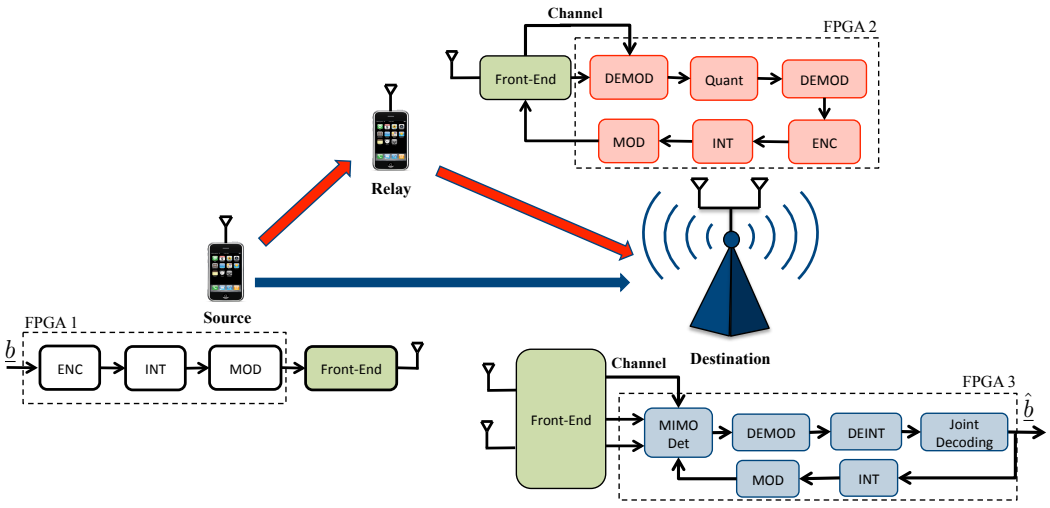

Figure 5.1: Single-relay cooperative link implementation. The source, relay and destination basebands are each implemented on a different FPGA chip.

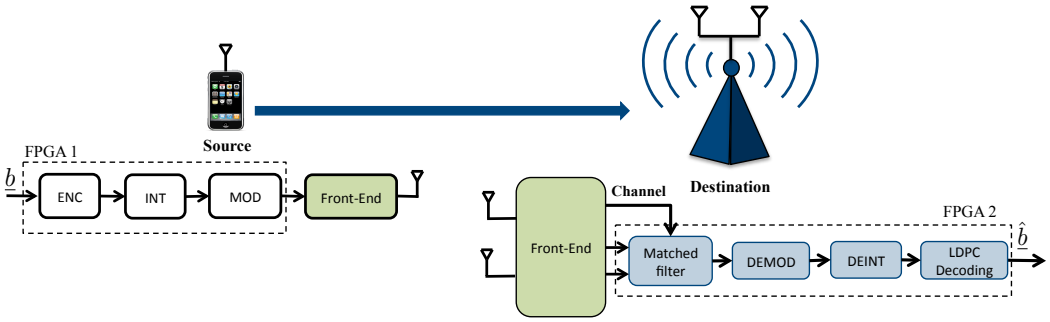

Figure 5.2: Direct-link implementation. The source and destination basebands are implemented using two FPGA chips.

trade-off almost evenly. The complexity of the source's transmitter is, as expected, not influenced by the introduction of the relays.

5.1 Transmitter at the Source

In terms of the hardware architecture, the transmit baseband chain at the source is the same for the direct and the cooperative-relaying scenarios and the same as the transmit chain at the relay, as shown in Figures 5.1 and 5.2. It consists of a block encoder (LDPC or LDGM at the source and the relay, respectively), a bit-interleaver and the QAM modulator. We use the same notation as introduced in Chapter 4.

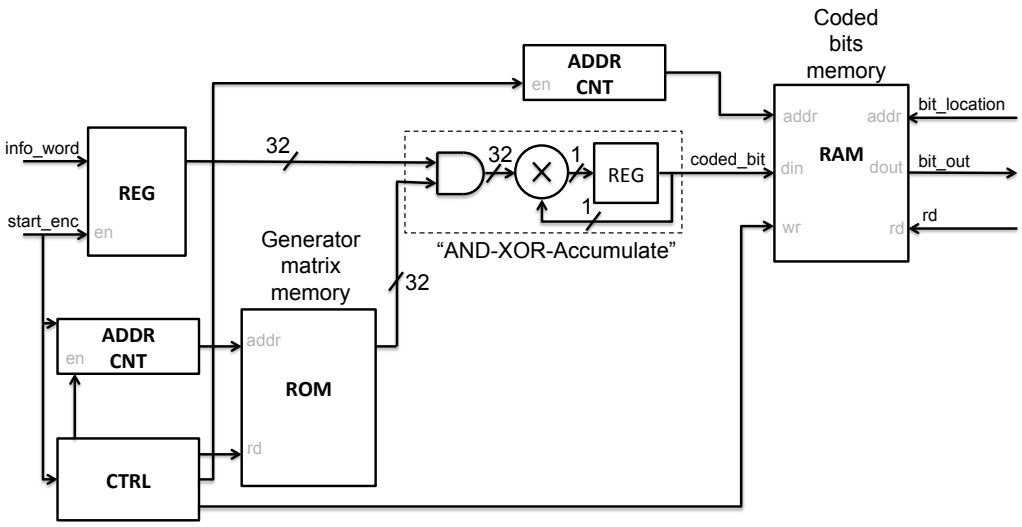

Figure 5.3: Hardware architecture of the block encoder, specified by its generator matrix. The information word is XOR-ed with the appropriate column from the generator matrix to produce a coded bit.

5.1.1 Block Encoder

Block encoder essentially implements a vector multiplication operation, by multiplying input information words $\{\underline{b}_I^{l,m}\}$ with the generator matrix G_S. Because the elements are binary, all operations are over the binary Galois Field, GF(2). That means that the multiplication simplifies to an AND operation and the addition to an XOR operation. The information word vector is bitwise multiplied by the column of a generator matrix, then the results are added to produce a corresponding coded bit.

Hardware block diagram of the block-encoder is shown in Figure 5.3. The "start_enc" signal starts the encoder block and registers the input information word. Multiplication of the information word vector with the generator matrix column is implemented using a MAC unit over GF(2), which simplifies to bitwise AND operation and accumulation by XOR-ing the results to produce a single coded bit. Because the information word length is typically much larger than the size of the memory word, the information word is XOR-ed with the generator matrix column in 32-bit sections. In that case the XOR block becomes XOR-accumulate block, as presented in Figure 5.3. The coded bits are written to a memory, to be read by the next block in the transmit chain.

5.1.2 Bit-Interleaver

The goal of the bit-interleaver block is to shuffle the encoded bits before modulation, so that bits that belong to the same codeword get modulated to different QAM symbols at different bit-locations. That way it is ensured that all codewords in the source data frame

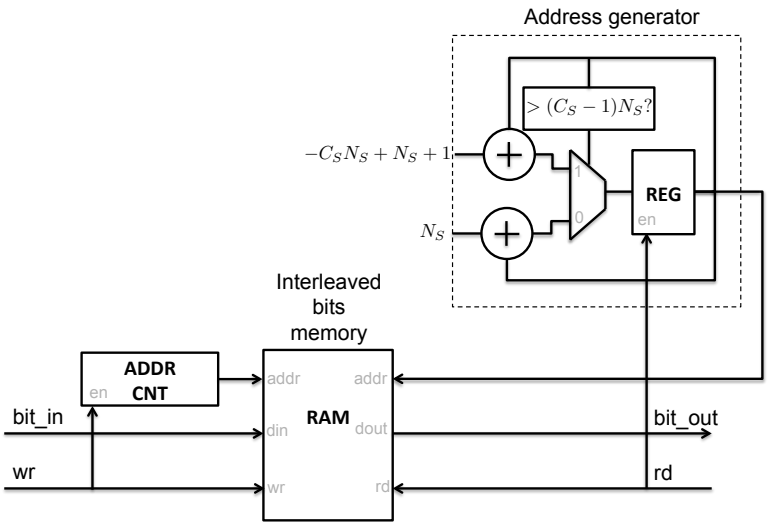

Figure 5.4: Bit-interleaver with serial input and serial output. Bits are written into the memory using a simple counter as an address generator. They are read out of the memory so that consecutive bits in a symbol do not belong to the same codeword.

see the same equivalent BIAWGN channel, according to the BICM scheme [7]. Hardware block diagram of the bit-interleaver implemented in the transmitter is presented in Figure 5.4. The size of the interleaver memory corresponds to the number of bits in the source data frame: $C_S \cdot N_S$.

Encoded bits are written to the interleaver memory such that bits that belong to the same codeword reside at the consecutive memory locations. Bits are read from the memory in steps that are exactly equal to the blocklength of the code (denoted as N_S in Figure 5.4). This means that consecutive output bits, which are passed to the QAM modulator in groups of C_S bits, do not belong to the same codeword. The read-address counter gets incremented by N_S until it is about to reach the maximum number of bits in the frame. Then it rolls back to the beginning of the interleaver memory, according to the structure shown in Figure 5.4. The address generator unit is designed such that the next group of C_S bits starts with the bit from the codeword which is not the same as the one the previous QAM symbol started with. That way it is ensured that bits from the same codeword take different bit-locations within QAM symbols, which is essential for the BICM operation.

5.1.3 QAM Modulator

QAM modulator is implemented as a simple look-up table (LUT), with input bits serving as an address to the memory of stored QAM symbols, as shown in Figure 5.5. Up to 6 coded bits are concatenated to the 2-bit constellation code and then the corresponding QAM symbol is read from the LUT. The output QAM symbol x_S is represented using a

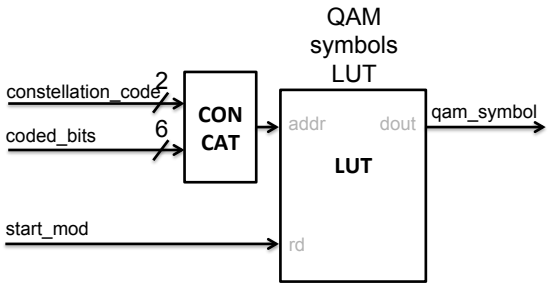

Figure 5.5: QAM modulator, implemented as a simple LUT.

fixed point format, such that $x_S \in (-1, +1)$. There are 4 constellation codes, corresponding to 4 QAM modulations used in the design: BPSK, QPSK, 16-QAM and 64-QAM. The choice of the modulation schemes has been driven by both the LTE and the 802.11n WiFi standards, which use these four schemes [33, 27].

5.2 Receiver at the Relay

Relay's receiver block diagram is shown in Figure 5.1, with the same notation as introduced in Chapter 4. The baseband receiver chain consists of the matched filter, the quantizer and the bit-deinterleaver.

5.2.1 Matched Filter

The matched filter has been shown in Chapter 2 to be an ideal SIMO receiver. Since the relay has a single antenna, the filter consists of a single tap that multiplies the received signal:

$$\hat{y}_{SR} = \frac{h_{SR}^H}{|h_{SR}|^2} \cdot y_R \tag{5.1}$$

The key processing step is a real division by $|h_{SR}|^2$. It has been implemented using a divider embedded into a design tool, which relies on the coordinate rotation digital computer (CORDIC) algorithm to perform numerical computation of the ratio of two real numbers. A detailed presentation of CORDIC algorithm is given in Section 5.4. It takes two real multiplications and an addition to compute the term $|h_{SR}|^2$, before it is passed to the real divider, as shown in Figure 5.6. The result is multiplied with the real and negated imaginary values of the channel coefficient, to get respectively the real and imaginary parts of the matched filter coefficient.

The channel equalization is implemented in Figure 5.6 as a single-tap complex FIR filter. All arithmetic blocks are implemented using fixed point operations, and represented with enough bits to support SNR values: $SNR_{SR} \in [-10\text{dB}, 30\text{dB}]$. The output symbol estimate

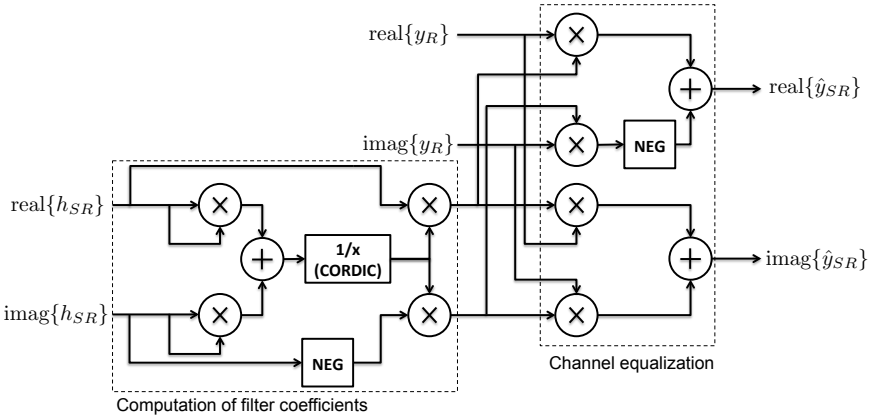

Figure 5.6: Matched filter implemented using real multiplications and division.

$\hat{y}_{SR}$ lies within the transmit QAM constellation and therefore has limited energy, which defines the number of bits for its representation.

5.2.2 Quantizer

We have already discussed two ways to implement scalar quantizer in Chapter 4. The hard-bit demodulator is slightly less complex to implement than the soft-bit demodulator (implemented using a max-log approximation) followed by the single-bit quantizer. The purpose of the hard-bit demodulator is to find the bit sequence that corresponds to the QAM symbol closest to the received symbol $\hat{y}_{SR}$.

QAM symbols for any of the four constellations are in principle irrational values, clipped to the certain fixed-point precision. To find the closest QAM symbol, however, we do not need to go beyond the number of bits used to represent a symbol in that constellation. This is because each 2^{C_S}—QAM symbol starts with the different C_S bits in the chosen fixed-point representation. Therefore, by looking into the first C_S bits of the received symbol $\hat{y}_{SR}$ we can determine which QAM symbol is the closest one, as shown in Figure 5.7. This operation is done separately on the real and imaginary part of the received symbol. After the closest QAM symbol has been detected by slicer, the bit sequence that corresponds to that QAM symbol is read from the two (equivalent) LUT tables.

