Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia Elétrica e da Computação

Algoritmos e Arquiteturas VLSI para Detectores MIMO com Decisão Suave

José Marcelo Lima Duarte

Orientador: Prof. Dr. José Alberto Nicolau de Oliveira

Co-orientador: Prof. Dr. Jorge Dantas de Melo

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e da Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Doutor em Ciências.

Natal, RN, outubro de 2012

UFRN / Biblioteca Central Zila Mamede.

Catalogação da Publicação na Fonte.

Duarte, José Marcelo Lima.

Algorítmo e arquiteturas VLSI para detectores MIMO com decisão suave / José Marcelo Lima Duarte. – Natal, RN, 2012.
94 f. : il.

Orientador: Prof. Dr. José Alberto Nicolau.
Co-Orientador: Prof. Dr. Jorge Dantas de Melo.

Tese (Doutorado) – Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica e de Computação.

1. Antenas - Tese. 2. MIMO - Tese. 3. Algorítmo de detecção - Tese. 4. Processamento digital de sinais - Tese. 5. Engenharia Elétrica e de Computação - Tese. I. Nicolau, José Alberto. II. Melo, Jorge Dantas de. III. Universidade Federal do Rio Grande do Norte. III. Título.

RN/UF/BCZM CDU 621.396.67

Algoritmos e Arquiteturas VLSI para Detectores MIMO com Decisão Suave

José Marcelo Lima Duarte

Tese de Doutorado aprovada em 17 de agosto de 2012 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. José Alberto Nicolau de Oliveira (orientador) ..... UFRN

Prof. Dr. Jorge Dantas de Melo (co-orientador) ..... UFRN

Prof. Dr. Dalton Soares Arantes ..... UNICAMP

Prof. Dr. David Simonetti Barbalho ..... UFRN

Prof. Dr. Luiz Gonzaga De Queiroz Silveira Junior ..... UFRN

Prof. Dr. Manoel Eusébio de Lima ..... UFPE

Prof. Dr. Marcelo Augusto Costa Fernandes ..... UFRN

Tati, conseguimos antes de Lucas chegar!

Agradecimentos

A Tati, minha esposa, pela paciência durante a escrita dessa tese.

Aos meus pais, pelo apoio durante esta jornada.

Aos meus orientador e co-orientador, professor Alberto e professor Jorge, sou grato pela orientação e incentivo.

Ao professor Davi Simonetti Barbalho, pelas preciosas críticas e sugestões.

Aos meus colegas de trabalho do LSITEC, pelo incentivo.

Ao LSITEC, por permitir a utilização das ferramentas da empresa no desenvolvimento dessa tese.

Resumo

O uso de sistemas de Múltiplas Entradas e Múltiplas Saídas (*Multiple Input Multiple Output* - MIMO) tem permitido a recente evolução dos novos padrões de comunicação móvel. A técnica MIMO da Multiplexação Espacial, em particular, provê um aumento linear na capacidade de transmissão com o mínimo entre número de antenas transmissoras e antenas receptoras. Para se obter um desempenho próximo a capacidade em sistemas com Multiplexação Espacial faz-se necessário o uso de um detector MIMO com decisão suave do tipo *Maximum A Posteriori Probability*. Entretanto, tal detector é muito complexo para soluções práticas. Assim, o objetivo dos algoritmos de detecção MIMO voltados para implementação é obter uma boa aproximação do detector ideal mantendo um nível de complexidade aceitável. Além disso, o algoritmo precisa ser mapeado para uma arquitetura VLSI de área pequena e que atenda a taxa de transmissão exigida pelos padrões de comunicações móveis. Sendo a multiplexação espacial uma técnica recente, defende-se que ainda há muito espaço para evolução dos algoritmos e arquiteturas relacionadas. Por isso, esta tese se focou no estudo de algoritmos sub-ótimos e arquiteturas VLSI para detectores MIMO de banda larga com decisão suave. Como resultado, algoritmos inéditos foram desenvolvidos partindo de propostas de otimizações para algoritmos já estabelecidos. Baseado nesses resultados, novas arquiteturas de detectores MIMO com modulação configurável e competitivos parâmetros de área, desempenho e taxa de processamento são aqui propostas. Os algoritmos desenvolvidos foram extensivamente simulados e as arquiteturas sintetizadas para que os resultados pudessem servir como referência para outros trabalhos na área.

Palavras-chave: MIMO, detecção, demodulação, processamento digital de sinais, arquitetura, VLSI, ASIC, FPGA.

Abstract

The use of Multiple Input Multiple Output (MIMO) systems has permitted the recent evolution of wireless communication standards. The Spatial Multiplexing MIMO technique, in particular, provides a linear gain at the transmission capacity with the minimum between the numbers of transmit and receive antennas. To obtain a near capacity performance in SM-MIMO systems a soft decision Maximum A Posteriori Probability MIMO detector is necessary. However, such detector is too complex for practical solutions. Hence, the goal of a MIMO detector algorithm aimed for implementation is to get a good approximation of the ideal detector while keeping an acceptable complexity. Moreover, the algorithm needs to be mapped to a VLSI architecture with small area and high data rate. Since Spatial Multiplexing is a recent technique, it is argued that there is still much room for development of related algorithms and architectures. Therefore, this thesis focused on the study of sub optimum algorithms and VLSI architectures for broadband MIMO detector with soft decision. As a result, novel algorithms have been developed starting from proposals of optimizations for already established algorithms. Based on these results, new MIMO detector architectures with configurable modulation and competitive area, performance and data rate parameters are here proposed. The developed algorithms have been extensively simulated and the architectures were synthesized so that the results can serve as a reference for other works in the area.

Keywords: MIMO, detection, demodulation, digital signal processing, architecture, VLSI, ASIC, FPGA.

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

$(.)^*$	conjugado transposto da matriz (.)
$(.)^t$	transposto da matriz (.)
$E(.)$	Esperança ou valor médio da variável aleatória (.)
E_s	Energia média do vetor transmitido s
H^+	Pseudo-inversa de Moore-Penrose da matriz H
M	Número de bits que mapeiam um símbolo da constelação
N_0	Energia do ruído em cada antena receptora
N_r	Número de antenas receptoras
N_t	Número de antenas transmissoras
R	Taxa de codificação
T	Taxa de transmissão de bits de informação por uso do canal
X	Conjunto de todos os possíveis valores para $\boldsymbol{x}$
Ω	Conjunto de valores que s_m pode assumir
Ω^{N_t}	Conjunto de valores que $\boldsymbol{s}$ pode assumir
$\hat{s}$	Estimação de s
$\boldsymbol{H}$	Matriz complexa do canal. Elementos independentes e identicamente distribuídos, com distribuição de Rayleigh
$\boldsymbol{G}_{\text{MMSE}}$	Matriz de equalização gerada a partir do método do MMSE
$\boldsymbol{G}_{\text{ZF}}$	Matriz de equalização gerada a partir do método do ZF
$\boldsymbol{I}_k$	Matriz identidade com dimensão $k \times k$

$\boldsymbol{R}$	Matriz triangular superior de dimensão $N_t \times N_t$ resultante da decomposição QR de $\boldsymbol{H}$
$\boldsymbol{n}$	Vetor complexo de ruído Gaussiano com dimensão N_r, média zero e desvio padrão σ
$\boldsymbol{r}$	O vetor de símbolos recebidos
$\boldsymbol{s}$	O vetor de símbolos transmitidos
$\boldsymbol{x}$	Vetor de bits que mapeiam o símbolo s
$\boldsymbol{z}$	O vetor de símbolos resultante da transformação $\boldsymbol{Q}_1^* \boldsymbol{r}$
σ^2	Variância do ruído
r_m	O m-ézimo elemento do vetor r, com $m = 1, 2, \ldots, N_r$
s_m	O m-ézimo elemento de $\boldsymbol{s}$, com $m = 1, 2, \ldots, N_t$
x_m	O m-ézimo bit de $\boldsymbol{x}$
ASIC	*Application-Specific Integrated Circuit*
AWGN	*Additive White Gaussian Noise*
BER	*Bit Error Rate*
ED	Distância Euclidiana (*Euclidean Distance*)
FPGA	*Field-Programmable Gate Array*
HDL	*Hardware Description Language*
LAN	*Local Area Network*
LDPC	*Low Density Parity Check (LDPC)*
LLR	*Log-Likelihood Ratio*
LSD	*List Sphere Detector - LSD*
LTE	*Long Term Evolution*
LUT	*Lookup Table*

MAP	*Maximum A Posteriori Probability*
MIMO	*Multiple Input Multiple Output*
ML	Máxima Verossimilhança (*Maximum Likelihood*)
MMSE	*Minimum Mean Square Error*
PED	*Partial Euclidian Distance*
QAM	*Quadrature Amplitude Modulation*
QPSK	*Quadrature Phase Shift Keying*
RTL	*Register-Transfer Level*
SFS	Busca Exaustiva Simplificada (*Simplified Full Search*)
SISO	*Single Input Single Output*
SNR	*Signal-to-Noise Ratio* $\frac{1}{\sigma^2}$
VLSI	*Very-Large-Scale Integration*
ZF	*Zero Forcing*

Capítulo 1

Introdução

Até o ano de 1990, aproximadamente, a telefonia celular era focada apenas na transmissão de voz. No período de 1990 a 2000, foram introduzidos no mercado celulares que possibilitaram a transmissão de mensagens de texto, seguidos dos PDAs (*personal digital assistant*) e *smartphones*, que vieram permitir acesso à Internet [1]. O serviço de transmissão de dados passou então a ser o centro das atenções. Devido aos constantes avanços na área de microeletrônica e telecomunicações, os padrões de sistemas de comunicação tiveram uma evolução constante na taxa de transmissão. Essa maior taxa de transmissão de dados dos sistemas de comunicação sem fio modernos tornou disponível um conjunto de serviços que eram inviáveis para os sistemas mais antigos. Entre esses serviços podem-se citar: *download* de músicas e vídeos, acesso a *sites* com conteúdo multimídia, teleconferência, entre outros. O aumento na taxa de transmissão das redes sem fio móveis, como são hoje chamadas as redes de celulares, foi seguido na mesma escala pelas redes sem fio de área mais limitada como a LAN (*Local Area Network*). O histórico da evolução dos sistemas de comunicação sem fio tem apresentado um passo de duplicar a taxa de transferência a cada 18 meses (Lei de Edholm) [2].

Infelizmente, sistemas de comunicação sem fio são limitados pela capacidade do canal de rádio. Segundo o teorema de Shannon [3] existe um limite para a taxa de transferência de dados em um canal com AWGN (*Additive White Gaussian Noise*) que é dado em função da potência e banda de transmissão do sinal. Essa taxa limite é referenciada com capacidade do canal, ou simplesmente de "capacidade". Aumentar a banda utilizada para transmissão nem sempre é uma solução viável para aumentar a capacidade, visto que o espectro é um recurso bastante caro. A potência usada para transmissão também é limitada, principalmente nos terminais móveis em que a fonte de energia é uma bateria. Além disso, existe uma potência máxima de transmissão estabelecida pelas agências reguladoras para impedir que

haja interferência em outros sistemas. Por isso, para atender à crescente demanda por taxa de transferência de dados e qualidade de serviço, são necessários novos algoritmos e arquiteturas que explorem de forma mais eficiente a banda disponível e que tornem o sistema mais robusto aos fatores que prejudicam a transmissão, ou seja, que consigam atingir taxas de transmissão mais próximas à capacidade do canal.

Pesquisas recentes na área de teoria da informação mostraram, com base no teorema de Shannon, que um aumento na capacidade de comunicação em sistemas sem fio pode ser obtido com o uso de múltiplas entradas e múltiplas saídas (MIMO - *Multiple Input Multiple Output*) [4] [5]. Isto é, com o uso de mais de uma antena no receptor e no transmissor é possível aumentar a taxa de transmissão sem alterar a banda e a potência usada para transmissão. Existem basicamente dois motivos para esse ganho na capacidade dos sistemas MIMO sobre os sistemas de única entrada e única saída (SISO - *Single Input Single Output*): a diversidade espacial (*space diversity*), e a multiplexação espacial (*spatial multiplexing*).

A ideia básica da diversidade é transmitir múltiplas vezes o mesmo sinal de forma que haja uma correlação muito baixa do *fading* [1] de cada cópia no receptor. Assim, a probabilidade de todas as cópias do sinal sofrerem forte atenuação é mais baixa quanto maior for o número de cópias enviadas. Nos sistemas SISO a diversidade só pode ser obtida no domínio do tempo ou da frequência, ou seja, as diversas cópias do sinal são transmitidas ou em diferentes instantes de tempo e/ou em diferentes bandas de frequência. Desse modo, o uso da diversidade em sistemas SISO consome recursos de transmissão, o que limita o benefício da técnica. Num sistema MIMO, por sua vez, existe a possibilidade de aplicar a diversidade em um outro domínio, o espaço. Isto é, explorar os diversos canais de comunicação existentes em um sistema MIMO, cada um formado por uma diferente combinação entre um transmissor e um receptor [7]. Existem basicamente duas categorias de diversidade espacial: diversidade de recepção e diversidade de transmissão. Na diversidade de recepção, existem múltiplas antenas receptoras, cada uma captando uma cópia do sinal transmitido com um *fading* e ruído independente. As múltiplas cópias são, então, combinadas no receptor de forma a maximizar a relação sinal-ruído do sinal resultante. Quando a correlação entre o *fading* dos diferentes canais de comunicação é baixa, a probabilidade dos sinais de todos os receptores estarem fortemente atenuados é menor quanto maior for o número de antenas receptoras. Na diversidade

[1] O termo *fading* é usado na área de sistemas de comunicação para descrever o ganho em amplitude e a mudança de fase no sinal devido ao canal de propagação [6]

de transmissão, um mesmo sinal é transmitido por múltiplas antenas transmissoras com um ajuste de amplitude e fase sendo aplicado no sinal de cada transmissor de forma a maximizar a interferência construtiva desses sinais na antena receptora e, com isso, obter um ganho no SNR [7]. Quando existem múltiplas antenas tanto no receptor quanto no transmissor é possível combinar as duas técnicas maximizando o ganho na capacidade devido à diversidade espacial.

A multiplexação espacial (*Spatial Multiplexing - SM*), por sua vez, consiste em transmitir dados diferentes por cada uma das antenas transmissoras na mesma frequência, formando, desse modo, canais em paralelo de transmissão. O número de canais em paralelos que podem ser criados é limitado pelo mínimo entre o número de antenas transmissoras e receptoras [5] [7]. O uso da multiplexação espacial na condição de SNR alto permite um aumento aproximadamente linear na capacidade de transmissão com o número de transmissões em paralelo, provendo um ganho de capacidade muito superior ao obtido com a diversidade espacial nessa mesma condição [5] [7].

Para atingir uma taxa de transmissão próxima à capacidade é preciso fazer uso de uma técnica de codificação de canal de alto desempenho, como o *turbo-code* [8] e o *Low Density Parity Check* (LDPC) [9]. Essas e outras técnicas de codificação podem ser aplicadas em sistemas MIMO de forma a introduzir correlação no sinal no domínio do tempo e também no espaço, ou seja, entre os sinais de cada transmissor. Quando isto é feito, a codificação recebe a classificação de *space-time code* [7]. Para maximizar o desempenho do processo de decodificação de canal, o detector utilizado deve prover uma saída suave. Isto é, a saída do detector deve fornecer um grau de certeza sobre a detecção de cada bit. A complexidade computacional de um detector MIMO de decisão suave ideal para sistemas com multiplexação espacial é extremamente alta, o que o inviabiliza em soluções práticas. Assim, existe o desafio do desenvolvimento de algoritmos de detecção MIMO que se aproximem do desempenho do detector ideal, mas mantendo um nível de complexidade computacional aceitável para uma implementação prática.

O ganho na capacidade de transmissão oferecido pelo MIMO fez com que este sistema fosse adotado em vários padrões de sistema de comunicação móvel já comercializados, tais como o HSPA , o LTE, o IEEE 802.22 e o IEEE 802.16m, também conhecido como WiMax. Devido à complexidade computacional dos algoritmos de processamento de sinais para MIMO e à grande demanda no mercado por produtos com essa tecnologia, as soluções são normalmente implementadas em circuitos VLSI (*Very-large-scale integration*) do tipo ASIC (*Application-Specific Integrated*

Circuits), por permitirem alta taxa de processamento com baixa potência de consumo e resultarem em baixo custo por unidade, quando produzidos em larga escala. Na fase de desenvolvimento e teste, os circuitos são inicialmente prototipados em tecnologia FPGA (*Field-programmable Gate Array*) devido a capacidade de reprogramação [10] e baixo NRE (*Non-recurring engineering*), ou custo de desenvolvimento. Portanto, existe também o problema do mapeamento do algoritmo de detecção MIMO para uma arquitetura VLSI de área reduzida que vise baixa potência.

Por o MIMO ser uma tecnologia recente, defende-se que ainda há muito espaço para evolução dos algoritmos e das arquiteturas relacionadas. Assim, o objetivo dessa tese é o desenvolvimento de algoritmos de detecção MIMO que melhorem a relação entre complexidade do algoritmo e desempenho do mesmo, bem como o mapeamento desses algoritmos para uma arquitetura VLSI. Dentro desse contexto, decidiu-se abordar o problema de detecção com decisão suave para sistemas de banda larga com esquema de transmissão MIMO 2x2 e 4x4 com multiplexação espacial. Além disso, foi dada atenção ao problema da configurabilidade da modulação, o que representa um desafio para a arquitetura VLSI. Abaixo, são citadas as contribuições dessa tese:

- Busca Exaustiva Simplificada: Otimização do método da busca exaustiva para obtenção da solução max-log-ML, aproximação da Máxima Verossimilhança (ML), no problema da deteção suave.
- K-Melhores Espalhado: Variação do algoritmo clássico de detecção K-Melhores que resulta em uma melhor implementação em hardware do que o algoritmo original. Enquanto que no K-Melhores são selecionados as K melhores hipóteses entre todas em cada estágio de processamento da detecção, o K-Melhores Espalhados separa as hipóteses de um estágio em N grupos e seleciona as K/N melhores de cada grupo. Como a implementação em hardware do classificador dos K melhores possui um caminho crítico muito longo, que é proporcional ao valor de K, o K-Melhores Espalhados permite operar com uma frequência de clock maior, ou utilizar um valor maior para o parâmetro K, mantendo a mesma frequência de clock. O desempenho do algoritmo K-Melhores Espalhados é comparado com o K-melhor clássico para diversas configurações de K e N em uma transmissão MIMO 4x4.
- Detector MIMO 2x2: Arquitetura de um detector MIMO 2x2 baseado no algoritmo da Busca Exaustiva Simplificada e o resultado de sua síntese em ASIC, que obteve área menor que as soluções no estado da arte com desempenho

equivalente.

Em seu delineamento, esta tese está estruturada em cinco capítulos onde: O Capítulo 1 corresponde à apresentação formal da tese, sua fundamentação e seus direcionamentos; o Capítulo 2 descreve o problema da detecção MIMO e as soluções clássicas para o problema; o Capítulo 3 apresenta o Busca Exaustiva Simplificada; O Capítulo 4 detalha o K-Melhores e apresenta o K-Melhores Espalhados; o Capítulo 5, apresenta a arquitetura VLSI de um detector 2x2 que implementa o algoritmo Busca Exaustiva Simplificada, o fluxo de projeto adotado para implementação dessa arquitetura em ASIC e o resultado de sua síntese lógica, estágio do projeto em que o projeto se encontrava quando essa tese foi escrita; o Capítulo 6, concluí a tese relacionando os resultados obtidos com os objetivos estabelecidos, e aponta vertentes e perspectivas para futuras pesquisas.

Capítulo 2

Multiplexação Espacial

Sistemas de comunicação com Múltiplas Entradas e Múltiplas Saídas (MIMO - *Multiple Input Multiple Output*) são sistemas que empregam múltiplas antenas na transmissão e na recepção para aumentar a capacidade de comunicação. Entre as técnicas de transmissão para sistemas MIMO está a multiplexação espacial, que consiste em transmitir diferentes dados por cada uma das antenas transmissoras na mesma frequência. Esta técnica oferece um ganho na capacidade de transmissão linearmente proporcional ao mínimo entre número de antenas transmissoras e receptoras. Entretanto, associado a esse ganho existe um grande aumento na complexidade do processo de detecção no receptor conforme será visto. Neste capítulo, o sistema MIMO com multiplexação espacial será descrito juntamente com as métricas e metodologias para analisar o desempenho de um receptor MIMO. A fim de atingir esse objetivo será inicialmente apresentado o modelo do sistema MIMO adotado nas simulações realizadas, na Seção 2.1. Em seguida, a capacidade de transmissão de um sistema MIMO para esse modelo de canal é apresentado na Seção 2.2. As métricas de desempenho de um sistema MIMO e os métodos usados para suas medições são descritos na Seção 2.3. Os algoritmos clássicos para detectores MIMO com detecção abrupta e suave são expostos nas Seções 2.4 e 2.5, respectivamente. Por fim, na Seção 2.6 alguns dos algoritmos recentes e mais relevantes são apresentados, juntamente com os resultados de suas implementações.

2.1 Modelo Sistema MIMO

A Figura 2.1 ilustra elementos básicos que compõem um sistema de comunicação digital. A funcionalidade de cada um desses elementos e suas particularidades no modelo usado nessa tese são descritos a seguir. A fonte de dados no início desse sistema é uma fonte binária aleatória que provê uma sequência de bits de informação

ao codificador de canal, primeira etapa de processamento. Os bits na saída da fonte de dados têm igual probabilidade de ser 0 ou 1.

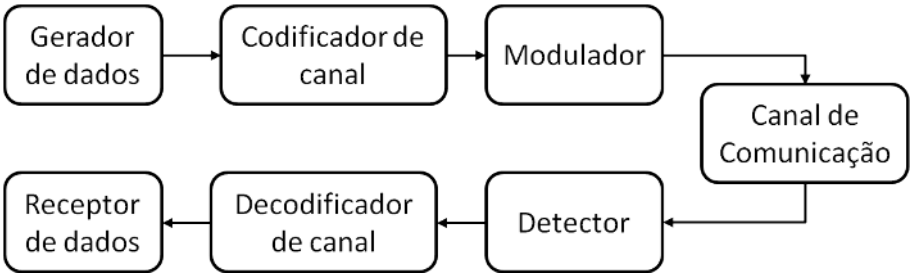

Figura 2.1: Diagrama de blocos de um sistema de comunicação digital.

2.1.1 Codificador de Canal

O propósito do codificador de canal é introduzir, de maneira controlada, redundância na sequência de informação binária a fim de permitir ao receptor superar os efeitos do ruído e de interferências existentes na transmissão do sinal através do canal. Portanto, a redundância adicionada serve para ajudar o receptor a decodificar corretamente a informação. Um exemplo de uma codificação de canal trivial é simplesmente repetir cada dígito binário a ser transmitido m vezes, sendo m um número positivo qualquer. Técnicas de codificação mais sofisticadas envolvem pegar k bits de informação por vez e mapear cada sequência de k-bit em um sequência única de n-bits, chamada de palavra de código [11] [6]. A quantidade de redundância inserida dessa maneira é medida pela razão entre n e k. A razão inversa,

$$R = \frac{k}{n}, \tag{2.1}$$

é chamada de taxa de codificação e determina a quantidade de informação que cada bit na saída do codificador carrega. Assim, um R menor se traduz em uma maior robustez contra ruído do canal devido à maior redundância, e também em menor taxa de informação por bits transmitidos.

Além da taxa de codificação, existem outros parâmetros que afetam a confiabilidade no processo de decodificação, como a técnica de codificação usada e o tamanho da palavra de código. Segundo Shannon, se a taxa de transmissão for menor que a capacidade do canal, a probabilidade de erro na decodificação pode ser reduzida arbitrariamente aumentando-se o tamanho da palavra de código [3]. Entretanto, na

prática não é possível aumentar indefinitivamente o tamanho da palavra de código por dois motivos. Primeiro, a complexidade computacional do processo de decodificação aumenta com o aumento do tamanho da palavra de código, e segundo, o processo de decodificação precisa esperar que todos os bits da palavra de código sejam recebidos para realizar a decodificação, o que gera um inevitável incremento na latência da transmissão [9].

Entre os diversos algoritmos para codificação de canal, o *turbo code* [8] e o *Low Density Parity Check (LDPC)* [9] são os mais adotados por padrões recentes. O turbo code, por exemplo, foi adotado pelos padrões HSDPA, LTE e LTE-Advanced, enquanto que o LDPC se encontra nos padrões IEEE 802.11 e IEEE 802.22. No modelo dessa tese será utilizado o codificador do tipo LDPC, usando uma taxa de codificação de 0,5 e um tamanho de palavra de código de 2.304 bits, e o número máximo de iterações do decodificador sendo 20. A escolha pelo LDPC foi motivada por já existir uma função na biblioteca do Matlab que a implementa. Como o processo de codificação de canal não é o tema central dessa tese, o algoritmo LDCP não será descrito aqui.

2.1.2 Modulador

A sequência binária na saída do codificador de canal é passada para o modulador digital, que serve como interface com o canal de comunicação. O propósito do modulador digital é mapear uma sequência binária em um sinal com duração finita a ser transmitido pelo canal de comunicação. Para melhor explicar essa funcionalidade, suponha o caso em que a informação codificada será transmitida um bit por vez. O modulador digital, então vai mapear o dígito binário 0 em uma forma de onda $s_0(t)$ e o digito binário 1 em uma forma de onda $s_1(t)$. Sendo o tempo de duração dessas duas formas de onda iguais. Generalizando, se o número de bits transmitidos por uso do canal for M, então é definido um conjunto Ω de formas de ondas com mesmo tempo de duração, sendo o tamanho do conjunto igual a $|\Omega| = 2^M$.

Um conjunto de símbolos é geralmente utilizado para representar as formas de onda geradas pelo modulador em um modelo em banda base e tempo discreto de um sistema de comunicação digital. No caso em que as diversas formas de ondas são senoides com mesma frequência, diferindo uma das outras pela amplitude e fase, os símbolos costumam ser números complexos cuja amplitude e fase representam a modulação aplicada à portadora do transmissor para geração da respectiva onda. Neste caso, o conjunto Ω compreende esses símbolos e é chamado de conste-

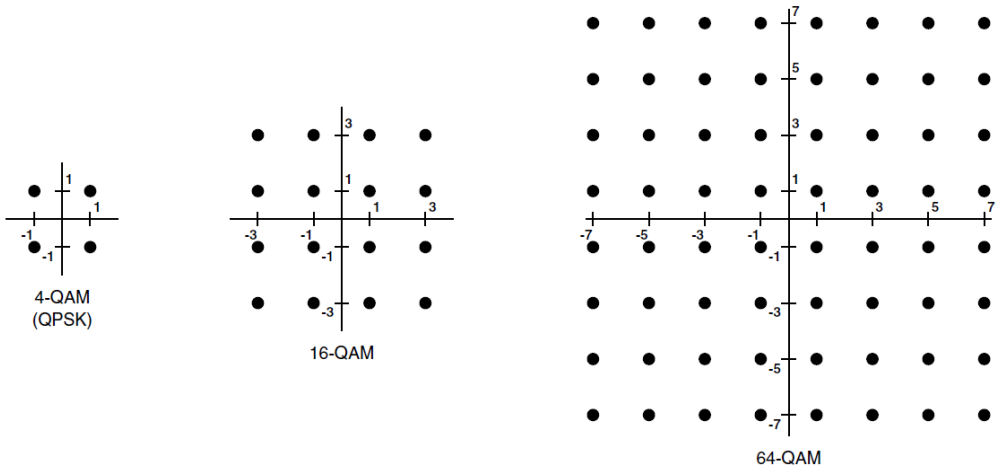

Figura 2.2: Constelação para as modulações QPSK, 16-QAM e 64-QAM.

lação. Como exemplo têm-se as constelações QPSK (*quadrature phase-shift keying*), 16QAM (*Quadrature Amplitude Modulation*) e 64QAM utilizadas nas simulações dessa tese, e que permitem respectivamente a transmissão de 2, 4 e 6 bits por símbolo transmitido. Essas constelações estão apresentadas nos planos cartesianos expostos na Figura 2.2, em que os possíveis símbolos correspondem aos pontos no plano. Normalmente, metade dos bits mapeia a componente real do símbolo e a outra metade, a componente imaginária. O mapeamente é feito de forma que símbolos adjacentes na constelação difiram no vetor de bits que o mapeiam em um único bit, ou seja, o mapeamento segue um código de Gray [12].

No sistema MIMO tem-se múltiplos transmissores. Por isso, a saída do modulador digital MIMO é um vetor de símbolos $\boldsymbol{s} = [s_1 s_2 \cdots s_{N_t}]^t$ com N_t sendo o número de antenas transmissoras e $.^t$ o operador de transposição. Cada componente de $\boldsymbol{s}$ é então um número complexo que define a modulação aplicada a uma portadora. No modelo escolhido, a mesma constelação de modulação é usada para todas as portadoras e nenhuma técnica de diversidade de transmissão é empregada. Assim, tem-se um vetor de bits $\boldsymbol{x} = [x_1\, x_2 \cdots x_{N_t M}]$ mapeando um vetor de símbolos $\boldsymbol{s}$ pertencente a um conjunto Ω^{N_t}, definido pela constelação adotada. Esse mapeamento pode ser descrito por

$$\boldsymbol{x} \rightarrow \boldsymbol{s} \tag{2.2}$$

Sendo que cada componente de $\boldsymbol{s}$ é mapeado por um conjunto distinto de M bits

pertencente a $\boldsymbol{x}$,

$$[x_{(m-1)M+1}\ x_{(m-1)M+2} \cdots x_{mM}] \rightarrow s_m \tag{2.3}$$

sendo s_m pertencente a Ω.

Como o mapeamento dos elementos de X (conjunto dos possíveis valores para $\boldsymbol{x}$) para os elementos de Ω^{N_t} é de um para um, o mapeamento inverso faz sentido,

$$\boldsymbol{s} \rightarrow \boldsymbol{x}. \tag{2.4}$$

As potências médias transmitidas em cada antena transmissora são idênticas e dadas por

$$E\|s_m\|^2 = \frac{E_s}{N_t}, \tag{2.5}$$

sendo E o operador esperança e $E_s = E\|\boldsymbol{s}\|^2$ a potência média total transmitida.

