HARDWARE ACCELERATOR FOR MIMO WIRELESS SYSTEMS

A Dissertation

by

PANKAJ BHAGAWAT

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2011

Major Subject: Computer Engineering

HARDWARE ACCELERATOR FOR MIMO WIRELESS SYSTEMS

A Dissertation

by

PANKAJ BHAGAWAT

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Gwan Choi
Committee Members,	Jiang Hu
	Rabi Mahapatra
	Jim Ji
Head of Department,	Costas Georghiades

December 2011

Major Subject: Computer Engineering

ABSTRACT

Hardware Accelerator for MIMO Wireless Systems. (December 2011)

Pankaj Bhagawat, B.S., National Insitute of Technology, Tiruchirapally, India;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. Gwan Choi

Ever increasing demand for higher data rates and better Quality of Service (QoS) for a growing number of users requires new transceiver algorithms and architectures to better exploit the available spectrum and to efficiently counter the impairments of the radio channel.

Multiple-Input Multiple-Output (MIMO) communication systems employ multiple antennas at both transmitter and at the receiver to meet the requirements of next-generation wireless systems. It is a promising technology to provide increased data rates while not involving an equivalent increase in the spectral requirements. However, practical implementation of MIMO detectors poses a significant challenge and has been consistently identified as the major bottleneck for realizing the full potential that multiple antenna systems promise. Furthermore, in order to make judicious use of the available bandwidth, the baseband units have to dynamically adapt to different modes (modulation schemes, code rates etc) of operations. Flexibility and high throughput requirements often place conflicting demands on the Very Large Scale Integration (VLSI) system designer. The major focus of this dissertation is to present efficient VLSI architectures for configurable MIMO detectors that can serve as accelerators to enable the realization of next generation wireless devices feasible. Both, hard output and soft output detector architectures are considered.

To My Parents, Brother, Sister-in-Law, Sidharth and Anirudh

ACKNOWLEDGMENTS

My doctoral studies would not have been possible without the support and guidance of my advisor, Dr. Gwan Choi. He consistently helped me to grasp concepts of technical work and bigger things in life. I would like to thank all my committee members, Dr.Mahapatra, Dr.Hu, and Dr. Ji, for valuable inputs and advice. The students of the computer engineering and systems group were very equally helpful to me. I would especially like to express my gratitude to Rajballav Dash, Yoonseoke Yang, Kiran Gunnam, Rohit Singhal, Richeek Arya, Ehsan Rohani, and Euncheol Kim. I would like to thank all my friends in CollegeStation whose company I enjoyed fully.

My family always gave me unconditional love during the course of my stay here. My parents, brother, and sister-in-law were always solidly behind me. I will always cherish the experiences that I had with my two young nephews, Sidharth and Anirudh. Last but not the least, I am thankful to Mr. Roger Lorenzo, Dr. Jay Porter, Dr. Joseph Morgan for constantly helping me enhance my teaching skills.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

Wireless communication has become one of the most important technological achievements of humankind in recent years. Not surprisingly, it has led to huge commercial opportunities for a large number of entrepreneurs. The main reasons for this commercial success is the advent of low cost end-user equipments, worldwide standardization of the technology and affordable services. With the increasing usage of portable computers and mobile phones capable of supporting multimedia content, wireless services have attracted lot of attention.

Due to increasing popularity of multimedia applications the demand for high data rates and better Quality of Service (QoS)is also increasing rapidly. This has led to evolution of wireless standards that can support high data rates. In fact the demand for data rates have been almost doubling every 18 months [1]. Unfortunately, the data rates over a wireless communication channel are limited by the capacity of channel. Further exacerbating the situation is the fact that even this capacity is mostly unachievable owing to a variety of channel impairments. One way to increase the data rate is by increasing the bandwidth of the wireless channel. However, this approach is impractical because spectrum is scarce resource. Hence, meeting the demand for higher data rates and better QoS requires new technologies algorithms and techniques to make judicious use of the the available spectrum.

The journal model is *IEEE Transactions on VLSI Systems*.

A. Approaches to Achieve High Data Rates and Better QoS

Important signal processing techniques developed in recent history that has enabled us to achieve higher data rates with good QoS include:

- Multiple Antenna Systems.
- Transmitter Side Precoding.
- Strong Forward Error Control (FEC) Codes.

Multiple Antenna Systems: A Multiple-input multiple-output (MIMO) communication system [2] uses multiple antennas at the transmitter as well as at the receiver to support the high data rate requirements of next-generation wireless systems. MIMO technology is a promising approach for providing increased data rates without requiring an equivalent increase in the bandwidth requirements. Fig. 1 [3] shows block

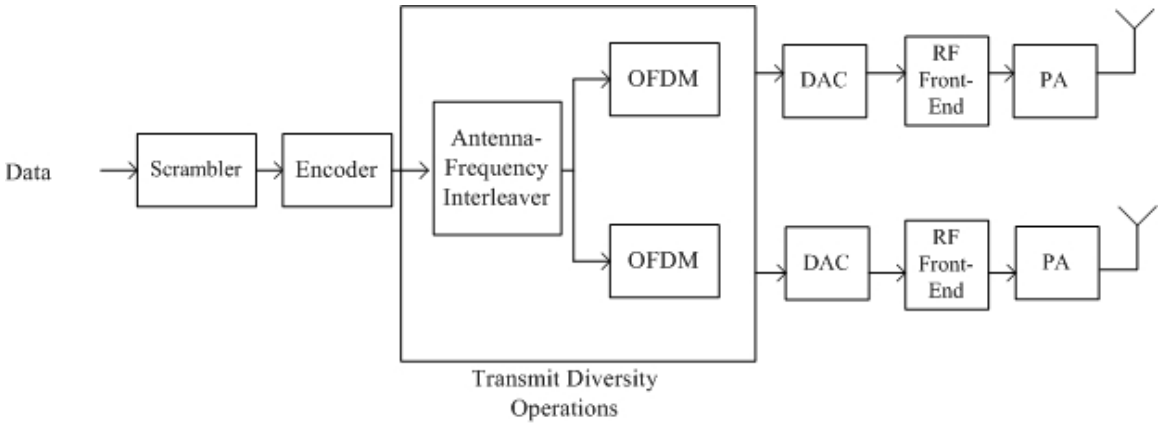

Fig. 1. Wireless Transceiver

diagram of a wireless transceiver for MIMO system. The technique achieves better

data rates and QoS by using three types of gains, namely, array gain, diversity gain, and multiplexing gain[2].

1. Array gain basically means that the receiver side is able to pick up a larger part of the transmitted power. This leads to increased range of the communication system.

2. Diversity gain refers to the improved QoS due to the fact that the receiver gets multiple copies of the same data. This is an effective technique to counter fading.

3. Multiplexing gain is achieved because the transmitter sends independent data streams over the same channel at the same time. This leads to a linear increase in the data rates.

A trade off between these three gains is possible. For instance, space-time codes usually does well to achieve more diversity. Whereas, beamforming technique is good for exploiting array gain. On the other hand spatial multiplexing (SM) uses all available antennas to possibly achieve the highest data rates that can be supported by the channel. For this reason SM based MIMO systems have emerged as a very important scheme to improve the data rates. SM achieves higher data rate by sending independent data from each transmit antenna using same frequency band. For example, in Fig. 2 the system sends two data symbols s_1 and s_2 at the same time and on the same frequency band. The result is that the system can now support two data symbols instead of just one (unlike a Single Input Single Output(SISO) system). Hence, SM-MIMO boosts the data rate proportionately with the number of antennas. However, on the receiver side each antenna sees a weighted superposition (or interference) of all the transmitted symbols. For example Rx_1 sees $s_1h_{11} + s_2h_{21}$ and Rx_2 sees

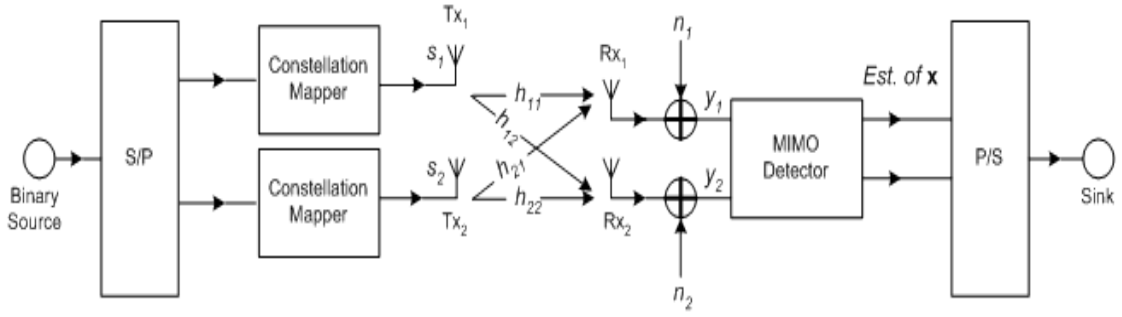

Fig. 2. SM Based MIMO System

$s_1h_{12} + s_2h_{22}$, where the terms h_{11}, h_{12} etc are the channel gains (assumed to be known by the receiver) that distort the transmitted symbols. This interference needs to be removed at the receiver to maintain the data fidelity.

Transmitter Side Precoding: The removal or cancellation of the interference can be done at transmitter as well. This scenario can occur, for example, in a broadcast channel where a single base station intends to communicate with multiple users Fig. 3. Each user ends up receiving not only its own data, but also sees the data symbols intended for other users as unknown interference. This interference, if left as is can cause a significant degradation in the system capacity. However, this interference can be removed by precanceling it at the transmitter. One way to precancel the interference is by using Dirty Paper Precoding (DPC) scheme [4]. Besides broadcast channels, DPC finds applications in numerous other situations; inter-symbol interference (ISI) channels, cooperative networks, and digital watermarking to name a few.

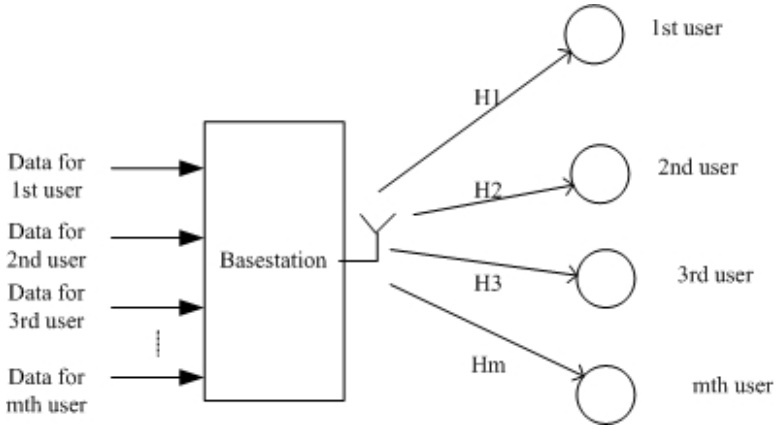

Fig. 3. Multi User Scenario in a Broadcast Channel

Low Density Parity Check (LDPC) Codes: LDPC codes [5] are known to achieve information rates very close to the capacity of the channel when iteratively decoded. Other competing FEC codes such as Turbo codes [6] are also know to be performing close to the capacity. However, the decoders for LDPC codes have arithmetic computation requirements that are an order of magnitude less than Turbo decoders for similar bit-error rate (BER) performance. Algorithms for decoding LDPC codes also have the advantage of being inherently parallel. In principle, this permits the use of multiple parallel processing elements to increase the throughput of the decoder. For these reasons the LDPC codes are increasingly finding applications in upcoming wireless standards.

B. Hardware Implementation Perspective

Fig. 4 depicts how the complexity of algorithms vs. processing power will likely evolve with newer generation of wireless standards. To further improve usage of the

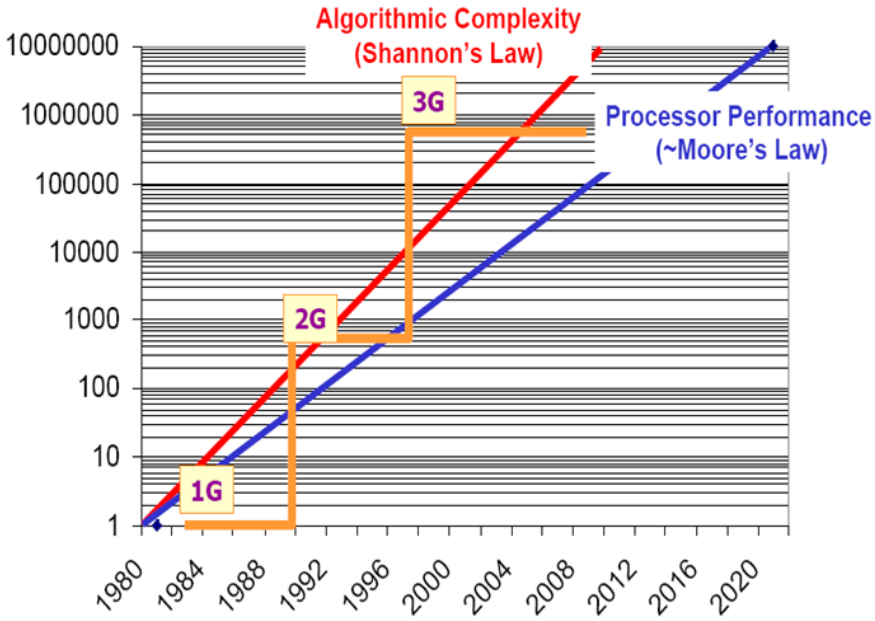

Fig. 4. Algorithmic Complexity Vs. Available Processing Power [7]

spectrum the transmitter at the basestation the changes the transmission modes (e.g. modulation schemes, number of antennas, code rates) on the fly. This entails that the baseband hardware must be flexible enough to support runtime configuration.

The general purpose processors allow the designers maximum flexibility with lowest efficiency (in terms of throughput/area and power consumption). Whereas the ASICs offer most efficient solutions but is not flexible. DSPs, Application Specific Instruction Processors (ASIP) and FPGA offer a mix of efficiency and flexibility Fig. 5.

Because of increased availability of silicon area it is now economical to build a network of function specific processors that can work in unison on a same chip. Het-

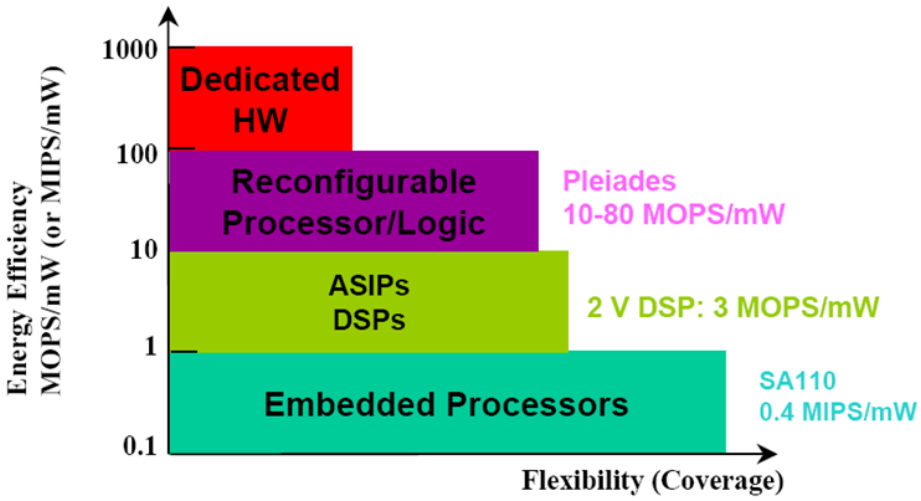

Fig. 5. Architectural Flexibility Vs. Hardware Efficiency [8]

erogeneous MultiProcessor System on Chip (MPSoC), which have multiple function hardware accelerators governed by a central processor, is becoming popular approach for realizing future wireless systems. One possible MPSoC for baseband processing is depicted in Fig. 6. Because a wide range of functional units can be integrated on a single chip it is now possible to realize a highly flexible solution which is far more efficient than DSP or ASIPs. This approach also leads to improved productivity of engineers as the design process is more decentralized [9].

In a large system on chip power considerations are important not just because of battery life concerns but also due to concerns about the chip reliability. Smaller transistors leads to higher density and the circuit can be run at higher speeds. Although smaller transistors consume lesser power, the increased density and speed leads to higher thermal density which can cause the chip to fail. Another important design

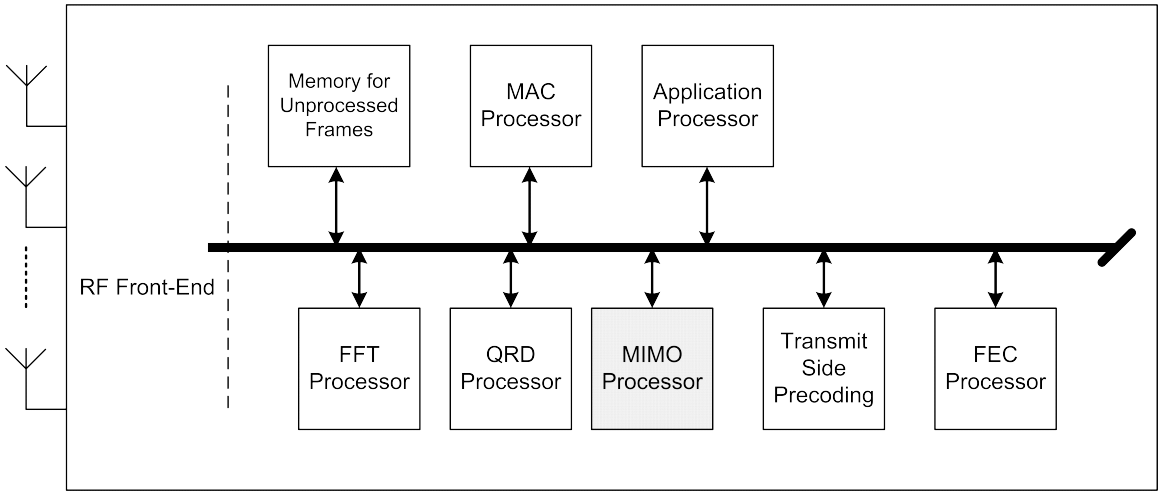

Fig. 6. Heterogeneous MultiProcessor System Approach

criteria from a power perspective is the energy efficiency (J/bit) of the design. The energy efficiency is important for the battery life of devices, whereas, the thermal distribution of the chip is critical for faultless functioning of the chip.

C. Motivation and Contributions

The implementation complexity of the detector for MIMO systems has been consistently identified as the major bottleneck for realizing the full potential that multiple antenna systems promise. For this reason this dissertation deals with VLSI architectures for MIMO detectors.

Implementation Challenges for MIMO Detectors: The practical implementation of MIMO systems is constrained by the high complexity of the detection algorithms at the receiver. This is primarily due to the need for complex arithmetic and the

complicated data flow requirements of the underlying algorithms. The MIMO detectors can be broadly classified into two classes: 1) Linear detectors, and 2) Non-linear detectors. Linear detectors take a suboptimal approach in demapping the received symbols. They can be implemented in a systolic like manner with constant throughput and good scalability w.r.t. the antenna size and modulation scheme. The detection algorithms in this domain are highly suitable for practical implementation and will likely meet the exacting throughput requirements of next generation wireless systems. Linear detectors, however, suffer from poor BER performance which makes them unattractive from wireless system designer's point of view. On the other hand non-linear detectors have complicated data-flow, variable throughput, and unfavorable scalability. They, however, provide close to optimal BER performance and are hence more suitable from algorithmic point of view. The need for close to optimal BER performance and flexibility requirements of the future wireless standards makes the detector design even more challenging. This is because flexibility and throughput often place conflicting demands on the designer. The contributions of this dissertation are as follows:

Architectural evaluation of hard decision detector: We consider three different architectures for MIMO detectors. Three architectures considered are: 1) Sequential 2) Staggered and 3) Parallel. We present detailed hardware architectures and their ASIC implementation. We compare the architectural figures of merit of Sequential and Staggered architectures under two situations, first, when the detection process (carried out in multi-core setup) has variable runtime and second when a fixed runtime budget is enforced via scheduling. We show that enforcing/scheduling runtime budget per block of data causes deterioration in the efficiency of the architectures. The staggered architecture achieves significantly higher throughput than the sequential one, and half the throughput of the parallel approach (using an area of much less

than half of the parallel implementation). Part of this work can be found in [10].

Runtime configurable hard detector: We developed a flexible architecture capable of configuring itself to process QPSK, 16-QAM, and 64-QAM modulation schemes. This architecture is further modified to support variable number of antennas, i.e., for 2x2, 3x3, and 4x4 MIMO systems. The architecture has a systolic like flow with excellent scalability. The degree of pipelining can be changed with little redesign effort. Another important feature of the architecture is that it does not incur any latency for configuring itself. Moreover, for a given operating mode its throughput is completely predictable. These qualities makes it much easier to integrate in a wireless MPSoC. Additionally, the detector provides close to optimal Maximum Likelihood(ML) solution leading to close to the best attainable BER [11].