5.2.3 Bit-Deinterleaver

Bit-deinterleaver is supposed to order interleaved bits back to the original data frame structure, with consecutive bits belonging to the same codeword. Because the functionality of the bit-deinterleaver is exactly opposite to that of the bit-interleaver, they can be implemented in exactly the same way, but with memory read and write controllers interchanged, as shown in Figure 5.8.

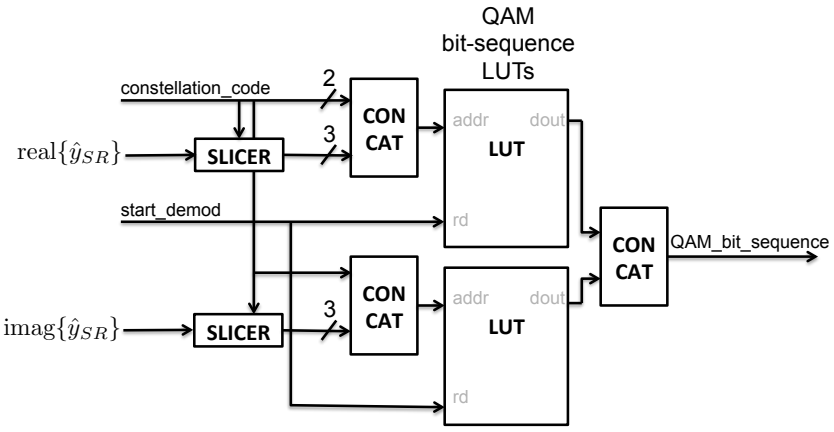

Figure 5.7: Quantizer implemented as a hard-bit demodulator. Slicing the received signal to the top few bits determines the quantized version of the closest QAM symbols in the corresponding constellation.

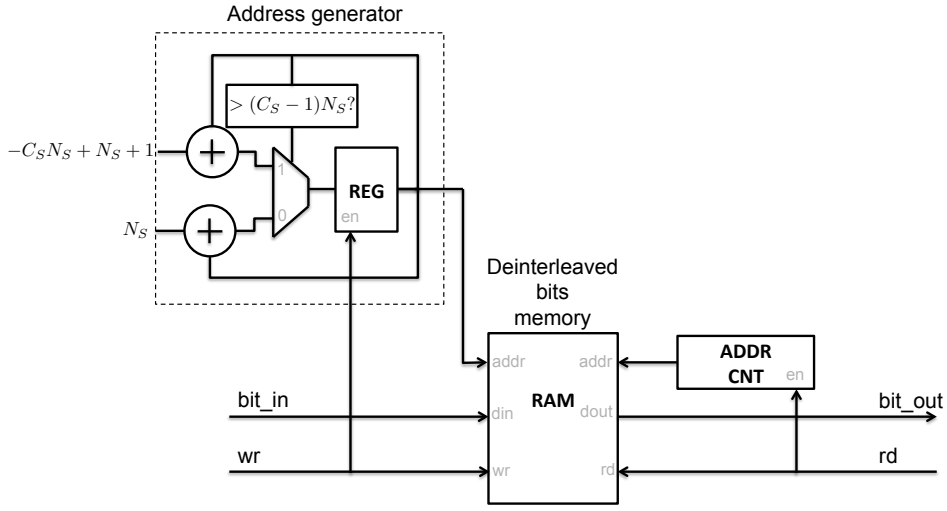

Figure 5.8: Bit-deinterleaver has exactly the same structure as the bit-interleaver, with swapped address counters between the read and write side of RAM memory.

5.3 Receiver at the Destination

5.3.1 Direct-Link Receiver

Block diagram of a direct-link receiver at the destination is shown in Figure 5.2. It consists of the matched filter, the soft-bit demodulator, the deinterleaver and the LDPC decoder.

The matched filter has the same architecture as the matched filter for the relay's receiver from Figure 5.6. Because the destination has two receive antennas, all summation blocks from Figure 5.6 are replaced with accumulators, to include samples from both antennas. The deinterleaver at the destination has exactly the same architecture as the deinterleaver at the relay. Soft-bit demodulator is presented next, and the LDPC decoder has a whole Section 5.5 devoted to it.

5.3.2 Soft-Bit Demodulator

We established in Chapter 2 that calculating exact soft-bit estimates requires a very complex processing. For hardware implementation, typically a max-log approximation is chosen, for it's simplicity in terms of avoiding "exp" and "log" operations:

$$LLR_{TD}[m] = \frac{SINR}{2}\left(\min_{x\in\mathcal{X}_S, b_m=0}\left\{|\hat{y}_{TD}-x|^2\right\} - \min_{x\in\mathcal{X}_S, b_m=1}\left\{|\hat{y}_{TD}-x|^2\right\}\right) \tag{5.2}$$

The main processing step in the above expression includes finding the two closest symbols from the constellation $\mathcal{X}_S$ to the equivalent received symbol $\hat{y}_{TD}$, such that one of them has the bit at position m, $b_m = 1$, and the other $b_m = 0$. This functionality is again implemented using LUTs, just like in the case of the hard-bit demodulator. The hardware implementation is the same as the one presented in Figure 5.7, but with the updated content of the LUTs.

5.3.3 Cooperative-Relaying Link Receiver

Block diagram of a cooperative-relaying receiver is shown in Figure 5.1. It consists of the MMSE-SIC MIMO detector, followed by the soft-bit demodulator, the deinterleaver and the joint LDPC-LDGM decoder. The max-log soft-bit demodulator and the bit-deinterleaver have already been presented in the previous sections, and exactly the same architecture is used for the blocks at the destination as well. The MMSE-SIC detector and the joint LDCP-LDGM decoder are explained in the next two sections.

5.4 MMSE-SIC MIMO Detector for DBLAST

In Chapter 2 we presented a handful of MIMO detectors and argued that the MMSE-SIC MIMO detector is the best suited architecture for cooperative relaying with DBLAST. Besides having good performance, the MMSE-SIC detector can be implemented in a very

efficient way using the square-root algorithm [25]. This algorithm has been originally developed for VBLAST with stream ordering [25], and then later adapted for DBAST in [29]. This section presents an abridged version of author's earlier work in [29].

While the cooperative-relaying link implemented in this chapter has only one relay, the MMSE-SIC MIMO detector has been designed by keeping in mind possible extension to multiple relays. Therefore, it has a flexible architecture that can support a variable number of transmit and receive antennas. It takes as an input the vector of received symbols $\underline{y}_D$, the matrix of MIMO channel coefficients $H = [\underline{h}_{SD}, \underline{h}_{R_iD}, \ldots, \underline{y}_{R_ND}]$ and the estimated transmitted symbol from the previous message, $\hat{x}_T$ and provides the equivalent symbol estimates $\hat{y}_{TD}$ and the stream SINR, $SINR_{TD}$, for all transmit terminals $T \in \{S, R_1, \ldots, R_N\}$. Frame and symbol indices follow the notation established in Chapter 4 and are omitted throughout this section for notation simplicity.

Block diagram of the implemented MMSE-SIC detector is presented in Figure 5.9. It consists of the following blocks:

- The MMSE-SIC preprocessing unit, which calculates the filter coefficients $\underline{g}_T$ using the square-root algorithm. This block can typically have lower throughput than the MMSE filter, unless the channel is changing very fast. However, the architecture is pipelined and it can be easily extended to support a multi-carrier system, so that many channel instances can be processed at the same time.

- The MMSE-SIC filter unit does the FIR filtering of the received symbol $\tilde{\underline{y}}_{TD}$ and calculation of the parameters necessary for soft bit demodulation, $\hat{y}_{TD}$ and $SINR_{TD}$.

- The interference cancellation (SIC) unit is represented as multiplication of the reconstructed symbol $\hat{x}_T$ with the appropriate channel matrix column, $\underline{h}_{TD}$, and then subtraction from the received vector $\underline{y}_D$ that was previously buffered as described in Chapter 2.

In the rest of this section we provide a brief description of the square-root algorithm for DBLAST and show how it is implemented in hardware using systolic arrays for nulling matrix rows. After that, we show the implementation of the MMSE-SIC filtering unit and the SIC unit.

5.4.1 Square-Root Algorithm

To find the MMSE-SIC filter coefficients according to expressions derived in the previous chapter, one would need to invert complex matrices. As a matter of fact, expressions given by (4.31) suggest that for each of the MMSE-SIC filter vectors, a different matrix would need to be inverted. The key benefit of the square-root algorithm presented in this section is that it computes filter coefficients $\underline{g}_T$ without any matrix multiplications or inversions. To do that, the traditional square-root algorithm explained in [54] has been modified in [25] and then in [29] to accommodate MIMO detection for the DBLAST space-time scheme.

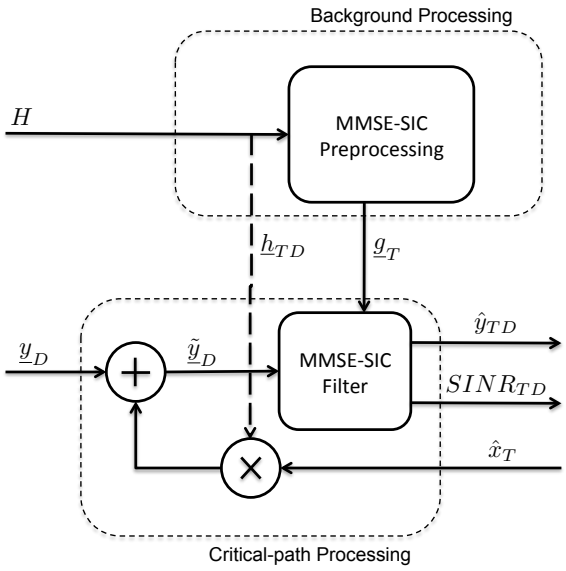

Figure 5.9: Block diagram of the implemented MMSE-SIC MIMO detector.

The computation of filter coefficients $\underline{g}_T$ is done in two steps. First, two auxiliary matrices $P^{1/2}$ and Q are computed. In the second step, we calculate filter coefficients based on the auxiliary matrices. For easier notation, we assume the MIMO channel matrix H has n_t columns and n_r rows, where n_t is the number of transmit terminals during specific symbol time and $n_r = N_{RX}$. We denote the columns of the MIMO channel matrix H with $\underline{h}_i$, where $i \in \{1, 2, \ldots, n_t\}$.

Algorithm for computation of $P^{1/2}$ and Q:

$P^{1/2}$ and Q can be computed using the following recursion, initialized with $P^{1/2}_{|0} = I_{n_t}$ and $Q_0 = 0_{n_r \times n_t}$:

$$\begin{bmatrix} 1 & H_i P^{1/2}_{|i-1} \\ 0 & P^{1/2}_{|i-1} \\ -e_i & Q_{i-1} \end{bmatrix} \Theta_i = \begin{bmatrix} r^{1/2}_{e,i} & 0 \\ K_{p,i} & P^{1/2}_{|i} \\ A_i & Q_i \end{bmatrix}, \tag{5.3}$$

where e_i is the i-th unit vector of dimension n_r (i.e. it is an $n_r \times 1$ vector of all zeros except for the i-th entry which is 1), and Θ_i is any unitary transformation that transforms the first row of the pre-array to lie along the direction of the first unit row vector. After n_r steps the algorithm yields the desired quantities via:

$$P^{1/2} = P^{1/2}_{|n_r} \quad \text{and} \quad Q = Q_{n_r} \tag{5.4}$$

After the auxiliary matrices have been computed, the MMSE-SIC filter vector for the k-th stream is calculated as

$$\underline{g}_k = p_k^{1/2} \underline{q}_k^H, \tag{5.5}$$

where $p_k^{1/2}$ is the k-th diagonal element of $P^{1/2}$, and $\underline{q}_k$ is the k-th column of Q. $\underline{g}_1$ corresponds to $\underline{g}_S$ and $\underline{g}_j$, where $j \in \{2, \ldots, n_t\}$, correspond to $\underline{g}_{R_k}$, where R_k are all relays that are transmitting at the corresponding symbol time.

Next we explain how this process of nulling the whole row of the matrix from expression (5.3) can be done with the sequence of rotations using CORDIC circuits.

5.4.2 Implementation of the Systolic Array

The central part in the preprocessing unit is the row nulling operation given by (5.3). The main idea of nulling the row is to use one column as the pivot, and then apply series of unitary transformations (matrix rotations) that will keep changing only the pivot and the targeted column, until the first (leading) element of the targeted column is nulled [48]. In general, matrix elements are complex because of the complex MIMO channel matrix H, but the very first element in the row we are trying to null is indeed always going to be real. This is true because in expression (5.3) the first element of the first row is 1 in every iteration.

Nulling a Single Element in a Row

Let's denote the complex matrix whose elements we are trying to null with C. To null the i-th element of the first row, $C_{1,i}$, we use the first column as a pivot. In general, this element is a complex number and it can be represented using the Euler form:

$$C_{1,i} = R_{1,i} e^{j\theta_{1,i}} \tag{5.6}$$

Nulling of $C_{1,i}$ is done by nulling the real and imaginary parts in two processing steps. To null the imaginary part of $C_{1,i}$, we rotate this complex number to align with the real axis by multiplying it with $e^{-j\theta_{1,i}}$. This operation is performed using unitary operation Q_{θ_i} (5.7), which is identity matrix with i-th diagonal element replaced by $e^{-j\theta_{1,i}}$.

$$Q_{\theta_i} = \begin{bmatrix} 1 & 0 & \ldots & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & e^{-j\theta_{1,i}} & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & \ldots & 1 \end{bmatrix}. \tag{5.7}$$

The multiplication of the original matrix with this one now results in (5.8):

$$\begin{bmatrix} R_{1,1} & \dots & C_{1,i} & \dots \\ C_{2,1} & \dots & C_{2,i} & \dots \\ \vdots & \dots & \vdots & \dots \\ C_{n,1} & \dots & C_{n,i} & \dots \end{bmatrix} Q_{\theta_i} = \begin{bmatrix} R_{1,1} & \dots & R_{1,i} & \dots \\ C_{2,1} & \dots & C'_{2,i} & \dots \\ \vdots & \dots & \vdots & \dots \\ C_{n,1} & \dots & C'_{n,i} & \dots \end{bmatrix}. \tag{5.8}$$

Note that the first element of the first row was real to begin with, and so we denoted it as $C_{1,1} = R_{1,1}$.