Por fim, é importante salientar que a correlação entre os bits que compõem $\boldsymbol{x}$ é nula, apesar de serem parte de um mesmo bloco de código. Isto se deve ao algoritmo de codificação que opera de forma a alcançar essa característica [13]. Assim, pode-se dizer que a correlação entre os elementos de $\boldsymbol{s}$ também é aproximadamente nula.

2.1.3 Canal de Comunicação MIMO

O modelo em banda base e tempo discreto do canal de comunicação MIMO é dado por

$$\boldsymbol{r} = \boldsymbol{H} \cdot \boldsymbol{s} + \boldsymbol{n}, \tag{2.6}$$

sendo $\boldsymbol{r} = [r_1\, r_2 \cdots r_{N_r}]^t$ o vetor de símbolos recebidos com N_r representando o número de antenas receptoras, $\boldsymbol{n}$ o vetor ruído com dimensão $N_r \times 1$ e com cada elemento sendo uma variável complexa com distribuição gaussiana independente com média zero e variância σ^2. A matriz de covariância do ruído é dada por

$$E[\boldsymbol{n} \cdot \boldsymbol{n}^*] = \sigma^2 \cdot \boldsymbol{I}_{N_r}, \tag{2.7}$$

em que $(.)^*$ é o conjugado transposto da matriz $(.)$, e $\boldsymbol{I}_{N_r}$ é a matriz identidade com dimensão $N_r \times N_r$.

A matriz complexa $\boldsymbol{H}$ de dimensão $N_r \times N_t$ representa o ganho do canal entre cada transmissor e receptor. Um exemplo desse modelo é dado pela Figura 2.3, onde um sistema MIMO 3x3 é apresentado.

Escolheu-se utilizar um modelo de canal do tipo Rayleigh com distribuição in-

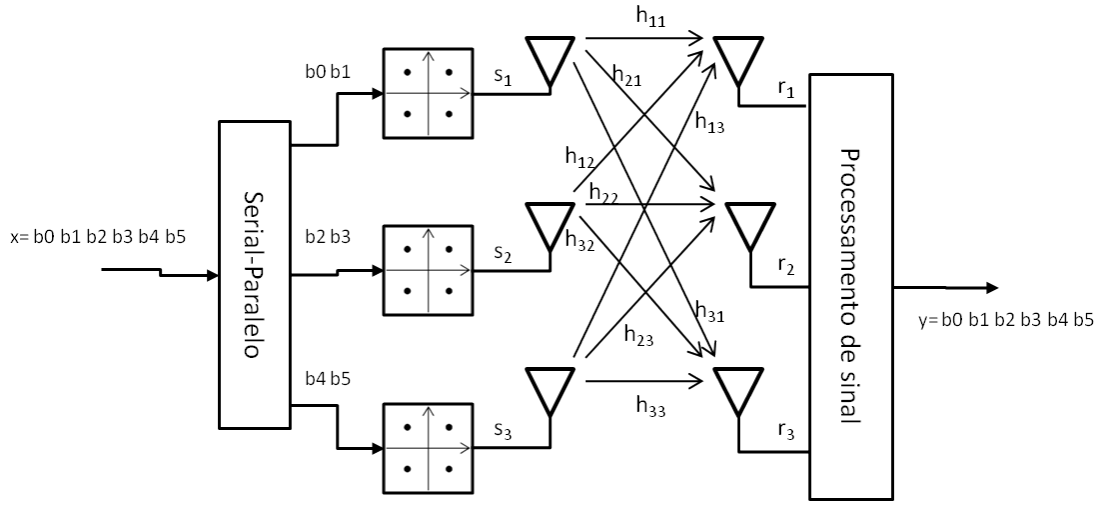

Figura 2.3: Canal MIMO caracterizado por uma matriz 3x3.

dependente e idênticamente distribuida (i.i.d.). Ou seja, os elementos da matriz $\boldsymbol{H}$ serão variáveis complexas aleatórias independentes uma das outras com distribuição Gaussiana com média zero e variância $\frac{1}{2}$ por componente real. Quanto ao tempo de coerência do canal, período de tempo em que pode-se considerar que a resposta do canal permance constante, o canal será caracterizado como *fast fading*, randomização da matriz $\boldsymbol{H}$ feita em período de símbolo. Outra opção seria manter a matriz $\boldsymbol{H}$ constante por um número constante de transmissões consecutivas e só depois randomiza-la. Nesse caso, o modelo é classificado com *block fading*.

A distribuição Gaussiana com média zero representa bem a condição em que não há uma visada direta entre tranmissor e receptor, e assim, a transmissão se dá principalmente por reflexões [14] [11] [6]. Por sua vez, a característica de distribuição independente modela a condição em que as antenas possuem baixa correlação entre seus *fading*. Essa característica é muito desejada em sistemas MIMO porque o ganho oferecido pela técnica da multiplexação espacial, e também da diversidade espacial, é degradado quando há correlação [7]. Na prática, uma baixa correlação entre os *fading* pode ser alcançada afastando suficientemente as antenas uma das outras ou atribuindo polarização diferente para cada antena. Em ambientes ricos em reflexões, é possível direcionar o ganho das antenas para diferentes direções, o que pode reduzir o afastamento necessária para obtênção de uma baixa correlação para menos que um comprimento de onda, viabilizando assim o uso da multiplexção espacial em dispostivos pequenos como celulares [15]. O tempo de coerência do canal é inversamente proporcional ao efeito Doppler, ou seja, quanto maior for o

deslocamento entre transmissor e receptor e maior for a frequencia da portadara menor será o tempo de coerência do canal [11]. Sendo assim, o *fast fading* representa bem o caso em que existe um movimento constante do transmissor ou do receptor. Além disso, existem técnicas de transmissão que visam atingir uma descorrelação da matriz $\boldsymbol{H}$ em cada uso do canal como o frequency-hopping [15], que consiste em alternar constantemente a banda usada para transmissão [1].

O número máximo de símbolos independentes transmitidos deve ser igual ou menor que $N = \min(N_r, N_t)$. Isto se dá porque a dimensão do vetor s é limitada por N_t e o receptor é incapaz de cancelar mais do que $N_r - 1$ sinais de interferência [7]. Quando $N_t > N_r$, a solução é utilizar redundância na transmissão de forma que o número de símbolos transmitidos efetivamente por uso do canal seja igual a N_r, como em [16]. Neste estudo, no entanto, será considerado apenas o caso em que $N_r \geq N_t$. Quando $N_r > N_t$, o processamento geralmente consiste em converter a Equação (2.6) com matriz $\boldsymbol{H}$ retangular em uma equação equivalente em que a nova matriz canal é quadrada de dimensão $N_t \times N_t$. Esta nova equação é obtida através da multiplicação de ambos os lados de (2.6) por uma matriz de equalização. Existem diferentes métodos para conversão da Equação (2.6) com matriz retangular em uma equação equivalente com matriz quadrada. Na Seção 2.4 alguns métodos serão apresentados. O uso de mais antenas receptoras do que transmissoras possibilita um ganho na capacidade devido a diversidade de recepção. Assim um sistema MIMO 4x2 está limitado ao mesmo ganho de capacidade por multiplexação que pode ser obtido com um MIMO 2x2, sendo que oferece um ganho adicional devido à diversidade de recepção. A capacidade de transmissão do canal MIMO adotado nas simulações dessa tese para diferentes configurações no número de antenas transmissoras e receptoras é apresentada na Secção 2.2.

A taxa de transmissão em bits de informação por uso do canal no modelo de sistema MIMO escolhido é dada por

$$T = R \cdot N_t \cdot M \text{ bits/(uso do canal)}. \tag{2.8}$$

Isto porque cada $\boldsymbol{s}$ transmitido é mapeado por $N_t \cdot M$ bits que carregam cada um R bits de informação.

A relação sinal ruído, SNR (*Signal-to-noise ratio*), em sistema MIMO é definida como sendo a razão entre a energia (média) do sinal transmitido E_s, e a energia

média do ruído em cada antena receptora $N_0 = \sigma^2$.

$$SNR = \frac{E_s}{\sigma^2} \tag{2.9}$$

2.1.4 Detector e Decodificador

A função do detector é obter a estimativa mais precisa do vetor de bits transmitido $\boldsymbol{x}$, tendo conhecimento do vetor recebido $\boldsymbol{r}$, da matriz $\boldsymbol{H}$ e, opcionalmente, da variança do ruído de canal σ^2. Neste modelo de transmissão o canal $\boldsymbol{H}$ é considerado perfeitamente estimado pelo receptor e a modulação utilizada na transmissão também é conhecida. Na prática, o canal $\boldsymbol{H}$ é estimado numa fase de treinamento, em que um sinal de referência é transmitido por um transmissor enquanto os outros ficam em silêncio, sendo esse procedimento feito para todos os transmissores. O período ideal entre as medições depende do tempo de coerência do canal, que é inversamente proporcional ao efeito Doppler [11] [6]. Assim, quando não há deslocamento entre transmissor e receptor, ou a velocidade do deslocamento é baixa, o canal pode ser considerado constante durante várias transmissões seguidas, e a ocupação do canal pelo processo de estimação de $\boldsymbol{H}$ pode ser reduzida. Na prática, sistemas de comunicação digital sem fio, geralmente, fixam a taxa de medição de $\boldsymbol{H}$ e estabelecem uma velocidade limite de deslocamento do usuário para a qual o funcionamento do sistema é garantido [17].

O decodificador opera após receber a saída do detector para todos os bits que compõem um bloco de código. Sua função é utilizar a informação passada pelo detector, juntamente com o conhecimento da codificação usada pelo codificador de canal e da redundância contida nos dados recebidos para tentar corrigir as estimações feitas pelo detector e, assim, reconstruir corretamente a sequência original de bits de informação.

Os detectores podem ser classificados como sendo de detecção abrupta ou suave. Os detectores de decisão abrupta geram uma estimativa do vetor de símbolo transmitido, $\hat{\boldsymbol{s}}$, e em seguida fazem o mapeamento $\hat{\boldsymbol{x}} = map(\hat{s})$, obtendo assim uma estimativa dos bits transmitidos. Os detectores de decisão suave, por sua vez, fornecem um nível de certeza sobre o valor de cada bit individualmente. Esse nível de certeza é normalmente expresso pelo *Log-likelihood ratio - LLR* ou uma aproximação desse, cujo cálculo será explicado na Secção 2.5. O resultado do cálculo do LLR é um número no domínio dos reais que quanto mais negativo indica uma maior probabilidade do bit transmitido ser 0 e quanto mais positivo uma maior probabilidade

do bit ser 1.

Além disso, os detectores de decisão suave podem ser subdivididos em detectores iterativos ou não-iterativos. Nos iterativos, existe um processo de realimentação entre decodificador e detector. Esse processo ocorre da seguinte forma: o decodificador retorna para o detector novos valores para o LLR dos bits que compõem o bloco de código. Essa sequencia de LLR devolvida é gerada usando o conhecimento da correlação dos bits no bloco de código. O detector, então repete o processo de detecção dos vetores $\boldsymbol{x}$ que compõem o bloco de código, sendo que dessa vez usando os LLRs passados pelo decodificador como conhecimento inicial, ou *apriori*, sobre os bits a serem detectados. Assim, novos LLRs são gerados pelo detector e repassados para o decodificador. A interação entre detector e decodificador pode ocorrer até que um número pré-determinado de iterações seja atingido, ou até que se detecte que o processo tenha convergido para uma determinada solução.

Apesar de oferecer um desempenho superior [18] [13], o processo de iteração aumenta significativamente a complexidade computacional do processo de detecção. Por isso, nem sempre é utilizada na prática.

A Figura 2.4 mostra o diagrama de bloco de um transmissor e receptor MIMO com codificação de canal e que permite a iteração entre decodificador e detector.

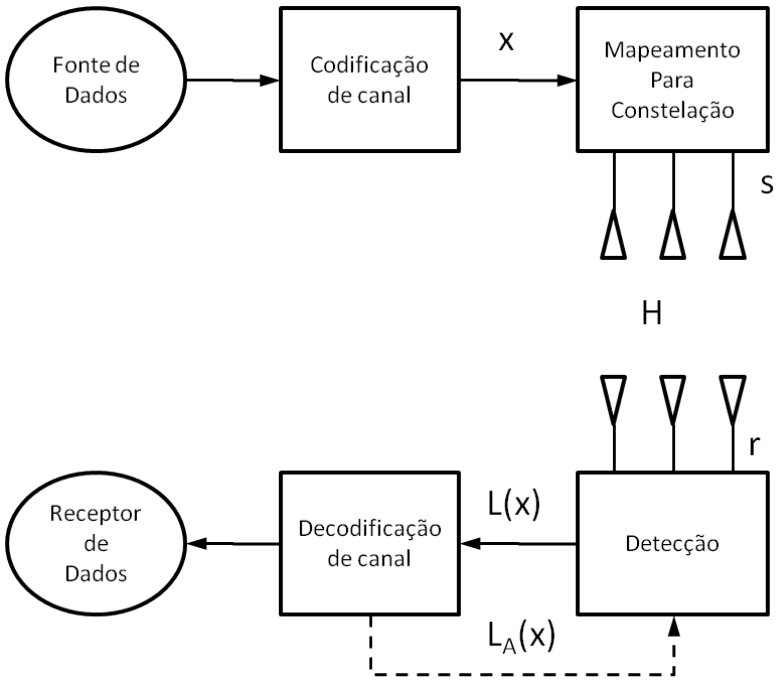

Figura 2.4: Diagrama de blocos de um sistema MIMO com codificação de canal.

Por utilizarem a informação do nível de certeza sobre a detecção dos bit, os decodificadores que operam em cima de decisão suave apresentam um desempenho superior aos que trabalham com decisões rígidas. Existe, no entanto, um grande desafio no desenvolvimento de um detector de decisão suave para sistemas MIMO que é a complexidade computacional do cálculo do LLR dos bits. Mesmo sem a interação com o decodificador, esse cálculo ainda possui uma alta complexidade computacional, como será demonstrado nessa tese.

2.2 Capacidade do Canal MIMO

A capacidade é definida como sendo a máxima taxa de transferência em que é possível obter uma taxa de erro arbitrariamente baixa. A capacidade para o modelo de canal apresentado na secção 2.1 é dada pela fórmula [7] [13]

$$C/W = E \log \det \left(\boldsymbol{I}_{N_r} + \frac{E_s}{\sigma^2 N_t} \boldsymbol{H}^* \boldsymbol{H}, \right) \tag{2.10}$$

em que, C/W é a capacidade por uso do canal e representa a máxima taxa de bits de informação por realização de (2.6). A Figura 2.5 apresenta a curva da capacidade para diferentes configurações de sistemas MIMO. Observe que o ganho de diversidade de recepção oferecido pela configuração 4x2 em relação a 2x2 se traduz em um ganho de SNR, já que não há uma mudança de inclinação na curva, apenas um deslocamento da mesma.

É importante lembrar que a constelação usada para transmissão limita a taxa máxima de transferência. Para analisar o efeito da constelação na taxa máxima de transferência, é necessário calcular a informação mútua entre a variável aleatória de saída $\boldsymbol{r}$ e a de entrada $\boldsymbol{s}$. A informação mútua é dada por

$$I(\boldsymbol{s}, \boldsymbol{r}) = \int_r \sum_{\boldsymbol{s} \in \Omega^{N_t}} p(\boldsymbol{r}, \boldsymbol{s}) \cdot \log_2 \left(\frac{p(\boldsymbol{r}, \boldsymbol{s})}{p(\boldsymbol{r}) \cdot p(\boldsymbol{s})} \right) dr, \tag{2.11}$$

em que $p(\boldsymbol{r}, \boldsymbol{s})$ é a densidade de probabiliade de um determinado $\boldsymbol{r}$ e $\boldsymbol{s}$ ocorrerem simultâneamente [11].

Uma vez que todos os símbolos têm igual probabilidade de serem transmitidos tem-se $p(\boldsymbol{s}) = \frac{1}{|\Omega^{N_t}|}$ para $\forall \boldsymbol{s} \in \Omega^{N_t}$. Fazendo as substituições $p(\boldsymbol{r}, \boldsymbol{s}) = p(\boldsymbol{r}|\boldsymbol{s}) \cdot p(\boldsymbol{s})$ e $p(\boldsymbol{r}) = \sum_{\boldsymbol{s} \in \Omega^{N_t}} p(\boldsymbol{r}|\boldsymbol{s}) \cdot p(\boldsymbol{s})$, em que $p(\boldsymbol{r}|\boldsymbol{s})$ é a densidade de probabilidade de $\boldsymbol{r}$

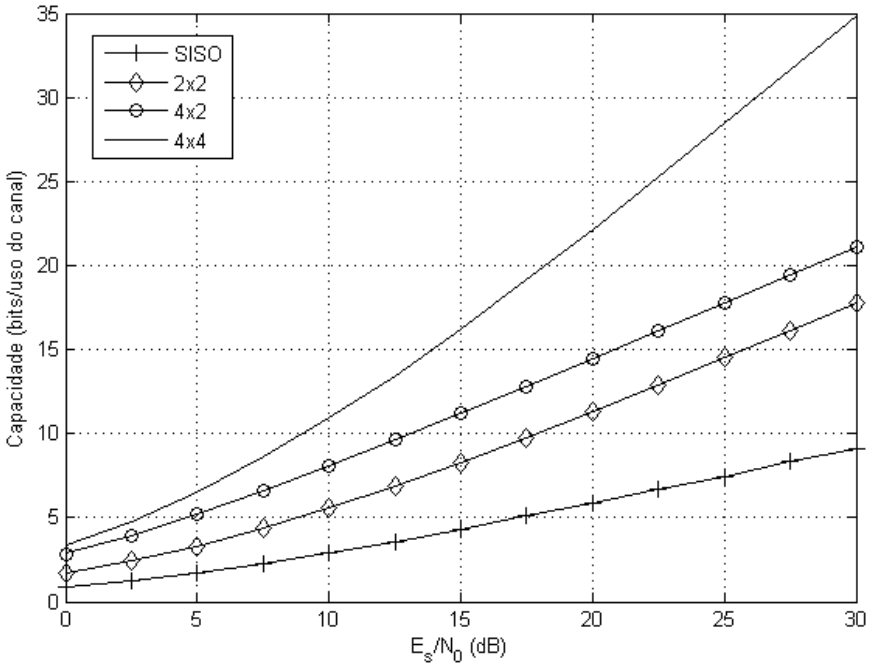

Figura 2.5: Capacidade do canal MIMO do tipo Rayleigh para diferentes configurações de $N_r \times N_t$.

condicionado $\boldsymbol{s}$, e desenvolvendo a equação chega-se há equação

$$I(\boldsymbol{s},\boldsymbol{r}) = N_t M + E\left[\sum_{\boldsymbol{s}\in\Omega^{N_t}} \frac{p(\boldsymbol{r}|\boldsymbol{s})}{\sum_{\boldsymbol{s}\in\Omega^{N_t}} p(\boldsymbol{r}|\boldsymbol{s})} \log_2 \frac{p(\boldsymbol{r}|\boldsymbol{s})}{\sum_{\boldsymbol{s}\in\Omega^{N_t}} p(\boldsymbol{r}|\boldsymbol{s})}\right], \tag{2.12}$$

em que a integral foi aproximada pela operação esperança sobre a variável aleatória $\boldsymbol{r}$, gerada a partir das outras três variáveis aleatórias: $\boldsymbol{s}$, $\boldsymbol{n}$, $\boldsymbol{H}$. Observe que o cálculo de (2.12) se torna complexo quando $|\Omega| = 2^{N_t M}$ é muito grande. Neste caso a equação precisa ser organizada de forma que o somatório sobre todas as possibilidades de $\boldsymbol{s}$ seja substituído por uma operação esperança em relação a variável $\boldsymbol{s}$.

O resultado do cálculo de (2.12) para diferentes constelações para configuração de 2 transmissores e 2 receptores é mostrado na Figura 2.6. A capacidade é representada pela curva mais alta. As demais curvas delimitam a taxa máxima de transmissão para uma determinada constelação. Observe que a taxa máxima de transmissão usando a modulação 64-QAM e MIMO 2x2 é de 12 bits por uso do canal, conforme esperado já que essa é a taxa que se obtém na condição de redundância nula na informação transmitida, R=1. O algoritmo MatLab usado para gerar esse gráfico se

encontra no Anexo A.

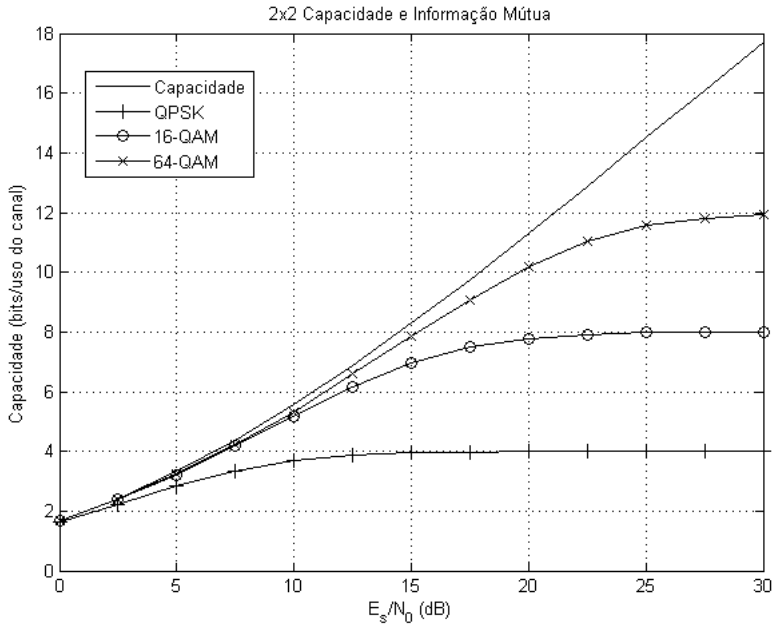

Figura 2.6: Capacidade do canal MIMO 2x2.

Agora, considere que se queira tentar atingir uma taxa de transmissão C na curva de capacidade. Para transmitir dados em uma taxa C é preciso escolher uma constelação de tamanho 2^{MN_t} e uma taxa de codificação R de tal forma que $RMN_t = C$, uma vez que segundo o teorema da codificação de canal, uma transmissão sem erro somente é possível para $RMN_t < C$ [3]. Em seguida, é preciso verificar a Figura 2.6 para se certificar que a informação mútua da constelação escolhida fica próxima da capacidade do canal no ponto C. Por exemplo, suponha que deseja-se atingir $C = 6$ com $\frac{Es}{N0} \approx 11dB$. Uma possibilidade é escolher um constelação de símbolo 64-QAM e uma taxa de codificação de canal R=1/2, o que resulta na taxa de transmissão de 6 bits por uso do canal. A Figura 2.6 confirma que a informação mútua para uma constelação de 64-QAM a uma taxa de 6 bits por uso de canal é muito próxima a capacidade. Na simulação do sistema deve-se então verificar a taxa de erro de transmissão tendendo a zero em uma $\frac{Es}{N0}$ acima de 11 dB. A diferença entre o SNR necessário para que o erro do sistema tenda a zero e o SNR da capacidade informa o quão perto conseguiu-se chegar da capacidade.

2.3 Análise do Sistema

Um detector MIMO deve atender os modos de operação e a taxa de processamento exigidos pela aplicação a que se destina. Os sistemas de banda larga visionados para os próximos anos exigem detectores MIMO configuráveis para diversos esquemas de configuração de antena e modulação (QPSK, 16QAM, 64QAM), e taxas de transmissão superiores a 1 Gbps [19]. Além desses requisitos, existem métricas qualitativas como: desempenho, potência e área de implementação em ASIC, que determinam se uma solução é comercialmente viável. Nesta seção serão apresentadas as métricas de um detector MIMO e os métodos para avaliação dessas métricas.

2.3.1 Taxa de processamento

Existem duas taxas de processamento importantes para um sistema MIMO. A taxa de atualização da matriz H e a taxa de transmissão de símbolos, sendo a primeira inferior ou no máximo igual a segunda. Normalmente, um sistema de detecção MIMO tem parte do processamento operando segundo a taxa de atualização da matriz canal, enquanto outra parte segue a taxa de transmissão de símbolos. O sistema pode então ser dividido nessas duas partes. A primeira delas é denominada de **pré-processamento** e tem como entrada a matriz do canal e, opcionalmente, a variância do ruído do canal. A segunda é chamada de **detector**, e tem como entradas os símbolos recebidos e a saída do pré-processador. A Figura 2.7 mostra essa divisão. Assim, um sistema MIMO possui dois parâmetros de *throughput*: a taxa de processamento de canal e a taxa de processamento de símbolos, e o *hardware* associado a cada uma dessas partes pode ser trabalhado individualmente.

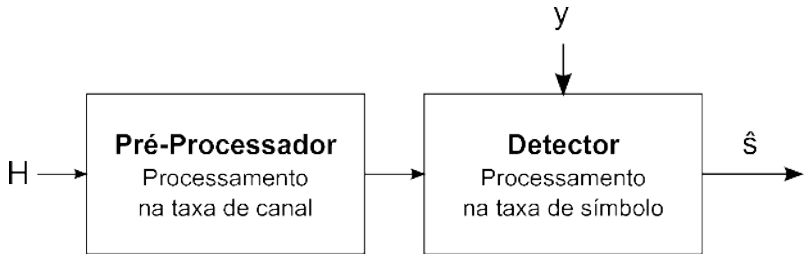

Figura 2.7: Diagrama de blocos de um sistema MIMO em alto nível.

2.3.2 Desempenho

Nesta tese o desempenho dos diversos algoritmos de detecção MIMO foram medido através de simulações feitas em computador usando o modelo descrito na Seção 2.1 para o modulador, canal de comunicação e codificador/decodificador. A métrica de desempenho utilizada foi o *bit-error-rate* (BER), que é a taxa de erro no processo de estimação dos bits transmitidos. As diferentes simulações apresentadas nesta tese diferem apenas no algoritmo do detector ou/e nos parâmetros do canal de comunicação: número de antenas transmissoras e número de antenas receptoras. A linguagem MatLab foi utilizada para descrição do modelo do sistema MIMO. Todas as funções que compõe o modelo e o ambiente de simulação foram frutos desse trabalho, com exceção das funções que realizam a codificação e a decodificação. Essas foram importadas da biblioteca Iterative Solutions Coded Modulation Library, disponível na internet sob licença GPL (*GNU General Public License*).

Nos casos em que o algoritmo de detecção simulado é de decisão abrupta e deseja-se obter a taxa de erro na saída do detector, foi utilizado a curva BER em função do SNR para representar o desempenho do sistema. Nesse tipo de simulação o codificador e o decodificador são excluídos do modelo para acelerar a simulação, já que a ausência desses não afeta o resultado. Já no casos em que se deseja medir o desempenho do conjunto detector-decodificador, foi utilizado a curva do BER em função de E_b/N_0, energia por bit de informação sobre energia do ruído. O BER nesse caso é a razão entre número de bits de informação decodificados corretamente sobre o número total de bits de informação transmitidos. Segundo o que foi definido na Seção 2.1 a energia (média) por símbolo transmitido para cada antena é E_s/N_t. Como os coeficientes da matriz canal são independentes e com variância 1, a energia média recebida por cada antena receptora é E_s. Assim a energia recebida por todas as antenas juntas é $N_r \cdot E_s$. O número de bits transmitidos é $M\,N_t$, sendo que no caso de um sistema com codificação deve-se fazer $R\,M\,N_t$ para se obter o número de bits de informação. Com isso tem-se que $E_b = \frac{N_r}{R\,M\,N_t} E_s$. Expressando E_b/N_0 em termo de logaritmo tem-se

$$\frac{E_b}{N_{0\,dB}} = \frac{E_s}{N_{0\,dB}} + 10\log_{10}\frac{N_r}{R\,M\,N_t}. \tag{2.13}$$

2.3.3 Complexidade e Área

A área de um ASIC é um parâmetro de grande importância porque determina o custo de fabricação do chip. Este parâmetro está ligado a complexidade da solução,

visto que uma área maior significa um maior número de portas lógicas e conexões. Infelizmente a área só é obtida num estágio avançando do projeto e percorrer todas as etapas de desenvolvimento é uma tarefa que demanda muito tempo. Para se obter a área deve-se passar obrigatoriamente pelas seguintes etapas do projeto: desenvolvimento do algoritmo, escolha da arquitetura, codificação da arquitetura usando uma linguagem de descrição de hardware, conversão do código para portas lógicas (síntese lógica) e posicionamento e roteamento das portas lógicas (síntese física), sendo essas últimas duas etapas feitas com auxilio de ferramentas EDA (*Electronic Design Automation*). Felizmente, outras métricas podem ser usadas para medir a complexidade em estágios anteriores do projeto, servindo como indicativo da área final que será alcançada. Assim é possível trabalhar na redução da complexidade em uma etapa antes de se avançar para a etapa seguinte. Além disso é possível comparar soluções distintas sem ter que percorrer todos as etapas do projeto. Na próxima subseção serão descritas métricas de complexidade que se aplicam a diferentes etapas de desenvolvimento de um ASIC, e que são comumente usadas na literatura. Naturalmente, as métricas das etapas mais próximas à síntese física possuem uma maior correlação com a área final. Isto também significa que o espaço para simplificações decresce a medida que se avança de etapa.

2.3.4 Descrição das Métricas de Complexidade

Na etapa de desenvolvimento do algoritmo, o número de operações aritméticas a serem computadas serve como medida de complexidade. Muitas vezes essa métrica é escrita em função dos parâmetros do sistema MIMO como o número de antenas e dos parâmetros do algoritmo que afetam o desempenho. O que facilita a comparação com outras soluções que tenham seus resultados apresentados apenas para uma determinada configuração.

No projeto da arquitetura, o número de componentes é usado como métrica já que é possível contar o número de operadores (somadores, multiplicadores, buffer etc) a serem usados na implementação, bem como o tamanho das memórias em número de palavras. Para comparar arquiteturas com diferente relação entre o número de operadores, um peso pode ser atribuído a cada tipo de operador com base numa estimativa da relação entre suas áreas.