We also present a constraint aware architecture space exploration for the 802.11n standard [12]. We carry out extensive architectural space exploration to address the issues of power consumption, area for the configurable detectors. Ultimately, we come up with two designs that target low area and low power respectively while meeting the demands of the baseband time constraints.

Runtime configurable soft decision detector: Soft decision basically means that the detector provides the FEC decoder(like LDPC) with the confidence of its decisions. This leads to significant improvement in the BER performance of the whole system. The process of getting soft decisions is even more complicated than getting hard decisions. We present a configurable detector architecture for in a high order(4x4) MIMO wireless system. It is a customized architecture that provides soft values with systolic-like data and control flow. The detector is able to switch between three different modulation schemes (QPSK, 16-QAM, and 64-QAM) without any configuration latency. Multiple detector cores can be stacked to achieve very high throughput. We also derive a lower complexity algorithm soft decision algorithm. We

then present a systolic like for this algorithm. A Single detector core achieves constant throughput of 215Mbps (even for a 64QAM 4x4 MIMO system).

D. Organization of Thesis

The organization of the dissertation is as follows:

Chapter II provides background on the MIMO based wireless systems. We explain how various MIMO techniques achieve higher data rates and better QoS. We also describe basic algorithms used for detection in a MIMO system. We describe the system model that will used in this work. In Chapter III we present comparative study of three architectures and their ASIC implementation results. First half of the chapter provides description of the architectures and second half is devoted to their evaluation in practical scenarios. Chapter IV deals with the configurable MIMO detector with close to optimal error rate performance. We present the detailed architecture and its FPGA and ASIC implementation results. This chapter is concluded by considering a design case study for detectors in a 802.11n WLAN system that require on the fly configuration. Chapter V deals with systolic like architectures for soft output MIMO detection. We present a detector that can be configured to support three different modulation schemes. In this chapter we present a lower complexity algorithm and corresponding architecture for soft detection. Finally, Chapter VI concludes the dissertation.

CHAPTER II

BACKGROUND

This chapter begins with providing a broad perspective of a MIMO wireless system. We explain various techniques available to extract better diversity and maintain QoS. We also explain the MIMO system model that is under consideration and introduces the notations that will be used. A brief description of the fundamental algorithmic choices for MIMO detection is also provided. System and hardware design considerations are discussed.

A. MIMO Wireless Systems: An Overview

The increased number of antennas in a MIMO wireless system leads to increased spatial diversity. Diversity basically means that independent versions of signals are transmitted so that the probability of all of them being corrupt is reduced. Spatial diversity means that these multiple copies of signals are transmitted from elements in space i.e. by using multiple antennas.

Techniques such as Maximum Ratio Combining [13] or Space-Time-Block-Coding [13] allows for using multiple antennas at either the transmit or the receive side. In the beginning the aforementioned techniques provided only receive or transmit diversity. These methods successfully achieved diversity gain by providing multiple copies of same data leading to improved BER. Diversity gain is quantified as a ratio of the SNR with diversity to SNR without diversity. This is reflected as the slope of the BER curve in a log-scale. It has been shown that the the maximum diversity gain is N, if N independent copies (using N antennas) of a given signal are transmitted [14]. These techniques basically use diversity to combat the effects of fading.

However, using multiple antennas at the receive and transmit allows us to ob-

tain spatial multiplexing gain . Spatial multiplexing gain means that if M transmit antennas and N receive antennas are used, recovery of min(M,N) different signals is possible. Important implication of this is that higher data rates can be achieved without requiring additional bandwidth. A comparison of some schemes at the algorithmic level can be found in [15] and [9]. We reproduce it in Fig. 7.

Code	Diversity Gain	Max Rate [sym/s/Hz]
Alamouti STBC	$2M$	1
Orthogonal Designs	NM	1
Linear Dispersion	$< NM$	$\approx\leq \min(M,N)$
TR-STBC	NML	<1
STBC + Frequency domain Processing	NML	<1
Tarokh STTC	NM	1
V-BLAST	$\leq M$	$\min(M,N)$
H-BLAST	$\leq M$	$\approx\leq \min(M,N)$
D-BLAST	$\leq M$	$\approx\leq \min(M,N)$

Fig. 7. Performance of Basic MIMO Techniques, [15], [9]

B. MIMO System Model

In this section we start by explaining the MIMO-OFDM model. Orthogonal Frequency Division Multiplexing (OFDM) technique is used in wideband communication systems to combat the multipath (which leads to frequency selective channel) fading. Wideband MIMO systems Fig. 8 can be considered equivalent to a set of narrowband MIMO systems Fig. 9 if OFDM is also used. This equivalence simplifies the analysis of the system and make it easier to design receivers.

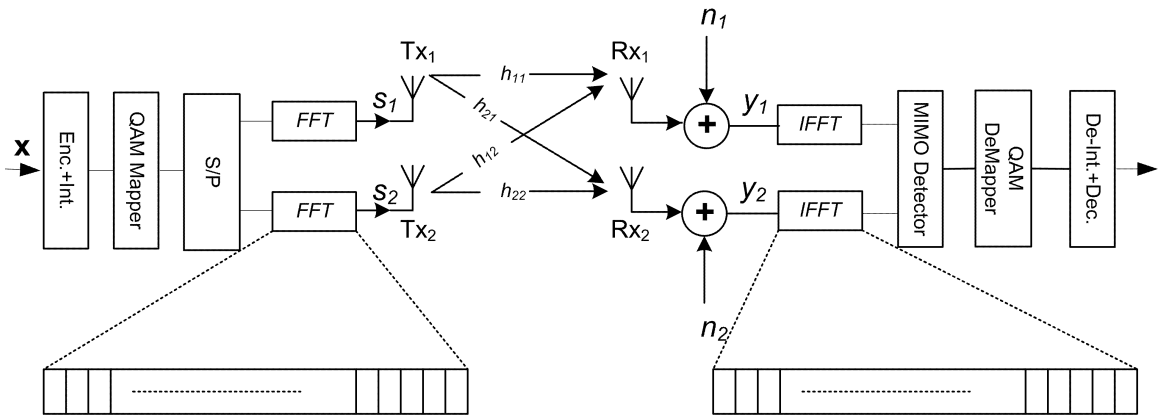

Fig. 8. Wideband MIMO (OFDM) System

Hence, we will consider a simple narrowband system model as the basis for the discussions in this work. In the system under consideration, as shown in Fig. 9, the term M_T and M_R denotes the number of transmit antennas and the number of receive antennas respectively. In this work we will assume M_R=M_T=4 unless otherwise specified.

MIMO Channel: The equivalent baseband model of the MIMO wireless channel can be expressed as the following relation.

$$\mathbf{y} = \mathbf{Hs} + \mathbf{n} \tag{2.1}$$

where $\mathbf{y}$=$[y_1, y_2, ..., y_{M_R}]^T$ is a $M_R \times 1$ received vector, $\mathbf{s}$=$[s_1, s_2, ..., s_{M_T}]^T$ is $M_T \times 1$ transmitted vector and will be referred to as a MIMO symbol in the sequel, $\mathbf{n}$ is $M_R \times 1$ zero mean complex Gaussian noise vector, and $\mathbf{H}$ is a $M_R \times M_T$-dimensional complex matrix. The $(i, j)^{th}$ element, h_{ij}, of the matrix $\mathbf{H}$ denotes the complex channel gain from the j^{th} transmit antenna to the i^{th} receive antenna. For the simulations in this

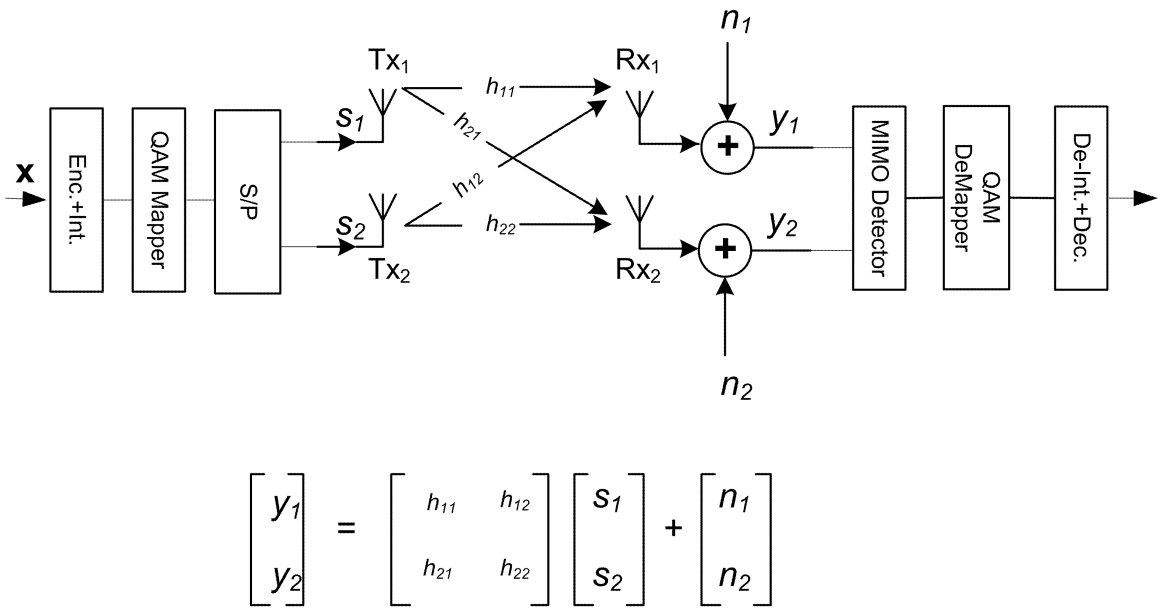

Fig. 9. Equivalent Narrowband MIMO System

work, an i.i.d. Rayleigh fading channel model is assumed. Hence, the entries of **H** are chosen independently as zero mean complex Gaussian random variables.

- **Transmitter:** Each entry s_i $(i = 1, 2, ..M_T)$ in the MIMO symbol **s**, is drawn from a set Ω of cardinality η. In general the members of the set Ω are complex numbers with their real and imaginary parts of the form $\{-\sqrt{\eta} + (2k-1)\}$ where k=1,2,...$\sqrt{\eta}$. This scheme is called as a η-ary Quadrature Amplitude Modulation (QAM) and s_i are called QAM symbols. The QAM symbols are generated by mapping(using look-up table, for instance) a group (of size $\log_2 \eta$) of binary bits onto a symbol(a complex number) from Ω.

- **Receiver:** The received signal vector **y** is picked up by the M_R antennas at

the receiver. The objective of the MIMO detector is to estimate $\hat{\mathbf{s}}$ of $\mathbf{s}$ based on the the observation of $\mathbf{y}$ along with the knowledge of $\mathbf{H}$. The knowledge of $\mathbf{H}$ can be obtained by using some training (e.g. using pilot symbols).

C. Important Available Algorithms

In this section important algorithms that are available for MIMO detection are described.

1. Linear Detection

Linear detectors attempt to find the solution by multiplying the inverted channel matrix $\mathbf{H}$ with the received vector $\mathbf{y}$. More specifically, the estimate $\hat{s}$ of the transmitted vector $\mathbf{s}$ is computed by:

$$\widehat{\mathbf{s}} = \mathbf{G}\mathbf{y} \tag{2.2}$$

where $\mathbf{G}$ is an equalization matrix. However, equation (2.2) provides the *unconstrained* estimate. This basically means that the result of equation (2.2) is a complex number and not a QAM symbol(QAM symbols are *constrained* to a limited set of constellation points). To get to the constrained solution the detector carries out a *slicing* operation. In slicing operation, the detector compares the real and imaginary parts of $\hat{s}$ from equation (2.2) with the appropriate thresholds to find out the closest QAM symbol.

The complexity of this algorithm is dominated by computation of the equalization matrix. In packet based systems, where data is transmitted in packets (consisting of several MIMO symbols) that experience constant $\mathbf{H}$. This means that the equalization matrix needs to be computed only once per packet. This not a big problem if packet size is large. However, if the packet size is small then the overall complexity will

increase as the receiver has to invert the matrix more often. The detection of all the MIMO symbols in a packet can then use same estimation matrix and the detector only has to carry out the slicing operation (which relatively simple operation).

There are two important algorithms in this class of detection scheme: Zero Forcing (ZF), and Minimum Mean Squared Error (MMSE). Basically both these algorithms make use of channel matrix inversion and vector multiplication with the received vector $\mathbf{y}$. The ZF detector uses the pseudo-inverse of $\mathbf{H}$ given by:

$$\mathbf{G} = (\mathbf{H}^H)^{-1}\mathbf{H}^H \tag{2.3}$$

where $\mathbf{H}^H$, is the hermitian of the matrix $\mathbf{H}$. The corresponding ZF estimate $\hat{\mathbf{s}}_{ZF}$ of the transmitted vector $\mathbf{s}$ is then given by:

$$\hat{\mathbf{s}_{ZF}} = (\mathbf{H}^H)^{-1}\mathbf{H}^H\mathbf{y} = \mathbf{s} + \tilde{\mathbf{n}}_{ZF} \tag{2.4}$$

The effective channel matrix $\tilde{\mathbf{H}}$ is now an identity matrix $\mathbf{I}$. This means that the interference from the other streams is completely eliminated as desired. However, the major disadvantage of this algorithm is that the term $\tilde{\mathbf{n}}_{ZF}$ leads to noise enhancement.

In case of MMSE detection the estimator matrix $\mathbf{G}$ is mathematically expressed as:

$$\mathbf{G} = (\mathbf{H}^H + N_0\mathbf{I})^{-1}\mathbf{H}^H \tag{2.5}$$

This algorithm essentially tries to minimize the error $\mathrm{E}\|\mathbf{s}_{ZF} - \mathbf{s}\|^2$. This algorithm tries to strike a balance between interference and noise enhancement. The problem with this algorithm is that accurate estimate of N_0 is needed.

These detectors are low complexity and their VLSI architectures have highly regular data and control flow. This makes it very attractive in terms of meeting the throughput and power consumption requirements.However, both algorithms suffer

from poor BER because of noise enhancements and remaining interference. Moreover, the process of channel inversion is prone to numerical errors. Therefore, the results from these detectors are found to be severely sub-optimal BER performance and is unable to exploit the full potential of a MIMO wireless system.

2. Successive Interference Cancellation

SIC is a recursive algorithm which provides a BER performance that is between the linear detectors and optimal detectors. The complexity of SIC detectors is also increased in comparison to linear detectors, but is far less than the brute force optimal detector. SIC detector proceeds in serial fashion by first detecting a symbol then subtracting an estimate of the detected symbol's contribution to the received vector. Thus once a symbol is detected it is considered as interference for other symbols and is canceled out.

For mathematical description of SIC, consider the model given by equation (2.1). The algorithm first initializes $y^1 = y$ and defines $H^i = [h_i, h_{i+1}, ...h_{M_T}]$, where elements h_i denotes the i^{th} column of the matrix $\mathbf{H}$. The estimation matrix for H^i is denoted by $G^(i)$. Beginning with $i = 1$, SIC goes through following steps until $i = M_T$:

$$\hat{x}_i = \mathbf{G}_i^{(i)} y^{(i)} \tag{2.6}$$

$$\hat{s}_i = Q(\hat{x}_i) \tag{2.7}$$

$$\mathbf{y}^{(i+1)} = \mathbf{y}^{(i)} - \hat{s}_i \mathbf{h}_i \tag{2.8}$$

where Q(.) is the slicing function. The vectors $\mathbf{G}_i^{(i)}$ are called nulling vectors and are simply the i_{th} row of the corresponding estimator matrix $\mathbf{G}$. The estimator matrix $\mathbf{G}$ can be one of the two as described earlier. This leads to ZF-SIC or MMSE-SIC depending on which estimator matrix is chosen.

SIC algorithms do not completely exploit the available diversity in the system. Moreover, they suffer from error propagation problems. This happens when an error is made in detecting the first symbol. Due to recursive nature of the algorithm, this error in decision is propagated to other symbols in the vector. Though better than the linear detectors, this algorithm is still sub-optimal with increased implementation complexity and increased power consumption.

3. Maximum Likelihood Detection

It has been shown that the optimal maximum likelihood (ML) estimate $\hat{\mathbf{s}}_{ml}$ of $\mathbf{s}$ is given by equation (2.9):

$$\hat{\mathbf{s}}_{ml} = arg \min_{\mathbf{S} \in \Omega^{M_T}} \|\mathbf{y} - \mathbf{Hs}\|^2 \tag{2.9}$$

Basically the above equation means that the optimal estimate $\hat{\mathbf{s}}$ of the transmitted vector $\hat{\mathbf{s}}$ is the one that minimizes the euclidean distance between the received vector $\mathbf{y}$ and the *transformed* transmitted vector $\mathbf{Hs}$. Straightforward way to solve (2.9) is by computing the distances for all possible combinations of $\mathbf{Hs}$ and declaring the combination that gives minimum distance as the estimate $\hat{\mathbf{s}}$.

An important advantage of this algorithm is that there is no matrix inversion involved. The absence of matrix inversion greatly aids the hardware implementation of the algorithm. Major disadvantage of this algorithm is that the search space grows very quickly with the modulation scheme (i.e. η), and the number of transmit antennas (M_T). If η-ary modulation scheme is used in a $M_T \times M_R$ MIMO system the total possible combinations for $\mathbf{s}$ is η^{M_T}. This search space quickly becomes too large to be processed in real-time in today's computing technologies.

4. Sphere Decoding

This is also algorithmically optimal detection method. It constraints the search space for transmitted symbol vector. This algorithm has shown promise at being able to handle strong algorithmic requirements (BER Performance) with reduced design complexity and power requirements. One way to efficiently search for $\hat{\mathbf{s}}_{ml}$ is to evaluate only a small subset of all the possible vectors. This can be achieved by noting that $\mathbf{H}$ can be triangularized using QR decomposition: $\mathbf{H} = \mathbf{QR}$, where, $\mathbf{R}$ is an upper triangular matrix, and $\mathbf{Q}^H$ is the Hermitian of a unitary matrix $\mathbf{Q}$. Hence, the cost function given by equation (2.9) can now be rewritten as [16],

$$\hat{\mathbf{s}} = \|\mathbf{y} - \mathbf{Hs}\|^2 = \|\hat{\mathbf{y}} - \mathbf{Rs}\|^2, and\ \hat{\mathbf{y}} = \mathbf{Q^H y} \tag{2.10}$$

Equation 5.3 can be further expanded as shown in equations (5.4),(5.5),and (5.6).

$$d_i(s^{(i)}) = d_{i+1}(s^{(i+1)}) + |e_i(s^{(i)})|^2 \tag{2.11}$$

$$|e_i(s^{(i)})|^2 = |c_{i+1}(s^{(i+1)}) - R_{ii}.s_i|^2 \tag{2.12}$$

$$c_{i+1}(s^{(i+1)}) = \hat{y}_i - \sum_{j=i+1}^{M_T} R_{ij}.s_j \tag{2.13}$$

The quantity $|e_i(s^{(i)})|^2$ will be called the Incremental Euclidean Distance (IED), and the term $d_i(s^{(i)})$ will be called Partial Euclidean Distance (PED) for $i > 1$, and Euclidean Distance (ED) for $i = 1$. The fact that $\mathbf{R}$ is upper-triangular ensures that each term on LHS of equations (5.4),(5.5),and (5.6) depends only on the current level i, and the history of the path to reach that level(note that in equations (5.6), the index j runs from $i + 1$ to M_T). Because the PED's depend only on $s^{(i+1)}$, they can be associated with corresponding nodes in a η-*ary* tree with M_T levels. The computation of the terms $d_1(s^{(1)})$ can then be interpreted as a traversal of the tree from the root($i = M_T$) to the leaf ($i = 1$)corresponding to $\mathbf{s}$. The estimate can now

be obtained by searching the leaf with the smallest ED and returning the path from the top level to that leaf as $\hat{\mathbf{s}}_{ml}$. The complexity of this tree search can be greatly reduced by noting that IEDs are always positive, and hence if the PED of a node exceeds a predefined threshold (called radius) the subtree rooted at that node can be excluded from further search. This approach is commonly known as sphere decoding [16].