To null the real part of $C_{1,i}$, $R_{1,i}$, we use another type of unitary transformation called Givens transformation, denoted with Q_{ϕ_i}. This transformation is derived again from the identity matrix, by modifying the first and the i^{th} row and column:

$$Q_{\phi_i} = \begin{bmatrix} \cos\phi_i & 0 & \dots & \sin\phi_i & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \dots & \vdots \\ -\sin\phi_i & 0 & \dots & \cos\phi_i & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 1 \end{bmatrix} \tag{5.9}$$

When we apply the Q_{ϕ_i} transformation, we only affect the first and the i-th column. Given that the elements of the first row in these two columns were $R_{1,1}$ and $R_{1,i}$, we can determine the coefficients $\cos\phi_i$ and $\sin\phi_i$ from the following expressions:

$$\begin{aligned} R'_{1,1} &= R_{1,1}\cos\phi_i - R_{1,i}\sin\phi_i \\ 0 &= R_{1,i}\cos\phi_i + R_{1,1}\sin\phi_i. \end{aligned} \tag{5.10}$$

Note that since the transformation is real, the leading element of the updated pivot column ($R'_{1,1}$) stays real. This implies that the updated pivot column can be reused as a pivot column for nulling the next non-zero element of the first row.

We conclude that the two unitary transformations, Q_{θ_i} and Q_{ϕ_i}, are essentially of the same form. The only difference is that Q_{θ_i} is applied to the real and imaginary parts of the column i whose leading element we want to null. Transformation Q_{ϕ_i} on the other hand has to be applied twice, to the real parts of the first and the i-th column, as well as to their imaginary parts.

Therefore, we have managed to null the i-th element of the first row by applying two simple unitary transformations and changing only the elements of the first and the i-th column. This is very important since it promises low complexity implementation - only two columns are involved, not the whole matrix. Of course, in order to null the whole row, these two operation are repeated for each element of the first row.

CORDIC Implementation

We can implement both of the above two unitary transformations using an algorithm for coordinate rotation that is well suited for digital implementation, known as CORDIC (CO-

$$\begin{pmatrix} x \\ y \end{pmatrix} \Leftarrow \cos\xi_0 \begin{pmatrix} x - 2^{-0}\rho_0 y \\ y + 2^{-0}\rho_0 x \end{pmatrix}$$
$$\begin{pmatrix} x \\ y \end{pmatrix} \Leftarrow \cos\xi_1 \begin{pmatrix} x - 2^{-1}\rho_1 y \\ y + 2^{-1}\rho_1 x \end{pmatrix}$$
$$\begin{pmatrix} x \\ y \end{pmatrix} \Leftarrow \cos\xi_2 \begin{pmatrix} x - 2^{-2}\rho_2 y \\ y + 2^{-2}\rho_2 x \end{pmatrix}$$
$$\vdots$$

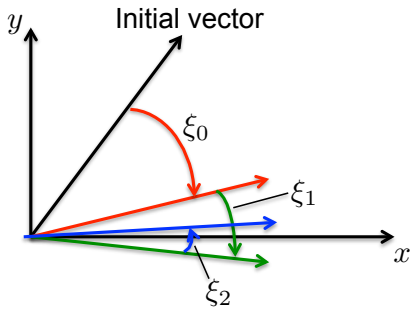

Figure 5.10: Achieving rotation through ξ by using minirotations through the special angles $\pm\xi_\upsilon$.

ordinate Rotation DIgital Computation). The basic idea was first published in the 1950s by Volder [58].

Suppose we want to rotate a point whose coordinates are (x, y) to the new point with coordinates $(x^{'}, y^{'})$, such that the angle between them (measured counterclockwise) is ξ. We have:

$$\begin{aligned} x^{'} &= (\cos\xi)(x - y\tan\xi) \\ y^{'} &= (\cos\xi)(y + x\tan\xi) \end{aligned} \tag{5.11}$$

This involves four multiplications. However, there are many special angles for which some of the multiplications would simplify to shift operations. We will concentrate on the particular angles ξ_υ for which:

$$\tan\xi_\upsilon = \pm 2^{-\upsilon}. \tag{5.12}$$

Then the multiplications by $\tan\xi_\upsilon$ become right-shifts by υ bit positions, which is a very inexpensive operation in hardware. For fixed υ, the two special angles are of the same magnitude but opposite sign and therefore they have the same cosine.

The first step of the CORDIC algorithm is to express any arbitrary angle ξ by a sequence of rotations either forward or backward, by ξ_υ, $\upsilon = 0, 1, ...$ Let $\rho_\upsilon = \pm 1$ determine whether a particular "minirotation" is forward or backward. Thus

$$\xi = \sum_{\upsilon=0}^{\infty} \rho_\upsilon \xi_\upsilon, \tag{5.13}$$

and the rotation of (x, y) through the angle ξ is accomplished by the sequence of rotations by angles ξ_υ, $\upsilon = 0, 1, ...,$ shown in Figure 5.10. The multiplicative factors $\cos\xi_\upsilon$ can be collected together into a single constant:

$$\kappa = \prod_{\upsilon=0}^{\infty} \cos\xi_\upsilon \tag{5.14}$$

which is independent of the overall angle ξ by which we rotate. Thus, we conclude that the CORDIC algorithm is composed of "stages" — most of the stages perform microrotations and the "last" stage performs a gain correction, by multiplying with κ. For any angle we use, all the microrotations are employed in either a clockwise or counterclockwise direction. Although a CORDIC algorithm would seem to call for an infinite number of microrotations to realize rotation by an exact angle, it is practical to use a finite number of stages since the higher numbered stages contribute less and less to the accuracy of the achieved angle. The correction stage is a multiplication by the fixed quantity κ, not a general purpose multiplier. The exact value of κ depends on how many CORDIC stages are used and this dependance is given in [48]. As that work suggests, for ten or more stages κ is essentially saturating to the value of 0.60725.

A CORDIC method therefore achieves rotation without using any of the trigonometric functions and without explicit multiplications. If we knew in advance the angle by which we wished to rotate the pair (x, y), we could determine the set of controls $(\rho_\upsilon, \upsilon = 0, 1, ..., \upsilon_{max})$, each represented by a single bit.

However, in the present application we do not know the angle of rotation in advance. We are given a pair (x, y) and we must rotate that pair through the angle such that the resulting rotated pair takes the form $(x^{'}, 0)$. This operation is called *vectoring*. Then we have to rotate some number of other pairs through the same angle. We have no need to actually know what the angle is, as long as we are able to rotate by that angle. This operation is called *rotating*. For vectoring followed by rotating what we really need is an algorithm to determine the CORDIC controls ρ_υ. A major advantage of the CORDIC algorithm is that the same circuit which is used for vectoring can be used for rotating.

Consider just the first CORDIC stage, for which the special angle is either $45°$ or $-45°$. Since our purpose is to rotate the input (x, y) toward the x axis, if y is "above" the x axis we should rotate "down" and if y is "below" the x axis we should rotate "up", as shown in the right part of Figure 5.10. The angular direction of "down" depends on whether x is positive or negative (if either x or y is zero, we can choose any direction). Therefore,

$$\rho_0 = \text{sgn}(x)\text{sgn}(y). \tag{5.15}$$

Once we have determined ρ_0, we compute the effect of the first stage on x and y and pass the modified (x, y) to the second stage. Here, again, our rule is to rotate "down" if y is "above" the x axis and "up" if it is "below" the x axis:

$$\rho_1 = \text{sgn}(x)\text{sgn}(y), \tag{5.16}$$

and so on. These controls, once determined, are saved in the flip-flops of the specialized stages and used to control those stages for the succeeding (x, y) pairs which are to be rotated through the same angle.

In Figure 5.11 we show the concept of a CORDIC circuit made up of independent stages. The inputs (x, y) are modified by minirotations as they proceed from one stage to another. Only an addition and a subtraction need to be performed in each stage, except for the last

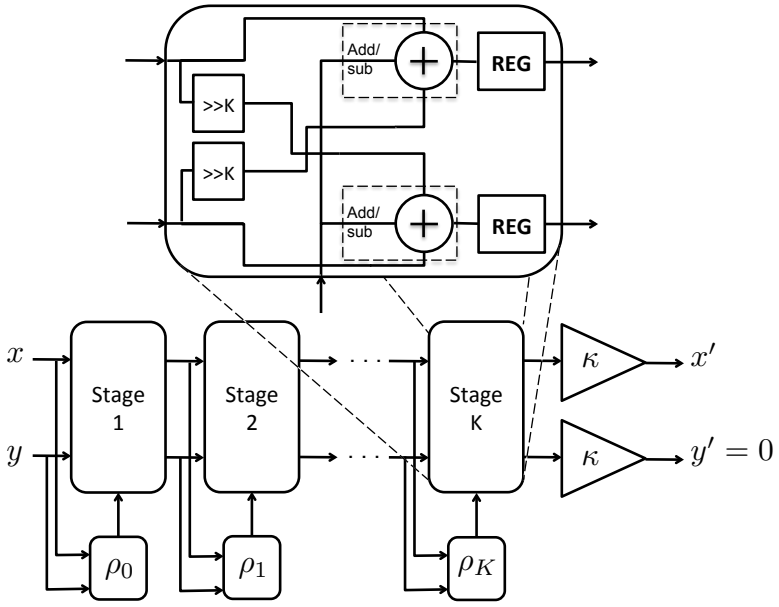

Figure 5.11: Pipeline CORDIC circuit.

stage which is the multiplication by a constant factor given by (5.14). These operations can be carried out very efficiently in digital logic.

This circuit has the virtue of natural pipelining. Since registers are placed between the stages, then a new rotation, involving a new pair (x, y), can be started by the circuit as soon as the preceding pair has been latched at the output of the first stage. Rotation may follow vectoring in this pipelined fashion as long as the rotation angle is the angle determined by the vectoring operation. The only difference between vectoring and rotation is whether the controls ρ_υ are set or remain unchanged as the data passes through the stage.

The performance of the MMSE-SIC detector will depend on the number of stages in the CORDIC. As the SNR gets higher, nulling the first row in equation (5.3) becomes more sensitive to how close to zero those elements are, and in order to achieve better precision, more CORDIC stages are required. To support the SNR values of up to 25dB, we chose CORDIC with $K = 15$ stages.

Hardware Architecture for Nulling One Matrix Element

Earlier in this section we have explained how to null a single element in the first row. Then we explained the CORDIC architecture that we use to null the real leading element (vectoring) and rotate by the same angle the other elements of the same column. Now we put these pieces together and present the hardware architecture for nulling a complex element of the first row.

To null the complex part of the first element in the targeted column, we have to perform the Q_{θ_i} transformation. We assign one CORDIC circuit to perform this transformation. To

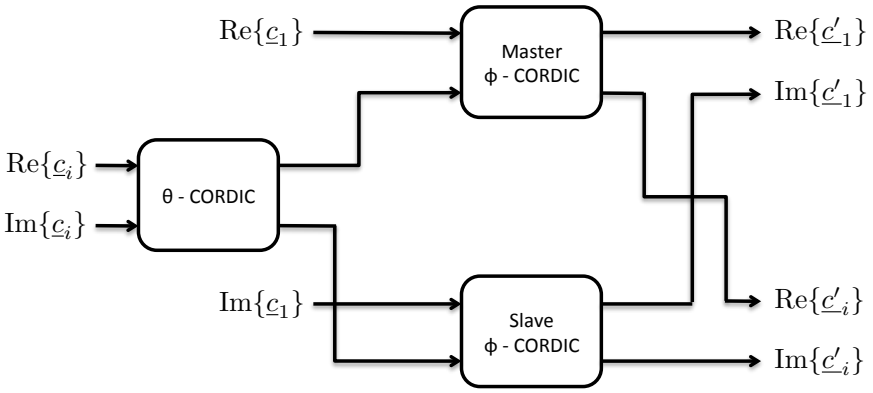

Figure 5.12: CORDIC Super Cell block.

null the real part of the targeted element, Q_{ϕ_i} transformation is performed. This transformation has to be applied to the two complex columns (the first one and the i-th one), but since the operation is real it must be applied separately to the real and imaginary parts of these two columns. This means that in order to process the two complex columns we need to assign two CORDIC circuits to perform this operation.

Data, in the form of columns of complex numbers, is presented to the circuitry sequentially, with the time period during which one complex word enters a CORDIC block called a *microcycle*. Every column that enters a circuit will have a leader, its first element, and some number of followers, which are all the other elements in the column. The CORDIC block performs vectoring on the leader (i.e. the first element of the column) and rotation by the same angle to the followers. A column composed of n_r elements will flow into a CORDIC circuit during n_r consecutive microcycles. The elements making up such a column will flow out of the CORDIC circuit at the same rate they entered, one element per microcycle. Although the elements are modified by passing through the CORDIC block they retain their identity and their order, and the leader on input remains the leader at the output.

The Q_{θ_i} transformation means that the i-th column of the complex matrix C, $\underline{c}_i$, passes through the designated CORDIC block. The action of the CORDIC block on the leader is a phase change such that the leader becomes real. The action of the CORDIC circuit on the followers is to change their phases by the same amount. A CORDIC circuit used in this manner is called a θ-CORDIC (Figure 5.12).