A etapa de síntese lógica fornece o circuito de portas lógicas e as memórias que serão usadas para implementar o ASIC, enquanto a etapa de síntese física cuida do posicionamento físico do circuito, o que deve ser feito obedecendo um série de

restrições impostas pela tecnologia usada. Ao fim da etapa de síntese lógica tem-se como informação sobre complexidade, a lista das portas lógicas usadas juntamente com a área ocupada por elas, e a área das memórias. Nesta etapa costumasse usar o número de porta como métrica de complexidade para comparar diferentes arquiteturas. O número de portas lógicas é obtido dividindo a área total, portas lógicas mais memórias, pela área da menor porta NAND de duas entradas disponível na biblioteca da tecnologia. Este parâmetro é comumente usado na literatura por permitir uma comparação mais justa entre arquiteturas sintetizadas com tecnologias diferentes.

Por fim, tem-se a etapa de síntese física que provê a área que o sistema terá quando fabricado. Além da área, outras métricas são obtidas com maior exatidão após a síntese física como a potência do circuito e a frequência máxima de operação.

2.4 Algoritmos de Detecção com Decisão Abrupta

O critério ideal para estimar o símbolo transmitido no caso de detectores com decisão abrupta consiste em escolher o símbolo mais provável. Esse critério é conhecido como máxima verosimilhança (*Maximum Likelihood - ML*). No caso de sistemas com ruído gaussiano, a solução ML é dada por

$$\hat{\boldsymbol{s}} = \underset{\boldsymbol{s}\in\Omega^N}{\arg\min}(\|\boldsymbol{r} - \boldsymbol{H}\cdot\boldsymbol{s}\|^2), \tag{2.14}$$

em que $\hat{\boldsymbol{s}}$ é o vetor de símbolo estimado, e Ω^N é o conjunto dos possíveis vetores de símbolos enviados. A equação pode ser interpretada como uma busca para se achar o $\boldsymbol{s}$ entre todos os possíveis que resulte na menor Distância Euclidiana quadrática (*Euclidean Distance* - ED) $\|\boldsymbol{r} - \boldsymbol{H}\cdot\boldsymbol{s}\|^2$.

Os diversos algoritmos para detecção MIMO de decisão abrupta podem ser classificados como ML, quase-ML ou não-ML, segundo o seu desempenho de detecção. Outra qualificação possível é quanto ao método utilizado para o cálculo. Neste quesito pode-se estabelecer os seguintes métodos clássicos:

- Linear (*Zero Forcing* e *Minimum Mean Square Error - MMSE*)
- SIC (*Successive Interference Cancellation*)
- Busca exaustiva
- Busca em árvore (*search-tree*)

2.4.1 Métodos Lineares

Os métodos lineares possuem baixo BER, em compensação possuem também baixa complexidade computacional [12]. Nos métodos lineares, uma estimação de $\boldsymbol{s}$ é formada usando uma matriz de equalização $\boldsymbol{G}_{N_t \times N_r}$ gerada a partir de $\boldsymbol{H}$. A primeira etapa para estimação é calcular $\boldsymbol{y} = \boldsymbol{G} \cdot \boldsymbol{r}$. Em seguida, quantiza-se os elementos de $\boldsymbol{y}$ segundo a constelação usada, $\hat{\boldsymbol{s}} = Q(\boldsymbol{y})$, onde $Q()$ é a função de quantização e $\hat{\boldsymbol{s}}$ a estimativa do vetor transmitido $\boldsymbol{s}$. O processo de quantização consiste em arredondar cada símbolo do vetor $\boldsymbol{y}$ para o valor mais próximo da constelação Ω. Este método não leva em consideração que cada elemento de $\boldsymbol{s}$ interfere em todas as linhas de $\boldsymbol{r}$ e que por isso há uma correlação entre seus valores. É exatamente por não levarem em conta a correlação entre os símbolos que os métodos lineares possuem baixo BER [12].

No caso do *Zero Forcing (ZF)* a matriz $\boldsymbol{G}$ é dada pela pseudo-inversa de Moore-Penrose de $\boldsymbol{H}$, representada por $\boldsymbol{H}^+$. Para o caso em que $N_r \geq N_t$, a pseudo-inversa é dada por

$$\boldsymbol{G}_{\mathrm{ZF}} = \boldsymbol{H}^+ = (\boldsymbol{H}^* \boldsymbol{H})^{-1} \boldsymbol{H}^*, \tag{2.15}$$

em que $\boldsymbol{H}^*$ é a matriz transposta conjugada de $\boldsymbol{H}$. No caso específico de uma matriz quadrada $\boldsymbol{H}^+ = \boldsymbol{H}^{-1}$.

Aplicando a transformação $\boldsymbol{G}_{\mathrm{ZF}}$ nos dois lados da Equação (2.6), obtém-se

$$\boldsymbol{y}_{\mathrm{ZF}} = \boldsymbol{s} + \boldsymbol{v}_{\mathrm{ZF}}, \tag{2.16}$$

onde $\boldsymbol{y}_{\mathrm{ZF}} = G_{\mathrm{ZF}} \cdot \boldsymbol{r}$ e $\boldsymbol{v}_{\mathrm{ZF}} = G_{\mathrm{ZF}} \cdot n$. A matriz de equalização G converte a matriz canal em uma matriz identidade. Com isso a interferência entre os sinais paralelos é eliminada completamente. No entanto, esta perfeita separação entre as componentes vem com o custo de um incremento do ruído aditivo, uma vez que $\|\boldsymbol{v}_{\mathrm{ZF}}\|$ é normalmente maior que $\|\boldsymbol{n}\|$.

Ao invés de forçar os termos relativos à interferência para zero, independentemente do incremento que isso irá causar no ruído, a detecção MMSE busca minimizar a expectativa do erro total $E\{\|\boldsymbol{G}\,\boldsymbol{r} - s\|^2\}$ usando o conhecimento sobre o ruído. A partir da teoria da estimação pode ser mostrado que a combinação ótima entre cancelamento de interferência e amplificação do ruído é alcançada fazendo [20]

$$\boldsymbol{G}_{\mathrm{MMSE}} = (\boldsymbol{H}^* \boldsymbol{H} + N_t \sigma^2 I)^{-1} \boldsymbol{H}^*. \tag{2.17}$$

Aplicando (2.17) aos dois lados da equação (2.6), tem-se

$$\boldsymbol{y}_{\mathrm{MMSE}} = \tilde{\boldsymbol{H}} s + \boldsymbol{v}_{\mathrm{MMSE}}, \tag{2.18}$$

em que $\tilde{\boldsymbol{H}} = \boldsymbol{G}_{\mathrm{MMSE}} \cdot \boldsymbol{H}$ é a matriz canal efetiva após a equalização MMSE. Ao contrário do caso do ZF, os elementos fora da diagonal principal de $\tilde{\boldsymbol{H}}$ são não nulos, o que corresponde à interferência residual esperada. Como pode ser visto, o MMSE requer que σ^2 seja medido, o que não é necessário no caso do ZF.

A Figura 2.8 mostra o desempenho em termo de BER dos métodos lineares e dos demais algoritmos de detecção rígida que serão descritos nesta seção para o caso de um sistema 4x4 16QAM.

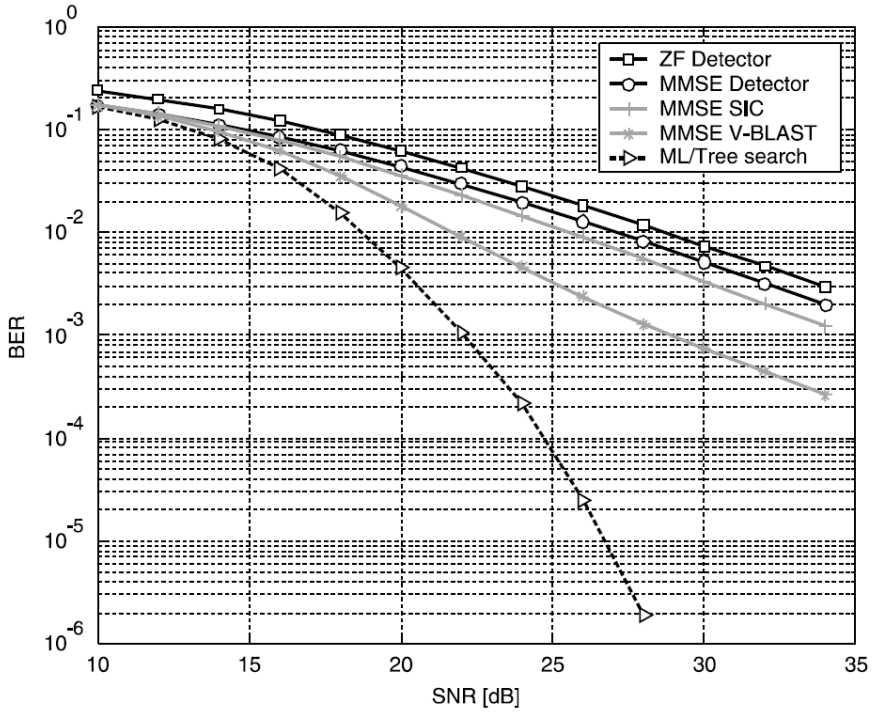

Figura 2.8: Comparação da BER dos algoritmos lineares ZF e MMSE e dos algoritmos não lineares SIC, V-BLAST e ML [12]

2.4.2 Cancelamento Sequencial de Interferências

O Cancelamento Sequencial de Interferências realiza um processo iterativo onde, a cada iteração, um único símbolo de $\boldsymbol{s}$ é estimado. A interferência deste é então removida e passa-se a estimação do próximo símbolo. Se o símbolo for corretamente

estimado, sua interferência será perfeitamente cancelada nas estimações seguintes. No entanto, um erro na estimação do primeiro símbolo vai se refletir em todas as demais estimações. Por esse motivo, o desempenho do SIC depende muito da primeira estimação. Para minimizar a probabilidade de erro, a detecção deve seguir a ordem dos símbolos com melhor probabilidade de serem decodificados corretamente. Uma das possíveis métrica para medir essa confiabilidade na detecção é a norma das linhas da matriz de equalização $\boldsymbol{G}$. Esta métrica pode ser entendida mais facilmente tomando-se, por exemplo, o caso em que ZF é utilizado para equalização. Na Equação (2.16) a relação sinal ruído entre o símbolo s_j e o ruído modificado v_j é dado por

$$\frac{E\|s_j\|}{E\|v_j\|} = \frac{E\|s_j\|}{E\|(\boldsymbol{G}_{\mathrm{ZF}})_j \cdot n\|}, \tag{2.19}$$

em que $(\boldsymbol{G})_j$ é a linha j da matriz de equalização. Pela Equação (2.19) fica claro que quanto menor for a norma de $(\boldsymbol{G})_j$ melhor será o SNR pós-equalização para o símbolo s_j. Portanto, segundo essa métrica, $\arg\min_j \|(\boldsymbol{G}_1)_j\|^2$ determina o primeiro símbolo a ser decodificado.

O ordenamento da detecção pode ser feito uma única vez no início do processo ou após cada iteração. Nesse segundo caso, a menor norma entre as linhas de $\boldsymbol{G}$ determina o primeiro símbolo a ser detectado, e a cada iteração, deve-se calcular a menor norma entre as linhas restantes para determinar o próximo símbolo a ser detectado. Sendo que esse cálculo é feito após eliminar a interferência do símbolo já estimado. Por isso, essa varição do SIC, denominada de V-BLAST (*Vertical-Bell Laboratories Layered Space-Time*) [21], possui um desempenho superior ao *Ordered*-SIC (OSIC) que corresponde ao primeiro caso. O Algoritmo 2.1 descreve o funcionamento do V-BLAST.

Algoritmo 2.1 V-BLAST

$i \leftarrow 1$
$\boldsymbol{G}_1 = \boldsymbol{H}^+$
$k_1 = \arg\min_j \|(\boldsymbol{G}_1)_j\|^2$
for $i = 1$ **to** $i = N_t$ **do**
  $y_{k_i} = (\boldsymbol{G}_i)_{k_i}\, r$
  $\hat{s}_{k_i} = Q(y_{k_i})$
  $\boldsymbol{r} = \boldsymbol{r} - \hat{s}_{k_i}(\boldsymbol{H}^t)_{k_i}$
  $\boldsymbol{G}_{i+1} = \boldsymbol{H}^+_{k_i}$
  $k_{i+1} = \arg\min_{j \notin (k_1 \cdots k_i)} \|(\boldsymbol{G}_{i+1})_j\|^2$
end for

Pode-se então subdividir o algoritmo SIC nas seguintes categorias: sem ordenamento, com um único ordenamento, ou com ordenamento a cada detecção (V-BLAST); outros parâmetros são o método de equalização e a métrica para determinar a ordem de detecção. Como pode ser visto na Figura 2.8, o SIC apresenta um BER menor do que os métodos lineares para um mesmo SNR. O motivo desse desempenho superior esta no fato da detecção dos símbolos seguintes levarem em consideração os símbolos já detectados.

2.4.3 Busca Exaustiva

O algoritmo da busca exaustiva obtém a solução da Equação (2.14) calculando o erro quadrático $\|\boldsymbol{r} - \boldsymbol{H} \cdot \boldsymbol{s}\|^2$ de todas as possibilidades para $\boldsymbol{s}$ e fazendo a busca pelo menor erro. Apesar de obter um desempenho ótimo para um detector de decisão rígida (ML), a complexidade desse método cresce de forma exponencial com o número de bits transmitidos por uso do canal, $|\Omega^{N_t}| = 2^{M \cdot N_t}$. Isto torna essa solução inviável para sistema com alta taxa de transmissão. Para demonstrar a severidade desse crescimento considere os sistemas 2x2 QAM64 e 4x4 QAM16, com $M\,N_t$ igual a 12 e 16 respectivamente. Seria necessário calcular o erro quadrático de 4.096 e 65.536 hipóteses, respectivamente, para cada transmissão.

2.4.4 Busca em Árvore

Os algoritmos de busca em árvore têm como objetivo obter a solução ML ou uma solução quase-ML com uma complexidade computacional pelo menos em média muito inferior ao método da busca exaustiva. Todos esses algoritmos modificam a equação (2.6) de forma a obter uma equação equivalente mas com uma matriz canal que seja triangular superior. O método mais comum para obter isso é fazendo a decomposição QR da matriz $\boldsymbol{H}$, isto é,

$$\boldsymbol{H} = [\boldsymbol{Q}_1\, \boldsymbol{Q}_2] \cdot \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix}, \tag{2.20}$$

em que $\boldsymbol{R}$ é $N_t \times N_t$ e triangular superior, $\boldsymbol{0}$ é um matriz nula de dimensão $(N_r - N_t) \times N_t$, e $\boldsymbol{Q}_1$ e $\boldsymbol{Q}_2$ são de dimensões $N_r \times N_t$ e $N_r \times (N_r - N_t)$, respectivamente, e possuem colunas ortogonais e unitárias, ou seja, com módulo um. A matriz $\boldsymbol{Q} = [\boldsymbol{Q}_1\, \boldsymbol{Q}_2]$ é unitária, o que significa que $\boldsymbol{Q}\,\boldsymbol{Q}^* = \boldsymbol{Q}^*\,\boldsymbol{Q} = \boldsymbol{I}$. Existem múltiplas soluções para a decomposição QR, contudo a solução específica em que os termos da diagonal de $\boldsymbol{R}$

são reais e positivos é preferível por criar possibilidades para simplificações conforme será visto nas propostas de algoritmos e arquiteturas que serão apresentadas. Além disso, a grande maioria das arquiteturas para decompositores QR complexos, como [22] e [23] por exemplo, calculam essa solução específica. Assim, será considerado que a matriz $\boldsymbol{R}$ possuí a forma

$$\boldsymbol{R} = \begin{bmatrix} a_{1,1} & a_{1,2} + jb_{1,2} & \dots & a_{1,N_t} + jb_{1,N_t} \\ 0 & a_{2,2} & \dots & a_{2,N_t} + jb_{2,N_t} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{N_t,N_t} \end{bmatrix} \tag{2.21}$$

Aplicando a transformação $\boldsymbol{Q}^*$ nos dois lados de (2.6), tem-se

$$\begin{aligned} \boldsymbol{Q}^*\boldsymbol{r} &= \boldsymbol{Q}^*\boldsymbol{Q}\boldsymbol{R}\boldsymbol{s} + \boldsymbol{Q}^*\boldsymbol{n} \\ \begin{bmatrix} \boldsymbol{Q}_1^* \\ \boldsymbol{Q}_2^* \end{bmatrix} \boldsymbol{r} &= \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{s} + \begin{bmatrix} \boldsymbol{Q}_1^* \\ \boldsymbol{Q}_2^* \end{bmatrix} \boldsymbol{n}. \end{aligned} \tag{2.22}$$

Eliminando-se as $N_r - N_t$ linhas inferiores de 2.22, chega-se à

$$\boldsymbol{z} = \boldsymbol{R}\boldsymbol{s} + \boldsymbol{v}, \tag{2.23}$$

sendo $\boldsymbol{z} = \boldsymbol{Q}_1^*\boldsymbol{r}$ e $\boldsymbol{v} = \boldsymbol{Q}_1^*\boldsymbol{n}$.

É interessante calcular a razão entre $\|\boldsymbol{R}\boldsymbol{s}\|^2$ e $\|\boldsymbol{v}\|^2$ para obter o SNR pós transformação. Analisando a energia do ruído verifica-se que $E[\|v_i\|^2] = \sigma^2$ pelo fato das linhas de $\boldsymbol{Q}_1^*$ serem vetores de módulo 1. Consequentemente, tem-se que $E[\|\boldsymbol{v}\|^2] = N_t\,\sigma^2$. Por $\boldsymbol{Q}^*$ ser uma matriz unitária, $\|\boldsymbol{Q}^*\boldsymbol{a}\| = \|\boldsymbol{a}\|$, sendo $\boldsymbol{a}$ um vetor qualquer. Então, sabendo que $E\|\boldsymbol{H}\boldsymbol{s}\|^2 = N_r\,E_s$, chega-se a $E\|\boldsymbol{R}\,\boldsymbol{s}\|^2 = N_r\,E_s$. Assim,

$$SNR = \frac{N_r}{N_t} \cdot \frac{E_s}{\sigma^2}, \tag{2.24}$$

o que deixa claro como a diversidade de recepção ($N_r > N_t$) se traduz em um ganho de SNR com o devido processamento do sinal no receptor.

A distância euclidiana (quadrática) de uma determinada hipótese $\boldsymbol{s}$ é então dada

por

$$d(\boldsymbol{s}) = \|\boldsymbol{z} - \boldsymbol{R}\boldsymbol{s}\|^2 \tag{2.25}$$

$$\left\| \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{N_t} \end{bmatrix} \right\|^2 = \left\| \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{N_t} \end{bmatrix} - \begin{bmatrix} a_{1,1} & a_{1,2} + jb_{1,2} & \cdots & a_{1,N_t} + jb_{1,N_t} \\ 0 & a_{2,2} & \cdots & a_{2,N_t} + jb_{2,N_t} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{N_t,N_t} \end{bmatrix} \cdot \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{N_t} \end{bmatrix} \right\|^2$$

Uma estrutura tipo árvore pode ser construída considerando um cálculo linha a linha da Equação (2.25) começando da linha N_t e indo até a primeira linha. Para facilitar a visualização considere o exemplo da árvore criada para um sistema 3x3 BPSK apresentado na Figura 2.9, onde os dois símbolos possíveis da modulação BPSK foram representados por $+1$ e -1. As folhas de tal árvore, na parte de baixo, representam todas as possibilidades para o vetor de símbolo $\boldsymbol{s}$ e os nós são vetores de símbolo parcial $\boldsymbol{s}^{(i)} = [s_i\, s_{i+1} \ldots s_{N_t}]^t$, sendo o nó inicial o vetor vazio $\boldsymbol{s}^{(N_t+1)} = [\,]$. Cada nó possui uma distância euclidiana (quadrática) parcial (*Partial Euclidian Distance* - PED) $T(\boldsymbol{s}^{(k)})$ associada. O PED para o nó inicial é $T(\boldsymbol{s}^{(N_t+1)}) = 0$, enquanto os dos demais nós é dado por

$$T_i(\boldsymbol{s}^{(i)}) = T_{i+1}(\boldsymbol{s}^{(i+1)}) + \|e_i(\boldsymbol{s}^{(i)})\|^2 \tag{2.26}$$

em que $\|e_i(\boldsymbol{s}^{(i)})\|^2$ é o incremento de distância associado a transição do nó $\boldsymbol{s}^{(i+1)}$ para o $\boldsymbol{s}^{(i)}$. Este erro incremental pode ser expresso separando a influência de s_i dos termos previamente definidos em $\boldsymbol{s}^{(i+1)}$.

$$\|e_i(\boldsymbol{s}^{(i)})\|^2 = \|u_i(\boldsymbol{s}^{(i+1)}) - a_{i,i}s_i\|^2 \tag{2.27}$$

$$u_i(\boldsymbol{s}^{(i+1)}) = z_i - \sum_{j=i+1}^{N_t} (a_{i,j} + jb_{i,j}) \cdot s_j \tag{2.28}$$

Assim, caso se deseje calcular o erro parcial para os outros nós filhos de $\boldsymbol{s}^{(i+1)}$ pode-se reaproveitar o cálculo do termo $u_i(\boldsymbol{s}^{(i+1)})$.

Como o incremento $\|e_i(\boldsymbol{s}^{(i)})\|^2$ é sempre não negativo, então tem-se que

$$T_i(s^{(i)}) \geq T_{i+1}(s^{(i+1)})$$

Esta característica é que permite o podamento da árvore pelos algoritmos de busca. Uma das estratégias de podamento é a esfera de detecção (*sphere-detector*) que es-

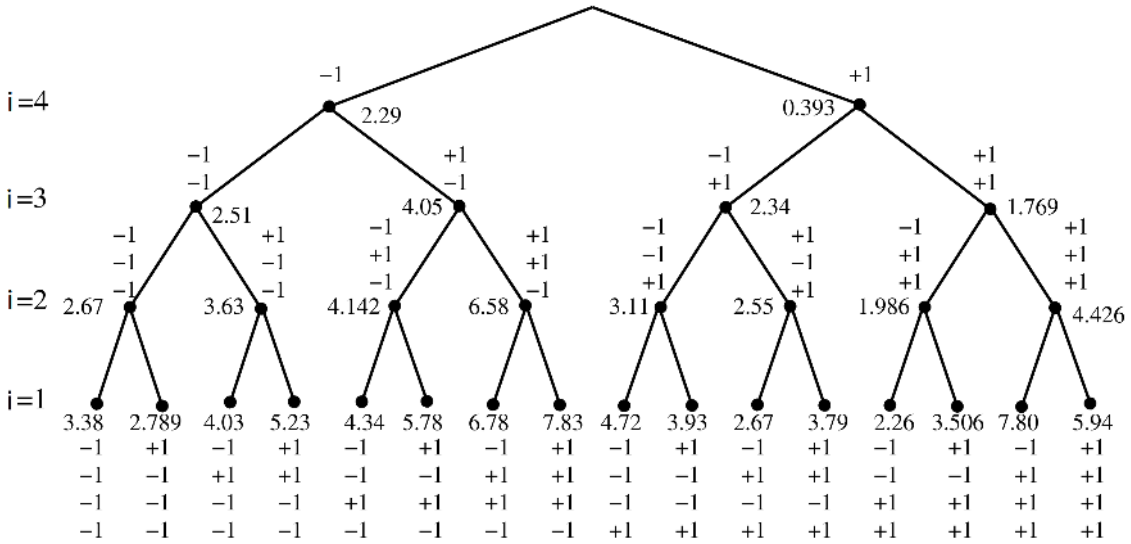

Figura 2.9: Árvore de busca para um sistema MIMO 3x3 com modulação BPSK.

tabelece uma distância quadrática máxima d^2. Assim, se o PED de um nó for maior que d^2, $T_i(\boldsymbol{s}^{(i)}) > d^2$, este nó é descartado, assim como todos os que estão abaixo dele. Se o valor escolhido para d^2 for adequado, consegue-se uma grande redução no número de folhas (hipótese) que precisam ter seu $d(s)$ calculado para se chegar no ML. Entretanto, um valor muito pequeno para d^2 pode cortar todas as soluções. A forma com que a árvore é percorrida pode ser classificada como profundidade primeiro (*Depth First*) ou largura primeiro (*Breadth First*). No profundidade primeiro a árvore é percorrida descendo sempre de nível a cada instante até não ser mais possível seguir descendo. Isto pode ocorrer por dois motivo: atingiu-se uma folha (nível $i = 1$), ou todos os nós filhos foram descartados por não atender ao requisito $T_i(s^{(i)}) < d^2$. Em qualquer caso sobe-se de nível até atingir um nível onde seja possível voltar a descer. A busca em espessura primeiro, como o nome sugere, consiste em varrer todos os nós de um nível antes de descer para o próximo nível, não havendo regressão a níveis superiores. Na prática, a espessura primeiro costuma ser utilizado com o critério *K-Best*, que consiste em manter no máximo os K melhores nós em cada nível. Isto porque, se o único critério de podamento da árvore fosse a esfera de detecção, a espessura primeiro poderia levar a um número de nós em aberto muito grande, o que iria exigir muita memória. O critério *K-Best* não garante que a solução ML seja encontrada, entretanto possui a vantagem de ter complexidade fixa, o que em *hardware* se traduz em uma taxa de processamento determinística. A Figura 2.10 ilustra o esquema de busca desse algoritmo.

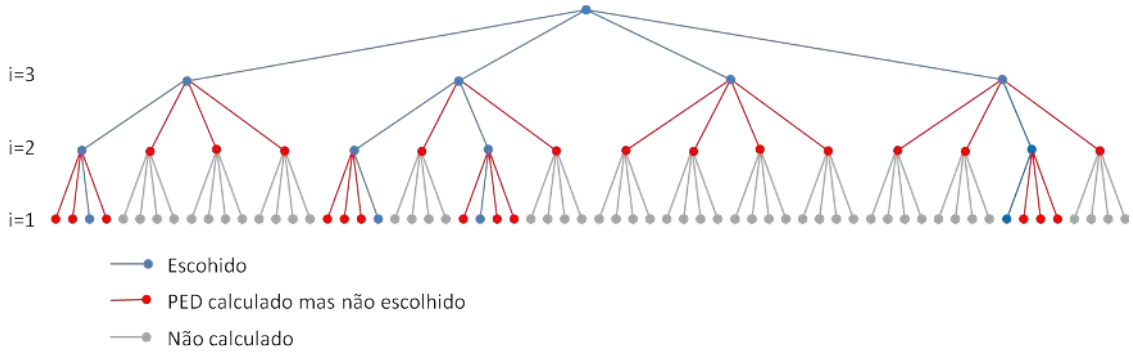

Figura 2.10: Exemplo de busca em árvore com *K-Best* para k=4 e MIMO 3x3 QPSK.

2.5 Algoritmos de Detecção com Decisão Suave

Os detectores de decisão suave calculam o LLR de um bit para informar o grau de certeza sobre o seu valor, ou uma versão aproximada e/ou quantizada desse valor. O termo bit suave serve para se referir a saída de detectores de decisão suave de forma mais geral.

O LLR de um bit x_k pertencente a um vetor $\boldsymbol{x}$ é definido como sendo

$$L(x_k|\boldsymbol{r}) = \ln \frac{p(x_k = +1|\boldsymbol{r})}{p(x_k = -1|\boldsymbol{r})} \tag{2.29}$$

em que x_k representa o k-ézimo bit do vetor $\boldsymbol{x}$, $p(x_k = +1|\boldsymbol{r})$ e $p(x_k = -1|\boldsymbol{r})$ são as probabilidades do bit x_k ser igual +1 e -1, respectivamente, dado que $\boldsymbol{r}$ foi recebido [1]. Como pode ser verificado, um valor negativo para $L(x_k|r)$ representa uma maior probabilidade do valor do bit ser -1. Um valor positivo, uma maior probabilidade de +1.

Aplicando o teorema de Bayes em (2.29) e fazendo $p(\boldsymbol{x}) = \prod_{n=1}^{N_t M} p(x_n)$ chega-se a:

$$L(x_k|\boldsymbol{r}) = \ln \frac{\sum_{\boldsymbol{x} \in X_{k,+1}} p(\boldsymbol{r}|\boldsymbol{x}) \prod_{n=1}^{N_t M} p(x_n)}{\sum_{\boldsymbol{x} \in X_{k,-1}} p(\boldsymbol{r}|\boldsymbol{x}) \prod_{n=1}^{N_t M} p(x_n)} \tag{2.30}$$

em que $X_{k,+1}$ e $X_{k,-1}$ são o conjunto de todos os $\boldsymbol{x}$ em que $x_k = +1$ e $x_k = -1$, respectivamente.

Quando não há informação a priori sobre o valor dos bits em $\boldsymbol{x}$, considera-se que $p(x_n = +1) = p(x_n = -1) = 1/2$ para $n = 1 \cdots N_t M$. Com isso, os produtórios em (2.30) podem ser eliminados, e o detector é classificado como ML. No entanto, no

[1] Ao invés do tradicional 0 e 1 para representar os valores que um bit pode assumir, serão utilizados os valores -1 e +1.

caso de detectores iterativos, o decodificador realimenta o detector com o LLR dos bits de $\boldsymbol{x}$, $L(x_n) = ln\frac{P(x_n=+1)}{P(x_n=-1)}$. A relação entre $p(x_n = 1)$ e $p(x_n = -1)$ com $L(x_n)$ é dada por:

$$p(x_n = \pm 1) = \left(\frac{e^{-L(x_n)}}{1 + e^{-L(x_n)}}\right) \cdot e^{L(x_n)\cdot x_n/2} \tag{2.31}$$

$$= A_n \cdot e^{L(x_n)\cdot x_n/2} \tag{2.32}$$

sendo $A_n = \frac{e^{-L(x_n)}}{1+e^{-L(x_n)}}$, e portanto independente do valor de x_n.