CHAPTER III

VLSI ARCHITECTURES FOR MIMO DETECTION:A COMPARATIVE ANALYSIS

In this chapter we present ASIC implementations of three different architectures for MIMO detectors. Three architectures considered are: 1) Sequential 2) Staggered and 3) Parallel. We also compare the architectural figures of merit of sequential and staggered architectures under two situations, first, when the detection process (carried out in multi-core setup) has variable runtime and second when a fixed runtime budget is enforced per block of data. We show that enforcing/scheduling runtime budget per block of data causes deterioration in the efficiency of the architectures. The parallel architecture provides the highest throughput, albeit at a much higher cost. The sequential architecture provides the lowest throughput at the lowest cost. The throughput and the cost of staggered design lies in between parallel and sequential architectures.

Although low complexity algorithms such as VBLAST [17] are suitable for hardware implementation they suffer from significant degradation in their error rate performance. A family of algorithms under active consideration relies on a tree based search that is more complex but provide excellent BER. Two of the most notable approaches in this family are the Sphere Decoding (SD) algorithm [18], which is a Depth First Search (DFS) based algorithm, and the K-best algorithm which is a Breadth First Search (BFS) based algorithm [19]. Authors in [20] provide two ASIC implementations of the SD algorithm. Implementation of K-best algorithm has been reported in [21]. The SD implementation in [20] takes a sequential approach, whereas in [19],[21] the authors take a highly parallel approach to MIMO detection. Both approaches have their merits and demerits. For instance, being serial in nature the

implementation of the SD algorithm requires smaller silicon area than the K-best algorithm, but has highly variable throughput. Hence, it is difficult to integrate it into the overall communication system. On the other hand, the K-best algorithm provides a fixed throughput while utilizing a much higher silicon area due to extensive sorting operations [21]. In general, the tree based algorithms offer a good trade-off between performance and complexity [22]. A low complexity algorithm has been reported in [23]. This algorithm, called the Fixed-Throughput Sphere Decoder (FSD), relies on a special ordering of the columns of the channel matrix, which results in a significantly reduced search space. Further simplification is proposed in [24], that simplifies the complexity associated with the column ordering, this algorithm has been names Conditionally Ordered Successive Interference Cancellation (COSIC). The resulting implementation has a very simple data path and is able to achieve close to Maximum Likelihood (ML) solution. The authors in [23] have presented an implementation of fully parallel architecture (BFS based) to implement the algorithm, which provides very high and fixed throughput. However, a fully parallel approach consumes large area and incurs many redundant computations. A sequential architecture [24], in conjunction with radius reduction technique reduces redundant computations and consumes less silicon real estate, but has lower throughput than the parallel scheme. In [20], ASIC implementations of different architectures have been compared.

A. MIMO Detection

1. Model for Spatial Multiplexing

For convenience we will reproduce the information from chapter II. The baseband system model for a MIMO system with M_T transmit and M_R receive antennas can

be given by equation (3.1) [25].

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \tag{3.1}$$

where $\mathbf{y}$=$[y_1, y_2, ..., y_{M_R}]^T$ is a $M_R \times 1$ received vector, $\mathbf{s}$=$[s_1, s_2, ..., s_{M_T}]^T$ is $M_T \times 1$ transmitted vector (will be referred to as a MIMO symbol in the sequel), $\mathbf{n}$ is $M_R \times 1$ zero mean complex Gaussian noise vector, and $\mathbf{H}$ is a $M_R \times M_T$-dimensional complex matrix. The $(i, j)^{th}$ element, h_{ij}, of the matrix $\mathbf{H}$ denotes the complex channel gain from the j^{th} transmit antenna to the i^{th} receive antenna.

Each entry s_i $(i = 1, 2, ..M_T)$ in the MIMO symbol $\mathbf{s}$, is drawn from a set Ω of cardinality η. In general the members of the set Ω are complex numbers with their real and imaginary parts of the form $\{-\sqrt{\eta} + (2k - 1)\}$ where k=1,2,...$\sqrt{\eta}$. This scheme is called as a η-ary Quadrature Amplitude Modulation (QAM) and s_i are called QAM symbols. The QAM symbols are generated by mapping a group (of size $\log_2 \eta$) of binary bits onto a symbol(a complex number) from the set Ω.

The objective of the MIMO detector is to estimate $\hat{\mathbf{s}}$ of $\mathbf{s}$ based on the the observation of $\mathbf{y}$ along with the knowledge of $\mathbf{H}$. It has been shown that the optimal or the ML estimate $\hat{\mathbf{s}}_{ml}$ of $\mathbf{s}$ is given by equation (3.2) [25]:

$$\hat{\mathbf{s}}_{ml} = arg \min_{\mathbf{s} \in \Omega^{M_T}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \tag{3.2}$$

A straightforward way to compute $\hat{\mathbf{s}}_{ml}$ is to evaluate equation (3.2) for all possible $\mathbf{s}$ and pick the one that minimizes R.H.S of equation (3.2). This exhaustive search approach is impractical even for moderately sized MIMO systems. For a 4x4 16-QAM MIMO system this approach would require evaluating 16^4 combinations of $\mathbf{s}$ before a decision on $\hat{\mathbf{s}}_{ml}$ can be made.

a. Sphere Decoding

One way to efficiently search for $\hat{\mathbf{s}}_{ml}$ is to evaluate only a small subset of all the possible vectors. The sphere decoder achieves this by searching for the potential candidates for $\hat{\mathbf{s}}_{ml}$ within a *sphere* of radius r. This can be mathematically expressed as:

$$\|\mathbf{y} - \mathbf{Hs}\|^2 \leq r \tag{3.3}$$

The Sphere Decoding (SD) algorithm works by transforming the original problem as a tree search, wherein any branch of the tree with a path metric greater than a predefined threshold (or radius r) is pruned. This helps in reducing the search space greatly. The SD algorithm tends to have exponential complexity at low SNRs and polynomial complexity at higher SNRs [26], [27]. This algorithm, however, does provide the optimal (ML) estimate $\hat{\mathbf{s}}$ of $\mathbf{s}$.

The hardware architecture and its implementation for SD has been studied extensively in [16]. One of the major problems with SD is that it takes highly variable amount of time to converge to a solution. This attribute is undesirable in practical situation, as it makes the integration of the detector hardware very difficult in an overall system. Several algorithms have been proposed to reduce (or completely get rid) of this variability. Popular among them include K-Best, FSD/COSIC. These algorithms, however, do not provide the exact ML solution. But as shown in aforementioned references, their BER performance is very close to the optimal. This makes them very attractive vis-a-vis hardware implementation.

b. K-Best and FSD/COSIC Algorithms

K-Best differs from sphere decoder algorithm in that it retains a fixed number(K) of branches (or nodes) at each level of the tree. The path with least cumulative metric

at the leaf level is declared as the estimate $\hat{\mathbf{s}}$. This non-recursive nature of K-best has important implications for its hardware implementation. Major advantage of K-best is its constant throughput and ease of designing pipelined architecture (owing to its non-recursive nature). However, this algorithm tends hit error floor as SNR increases [28]. This loss can be partially recovered by increasing the value of K for higher SNRs. Also, K-best involves sorting requiring large amount of data flow which is generally very power hungry. Some sort free versions of K-best algorithms have been proposed recently [29],[30].

The FSD algorithm obviates the need for sorting. It is shown in [31] that this algorithm also achieves full diversity. Moreover, its Bit Error Rate (BER) performance is very close to the ML performance. The COSIC algorithm differs from FSD only in its preprocessing stage. Empirical studies suggest that COSIC algorithm also performs close to ML, and performs well in higher SNR regimes as well (unlike K-best algorithm). Moreover, FSD/COSIC both can provide fixed throughput depending on how the hardware architecture is designed. FSD/COSIC offers more avenues for pipelining and parallelism than SD or even K-best. Hence, we choose to use COSIC algorithm for our detector.

B. Hardware Architectures

In this section we begin by reviewing the mathematics of the MIMO detection problem. This is followed by a brief review of FSD/COSIC algorithm, and then we present the sequential, fully serial, and parallel architectures to implement it. Equation (3.2) can be simplified by triangularizing $\mathbf{H}$ using QR decomposition: $\mathbf{H} = \mathbf{QR}$, where, $\mathbf{R}$ is an upper triangular matrix, and $\mathbf{Q}^H$ is the Hermitian of a unitary matrix $\mathbf{Q}$. Hence, the cost function given by equation (3.2) can now be rewritten as shown in

(3.4) [22],

$$\hat{\mathbf{s}} = \|\mathbf{y} - \mathbf{Hs}\|^2 = \|\hat{\mathbf{y}} - \mathbf{Rs}\|^2, and\ \hat{\mathbf{y}} = \mathbf{Q^H y} \tag{3.4}$$

Equation (3.4) can be further expanded as shown in (3.5)-(3.7).

$$d_i(s^{(i)}) = d_{i+1}(s^{(i+1)}) + |e_i(s^{(i)})|^2 \tag{3.5}$$

$$|e_i(s^{(i)})|^2 = |c_{i+1}(s^{(i+1)}) - R_{ii}.s_i|^2 \tag{3.6}$$

$$c_{i+1}(s^{(i+1)}) = \hat{y}_i - \sum_{j=i+1}^{M_T} R_{ij}.s_j \tag{3.7}$$

The quantity $d_i(s^{(i)})$ is called the cumulative metric. The quantity $|e_i(s^{(i)})|^2$ is called the incremental metric. The vector $s^{(i)} = [s_i, s_{i+1}, ..., s_{M_T}]$ in equation (3.5)-(3.7) denotes a partial vector symbol candidate. Term $d_i(s^{(i)})$ will be called Partial Euclidean Distance (PD) for $i > 1$, and Euclidean Distance (D) for $i = 1$. Because the PD's depend only on $s^{(i+1)}$, they can be associated with corresponding nodes in a η-ary tree with M_T levels. Alternatively, the computation of the terms $d_i(s^{(i)})$ can be interpreted as a traversal of the tree from the root to the leaf corresponding to $\mathbf{s}$, where $(i = 1)$ corresponds to leaf nodes. The estimate can now be obtained by searching the leaf with smallest D and returning the path from the top level $(i = M_T)$ to that leaf as $\mathbf{s}$. The PD's and D's in equation (3.5) are equivalently referred to as the node's metric in the sequel. To reduce the search space aggressive pruning of the tree is needed. Sphere decoding does this by pruning a branch of the tree whenever the PD of a node on the branch exceeds a certain limit (radius r). Appropriately setting the radius r restricts the number of nodes visited in the search, since $d_i(s^{(i)})$ in equation (3.5) is monotonically increasing. Hence any node that has a PD more than r need not be considered any further. This is called the Sphere Criterion or SC [20]. This effectively prunes the whole sub-tree rooted at that node, thus greatly reducing the search space. Fixing the value of r is in general difficult. If r is too small then we

may have to restart the decoder as no point may be found. Similarly if r is too large the decoder may visit a very large number of nodes leading to excessive latency. To counter this problem a scheme known as radius reduction [25] is used, wherein the initial value of r is set to infinity, and is updated whenever the D of a newly evaluated leaf is less than the current value of the radius. This scheme works best if the nodes are processed in an ascending order of their PD's (i.e. the most promising candidates are processed first). This scheme is called Schnorr-Euchner (SE) ordering [20].

1. FSD/COSIC Algorithm

The FSD algorithm is essentially based on reordering the columns of the **H** matrix such that the tree search is simplified [22]. The idea is to evaluate all candidate QAM symbols for the stream with lowest SNR and only the best child of parents for subsequent levels. The reason for this is that the weakest stream has largest probability of error and because all candidates are considered this error probability does not influence the decision later. For subsequent levels, this algorithm follows strongest stream first philosophy to achieve best possible BER performance. On a tree this translates to computing PD's for all nodes at the top level, but only the best child (child with least PD/D) is considered for lower levels. For a 4x4 system with 16 QAM, this entails computing PD's/D's for a maximum of 16x4=64 nodes, significantly reducing the search space. Fig. 10 shows the tree structure for FSD (thick lines) for the case of 3x3 with 4 QAM modulation. Each circle represents a computation of the equations from equation (3.5)-(3.7). As mentioned, earlier all nodes at level i=3 are evaluated and only the best child is considered for levels i=2 and i=1.

Here we use COSIC algorithm to demonstrate the performance of architectures.

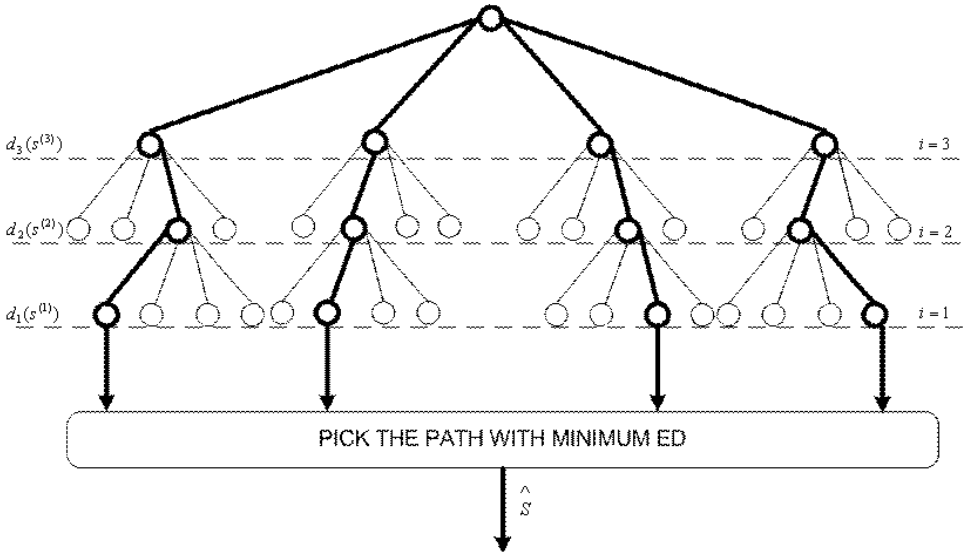

Fig. 10. Simplified Tree Structure for Detection

2. Parallel, Sequential and Staggered Detector Architectures

In this section we describe different architectures with that can be used to implement COSIC. In Fig. 11 we show the serial, sequential [24] and fully parallel [23] architectural dataflow. The numbers next to the nodes indicates the clock cycle in which that node was processed.

a. Sequential Data Flow

The sequential architecture consists of two units. The first unit is capable of computing the top level ($i = 4$) PD's in ascending order and perform radius check (check for SC violation). The second unit concurrently computes PD's/D's for $i = 1, 2, 3$ (second unit is idle for the first clock cycle) and performs radius check. PD's/D's

are computed using a Metric Computation Unit(MCU). The sequential architecture makes use of the radius reduction scheme and node pruning. The nodes labeled 4,7,...can potentially update the radius value. Note that in sequential scheme the radius update may happen every 3 cycles (after first 4 cycles). The major drawback of this architecture is that it provides a highly variable throughput. The latency for detecting one vector symbol is dependent on the channel conditions and noise level.

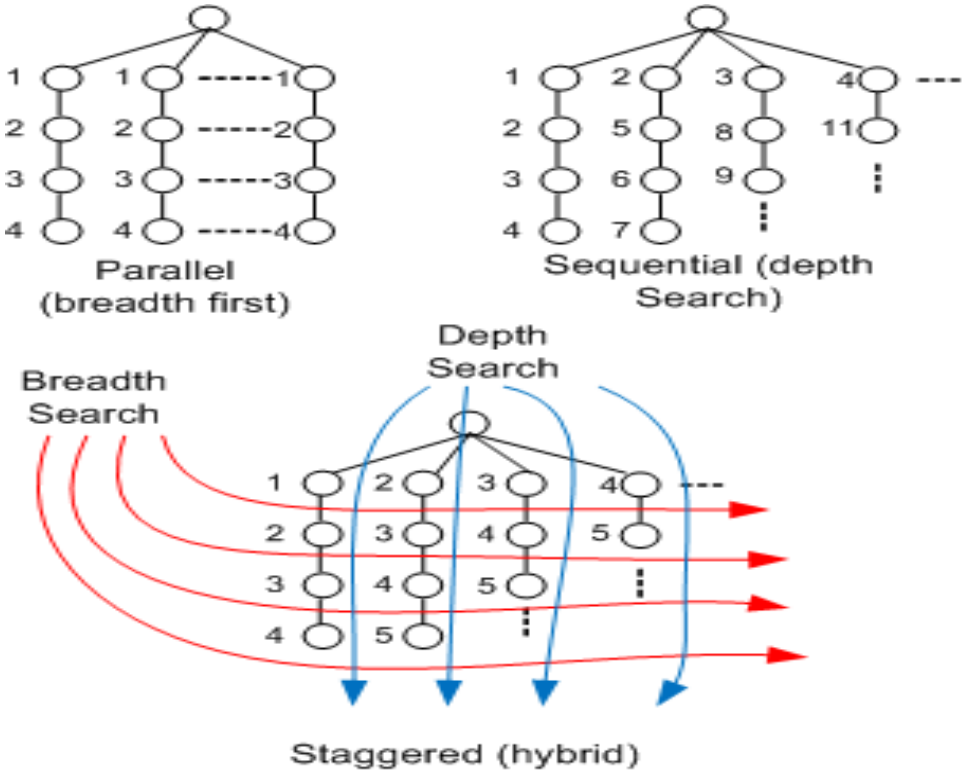

Fig. 11. Dataflow for Architectures

b. Parallel Data Flow

In parallel architecture, all η nodes at every level are evaluated in parallel using η units. Each unit is capable of computing PD's/D's, but does not perform a radius check. After reaching level $i = 1$, a compare/select logic picks the path with least D, and declares it as $\widehat{s}$. This scheme has a fixed throughput because it takes 4 cycles to compute the estimate $\widehat{s}$ irrespective of channel conditions and noise level. The parallel architecture does not do any node pruning, however, it incurs redundant computations leading to higher energy consumption.

c. Staggered Data Flow

The staggered architecture has a schedule that is staggered in time. One copy of MCU is deployed at each level. The staggered computation is explained as follows. In the first cycle a node is evaluated at level 4 (labeled 1 in the Fig. 11). In second cycle a node at level 4 and level 3 are evaluated (labeled 2 in the Fig. 11). From the 4th cycle onwards one node (subject to SC) from each level is evaluated (resulting in a potential radius update at every cycle). This approach is a hybrid between the fully serial and the fully parallel architecture, in that it initiates a new breadth and depth search every cycle (Fig. 11). It does not deliver a fixed throughput in contrast to the fully parallel implementation of the COSIC algorithm, but has variability which is much less than serial and sequential architecture due to per cycle radius update. Like sequential architecture, the staggered architecture too follows the radius reduction technique with pruning, thus reducing redundant computations in comparison to the parallel architecture. Also, after the first four cycles, potentially a new leaf node is reached which leads to faster radius reduction. This results in reduced variability of runtime.

3. Architecture and Implementation Details

In this subsection, we describe the architecture and implementation of details for staggered architecture. Details for sequential and parallel architectures can be deduced easily from this.

Fig. 12 shows high level architecture for the staggered decoding scheme. It has four units MCU_4 through MCU_1 operating in a pipeline. The task of MCU_4 is to generate QAM symbols (s_4) and their metric in ascending order (using the enumeration scheme in Fig. 13) and to perform the radius check. The outputs of MCU_4 are: SC_4 (a '1' indicates an SC violation at $i = 4$), s_4 , the QAM symbol currently under consideration, and $d_4(s^{(4)})$, the PD associated with s_4.

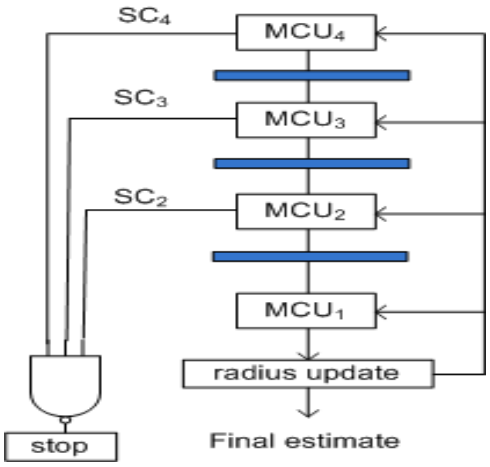

Fig. 12. High Level Architecture for Staggered Sphere Decoding

MCU_3, MCU_2, and MCU_1 differ from MCU_4 in that they do not enumerate the QAM symbols. Instead they find the best child of the input node, compute the associated PD (or D), and perform the radius check (MCU_1 does not perform a radius

check, since it is implicitly done in the "Radius update" block).