The Q_ϕ transformation represents rotation of two columns of complex numbers by a real angle. To accomplish Q_ϕ transformation we use two CORDIC blocks, which we call the *master* ϕ-CORDIC and the *slave* ϕ-CORDIC. The master ϕ-CORDIC deals with the real parts of columns $\underline{c}_1$ and $\underline{c}_i$, while the slave ϕ-CORDIC deals with their imaginary parts, as presented in Figure 5.12.

The master ϕ-CORDIC gets its two inputs from the real parts of the columns $\underline{c}_1$ and $\underline{c}_i$. The first pair of inputs $(\mathrm{Re}\{\underline{c}_{1,1}\}, \mathrm{Re}\{\underline{c}_{1,i}\})$ is marked as the leading one. As it passes through the master ϕ-CORDIC, it determines the angle controls ρ_υ for that CORDIC, which

are to be used to rotate all the following pairs of elements. However, the same angle should be used in the slave ϕ-CORDIC to rotate the imaginary parts of these two columns. Here we recall that the leading elements of both the pivot column ($\underline{c}_1$) and the targeted column ($\underline{c}_i$) are real. This means that the leading pair of the slave ϕ-CORDIC is going to be $(0, 0)$ and so it is going to stay $(0, 0)$ at the output as well. Therefore, instead of feeding the slave ϕ-CORDIC with $(0, 0)$, we can feed the same leading pair as to the master ϕ-CORDIC, and have it set the controls for the ϕ-CORDIC.

This combination of three CORDIC elements accomplishes the pair of transformation Q_{Θ_i} and Q_{ϕ_i} needed to zero out the first element of the targeted column, $\underline{c}_i$. We name this configuration the Super Cell block. Note that the outputs of the two ϕ-CORDICs have to be reassembled. The real parts of the ϕ-CORDICs become the real and imaginary parts respectively of the updated pivot column, $\underline{c}'_1$. The imaginary parts of the ϕ-CORDIC outputs become the real and imaginary parts respectively of the updated targeted column, $\underline{c}'_i$, as presented on Figure 5.12. For simplicity of the Super Cell block diagram, the multiplexing that happens at the input and the output of the slave ϕ-CORDIC block is not shown.

Nulling the Row of the Matrix

In Figure 5.13 we present the complete circuitry that performs nulling of the elements of the first row (except for the pivot) of the input matrix. The central part is the Super Cell block that performs nulling of a single element. The operation starts by loading the matrix elements into the read memory block in Figure 5.13. In the first iteration, the control feeds the Super Cell block with the first column ($\underline{c}_1$) as the pivot and the second column ($\underline{c}_2$) as the target. The targeted column at the output is stored in the "Write Mem" block as the second column of the resulting matrix (its first element being nulled).

The updated pivot column at the output of the Super Cell block however, has to be reused as the pivot for the second iteration when we target the third column of the matrix, $\underline{c}_3$. Thus, the next target column, $\underline{c}_3$, is fed to the Super Cell block such that it gets aligned with the updated pivot column from the first iteration. This is ensured by the "Read Ctrl" block. Note that the leading element of the updated pivot column stays real, as shown in equation (5.10). The iterations continue in the same fashion, until all the columns have been targeted and their leading elements nulled. Finally, we read the "Write Mem" block that contains the updated matrix, which has the whole first row (except for the first element) nulled.

Block diagram shown in Figure 5.13 does not present all control signals for simplicity, but the implemented block allows nulling the first row of any complex matrix ($M \times N$) up to the size of 31x16. These dimensions ensure matrix processing for arrays 8x8 and larger, depending on the exact number of transmit and receive antennas.

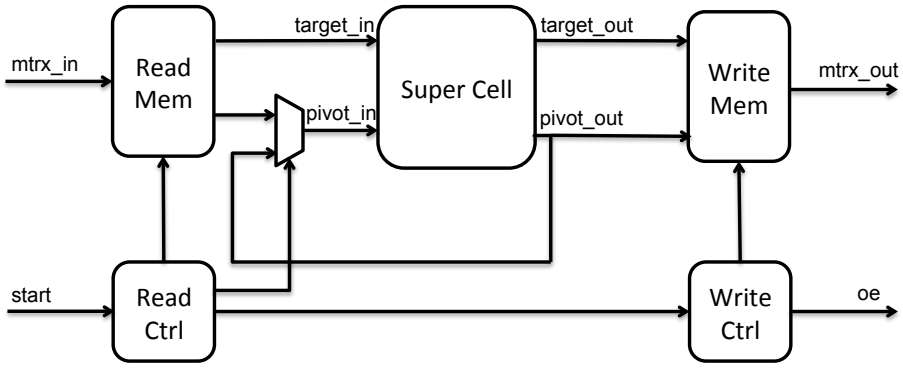

Figure 5.13: Block diagram of the Null Row block.

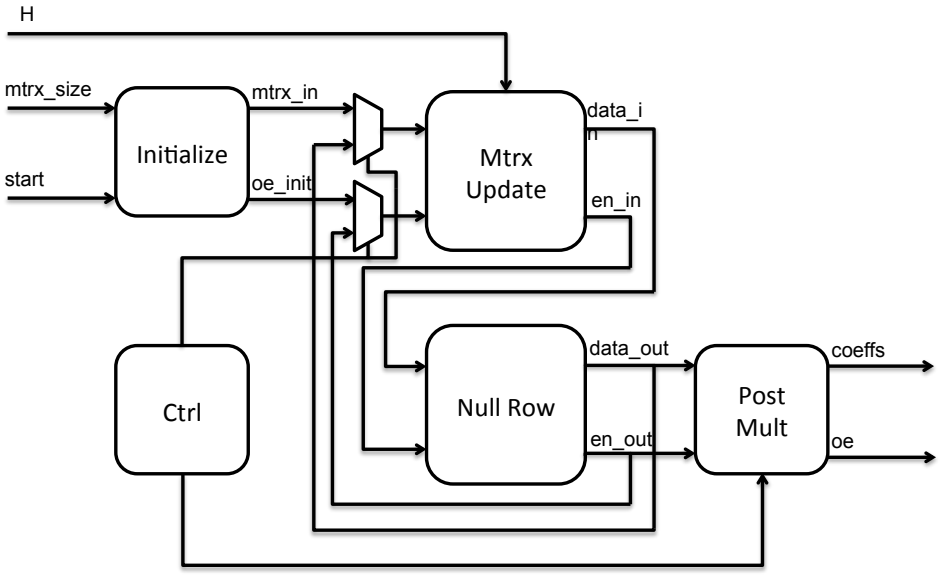

Figure 5.14: Simplified block diagram of the MMSE-SIC preprocessing unit.

MMSE-SIC Preprocessing: Coefficient Calculation

A simplified block diagram of the complete reprocessing unit is presented in Figure 5.14. The inputs are the channel matrix H and its size (n_r, n_t) and the control signal "start" that initializes the computation. The outputs are the MMSE-SIC filter vectors, which are further fed to the MMSE-SIC filter blocks.

Two central blocks in the preprocessing unit are the "Mtrx Update" block and the "Null Row" block. While the operation and the implementation of the latter has been explained in detail in the previous sections, we concentrate on the former one which updates the matrix in expression (5.3) before the row nulling operation is applied.

Matrix Update for Square-root Algorithm

Looking into the main expression of the square-root algorithm (5.3), we notice that the matrix that is being updated depends on the result of the previous iteration. Instead of zeroes in the first row, the new input matrix has the i-th row of the channel matrix multiplied with the current value of $P^{1/2}$ matrix, $P^{1/2}_{|i-1}$, and the first column has to be set according to the current iteration number. The block that takes care of this operation is denoted in Figure 5.14 as "Mtrx Update."

The block diagram of this unit is shown in Figure 5.15. The channel matrix H is stored in "Channel Mem" memory for reading its rows in each iteration. The multiplication of row H_i of the channel matrix H with the part of the input matrix denoted by $P^{1/2}_{|i-1}$ in (5.3) is done as a sequence of vector-to-vector multiplications. A vector-to-vector multiplication is implemented as a sum of products of complex elements of these vectors, by using the accumulate blocks. The elements of the vectors are entering the complex multiplier in the pipeline fashion and the output is the single complex number, stored in "Out Mem" memory block for final reading.

This buffering at the output is necessary since different parts of the updated matrix are ready at different times. The first column is initialized just by knowing the current iteration, the first row (except for the first element) has to be calculated by multiplication of the complex vector with the complex matrix as explained, and the rest of the updated matrix is just propagated from the input, according to expression (5.3).

Going back to the block diagram of the preprocessing unit from Figure 5.14 and referring to the main square-root algorithm operation given by (5.3), we conclude that one iteration of (5.3) consists of one matrix update operation and one row nulling. That means that after the initialization of the matrix, which is done in "Initialize" block in Figure 5.14, we have to iterate n_r times between "Mtrx Update" and "Null Row" blocks. After n_r iterations we get the desired matrices $P^{1/2}$ and Q. The only step left is to get the exact MMSE-SIC filter coefficients by post-multiplying columns of the Q matrix with the corresponding diagonal values of $P^{1/2}$ matrix, as suggested by (5.5). The "Post Mult" operation can be implemented with the same architecture as "Mtrx Update" unit, simplified to accommodate scalar-to-vector multiplication.

5.4.3 Critical Path Processing

Critical path processing executes every time the new signal $\underline{y}_D$ is received, which in practical communication systems usually means continuously. Luckily, blocks on the critical path are fairly simple and can be made very fast by pipelining. In the remaining of this section we briefly explain the implementation of the two implemented blocks: SIC and the MMSE-SIC filter.

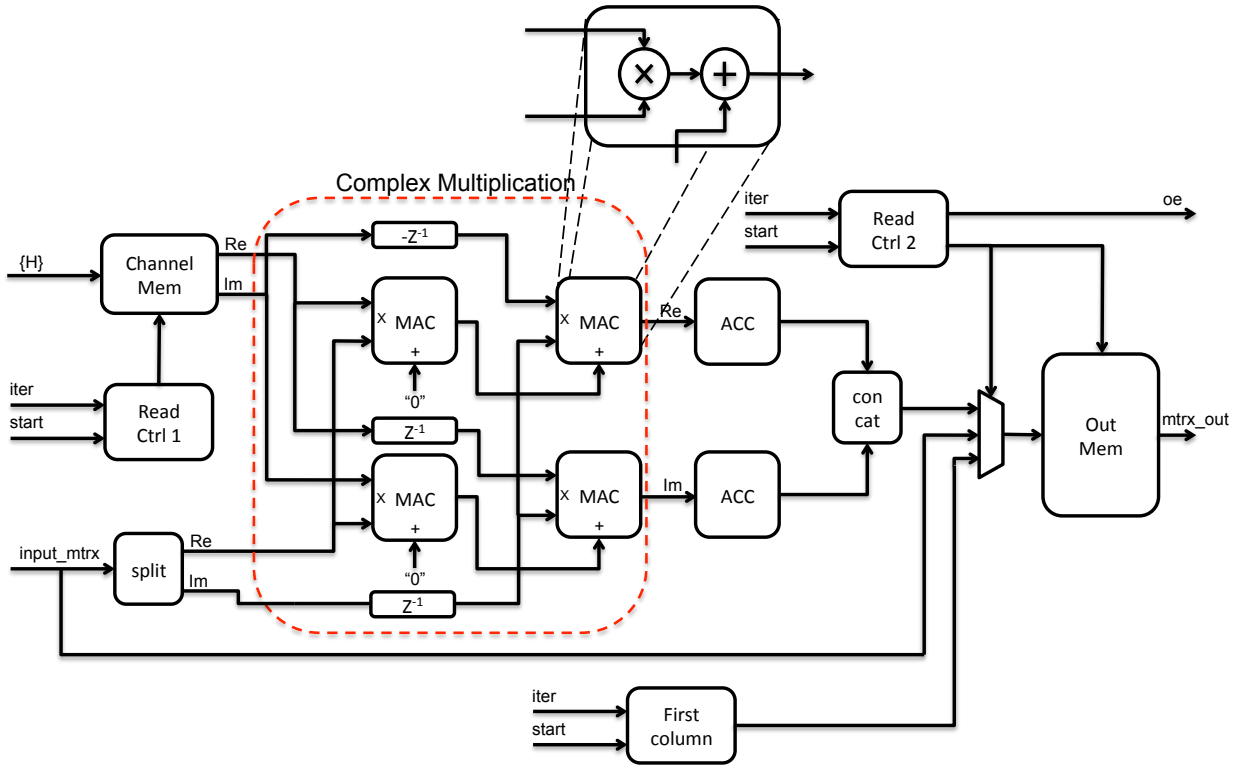

Figure 5.15: Block diagram of the "Mtrx Update" unit.

SIC

The function of the SIC block is to cancel the interference of the decoded transmitted symbol, $\hat{x}_T$. This complex symbol has to be multiplied by $k+1$-st channel column, $\underline{h}_{k+1}$, and then subtracted from the remaining signal vector, $\underline{\tilde{y}}_D$. Block diagram of the SIC unit has been implemented in Figure 5.16.

Note that the the whole computation chain is registered after each add or multiply operation. This block is fully pipelined in terms of the remaining signal vector $\underline{\tilde{y}}_D$, assuming that the decoded transmit symbol $\hat{x}_T$ arrives aligned with it.

MMSE-SIC Filter

The MMSE-SIC filter block performs vector multiplication of the remaining signal vector $\underline{\tilde{y}}_T$ with the MMSE-SIC filter vector, $\underline{g}_T$. This function is performed with the complex MAC block, as shown in Figure 5.17. In order to calculate the $SINR_{TD}$ according to expression (2.33) provided in Chapter 2, we have to filter the channel vector $\underline{h}_{TD}$ with the same MMSE-SIC filter $\underline{g}_T$ and then perform the $x/(1-x)$ operation. Because $x \in (0,1)$, this operation can be efficiently implemented using a look-up table (LUT).