Substituindo (2.32) em (2.30) e fazendo manipulações algébricas, detalhadas em [18], chega-se a equação do detector *Maximum A Posteriori* (MAP) dado por:

$$L(x_k|\boldsymbol{r}) = L_A(x_k) + \frac{\sum_{\boldsymbol{x}\in X_{k,+1}} p(\boldsymbol{r}|\boldsymbol{x}) \cdot \exp\left(\frac{1}{2}\sum_{n=1,n\neq k}^{N_tM} L_A(x_n)\, x_n\right)}{\sum_{\boldsymbol{x}\in X_{k,-1}} p(\boldsymbol{r}|\boldsymbol{x}) \cdot \exp\left(\frac{1}{2}\sum_{n=1,n\neq k}^{N_tM} L_A(x_n)\, x_n\right)} \tag{2.33}$$

em que $L_A(x_n)$ é o LLR a priori do bit em x_n.

A função densidade de probabilidade de $p(\boldsymbol{r}|\boldsymbol{x})$ é dada por

$$p(\boldsymbol{r}|\boldsymbol{s}\to\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{\sigma^2}d(\boldsymbol{s})\right)}{(\pi\sigma^2)^{N_r}}. \tag{2.34}$$

em que $d(\boldsymbol{s}) = \|\boldsymbol{r} - \boldsymbol{H}\cdot\boldsymbol{s}\|^2$.

Devido a complexidade do cálculo de $\ln(\sum_{i=0}^{i=M-1} e^{a_i})$, resultado da substituição de (2.34) em (2.33), uma aproximação desse cálculo costuma ser usada. A aproximação $ln(e^a + e^b) \approx max(a, b)$, quando aplicado ao MAP resulta em pouca perda de desempenho [24]. Com isso, tem-se o detector max-log-MAP dado por

$$L(x_k|\boldsymbol{r}) \approx L_A(x_k) + \min_{\boldsymbol{x}\in X_{k,-1}}\left(\frac{1}{\sigma^2}d(\boldsymbol{s}) - \frac{1}{2}\sum_{n=1,n\neq k}^{N_tM} L_A(x_n)\, x_n\right) - \min_{x\in X_{k,+1}}\left(\frac{1}{\sigma^2}d(\boldsymbol{s}) - \frac{1}{2}\sum_{n=1,n\neq k}^{N_tM} L_A(x_n)\, x_n\right) \tag{2.35}$$

No caso de detector ML de decisão suave, a aproximação max-log resulta em

$$L(x_k|\boldsymbol{r}) \approx \frac{1}{\sigma^2}\left\{\min_{\boldsymbol{x}\in X_{k,-1}} d(\boldsymbol{s}) - \min_{\boldsymbol{x}\in X_{k,+1}} d(\boldsymbol{s})\right\} \tag{2.36}$$

que corresponde ao detector max-log-ML. Tal detector pode ser interpretado como

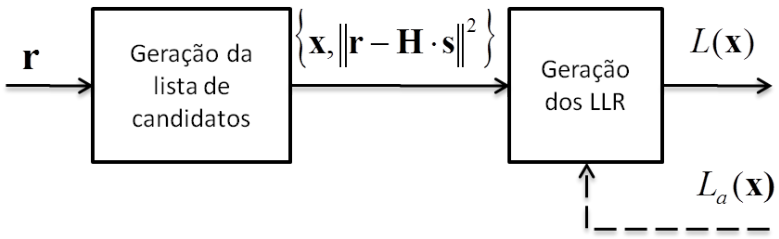

Figura 2.11: Diagrama de blocos de um detector com lista.

sendo a diferença entre o ED da melhor hipótese para $x_k = -1$ e $x_k = +1$, multiplicado por um fator de escalonamento inversamente proporcional ao ruído.

O número de ED a ser calculado pelo detector de decisão suave max-log-MAP (iterativo) e max-log-ML (não iterativo) cresce de forma exponencial com o número de bits transmitidos por uso do canal. Isto impede que sejam usados em sistemas MIMO com alta taxa de transmissão. Uma solução para limitar essa complexidade é a lista de detecção, *List Sphere Detector - LSD* [13]. A ideia do LSD é criar uma lista de $\boldsymbol{x}$ com boas métricas (baixo ED) e limitar a busca das funções min() a esse subconjunto X. O resultado do cálculo é uma aproximação do detector desejado. A lista pode ser criada usando um algoritmo de busca em árvore. Por exemplo, as hipóteses não eliminadas pelo critério do raio de busca do algoritmo *sphere-detector* ou as K hipóteses sobreviventes do algoritmo *K-Best* formariam a lista. A Figura 2.11 mostra o diagrama de blocos de um detector de decisão suave com LSD. O bloco "Gerador de Candidatos" corresponde ao algoritmo de busca em árvore que fornece a lista de hipóteses para $\boldsymbol{x}$ com seus respectivos ED, enquanto que o "Gerador dos LLR" efetua os demais cálculos. A linha tracejada na figura, indica a realimentação existente no caso do detector ser iterativo. Note que as informações do decodificador não realimentam o bloco "Gerador de Candidatos". Ou seja, a lista de candidato não é recalculada a cada interação entre detector e decodificador. Assim, o principal custo em converter um detector de decisão suave não-iterativo em um iterativo não está no processamento extra, e sim na memória para armazena a lista de candidatos de todos as transmissões que compõe o bloco de código.

2.6 Estado da Arte

O padrão para sistemas de comunicação móvel *LTE-Advanced* publicado em janeiro de 2011 possui taxa máxima de recepção para usuários estáticos superior a 1 Gbps com 8 sinais multiplexados no espaço e taxa superior a 500 Mbps com 4 si-

nais multiplexados no espaço [19]. Além do *LTE-Advanced*, o padrão IEEE 802.16m superou a taxa de 1 Gbps, também com o uso da multiplexação espacial, e o WiFi (IEEE 802.11n) superou os 600 Mbps com quatro sinais multiplexados no espaço. Apesar de taxas de transmissão superior a 500 Mbps usando MIMO 4x4 já serem contempladas por esses padrões de comunicações, o desenvolvimento de detectores que atendem esses requisitos e que tenham uma implementação economicamente viável permanece como desafio visto as soluções hoje existentes na literatura. A Tabela 2.1 apresenta alguns parâmetros de desempenho de detectores MIMO desenvolvidos recentemente. O parâmetro *taxa fixa* indica se a taxa de saída, dada em Mbps (Megabit por segundo), é fixa ou varia segundo o SNR. O parâmetro *clock* indica a frequência do clock usada para obter a dada taxa de saída e potência apresentada. No caso dos detectores que permitem múltiplas configurações no modo de transmissão MIMO, os dados apresentados são todos para configuração 4x4 64QAM.

Tabela 2.1: Resultado de Implementações de Detectores MIMO.

referencia	TxR	QAM	taxa fixa	tecnologia (nm)	gates (K)	clock (MHz)	taxa (Mbps)	potência (mW)
[25]	4x4	16	sim	350	97	200	106.6	-
[26] IC1	2x2	4-64	sim	65	408	80	240	38
[26] IC2	2x2	4-64	sim	65	135	80	164	14
[27]	2x2-8x8	4-64	não	130	350	198 (24.2dB)	431	58.2
[28]	2x2	4-64	sim	65	55	300	225	-
[29]	4x4	QAM	não	250	50	71 (20dB)	169	-
[30]	4x4	4-64	sim	45	70	500	187.5	113.9

Todos os detectores apresentados na Tabela 2.1 utilizam algoritmos de busca em árvore e todos com exceção de [29] geram saída suave usando o método LSD. O número de portas lógicas apresentado na tabela não contempla a lógica do pré-processador que realiza as decomposições QR. Em [29] tem-se uma arquitetura para um detector MIMO 4x4 16QAM de decisão abrupta com um algoritmo de busca em profundidade primeiro e esfera de busca com raio variável. O raio da esfera começa sendo infinito e decresce a cada nova folha alcançada. Em [27], tem-se uma arquitetura com número de antenas configurável entre 2x2 até 8x8, e que utiliza um algoritmo de busca em árvore, denominado melhor-primeiro, capaz de atingir o mesmo BER do profunidade primeiro com um número menor de nós visitados. No entanto, sua arquitetura é complexa por envolver diversos sinais de controle e a taxa

de saída de dados não é fixa, assim como no profundidade primeiro. Os detectores [26], [28], [25] e [30] utilizam algoritmos de busca em árvore do tipo largura primeiro com um número determinístico de folhas a serem visitadas. Isto faz com que suas taxa de processamento sejam fixas independentemente da qualidade do canal. No entanto, a potência desses é superior às soluções com complexidade variável por precisarem visitar um número maior de nós para atingir o mesmo desempenho de um algoritmo do tipo busca em profundidade ou melhor-primeiro [27].

2.7 Conclusão

Foi visto que no sistema MIMO um vetor de bits é convertido em um vetor de símbolos complexo $\boldsymbol{s}$ pertencente a um alfabeto finito. O vetor $\boldsymbol{s}$ é então transmitido com cada antena enviando um de seus elementos. O vetor complexo $\boldsymbol{r}$ captado na recepção é o resultado da transformação de $\boldsymbol{s}$ pelo canal, modelado pela matriz $\boldsymbol{H}$, e da adição de ruído. O problema da detecção MIMO foi então definido como sendo estimar o vetor de símbolo $\boldsymbol{s}$, se o detector for de decisão abrupta, ou calcular os LLR dos bits transmitidos, se o detector for de decisão suave. Para isso, utiliza-se do conhecimento de $\boldsymbol{r}$, $\boldsymbol{H}$ e σ^2 e, no caso de detectores com interação com o decodificador, dos LLR a priori dos bits que compõem o bloco de código. As principais métricas relacionadas à implementação de um detector MIMO em um dispositivo VLSI são: a taxa de processamento, a área e o desempenho do detector, analisado através da curva do BER em função do $E_b/N0$. Os detectores MIMO ideais para o caso de detecção abrupta e detecção suave sem e com iteração são o ML de decisão abrupta, o ML de decisão suave e o MAP, respectivamente. Esses detectores, especialmente os de detecção suave, são impraticáveis por possuírem alta complexidade computacional. Por isso, algoritmos de detecção com resposta não ideal são utilizados. Entre os algoritmos de decisão abrupta não ideais foram vistos: os métodos lineares *zero-forcing* e MMSE, que se destacam pela baixa complexidade; o SIC e seus derivados, que tem desempenho superior aos métodos lineares; e os algoritmos do tipo *sphere-detector*, que possuem desempenho igual ou próximo ao ML, dependendo do algoritmo de busca em árvore que implementam. Esses últimos podem ser convertidos em detectores de decisão suave usando a técnica LSD, que resulta em uma aproximação do max-log-ML ou max-log-MAP dependendo da existência ou não de interação com o decodificador. Algoritmos baseados em busca em árvore com uso do LSD tem sido bastante usados em implementações recentes de detectores MIMO em ASIC. Por se tratarem de técnicas recentes, existe hoje uma constante

evolução desses algoritmos e de suas arquiteturas ASIC. Neste contexto será apresentado no próximo capítulo um algoritmo de detecção MIMO baseado no método da busca em árvore que reduz a complexidade do max-log-ML.

Capítulo 3

Busca Exaustiva Simplificada

O detector de decisão suave ML é obtido calculando-se o ED de todos os possíveis vetores de símbolos transmitidos e, em seguida, computado-se o LLR dos bits com esses EDs. Devido à alta complexidade computacional desse detector, as soluções propostas têm mirado o max-log-ML, uma aproximação do ML. Mesmo assim, a maioria dos detectores propostos são, por sua vez, aproximações do max-log-ML. Segundo o nosso conhecimento, apenas [31] e [26] propõem algoritmos que calculam a solução exata do max-log-ML de forma otimizada, e ainda assim, o algoritmo usado em [26] só resulta no max-log-ML no caso de duas antenas transmissoras. Neste Capítulo, é proposto um algoritmo que otimiza o max-log-ML e se aplica à qualquer configuração MIMO. Comparado com o cálculo direto do max-log-ML, o algoritmo proposto, nomeado de Busca Exaustiva Simplificada (*Simplified Full-Search* - SFS), alcança 86% de redução na complexidade para constelação 64-QAM, um pouco menos que [31] que alcança em média 89%. Entretanto, o algoritmo proposto possui a vantagem de ter uma complexidade determinística, o que normalmente resulta em uma estrutura mais regular quando mapeado para uma arquitetura VLSI.

Apesar do SFS otimizar o cálculo do max-log-ML, sua complexidade ainda é alta comparada com algoritmos implementados em VLSI. Por isso, uma aproximação do SFS para a configuração 2x2 foi desenvolvida visando uma implementação.

Este capítulo é organizado da seguinte forma: na Seção 3.1, o SFS é explicado; e na Seção 3.2, a aproximação do SFS é descrita. O resultado das simulações da aproximação do SFS são analisados na Seção 3.3. Finalmente, na Seção 3.4, conclusões e considerações finais são dadas.

3.1 Busca Exaustiva Simplificada

O max-log-ML do bit x_k é dado por

$$L(x_k|\boldsymbol{r}) \approx \frac{1}{\sigma^2}\left\{\min_{\boldsymbol{s}\to\boldsymbol{x}\in X_{k,-1}} d(\boldsymbol{s}) - \min_{\boldsymbol{s}\to\boldsymbol{x}\in X_{k,+1}} d(\boldsymbol{s})\right\} \tag{3.1}$$

em que $d(\boldsymbol{s}) = \|\boldsymbol{r} - \boldsymbol{H}\cdot\boldsymbol{s}\|^2$ é o ED de $\boldsymbol{x}\to\boldsymbol{s}$, e $X_{k,-1}$ e $X_{k,+1}$ são o conjunto de todos os $\boldsymbol{x}$ em que $x_k = -1$ e $x_k = +1$, respectivamente.

O cálculo direto do max-log-ML requer que todos os 2^{N_tM} EDs sejam pré-calculados. Em seguida, para calcular o *soft-bit* de cada bit x_k, com $k = 1,\ldots,N_tM$, uma comparação entre os 2^{N_tM-1} EDs em que $x_k = +1$ é feita para achar o menor ED para condição $x_k = +1$, e a mesma quantidade de comparações para achar o menor ED para condição $x_k = -1$. Ao todo, $(N_tM)\cdot 2^{N_tM}$ comparações são necessárias para calcular o max-log-ML de todos os bits. No caso de um sistema MIMO 2x2 64-QAM, 4.096 EDs são pré-calculados e 49.152 comparações são efetuadas para calcular o LLR dos 6 bits transmitidos. Este esforço computacional torna o cálculo direto do max-log-ML inviável em sistemas de banda larga.

A complexidade do max-log-ML pode ser reduzida se um sub conjunto $B_k \in X$ contendo os elementos $\boldsymbol{x} = \arg\min_{\boldsymbol{x}\in X_{k,-1}} d(\boldsymbol{s})$ e $\boldsymbol{x} = \arg\min_{\boldsymbol{x}\in X_{k,+1}} d(\boldsymbol{s})$ puder ser determinado sem calcular todos os EDs. Isto iria permitir que o max-log-ML fosse calculado com

$$L(x_k|\boldsymbol{r}) \approx \frac{1}{\sigma^2}\left(\min_{\boldsymbol{x}\in B_{k,-1}\subset X_{k,-1}} d(\boldsymbol{s}) - \min_{\boldsymbol{x}\in B_{k,+1}\subset X_{k,+1}} d(\boldsymbol{s})\right) \tag{3.2}$$

sendo $B_{k,-1}$ e $B_{k,+1}$ subconjuntos complementares de B_k composto por vetores $\boldsymbol{x}$ em que $x_k = -1$ e $x_k = +1$, respectivamente. Neste caso, o conjunto de todos os ED pré-calculados é dado por $B = \bigcup_{k=1}^{N_tM} B_k$. Se $|B| < |X|$, uma redução no número de ED calculados é alcançada. O SFS implementa a otimização descrita acima efetuando 5 passos. **Passo 1**: o problema de detecção é convertido em um problema de busca em árvore, conforme descrito na Seção 2.4.4. **Passo 2**: todos os nós $\boldsymbol{s}^{(2)}$ são calculados. **Passo 3**: uma técnica que permite ordenar os filhos de um nó segundo a ordem crescente de seus PED sem calcula-los é aplicada nos nós $\boldsymbol{s}^{(2)}$. **Passo 4**: a informação da ordem dos nós filhos é usada para determinar os $\boldsymbol{s}$ relevantes para o cálculo do max-log-ML. As regras dessa seleção constituem a principal contribuição do SFS. Finalmente, no **Passo 5**, os $\boldsymbol{s}$ que tiveram seu ED calculado são usados para o cálculo do LLR dos bits de $\boldsymbol{x}$ segundo (3.2).

3.1.1 Ordenamento dos Nós Filhos

O SFS possui como etapa de pré-processamento a decomposição QR da matriz $\boldsymbol{H}$ e a transformação $\boldsymbol{z} = \boldsymbol{Q}_1^* \boldsymbol{r}$ que permitem converter o problema de detecção em um problema de busca em árvore, conforme descrito na seção 2.4.4. Além disso, o SFS usa uma técnica que ajuda a classificar os nós filhos segundo a ordem de seus PED, sem haver a necessidade de se calcular o valor dos PEDs.

Conforme foi visto em 2.4.4, o incremento no PED devido a transição de um nó $\boldsymbol{s}^{(k+1)}$ para um determinado nó $\boldsymbol{s}^{(k)}$ é dado por (2.27). Este incremento pode ser decomposto em duas partes: uma relativa a parte real do erro, e outra relativa a parte imaginária do erro,

$$\|e_k(\boldsymbol{s}^{(k)})\|^2 = \Re\{e_k(\boldsymbol{s}^{(k)})\}^2 + \Im\{e_k(\boldsymbol{s}^{(k)})\}^2 \tag{3.3}$$

Por motivo de simplificação, de agora em diante serão usadas as notações: $e_{r,k} = \Re\{e_k(\boldsymbol{s}^{(k)})\}$ e $e_{i,k} = \Im\{e_k(\boldsymbol{s}^{(k)}\}$

O erro real e imaginário são dados por

$$e_{r,k} = u_{r,k} - a_{k,k} \cdot s_{r,k} \tag{3.4}$$

$$e_{i,k} = u_{i,k} - a_{k,k} \cdot s_{i,k} \tag{3.5}$$

$$u_{r,k} = z_{r,k} - \sum_{j=k+1}^{N_t} (a_{k,j} \cdot s_{r,j} - b_{k,j} \cdot s_{i,j}) \tag{3.6}$$

$$u_{i,k} = z_{i,k} - \sum_{j=k+1}^{N_t} (a_{k,j} \cdot s_{i,j} + b_{k,j} \cdot s_{r,j}), \tag{3.7}$$

em que $s_{r,k} = \Re\{s_k\}$ e $s_{i,k} = \Im\{s_k\}$. Como pode ser observado, $e_{r,k}$ é independente de $s_{i,k}$ e $e_{i,k}$ é independente de $s_{r,k}$.

É possível classificar os valores para $s_{r,k}$ que completam um vetor parcial $\boldsymbol{s}^{(k+1)}$, segundo a ordem ascendente de seus respectivos erros incrementais $e_{r,k}^2$, sem precisar calcular esses valores. Para isso, é necessário apenas comparar a solução que zera o erro (ZF) em (3.4), $s_{\text{zf}} = u_{r,k}/a_{k,k}$, com algumas constantes e utilizar o resultado das comparações como um sinal para endereçar uma *lookup table* (LUT) [32]. Por exemplo, se a modulação for 64-QAM e $s_{\text{zf}} > 6$, ou seja, $u_{r,k} > 6 \cdot a_{k,k}$, então a ordem ascendente de erro incremental para $s_{r,k}$ é $[7, 5, 3, 1, -1, -3, -5, -7]$. O funcionamento dessa técnica pode ser melhor compreendido analisando-se o gráfico da função (3.4) na Figura 3.1. A Tabela 3.1 mostra como obter a classificação de

$s_{r,k}$ para o caso de 64-QAM. A mesma estratégia pode ser aplicada para o símbolo $s_{i,k}$. Uma vez que a classificação de $s_{r,k}$ é independente do valor de $s_{i,k}$, e vice-versa, ambas classificações podem ser calculadas em paralelo.

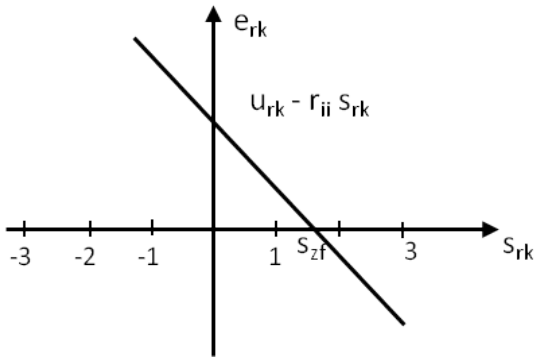

Figura 3.1: Gráfico do erro em função do símbolo.

Tabela 3.1: LUT para classificação dos nós no caso do 64QAM

If	$u \geq 0$	$u < 0$
$\lvert u\rvert > 6$	[7 5 3 1 -1 -3 -5 -7]	[-7 -5 -3 -1 1 3 5 7]
$6\,a_{i,i} \geq \lvert u\rvert > 5\,a_{i,i}$	[5 7 3 1 -1 -3 -5 -7]	[-5 -7 -3 -1 1 3 5 7]
$5\,a_{i,i} \geq \lvert u\rvert > 4\,a_{i,i}$	[5 3 7 1 -1 -3 -5 -7]	[-5 -3 -7 -1 1 3 5 7]
$4\,a_{i,i} \geq \lvert u\rvert > 3\,a_{i,i}$	[3 5 1 7 -1 -3 -5 -7]	[-3 -5 -1 -7 1 3 5 7]
$3\,a_{i,i} \geq \lvert u\rvert > 2\,a_{i,i}$	[3 1 5 -1 7 -3 -5 -7]	[-3 -1 -5 1-7 3 5 7]
$2\,a_{i,i} \geq \lvert u\rvert > a_{i,i}$	[1 3 -1 5 -3 7 -5 -7]	[-1 -3 1 -5 3 -7 5 7]
$a_{i,i} \geq \lvert u\rvert$	[1 -1 3 -3 5 -7 7 -7]	[-1 1 -3 3 -5 5 -7 7]

3.1.2 Determinando as Hipóteses Sobreviventes

As hipóteses para $\boldsymbol{s}$ que vão sobreviver ao podamento da árvore no nível $k = 1$ são aquelas que mapeiam para um vetor $\boldsymbol{x} \in B$. Isto pode ser representado por

$$\bar{\boldsymbol{s}} \rightarrow \boldsymbol{x} \in \bigcup_{k=1}^{N_t M} B_k$$

Para explicar as regras que definem $B_1, B_2, \ldots, B_{NtM}$, é necessário primeiramente definir que bits de $\boldsymbol{x}$ mapeiam $s_{r,1}$, $s_{i,1}$ e $\boldsymbol{s}^{(2)}$. Os M bits que mapeiam para um símbolo complexo s_n podem ser divididos em bits que mapeiam a parte real e bits que mapeiam a parte imaginária do símbolo. Assim, será considerado que os bits $x_1, \ldots, x_{M/2}$ mapeiam o símbolo $s_{r,1}$ e os bits $x_{M/2+1}, \ldots, x_M$ mapeiam o símbolo $s_{i,1}$,

enquanto que os demais bits, $x_{M+1}, \ldots, x_{N_tM}$, mapeiam o vetor parcial de símbolos $\boldsymbol{s}^{(2)}$.

Qualquer conjunto de $2^{\frac{M}{2}-1}+1$ símbolos $s_{r,1}$ (mais da metade das possibilidades) sempre cobre os valores +1 e -1 para os bits x_1 a $x_{M/2}$. A mesma regra é valida para um conjunto de $2^{\frac{M}{2}-1}+1$ símbolos $s_{i,1}$ e os bits $x_{M/2+1}$ a x_M. Esta propriedade, juntamente com a equação

$$d(\boldsymbol{s}) = T_2(\boldsymbol{s}^{(2)}) + e_{r,1}^2 + e_{i,1}^2 \tag{3.8}$$

que resulta de (2.26) e (3.3), permitem que os conjuntos B_1 a B_{N_tM} sejam restritos pelas seguintes regras:

1. $B_{M+1}, B_{M+2}, \ldots, B_{N_tM}$. Apenas a combinação entre o melhor $s_{r,1}$ e o melhor $s_{i,1}$ para cada $\boldsymbol{s}^{(2)}$ está incluída. As outras hipóteses para s_1 aumentam $e_{r,1}^2+e_{i,1}^2$ e, consequentemente, não tem como resultar no melhor ED para nenhum bit relacionado a $\boldsymbol{s}^{(2)}$, ou seja, $x_{M+1}, \ldots, x_{N_tM}$.
2. $B_1, B_2, \ldots, B_{M/2}$. Apenas as combinações entre os $2^{M/2-1}+1$ melhores $s_{r,1}$ e o melhor $s_{i,1}$ para cada $\boldsymbol{s}^{(2)}$ estão incluídas. Se menos que $2^{M/2-1}+1$ possibilidades para $s_{r,1}$ fossem incluídas, alguma valor para os bits $b_1, \ldots, b_{M/2}$ poderia faltar. As outras hipóteses para $s_{r,1}$ e $s_{i,1}$ aumentam o ED e não cobrem nenhum novo valor para os bits relacionados a $s_{r,1}$.
3. $B_{M/2}, B_{M/2+1}, \ldots, B_M$. Apenas as combinações entre os $2^{M/2-1}+1$ melhores $s_{i,1}$ e o melhor $s_{r,1}$ para cada $\boldsymbol{s}^{(2)}$ estão incluídas devido à mesma lógica do item anterior.

Para deixar as regras descritas acima mais claras, considera-se o caso de uma transmissão 64-QAM, onde as possibilidades para os símbolos $s_{r,1}$ e $s_{i,1}$ que completam um determinado vetor parcial $\boldsymbol{s}^{(2)}$ foram classificadas segundo a ordem crescente de $e_{r,1}^2$ e $e_{i,1}^2$, respectivamente (Figura 3.2). As linhas entre os símbolos $s_{r,1}$ e $s_{i,1}$ mostram as combinações entre esses símbolos usadas para obter vetores $\boldsymbol{s} \in B$. Observe-se que, ao selecionar os cinco melhores $s_{r,1}$, todos as possibilidades de valores para os bits $[x_1\, x_2\, x_3]$ foram cobertas. Se apenas os quatro melhores $s_{r,1}$ fossem selecionados, não haveria nenhuma hipótese investigada com $x_2 = +1$.

O podamento da árvore feito pelo algoritmo SFS pode ser visualizado na Figura 3.3, onde uma árvore de busca de um sistema 3x3 16QAM, começando de um nó intermediário $\boldsymbol{s}^{(2)}$, é apresentada. O nível 1 da árvore foi dividido em dois níveis, um relativo a decisão sobre $s_{i,1}$, com incremento de PED dado por $e_{i,1}^2$, e outro

nível relativo a $s_{r,1}$, com incremento de PED dado por $e^2_{r,1}$. As linhas mais escuras representam um maior incremento no PED. Observe-se que a escolha de $s_{i,1}$ não afeta a ordem ascendente dos PED para os símbolos $s_{r,1}$. Os bits sublinhados nos vetores $\boldsymbol{x}$, na parte de baixo da árvore, indicam em que conjuntos B_k aquela hipótese para $\boldsymbol{x}$ está incluída. Vetores $\boldsymbol{x}$ que não tem um único bit sublinhado não pertence a nenhum B_k e, portanto, seu ED não precisa ser computado, já que não é requerido para o cálculo do Max-log-ML de nenhum bit.

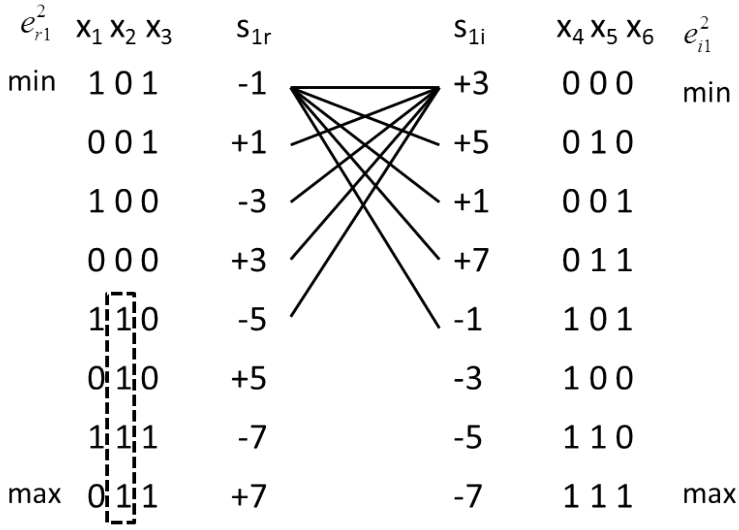

Figura 3.2: Combinações entre s_{r1} e s_{i1} investigadas pelo algoritmo SFS para um vetor parcial de símbolos $s^{(2)}$ particular com modulação 64-QAM.

De acordo com o que foi colocado, o número total de EDs necessários para calcular o max-log-ML de todos os bits é reduzido de $|X| = 2^{N_t \cdot M}$ para

$$|B| = 2^{(N_t-1)M} \cdot (2^{\frac{M}{2}} + 1)$$

. Além disso, o número de EDs comparados para obtenção da melhor hipótese para $x_k = +1$ e $x_k = -1$ é reduzido a $2^{(N_t-1)M}$ para bits relacionados ao vetor de símbolos $s_2, s_3, ..., s_{N_t}$, e a $2^{(N_t-1)M} \cdot (2^{\frac{M}{2}-1} + 1)$ para bits relacionados ao símbolo s_1. Assim, o número total de comparações é $(N_t - 1)M \cdot 2^{(N_t-1)M} + M \cdot 2^{(N_t-1)M} \cdot (2^{\frac{M}{2}} + 1)$.

O Algoritmo SFS para o caso particular do MIMO 2x2 foi codificado na linguagem Matlab. A constelações ficou configurável entre QPSK, 16-QAM e 64-QAM. O código se encontra no Apêndice A na Seção A.2.