4. Metric Computation Unit

Each node in Fig. 12 represents a Metric Computation Unit (MCU) that computes metrics $d_i(s^{(i)})$ using equations (3.5)-(3.7). For $i = 4, 3, 2, 1$, the equations equations (3.5)-(3.7) can be expanded as follows:

$$d_4(s^{(4)}) = |y_4 - R_{44}.s_4|^2 \tag{3.8}$$

$$d_3(s^{(3)}) = d_4(s^{(4)}) + |y_3 - R_{34}.s_4 - R_{33}.s_3|^2 \tag{3.9}$$

$$d_2(s^{(2)}) = d_3(s^{(3)}) + |y_2 - R_{24}.s_4 - R_{23}.s_3 - R_{22}.s_2|^2 \tag{3.10}$$

$$d_1(s^{(1)}) = d_2(s^{(2)}) + |y_1 - R_{14}.s_4 - R_{13}.s_3 - R_{12}.s_2 - R_{11}.s_1|^2 \tag{3.11}$$

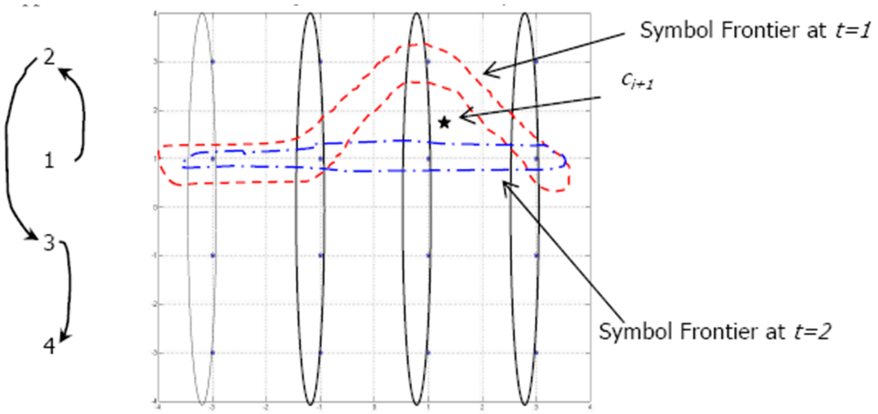

Fig. 13. Enumeration Scheme for 16-QAM

Notice that the number of computations required in equations (3.8)-(3.11) dependent on the level i of the tree. The largest number of computations are required at

$i = 1$ i.e. at the leaf nodes equation (3.11). Fig. 14 shows an MCU which can handle computations at $i = 1$. Note that for, the parallel,and sequential architectures the same MCU is used to compute metrics for different values of i. This can be achieved by adapting the MCU to perform the computations of equations (3.10),(3.9) and (3.8). For the sequential architecture, however, the MCU has to be adapt between levels $i = 1, 2, 3$ only, since at the $i = 1$ level we have a dedicated unit to compute PD's.

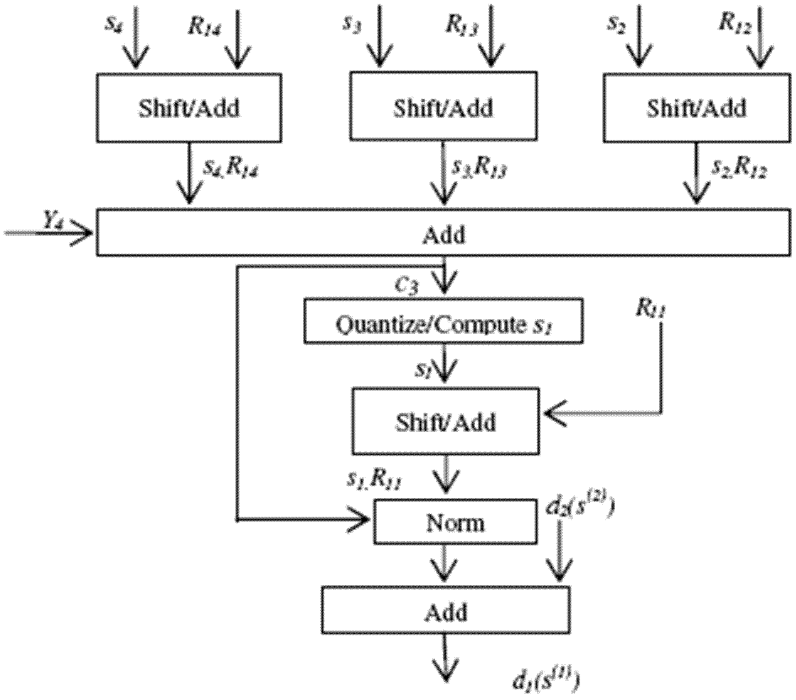

Fig. 14. Architecture for MCU1

It is clear from equations (3.8)-(3.11), that the adaptation can be carried out simply by input reassignment and by setting some of the operands to zero. In the

sequel we will refer to MCU configured for a level i as MCU_i. Physically the MCU is the same; it is merely reconfigured to operate at different levels i. When the MCU is configured to evaluate nodes at level $i > 1$ many adders remain idle during the computation. The terms in equations (3.8)-(3.11) involving multiplications of the form $R_{ij}.s_i$ can be computed using shift and add operation. This is possible because the real and imaginary parts of s_i can only take values for example {-3,-1, 1, 3} for 16-QAM.

5. Enumeration

To process the children nodes of a node in ascending order of their PD's, the distances of c_{i+1} from the scaled (by R_{ii}) QAM points need to be ordered. A straightforward way to do this is to directly compute all distances and sort them. However, this approach is not efficient as only one node from the top level is needed at a time.

A more efficient way to achieve this is through direct enumeration of QAM symbols based on the location of c_{i+1} in the constellation. Many enumeration techniques exist in literature. In [25] the author proposes a method based on dividing the constellation into concentric circles [25]-[20]. Recently a new technique has been proposed in [24],[33]. The basic idea is to divide the constellation (Fig. 13) in a number of columns (four in case of 16-QAM). The order of points in these columns is already known given the location of c_{i+1}. For the example shown in the Fig. 13, the order is shown on the left hand side of the constellation diagram. This order is identical for every column. Also notice that only one element in SF and MF will change at a time. For instance, in the SF at time $t = 1$ is 2, 6, 10, 14. Since the minimum distance symbol is '10', the SF is updated at $t = 2$ to 2, 6, **9**, 14.The set of best QAM symbols (or simply symbol) from every column will be called a Symbol Frontier or SF. The SF can at most have 4 elements in it. Using symbols in the SF their PD's can be

computed using equations (3.8)-(3.11). The set of PD's of the elements in the SF will be called Metric Frontier or MF. The best symbol (closest to c_{i+1}) out of all 16 symbols can now be found by picking the symbol associated with the minimum element in the MF. Note that the SF and hence MF keep changing with time. Note that only the location of the imaginary part of c_{i+1} is required to compute the order of symbols in each column. Symmetry of the QAM constellation can be exploited to reduce the number of decisions we have to make in order to locate c_{i+1} in the constellation. For instance if c_{i+1} was in the third quadrant of the constellation, we can ignore the signs of its real and imaginary part to get $c\prime_{i+1}$. The SF can be computed for $c\prime_{i+1}$. To get back the SF associated with the original c_{i+1} we simply change the signs of real and imaginary parts of the SF elements. This way we only need to compare the imaginary part of c_{i+1} with R_{ii}, and $2.R_{ii}$. Also note that for $i = 4$, $c_{i+1} = \hat{y}_4$.

6. Simplified Norm Computation

The MCU as shown in Fig. 14 has a block labeled "Norm". The Euclidean norm or l^2 norm involves a squaring operation which requires multipliers. Multipliers are in general expensive in terms of hardware cost. In [20] it has been shown that the use of simplified norms leads to significant reduction in hardware cost with some BER degradation. The Euclidean norm in equation (3.5) can be replaced with l^1 or l^∞ norm to simplify the implementation of equations (3.5)-(3.7). The l^1 norm approximation is given by:

$$d_i(s^{(i)}) = d_{i+1}(s^{(i+1)}) + |Re\{e_i(s^{(i)})\}| + |Im\{e_i(s^{(i)})\}| \tag{3.12}$$

and the l^∞ norm approximation is given by:

$$d_i(s^{(i)}) = max(d_{i+1}(s^{(i+1)}), |Re\{e_i(s^{(i)})\}|, |Im\{e_i(s^{(i)})\}|) \tag{3.13}$$

where Re and Im denotes real and imaginary parts. The use of l^1 norm is of particular interest as it causes the BER to degrade only by about 0.4dB [20]. The use of l^∞ norm on the other hand causes about 1.4 dB loss [20]. Both these norms have almost the same hardware complexity [20]; however the algorithm takes much longer to converge when the l^1 norm is used [20],[22]. In the rest of this chapter, we will use the l^1 norm due to its better BER.

7. Architecture of MCU_4

Fig. 15 shows the detailed structure of MCU_4. As mentioned earlier, MCU_4 outputs symbols and associated PD's in their ascending order. To this end it follows the enumeration strategy outlined earlier. Recall that knowing the location of c_{i+1} in the constellation tells us the order of the QAM points in each column. Once the location of c_{i+1} is known, the SF is computed using a bank of PC_k blocks (k=1,2,3,4). A bank of Partial Distance Units (PDUs) operates on the SF to compute MF. A PDU block computes the distance of c_{i+1} from a scaled QAM symbol. The compare select logic picks the best symbol based on the minimum PD. Fig. 16 shows details of the PC_k block. It consists of a combinational logic block (F), which outputs the imaginary parts of the QAM symbols in signed magnitude format. It has four outputs of three bits each. These outputs correspond to the unique pattern of imaginary parts of the four symbols in a column depending on location of c_{i+1} in the constellation. These imaginary parts are finally concatenated with the real part (notice that in each column the symbols have identical real parts). The counter labeled $cntr_k$, is a 3 bit counter whose two LSBs select the best available symbol in column k. Note that we need a three bit counter in order to identify the exhaustion of a column. Since the column has only 4 symbols in it, a value of 4 (counter starts from value '000') indicates that all the symbols in that column have been exhausted. Hence,

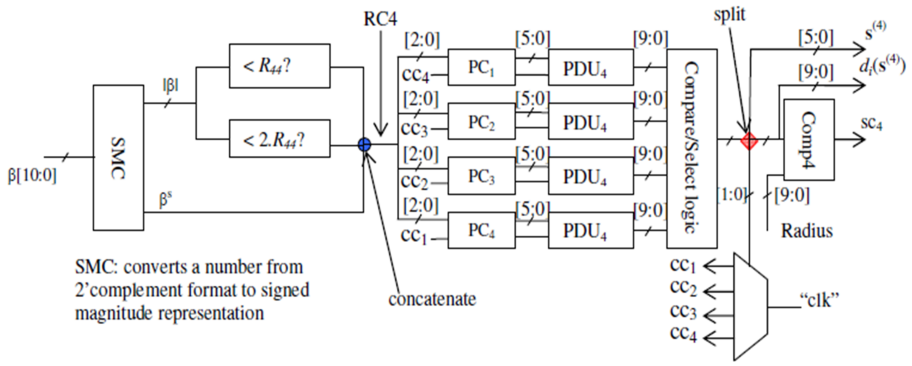

Fig. 15. Details of MCU_4

whenever counter value is 4 a signal f_{ck} is generated. This signal alerts the PDU to saturate its output. This essentially excludes symbols from that column from further consideration. The outputs of a PC_k block (a frontier symbol from k^{th} column FS_k and a control signal f_{ck},) are fed to a PDU. The PDU computes the associated PD of the FS_k. The compare/select logic block along with the demultiplexer identifies which frontier symbol needs to be replaced. Fig. 17 shows details of a PDU. The PDU at the top level differs with the one shown in Fig. 17 in that there is no $d_{i+1}(s^{(i+1)})$ input, since $d_5(s^{(5)}) = 0$. Also, for PDUs at levels $i < 4$ there is no f_{ck} input, and no multiplexer M2. This difference is indicated in Fig. 17 by means of dotted lines.

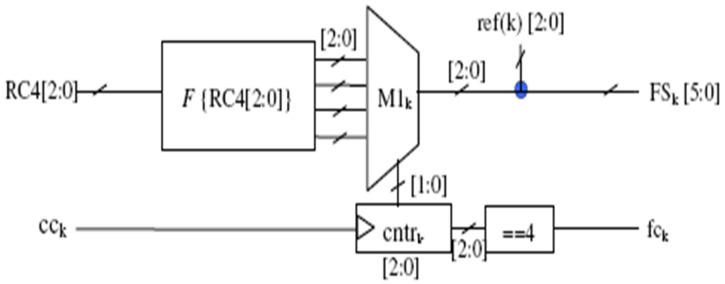

Fig. 16. Details of PC_k k=1,2,3,4

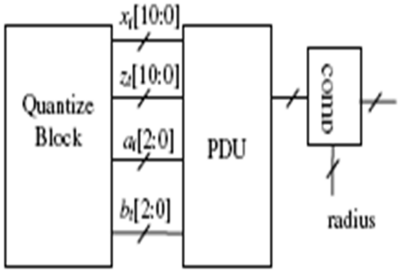

Fig. 18. Details of MCU_k k=1,2,3,4

8. Architecture of MCU_3,MCU_2,MCU_1

The top level architecture of MCU_3 through MCU_1 is shown in Fig. 18. MCU_3 through MCU_1 differ from MCU_4 in that they don't have to explicitly enumerate QAM symbols in sorted order; all that is needed to be done by MCU_3-MCU_1 is to find the best child. Finding best child essentially means "quantizing" c_{i+1} (Fig. 13)

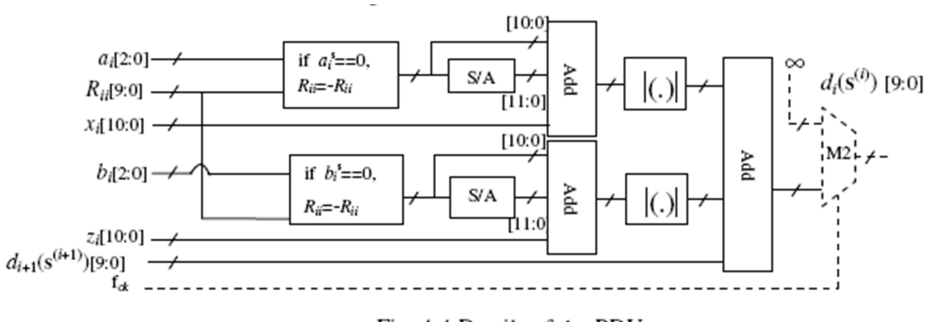

Fig. 17. Details of the PDU

to the nearest QAM symbol. Fig. 19 shows the details of the "Quantize" block.

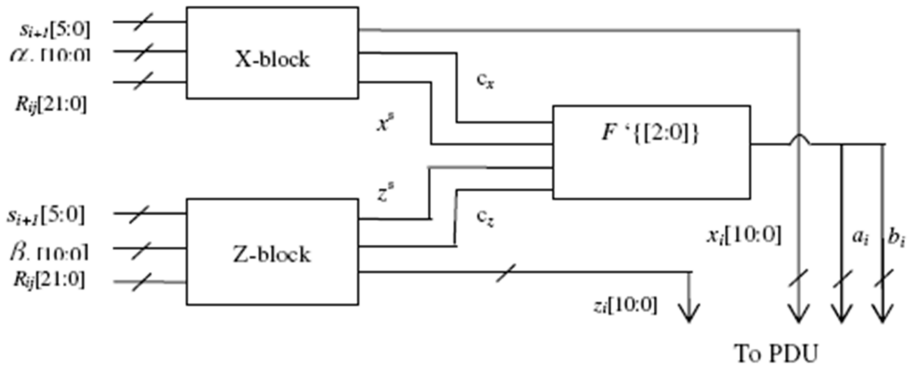

Fig. 19. Details of the Quantize Block

Let $y_i = \alpha_i + j\beta_i$, $s_i = a_i + jb_i$, $R_{ij} = u_{ij} + jv_{ij}$, and $c_{i+1} = x_i + jz_i$. Substituting these expressions in equation (3.8) and after some algebraic manipulations equation

(3.8) can be written as (for i=1,2,3):

$$x_i = \alpha_i + \sum_{k=1}^{M_T - i} [(-u_{i,i+k})(a_{i+k}) + (v_{i,i+k})(b_{i+k})] \tag{3.14}$$

$$z_i = \beta_i + \sum_{k=1}^{M_T - i} [(-u_{i,i+k})(b_{i+k}) + (-v_{i,i+k})(a_{i+k})] \tag{3.15}$$

Note that for $i = 4$ $x_i = \alpha_i$, and $y_i = \beta_i$. Since c_{i+1} is also a complex number we need to quantize it along both the real and imaginary dimensions. For this purpose we use the X-block (for the real part) and the Z-block (for the imaginary part). These blocks compute x_i and z_i using equations (3.14),(3.15). Note that here we have exploited the symmetry of the QAM constellation. First we find the magnitude of real and imaginary parts of c_{i+1}, so that the new $c'_{i+1} = |x_i| + j.|y_i|$ is now in the first quadrant. Using the sign bits x^s and z^s along with the location of the c'_{i+1} we can find the location of c_{i+1}. This function is carried out by F' block which is a very simple combinational block which directly outputs the QAM symbol (indicated by a_i and b_i in the Fig. 19 in signed magnitude format. Fig. 20 shows more details of the X block. The X-block carries out computations according to equation (3.14). Product terms in equation (3.14) are implemented using shift and add function (shown by the dotted boxes). The multi-operand additions are performed by an arithmetic Sum of Products (SOP) block [40]. The Z-block is structurally similar to the X-block with inputs reassigned, as can be seen from equation (3.15).

Sequential architecture implementation: Recall that this architecture has two MCU's. The first MCU computes the PD's of children of the root node in their ascending order. The second MCU serially evaluates nodes at levels $i = 3, 2, 1$. Both the units are capable of carrying out a radius check. The MCU at the top level stops evaluating nodes whenever the SC is violated. The second MCU continues

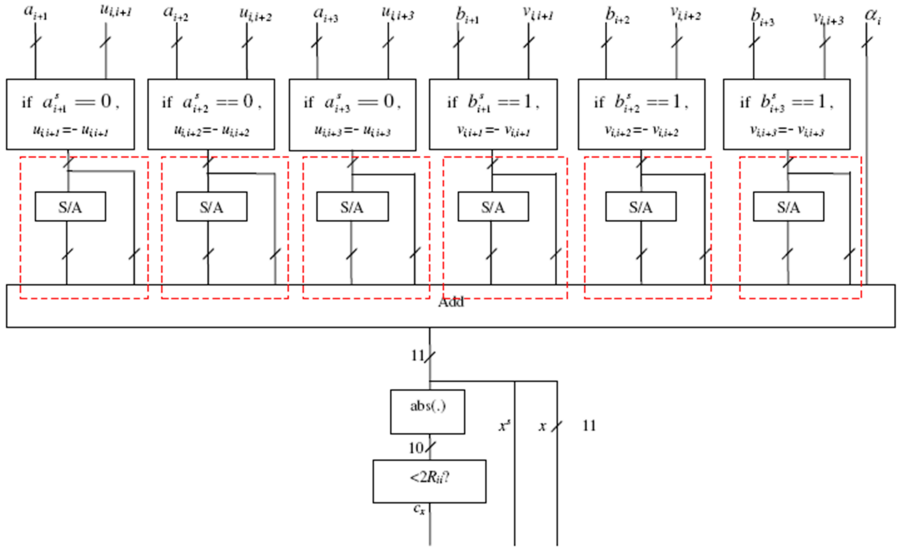

Fig. 20. Details of the X-block

until all the nodes evaluated by the first unit are processed. Notice that the top level MCU is exactly same as that in the staggered architecture. The second MCU has to be reconfigured according to the level of the node it is computing. Fully serial architecture implementation: In this architecture there is only one MCU that is shared between all the levels of the tree. Notice that the first four cycles are spent to compute the SF/MF. Hence, the first node at the top level is available for computation only at the 5^{th} clock cycle. Notice that in this scheme the unit has to be reconfigured for $i = 4, 3, 2, 1$.

Fully parallel architecture implementation: In this architecture, there are 16 MCU's operating concurrently. These MCU's evaluate all 16 nodes at a given level in a single cycle. In this architecture the MCU's also have to be reconfigured for $i = 4, 3, 2, 1$. The difference between the MCU's in the parallel and the staggered

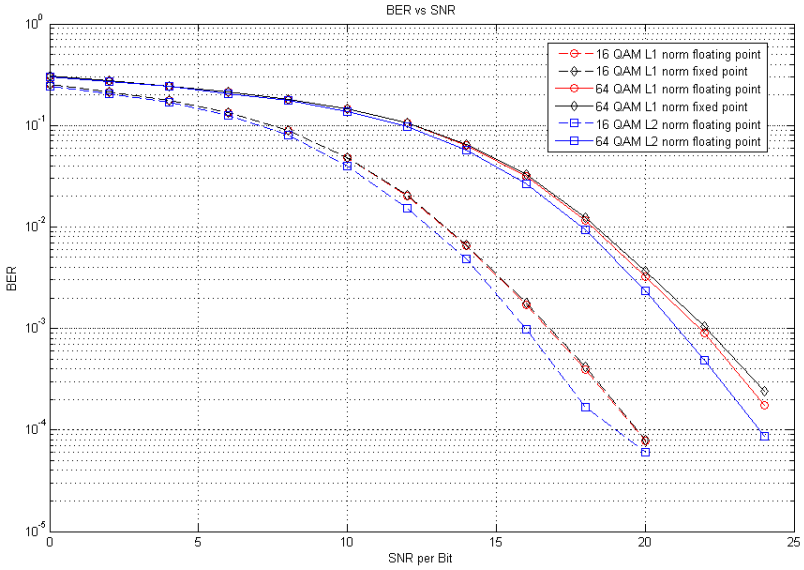

Fig. 21. BER for Suboptimal Norms and Finite Bit Precision

architectures is that in the parallel architecture, the enumeration is not carried out. The quantization operation (of c_{i+1}), however, is still needed for levels $i = 1, 2, 3$ in the parallel architecture.