Because the denominator is the same, we can reuse the same $x/(1-x)$ LUT for calculating

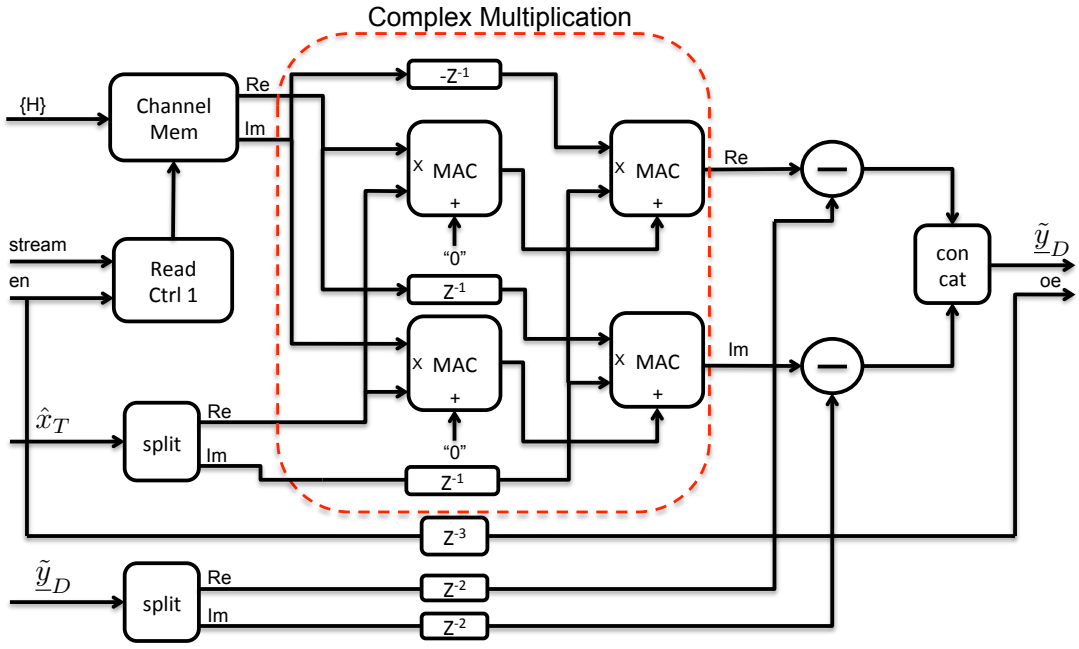

Figure 5.16: Block diagram of the interference cancellation unit.

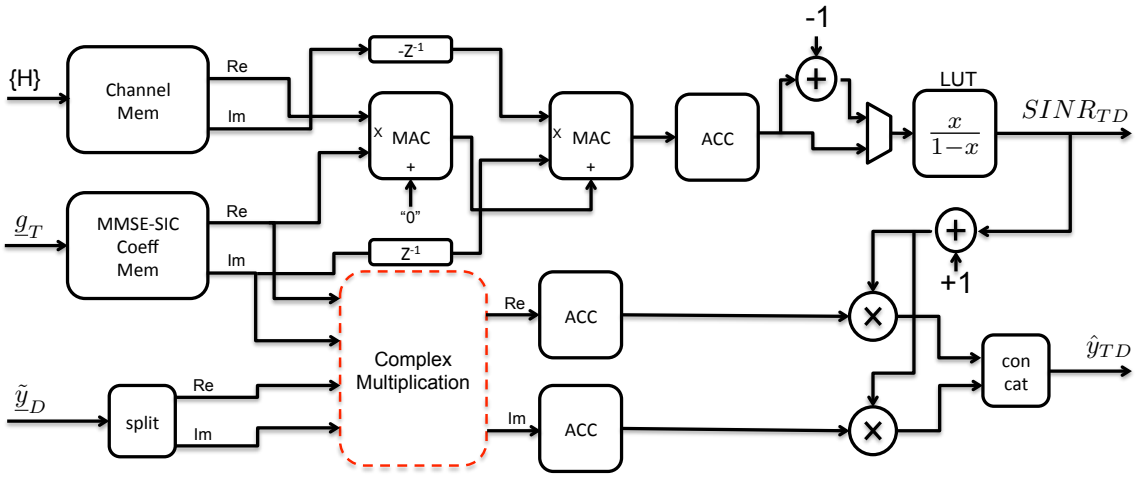

Figure 5.17: Block diagram of the MMSE-SIC filter block.

the received symbol $\hat{y}_{TD}$. The outputs of the MMSE-SIC filter block, $\hat{y}_{TD}$ and $SINR_{TD}$, are the inputs to the soft demodulator block.

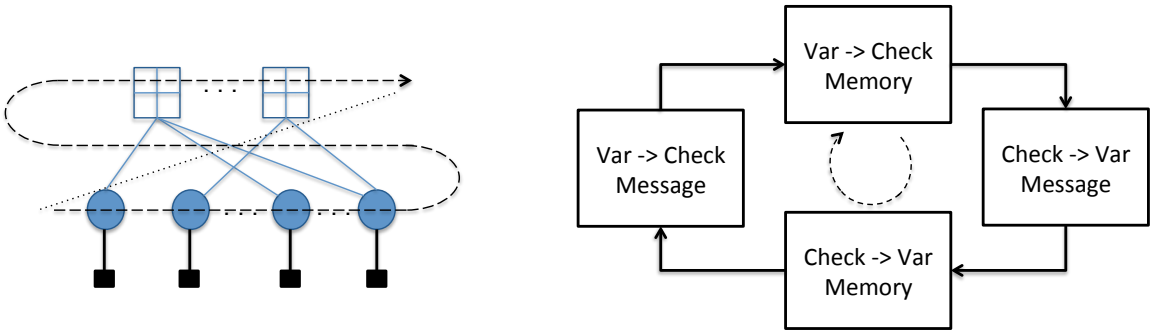

Figure 5.18: Serial implementation of a Tanner-Graph decoder. Messages from the variable bits are processed first, then the messages from the check nodes. This completes one decoding iteration.

5.5 Tanner-Graph Decoder

There are many hardware implementations of LDPC decoders. Typically the hardware is designed for a particular parity-check matrix, which has a convenient sub-matrix structure that can be easily parallelized in hardware. For example, all LDPC codes used in commercial systems such as 802.11n or 802.11ac have sub-matrix structure that can be implemented very efficiently using barrel-shifters [63, 60, 61].

In case of the LDPC-LDGM codes designed in Chapter 4, there is no particular matrix structure that was adopted. The code design suggests using codes with particular profiles, which determine if the code would perform well or not. To make these parity check matrices into an industry-standard sub-matrix structure would require a great amount of effort and code design knowledge. Instead of that, we opted for a simpler implementation structure, which relies on the serial architecture.

The serial architecture is presented in Figure 5.18. One Tanner-graph decoder iteration consists of the following two steps: 1) computing all messages from the variable nodes to the check nodes, and 2) computing all messages from the check nodes to the variable nodes, along all edges of the Tanner graph. After messages along all edges have been computed, the algorithm rewinds to the beginning and starts the computation over, with the updated message values.

Message values are stored in two memory blocks: variable-to-check memory and check-to-variable memory. Messages are read from one of these two memories serially, processed with the corresponding computation block, and stored to another memory block, as shown in Figure 5.18 on the right.

5.5.1 Data Processing

A block diagram of the data processing part of Tanner-graph decoder is presented in Figure 5.19. We distinguish four main components from Figure 5.18: the two memory blocks and the

two processing units, for storage and computation of variable-to-check and check-to-variable messages.

Each memory block consists of the two physical memories. One of them is storing the actual messages (LLRs) being transmitted along the edges of the Tanner graph, as well as one additional bit, positioned as the most-significant bit (MSB), which shows whether the message at that address has been updated in the current iteration. This memory is a dual-port RAM, to allow simultaneous read and write operation by the two processing units. The other memory stores the locations of the edges, i.e. it contains the description of the Tanner graph. It is used to provide the address for the message memory to the other memory block, where the updated message is written to.

One decoding iteration consists of the following two steps:

1. Computation of the variable-to-check messages (denoted in red in Figure 5.19). Messages from the previous iteration (or initial messages, in case of the first iteration) are read serially from the "FV MESSAGES MEMORY" and passed to the variable-to-check computation unit, along with the messages from the channel. This block computes the message to check node n_i by summing over all the messages sent along the incident edges of the variable node n_V that is being processed, except for the message sent along the edge that leads to n_i:

$$m(n_V, n_i) = \sum_{\substack{j=1 \\ j \neq i}}^{j=P} m(n_j, n_V), \quad \forall i \in \{1, 2, \ldots, P\}, \tag{5.17}$$

   where P denotes the number of incident edges to variable node n_V. After the messages to all the check nodes that have connection to n_V have been computed, they are stored into the "VF MESSAGES MEMORY", at the locations specified by entries read from the "FV EDGES MEMORY". Every time the message is written to the "VF MESSAGES MEMORY", the MSB of the corresponding memory word is set to "1".

2. Computation of the check-to-variable messages is denoted in blue in Figure 5.19. Variable messages from the previous iteration are read serially from the "VF MESSAGES MEMORY" and passed to the check-to-variable computation unit. This block performs computation of the check-to-variable messages according to expression (4.37) from Chapter 4. Original computation provided by expression (4.37) includes hyperbolic tangens and co-tangens operations. It is a standard practice to simplify these operations using the max-log approximation, which is a similar method to the one used in soft-bit demodulation presented in Chapter 4. With the max-log approximation, the message from the check node n_C to the variable node n_i becomes:

$$m(n_C, n_i) = \min_{\substack{j \in \{1,\ldots,P\} \\ j \neq i}} \{m(n_j, n_C)\}, \quad \forall i \in \{1, 2, \ldots, P\}, \tag{5.18}$$

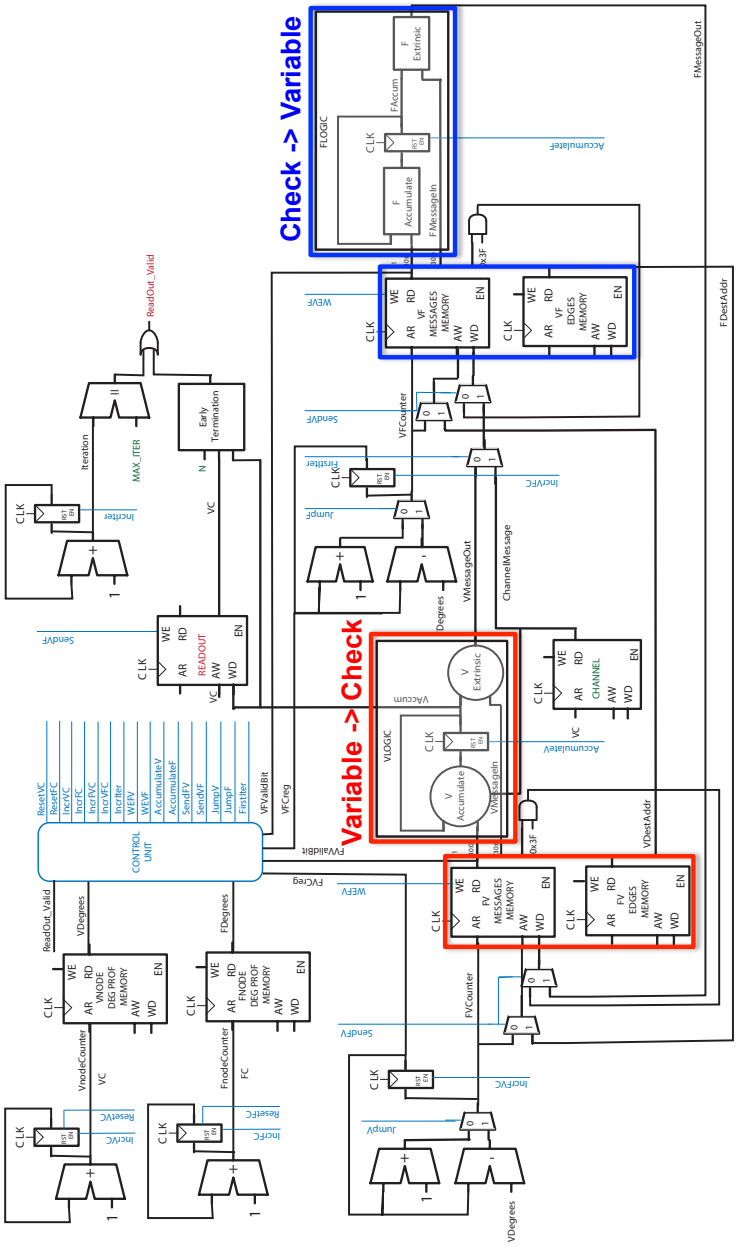

Figure 5.19: Block diagram of a Tanner-graph decoder. The main processing blocks are the computation of the messages for the check nodes (circled in red) and the computation of the messages for the variable nodes (circled in blue).

where P is the degree of the check node n_C and n_j are the variable nodes connected to it. Computation includes finding the smallest and the second smallest of all the messages sent along the incident edges of the check node n_C. The minimum message is sent to every node n_i, except for the message sent along the edge to the variable node which has sent the smallest message in the previous iteration. After the messages to all the variable nodes that have connection to n_C have been computed, they are stored into the "FV MESSAGES MEMORY", at the locations specified by entries read from the "VF EDGES MEMORY". Every time the message is written to the "FV MESSAGES MEMORY", the MSB of the corresponding memory word is set to "1".

At the very beginning, the variable-to-check messages are initialized to the messages from the LLR estimates, read from the "CHANNEL" memory block shown in Figure 5.19. At the end of each iteration, "Early Termination" block checks if the current values of the variable nodes satisfy all parity-check constraints. If they do, further decoding is stopped and the current LLR values are provided at the output.

Variable-to-check and check-to-variable computation units are using information about the variable and check nodes based on the degree profile information stored in "VNODE DEG PROF MEMORY" and "FNODE DEG PROF MEMORY", respectively.

5.5.2 Control FSMs

The controller is implemented using two finite-state machines (FSMs) working in parallel — one for controlling the variable-to-check message computation and the other for controlling the check-to-variable message computation. These FSMs follow the algorithm for Tanner-graph decoding described in Chapter 4, and are shown in Figure 5.20.