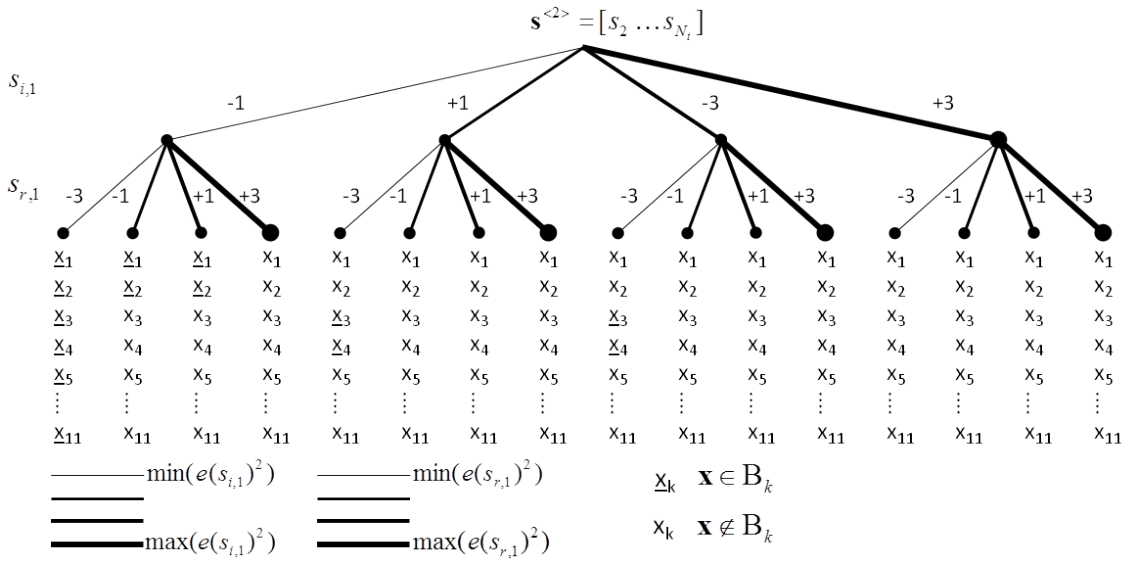

Figura 3.3: Podamento de árvore de busca feito pelo algoritmo SFS em um sistema 3x3 16QAM começando de um nó $[s_2\, s_3]$.

3.2 Aproximação do SFS para Sistemas MIMO 2x2

O SFS quando aplicado em um sistema 2x2 64-QAM calcula 576 EDs e 64 PEDs. Este valor ainda é alto se comparado com algoritmos como [26], que para mesma configuração computa apenas 128 EDs e 128 PEDs. Por isso, é proposto uma aproximação para o SFS. Essa aproximação consiste em calcular todos os $\boldsymbol{s}^{(2)}$, assim como é feito no SFS original, mas estender apenas os 16 $\boldsymbol{s}^{(2)}$ de menor PED com todas as hipóteses para s_1 contempladas pelo SFS. Os demais $\boldsymbol{s}^{(2)}$ seriam estendidos pegando-se apenas o melhor s_1 para cada um deles. Essa aproximação reduz a complexidade para 192 EDs e 64 PEDs. O algoritmo pode então ser descrito pelos seguintes passos:

1. Calcular todos os nós $\boldsymbol{s}^{(2)}$;
2. Estender todos os nós $\boldsymbol{s}^{(2)}$ com o melhor s_1 para cada nó;
3. Estender os 16 $\boldsymbol{s}^{(2)}$ de menor PED com todas as outras possibilidades para o símbolo s_1 contempladas pelo SFS;
4. Gerar os soft-bits usando os $\boldsymbol{s}$ que tiveram seus respectivo ED calculado nos passos 2 e 3.

As hipóteses para $\boldsymbol{s}$ geradas pelo passo 2 permitem a obtenção do max-log-ML para os bits relacionados ao símbolo s_2, segundo a lógica do SFS. O terceiro passo do

algoritmo melhora a aproximação do max-log-ML dos bits relacionados ao símbolo s_1. Se esse algoritmo fosse aplicado para os casos de 16-QAM e QPSK, a seleção dos 16 melhores $\boldsymbol{s}^{(2)}$ não resulta em nenhuma exclusão, e assim o algoritmo fica equivalente ao SFS, obtendo a solução max-log-ML para todos os bits.

Para reduzir a complexidade da seleção dos 16 melhores vetores parciais $\boldsymbol{s}^{(2)}$, uma otimização é feita para essa operação. A otimização consiste em selecionar os 9 nós resultantes da combinação entre os 3 melhores símbolos $s_{r,2}$ com os 3 melhores símbolos $s_{i,2}$, e os 7 nós de menor PED dentre os restantes. Esta é uma boa aproximação, visto que, se a solução de erro zero $\left(s_{2,\mathrm{zf}} = \frac{1}{a_{2,2}} \cdot (z_{r,2} + j\, z_{i,2})\right)$ não estiver fora da grade da constelação, todos os 9 nós escolhidos sem comparação de seus PED estarão de fato dentre os 16 melhores. A Figura 3.4 exemplifica esse caso. Na figura, os cruzamentos das linhas horizontais com as verticais são as hipóteses para $[s_{r,2}\ s_{i,2}]$. O 'x' indica a posição de $s_{2\,\mathrm{zf}}$. As hipóteses para os símbolos $s_{r,2}$ e $s_{i,2}$ estão numeradas segundo a classificação de seus PEDs. Os pontos pretos mostram os nós resultantes da combinação entre os 3 melhores símbolos $s_{r,1}$ com os 3 melhores símbolos $s_{i,1}$, enquanto que os pontos cinzas mostram as outras possibilidades que estão dentro do conjunto dos 16 melhores vetores parciais $\boldsymbol{s}^{(2)}$.

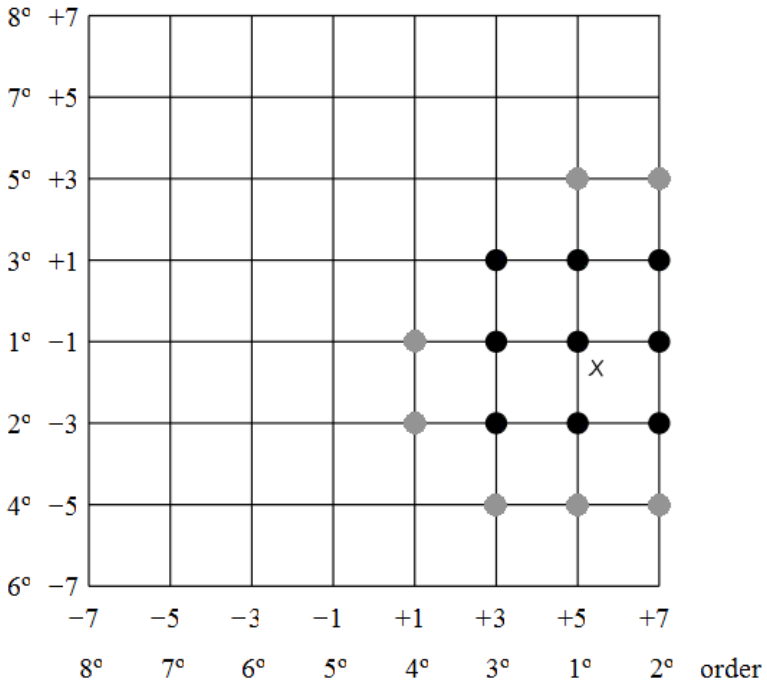

Figura 3.4: Seleção dos 16 melhores vetores parciais $s^{(2)}$ com pré-seleção de 9 nós.

O código Matlab dessa aproximação do SFS encontra-se no Apêndice na Seção A.3.

3.3 Resultado das Simulações

O desempenho do algoritmo da Seção 3.2 foi comparado com o max-log-ML para medir a perda devido a aproximação. Para isso utilizou-se a curva de BER em função da energia por bit de informação $\left(\frac{E_b}{N_0}\right)_{dB}$, formada a partir do resultado de diversas simulações do modelo MatLab do sistema. A configuração 64-QAM foi a única utilizada na comparação por já se saber que para as demais constelaçãoes o algoritmo da Seção 3.2 reproduz o max-log-ML. Um modelo em ponto flutuante e outro em ponto fixo do algoritmo foram simulados. O primeiro para analisar a perda de desempenho devido ao algoritmo apenas, e o segundo para analisar o desempenho que será atingindo pela implementação VLSI do detector, que introduz uma perda de desempenho adicional devido a maior limitação no número de bits para representar as variáveis internas do algoritmo. O algoritmo SFS 2x2 com precisão floating point foi utilizado para obtenção da curva de desempenho do max-log-ML.

No desenvolvimento do modelo em ponto fixo foi preciso determinar para cada variável do algoritmo um intervalo e uma resolução para seus valores, sendo o intervalo na forma $[-2^{k-1} \quad 2^{k-1}-1]$ para variáveis com sinal, e $[0 \quad 2^k-1]$ para variáveis sem sinal, e a resolução 2^{k-b}, com k sendo um inteiro qualquer e b o número de bits usado. O intervalo para representar os elementos de $\boldsymbol{z}$ foi deterimado adicionando-se $+2$ ao valor de k que permite representar o maior RMS (*Root Mean Square*) entre os elementos de $\boldsymbol{z}$ em condição de ruído nulo. Ao contrário do vetor $\boldsymbol{r}$, os elementos do vetor $\boldsymbol{z}$ possuem energia média diferentes, sendo o elementos z_1 o de maior RMS. A mesma estratégia foi utilizada para determinar o intervalo dos elementos de $\boldsymbol{R}$. Para reduzir o número de parâmetros envolvidos na simulação do modelo em ponto fixo, decidiu-se por utilizar o mesmo número de bits para os elementos de $\boldsymbol{R}$, $\boldsymbol{z}$, $\boldsymbol{e}$, e dos PEDs. O valor de 10 bits foi então escolhido por garantir um relação sinal ruído de discretização de aproximadamente 50dB para os elementos de $\boldsymbol{R}$.

O modelo do canal MIMO adotado nas simulações é o apresentado na Seção 2.1.3. O codificador de canal é um LDPC com $R = 1/2$, com tamanho da palavra de código sendo 2.304 bits, e o número máximo de iterações do decodificador sendo 20. As simulações foram executadas até que se atingisse um total de 100 erros de frame ou 20.000 ensaios. A Figura 3.5 apresenta o resultado das simulações na forma de uma curva de BER em função de $\frac{E_b}{N_0}$. É possível verificar na Figura 3.5 que a degradação no desempenho em relação ao max-log-ML é inferior a 0.1 dB tanto para o modelo em ponto flutuante quanto para o modelo em ponto fixo.

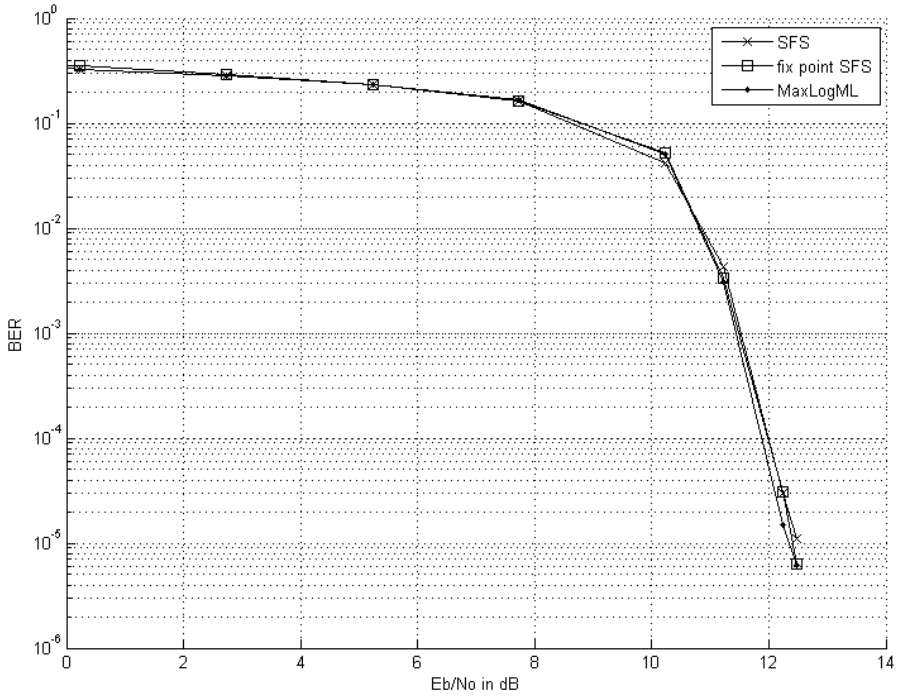

Figura 3.5: Comparação entre max-log-ML e o algoritmo proposto.

3.4 Conclusão

O SFS otimiza o cálculo do max-log-ML operando como um algoritmo de busca em árvore que realiza um podamento no último nível da busca. Pelo fato do podamento só ocorrer no último nível, o SFS não consegue uma boa redução de complexidade em sistemas MIMO com mais de duas antenas transmissoras. Uma possível solução para essa limitação é combinar o SFS com outras técnicas, tentando amenizar a degradação no desempenho que isso irá causar. Um exemplo seria utilizar o K-Melhores nos primeiros $N_t - 1$ níveis e no último nível usar o SFS. A aproximação do SFS para o caso de sistemas 2x2 mostrou ter uma complexidade menor do que outras soluções com desempenho equivalente e, apesar de não garantir a solução max-log-ML na configuração 64-QAM, como é o caso de [26], possui perda de desempenho completamente desprezível em relação ao max-log-ML.

Capítulo 4

K Melhores Espalhados

O algoritmo de detecção K-Melhores é um dos mais populares por resultar em uma estrutura muito regular quando mapeada para uma arquitetura VLSI e por ter um controle simples, se comparado com o algoritmo *depth-first*. Um problema da implementação em hardware do detector K-melhores é o caminho crítico gerado pelo hardware que classifica as K melhores hipóteses de um nível. Esse caminho crítico, proporcional a K, não pode ser reduzido com a inserção de estágios de *pipeline* quando uma arquitetura iterativa é utilizada, ou seja quando há realimentação. Nesse caso o classificador de K melhores acaba limitando a frequência máxima de operação do hardware. Neste capítulo, é proposto um algoritmo, denominado de K-Melhores Espalhados, que soluciona esse problema. Além disso, é feito um estudo sobre o desempenho do detector K-Melhores Complexo com decisão suave e não iterativo em sistemas MIMO 4x4 16-QAM com diferentes configurações para seus parâmetros. Esse estudo tem como objetivo analisar o impacto desses parâmetros no desempenho do K-Melhores e servir como referência para trabalhos futuros.

O capitulo é organizado da seguinte forma. Inicialmente o algoritmo K-Melhores é introduzido na Seção 4.1. Na Seção 4.2, um novo método para classificar os nós filhos no caso do K-Melhores Complexo é apresentado. Na Seção 4.3, o algoritmo K-Melhores Espalhado é descrito e comparado com o K-Melhores. Finalmente, na Seção 4.4, algumas considerações e conclusões sobre esse estudo do K-Melhores são dadas.

4.1 Algoritmo K-Melhores

O algoritmo de detecção K-Melhores, Algoritmo 4.1, possui basicamente dois parâmetros. O parâmetro K, que define o número máximo de nós mantidos em um nível da árvore quando se procede para o nível seguinte, e o número de nós filhos

admitidos, aqui referido pela variável A. O parâmetro A é utilizado para limitar

Algoritmo 4.1 K-Melhores

```
Calcular os filhos do nó inicial até o limite do K-ézimo melhor filho.
for n = N_t to 1 do
  for k = 1 : K do
    Calcular os A melhores filhos do k-ézimo nó do nível n
    Atualizar a lista dos K melhores nós do nível n − 1
  end for
end for
Gerar os soft-bits usando os K melhores nós do nível 1.
```

a complexidade computacional do K-Melhores. A ideia desse parâmetro é que os piores filhos de um nó têm pouca chance de ficarem entre os K melhores do seu nível, portanto podem ser previamente excluídos sem causar perda no desempenho. A Figura 4.1, mostra como o parâmetro A afeta o desempenho de um detector 4x4 16-QAM com $K = 16$. Verifica-se nessa simulação que o ganho de desempenho é insignificante para $A > 6$. A Figura 4.2 mostra o desempenho do detector MIMO 4x4 para diferentes valores de K. Como era de se esperar, o parâmetro K tem forte impacto no desempenho do detector. Neste trabalho não se investigou o desempenho para $K > 16$ por conta do tempo requerido para essas simulações.

Os K nós sobreviventes ao final do processamento do algoritmo K-Melhores formam a lista de candidatos para geração dos *softbits*. Devido à limitação no tamanho da lista de candidatos, alguns bits de $\boldsymbol{x}$ podem ficar sem nenhuma hipótese para um de seus valores, e assim, o cálculo do LLR desses bits fica comprometido, já que uma das funções $min()$ da equação do max-log-ML, (2.36), não terá nenhum ED para selecionar. A solução para esse problema é atribuir um valor limite para o LLR. Já foi verificado que a escolha do ponto de saturação do LLR afeta significativamente o desempenho do detector [33]. Entretanto, nessa tese não foi possível efetuar um estudo mais aprofundado sobre a relação desse parâmetro com o desempenho do detector. O método adotado aqui para limitar o valor do LLR foi atribuir um $d(\boldsymbol{s})$ máximo para os valores de bits não cobertos. Sendo esse $d(\boldsymbol{s})$ máximo igual a 0,25 para 64-QAM, 1 para 16-QAM e 2 para QPSK. Esses valores foram escolhidos com base no resultado de simulações e levando em consideração que $E_s = 1$.

O K-Melhores, assim como o SIC, tem seu desempenho afetado quando se modifica a ordem de detecção dos símbolos. Isto porque, um erro na detecção de um símbolo repercute na detecção dos símbolos seguintes. Duas estratégias de ordenamento são comumente usadas em detectores K-melhores [25] [34]. A primeira es-

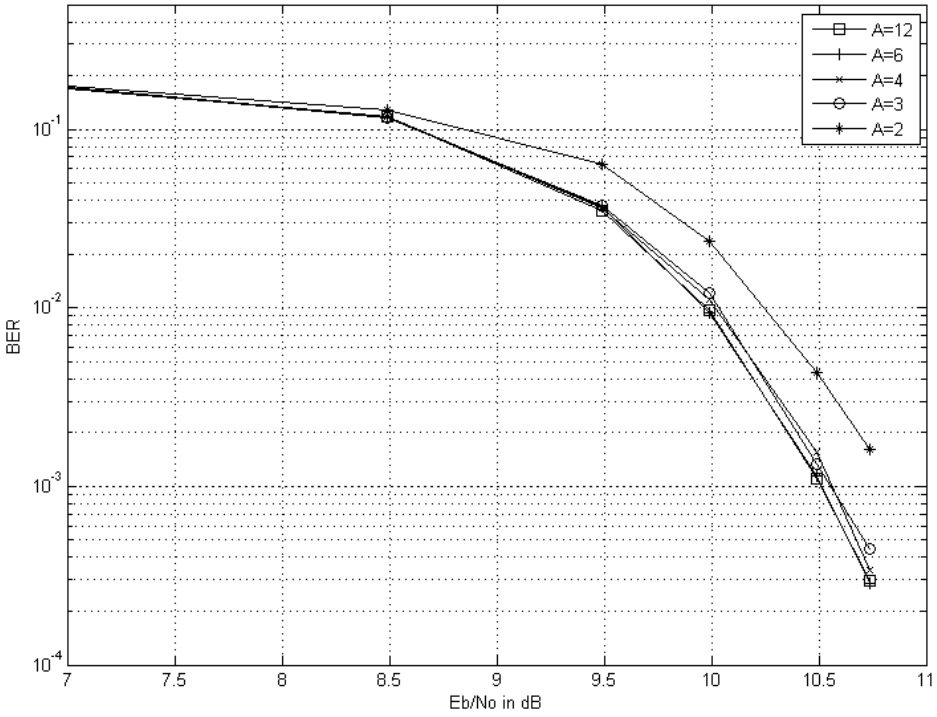

Figura 4.1: Simulação MIMO 4x4 16-QAM K=16

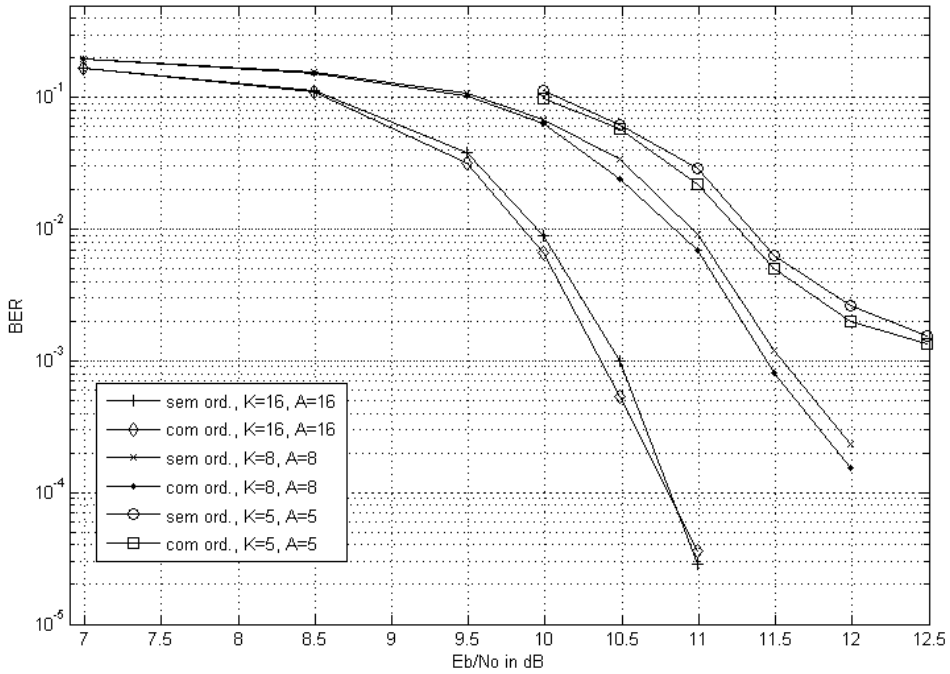

Figura 4.2: Simulação MIMO 4x4 16-QAM com K=5, K=8 e K=16, e com e sem modificação da ordem de detecção.

tratégia consiste em seguir a ordem dos símbolos com melhor SNR pós-equalização, assim com é feito no SIC. Esta estratégia é mais indicada para os casos em que $K < |\Omega|$ por haver chance da solução ML ser perdida já na primeira seleção do algoritmo. A segunda estratégia consiste em começar pelo símbolo com pior SNR pós-equalização e depois seguir a ordem do melhor SNR para os símbolos restantes. Esta estratégia é mais indicada para os casos em que $K >= |\Omega|$, visto que não existe nenhuma chance da solução ML ser perdida na primeira seleção. Para modificar a ordem de detecção, utiliza-se um decompositor QR no pré-processamento que efetua permutações entre as linhas da matriz do canal segundo a lógica de um algoritmo, como apresentado em [23] e [22]. A informação sobre as permutações efetuadas é passada ao detector para que, ao final do processo de detecção, os *softbits* possam ser reordenados corretamente. A Figura 4.2 mostra como a primeira estratégia de ordenamento afeta o desempenho de um detector K-best. Observe-se na figura que o ganho dessa estratégia de ordenamento diminui a medida que se aumenta K. Como essa estratégia demonstrou resultar em ganho de desempenho, especialmente nos casos em que o valor de K é pequeno, decidiu-se por utiliza-la nas simulações seguintes. A segunda estratégia de ordenamento não foi implementada nessa tese devido a sua maior complexidade.

4.2 Ordenamento dos Nós Filhos

Para facilitar a seleção dos K melhores, os filhos de cada nó sobrevivente devem ser ordenados segundo a ordem ascendente de seus PED. A técnica de ordenamento dos nós filhos usada no Capítulo 3 só consegue ordenar as hipóteses para componente real e para componente imaginária de um símbolo complexo. Essa técnica pode ser usada no K-Melhores se uma decomposição em valores reais (*Real-Value Decomposition*) RVD [25] for aplicada. O RVD modifica a árvore de busca de forma que cada nível passa a representar uma decisão sobre uma única componente real de um símbolo. Assim a árvore passa a ter o dobro de níveis sendo que cada nó passa a ter apenas $2^{M/2}$ nós filhos, que podem ser facilmente classificados segundo a ordem ascendente de seus PED. Entretanto, o uso do parâmetro A em uma árvore complexa resulta em uma maior redução no número de nós calculados do que numa árvore real, o que numa implementação em hardware se traduz em uma menor potência. Para atingir a mesma redução no número de nós calculados, uma árvore real teria que limitar o número de nós filhos para cada nó sobrevivente em $\sqrt{A}$, sendo que A estaria limitado a valores cuja raiz resulte em um número inteiro. Além dessa limi-

tação no valor de A, a combinação entre as $\sqrt{A}$ hipóteses para componente real com menor incremento de PED com as $\sqrt{A}$ hipóteses para componente imaginária com menor incremento de PED nem sempre leva as A hipóteses com menor PED para esse símbolo complexo. Portanto, é possível que haja uma perda de desempenho.

Segundo o conhecimento do autor não existir um método para classificar com exatidão os nós filhos de uma árvore complexa sem efetuar uma comparação entre os PEDs de diferentes nós filhos. Por isso, métodos de classificação aproximada já foram propostos para esse caso [35][27]. Nesta tese é proposto um novo método para ordenamento aproximado dos nós filhos baseado no método apresentado em [27]. O método proposto é composto dos seguintes passos. Primeiro deve-se achar o ponto da constelação mais próximo à solução que zera o erro incremental referente a passagem de um nó pai $\boldsymbol{s}^{(i+1)}$ no nível $i+1$ para o nível i, (2.27)

$$q_i = Q(s_{zf})$$

sendo $Q(.)$ a função que arredonda o valor de $(.)$ para o ponto mais próximo na constelação Ω e $s_{zf} = \frac{u_i}{a_{i,i}}$ o valor para s_i que zera o erro $e_i(s^{(i)})$. A região em torno desse ponto é dividia em 16 regiões triangulares, ver Figura 4.3. Para cada uma dessas regiões, a ordem ascendente de PED mais provável para os símbolos da constelação é previamente calculada e armazenada em uma tabela. Como existem ao todo $|\Omega|$ pontos na constelação, seriam necessários $16\,|\Omega|$ endereços na tabela. Para reduzir o requisito de memória, utiliza-se da simetria da constelação para que apenas o primeiro quadrante seja armazenado em uma memória e os demais sejam obtidos a partir desse. Assim, apenas $4\,|\Omega|$ posições de memória são necessárias.

Para determinar em qual das 16 regiões a solução que zera o erro se encontra, quatro comparações são necessárias: $w_r > 0$, $w_i > 0$, $|w_r| > |w_i|$ e $|w_r| + |w_i| > 1$. Sendo $w = s_{zf} - q_i$, e $w_r = \Re\{w\}$ $w_i = \Im\{w\}$. Esse método de ordenamento dos nós filhos sempre acerta a ordem dos três primeiros nós, mas possui uma certa probabilidade de erro nos nós seguintes. Um erro na classificação dos nós filhos pode gerar um erro no classificador dos K melhores porque esse pressupõe que o vetor de PED fornecido está corretamente ordenado. A Figura 4.4 apresenta o resultado de simulações de um detector MIMO 4x4 16-QAM K=16 A=6 com ordenamento aproximado e ordenamento exato dos nós filhos. A partir do resultado dessa simulação pode-se concluir que a degradação no desempenho devido ao uso do ordenamento aproximado é desprezível. No Apêndice A.6 o código MatLab que gera a LUT usada no ordenamento dos nós filhos é exposto.

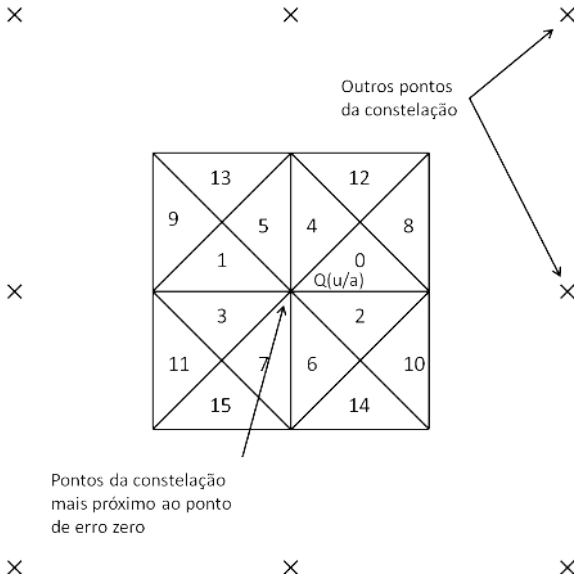

Figura 4.3: Divisão em 16 áreas da região em torno do ponto $Q(\frac{u_i}{a_{i,i}})$.