C. Discussion

In this section we present the implementation results and analyze the architectures in situations that occur in real-life baseband systems. We start by noting that both sequential and staggered architectures have variable throughputs, something which is undesirable in practical systems. On the other hand, parallel architecture provide constant throughput at a much higher cost per unit area.

The implementation of the aforementioned architectures was done using a customized Compare/Select and Comparator blocks and standard cell based design in 100 nm technology [34]. The customized Compare/Select block was simulated in SPICE3 [35] to obtain its delay and power. The customized Comparator block is implemented as designed in [39]. Hence the area, delay and power of the comparator as reported in [39] (for a 100 nm technology) are used as such. For the standard cell based design, the delay was computed from a sensitizable timing analysis tool (Sense) [36] and the power was computed using a script written in the SIS [37] logic synthesis environment. This script computes the re-convergence adjusted signal probability, which in turn is used to compute the dynamic power consumption of the design. For both designs, the active area was computed and used to estimate the area of the architectures discussed above. We chose wordlengths of elements of coefficients as 11bits for 16-QAM and 12bits for 64-QAM. The fixed point vs floating point BER is compared in Fig. 21.

1. Building Blocks

Compare Select Circuit: The Compare/Select block uses the Longest Prefix Matching (LPM) circuit designed in [38]. The logic behind the Compare/Select block is illustrated in Fig. 22 with an example. Consider the problem of finding the maximum among three 4 bit operands as shown in part (a) of Fig. 22. The LPM looks each column of bits at a time (starting from the MSB) and eliminates the operand that has a '0' in that column, if anyone of the other bits is a '1'. Hence in part (b) of Fig. 22 operand '0101' is eliminated. Hence, only the first two operands contend. Since both these operands have the same bit in column 2 (part (a) of Fig. 22), the LPM looks for the next column. In column 3, the second operand has a '0' and hence gets eliminated. Thus the LPM declares the first operand as the largest value.

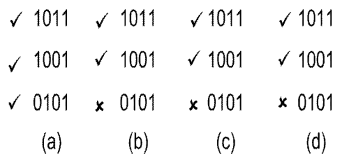

Fig. 22. Compare Select Operation

Sum-of-Product(SOP): The remainder of the design is standard cell based, implemented in 100 nm technology. The multi-operand addition is done using a Sum-of-Product (SOP) design [40]. The SOP is better than a tree of adders, since it has a single carry chain. The SOP block consists of 3 steps - Partial Product generation, Partial Product reduction using half and full adders and a single final 2 operand binary adder. The final adder is implemented using a fast Kogge-Stone adder [41].

2. Detection in Multicore Setup:

Due to the ever increasing throughput requirements placed on the baseband processor multiple detector cores will have to be used to support it. This arrangement is shown in (Fig. 23) where we employ m cores for detection. The aim is to analyze the area efficiency (throughput/area) as we increase m. The exact value of L changes with different standards, for example, 802.11n has 64 tones, LTE has 512 tones.

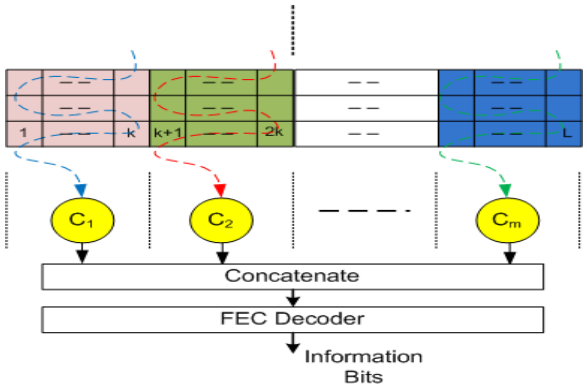

Fig. 23. Multicore Detection for MIMO-OFDM Systems

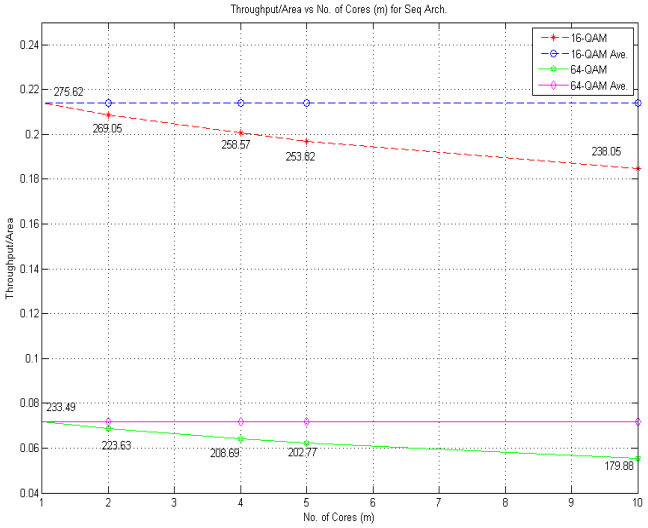

Fig. 24. Throughput/Area Efficiency for Sequential Arch

Some other standards (such as DVB) have number of tones running into several thousand. As a representative number we choose L=500 and study the architectural efficiency as m=1,2,4,5,10.Fig. 24 shows the behavior of the sequential architecture as m increases. The figure essentially shows the area efficiency (throughput/area) for 16/64 QAM. The numbers close to the plotted points indicates the throughput/core. We see that the of the area efficiency drops by $\approx$ 13.6% for 16-QAM as m goes from 1 to 10 (the throughput/core drops from 275.62Mbps to 238.05Mbps). In case of 64-QAM the drop is higher at $\approx$ 23%which can be explained by the fact that higher modulation schemes has larger search space and higher runtime variability. Similar behavior is observed for staggered architecture (Fig. 25). The area efficiency for 16-QAM drops by $\approx$ 12.5% (the throughput/cores drops to 362.93Mbps from 414.75Mbps). For 64-QAM the corresponding drop is $\approx$ 22.8%.

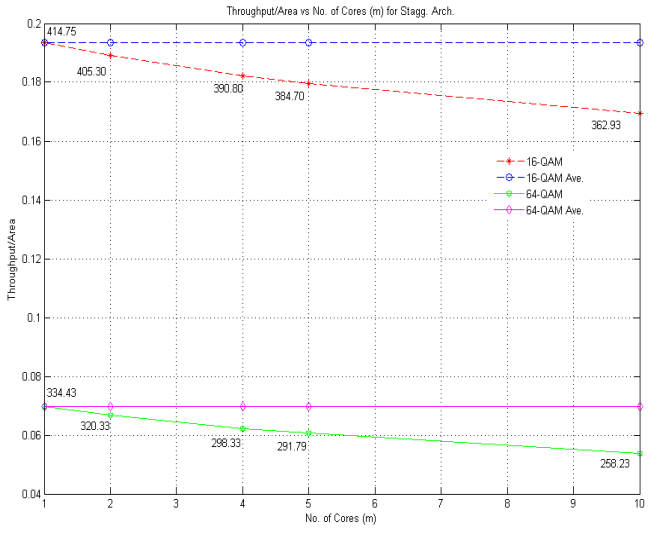

Fig. 25. Throughput/Area Efficiency for Staggered Arch

Block Early Termination (BET): The variable throughput is a major problem as far as practical systems are concerned. To alleviate this problem, the authors of [22] have proposed a scheme called Block Early Termination or BET. Using this scheme, a runtime constraint is established for a block of vector symbols. The runtime is decided by dynamically allocating a limited number of cycles to processing a vector symbol. A scheduling algorithm is used to distribute the available run time over the vector symbols in the block. The strategy allocates a maximum runtime equivalent of $clk_{max}(n)$ to the n^{th} vector symbol in a block according to

$$clk_{max}(n) = N_{max}clk_{ave} - \sum_{i=1}^{n-1} clk(i) - (N_{max} - n)M_T \tag{3.16}$$

where $clk(i)$ denotes the actual number of clock cycles used up for the i^{th} vector symbol. The idea behind equation (3.16) is that a vector symbol is allowed to use up all of the remaining run time within the block up to a safety margin of $(N_{max} - n)M_T$ visited nodes, which allows to find at least the zero-forcing decision feedback solution for the remaining vector symbols [22].

For BET $N_{max} = 50$ (corresponding to m=10), and $clk_{ave} = 9, 19$ for the staggered and the sequential architectures respectively for 16-QAM. These parameters were chosen such that the BER performance (Fig. 26) for each algorithm is very close to that of the unconstrained (at BER of 10^{-3}). Corresponding numbers for 64-QAM are $clk_{ave} = 15, 23$.

Based on Table 1, we note that the Staggered approach achieves about half the throughput of the parallel approach, using about a fifth of the area of the parallel scheme. Also, the throughput of the Staggered approach is twice that of the Sequential approach. In terms of the power per decoded symbol vector, the Staggered approach consumes twice the power as Sequential approach. The throughput per area, the Staggered approach is slightly better than the Sequential approaches.

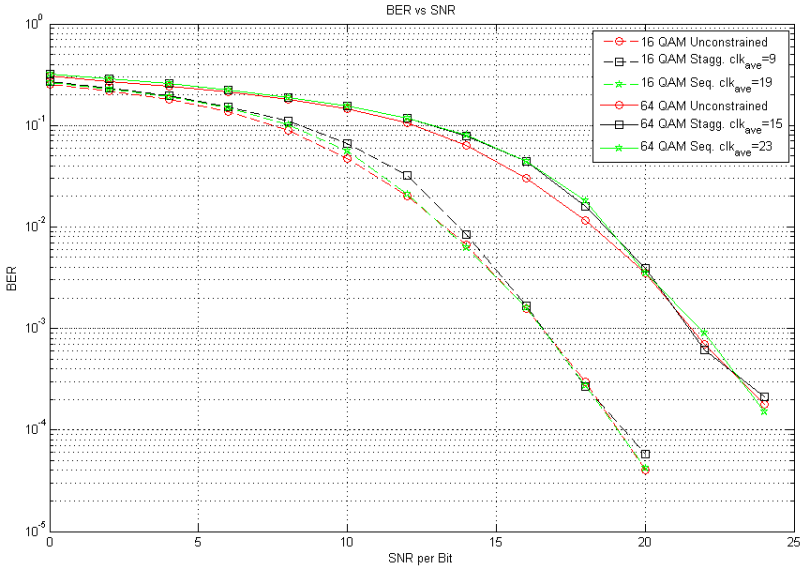

Fig. 26. BER Performance of the Sequential,and Staggered Architectures Under Resource Constraint

Whereas, the Sequential approach is slighly better when is comes to throughput per unit power(calculated at a SNR of 12dB). Sequential and staggered both have better architectural efficiencies than the parallel architecture.Results for 64-QAM are shown in Table 2, here we see similar behaviour in that the sequential architecture is slightly better when it comes to throughput per unit power. Whereas staggered architecture is slightly better for throughput per unit area. Fig. 27 shows the power consumption profile of the staggered as well as sequential architecture as a function of SNR.

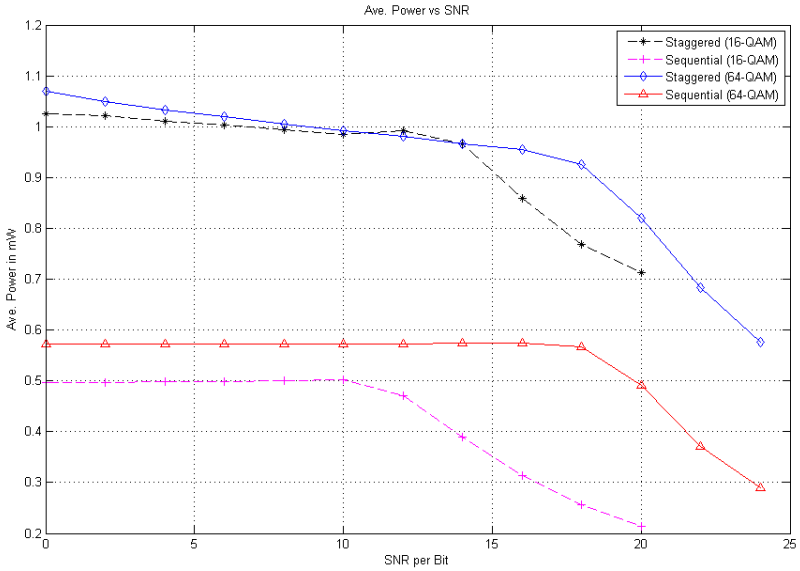

Fig. 27. Power Consumption Per Vector Symbol as a Function of Eb/No

Table 1. Implementation Results (16-QAM)

Arch.	f_{clk} (MHz)	Throughput (T)(Mbps)	Area (μm^2)	T/mW (Mbps/mW)	T/Area (Mbps/μm^2)
Stag.	179.53	319.16	2143.83	325.67	0.15
Seq.	179.53	151.18	1288.33	335.95	0.12
Par.	161.55	646.20	10350.88	118.29	0.06

Table 2. Implementation Results (64-QAM)

Arch.	f_{clk} (MHz)	Throughput (T)(Mbps)	Area (μm^2)	T/mW (Mbps/mW)	T/Area (Mbps/μm^2)
Stag.	138.33	221.33	4804.35	240.58	0.046
Seq.	138.33	144.34	3248.61	262.44	0.044

CHAPTER IV

CONFIGURABLE HARD OUTPUT DETECTOR

MIMO systems is a key technology for future high speed wireless communication standards like 802.11n,LTE, and WiMax. To make judicious use of the spectrum these standards require support for multiple modulation and coding schemes (MCS). Hence, the receiver hardware should be able to accommodate these schemes preferably on a single configurable architecture. The difficulty of implementing MIMO detectors is further compounded by the dynamic configurability requirements. In this chapter, we present a configurable architecture for MIMO detection and its FPGA implementation. The design is able to configure on the fly which is one of the prime requirements for future wireless standards. This chapter also presents detector architecture space exploration for 802.11n standard.

A. Configurable Detector

The choice of algorithm and architecture has a significant impact on the final hardware complexity and configurability. The algorithm should be chosen such that it leads to a highly pipelined and parallel architecture. BER performance of the algorithm crucial as well. The algorithm/architecture should be amenable to dynamic configuration. One implementation that supports configuration is given in [44]. However, the authors in [44] use VBLAST based detection scheme that incurs significant error rate degradation. We chose to use COSIC algorithm [24] for our detector because it can be implemented in a highly parallel and pipelined manner, has fixed throughput(for a given modulation scheme), delivers close to maximum likelihood (ML) performance and is amenable to runtime configuration to detect QPSK, 16-QAM and 64-QAM modulation schemes.

1. Hardware Architecture to Detect QPSK,16-QAM and 64-QAM Symbols

Fig. 28 shows the high level architecture of detector. The choice of 4-way parallelism was made because the smallest constellation supported on our decoder is QPSK (which has four symbols,η=4). If this architecture is pipelined with m stages then it has an initial latency of $m+\eta/4$ clock cycles. Since the COSIC tree has η paths for a η-ary modulation scheme the proposed architecture takes $\eta/4$ clock cycle to detect a η-ary modulated MIMO symbol. At each level of the COSIC tree (Fig. 10) we need to

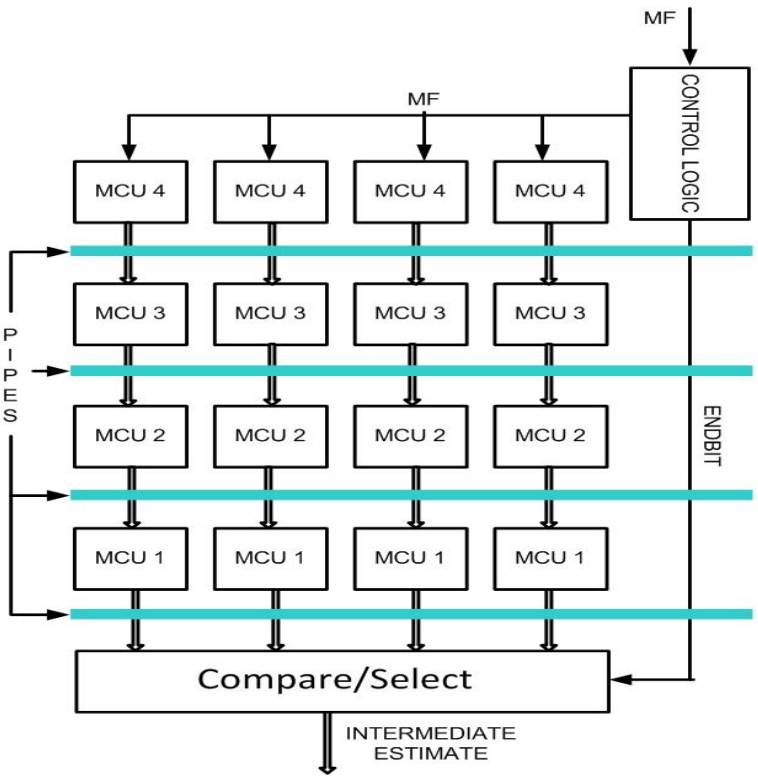

Fig. 28. High level architecture

compute the $d_i(\mathbf{s}^{(i)})$ metrics using equations (3.5)-(3.7). Each of these equations are

computed by dedicated MCUs. Fig. 29 shows the microarchitecture of the MCU at level 1. The upper box in the Fig. 14 evaluates equation (3.7). Recall from chapter III that there is no need to implement the product terms in equation (3.7) using a multiplier. This product can be achieved by shift and add operation, because the QAM constellation points only take on a finite number of integer values (e.g. in 16-QAM scheme the real and imaginary part of $s_j \epsilon \{-3, -1, 1, 3\}$). The block named slicer

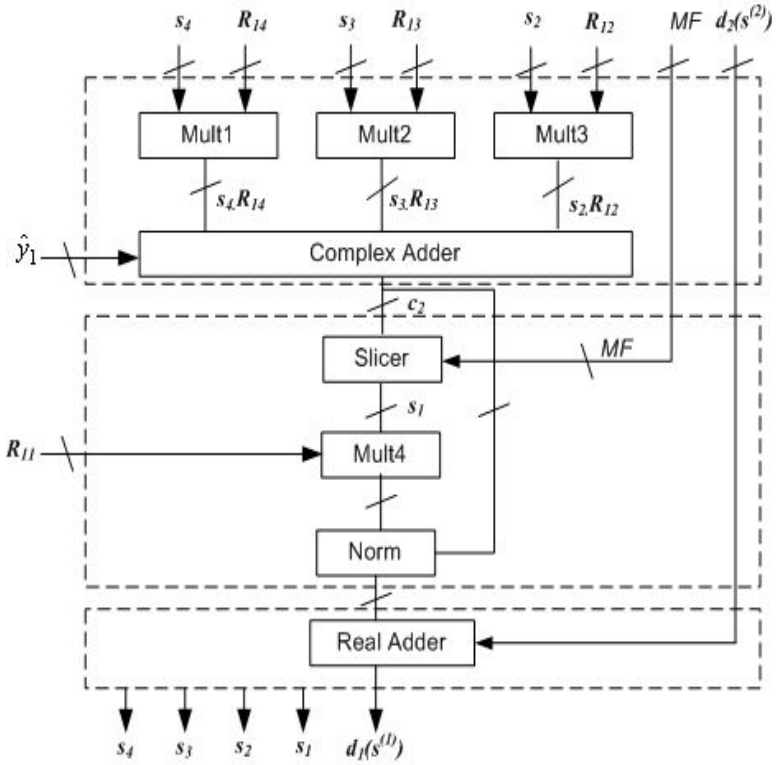

Fig. 29. Metric Computation Unit of Level 1

picks the nearest QAM symbol to c_{i+1} as shown in Fig. 30(a). The slicing operation involves independently comparing the real and imaginary parts of c_{i+1} with appro-

Table 3. Comparison with Existing Designs

Ref.	QPSK	16-QAM	64-QAM	Dynamic Config.	BER
[21]	No	Yes	No	No	Close to ML
[23]	No	Yes	No	No	Close to ML
[20]	No	Yes	No	No	ML
[10]	No	Yes	No	No	Close to ML
[44]	Yes	Yes	Yes	Yes	Sub-Optimal
[45]	No	Yes	No	No	Close to ML
[*Ours*]	Yes	Yes	Yes	Yes	Close to ML

priate decision thresholds. The decision thresholds are given by $(-(\sqrt{\eta}-2)+2j)R_{ii}$, where j is an integer such that $0 \leq j \leq (\sqrt{\eta}-2)$. The detector configures the slicer based on Modulation Format (MF) bits, which indicates the modulation scheme of the current MIMO symbol. The control unit of our design is a simple Finite State Machine (FSM) which takes in MF[1:0] (00=>QPSK, 01=>16-QAM,and 10=>64-QAM) and generates a signal 'endbit' every $(\eta/4)$ clock cycle. This signal indicates the completion of detecting one MIMO symbol. The waveforms in Fig. 31 shows the relation of the control signals with respect to the MIMO symbol. From Fig. 31 it can be seen that the detector has systolic like qualities and provides uninterrupted detected symbols. The design of the control unit is independent of m, this implies that very little redesign effort is required in case one wants to achieve very high throughput by increasing m(subject to latency constraint). The parallelism factor can also be increased (with corresponding changes in the control logic) to further increase the throughput. Table 3 shows a qualitative comparison of existing architectures for the detector.