The variable-node FSM consists of the following four states:

- The INITIAL SEND state initializes all variable-to-check messages to the LLR estimates from the soft-bit demodulator block that precedes the Tanner-graph decoder. Each message from a variable node n_V to a check node n_C, $m(n_V, n_C)$, is initialized to the LLR estimate of the node n_V and the value written to the "VF MESSAGES MEMORY". The MSBs of all entries are set to "1", since all messages in the memory have been updated.

- The ACCUMULATE state reads and processes all messages $m(n_i, n_V)$ incident to the current variable node n_V. For all of these messages, the controller checks if the message $m(n_i, n_V)$ has been updated by the check-node FSM, by reading the MSB at the location of message $m(n_i, n_V)$ in "FV MESSAGES MEMORY". If the MSB has value "1", the message $m(n_i, n_V)$ is successfully read and accumulated, and the block moves to reading the next incoming message for node n_V. Before it does so, it writes "0" to the MSB at the location of message $m(n_i, n_V)$, to signal that the message has been read and needs to be updated for the next iteration.

The FSM stays in ACCUMULATE state until messages from all check nodes n_i connected to node n_V have been successfully read from the "FV MESSAGES MEMORY", and the corresponding MSBs at those locations set to "0". Once all input messages to the node n_V have been processed, the FSM checks if the node n_V is the very last one in the codeword. If it is, it checks if the early termination condition is satisfied, or if the maximum number of iterations has been reached. If yes, the FSM moves to the DONE state. In all other cases, it moves to the SEND state.

- The SEND state computes the variable-to-check messages $m(n_V, n_i)$ for all check nodes n_i connected to the node n_V, and writes them to "VF MESSAGES MEMORY". The address of each message is specified by the entry in "FV EDGES MEMORY". Along with the message $m(n_V, n_i)$, the MSB at the corresponding location in "VF MESSAGES MEMORY" is set to "1", to signal that the variable-to-check message has just been updated.

  Once all messages from the node n_V have been updated in the "VF MESSAGES MEMORY", the FSM moves to the next variable node and goes back to the ACCUMULATE state.

- The DONE state means that the messages are ready to be sent to the output pins of the Tanner-graph decoder. The controller takes accumulated values of each variable node and passes them to the output.

Because the initialization always starts with the variable node, the check-node FSM consists of the following three states:

- The ACCUMULATE state reads and processes all messages $m(n_i, n_C)$ incident to the current check node n_C. For all of these messages, the controller checks if message $m(n_i, n_C)$ has been updated by the variable-node FSM, by reading the MSB bit at the location of message $m(n_i, n_C)$ in "VF MESSAGES MEMORY". If the MSB has value "1", the message $m(n_i, n_C)$ is successfully read and accumulated, and the block moves to reading the next message for node n_C. Before it does so, it writes "0" to the MSB at the location of the message $m(n_i, n_C)$, to signal that the message has been read and needs to be updated for the next iteration.

  The FSM stays in ACCUMULATE state until messages from all variable nodes n_i connected to node n_C have been successfully read from the "VF MESSAGES MEMORY", and the corresponding MSBs at those locations set to "0". Once all input messages of the node n_C have been processed, the FSM moves to the SEND state, to compute and output the check-to-variable messages from node n_C.

- The SEND state computes the check-to-variable messages $m(n_C, n_i)$ for all variable nodes n_i connected to the node n_C, and writes them to "FV MESSAGES MEMORY". The address of each message is specified by the entry in "VF EDGES MEMORY". Along with the message $m(n_C, n_i)$, the MSB at the corresponding location in "FV

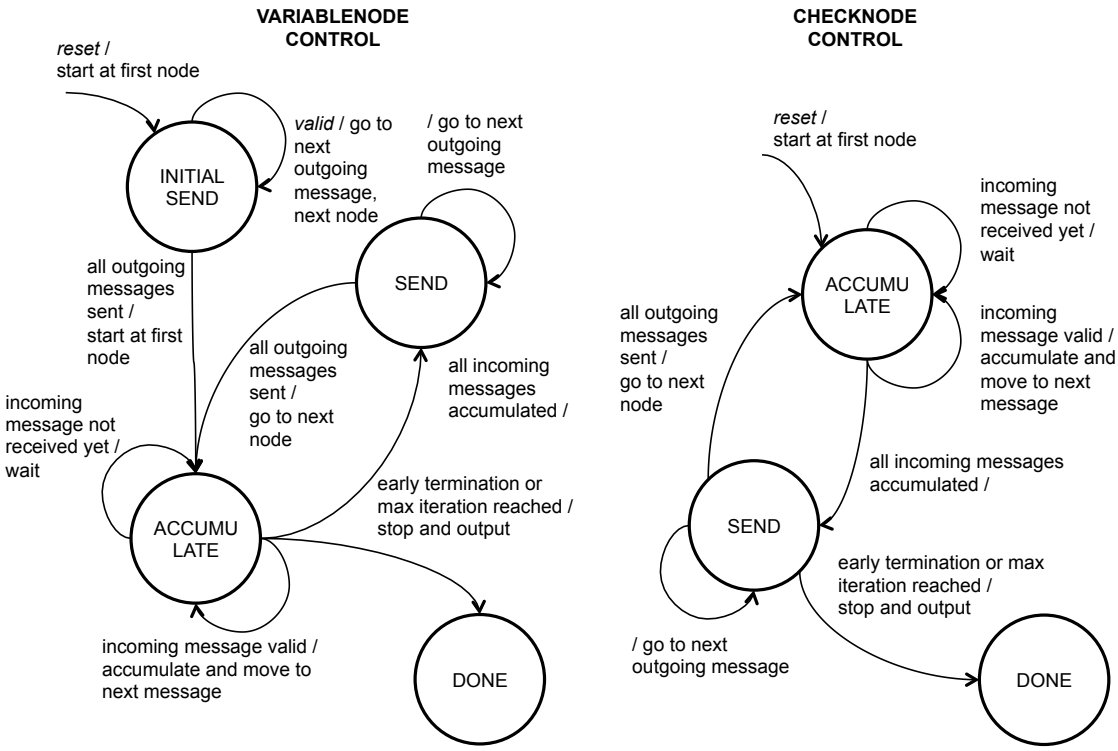

Figure 5.20: Finite-state machines for variable and check nodes of the Tanner graph. The two FSM execute simultaneously: when one is in the SEND state, the other one is in the ACCUMULATE state, and vice-versa.

MESSAGES MEMORY" is set to "1", to signal that the check-to-variable message has just been updated.

Once all messages from the node n_C have been updated in the "FV MESSAGES MEMORY", the FSM checks if the early termination condition is satisfied, or if the maximum number of iterations has been reached. If yes, the FSM moves to the DONE state. In all other cases, it goes back to the ACCUMULATE state.

- The DONE state means that the messages are ready to be sent to the output pins of the Tanner-graph decoder. The check-node FSM stops at this point, because the output LLRs are passed by the variable-node FSM.

The two FSMs execute simultaneously. While say, the variable-node FSM is computing the variable-to-check messages and writing them to "VF MESSAGES MEMORY", the check-node FSM is computing the check-to-variable messages and writing them to "FV MESSAGES MEMORY". The two message memories are being written to and read from continuously, so that both the variable-to-check and the check-to-variable processing units can be ON 100% of the time.

Most times it happens that one of the FSMs needs to wait for the other one to update the particular message that needs to be read. For a general Tanner graph this may be a performance-limiting factor. For example, let's consider the case when a Tanner-graph code with N_V variable and N_C check nodes has the very first check node, n_C^1 connected to the very last variable node, $n_V^{N_V}$. That means that the check-node FSM has to wait for the variable-node FSM to update the whole "VF MESSAGES MEMORY", before it can read message $m(n_V^{N_V}, n_C^1)$ and finish accumulating variable-to-check messages to the very first check node, n_C^1.

5.5.3 Decoding LDPC-LDGM Codes

The LDPC-LDGM codes used for cooperative-relaying have a property that the two computation units, the variable-to-check and the check-to-variable, can work in parallel and process different parts of the joint Tanner graph. The reason for this is presented in Figure 5.21 and explained below.

We can distinguish two stages of processing of the variable and check nodes in LDPC-LDGM Tanner-graph:

1. While the variable-node FSM schedules the variable-to-check computation unit to process the LDPC variable nodes (circled in blue), the check-node FSM schedules the check-to-variable unit to process the check nodes belonging to the LDGM code (circled in red), as shown on the left side of Figure 5.21. Because these two groups of nodes have no common edges, the two computation units can fully work in parallel, without waiting for each other.

2. While the check-node FSM schedules the check-to-variable computation of the LDPC check nodes and the check nodes coming from the Q-nodes (circled in red), the variable-node FSM schedules the variable-to-check computation unit to process the LDGM variable nodes (circled in blue), as shown on the right side of Figure 5.21. These two groups of nodes do have common edges between the first N_Q variable nodes and the last N_Q check nodes. However, by the time the check-node FSM reaches the last N_Q nodes, the variable-node FSM will be long done with the first N_Q variable bits, and once again the two computing units do not have to wait for each other to finish.

Of course, the two operations may not take the same amount of time, depending on the number of edges connecting the two groups of nodes. The one with more edges will set the total computation time, and the other unit will spend part of that time idling. In the single-relay LDPC-LDGM example that we present in this chapter, the number of edges in these two groups of nodes was approximately the same, which means that the two computation units can work in parallel almost all the time.

We can easily extend the single-relay LDPC-LDGM decoder architecture to support the multi-relay Tanner graph. Because there are no direct connection between the LDGM

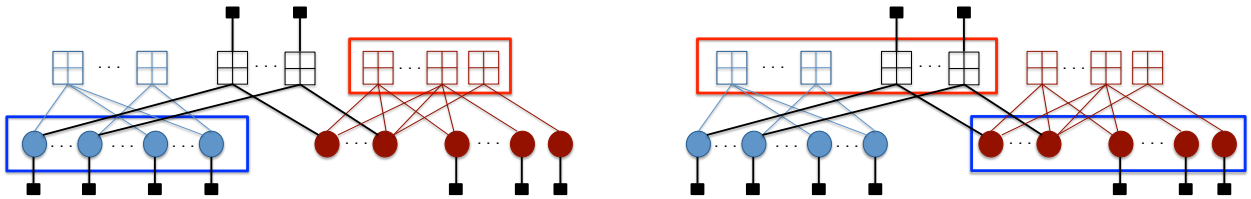

Figure 5.21: Two stages of processing of the variable and check nodes in an LDPC-LDGM Tanner-graph. Nodes circled in red and blue can be processed simultaneously by the respective computing units.

codes of multiple relays, as shown in Figure 4.4, computation of both the check-to-variable and variable-to-check messages of each LDGM code can be performed independently. This means that by parallelizing the variable-to-check and check-to-variable computation units, decoding of the whole multi-relay Tanner graph can be performed in about the same time as a single-relay LDPC-LDGM code.

5.6 Implementation Results

This section summarizes results of the hardware implementation of a single-relay cooperative link. The direct and the cooperative links are implemented on the FlexRIO FPGA boards, which are part of the National Instruments' PXIe platform [2]. The boards feature the XC5VSX95T FPGA chip, suitable for signal processing implementations [55, 56]. We demonstrate the functionality of a cooperative-relaying link when compared to the direct link, as well as the throughput and the complexity of implementation of each one of them.

5.6.1 BER Performance

To demonstrate the BER performance, we perform similar tests to those from Chapter 4. This time, however, signals are processed on the FPGAs instead of in software. Modulated symbols are transmitted over the baseband-equivalent channels, emulated on the real-time (RT) platform provided by National Instruments. In our simulations, we used quazi-static channels with Rayleigh distribution, with the new channel instance changing approximately every 20 QAM symbols. The frame structure is the same as the one from Chapter 4.

Under the assumption that the source-destination SNR has an average value of 8dB, we plot in Figure 5.22 a few BER curves showing performance of a direct-link and the single-relay cooperative link. With an average SNR of 20dB between the source and the relay, the cooperative link can achieve 2 b/s/Hz of spectral efficiency at the BER of 10^{-3}, with 7.5 dB of source-destination SNR. The listening time of the relay is $f = \frac{1}{3}$, the constellation is 16-QAM, and the same LDPC-LDGM codes from the example in Chapter 4 have been used. The direct link can achieve 1.5 b/s/Hz at the BER of 10^{-3} with 7.5 dB of SNR between the source and the destination, and 2 b/s/Hz with about 9.5 dB of SNR. The first curve

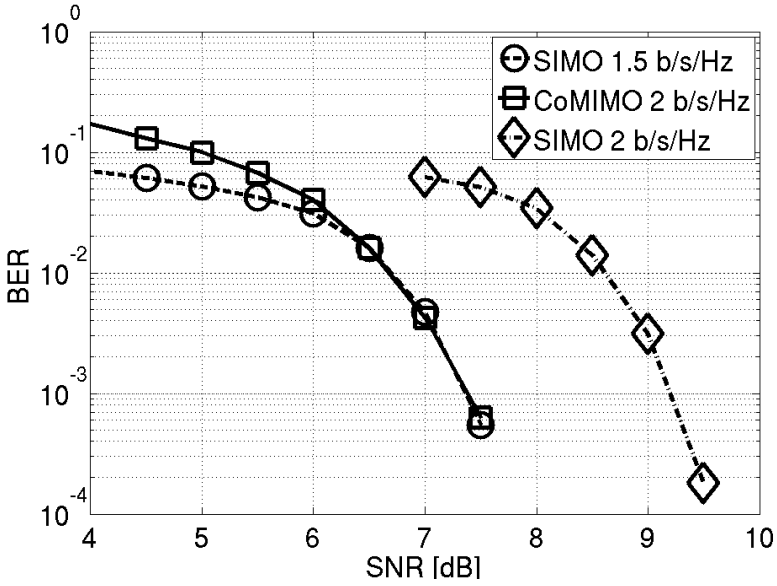

Figure 5.22: BER performance of hardware implementation of a single-relay cooperative link versus a direct link. We show that with the 7.5dB SNR between the source and the destination, cooperative relaying achieves 33% higher throughput than the direct implementation.

was obtained by using QPSK constellation and $r = \frac{3}{4}$ LDPC code from 802.11n standard with $N_S = 1944$, and the second by using 16-QAM constellation and $r = \frac{1}{2}$ LDPC code from 802.11n standard with $N_S = 1944$ [27]. Based on the BER plots from Figure 5.22, we conclude that the cooperation between the terminals improves the spectral efficiency by 33% compared to the direct-link implementation.