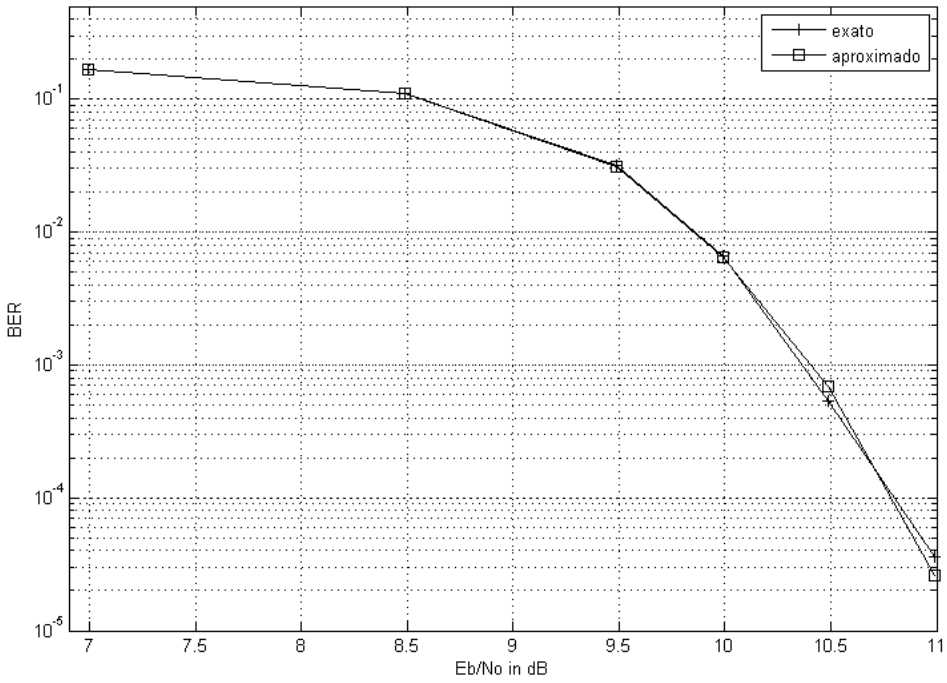

Figura 4.4: MIMO 4x4 16-QAM K=16 A=6 com classificação exata e aproximada dos nós filhos.

4.3 K-Melhores vs K-Melhores Espalhados

O processo de seleção dos K melhores costuma utilizar um algoritmo iterativo que recebe dois vetores de PEDs em ordem ascendente de valores, e gera um vetor

na saída com os melhores PEDs também ordenados. O primeiro vetor de entrada corresponde aos filhos de um determinado nó que sobreviveu ao processo de seleção no estágio anterior. Portanto, trata-se de um vetor de candidatos ao K melhores do estágio atual. O segundo vetor de entrada corresponde aos atuais K melhores do estágio atual, e o vetor de saída corresponde aos K melhores atualizados. Após receber sequencialmente todos os vetores de candidatos, o vetor de saída corresponderá aos K melhores do estágio atual. A arquitetura VLSI desse classificador iterativo de K melhores é mostrada na Figura 4.5, extraída de [12], para o caso de $A = 4$ e $K = 5$. Observe-se que a arquitetura possui um caminho crítico longo que não pode ser quebrado com um estágio de pipeline sem quebrar a lógica da seleção. Por conta disso, o classificador de K melhores é o bloco que limita a frequência máxima do clock em uma implementação VLSI do detector. Além disso, o caminho crítico aumenta de forma proporcional a K, o que impede que valores grandes para K sejam usados.

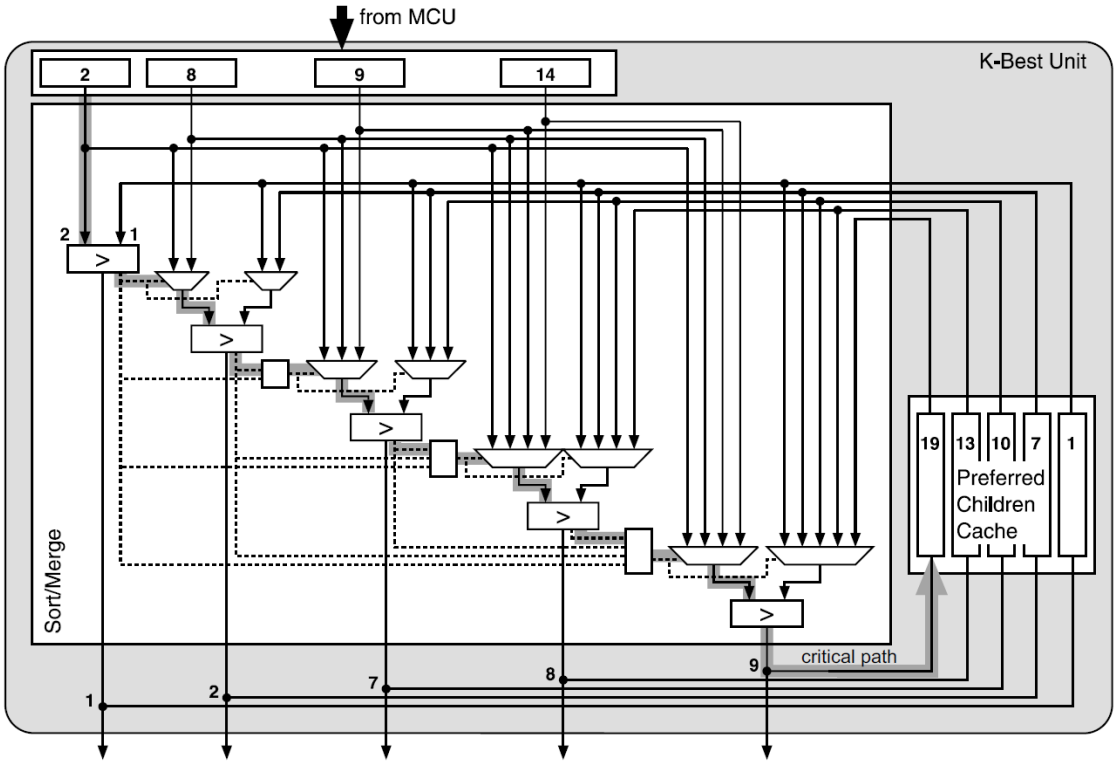

Figura 4.5: Arquitetura do classificador de K melhores iterativos

Nesta tese propõe-se uma modificação no algoritmo K-melhores para solucionar o problema do caminho crítico. A proposta consiste em ter múltiplos grupos de K melhores com valor de K pequeno, ao invés de um único grupo com valor de K alto.

O número de grupos vai ser representado pelo parâmetro L. A Figura 4.6 exemplifica esse algoritmo para o caso de um detector 3x3 QPSK L=2, K=3, A=3. Os nós filhos na Figura 4.6 estão organizados segundo a ordem ascendente de seus PED, sendo o nó filho mais a esquerda o melhor. Os conjuntos de nós filho circulados com uma linha continua competem entre si por uma posição no primeiro classificador de 3-melhores, e os circulados pela linha tracejada competem por um posição no segundo classificador de 3-melhores. A partir do segundo nível, o pior filho não é sequer calculado devido a $A = 3$. A Figura 4.7 mostra o resultado da simulação de um detector MIMO 4x4 16-QAM para diferentes configurações de L e K. Observe-se que a configuração $L = 1$ $K = 16$ e $L = 2$ $K = 8$ possuem desempenhos muito próximos, sendo que o $L = 2$ $K = 8$ tem a vantagem de possuir um classificador menos complexo.

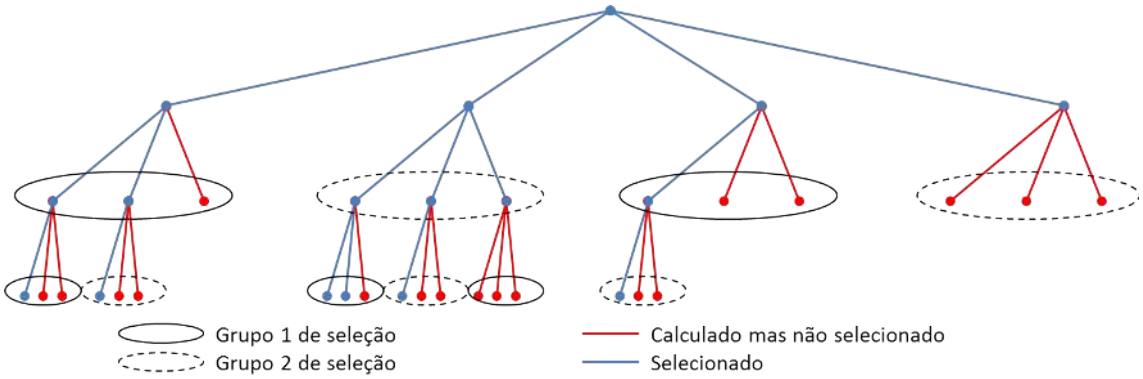

Figura 4.6: Algoritmo K-Melhores Espalhados com L=2, K=3, A=3 em uma sistema MIMO 3x3 QPSK.

O código MatLab que implementa o algoritmo K-Melhores Espalhados se encontra na Seção A.4 e o código que gera os nós filhos de um dado nó pai se encontra na Seção A.5.

4.4 Conclusão

O método de ordenamento aproximado dos nós filhos e o algoritmo K-Melhores Espalhados demonstraram não causar grande impacto no BER e podem servir para melhorar a arquitetura VLSI do detector. Entretanto, para verificar a real vantagem do detector K-Melhores Espalhados, é preciso que a arquitetura do mesmo seja desenvolvida e implementada, além de comparada com as melhores arquiteturas para o K-Melhores tradicional.

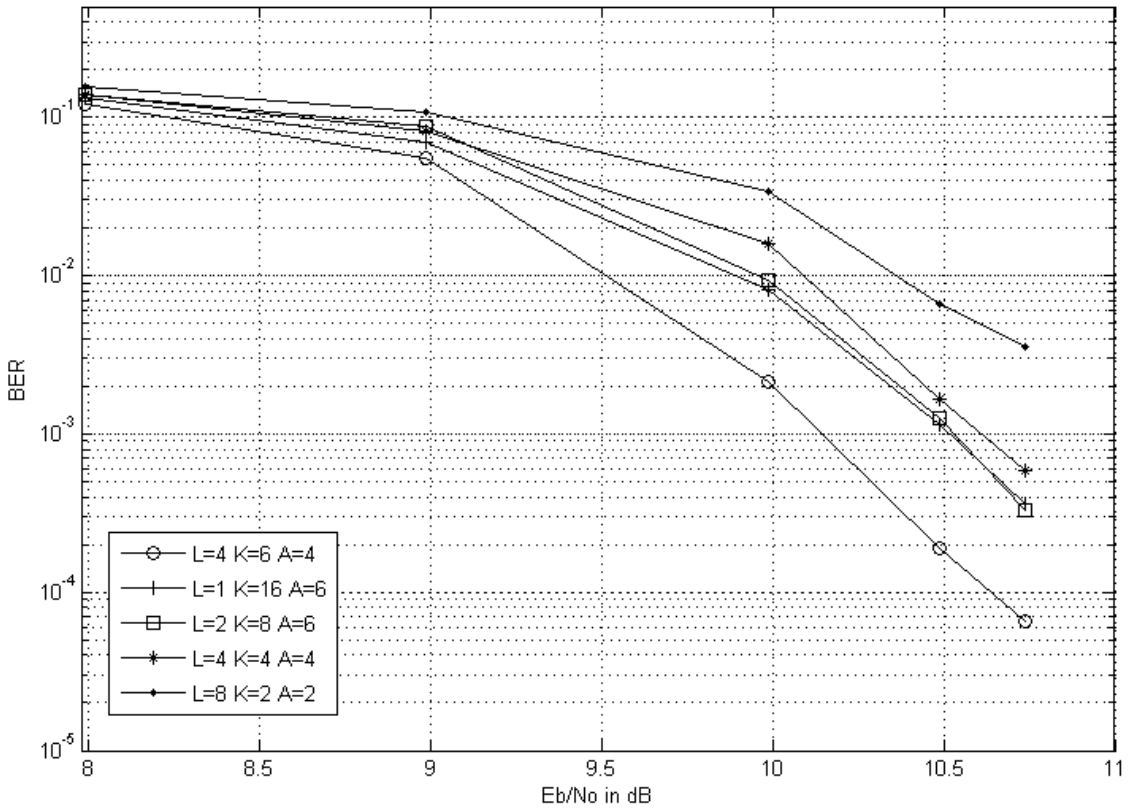

Figura 4.7: K-Melhores Espalhados 4x4 16-QAM com diferentes configurações para os parâmetros L e K.

Capítulo 5

Arquitetura para Detector SFS 2x2

Neste capítulo é apresentado o fluxo de projeto adotado para implmentação dos algoritmos desenvolvidos em ASIC. Em seguida, é desenvolvida uma arquitetura VLSI para detecção MIMO 2x2 configurável entre QPSK, 16-QAM e 64-QAM. A arquitetura implementa o algoritmo descrito na Seção 3.2, que equivale ao SFS para configuração QPSK e 16-QAM, e se limita a uma aproximação do SFS para 64-QAM. O fluxo de projeto para implementação dessa arquitetura se encontrava na etapa de síntese lógica quando essa tese foi escrita. Por isso, para analise da complexidade da arquitetura foi usado o resultado da síntese lógica, no lugar da síntese física que corresponde a etapa final do projeto.

Na Seção 5.1 o fluxo de projeto adotado para implementação em ASIC é descrito; na Seção 5.2, a arquitetura VLSI do detector 2x2 é detalhada; o resultado da síntese lógica é exposta na Seção 5.3; e conclusões são dadas na Seção 5.4.

5.1 Fluxo de Projeto de ASIC

O fluxo de projeto aqui adotado para implementação dos detectores em ASIC é constituído pelas etapas apresentadas na Figura 5.1.

A especificação do sistema é o ponto de partida do projeto. Nessa etapa devem ser definidos o algoritmo a ser implementado, a taxa de processamento e desempenho a ser atingido.

A etapa de modelagem do sistema consiste em implementar o algoritmo em uma linguagem de alto nível para servir de referência para o modelo em RTL. Nessa tese, a linguagem de alto nível escolhida foi o MatLab devido à sua vasta biblioteca de funções, que incluem a plotagem de sinais e ambiente de debugação. O modelo em

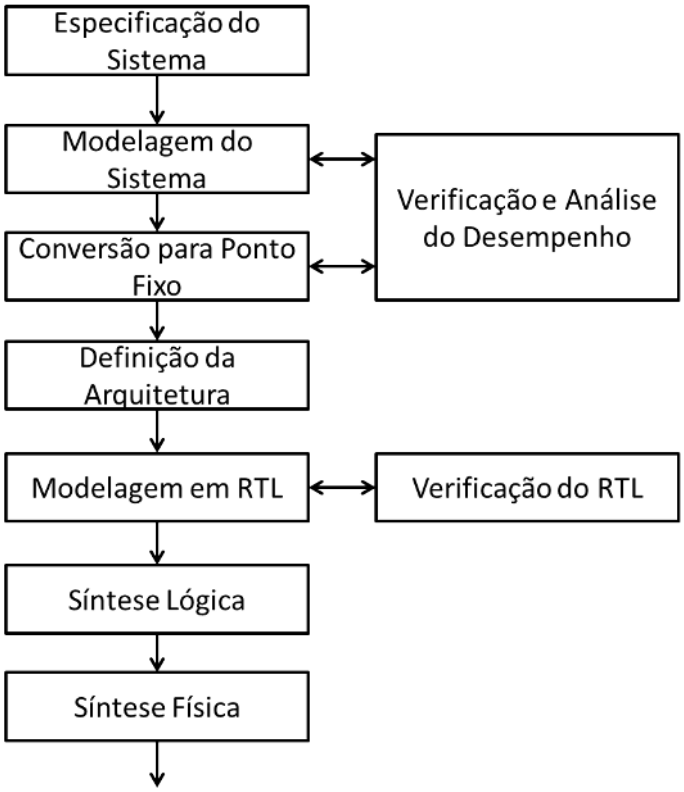

Figura 5.1: Fluxo de projeto de ASIC.

MatLab deve ser verificado para se ter certeza que está livre de erros. Para isso é preciso que um ambiente de simulação seja criado com a funcionalidade mínima de gerar vetores de entrada para o modelo e validar suas respostas, plotando a curva do BER, por exemplo. O ambiente de simulação desenvolvido nessa tese tem ainda as funcionalidades adicionais de permitir a execução de uma sequência de simulações com configurações diferentes, salvar os conjuntos de vetores de entrada com correspondentes vetores de saída e salvar o estado da simulação de tempo em tempo para permitir a recuperação de uma simulação interrompida, seja intencionalmente ou acidentalmente.

Para se verificar o modelo, deve-se definir uma sequência de simulação que cubra todas as linhas de código do algoritmo. Após se chegar a um modelo em MatLab livre de erros, simulações podem ser realizadas para medir o desempenho do algoritmo com diferente configurações paramétricas, conforme foi feito com o K-Melhores no Capítulo 4.

A etapa seguinte é inserir limitações na precisão das variáveis do algoritmo idênticas às limitações que existirão em sua implementação em ASIC. O objetivo disso é

determinar o número de bits que serão usados para representar os dados em seu modelo RTL (*Register-Transfer Level*). Para se chegar a esses valores deve-se primeiro determinar o formato que os valores serão representados no ASIC, que pode ser em ponto-fixo, ponto-flutuante em bloco (PFB) (um mesmo expoente compartilhado por mais de uma mantissa) ou ponto-flutuante (um expoente para cada mantissa). O ponto flutuante e PFB são mais indicados quando há necessidade de representar valores dentro de um intervalo muito grande, o que não é o caso em nosso problema, em que já se sabe que a energia de $\boldsymbol{s}$ e dos elementos de $\boldsymbol{H}$ é sempre em torno de um, e existe um limite para energia de $\boldsymbol{n}$ em que o detector consegue manter um BER abaixo de $0,01$. Por isso, o ponto-fixo fixo foi adotado. A escolha do número de bits é feita, inicialmente, efetuando uma análise matemática, em que se deve determinar os valores máximo e mínimos a serem representados e a relação sinal ruído de discretização a ser alcançada. Essas duas informações permitem determinar o número de bits e a posição do ponto para as variáveis do algoritmo. Uma simulação deve ser feita no final para garantir que a perda de desempenho devido à imposição das limitações de precisão esteja dentro de um limite aceitável. Nesse trabalho, o limite estabelecido foi uma perda de 0,1 dB na curva do BER.

Tendo-se concluído a etapa de sistema, deve-se definir a arquitetura VLSI. Para isso, é preciso estabelecer especificações como requisito de *throughput*, latência, frequência de operação do clock, protocolo de comunicação, entre outros. A arquitetura é geralmente descrita com diagramas de blocos em que se indica o caminho do fluxo de dados, os sinais de controle, e também com a descrição de uma eventual máquina de estado usada para gerar os sinais que controlam o fluxo de dados.

Com a arquitetura definida pode-se iniciar a etapa de codificação da mesma em uma linguagem de descrição de hardware (*Hardware Description Language* - HDL), gerando assim o modelo em RTL do sistema. A linguagem escolhida para essa tarefa foi o Verilog, devido a maior familiarização do autor com essa linguagem. Entre as alternativas ao Verilog, tem-se a linguagem VHDL, sendo essas duas as linguagens HDL mais usadas.

Para verificar o modelo RTL é preciso elaborar um plano de verificação que determina uma sequência de testes a serem efetuados. Esses testes devem cobrir todas as funcionalidades do sistema e casos críticos, como ocorrência de saturação ou transbordo em variáveis internas do modelo RTL. Para gerar os estímulos e as respostas esperadas desses testes, foi utilizado o modelo Matlab em ponto-fixo. O critério usado para considerar o resultado de um teste como positivo foi o *full bit mathching*, que consiste em obter uma resposta do modelo RTL idêntica à resposta do modelo

Matlab em ponto-fixo, até mesmo no bit menos significativo. Um parâmetro que serve como indicativo da qualidade do plano de teste é o índice de cobertura das linhas de código do HDL ao final de execução de todas as simulações. Esse índice é fornecido por alguns softwares de simulação de HDL, como o Incisive HDL Simulator da Cadence, utilizado nesse estudo. Um índice de 100 % é geralmente a meta para projetos do tipo processamento digital de sinais.

A síntese lógica consiste no processo de mapear o código HDL em portas lógicas. Para isso, utiliza-se uma biblioteca de porta-lógicas, que é um arquivo com informações sobre o conjunto de portas lógicas disponibilizadas por uma fábrica de ASIC. Entre as informações dadas para cada porta lógica, estão: lógica da saída, tempo de propagação entre cada pino de entrada e saída, área, capacitância de entrada e potência. Um software é então utilizado para fazer o mapeamento do código HDL em um circuito de portas lógicas utilizando como entrada os códigos HDL, a biblioteca de portas lógicas fornecida pela fábrica e um arquivo com restrições para síntese, como a frequência do clock, capacitância de entrada do circuito, e outros.

A síntese física é a etapa final de projeto. Nessa etapa o posicionado físico das portas lógicas no chip é determinado juntamente com o roteamento de suas conexões. Problemas envolvidos no roteamento, como interferência entre os sinais e outros, são tratados pelo software que realiza essa tarefa. Nesse trabalho não se chegou à etapa de síntese física por falta de tempo.

5.2 Arquitetura VLSI

O diagrama de blocos da arquitetura VLSI está exposto na Figura 5.2. A unidade MCU ALL gera os vetores parciais de símbolo $\boldsymbol{s}^{(2)}$ juntamente com seus respectivos PED. Isto é feito a uma taxa de 4 vetores parciais por pulso de *clock*. A unidade MCU BEST completa um $\boldsymbol{s}^{(2)}$ com o melhor s_1 para ele, conforme a etapa 2 do algoritmo (Seção 3.2). Como cada unidade MCU BEST opera na taxa de um $\boldsymbol{s}^{(2)}$ por pulso de clock, são necessárias 4 unidades para se igualar à taxa do MCU ALL. A etapa 3 do algoritmo é feita por duas unidades: o SORTER e o MCU 5BEST. O primeiro seleciona os 16 $\boldsymbol{s}^{(2)}$ de menor PED, e o segundo estende os $\boldsymbol{s}^{(2)}$ selecionados fazendo a combinação com todos os s_1 contemplados pelo SFS, com exceção do melhor s_1 por já estar coberto pelo MCU BEST. Finalmente, a etapa 4 do algoritmo é feita pelas unidades *Symbol Metric Computer (SMC)* e pelo *Softbit Generator (SBG)*. Existem ao todo 24 SMC: 4 para o símbolo $s_{r,2}$, 4 para o símbolo $s_{i,2}$, 8 para o símbolo $s_{r,1}$ e 8 para o símbolo $s_{i,1}$. Cada SMC gera um vetor de *Bit Metric (BM)*

para um símbolo. O termo BM designa um conjunto de dois valores: o menor ED (melhor métrica) para um bit específico ser +1 e o menor ED para esse mesmo bit ser -1. Vetores de BM relacionados ao mesmo símbolo são então fundidos gerando um único vector de BM para esse símbolo. Esta operação é feita tomando-se a melhor métrica para cada valor dos bits. Finalmente, o SBG gera os *soft-bits* subtraindo as duas palavras de cada BM e multiplicando o resultado pelo fator de escalonamento $\frac{1}{\sigma^2}$, conforme (2.36). A taxa de processamento dessa arquitetura é de 16 clocks por vetor $\boldsymbol{z}$ recebido, no caso de 64-QAM e 16-QAM, e 4 pulsos de clock por vetor $\boldsymbol{z}$, no caso de QPSK. Nas subseções seguintes as unidades MCU ALL, SORTER, MCU 5BEST e SMC são detalhadas.

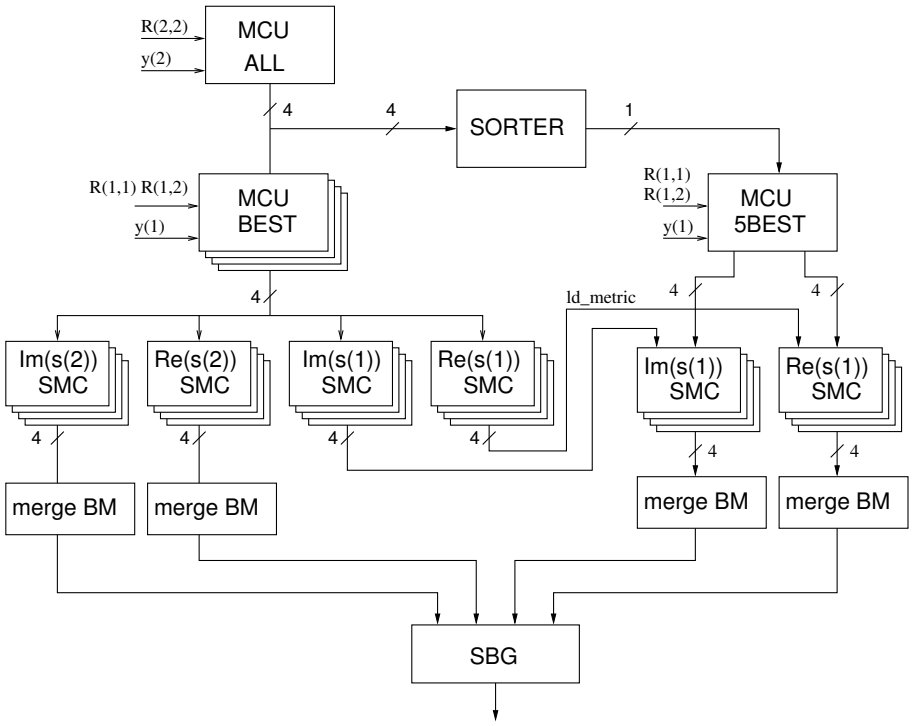

Figura 5.2: Visão geral da arquitetura do detector MIMO 2x2 com SFS.

5.2.1 MCU ALL

O MCU ALL calcula quatro nós $\boldsymbol{s}^{(2)}$ a cada pulso de clock. Os primeiros quatro $\boldsymbol{s}^{(2)}$ gerados são resultado da combinação entre o melhor $s_{r,2}$ com os quatro melhores $s_{i,2}$, ou dois melhores no caso de QPSK. O MCU ALL passa para o próximo $s_{r,2}$ depois de todos os $s_{i,2}$ terem sido combinados com o $s_{r,2}$ atual. A comparação entre $z_{r,2}$ e as constantes 1, 2, 3, 4, 5 e 6 multiplicadas por $r_{2,2}$ gera o sinal que seleciona a

ordem de classificação de $s_{r,2}$ de uma LUT (*look-up table*), conforme exemplificado na Tabela 3.1. O mesmo é feito para o símbolo $s_{i,2}$. Um contador de 3 bits é usado para alternar sequencialmente os símbolos para $s_{r,2}$ e um contador de 1 bit seleciona entre os quatro melhores ou os quatros piores $s_{i,2}$. Se a modulação for 16-QAM ou QPSK, o contador de 1 bit tem seu valor fixado em 0 e o valor máximo do contador de 3 bit passa a ser 3 ou 1.

Na Figura 5.3, a arquitetura para geração sequencial dos símbolos $s_{r,2}$ com respetivos erros incrementais é apresentada juntamente com a arquitetura que combina o $s_{r,2}$ atual com os 4 símbolos $s_{i,2}$ gerados simultaneamente. O número 1 circulado marca o ponto do resultado da equação (3.6) com $k = N_t$, o número 2 marca o resultado da equação (3.4) e o número 3 marca o resultado de (3.3).

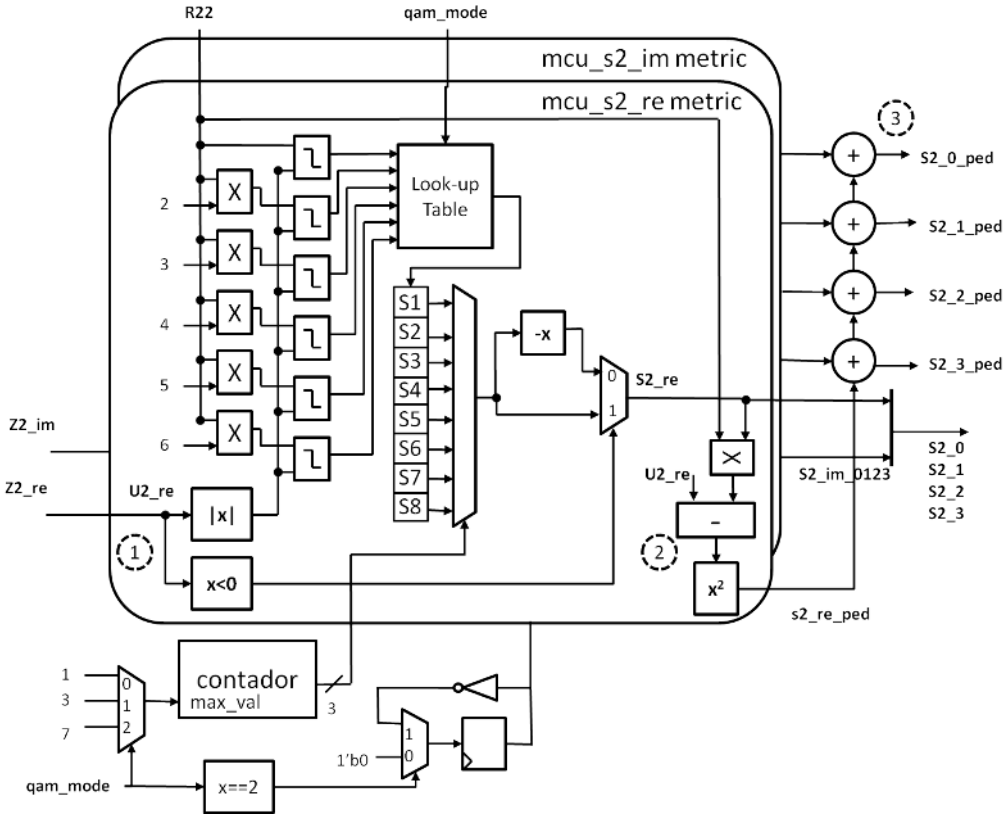

Figura 5.3: Arquitetura do MCU ALL.

5.2.2 SORTER

A cada pulso de clock, O SORTER recebe quatro nós $\boldsymbol{s}^{(2)}$, organizados em ordem ascedente de PED. Nove registradores são usados para armazenar os nós resultantes

da combinação entre os 3 melhores $s_{r,2}$ com os 3 melhores $s_{i,2}$ (conjunto dos pré-selecionados). Uma unidade chamada de Classificador de 7 melhores (*7-Best Sorter*) é usada para achar os sete melhores nós não pertencentes ao conjunto dos pré-selecionados. Isto é feito atualizando o registrador que armazena os 7 melhores a cada novo conjunto de entrada fornecido.

Há duas situações possíveis para o processo de seleção dos 7 melhores: os quatro nós da entrada do SORTER são válidos para seleção, ou apenas o quarto nó é válido, porque os três primeiros pertencem ao conjunto dos pré-selecionados. Nesse segundo caso, a posição do quarto nó é comutada com a do primeiro nó, e as entradas não usadas recebem um valor máximo de PED para serem anuladas.