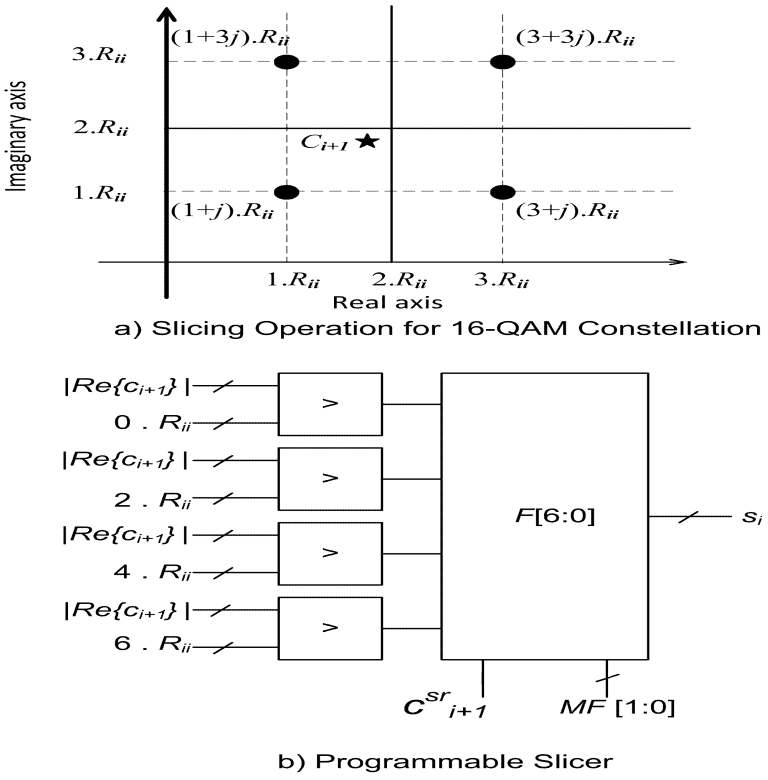

Fig. 30. Slicer

Table 4. FPGA Implementation Details

Target FPGA Device	xc4vfx60 (Xilinx Virtex-4)
Number of 4 input LUTs	10,745
Number of Slice Flip Flops	851
Multipliers	None
Maximum Frequency	35MHz
Decoding Rate: QPSK	280Mbps
Decoding Rate: 16-QAM	140Mbps
Decoding Rate: 64-QAM	52.5Mbps
Control Logic Overhead	0.3%

MATLAB is used to simulate bit accurate model of the decoder. We chose eleven bit fixed point quantization(internal precision is maintained). Based on the MATLAB model, a detailed hardware architecture is developed. The RTL coding and synthesis is done using Verilog HDL and Xilinx ISE 8.1 Embedded Development Kit respectively. Xilinx Virtex-4 [xc4vfx60] device is used for mapping the synthesized netlist. Floorplanning, Place and Route(P&R) of the design is done using the integrated Xilinx Floorplanner and automatic P&R tool. The input test vectors are generated by the fixed-point MATLAB model. The hardware design is validated by carrying out simulations on these test vectors with the Post-P&R simulation model using ModelSim PE 6.3c. Table 4 shows the implementation results.

2. Hardware Architecture to Detect QPSK,16-QAM and 64-QAM Symbols for 2x2, 3x3, and 4x4 MIMO Systems

Fig. 32(b) shows the high level architecture of the detector. It consists of a systolic-like array $MCUs$ along with a Find Minimum Unit (FMU). The output of the array

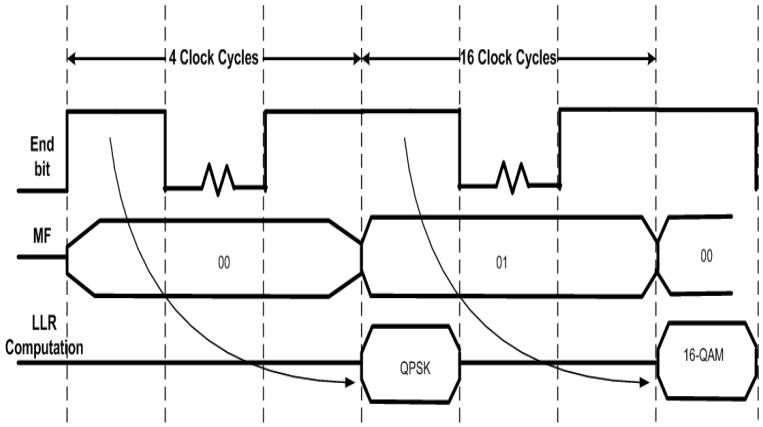

Fig. 31. Output and Control Waveforms

is fed to the FMU (Fig. 33), which iteratively computes the final estimate $\hat{\mathbf{s}}$ by picking the best path in the COSIC tree.

Fig. 34 shows the array nodes in some more details. The incoming signals from the left are provided by the local memory. Instruction $instr = \{MF, NS, NA\}$ indicate the modulation format and end of the current MIMO symbol processing (or arrival of new symbol), and number of antennas respectively. For example, the output of the node at level 4, which is $\{s_4, d_4(s^4), instr\}$, is then fed to the nodes at the lower level of the tree. Fig. 34 shows the details of nodes at various levels of the tree. The node at level 2 (MCU2) operates on $\{s_4, s_3, s_2, d_2(s^2), instr\}$ to produce $\{s_4, s_3, s_2, s_1, d_1(s^1), instr\}$ Fig. 35. FMU then operates on the output of the nodes at level 1 and iteratively computes the best MIMO symbol estimate,which is $\{s_4, s_3, s_2, s_1\}$.

At the start of the detection process, a QAM symbol (s_4) from the constellation is picked. In the next clock cycle next QAM symbol is picked and so on. This

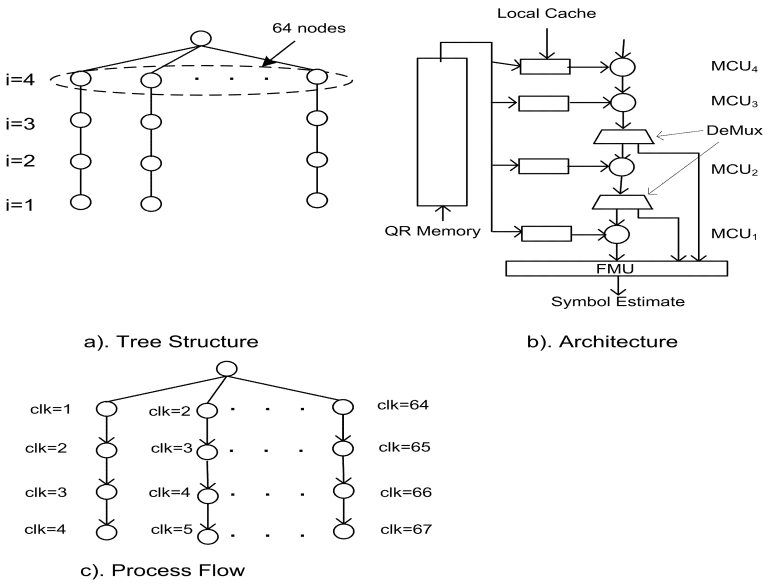

Fig. 32. Tree Structure for FSD Algorithm and Systolic-Like Array Architecture

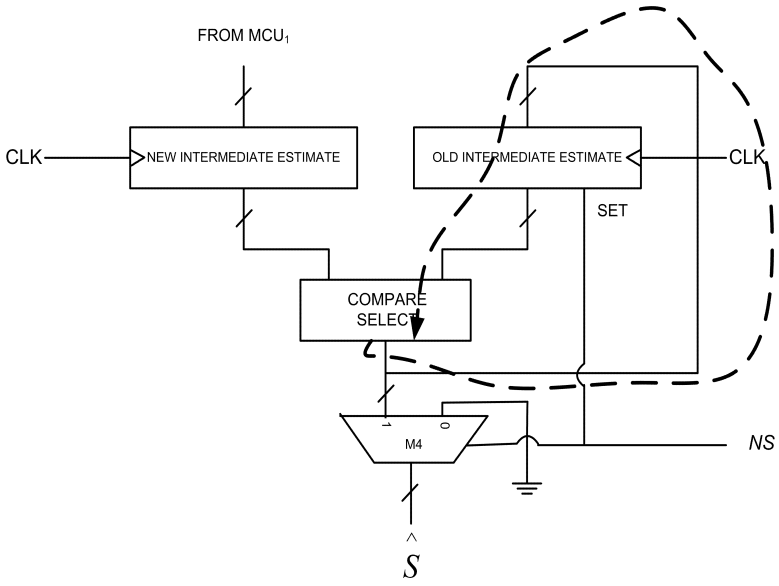
FROM MCU1
CLK
NEW INTERMEDIATE ESTIMATE
OLD INTERMEDIATE ESTIMATE
CLK
SET
COMPARE
SELECT
1
0
M4
NS
S

Fig. 33. FMU

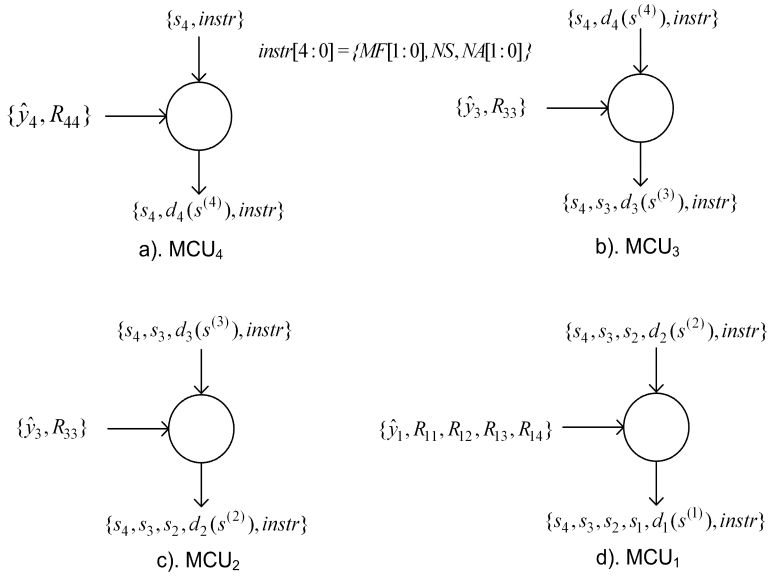

Fig. 34. Node Details

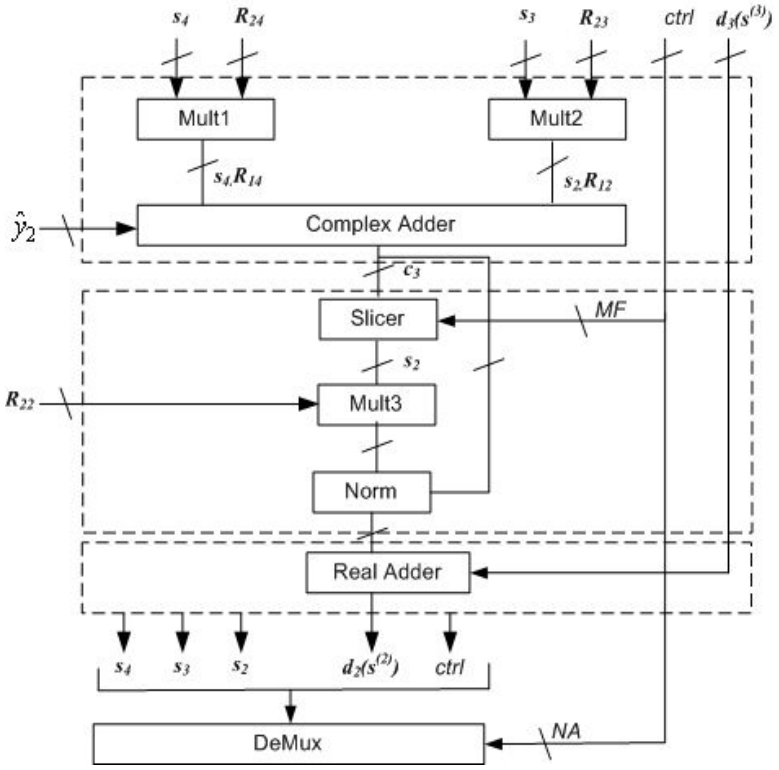

Fig. 35. Micro-Architecture of MCU2

continues until all the QAM symbols are exhausted. Once all symbols from the constellation are exhausted, processing on the next MIMO symbol can be started immediately. The initial latency is incurred only during the processing of the first MIMO symbol, and is hidden during the following MIMO symbols. This initial latency depends on the number of pipelines inserted in the array. Since the COSIC tree has η candidate MIMO vectors to be evaluated(for an η-ary modulation scheme), and because the detector incurs no configuration latency, it takes η clock cycles to detect an η-ary modulated MIMO symbol in steady state. The throughput of the

detector can be very easily computed as follows. We are considering a $n \times n$, where n=1,2,3,4 MIMO system, and an η-ary QAM symbol is constructed using $log_2(\eta)$ bits. Hence, per detected MIMO symbol there are $nlog_2(\eta)$ bits, the throughput will then be $(nlog_2(\eta)/\eta)f$ bps, where f is the clock frequency. Hence, higher throughput is achievable by introducing more pipeline stages. Also, multiple detector cores can be used to increase the throughput further.

Because the information (data and control signals) flow is unidirectional the architecture can be deeply pipelined. Only constraint on number of pipelines is the loop in the FMU (Fig. 33). We introduced 10 pipeline stages it(and re-timed) in the whole architecture(array+FMU). The detector core can operate at 500MHz clock frequency. As a result the proposed detector design is able to achieve very high throughput even for high order modulation scheme like 64-QAM (187.5Mbps), whereas, QPSK and 16-QAM are detected at 1Gbps and 500Mbps respectively. The basic aim of a communication system is to increase the throughput with constellation size. In the presented design the throughputs decrease with increasing constellation size. Also, multiple detector cores can meet the most stringent throughput requirements. For example, the 802.11n standard requires the throughput to be 346.7Mbps, 231.1Mbps, and 115.6Mbps for 64-QAM, 16-QAM and QPSK respectively [12]. Clearly just two detector cores can meet the requirements for all the modulation schemes for 802.11n. Table 5 shows the ASIC implementation estimates of our detector and how it compares with existing solutions.In table opt. means optimal and c-opt. means close to optimal.

Table 5. Comparison

Ref.	Supported Modes	BER	Area	Power (mW)	Tech. (nm)	Throughput (Mbps)
[42]	16-QAM	c-opt.	50KGE	473	250	169
[43]	16-QAM	c-opt.	91KGE	626	350	52
[48]	QPSK	opt.	685KGE	N/A	180	28.8
[49]	16-QAM	c-opt.	175KGE	407	180	160
Ours	nxn,QPSK	c-opt.	18KGE	N.A.	45nm	250.n
	nxn,16-QAM	c-opt.	18KGE	N.A.	45nm	125.n
	nxn,64-QAM	c-opt.	18KGE	N.A.	45nm	46.87.n
	where n=2,3 or 4.					

B. Configurable MIMO Detectors for 802.11n WLAN: A Design Case Study

While searching for an appropriate architecture attention needs to be paid to application requirements such as required throughput,limits on latency, and configuration between different modes of operations given the requirements of the standard. Important hardware design metrics such as area and power needs to be optimized over all the operating modes of the detector. Here we carry out extensive architectural space exploration to address the issues of power consumption,area, and configurability between different modes of operation while meeting the standards throughput requirement. Ultimately, we come up with two designs that target low area and low power respectively. The design estimates are based on 45nm technology library. We do not include the intermediate estimate unit in the analysis here, this analysis can be easily extended to the design that includes the intermediate estimate unit.

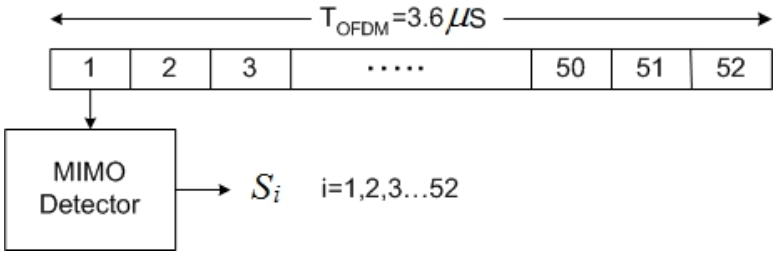

Fig. 36. MIMO-OFDM Detection Interface Timing

1. 802.11n Standard Requirements

In this section be begin with a brief discussion about the throughput limitations imposed by the 802.11n standard. We then develop a strategy to evaluate various architectural parameters that meet the requirements of the standard. In MIMO-OFDM systems, such as 802.11n, OFDM is used to mitigate the affect of multi-path fading. There are 52 data tones (or sub-carriers) to be processed at the receiver. Each tone carries a MIMO symbol, hence the detector has to process 52 MIMO symbols in the stipulated time of 3.6μs(as imposed by the standard). This fact is shown pictorially in Fig. 36. All 52 data tones are modulated using same modulation scheme (e.g. on the cell phone side), thus the detector does not have to switch between modes within an OFDM symbol. Fig. 37 shows the array detector that can process m candidate vectors in parallel. Furthermore, each data path in the array can be pipelined into k parts. The objective is to find m and k such that power and area is minimized as much as possible subject to the throughput constraint. Since, the decoder is configurable this optimization has to be over all the supported modes.

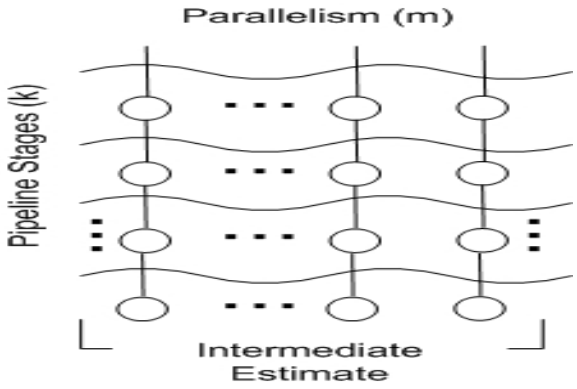

Fig. 37. High Level Architecture for FSD Based MIMO Detection

2. Throughput Planning

First we will establish the relationship between the time taken, T_p, to process 52 MIMO symbols with m, k and η. Since the COSIC tree has η candidate vectors to be evaluated(for an η-ary modulation scheme), it takes $\lceil \eta/m \rceil$ clock cycles to detect an η-ary modulated MIMO symbol in steady state (where $\lceil . \rceil$ is the ceil function). Hence, T_p is given by equation (4.1).

$$T_p = 52. \lceil \eta/m \rceil . freq \tag{4.1}$$

We assume that the critical delay of data-path after introducing k pipelines(and re-timing) reduces to C_d/(k+1). This assumption has been validated empirically for k=0 to 10, using Synopsys re-timing utility . equation (4.1) thus becomes:

$$T_p = 52. \lceil \eta/m \rceil . C_d/(k+1) \tag{4.2}$$

where the factor 52 corresponds to the number of data tones, C_d is the combina-

tional delay of the un-pipelined data-path (or equivalently un-pipelined array). Since 802.11n requires that the processing of all 52 tones be over in 3600ns, $T_p \leq 3600$ns. Using above discussed model we show (Fig. 38) how T_p behaves in the architecture space(with m,k) for QPSK, 16QAM, and 64QAM (η=4,16 and 64 resp.). Fig. 38 also shows the constraint imposed by the 802.11n standard on the maximum time alloted to process all MIMO symbols. Clearly, all the points above the constraint plane are unacceptable as they wont meet the throughput criteria. In practice we keep the constraint plane at 3000ns (rather than 3600ns), this is to accommodate 15-20% pessimism factor.