5.6.2 Complexity of the Back-End Baseband

In addition to the BER performance, complexity of implementation is another important metric for physical-layer design. When terminals are implemented on FPGAs, we can measure the complexity of implementation through the resource utilization of each of the FPGA chips. The resource utilization is presented as a percentage of the total amount of resources on a single chip, and is given for the three most relevant units on the FPGA chips: the logic blocks (slices), the DSP units (capable of doing an 18-bit fixed-point MAC operation) and the block-RAM (BRAM) units. In Tables 5.1 and 5.2 we present the implementation results of the direct-link and cooperative-link basebands, respectively.

We notice that the receiver at the destination uses much more resources than the other two terminals. This is mostly attributed to the MIMO detector and the channel decoder, which both account for more than 50% of resources of the destination's receiver. More important, we notice that the cooperative-relaying receiver requires around 40% more hard-

Table 5.1: FPGA resource utilization of a 1x2 SIMO direct link.

1x2 SIMO	Source [%]	Destination [%]
FPGA slices	36.5	52.9
DSP units	0	3.8
Block RAM	20.5	41.8

Table 5.2: FPGA resource utilization of a 1x1x2 cooperative-relaying link.

1x1x2 CoMIMO	Source [%]	Relay [%]	Destination [%]
FPGA slices	36.5	53.7	73.2
DSP units	0	2.3	4.7
Block RAM	20.5	32.8	62.7

ware resources compared to the direct-link receiver. This is approximately true for all three categories of resources that we observed in the last column of Tables 5.1 and 5.2.

5.6.3 Critical Path Throughput

The throughput that can be achieved over the designed direct and cooperative links is limited by the performance of the receiver at the destination, which is by far the most complex terminal. In particular, the critical path throughput at the destination is limited by the channel decoder, because of its serial implementation. In this section, we present the throughput results for the two most complex blocks: the MMSE-SIC MIMO detector and the channel decoder.

Throughput of the MMSE-SIC MIMO Detector

The maximum clock frequency of the MMSE-SIC MIMO detector is $f_{CLK} = 120$MHz, with the modest level of pipelining in the CORDIC blocks. The throughput of the critical path of the MMSE-SIC detector, which contains the MMSE-SIC filter, can be calculated as:

$$TH_{MIMO} = f_{CLK} \frac{C_S}{N_{RX}} \tag{5.19}$$

For the example design given in Section 5.6.1 with 16-QAM and two receive antennas at the destination, the measured throughput is $TH_{MIMO} = 240$Mb/s. The denominator reflects the time-sharing of the MAC units in the MMSE-SIC filter and if needed, this term can be easily eliminated by investing more MAC resources. To gain additional speed improvement, the MMSE-SIC filter and the background processing units can be separated into two different clock domains and run at different speeds.

Throughput of the LDPC-LDGM Decoder

The maximum clock frequency of the LDPC-LDGM decoder is $f_{CLK} = 70$MHz. Note that the focus of the design was to demonstrate the implementation complexity, so it was not optimized for speed. The throughput can be calculated by dividing the clock frequency and the number of cycles needed to decode a single LDPC-LDGM codeword, and then by multiplying it by the number of the decoded bits:

$$TH_{DEC} = f_{CLK} \frac{N_S}{N_{cycles}} \tag{5.20}$$

For the example design given in Section 5.6.1 with $N_S = 2040$ and $N_R = 1360$, the decoder needs approximately 150,000 cycles to finish decoding the joint LDPC-LDGM codeword, with five decoding iterations. This accounts for the throughput of $TH_{DEC} = 0.95$Mb/s. The throughput is low because of the serial implementation, which processes edges of the Tanner graph one-by-one. Ways to speed-up the decoder include designing codes with the special sub-matrix structure, which enables parallel processing, as well as having dedicated hardware resources for each decoding iteration. This has not been done in this work because of the resource limitations.

It is interesting to point out that the throughput of the LDPC decoder with $N_S = 2040$ is $TH_{DEC} = 1.13$Mb/s, not much different from the throughput of the joint LDPC-LDGM decoder. This result supports the claim made in Section 5.5.3 about efficient LDPC-LDGM decoding. The difference comes from the fact that the two computation engines can work in parallel almost 100% of time in case of the LDPC-LDGM code, whereas they need to wait for each other in case of the LDPC code.

5.7 Summary

In this chapter we have shown that in addition to the compelling performance benefits of cooperative relaying, it exhibits a moderate increase in hardware resources compared to the direct-link implementation. We presented the hardware architecture of all signal-processing blocks at the source, the relay and the destination. The two block requiring special attention are the MMSE-SIC MIMO detector that supports the DBLAST space-time scheme and the joint LDPC-LDGM decoder. We have shown that the MIMO detector can be efficiently implemented using the square-root algorithm that implements systolic arrays using CORDIC. The joint decoder uses serial architecture, which fits well into the LDPC-LDGM framework because two main computing units can work in parallel and thus improve the decoding efficiency. Lastly, we have demonstrated on an example that the cooperative-relaying link improves the spectral efficiency by 33%, with approximately 40% increase in destination receiver's complexity.

Chapter 6

Conclusion

6.1 Summary of the Results

In this thesis we present the theoretical analysis and the physical-layer design and implementation of a multi-relay QMF cooperative link. This research is motivated by the seminal work of Ozgur, Leveque and Tse on cooperation in wireless networks, which showed that the capacity of ad-hoc wireless networks can grow linearly with the node density [44]. In this work, we show that the spectral efficiency of a single cooperative MIMO link also grows approximately linearly, by using dedicated wireless relays located close to the information source. The benefit of implementing cooperation on a single link instead of the whole network is to simplify design and enable deployment of cooperative relaying in already existing wireless networks, such as cellular and WiFi.

This dissertation relies on the previous work by Avestimehr and Tse on the QMF relaying scheme [4] and our work on the channel coding for QMF relaying [38]. While these papers focus more on the theoretical performance and analysis of cooperating relaying, this work aims at bringing those concepts one step closer to implementation in practical communication systems. The three key results of this dissertation are:

1. Spectral efficiency scaling with the number of relays for the multi-relay cooperative link, including low-complexity relay scheduling scheme. We show that the achievable QMF rate scales approximately linearly with the number of relays, until the number of relays becomes larger than the proximity gain between the source and the relays. The local scheduling scheme presented in Chapter 3 is an analytical scheme which does not require global scheduler, and is therefore much more implementation-friendly than the optimal scheduling scheme. It has been shown to be equivalent to the optimal scheme for certain channel conditions, and performing close to it in other scenarios.

2. System design procedure for a multi-relay cooperative link, with the demonstration that the simulated spectral efficiencies follow theoretical expectations. We show design procedure for major baseband signal-processing blocks of a multi-relay cooperative

link. In particular, we provide ready-to-implement algorithms for the MIMO detection, the soft-bit demodulation, the quantization at the relays and the joint LDPC-LDGM decoding. The performance of suggested design procedure is showcased on an example with the three-relay link doubling the spectral efficiency of the direct link, which has been predicted by the theoretical analysis. Equivalently, the spectral efficiency gain can be seen as the power gain, with the five-relay system operating at 9 dB lower SNR than the direct link, with the same spectral efficiency.

3. Hardware implementation of the cooperative-relaying baseband, showcasing the trade-off between performance and complexity on a single-relay example. We extend the work on a single-relay cooperative link from [38], by implementing the physical-layer signal processing blocks in hardware and showing that the spectral efficiency improvement is approximately equivalent to the increase in hardware complexity of the destination's receiver. In particular, in the implementation example provided in Chapter 5, we demonstrate that the spectral efficiency of a direct link can be increased by 33% by using cooperative-relaying, with the complexity increase of the destination's receiver of approximately 40%.

6.2 Directions for Future Work

6.2.1 Time and Frequency Synchronization of the Terminals

An important aspect of the physical-layer design that has not been studied in this dissertation is how to synchronize the terminals in a cooperative-relaying link. A good study on the time and frequency synchronization of a single-relay link has been done in [53], where the authors argued that if the source-to-relay SNR is high enough, the relays can simply synchronize to the source using the same synchronization sequence as the destination.

Assuming that the relays can synchronize perfectly to the source, they could also help improve the synchronization at the destination, by transmitting the same synchronization sequence as the source, at exactly the same time. The destination would then see multiple copies of the same signal, and be able to synchronize to the source and the relays more reliably. This is possible as long as the relays are close to the source, so the propagation time to the destination is approximately the same.

6.2.2 Channel Estimation for Cooperative-Relaying Link

Like the time and frequency synchronization, the channel estimation is also considered overhead in a communication protocol. Throughout this dissertation, we assumed that the channel state information (CSI) is always available at the receive side. While this is straightforward to implement in the direct-link scenario, it is not trivial in the cooperative-relaying case when there are multiple transmitters.

Intuitively, channel estimation in the cooperative-relaying link can be compared to the multi-user MIMO (MU-MIMO) channel estimation in the modern cellular and WiFi standards. Well-developed techniques, such as transmitting orthogonal sequences over the same resource, can be used to perform estimation of both the source-destination and relay-destination channels. While this technique can be easily applied in a single-relay scenario, it is not clear if the same method for channel estimation can be used in a multi-relay system.

6.2.3 Code Design for Cooperative Link with Multiple Relays

Channel codes used in Chapters 4 and 5 are originally developed for a single-relay link, using techniques from [38]. While this philosophy of designing relays independently of other relays is shown to give good spectral efficiency performance, it is rather a heuristic that simplified the channel code design for a multi-relay cooperative link.

Relays' codes that will be used in a multi-relay cooperative link should, in general, be designed jointly with the source code. That means that in a three-relay link, four channel codes should be designed jointly. This is a very complex problem, and likely heuristics similar to those from [41] would need to be devised to simplify the code profile search. Developing new techniques for multi-relay joint code design would likely improve the spectral efficiency gain measured in Chapter 4.

6.2.4 Relay Selection

In Chapter 3 we studied the relay scheduling problem as if there was an exact number of relays available for cooperation. In a communication network with many source terminals, there may be a large number of relays available for cooperation, and each source would need to select a specific subset of relays to cooperate with. In that case, algorithms for relay selection and allocation need to be developed. Selecting a subset of relays out of the "pool" of available relays could significantly improve the performance of the cooperative MIMO links, because only the relays with the best channels would be selected.

6.3 The Future of Cooperative Relaying

Cooperation in wireless networks is already commercial. Relaying between a base-station and a mobile device, as well as the cooperation among base-stations (CoMP) have already been included in the few latest releases of the LTE standard [17, 11]. However, these efforts represent just the beginning of building practical communication systems with wireless cooperation. These two forms of cooperation improve only the diversity gain of wireless transmission by cooperating with dedicated base- or relay-towers, which are difficult and costly to deploy. Improving diversity and multiplexing gain through cooperation with phone-like devices, which could be deployed everywhere and available in abundance all around us is still an active area of research.

In the world where everyone and everything will have a radio, cooperation among wireless terminals will exert huge potential for harvesting additional spectral efficiency, much needed to satisfy the predicted increase in data traffic. We envision that the cooperation among wireless terminals will first be introduced in the existing wireless communication networks that need higher spectral and energy efficiencies, such as cellular and WiFi. In this dissertation we have shown one way to introduce cooperation among wireless terminals in a centralized communication network, by focusing on a single link. Cooperating over a single link instead of the whole network means that most of the network infrastructure can remain the same, because the main changes are at the physical and the MAC layer of the wireless link between a user and a base-station. Besides, cooperating over a single link can eliminate a potential privacy concern, by letting users own the relays that they cooperate with.

According to [44], the true power of cooperation among wireless terminals comes at the network level, when multiple users cooperate among themselves. Potential application scenario for such a scheme in the near future is in sensor networks, which deploy a large number of users over a given area. Using the proximity of the wireless terminals and enabling cooperation could significantly improve the energy efficiency of each user, which is one of the main design tasks in the sensor networks.

Similar design concepts can be applied to satisfy other system specifications, besides spectral and energy efficiency. For example, there are many wireless systems that would benefit from the reduced latency of communications, such as industry control systems, automotive, air-space industry, gaming platforms, etc. Relaying can enable wireless terminals to communicate not only more reliably or at higher data rates, but also with lower latency. Benefits of cooperative relaying in such communication systems are yet to be explored.

Bibliography

[1] E. Agrell et al. "Closest point search in lattices". In: *Information Theory, IEEE Transactions on* 48.8 (2002), pp. 2201–2214. ISSN: 0018-9448. DOI: 10.1109/TIT.2002.800499.

[2] H.A Andrade et al. "A methodology for the design and deployment of reliable systems on heterogeneous platforms". In: *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*. 2012, pp. 1–7. DOI: 10.1109/ReConFig.2012.6416722.

[3] A.S. Avestimehr, S.N. Diggavi, and D.N.C. Tse. "Approximate capacity of Gaussian relay networks". In: *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*. 2008, pp. 474–478. DOI: 10.1109/ISIT.2008.4595031.

[4] A.S. Avestimehr, S.N. Diggavi, and D.N.C. Tse. "Wireless Network Information Flow: A Deterministic Approach". In: *Information Theory, IEEE Transactions on* 57.4 (2011), pp. 1872–1905. ISSN: 0018-9448. DOI: 10.1109/TIT.2011.2110110.