A taxa de saída do SORTER é de um nó por pulso de clock. Para permitir que o SORTER inicie uma novo processo de seleção enquanto realiza a operação de saída da seleção anterior, um registrador de saída é usado. A Figura 5.4 ilustra a arquitetura do SORTER.

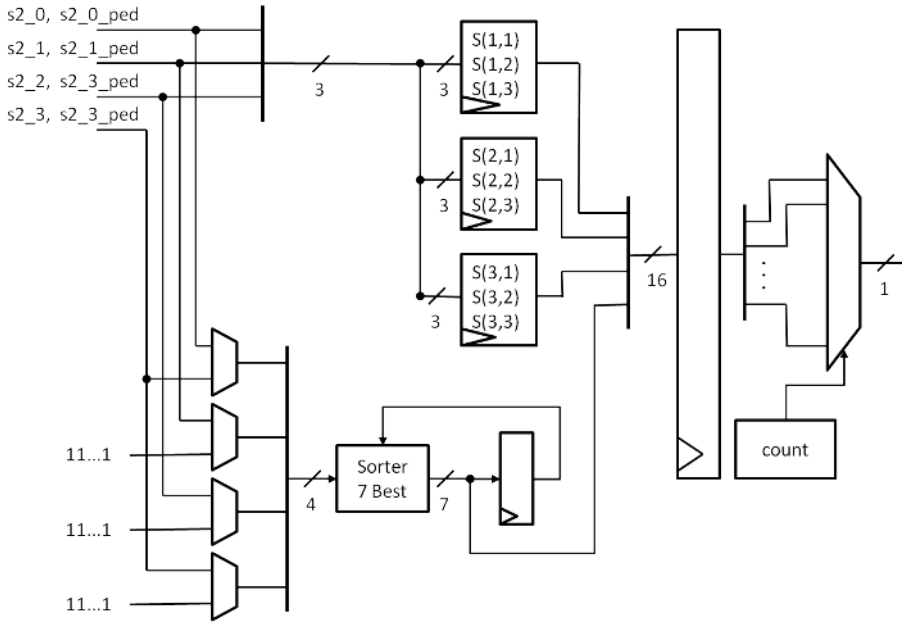

Figura 5.4: Arquitetura do Sorter

5.2.3 MCU 5BEST

O MCU 5BEST calcula o erro incremental dos $2^{M/2-1}+1$ melhores $s_{r,1}$ e os $2^{M/2-1}+1$ melhores $s_{i,1}$ para um dado $\boldsymbol{s}^{(2)}$. Em seguida, calcula o ED de todas as hipóteses para $\boldsymbol{s}$ geradas pela combinação do melhor $s_{i,1}$ com o segundo ao $2^{M/2-1}+1$

melhor $s_{r,1}$. O mesmo é feito entre o melhor $s_{r,1}$ e o conjunto dos $s_{i,1}$ calculados. A Figura 5.5 mostra a parte do MCU 5BEST que seleciona os $2^{M/2-1}+1$ melhores $s_{r,1}$ de um determinado $\boldsymbol{s}^{(2)}$ usando a LUT, e calcula os EDs que resultam da combinação entre estes $s_{r,1}$ com o melhor $s_{i,1}$. Os círculos numerados com 1, 2 e 3 mostram o ponto onde o resultado de (3.6), (3.4) e (3.8) estão disponível, respectivamente.

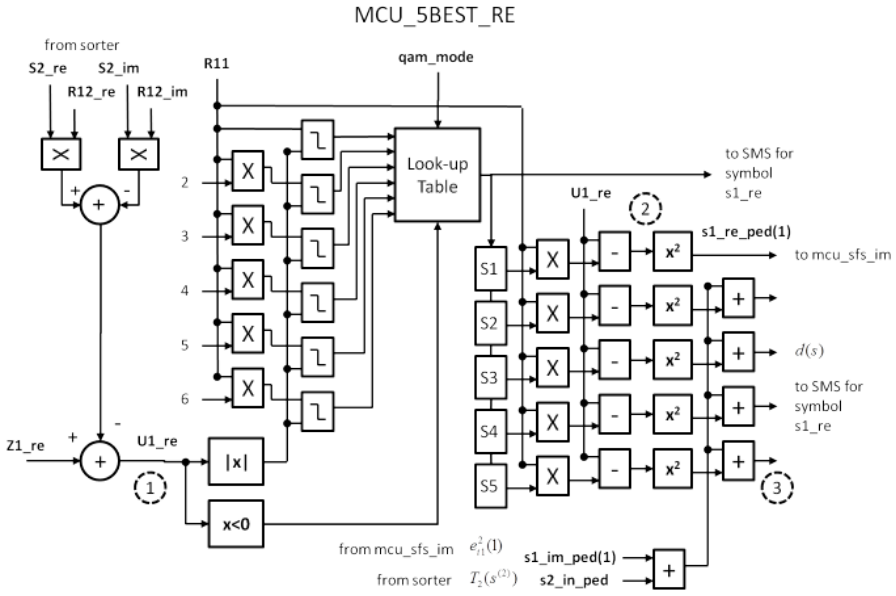

Figura 5.5: Arquitetura do MCU 5BEST

5.2.4 SMC e SBG

As unidades SMC tem como entrada um símbolo que é parte de um vetor $\boldsymbol{s}$, e o ED, ou *path metric* (PM) associado a esse símbolo. A partir de uma sequência dessas entradas, o SMC calcula um vetor de BM para esse símbolo. Cada SMC é composto por um decodificador de símbolo e 3 unidades BMC (*Bit Metric Calculator*), como mostrado na Figura 5.7. O número de BMC ativos depende da modulação. Quando a modulação for 64-QAM os 3 BMC estão ativos já que cada símbolo 64-QAM mapeia 3 bits (considerando a parte real e imaginária como símbolos distintos). Cada BMC possui dois registradores, um que armazena o atual menor erro para o caso do bit ser +1 e outro que armazena o atual menor erro para o caso do bit ser -1. Para cada símbolo fornecido, uma comparação é feita no BMC entre o *path metric* e o atual menor erro para o valor do bit decodificado. Se o *path metric* for menor, o registrador selecionado é atualizado com o valor do *path metric*. No caso do primeiro *path*, por ainda não haver valores válidos nos registradores, a comparação

é feita com as métricas passadas através da entrada LM (*load metric*). O LM tanto pode ser um BM com valores máximos de ED para inicialização do BMC, ou um BM já calculado por outra unidade SMC. A estrutura do BMC é detalhada na Figura 5.7. Na parte esquerda da figura está o hardware que compara o *path metric* com um dos dois valores que compõe o BM ou o LM. Na parte direita, está o mecanismo de atualização dos registradores, controlado pelo resultado da comparação, pelo valor do bit decodificado e pelo sinal *load metric*. Depois de todas as hipóteses para $\boldsymbol{s}$ terem sido processadas pelos SMC, os 12 BM finais são passados para o SBG que faz a subtração entre a métrica para o caso do bit ser -1 e a métrica para o caso do +1, e multiplica o resultado pelo fator $\frac{1}{\sigma^2}$.

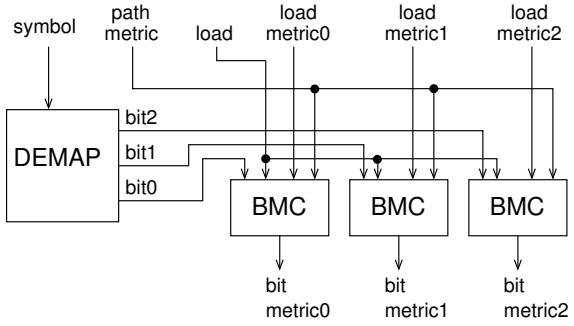

Figura 5.6: Arquitetura do SMC

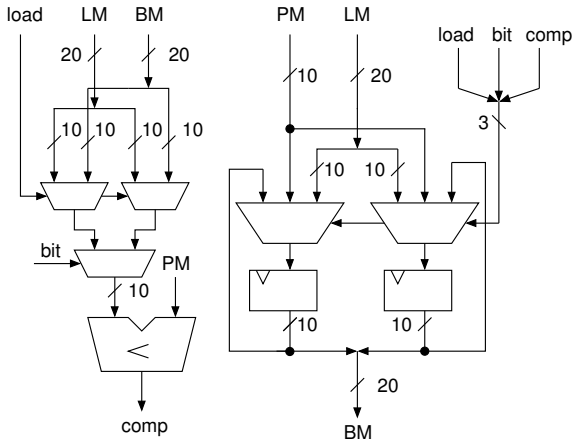

Figura 5.7: Arquitetura do BMC

5.3 Resultado da Síntese Lógica

A arquitetura foi sintetizada usando uma biblioteca de 90 nm e fixando a frequência de *clock* a ser atingida em 120 MHz. O resultado da síntese lógica é apresentado

na Tabela 5.1 juntamente com o resultado de outras arquitetura de detectores MIMO que também possuem a característica de número determinístico de EDs a serem calculados. A taxa de processamento apresentada na tabela é para a maior constelação. Na maioria das arquiteturas com constelação configurável, a taxa de processamento depende da constelação selecionada, sendo normalmente mais alta nas constelações menores.

As arquiteturas [26] IC1 e IC2, e [36] obtêm o max-log-ML para todas as constelações que suportam; a arquitetura aqui proposta e [28] obtêm o max-log-ML para as configurações QPSK e 16-QAM e uma aproximação do max-log-ML para 64-QAM.

Tabela 5.1: Resultado da Implementação

referência	TxR	QAM	taxa fixa	tecnologia (nm)	gates (K)	max. clock (MHz)	taxa (Mbps)
Proposto	2x2	4-64	sim	90	60	122	91.5
[26] IC1	2x2	4-64	sim	65	408	80	240
[26] IC2	2x2	4-64	sim	65	135	80	164
[28]	2x2	4-64	sim	65	55	300	225
[36]	4x4	QPSK	sim	180	168	38.4	19.2

Quanto ao algoritmo implementado, o detector [36] calcula o max-log-ML usando o método direto, mas para configuração 4x4 QPSK, que requer apenas 256 EDs; os detectores [26] IC1 e IC2 utilizam um algoritmo denominado LORD, e [28] utiliza uma variação do LORD. Os detectores baseados no LORD possuem a desvantagem de requererem N_t decomposições QR para cada matriz $\boldsymbol{H}$. Isto duplica a complexidade de seus pré-processador. Portanto, se a área do pré-processador fosse incluída na análise, é provável que o detector proposto obtivesse uma área menor que [28].

5.4 Conclusão

A arquitetura para detecção em sistemas MIMO 2x2 proposta neste capítulo mostrou ter uma área inferior a arquiteturas no estado da arte com desempenho equivalente. A comparação entre as arquiteturas ficou um pouco prejudicada por não incluir a etapa de pré-processamento que difere de um detector para outro.

Capítulo 6

Considerações Finais e Proposta de Trabalho Futuro

Esta tese partiu do pressuposto de que era possível melhorar os algoritmos e arquiteturas VLSI relacionadas ao problema da detecção SM-MIMO. A estratégia escolhida para confirmar essa suposição foi estudar o problema e as soluções atuais para depois desenvolver algoritmos e arquiteturas com ganho de desempenho em relação ao estado da arte. Seguindo essa estratégia, o problema da detecção SM-MIMO foi explicado detalhando-se as diferentes estratégias de detecção – abrupta, suave não iterativa e suave iterativa. Em seguida, os algoritmos clássicos de detecção para tal sistema foram apresentados. Neste ponto, verificou-se que os algoritmos que utilizam o método da busca em árvore permitem atingir ou se aproximar da resposta do detector de busca abrupta ideal com uma complexidade computacional muito inferior a esse. Além disso, foi visto que tais detectores podem ser convertidos em detectores de decisão suave com o método LSD, e que tal estratégia encontra-se no estado da arte da detecção SM-MIMO.

Baseando-se no método da busca em árvore e LSD desenvolveu-se um algoritmo para o cálculo exato da resposta do detector max-log-ML, aproximação muito boa do detector de decisão suave não iterativo ideal. O algoritmo, denominado SFS, oferece uma redução de complexidade de aproximadamente 86 % e 69% no caso da modulação 64-QAM e 16-QAM, respectivamente. Entretanto, o SFS não é praticável em sistemas MIMO com taxa de transmissão muito alta, como 4x4 64-QAM, porque a redução de complexidade do SFS cresce linearmente com o número de antenas transmissoras, e a complexidade do SM-MIMO cresce exponencialmente. Para esses casos, sugeriu-se a combinação de um algoritmo de busca em árvore, como o K-Melhores, com o SFS. O desempenho de tal estratégia não foi analisado.

Dois melhoramentos foram propostos para o algoritmo K-Melhores. O primeiro,

um método de baixa complexidade para ordenamento aproximado dos nós filhos que se aplica ao caso da busca em árvore complexa. Esse método demonstrou não trazer prejuízo significativo ao desempenho do detector. O segundo melhoramento, chamado de K-Melhores Espalhados, consiste em separar as hipóteses de um estágio de detecção em N grupos e selecionar as K/N melhores de cada grupo, ao invés das K-Melhores entre todas. Essa estratégia ameniza o problema do classificador de K Melhores, cujo hardware possui um caminho crítico longo proporcional a K. Assim, o K-Melhores Espalhados permite operar com uma frequência de clock maior, ou utilizar um valor maior para o parâmetro K mantendo a mesma frequência de clock. Portanto, o K-Melhores Espalhados pode ser usado tanto para obter um ganho na taxa de processamento, quanto um ganho de desempenho.

Uma arquitetura de um detector MIMO 2x2 com o algoritmo SFS configurável entre QPSK, 16-QAM e 64-QAM foi desenvolvido e síntetisado em portas lógicas. Na configuração 64-QAM utilizou-se uma aproximação do SFS para reduzir a complexidade. Essa aproximação não degradou, perceptivelmente, o desempenho. O resultado da síntese lógica apontou essa arquitetura como sendo a de menor área em comparação com outras arquiteturas para MIMO 2x2 com desempenho max-log-ML ou muito próximo a isso.

Com as contribuições dessa tese – o algoritmo SFS, o K-Melhores Espalhados e a arquitetura para detecção MIMO 2x2 como SFS – confirmou-se a suposição que ainda há espaço para evolução dos algoritmos de detecção SM-MIMO. Além disso, a constante evolução dos dispositivos microeletrônicos está constantemente elevando o limite da complexidade aceitável para implementação prática. Isto permite que algoritmos antes considerados muito complexos sejam utilizados.

Entre as propostas para trabalhos futuro estão:

- Concluir o fluxo de projeto para desenvolvimento de um ASIC com o detector SFS 2x2, incluíndo uma análise da potência de consumo do detector;
- Simular o K-Melhores Espalhados com outras configurações de MIMO, especialmente o 4x4 64-QAM;
- Implementar e analisar o algoritmo K-Melhores Espalhados terminado com o SFS;
- Desenvolver a arquitetura VLSI do K-melhores Espalhados;
- Fazer uma comparação extensiva entre o K-melhores Espalhados 4x4 com outras arquiteturas;
- Implementar a detecção/decodificação iterativa com o K-Melhores Espalhados.

Referências Bibliográficas

[1] SESIA, I. T. S.; BAKER, M. *LTE-The UMTS Long Term Evolution: From Theory to Practice*. [S.l.]: Academic Press, 2009.

[2] CHERRY, S. Edholm's law of bandwidth. *Spectrum, IEEE*, v. 41, n. 7, p. 58 – 60, 2004. ISSN 0018-9235.

[3] SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, v. 27, p. 379–423, 623–656, July, October 1948.

[4] TELATAR, E. Capacity of multi-antenna gaussian channels. *European Transactions on Telecommunications*, v. 10, n. 6, p. 585–595, Nov./Dec. 1999.

[5] FOSCHINI, G. J.; GANS, M. On limits of wireless communications in a fading environment when using multiple antennas. *Wireless Personal Communications*, v. 6, p. 311–335, 1998.

[6] SKLAR, B. *Digital Communication - Fundamentals and Applications*. 2. ed. [S.l.]: Prentice Hall PTR, 2001.

[7] VUCETIC, B.; YUAN, J. *Space-Time Coding*. [S.l.]: John Wiley & Sons, 2003.

[8] BERROU, C.; GLAVIEUX, A.; THITIMAJSHIMA, P. Near shannon limit error-correcting coding and decoding: Turbo-codes. In: *Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on*. [S.l.: s.n.], 1993. v. 2, p. 1064 –1070 vol.2.

[9] GALLAGER, R. Low-density parity-check codes. *Information Theory, IRE Transactions on*, v. 8, n. 1, p. 21 –28, january 1962. ISSN 0096-1000.

[10] DIGITAL Design with RTL Design, Verilog and VHDL. [S.l.]: John Wiley & Sons, 2011.

[11] PROAKIS, J. G. *Digital Communications*. [S.l.]: McGraw-Hill, 2000.

[12] BURG, A. *VLSI Circuits for MIMO Communication Systems*. Tese (Doutorado) — Swiss Federal Insititute of Technology, 2006.

[13] HOCHWALD, B.; BRINK, S. ten. Achieving near-capacity on a multiple-antenna channel. *Communications, IEEE Transactions on*, v. 51, n. 3, p. 389 – 399, 2003. ISSN 0090-6778.

[14] SOBHANMANESH, F. *Hardware Implementation of V-BLAST MIMO*. Tese (Doutorado) — The University of New South Wales, 2006.

[15] DAHLMAN STEFAN PARKVALL, J. S. E.; BEMING, P. *3G Evolution HSPA and LTE for Mobile Broadband*. [S.l.]: John Wiley & Sons, Ltd, 2008.

[16] ALAMOUTI, S. A simple transmit diversity technique for wireless communications. *Selected Areas in Communications, IEEE Journal on*, v. 16, n. 8, p. 1451 –1458, out. 1998. ISSN 0733-8716.

[17] LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation (3GPP TS 36.211 version 10.0.0 Release 10). [S.l.].

[18] HAGENAUER, E. O. J.; PAPKE, L. Iterative decoding of binary block and convolutional codes. *IEEE Transaction on Information Theory*, 1996.

[19] GHOSH, A. et al. Lte-advanced: next-generation wireless broadband technology [invited paper]. *Wireless Communications, IEEE*, v. 17, n. 3, p. 10 –22, june 2010. ISSN 1536-1284.

[20] ROSS, S. *Stochastic Processes*. [S.l.]: Wiley and Sons, 1996.

[21] WOLNIANSKY, P. et al. V-blast: an architecture for realizing very high data rates over the rich-scattering wireless channel. In: *Signals, Systems, and Electronics, 1998. ISSSE 98. 1998 URSI International Symposium on*. [S.l.: s.n.], 1998.

[22] LUETHI, P. et al. Vlsi implementation of a high-speed iterative sorted mmse qr decomposition. In: *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. [S.l.: s.n.], 2007. p. 1421 –1424.

[23] LIN, K.-H. et al. Iterative qr decomposition architecture using the modified gram-schmidt algorithm. In: *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*. [S.l.: s.n.], 2009. p. 1409 –1412.

[24] ROBERTSON, P.; VILLEBRUN, E.; HOEHER, P. A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain. In: *Communications, 1995. ICC '95 Seattle, 'Gateway to Globalization', 1995 IEEE International Conference on.* [S.l.: s.n.], 1995. v. 2, p. 1009 –1013 vol.2.

[25] GUO, Z.; NILSSON, P. Algorithm and implementation of the k-best sphere decoding for mimo detection. *Selected Areas in Communications, IEEE Journal on*, v. 24, n. 3, p. 491 – 503, march 2006. ISSN 0733-8716.

[26] CUPAIUOLO, T.; SITI, M.; TOMASONI, A. Low-complexity high throughput vlsi architecture of soft-output ml mimo detector. In: *Design, Automation Test in Europe Conference Exhibition (DATE), 2010.* [S.l.: s.n.], 2010. p. 1396 –1401. ISSN 1530-1591.

[27] LIAO, C.-H.; WANG, T.-P.; CHIUEH, T.-D. A 74.8 mw soft-output detector ic for 8 ×, 8 spatial-multiplexing mimo communications. *Solid-State Circuits, IEEE Journal of*, v. 45, n. 2, p. 411 –421, feb. 2010. ISSN 0018-9200.

[28] WU JOHAN EILERT, R. A. e. D. L. D. Vlsi implementation of a fixed-complexity soft-output mimo detector for high-speed wireless. *EURASIP Jornal on Wireless Communications and Networking*, 2010.

[29] BURG M. BORGMANN, M. W. M. Z. W. F. e. H. B. A. Vlsi implementation of mimo detection using the shpere decoding algorithm. *IEEE J. Solid-State Circuits*, v. 40, p. 1566–1577, 2005.

[30] BHAGAWAT, R. D. e. G. C. P. Dynamically reconfigurable soft output mimo detector. In: *Computer Design, 2008. ICCD 2008. IEEE International Conference on.* [S.l.: s.n.], 2008. p. 68 –73. ISSN 1063-6404.

[31] PARK, J. L. J.-W. C. H.-L. L. J. Soft mimo ml demodulator based on bitwise constellation partitioning. *Communications Letters, IEEE*, 2009.

[32] YIN, J. H. e S. Vlsi architecture of a k-best detector for mimo-ofdm wireless communication systems. *Journal of Semiconductors - Chinese Institute of Electronics*, v. 30, n. 7, july 2009.

[33] MYLLYLA, M. et al. The effect of llr clipping to the complexity of list sphere detector algorithms. In: *Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on.* [S.l.: s.n.], 2007. p. 1559 –1563. ISSN 1058-6393.

[34] SITI, M.; FITZ, M. On layer ordering techniques for near-optimal mimo detectors. In: *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*. [S.l.: s.n.], 2007. p. 1199 –1204. ISSN 1525-3511.

[35] CHEN, S.; ZHANG, T.; XIN, Y. Relaxed k -best mimo signal detector design and vlsi implementation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, v. 15, n. 3, p. 328 –337, march 2007. ISSN 1063-8210.

[36] GARRETT, L. D. S. B. B. H.; KNAGGE, G. Silicon complexity for maximum likelihood mimo detection using spherical decoding. *IEEE Journal of Solid-State Circuits*, 2004.

Apêndice A

Códigos MatLab

A.1 Modelo do Sistema MIMO

```
function state = main_mimo(param, state)

% load main_home variable which contains a string with main ...
    directory path
load( 'main_home.mat' );

% generate coded parameters
param = gen_coded_param(param);

% intialize state variables
state = initialize_state(param, state);

% constants
QAM = 2^param.bitQAM;
Nt=param.Nt;          % number of transmit antenna (not configurable)
Nr=param.Nr;          % number of receive antenna
M = param.bitQAM;     % num of bit mapped to one complex ...
    constelation symbol
Nt_M = Nt*M;          % number of bit mapped to one symbol vector
S = CreateConstellation('LTE', M, Nt);

% creates LDPC encode decoder object
ldpc = Ldpc(Matrix, ...
    param.rate, ...
    param.block_length, ...
    param.decisionType, ...
    'Information part', ...
    param.numIterations, ...
    param.doParityChecks);
```

```
28
29 % calculates systematics length
30 systematic_length = ldpc.getSystematicLength(param.rate, ...
       param.block_length);
31
32 % save time
33 t1 = tic;
34
35 for snr_idx = 1:length(param.SNR)
36     fprintf('SNR = %d\n', param.SNR(snr_idx));
37     N0 = 10^(-param.SNR(snr_idx)/10); % sigma^2
38
39     while (( state.trials(snr_idx) < param.max_trials ) && ...
40           ( state.frame_err(snr_idx) < param.max_frame_errors ))
41
42         state.trials(snr_idx) = state.trials(snr_idx) + 1;
43
44         % generate random input bit sequency
45         systematic = randi(0:1, 1, systematic_length);
46
47         % LDPC enconding
48         codeword = ldpc.encoderRef(systematic);
49
50         % Modulation
51         s_vec = map2sym(codeword, M, S);
52
53         % number of transmitted symbol vectors
54         num_of_trans = length(s_vec)/Nt;
55
56         % initialize variables
57         llr_codeword = zeros(1,num_of_trans*Nt_M);
58         llr_idx = 1;
59         for k=1:num_of_trans
60
61             % generate channel matrix
62             H = sqrt(1/2)*(randn(Nr, Nt)+1j*randn(Nr, Nt));
63             %H = eye(Nr,Nt); % for debug
64
65             % generate noise
66             noise = sqrt(N0/2)*(randn(Nr,1) + 1j*randn(Nr,1));
67             %noise = zeros(Nr,1); % for debug
68
69             % Channel transformation and noise addition
```

```
70              y = H*s_vec(Nt*(k-1)+1:Nt*k).' + noise;
71
72              %-------------------------------------------------------------------------
73              % Complex 2x2 QR Decomposition
74              % R = Q' * H
75              % z = Q' * y
76              %-------------------------------------------------------------------------
77              if(param.sorted_qr)
78                 [Q R p]= sorted_qr(H);
79              else
80                 %[Q R] = qr_givens_rot(H);
81                 [Q R] = qr_givens_rot_4x4(H);
82                 p = 1:Nt;
83              end
84              z = Q'*y;
85
86              %-------------------------------------------------------------------------
87              % MIMO detector algorithms
88              %-------------------------------------------------------------------------
89              switch(param.detector)
90                 case 'sfs'
91                    %mimo_output=sfs_2x2(z, R, QAM, N0);
92                    mimo_output=sfs_2x2_mex(z, R, QAM, N0);
93                 case 'aprox_sfs'
94                    %mimo_output=aprox_sfs_2x2(z, R, QAM, N0, 1);
95                    mimo_output=aprox_sfs_2x2_mex(z, R, QAM, N0, 1);
96                 case 'cplxkbest'
97                    %mimo_output = cplx_k_best(z, R, QAM, N0, ...
                          param.num_of_kbest, param.kbest, ...
                          param.admitchild, p, param.idealsort);
98                    mimo_output = cplx_k_best_4x4(z, R, QAM, N0, ...
                          param.num_of_kbest, param.kbest, ...
                          param.admitchild, p, param.idealsort);
99              end
100
101             llr_codeword(llr_idx:llr_idx+Nt_M-1) = -mimo_output;
102             llr_idx = llr_idx + Nt_M;
103         end
104
105         %-------------------------------------------------------------------------
106         % Decoding
107         %-------------------------------------------------------------------------
108         estimated_msg = ldpc.decoderRef(llr_codeword);
```

```
109
110         %-------------------------------------------------------------------------
111         % Computing Errors
112         %-------------------------------------------------------------------------
113         error_cnt = 0;
114         for k=1:length(systematic)
115             error_cnt=error_cnt+~isequal(systematic(k),estimated_msg(k));
116         end
117         state.bit_err(snr_idx) = state.bit_err(snr_idx) + error_cnt;
118
119         if (error_cnt>0)
120             state.frame_err(snr_idx) = state.frame_err(snr_idx)+1;
121         end
122
123         % if time from last tic is greater than save_period, save ...
                simulation state
124         if toc(t1) > param.save_period
125             t1 = tic;
126             saved_state = state;
127             saved_param = param;
128             save([main_home saved_param.filename], 'saved_param', ...
                    'saved_state');
129             fprintf('trial=%d  frame error = ...
                    %d\n',state.trials(snr_idx), state.frame_err(snr_idx));
130         end
131
132     end
133
134     %-------------------------------------------------------------------------
135     % BER and FER
136     %-------------------------------------------------------------------------
137     state.BER(snr_idx) = ...
            state.bit_err(snr_idx)/(state.trials(snr_idx)*systematic_length);
138     state.FER(snr_idx) = ...
            state.frame_err(snr_idx)/state.trials(snr_idx);
139
140      fprintf('BER = %f\n',state.BER(snr_idx));
141
142     if state.BER(snr_idx)<param.minBER
143         break;
144     end
145
146 end
```