3. Power, Delay, and Area Estimation

The power consumed comprises mainly of the core power (due to switching/leakage of logic gates), and due to clock network. The technology mapped verilog netlist is analyzed for core power using Synopsys Design/Power Compiler. The technology library used was a composite current source(typical) based 45nm library from Nangate. The power consumed due to clock depends on number of flops and the geometry of the clock network. The clock network was modeled as a symmetrical mesh, the global clock network power is estimated using HSPICE. The local clock power is estimated using capacitive load due to the number of flip-flops driven by an appropriately sized local clock buffer(number of flops can be found from the mapped gate level verilog netlist).

Synopsys Design Compiler is use to estimate the critical delay of the array (retimed circuit depending on the number of pipeline stages). Area estimates are also provided by Design Compiler.

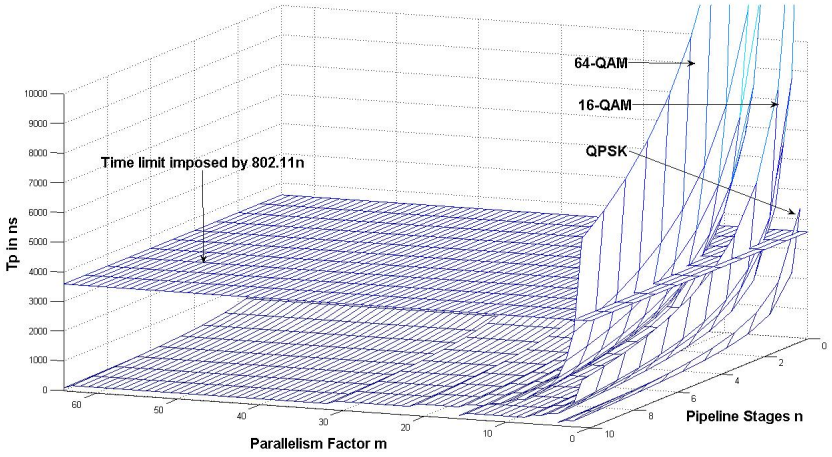

Fig. 38. Tp vs (m,k) and Constraint Due to 802.11n

4. Architectural Exploration for Low Area

In this subsection we describe our exploration procedure for low area, based on (m,k) constrained to throughput requirements of 802.11n. Fig. 38 shows the exploration space w.r.t m and k. The points below the constraint plane are called admissible points. Since 64-QAM is computationally most expensive(it takes most number of clock cycles), the admissible points for 64-QAM are admissible for 16-QAM and QPSK too. Hence, to find (m,k) for area optimized decoder we only need to meet throughput requirements for 64-QAM with minimum hardware. Fig. 39 shows variation of area of the decoder with m and k. The values of m=3 and k=8 meets the throughput requirement of 64QAM while occupying least area (Fig. 39).

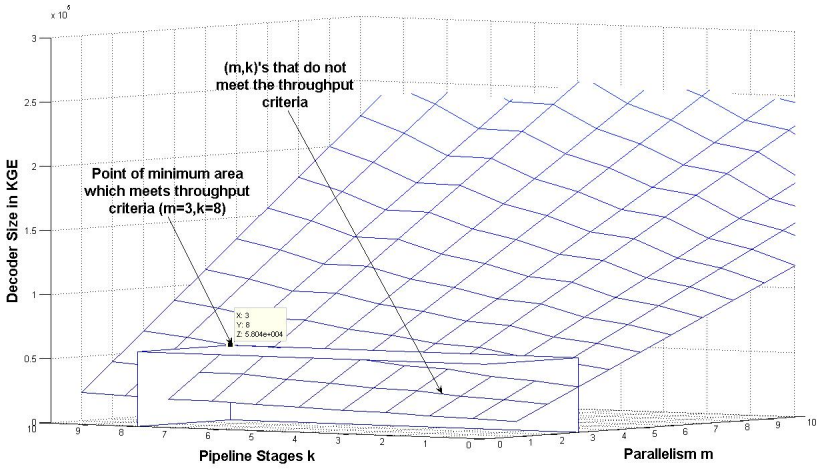

Fig. 39. Area vs. Tp for 64-QAM

5. Architectural Exploration for Low Power

Exploring the same space for power is more complicated, because different modulation schemes have different power consumption profiles while achieving the required throughput. Power consumption has to be optimized over all modes of operation. This means we have to pick (m,k) such that the aggregate power is minimized. We define aggregate power as Pow_{agg}=$Prob(QPSK) * Pow(QPSK)$+$Prob(16QAM) * Pow(16QAM)$+$Prob(64QAM) * Pow(64QAM)$, where Prob(QPSK) is the probability of the decoder being reconfigured to process QPSK MIMO symbols, and Pow(QPSK) is the power consumed by it while processing a QPSK MIMO symbol etc. Since there is no a-priori knowledge of the probabilities, we assume them to be equally likely,i.e. Prob(QPSK)=Prob(16QAM)=Prob(64QAM)=1/3.

Pow_{agg} is a function of (m,k) as shown in Fig. 40. However the point correspond-

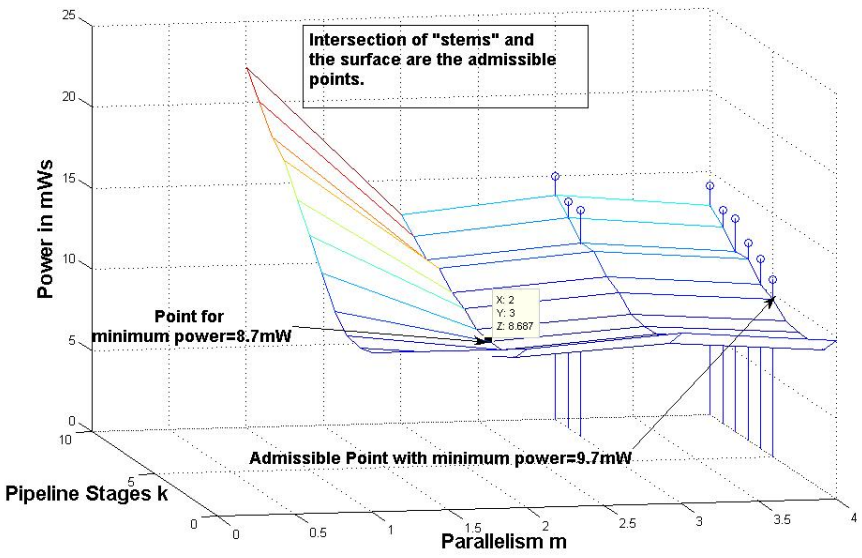

Fig. 40. Aggregate Power vs m,k

ing to the least power, does not meet the throughput criteria. Fig. 40 shows the points (using stems) that meet the throughput criteria. Hence, (m,k) for least power, needs to be searched among these points. Hence, in our power optimized design uses m=4 and k=5 corresponding to 9.7mW of power.

In Table 6 we present implementation results of our two designs (area and power optimized).

Table 6. ASIC Implementation Details

Design Parameters	Area Optimized	Power Optimized
Target Tech. Library	Nangate 45nm PDK	Nangate 45nm PDK
Pipeline Stages (k)	8	5
Parallelism (m)	3	4
Gate Equivalent	58.2k	67.7k
Power Consumption	11.91mW	9.7mW
Frequency: QPSK	38.8MHz	18MHz
Frequency: 16-QAM	116.3MHz	71.8MHz
Frequency: 64-QAM	426.6MHz	287.3MHz
Throughput Requirement: QPSK	115.6Mbps	115.6Mbps
Throughput Achieved: QPSK	155Mbps	144Mbps
Throughput Requirement: 16-QAM	231.1Mbps	231.1Mbps
Throughput Achieved: 16-QAM	310.13Mbps	287.2Mbps
Throughput Requirement: 64-QAM	346.7Mbps	346.7Mbps
Throughput Achieved: 64-QAM	465.38Mbps	430.95Mbps

CHAPTER V

CONFIGURABLE SOFT OUTPUT DETECTOR

Detectors can be classified as: hard output and soft output. In terms of the packet error rate (PER) or equivalently the frame error rate (FER), the soft detectors perform much better than their hard counterparts [25]. Soft detectors based on linear techniques such as zero-forcing(ZF) and Minimum Mean Squared Error(MMSE) are low complexity but incur high penalty in BER/FER performance. Non-linear soft detectors like Successive Interference Cancellation(SIC) are low complexity too, but provide only modest gain over their linear counterparts. Moreover, neither ZF nor SIC based receivers do well in a wireless channel with limited diversity. Authors in [57] provide excellent comparative study of various detectors in different channel conditions. From [57] it can be concluded that more sophisticated algorithms (non-linear) need to be considered for practical systems due their superior performance(especially in channels that offer little or no diversity). To get close to the optimum error rate performance researchers have proposed many algorithms, that do non-exhaustive tree search, such as List Sphere Decoder (LSD)[51], however, its complexity is still too large, and is very hard to map onto a parallel, pipelined architecture. Also, LSD converges to a solution in a random fashion making it difficult to integrate in a practical system. On the other hand, algorithms based on Breadth First Search (BFS) such as K-best, provides constant throughput but involves sorting operation which is very expensive, especially for higher order modulation schemes like 64-QAM. One of the reported implementation of a soft MIMO detector that supports 64-QAM is presented in [50].The soft detection is more complex process than the hard detection. There have been a few high speed configurable hard detectors presented in open literature [11],[44]. The configurable soft detectors have received little attention so far, some

of the notable ones are presented in [52],[53]. The detector in [52] uses Layered ORthogonal Detection (LORD) algorithm [54], while the one in [53] uses a linear MMSE approach to generate soft output. LORD outperforms the lower complexity MMSE based soft detector by a significant margin in a wide variety of channel conditions and MCS [54]. Authors in [55] also provides an excellent comparison of various detectors for various MCS and channel conditions; and concludes that the linear detectors are attractive only for channels with high diversity, low code rate and low order modulation schemes. In this chapter we present a configurable systolic-like architecture that can switch between three modulation schemes QPSK, 16-QAM, and 64-QAM on-the-fly. We also present a lower complexity algorithm and systolic like architecture for soft detection.

A. Soft Detection

For the sake of readability we briefly redescribe the system model before moving on to soft detection.

Channel Model and MIMO detection: As mentioned earlier for a MIMO system with M_T transmit and M_R receive antennas can be expressed as shown in equation 5.1.

$$\mathbf{y} = \mathbf{Hs} + \mathbf{n} \tag{5.1}$$

where $\mathbf{y}$=$[y_1, y_2, ..., y_{M_R}]^T$ is a $M_R \times 1$ received vector, $\mathbf{s}$=$[s_1, s_2, ..., s_{M_T}]^T$ is $M_T \times 1$ transmitted vector, $\mathbf{n}$ is $M_R \times 1$ zero mean complex Gaussian noise vector, and $\mathbf{H}$ is a $M_R \times M_T$-dimensional complex matrix. In this paper we will assume $M_T = M_R = 4$, unless specified otherwise.

Each entry s_i $(i = 1, 2, ..M_T)$ in the vector $\mathbf{s}$ is an η-ary Quadrature Amplitude Modulated (QAM) symbol. Each QAM symbol is constructed by *mapping* $\log_2 \eta$ data

bits onto a complex number. The objective of the MIMO detector is to estimate $\hat{\mathbf{s}}$ of $\mathbf{s}$ based on the the observation of $\mathbf{y}$ along with the knowledge of $\mathbf{H}$. It has been shown that the optimal or the Maximum Likelihood (ML) hard estimate $\hat{\mathbf{s}}_{ml}$ of $\mathbf{s}$ is given by equation 5.2 [25]:

$$\hat{\mathbf{s}}_{ml} = arg \min_{\mathbf{S}\in\Omega^{M_T}} \|\mathbf{y} - \mathbf{Hs}\|^2 \tag{5.2}$$

The above equation can be further simplified to get equations (5.3)-(5.6).

$$\hat{\mathbf{s}}_{ml} = arg \min_{\mathbf{S}\in\Omega^{M_T}} \|\hat{\mathbf{y}} - \mathbf{Rs}\|^2, and\ \hat{\mathbf{y}} = \mathbf{Q^H y} \tag{5.3}$$

Above equation can be further expanded as shown in equations (5.4)- (5.6).

$$d_i(s^{(i)}) = d_{i+1}(s^{(i+1)}) + |e_i(s^{(i)})|^2 \tag{5.4}$$

$$|e_i(s^{(i)})|^2 = |c_{i+1}(s^{(i+1)}) - R_{ii}.s_i|^2 \tag{5.5}$$

$$c_{i+1}(s^{(i+1)}) = \hat{y}_i - \sum_{k=i+1}^{M_T} R_{ik}.s_k \tag{5.6}$$

The above expressions enables the detector to compute a hard estimate of the transmitted signal. On the other hand the objective of a soft MIMO detector is to compute the *reliability* associated with each hard output bit. This reliability is expressed in terms of the Log-Likelihood Ratio (LLR) of each bit, and is defined as $L(x_{i,j}) = ln\frac{P(x_{i,j}=1|\mathbf{y})}{P(x_{i,j}=0|\mathbf{y})}$, where $x_{i,j}$ is j^{th} bit in label of the i^{th} constituent QAM symbol of $\mathbf{s}$. *Max-Log-Map* approximation of this can be expressed as [25]:

$$L(x_{i,j}) \approx \min_{\mathbf{s}\in\mathbf{X}_{i,j}^{(0)}} \{d(\mathbf{s})\} - \min_{\mathbf{s}\in\mathbf{X}_{i,j}^{(1)}} \{d(\mathbf{s})\} \tag{5.7}$$

where $d(\mathbf{s})$=$||\mathbf{y} - \mathbf{Hs}||^2$, and the variables $\mathbf{X}_{i,j}^{(0)}$ and $\mathbf{X}_{i,j}^{(1)}$ are sets of vector, with j^{th} bit in the label of i^{th} QAM symbol in $\mathbf{s}$, as 0 and 1 respectively.

One of the two terms in equation (5.7) will always correspond to the ML hard

estimate because its $(i,j)^{th}$ label has to be either a 1 or a 0. If we denote the ED $d(\hat{\mathbf{s}}^{ml})$ with just d^{ml} then equation (5.7) can be rewritten as [51]:

$$L(x_{i,j}) = d^{ml} - d_{i,j}^{\overline{ml}}, x_{i,j}^{ml} = 0 \tag{5.8}$$

$$= d_{i,j}^{\overline{ml}} - d^{ml}, x_{i,j}^{ml} = 1 \tag{5.9}$$

where $d_{i,j}^{\overline{ml}}$ is the ED of the best path with $(i,j)^{th}$ bit that is complement of the $(i,j)^{th}$ bit of the ml path. It is clear that to evaluate equation (5.7), we need to compute $\hat{\mathbf{s}}^{ml}$, d^{ml}, and $d_{i,j}^{\overline{ml}}$ for i=1,2,..M_T and j=1,2,...,$log_2\eta$ [50].

1. Layered Orthogonal Detection Algorithm

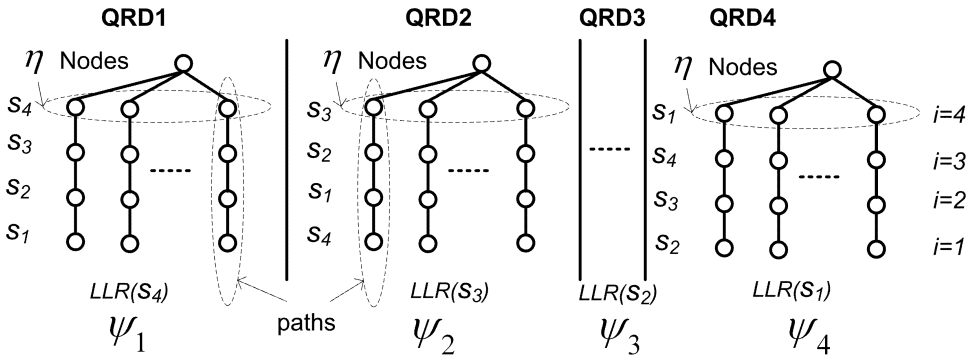

Fig. 41. The Algorithm Flow

The LORD algorithm has been proposed in [54]-[56]. In [56] the authors focus on the iterative turbo-MIMO systems. However, in the same paper they have shown that LORD (with some enhancements) performs very well even in a non-iterative MIMO system (which is the system we consider here).

LORD searches for $\hat{\mathbf{s}}^{ml}$, d^{ml}, and $d_{i,j}^{\overline{ml}}$ in a much reduced space (Fig. 41). It computes the LLRs for *individual* QAM symbols sent from four different transmit antennas. It does this by generating four sets (each of cardinality η) of paths and then searching for the required terms in these sets. A set of paths is constructed by evaluating all the children of the root node and best child of these nodes down the tree as shown. For example, to compute LLRs for the QAM symbol s_4 (transmitted from antenna no.4) it constructs the set ψ_1 of paths and searches for the LLR terms within it. Note that $\hat{\mathbf{s}}^{ml}$ is now actually a scalar $\hat{s_4}^{best}$, d^{ml} is d^{best}, and $d_{i,j}^{\overline{ml}}$ is $d_{i,j}^{\overline{best}}$ (we use superscript *best* because it is the best in the constructed set and not necessarily ML). It then permutes the columns of $\mathbf{H}$ matrix such that s_3 appears at i=4 of the tree, this entails another QRD (QRD2) computation. The above explained process is then repeated to get the LLRs for s_3. After four such iterations, LLRs for all the QAM symbols in $\mathbf{s}$ can be computed.

Authors in [56] have also suggested *metric-recycling* enhancement that significantly improves its error rate performance. The main observation is that the LLR terms computed in one set may be improved in another set. For example, if d^{best} and $d_{i,j}^{\overline{best}}$ are associated with $s_4 = a$ in ψ_1, it is possible that $s_4 = a$ (at $i = 3$) can occur in the ψ_4 with better d^{best} and/or $d_{i,j}^{\overline{best}}$. In this case the algorithm uses the better EDs to compute the LLRs. This essentially means that the enhanced LORD searches for the LLR terms in the augmented set $\psi = \psi_1 \cup \psi_2 \cup \psi_3 \cup \psi_4$, and not surprisingly its error rate performance improves.

B. On-the-Fly Configurable Soft Detector

1. Detector Architecture

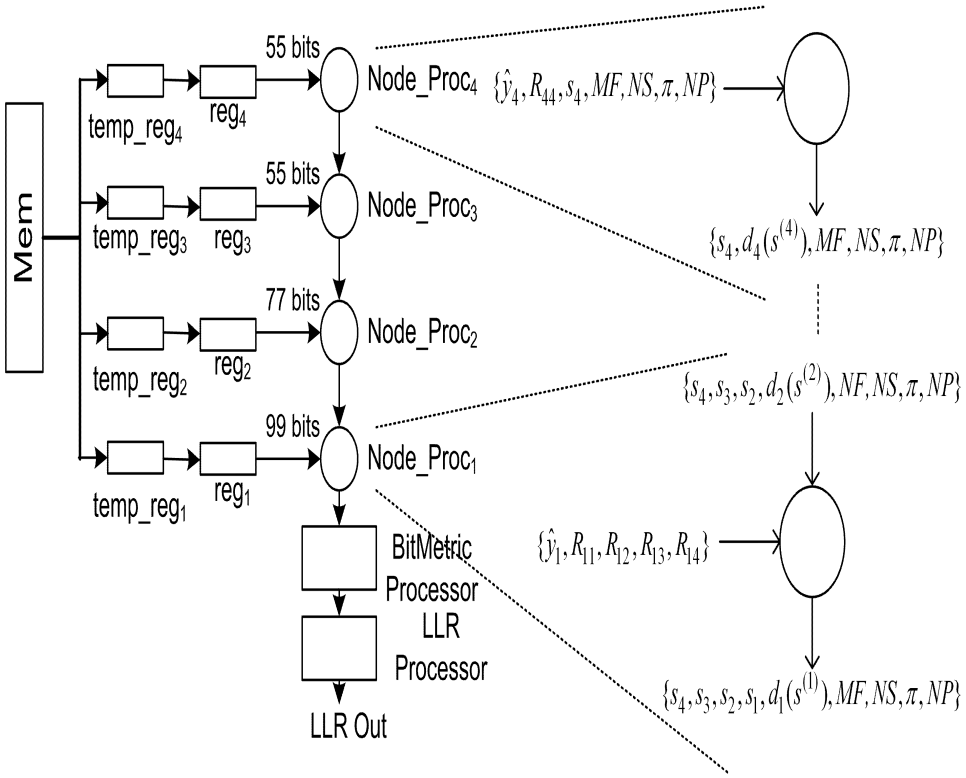

Fig. 42. High Level Architecture of the Detector

Fig. 42 shows the high level architecture of the proposed decoder. It consists of a array of tightly coupled heterogeneous processors arranged as shown. Each node processor(which is basically an MCU) is fed the data $(\hat{y}_i, R_{ik}, s_k)$ via dedicated flip-flop registers $(reg_4, reg_3$ etc.). The control instructions MF, NS, π, NP are fed through the topmost node processor. The temporary registers ($temp\_reg_4$ etc.), communicate

with memory which stores the matrix and control data for multiple received vectors. The new matrix data is loaded into the dedicated registers at every new permutation (signal NP denotes this event). π is a two bit signal that informs the detector about the permutation order. This signal is used to *de-permute* elements in $\mathbf{s}$ so that the correct QAM symbols gets compared during metric-recycling. The signal NS denotes the arrival of new MIMO symbol into the detector array. MF is a two bit signal that indicates the modulation format of the vector symbol being processed. The signal NP/NS stay high only during the first cycle of a new permutation/vector symbol. Each of the processor is pipelined to improve the operating frequency. The node processors computes equations (5.4)-(5.6) for i=4,3,2,1. These node processors switch between modulation schemes by a specially designed "slicer" (which acts on the value of MF). The slicer works in similar fashion as that of the configurable hard detector.