[5] S. Brahma, A. Ozgur, and C. Fragouli. "Simple schedules for half-duplex networks". In: *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*. 2012. DOI: 10.1109/ISIT.2012.6283025.

[6] S. ten Brink, G. Kramer, and A. Ashikhmin. "Design of low-density parity-check codes for modulation and detection". In: *Communications, IEEE Transactions on* 52.4 (2004), pp. 670–678. ISSN: 0090-6778. DOI: 10.1109/TCOMM.2004.826370.

[7] G. Caire, Giorgio Taricco, and Ezio Biglieri. "Bit-interleaved coded modulation". In: *Information Theory, IEEE Transactions on* 44.3 (1998), pp. 927–946. ISSN: 0018-9448. DOI: 10.1109/18.669123.

[8] Martina Cardone et al. "Gaussian half-duplex relay networks: Improved gap and a connection with the assignment problem". In: *Information Theory Workshop (ITW), 2013 IEEE*. 2013, pp. 1–5. DOI: 10.1109/ITW.2013.6691349.

[9] Sae-Young Chung, T.J. Richardson, and R.L. Urbanke. "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation". In: *Information Theory, IEEE Transactions on* 47.2 (2001), pp. 657–670. ISSN: 0018-9448. DOI: 10.1109/18.910580.

[10] Cisco. "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20132018". In: (2014).

[11] "Coordinated multi-point operation for LTE physical layer aspects". In: *3GPP Technical Report 36.819* (2013).

[12] T. Cover and A.E. Gamal. "Capacity theorems for the relay channel". In: *Information Theory, IEEE Transactions on* 25.5 (1979), pp. 572–584. ISSN: 0018-9448. DOI: 10.1109/TIT.1979.1056084.

[13] M.-O. Damen, H. El-Gamal, and G. Caire. "On maximum-likelihood detection and the search for the closest lattice point". In: *Information Theory, IEEE Transactions on* 49.10 (2003), pp. 2389–2402. ISSN: 0018-9448. DOI: 10.1109/TIT.2003.817444.

[14] Pramod Viswanath David Tse. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[15] Melissa Duarte et al. "Quantize-map-forward (QMF) Relaying: An Experimental Study". In: *Proceedings of the Fourteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*. MobiHoc '13. Bangalore, India: ACM, 2013, pp. 227–236. ISBN: 978-1-4503-2193-8. DOI: 10.1145/2491288.2491307. URL: http://doi.acm.org/10.1145/2491288.2491307.

[16] R. Etkin et al. "On efficient min-cut approximations in half-duplex relay networks". In: *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*. 2013, pp. 1949–1953. DOI: 10.1109/ISIT.2013.6620566.

[17] "Evolved Universal Terrestrial Radio Access (E-UTRA); Relay architectures for E-UTRA (LTE-Advanced)". In: *3GPP Technical Report 36.806* (2010).

[18] U. Fincke and M. Pohst. "Improved Methods for Calculating Vectors of Short Length in a Lattice, Including a Complexity Analysis". English. In: *Mathematics of Computation* 44.170 (1985), pp. 463–471. ISSN: 00255718. URL: http://www.jstor.org/stable/2007966.

[19] L. R. Ford and D. R. Fulkerson. "Maximal Flow through a Network." In: *Canadian Journal of Mathematics* 8 (), pp. 399–404. URL: http://www.rand.org/pubs/papers/P605/.

[20] Gerard J. Foschini. "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas". In: *Bell Labs Technical Journal* 1.2 (1996), pp. 41–59. ISSN: 1538-7305. DOI: 10.1002/bltj.2015. URL: http://dx.doi.org/10.1002/bltj.2015.

[21] M. Grieger et al. "Field trial results for a coordinated multi-point (CoMP) uplink in cellular systems". In: *Smart Antennas (WSA), 2010 International ITG Workshop on*. 2010, pp. 46–51. DOI: 10.1109/WSA.2010.5456387.

[22] Zhan Guo and P. Nilsson. "Reduced complexity Schnorr-Euchner decoding algorithms for MIMO systems". In: *Communications Letters, IEEE* 8.5 (2004), pp. 286–288. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2004.827376.

[23] P. Gupta and P.R. Kumar. "The capacity of wireless networks". In: *Information Theory, IEEE Transactions on* 46.2 (2000), pp. 388–404. ISSN: 0018-9448. DOI: 10.1109/18.825799.

[24] H. Harashima and H. Miyakawa. "Matched-Transmission Technique for Channels With Intersymbol Interference". In: *Communications, IEEE Transactions on* 20.4 (1972), pp. 774–780. ISSN: 0090-6778. DOI: 10.1109/TCOM.1972.1091221.

[25] B. Hassibi. "An efficient square-root algorithm for BLAST". In: *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*. Vol. 2. 2000, II737 –II740 vol.2. DOI: 10.1109/ICASSP.2000.859065.

[26] B.M. Hochwald and S. Ten Brink. "Achieving near-capacity on a multiple-antenna channel". In: *Communications, IEEE Transactions on* 51.3 (2003), pp. 389–399. ISSN: 0090-6778. DOI: 10.1109/TCOMM.2003.809789.

[27] "IEEE 802.11n Wireless Standard". In: (2009).

[28] M. Jorgovanovic et al. "Relay scheduling and interference cancellation for quantize-map-and-forward cooperative relaying". In: *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*. 2013, pp. 1959–1963. DOI: 10.1109/ISIT.2013.6620568.

[29] Milos Jorgovanovic. "Design of Flexible Soft Output MIMO Detector for Cooperative MIMO". MA thesis. EECS Department, University of California, Berkeley, 2013. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-170.html.

[30] Hyungjin Kim, D.-U. Lee, and J.D. Villasenor. "Design Tradeoffs and Hardware Architecture for Real-Time Iterative MIMO Detection using Sphere Decoding and LDPC Coding". In: *Selected Areas in Communications, IEEE Journal on* 26.6 (2008), pp. 1003–1014. ISSN: 0733-8716. DOI: 10.1109/JSAC.2008.080816.

[31] F.R. Kschischang, B.J. Frey, and H.-A Loeliger. "Factor graphs and the sum-product algorithm". In: *Information Theory, IEEE Transactions on* 47.2 (2001), pp. 498–519. ISSN: 0018-9448. DOI: 10.1109/18.910572.

[32] J.N. Laneman, D.N.C. Tse, and Gregory W. Wornell. "Cooperative diversity in wireless networks: Efficient protocols and outage behavior". In: *Information Theory, IEEE Transactions on* 50.12 (2004), pp. 3062–3080. ISSN: 0018-9448. DOI: 10.1109/TIT.2004.838089.

[33] "LTE physical layer; General description (Release 11)". In: *3GPP Technical Report 36.201* (2012).

[34] Upamanyu Madhow. *Fundamentals of Digital Communication*. Cambridge University Press, 2008.

[35] P. Marsch, M. Grieger, and G. Fettweis. "Large scale field trial results on different uplink coordinated Multi-Point (CoMP) concepts in an urban environment". In: *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*. 2011, pp. 1858–1863. DOI: 10.1109/WCNC.2011.5779416.

[36] Edward C. Van Der Meulen. "Three-Terminal Communication Channels". English. In: *Advances in Applied Probability* 3.1 (1971), pp. 120–154. ISSN: 00018678. URL: http://www.jstor.org/stable/1426331.

[37] C. Michalke, E. Zimmermann, and G. Fettweis. "Linear Mimo Receivers vs. Tree Search Detection: A Performance Comparison Overview". In: *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on*. 2006, pp. 1–7. DOI: 10.1109/PIMRC.2006.254377.

[38] V. Nagpal et al. "Coding and system design for quantize-map-and-forward relaying". In: *Selected Areas in Communications, IEEE Journal on* 31.8 (2013), pp. 1423–1435. ISSN: 0733-8716. DOI: 10.1109/JSAC.2013.130807.

[39] V. Nagpal et al. "Cooperative multiplexing in the multiple antenna half duplex relay channel". In: *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*. 2009, pp. 1438–1442. DOI: 10.1109/ISIT.2009.5205885.

[40] V. Nagpal et al. "Quantize-map-and-forward relaying: Coding and system design". In: *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*. 2010, pp. 443–450. DOI: 10.1109/ALLERTON.2010.5706940.

[41] Vinayak Nagpal. "Cooperative Multiplexing in Wireless Relay Networks". PhD thesis. EECS Department, University of California, Berkeley, 2012.

[42] H. Nishiyama, M. Ito, and N. Kato. "Relay-by-smartphone: realizing multihop device-to-device communications". In: *Communications Magazine, IEEE* 52.4 (2014), pp. 56–65. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6807947.

[43] A Osseiran et al. "Scenarios for 5G mobile and wireless communications: the vision of the METIS project". In: *Communications Magazine, IEEE* 52.5 (2014), pp. 26–35. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6815890.

[44] A. Ozgur, O. Leveque, and D.N.C. Tse. "Hierarchical Cooperation Achieves Optimal Capacity Scaling in Ad Hoc Networks". In: *Information Theory, IEEE Transactions on* 53.10 (2007), pp. 3549–3572. ISSN: 0018-9448. DOI: 10.1109/TIT.2007.905002.

[45] F. Parvaresh and R. Etkin. "On computing the capacity of relay networks in polynomial time". In: *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*. 2011, pp. 1342–1346. DOI: 10.1109/ISIT.2011.6033756.

[46] S. Pawar, AS. Avestimehr, and D.N.C. Tse. "Diversity-multiplexing tradeoff of the half-duplex relay channel". In: *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*. 2008, pp. 27–33. DOI: 10.1109/ALLERTON.2008.4797531.

[47] Michael Pohst. "On the Computation of Lattice Vectors of Minimal Length, Successive Minima and Reduced Bases with Applications". In: *SIGSAM Bull.* 15.1 (Feb. 1981), pp. 37–44. ISSN: 0163-5824. DOI: 10.1145/1089242.1089247. URL: http://doi.acm.org/10.1145/1089242.1089247.

[48] C.M. Rader. "VLSI systolic arrays for adaptive nulling [radar]". In: *Signal Processing Magazine, IEEE* 13.4 (July 1996), pp. 29 –49. ISSN: 1053-5888. DOI: 10.1109/79.526897.

[49] T.J. Richardson and R.L. Urbanke. "The capacity of low-density parity-check codes under message-passing decoding". In: *Information Theory, IEEE Transactions on* 47.2 (2001), pp. 599–618. ISSN: 0018-9448. DOI: 10.1109/18.910577.

[50] C.P. Schnorr and M. Euchner. "Lattice basis reduction: Improved practical algorithms and solving subset sum problems". English. In: *Mathematical Programming* 66.1-3 (1994), pp. 181–199. ISSN: 0025-5610. DOI: 10.1007/BF01581144. URL: http://dx.doi.org/10.1007/BF01581144.

[51] A. Sengupta, I-Hsiang Wang, and C. Fragouli. "Optimizing Quantize-Map-and-Forward relaying for Gaussian diamond networks". In: *Information Theory Workshop (ITW), 2012 IEEE*. 2012, pp. 381–385. DOI: 10.1109/ITW.2012.6404698.

[52] A. Sengupta et al. "Graph-based codes for Quantize-Map-and-Forward relaying". In: *Information Theory Workshop (ITW), 2011 IEEE*. 2011, pp. 140–144. DOI: 10.1109/ITW.2011.6089363.

[53] Oh-Soon Shin et al. "Design of an OFDM Cooperative Space-Time Diversity System". In: *Vehicular Technology, IEEE Transactions on* 56.4 (2007), pp. 2203–2215. ISSN: 0018-9545. DOI: 10.1109/TVT.2007.897642.

[54] B. Hassibi T. Kailath A.H. Sayed. *Linear Estimation*. Prentice Hall, 1999.

[55] *Virtex-5 FPGA User Guide*. www.xilinx.com. 2010.

[56] *Virtex-5 FPGA XtremeDSP Design Considerations*. www.xilinx.com. 2010.

[57] M. Tomlinson. "New automatic equaliser employing modulo arithmetic". In: *Electronics Letters* 7.5 (1971), pp. 138–139. ISSN: 0013-5194. DOI: 10.1049/el:19710089.

[58] Jack E. Volder. "The CORDIC Trigonometric Computing Technique". In: *Electronic Computers, IRE Transactions on* EC-8.3 (1959), pp. 330 –334. ISSN: 0367-9950. DOI: 10.1109/TEC.1959.5222693.

[59] Wei Wang, L. Ong, and M. Motani. "Transmission schedule optimization for half-duplex multiple-relay networks". In: *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, 2009. WiOPT 2009. 7th International Symposium on*. 2009, pp. 1–9. DOI: 10.1109/WIOPT.2009.5291631.

[60] M. Weiner, B. Nikolic, and Zhengya Zhang. "LDPC decoder architecture for high-data rate personal-area networks". In: *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*. 2011, pp. 1784–1787. DOI: 10.1109/ISCAS.2011.5937930.

[61] Matthew Weiner et al. "A Scalable 1.5-to-6Gb/s 6.2-to-38.1mW LDPC Decoder for 60GHz Wireless Networks in 28nm UTBB FDSOI". In: *International Solid State Circuits Conference*. 2014.

[62] AD. Wyner and J. Ziv. "The rate-distortion function for source coding with side information at the decoder". In: *Information Theory, IEEE Transactions on* 22.1 (1976), pp. 1–10. ISSN: 0018-9448. DOI: 10.1109/TIT.1976.1055508.

[63] Zhengya Zhang et al. "An Efficient 10GBASE-T Ethernet LDPC Decoder Design With Low Error Floors". In: *Solid-State Circuits, IEEE Journal of* 45.4 (2010), pp. 843–855. ISSN: 0018-9200. DOI: 10.1109/JSSC.2010.2042255.

[64] Lizhong Zheng and D.N.C. Tse. "Diversity and multiplexing: a fundamental tradeoff in multiple-antenna channels". In: *Information Theory, IEEE Transactions on* 49.5 (2003), pp. 1073–1096. ISSN: 0018-9448. DOI: 10.1109/TIT.2003.810646.