A.2 SFS 2x2

```
function softbit=sfs_2x2(zcplx, Rcplx, QAM, variance) %#codegen

Nt=2;                    % number of transmitter (not configurable)
Qrvd = round(sqrt(QAM)); % 8, 4, 2
M = log2(QAM);           % num of bit mapped to 1 complex ...
    constelation symbol

%-------------------------------------------------------------------------
% gain to allow use integer values for the hypotesis
%-------------------------------------------------------------------------
switch(QAM)
case 64
   zcplx = zcplx * sqrt(Nt*42);
case 16
   zcplx = zcplx * sqrt(Nt*10);
case 4
   zcplx = zcplx * sqrt(Nt*2);
end

%-------------------------------------------------------------------------
% Real Value Decompostion
%-------------------------------------------------------------------------
R = [ real(Rcplx(1,1)) -imag(Rcplx(1,1)) real(Rcplx(1,2)) ...
    -imag(Rcplx(1,2));
     imag(Rcplx(1,1)) real(Rcplx(1,1))  imag(Rcplx(1,2))  ...
         real(Rcplx(1,2));
     real(Rcplx(2,1)) -imag(Rcplx(2,1)) real(Rcplx(2,2)) ...
         -imag(Rcplx(2,2));
     imag(Rcplx(2,1)) real(Rcplx(2,1))  imag(Rcplx(2,2))  ...
         real(Rcplx(2,2));
    ];
z = [real(zcplx(1)); imag(zcplx(1)); real(zcplx(2)); imag(zcplx(2))];


%-------------------------------------------------------------------------
% First Level of Search-Tree
%-------------------------------------------------------------------------
switch(QAM)
    case 64
        si = [-7, -5, -3, -1, +1, +3, +5, +7];
    case 16
```

```
37          si = [-3, -1, +1, +3];
38      otherwise % case 4
39          si = [-1 +1];
40  end
41
42  ped2 = zeros(QAM,1);
43  path2 = zeros(QAM,2);
44  for k1=0:Qrvd-1
45     % comput error(imag(s(2)))^2
46     erro_im = (z(4)-R(4,4)*si(k1+1))^2;
47     for k2=0:Qrvd-1
48        % comput error(real(s(2)))^2
49        erro_re = (z(3)-R(3,3)*si(k2+1))^2;
50        % combine imag and real s(2) errors and symbols to
51        % generate all the possibilities
52        ped2(Qrvd*k1+k2+1,1) = erro_im + erro_re;
53        path2(Qrvd*k1+k2+1, 1:2) = [si(k2+1) si(k1+1)];
54     end
55  end
56
57
58  %-------------------------------------------------------------------------
59  % Second Level of Search-Tree (SFS)
60  %-------------------------------------------------------------------------
61  % initialize variable
62  ed_vec= zeros(QAM*(Qrvd+1),1);
63  path_vec = zeros(QAM*(Qrvd+1),4);
64
65  for k=0:QAM-1
66     % error(imag(s(1)))^2 in Ascending Order
67     [inc_dist_im, path1_im]=mcu5(2, z(2), R(2,:), [0 0 ...
          path2(k+1,:)], QAM);
68     % error(real(s(1)))^2 in Ascending Order
69     [inc_dist_re, path1_re]=mcu5(1, z(1), R(1,:), [0 0 ...
          path2(k+1,:)], QAM);
70
71     % ErrorDist = error(real(s(1)))^2 + error(imag(s(1)))^2 + ...
          parent_node PED
72     % path = [ real(s(1)) imag(s(1)) real(s(2)) imag(s(2))]
73     switch(QAM)
74     case 64  %effort reduced to 14.06%  9/64
75        ed_vec(9*k+1) = inc_dist_re(1)+inc_dist_im(1)+ped2(k+1); % Best
76        ed_vec(9*k+(2:5)) = inc_dist_re(1)+inc_dist_im(2:5)+ped2(k+1);
```

```
77        ed_vec(9*k+(6:9)) = inc_dist_re(2:5)+inc_dist_im(1)+ped2(k+1);
78
79        path_vec(9*k+1, :) = [path1_re(1) path1_im(1) path2(k+1,:)];
80        path_vec(9*k+2, :) = [path1_re(1) path1_im(2) path2(k+1,:)];
81        path_vec(9*k+3, :) = [path1_re(1) path1_im(3) path2(k+1,:)];
82        path_vec(9*k+4, :) = [path1_re(1) path1_im(4) path2(k+1,:)];
83        path_vec(9*k+5, :) = [path1_re(1) path1_im(5) path2(k+1,:)];
84        path_vec(9*k+6, :) = [path1_re(2) path1_im(1) path2(k+1,:)];
85        path_vec(9*k+7, :) = [path1_re(3) path1_im(1) path2(k+1,:)];
86        path_vec(9*k+8, :) = [path1_re(4) path1_im(1) path2(k+1,:)];
87        path_vec(9*k+9, :) = [path1_re(5) path1_im(1) path2(k+1,:)];
88
89     case 16 %effort reduced to 31.25%   5/16
90        ed_vec(5*k+1) = inc_dist_re(1)+inc_dist_im(1)+ped2(k+1); % Best
91        ed_vec(5*k+(2:3)) = inc_dist_re(1)+inc_dist_im(2:3)+ped2(k+1);
92        ed_vec(5*k+(4:5)) = inc_dist_re(2:3)+inc_dist_im(1)+ped2(k+1);
93
94        path_vec(5*k+1, :) = [path1_re(1) path1_im(1) path2(k+1,:)];
95        path_vec(5*k+2, :) = [path1_re(1) path1_im(2) path2(k+1,:)];
96        path_vec(5*k+3, :) = [path1_re(1) path1_im(3) path2(k+1,:)];
97        path_vec(5*k+4, :) = [path1_re(2) path1_im(1) path2(k+1,:)];
98        path_vec(5*k+5, :) = [path1_re(3) path1_im(1) path2(k+1,:)];
99     case 4 %effort reduced to 75%      3/4
100       ed_vec(3*k+1) = inc_dist_re(1)+inc_dist_im(1)+ped2(k+1); % Best
101       ed_vec(3*k+2) = inc_dist_re(1)+inc_dist_im(2)+ped2(k+1);
102       ed_vec(3*k+3) = inc_dist_re(2)+inc_dist_im(1)+ped2(k+1);
103       path_vec(3*k+1, :) = [path1_re(1) path1_im(1) path2(k+1,:)];
104       path_vec(3*k+2, :) = [path1_re(1) path1_im(2) path2(k+1,:)];
105       path_vec(3*k+3, :) = [path1_re(2) path1_im(1) path2(k+1,:)];
106    end
107
108 end
109
110 %-------------------------------------------------------------------------
111 % Bitmetric Computation
112 %-------------------------------------------------------------------------
113
114  % bitmetric vector inicialization
115  bm.val1 = 100000*ones(1,M*Nt);
116  bm.val0 = 100000*ones(1,M*Nt);
117  hypoth.val1 = zeros(M*Nt,4);
118  hypoth.val0 = zeros(M*Nt,4);
119
```

```
120  % The bm holds the minimal squared error for each bit value ...
        possibility
121  % when this external loop is done
122  for k=1:QAM*(Qrvd+1)
123
124     % get one path from path_vec and mapped it to a bit vector
125     bit_vec= map2bit(path_vec(k,:),QAM); %bit vector
126
127     % get its correspondent error distance
128     pmetric = ed_vec(k);      % path metric
129
130     % check if there is any bm value that must be updated
131     for bitp=1:M*Nt           % bit postion
132        if( bit_vec(bitp) == 1  &&  pmetric<bm.val1(bitp))
133           bm.val1(bitp)=pmetric;
134           hypoth.val1(bitp,:) = path_vec(k,:); % for debug propose only
135        elseif( bit_vec(bitp) == 0  &&  pmetric<bm.val0(bitp))
136           bm.val0(bitp)=pmetric;
137           hypoth.val0(bitp,:) = path_vec(k,:); % for debug propose only
138        end
139     end
140
141  end
142 hypoth.val1; % debug
143 hypoth.val0; % debug
144
145 %-------------------------------------------------------------------------
146 % Soft-Bit
147 % - Compute the soft-bits using the BMs
148 %-------------------------------------------------------------------------
149 softbit = (bm.val0 - bm.val1);
150
151 % power adjustment
152 % The reason for softbit be divied by 42 instead of sqrt(42) is because
153 % the softbits where computed using the squared error
154 switch(QAM)
155    case 64
156       softbit = softbit/(2*42);
157    case 16
158       softbit = softbit/(2*10);
159    case 4
160       softbit = softbit/(2*2);
161 end
```

```
162 softbit = softbit/variance;
```

A.3 SFS 2x2 Aproximado

```
1  function softbit=aprox_sfs_2x2(zcplx, Rcplx, QAM, variance, opt_sort)
2
3  Nt=2;                    % number of transmitter (not configurable)
4  Qrvd = round(sqrt(QAM)); % 8, 4, 2
5  M = log2(QAM);           % num of bit mapped to 1 complex ...
       constelation symbol
6
7  % limit precision of z and R
8  % fxp_var = fxpRound(flat_var, num_of_bit, num_of_frac_bit);
9  zcplx = fxpRound(zcplx, 10, 10-3);
10 Rcplx = fxpRound(Rcplx, 11, 7);
11
12 %-------------------------------------------------------------------------
13 % gain to allow use integer values for the hypotesis
14 %-------------------------------------------------------------------------
15 switch(QAM)
16 case 64
17    zcplx = zcplx * fxpRound(sqrt(Nt*42),12,6);
18 case 16
19    zcplx = zcplx * fxpRound(sqrt(Nt*10),12,6);
20 case 4
21    zcplx = zcplx * fxpRound(sqrt(Nt*2),12,6);
22 end
23 zcplx = fxpRound(zcplx, 10, 10-6);
24
25 %-------------------------------------------------------------------------
26 % Real Value Decompostion
27 %-------------------------------------------------------------------------
28 R = [ real(Rcplx(1,1)) -imag(Rcplx(1,1)) real(Rcplx(1,2)) ...
       -imag(Rcplx(1,2));
29       imag(Rcplx(1,1)) real(Rcplx(1,1))  imag(Rcplx(1,2))  ...
             real(Rcplx(1,2));
30       real(Rcplx(2,1)) -imag(Rcplx(2,1)) real(Rcplx(2,2)) ...
             -imag(Rcplx(2,2));
31       imag(Rcplx(2,1)) real(Rcplx(2,1))  imag(Rcplx(2,2))  ...
             real(Rcplx(2,2));
32     ];
33 z = [real(zcplx(1)); imag(zcplx(1)); real(zcplx(2)); imag(zcplx(2))];
34
```

```
35 %-------------------------------------------------------------------------
36 % Compute all nodes in level s(2)
37 %-------------------------------------------------------------------------
38 s= zeros(1,2*Nt); % intial parent node
39
40 % error(imag(s(2)))^2 in Ascending Order
41 [ped2_im, path2_im]=mcu(4, z(4), R(4,:), s, QAM, 10, 4);
42 ped2_im = fxpfloor(ped2_im, 10+1, 5);
43
44 % error(real(s(1)))^2 in Ascending Order
45 [ped2_re, path2_re]=mcu(3, z(3), R(3,:), s, QAM, 10, 4);
46 ped2_re = fxpfloor(ped2_re, 10+1, 5);
47
48 % Level1 Partial Error Distance (PED)
49 % ped = error(real(s(2)))^2 + error(imag(s(2)))^2
50 % path = [real(s(2)) imag(s(2))];
51  ped2 = zeros(QAM,1);
52  path2 = zeros(QAM,2);
53  for k1=0:Qrvd-1
54     for k2=0:Qrvd-1
55        ped2(Qrvd*k1+k2+1) = ped2_re(k1+1) + ped2_im(k2+1);
56        path2(Qrvd*k1+k2+1,:) = [path2_re(k1+1) path2_im(k2+1)];
57     end
58  end
59  ped2 = fxpfloor(ped2, 10+1, 5);
60
61 %-------------------------------------------------------------------------
62 % Extend the paths to the botton of tree getting only the best child
63 %-------------------------------------------------------------------------
64 ed_vec = zeros(QAM,1);
65 path_vec = zeros(QAM,2*Nt);
66 for k=0:QAM-1
67     [error_dist, path]= mcu_best(2, z, R, [0 0 path2(k+1,:)], ...
          ped2(k+1), QAM, 10, 10-6);
68     ed_vec(k+1)= error_dist;
69     path_vec(k+1,:)= path;
70 end
71 ed_vec = fxpfloor(ed_vec, 10+1, 5);
72
73 %-------------------------------------------------------------------------
74 % Sorting of 16 Best from previous level
75 %-------------------------------------------------------------------------
76 ped_16b = 100000*ones(min(QAM,16),1);  % sorter initialization
```

```
77 path_16b = zeros(min(QAM,16),2);       % sorter initialization
78 if(QAM==64)
79    % Commun Algorithm
80    if(opt_sort==0)
81       for k1 = 0:Qrvd-1
82        [ped_16b path_16b]= sorter( ped_16b, path_16b, ...
83                   ped2(Qrvd*k1+(1:Qrvd)), ...
                        path2(Qrvd*k1+(1:Qrvd),:), 16);
84       end
85    else
86       % Optimized Algorithm
87       % The 9 nodes that results from the combination between the
88       % 3 best Im(s(2)) with the 3 best Re(s(2)) are always inside ...
            the 16 bests.
89       ped_16b(1:9) = [ped2(0*8+(1:3)); ped2(1*8+(1:3)); ...
            ped2(2*8+(1:3))];
90       path_16b(1:9,:) = [path2(0*8+(1:3),:);
91                          path2(1*8+(1:3),:);
92                          path2(2*8+(1:3),:)];
93
94       % Now, we only need to find the other 7 nodes that are inside ...
            the 16 best
95       pedsort = 100000*ones(7,1);
96       pathsort = zeros(7,2);
97       for k1=0:7
98          if(k1<3)
99             [pedsort, pathsort]= sorter(pedsort, pathsort, ...
100                                  ped2(k1*8+(4:7)), ...
                                        path2(k1*8+(4:7),:), 7);
101         else
102            [pedsort, pathsort]= sorter(pedsort, pathsort, ...
103                                  ped2(k1*8+(1:7)), ...
                                        path2(k1*8+(1:7),:), 7);
104         end
105      end
106      ped_16b(10:16,1)=pedsort;
107      path_16b(10:16,:)=pathsort;
108   end
109 else  % QAM==16 or QAM==4
110   ped_16b =  ped2;
111   path_16b = path2;
112 end
113
```

```
114 %-------------------------------------------------------------------------
115 % - Get the Qrvd/2+1 best Im(s(2)) and Qrvd/2+1 best Re(s(2)) for ...
       each K-Best
116 % path from level 1
117 % - Combine the Best Re(s(2)) with the 2nd to the Qrvd/2+1 bests ...
       Im(s(2)) to
118 % generate hypotesis for Im(s(2)) bits.
119 % - Combine the Best Im(s(2)) with the 2nd to the Qrvd/2+1 bests ...
       Re(s(2)) to
120 % generate hypotesis for Re(s(2)) bits.
121 %-------------------------------------------------------------------------
122 if (QAM==4)
123     Kbest=4;
124 else
125     Kbest=16;
126 end
127 ed_vec_im = zeros(16*(Qrvd/2),1);
128 ed_vec_re = zeros(16*(Qrvd/2),1);
129 path_vec_im = zeros(16*(Qrvd/2),1);
130 path_vec_re = zeros(16*(Qrvd/2),1);
131 for k=0:Kbest-1
132    % error(imag(s(1)))^2 in Ascending Order
133    [inc_dist_im, path1_im]=mcu5(2, z(2), R(2,:), [0 0 ...
          path_16b(k+1,:)], QAM, 10, 4);
134
135    % error(real(s(1)))^2 in Ascending Order
136    [inc_dist_re, path1_re]=mcu5(1, z(1), R(1,:), [0 0 ...
          path_16b(k+1,:)], QAM, 10, 4);
137
138    % ed_vec_im = min(error(s1_re)^2) + error(s1_im)^2 + parent_node_ped
139    % path_vec_im = s1_im;
140    % ed_vec_re = error(s1_re)^2 + min(error(s1_im)^2) + parent_node_ped
141    % path_vec_re = s1_re;
142    switch(QAM)
143    case 64
144       ed_vec_im(4*k+(1:4)) = ...
             inc_dist_im(2:5)+inc_dist_re(1)+ped_16b(k+1); %ED
145       ed_vec_re(4*k+(1:4)) = ...
             inc_dist_re(2:5)+inc_dist_im(1)+ped_16b(k+1); %ED
146       path_vec_im(4*k+(1:4)) = path1_im(2:5);
147       path_vec_re(4*k+(1:4)) = path1_re(2:5);
148    case 16
149       ed_vec_im(2*k+(1:2)) = ...
```

```
            inc_dist_im(2:3)+inc_dist_re(1)+ped_16b(k+1); %ED
150         ed_vec_re(2*k+(1:2)) = ...
               inc_dist_re(2:3)+inc_dist_im(1)+ped_16b(k+1); %ED
151         path_vec_im(2*k+(1:2)) = path1_im(2:3);
152         path_vec_re(2*k+(1:2)) = path1_re(2:3);
153      case 4
154         ed_vec_im(k+1) = inc_dist_im(2)+inc_dist_re(1)+ped_16b(k+1); %ED
155         path_vec_im(k+1) = path1_im(2);
156         ed_vec_re(k+1) = inc_dist_re(2)+inc_dist_im(1)+ped_16b(k+1); %ED
157         path_vec_re(k+1) = path1_re(2);
158      end
159   end
160   ed_vec_re = fxpfloor(ed_vec_re, 11, 5);
161   ed_vec_im = fxpfloor(ed_vec_im, 11, 5);
162
163
164   %-------------------------------------------------------------------------
165   % Bitmetric Computation 1
166   % Paths with best cmplx s(1) are used to compute the BM of all bits
167   %-------------------------------------------------------------------------
168    % bitmetric vector inicialization
169    bm.val1 = 100000*ones(1,M*Nt);
170    bm.val0 = 100000*ones(1,M*Nt);
171
172    for k=0:QAM-1
173       bit_vec=map2bit(path_vec(k+1,:),QAM); %bit vector
174       pmetric = ed_vec(k+1);      % path metric
175       for bitp=0:M*Nt-1           % bit postion
176          if( bit_vec(bitp+1)==1 && pmetric<bm.val1(bitp+1))
177             bm.val1(bitp+1)=pmetric;
178          elseif( bit_vec(bitp+1)==0 && pmetric<bm.val0(bitp+1))
179             bm.val0(bitp+1)=pmetric;
180          end
181       end
182    end
183
184   %-------------------------------------------------------------------------
185   % Bitmetric Computation 2
186   % Paths without best cmplx s(1) are used improve BM related to s(1)
187   %-------------------------------------------------------------------------
188   % Initialize BMs with previous computed BMs for complex symbol s(1)
189   s1_re_bm.val0 = bm.val0(1, 1:M/2 );
190   s1_re_bm.val1 = bm.val1(1, 1:M/2 );
```

```
191
192 s1_im_bm.val0 = bm.val0(1, M/2+1:M );
193 s1_im_bm.val1 = bm.val1(1, M/2+1:M );
194
195 for k=0:Kbest-1
196    for child=0:Qrvd/2-1
197
198       s1_re = path_vec_re(Qrvd/2*k+child+1); % get symbol real(s(1))
199       bitvect_re = map2bit(s1_re,QAM);
200       pmetric_re = ed_vec_re(Qrvd/2*k+child+1);
201
202       % bit metric for simbol real(s_ref(1))
203       for bitp=0:M/2-1  % bit position
204          bit_val = bitvect_re(bitp+1);
205          if(bit_val==1)
206             if(pmetric_re<s1_re_bm.val1(bitp+1))
207                s1_re_bm.val1(bitp+1)=pmetric_re;
208             end
209          else
210             if(pmetric_re<s1_re_bm.val0(bitp+1))
211                s1_re_bm.val0(bitp+1)=pmetric_re;
212             end
213          end
214       end
215
216       s1_im = path_vec_im(Qrvd/2*k+child+1); % get symbol imag(s(1))
217       bitvect_im = map2bit(s1_im,QAM);
218       pmetric_im = ed_vec_im(Qrvd/2*k+child+1);
219
220       % bit metric for simbol imag(s_ref(1))
221       for bitp=0:M/2-1  % bit position
222          bit_val = bitvect_im(bitp+1);
223          if(bit_val==1)
224             if(pmetric_im<s1_im_bm.val1(bitp+1))
225                s1_im_bm.val1(bitp+1)=pmetric_im;
226             end
227          else
228             if(pmetric_im<s1_im_bm.val0(bitp+1))
229                s1_im_bm.val0(bitp+1)=pmetric_im;
230             end
231          end
232       end
233
```

```
234     end
235 end
236
237 % Update the full BM vector
238 bm.val0(1,1:M)=[s1_re_bm.val0 s1_im_bm.val0];
239 bm.val1(1,1:M)=[s1_re_bm.val1 s1_im_bm.val1];
240
241 %-------------------------------------------------------------------------
242 % Soft-bits generation using BMs
243 %-------------------------------------------------------------------------
244 softbit = bm.val0 - bm.val1;
245
246 % power adjustment
247 switch(QAM)
248     case 64
249         softbit = softbit * fxpRound(1/(2*42),11,10);
250     case 16
251         softbit = softbit * fxpRound(1/(2*10),11,10);
252     case 4
253         softbit = softbit * fxpRound(1/(2*2),11,10);
254 end
255 softbit = fxpRound(softbit,10,8);
256 softbit = softbit/variance;
```

A.4 K Melhores Espalhados

```
1  function softbit=cplx_k_best(z, R, QAM, variance, num_of_kbest, ...
       kbest, abest, p, idealsort) %#codegen
2
3  M = round(log2(QAM));
4  Nt = length(z);
5
6  % clipping_ed
7  % the transmitte signal power is 1
8  % let's limit ED to
9  % .5 for 64QAM symbols
10 % 1  for 16QAM
11 % 2  for QPSK
12 % max ED
13 % 64QAM is .5^2*Nt*42 = 10.5*Nt
14 % 16QAM is 1^2*Nt*10 = 10*Nt
15 % QPSK is 2^2*Nt*2 = 8*Nt
16 clipping_ed = .5^2*Nt*42;
```

```
17
18 %-------------------------------------------------------------------------
19 % gain to allow use integer values for the hypotesis
20 %-------------------------------------------------------------------------
21 switch(QAM)
22 case 64
23    z = z * sqrt(Nt*42);
24 case 16
25    z = z * sqrt(Nt*10);
26 case 4
27    z = z * sqrt(Nt*2);
28 end
29
30 %-------------------------------------------------------------------------
31 % K-Best algorithm
32 %-------------------------------------------------------------------------
33 % initalize variables
34 parent_path = complex(zeros(1,Nt), zeros(1,Nt));
35 kbest_path_size = [num_of_kbest*kbest, Nt];
36 kbest_path = complex(zeros(kbest_path_size), zeros(kbest_path_size));
37 kbest_ped = 100000*ones(num_of_kbest*kbest,1);
38
39 % intial number of children node cannot be bigger than QAM
40 child_num = min(num_of_kbest*kbest, QAM);
41
42 % compute inital children nodes
43 [kbest_path(1:child_num,:) kbest_ped(1:child_num,:)] = ...
44    k_best_mcu(Nt, z, R, parent_path, 0, child_num, QAM, idealsort);
45
46 for l=Nt-1:-1:1
47    % K best from previous level become parent nodes
48    parent_path = kbest_path;
49    parent_ped = kbest_ped;
50
51    % clear K best sets for new selection
52    kbest_path = complex(zeros(num_of_kbest*kbest,Nt), ...
          zeros(num_of_kbest*kbest,Nt));
53    kbest_ped = 100000*ones(num_of_kbest*kbest,1);
54
55    for k=1:child_num
56       % children number cannot grow bigger than num_of_kbest*kbest
57       child_num = min(child_num*abest, num_of_kbest*kbest);
58
```

```
59        % compute A best children from parent node k
60        [child_path child_ped] = k_best_mcu(l, z, R, ...
             parent_path(k,:), parent_ped(k), abest, QAM, idealsort);
61
62        % Select K best group and performe K best selection
63        sort_set = mod( k-1, num_of_kbest );
64        kbest_idx = sort_set*kbest+(1:kbest);
65        [kbest_ped(kbest_idx) kbest_path(kbest_idx,:)] = sorter(...
66           child_ped, child_path, ...
67           kbest_ped(kbest_idx), kbest_path(kbest_idx,:), kbest);
68     end
69  end
70
71
72  %-------------------------------------------------------------------------
73  % Bitmetric Computation
74  %-------------------------------------------------------------------------
75  % bitmetric vector inicialization
76  bm.val1 = clipping_ed*ones(1,M*Nt);
77  bm.val0 = clipping_ed*ones(1,M*Nt);
78
79  for k=1:num_of_kbest*kbest
80     bit_vec=symb2bit(kbest_path(k,:), QAM, p); %bit vector
81     pmetric = kbest_ped(k);       % path metric
82     for bitp=0:M*Nt-1            % bit postion
83        if( bit_vec(bitp+1)==1 && pmetric<bm.val1(bitp+1))
84           bm.val1(bitp+1)=pmetric;
85        elseif( bit_vec(bitp+1)==0 && pmetric<bm.val0(bitp+1))
86           bm.val0(bitp+1)=pmetric;
87        end
88     end
89  end
90
91
92  %-------------------------------------------------------------------------
93  % Softbit Generation
94  %-------------------------------------------------------------------------
95  softbit = bm.val0 - bm.val1;
96
97  % power adjustment
98  % The reason for softbit be divied by 42 instead of sqrt(42) is because
99  % the softbits where computed using the squared error
100 switch(QAM)
```

```
101     case 64
102         softbit = softbit/(2*42);
103     case 16
104         softbit = softbit/(2*10);
105     case 4
106         softbit = softbit/(2*2);
107 end
108 softbit = softbit/variance;
```

A.5 Ordenamento e Cálculo do PED dos Nós Filhos

```
1  % Description: Sort and Comput PED values of children nodes
2  % l: search tree level
3  % z: received vector
4  % R: channel matrix
5  % s: parent node partial symbol vector
6  % ped_in: parent node PED
7  % abest: number of admissable children nodes
8  % idealsort: select between ideal or aproximate children node sorting
9  % s_out: children node partial symbol vector
10 % ped: children node PED
11 function [s_out ped] = k_best_mcu(l, z, R, s, ped_in, abest, QAM, ...
       idealsort) %#codegen
12
13 persistent lut
14 if (isempty(lut))
15     if(QAM==64)
16         lut = creat_lut2(64, 16);
17     else
18         lut = creat_lut2(16, 16);
19     end
20 end
21
22 % independent term
23 Nt = length(z);
24 if (l~=Nt)
25     u=z(l)-R(l,l+1:end)*s(1,l+1:end).';
26 else
27     u = z(l);
28 end
29
```

```
30 % find the closest constelation point (best children)
31 R_ll = real( R(l,l) );
32 u_re = real(u);
33 u_im = imag(u);
34 u_re_abs = abs(u_re);
35 u_im_abs = abs(u_im);
36
37 if (QAM == 64)
38    if (u_re_abs>6*R_ll)
39       s_re=7;
40    elseif(u_re_abs>4*R_ll)
41       s_re=5;
42    elseif(u_re_abs>2*R_ll)
43       s_re=3;
44    else
45       s_re=1;
46    end
47 else % 16
48    if (u_re_abs>2*R_ll)
49       s_re=3;
50    else
51       s_re=1;
52    end
53 end
54
55 if (QAM==64)
56    if (u_im_abs>6*R_ll)
57       s_im=7;
58    elseif(u_im_abs>4*R_ll)
59       s_im=5;
60    elseif(u_im_abs>2*R_ll)
61       s_im=3;
62    else
63       s_im=1;
64    end
65 else % 16
66    if (u_im_abs>2*R_ll)
67       s_im=3;
68    else
69       s_im=1;
70    end
71 end
72
```

```
% find the position where the zf solution is located
x_re = u_re_abs - R_ll*s_re;
x_im = u_im_abs - R_ll*s_im;
x_re_abs = abs(x_re);
x_im_abs = abs(x_im);

c = zeros(1,8);

c(1) = x_re < 0;
c(2) = x_im < 0;
c(3) = x_re_abs < x_im_abs;
c(4) = (x_re_abs+x_im_abs) > R_ll;
if(QAM==64)
   c(5) = abs(s_re)==3 || abs(s_re)==7;
   c(6) = abs(s_re)>3;
   c(7) = abs(s_im)==3 || abs(s_im)==7;
   c(8) = abs(s_im)>3;
else
   c(5) = abs(s_re)==3;
   c(6) = abs(s_im)==3;
end
addr = 2.^(0:7)*c';

s_child = lut(addr+1,1:abest).';

% reflect the selected constelation points to the proper quadrant
if(u_re<0)
   s_child = -real(s_child) + 1j*imag(s_child);
end
if(u_im<0)
   s_child = real(s_child) - 1j*imag(s_child);
end

% compute the PED of the abest children nodes
ped = ped_in + abs( u - R_ll.*s_child(1:abest,1) ).^2;
if (idealsort)
  [ped idx] = sort(ped);
  s_child = s_child(idx);
end
s_out = ones(abest,1)*s;
s_out(:,l) = s_child(1:abest,1);
```

A.6 LUT para Ordenamento Aproximado dos Nós Filhos

```
function lut=creat_lut2(QAM, kbest)  %#codegen

% create constelation point vector
% sequence order is -7-7i, -7-5i, ..., +7+5i, +7+7i
if (QAM == 64)
   cnstl = [-7 -5 -3 -1 1 3 5 7];
   cnstl = ones(8,1)*cnstl;
   cnstl = cnstl + 1j*cnstl';
   cnstl = reshape(cnstl,1,64);
else
   cnstl = [-3 -1 1 3 ];
   cnstl = ones(4,1)*cnstl;
   cnstl = cnstl + 1j*cnstl';
   cnstl = reshape(cnstl,1,16);
end
%figure(1)
%plot(cnstl, 'Marker','x','LineStyle','none');


% create LUT points
% sequence order 1+1i, 3+1i, 5+1i, ..., 5+7i, 7+7i
if (QAM==64)
   q_a = [1 3 5 7];
   q_a = ones(4,1)*q_a;
   q_a = q_a' + 1j*q_a;
   q_a = reshape(q_a,1,16);
else
   q_a = [1 3];
   q_a = ones(2,1)*q_a;
   q_a = q_a' + 1j*q_a;
   q_a = reshape(q_a,1,4);
end
%figure(2)
%plot(q_a, 'Marker','x','LineStyle','none');

% creat areas around constelation points
%c0 = x_re < 0;
%c1 = x_im < 0;
%c2 = x_re_abs < x_im_abs;
```

```
%c3 = (x_re_abs+x_im_abs) > R_ll;

% \ 13  / | \ 12  /
% 9 \ / 5 | 4 \ / 8
%   / \   |   / \
% /  1  \ | /  0  \
% ———————————————————
% \  3  / | \  2  /
%   \ /   |   \ /
%11 / \ 7 |6  / \ 10
% /  15 \ | / 14  \
% Centroid of the 16 triangles
q_b_re = [3 -3 3 -3  1 -1 1 -1 5 -5 5 -5 3 -3 3 -3]/6;
q_b_im = [1 1 -1 -1 3 3 -3 -3 3 3 -3 -3 5 5 -5 -5]/6;
q_b = q_b_re + 1j * q_b_im;

% Combine each constelation points with the 16 centroid
q_a_len = length(q_a);
q_b_len = length(q_b);
q = 1j*zeros(1,q_b_len*q_a_len);
for i0 = 0:q_a_len-1
    for i1 = 0:q_b_len-1
        q(q_b_len*i0+i1+1) = q_a(i0+1)+q_b(i1+1);
    end
end
%figure(3)
%plot(q, 'Marker','x','LineStyle','none');


% compute distance
lut = 1j*zeros(length(q), kbest);
e = zeros(1,length(cnstl));
for i0 = 1:length(q)
    for i1 = 1:length(cnstl)
        % compute the distance of all constelation point to q(i0)
        e(i1)=abs(q(i0)-cnstl(i1));
    end
    % get the distance ascending order sequence
    [e_vec idx] = sort(e);
    lut(i0,:)= cnstl(idx(1:kbest));
end
```