2. BitMetric Processor

The BitMetric processor (Fig. 43) carries out the task of bit selections and metric comparisons to compute the LLR terms. For the sake of clarity in the figures, let $a_{i,j}$ denote the $(i,j)^{th}$ bit of the current best path, $b_{i,j}$ denote the $(i,j)^{th}$ bit of the new (or incoming) path, and let $c_{i,j}$ denote $d_{i,j}^{\overline{best}}$. Furthermore, let d_a denote d^{best}, and d_b denote the metric of the incoming path. The processor operates concurrently on the data stored in memory locations $a_{i,j}$, $b_{i,j}$, $c_{i,j}$ and d_a, d_b, where $i = 1, 2, ...4$ and $j = 1, 2, ...log_2 64$. Similar to [50], the operation of bit selection and ED comparisons can be expressed as follows: If $d_a > d_b$, it means the incoming path is the new best path. Hence, for all i, j where $a_{i,j}$ and $b_{i,j}$ differ, d_a is assigned to $c_{i,j}$. This would be followed by assignments $a_{i,j}$=$b_{i,j}$, and d_a=d_b. If $d_a < d_b$, it means the incoming path cannot be a new best path. It may however, still effect the EDs $c_{i,j}$. Hence, for all i, j where $a_{i,j}$ and $b_{i,j}$ are complements of each other, the processor will assign $c_{i,j}$=d_b

if $d_b < c_{i,j}$.

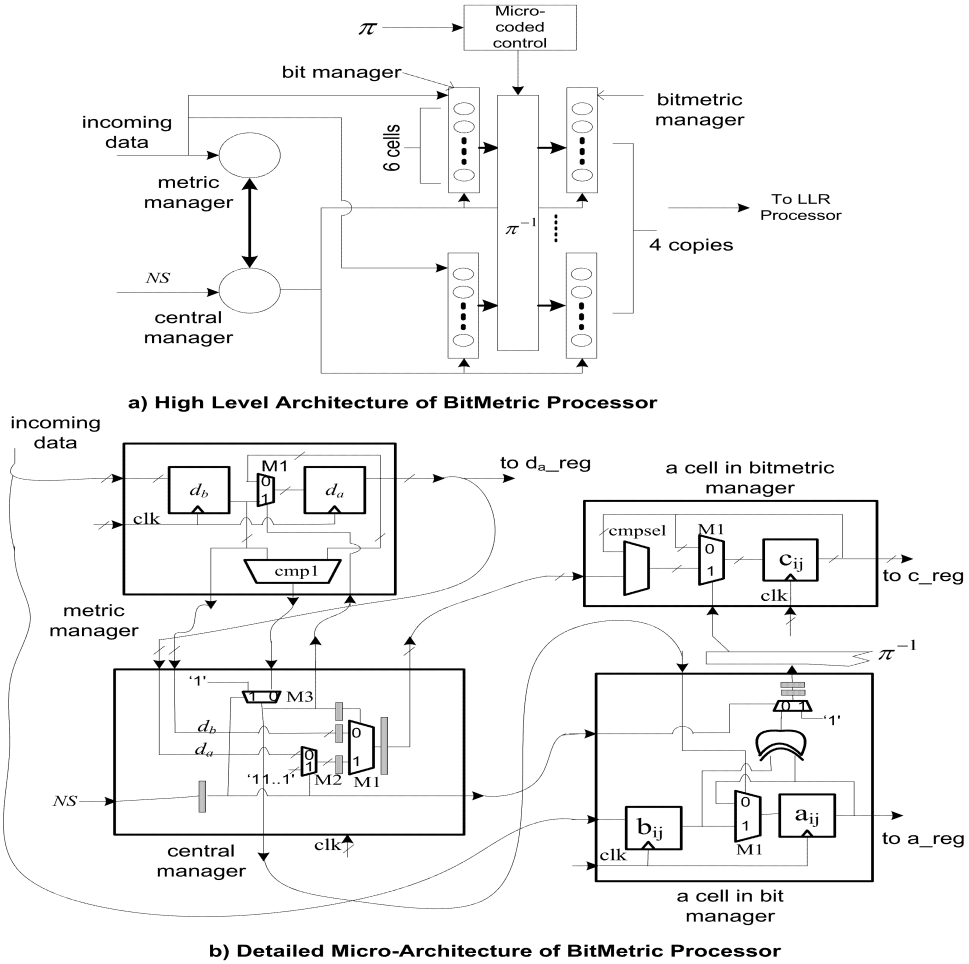

Fig. 43. BitMetric Processor

Above procedure for LLR update can be broken down into two fundamental steps.

1. Decide on the memory locations to be updated, i.e. the ordered pairs (i, j)'s: These locations are always the ones where the $a_{i,j}$ and $b_{i,j}$ differ, and hence can

be implemented using an X-OR operation (in a cell in bit manager).

2. Since (i, j)'s are now known we have to decide what to update $c_{i,j}$ with.

   (a) If $d_a > d_b$: Replace $c_{i,j}$ with d_a. Then, replace $a_{i,j}$ with $b_{i,j}$, and d_a with d_b

   (b) If $d_a < d_b$: Replace $c_{i,j}$ with d_b if $c_{i,j} > d_b$. Comparator *cmp1* checks for these conditions and instructs, via MUX M3 (in central manager), whether d_a or d_b is the candidate for replacing $c_{i,j}$. Compare select unit *cmpsel* further "filters" the candidate by comparing it with the existing value of $c_{i,j}$ (in a cell in bitmetric manager).

The π^{-1} block de-permutes the incoming bits for correct alignment of QAM symbols in different ψ_i's. The signal NS is used to enable the processing of new received vector by initializing the BitMetric processor.

3. LLR Processor

The LLR processor (Fig. 44) consists of storage elements, $c\_reg_{i,j}$ for $d_{i,j}^{\overline{best}}$, $a_{i,j}$ for bits in $\hat{s_i}^{best}$, and d_a_reg for d^{best}, where $i = 1, 2, 3, 4$ and $j = 1, 2.., log_2 64$. The processor serially reads $d_{i,j}^{best}$, $a_{i,j}$ from the addresses based on the value of MF. For example, if MF=0 then it reads from $i = 1, 2, 3, 4$, $j = 1, 2$, and if MF=2 then locations $i = 1, 2, 3, 4$, $j = 1, 2..., 6$ are read from. The final LLR values are computed using equation (5.9).

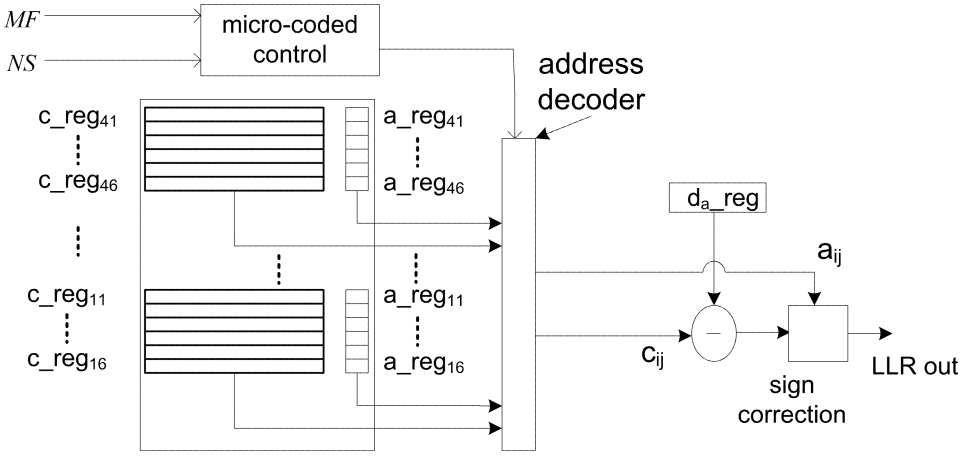

Fig. 44. LLR Processor

Throughput Analysis: Let the cumulative number of pipeline stages in node processor array be n1, n2 in the BitMetric Processor, and n3 in the LLR processor. Generally, an η-ary QAM vector symbol will spend n1+4η cycles in the node processor array, n2+4η cycles in the BitMetric processor. Finally, the serialized LLR processor will take n3+1+$4log_2\eta$ cycles to compute the LLRs. Note that, the BitMetric processor will be ready to write the data into the LLR processor every data 4η cycles. The node processors and the BitMetric processor will not stall even while switching between two modulation schemes as long as we have $4\eta_1 > n3 + 1 + 4log_2(\eta_2)$, where η_1 and η_2 corresponds to the new old vector symbols respectively. Aforementioned condition will be satisfied if $\eta_1 > \eta_2$ for a reasonable value of n3 (such as 2 or 3). This can be ensured by reading the vector symbols from the memory in their increasing modulation order.

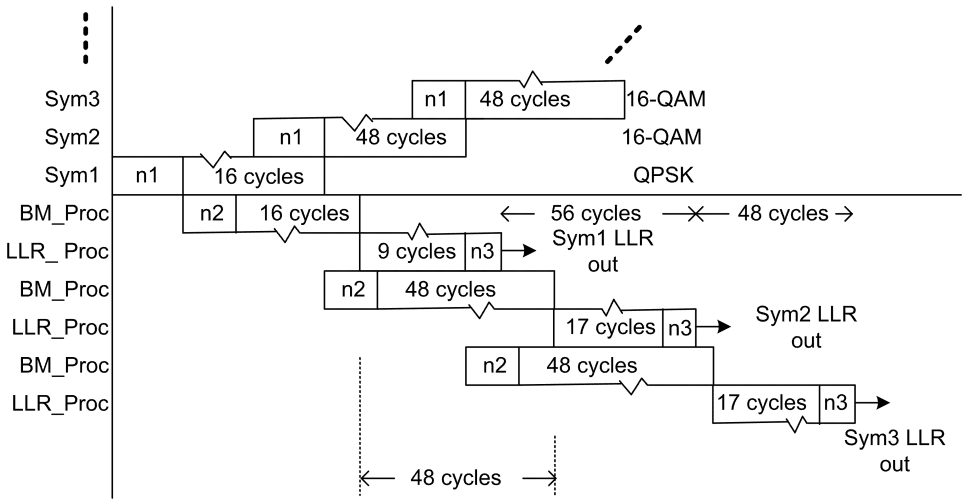

Fig. 45. Timing Diagram

The architecture has many qualities of a systolic architecture, in that the processors have mostly local connections, has continuous flow even while switching between modulation schemes, and data is read from memory only at the start of the detection process for a received vector.

Recall that every vector symbol is composed of $4log_2\eta$ bits, and it takes 4η cycles to generate LLRs in steady state per vector symbol. Hence, the throughput is given by $(log_2(\eta)/\eta)f$, where f is the clock frequency.

Fig. 45 shows the timing diagram of the detector. It is straightforward to see that the LLR values out of the LLR processor are independent of the number of pipeline stages n1, n2, n3. While switching between QPSK to 16-QAM it takes 56 cycles to compute LLRs for 16-QAM, this is because we are using a serial LLR processor. However, in steady state it takes 4η =48 cycles to generate LLRs for 16-QAM.

C. Discussion

We have made use of l^1 norm [20] to the avoid use of multipliers. Use of this norm leads to a loss of about 0.3-0.4dB w.r.t to the optimal l^2 norm (Fig. 46). The SNR gains due to metric-recycling vary from 0.9dB in case of 64-QAM to about 1.3dB in case of QPSK.

An estimate of hardware resources of the detector is shown in Table 7. Synthesis was done using Synopsys Design-Vision and mapped on 45nm Nangate library. The input data (matrix data) is 11 bits fixed point (for both, real and imaginary parts). The internal precision is maintained, which leads to word-length of 22bits for the EDs. The maximum achievable clock frequency is limited by the loop in the cell in bitmetric manager (containing 22 bit cmpsel, M1), this loop delay comes to about 2.3ns. Hence, it is estimated to be clocked at 434MHz. Assuming that the node processor array needs 55+55+77+99=286 bits every four cycles (recall that new matrix data is loaded every η cycles, and η=4 for QPSK) the memory bandwidth is upper bounded by about 9bytes/cycle. The detector achieves a throughput of 217Mbps for QPSK, 108.5Mbps for 16-QAM, and 40.6Mbps for 64-QAM can be achieved in steady state. The clock frequency can be improved further by reducing the wordlength of the EDs by using "clipping" [50]. However, the impact of clipping on the PER needs to be carefully studied for various MCS and channel conditions. Moreover, overall throughput can be scaled simply by letting multiple detector cores operate on different carrier tones in a multi-carrier system such as MIMO-OFDM. Dynamic configurability is especially useful in an OFDMA system (the tones in OFDMA is shared between multiple users transmitting at different MCS). The detector that can switch dynamically avoids incurring latency while processing different tones (potentially thousands).

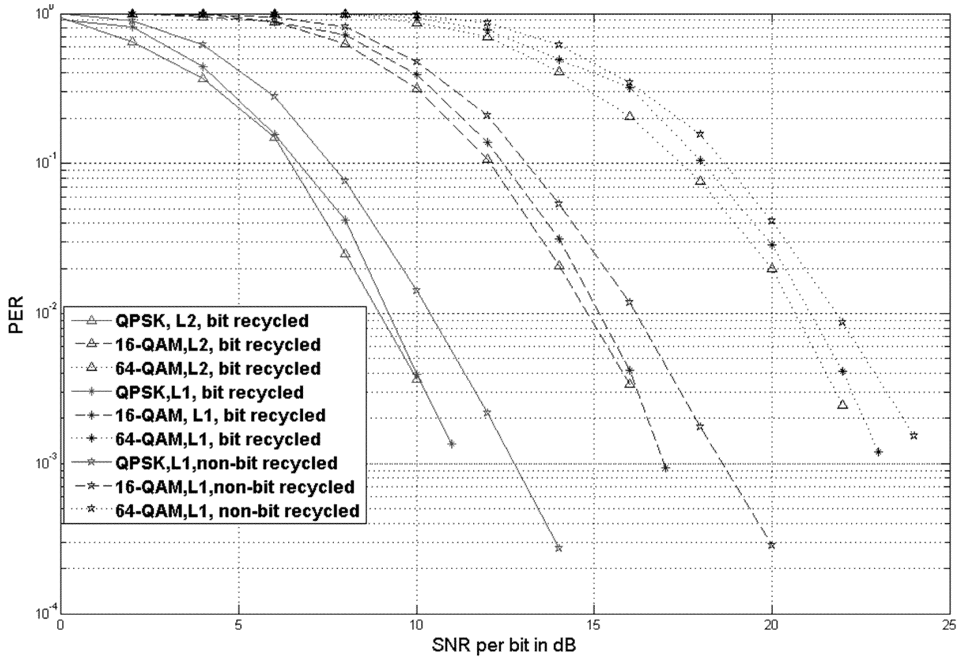

Fig. 46. PER Performance of LORD Algorithm

Table 7. Synthesis Results

	Gate Count	Storage FFs
Node Procs	10.3KGE	572
BitMetric Proc	6.5KGE	584
LLR Proc	1.1KGE	574

D. Systolic-Like Detector for High Order MIMO System

LORD algorithm does provide excellent error rate performance. It can be implemented in a highly parallel and pipelined manner and hence it is very suitable for

hardware implementation. However, it involves multiple QR decomposition operations (equal to the number of antennas) which are not only expensive (especially in a case when the channel changes quickly) but require larger memory to store the decomposed matrices. Furthermore, multiple decompositions increases the pre-processing latency, which is a concern in most upcoming wireless standards. In this section we derive a lower complexity algorithm which is inspired by COSIC and LORD algorithms.

E. Low Complexity Soft Detection Algorithm

Intuitively, the FER/BER performance of the soft MIMO detector will depend on the signs and magnitudes of the LLRs being fed to the FEC decoder. From the earlier discussion it is clear that sign of the LLRs crucially depend on the effectiveness of the decoder to get to $\hat{\mathbf{s}}_{ml}$. The COSIC algorithm is an efficient alternative for providing clsoe to *ml* hard performance, hence it is a suitable candidate for computing $\mathbf{s}_{ml}$ and d^{ml}(Because the BER performance of COSIC is close to ML, we can reasonably assume that $\hat{\mathbf{s}}_{ml}$=$\hat{\mathbf{s}}_{FSD}$ with very high probability. Hence, we treat the output of COSIC algorithm as $\hat{\mathbf{s}}_{ml}$).

In COSIC all children of the root node are processed, thereon, only their best child is extended. To compute the soft values of the associated bits we propose to use not only the *ml* path, but also the "surrounding" paths. As noted earlier that for every bit, one term in equation (5.7) is always associated with the ML path. To compute the other term we search for the paths with opposite bit and pick the one with least ED. If a path with a valid counter hypothesis is not found we simply assign the corresponding LLR, a *clipping* value with appropriate sign. Clipping is also applied to limit the maximum magnitude of the LLR.

1. Performance Analysis

We evaluated the above discussed algorithm on a block fading channel(this channel is equivalent to a channel with no diversity) of 120 information bits encoded by a rate 1/2 convolutional encoder with generator polynomial of [7,5]. Hence 240 coded bits were transmitted over which the fading matrix **H** was constant. **H** was generated independently for the next block. We counted 100 frame errors to get an estimate of FER.

Fig. 47 shows the impact of clipping value on the FER performance. We see that unlike in STS and LSD, the performance deteriorates after clip exceeds about 5. At FER of 1%, clip=3 achieves almost 3dB gain w.r.t to hard decision decoding(henceforth we refer to hard decision decoding performance a *ml* performance).

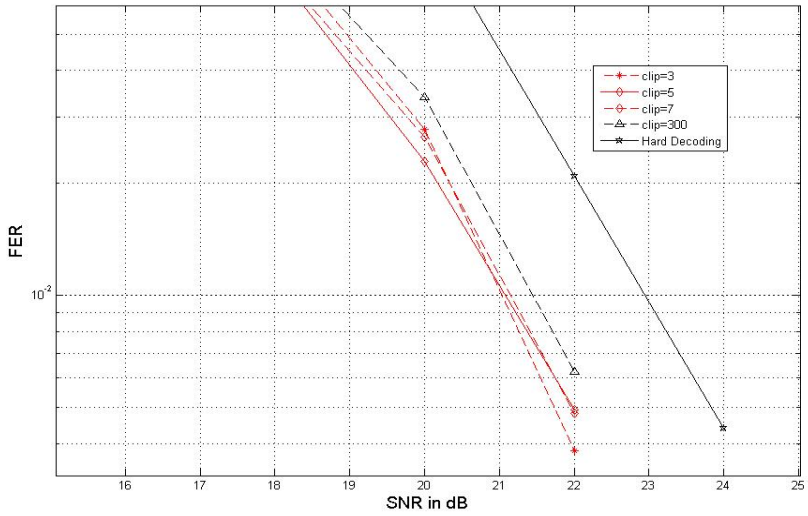

Fig. 47. FER vs SNR For Various Clipping Values

A clip=5 achieves about 1.89dB, for clip=7 the gain is about 1.8dB. To pronounce the affect of large clipping value we also show FER for clip=300, which achieves gain of only about 1.5dB. From now on we will only discuss FER/BER results corresponding to the best clipping value(will be called clip in sequel).

2. High Level Architecture and Data Flow

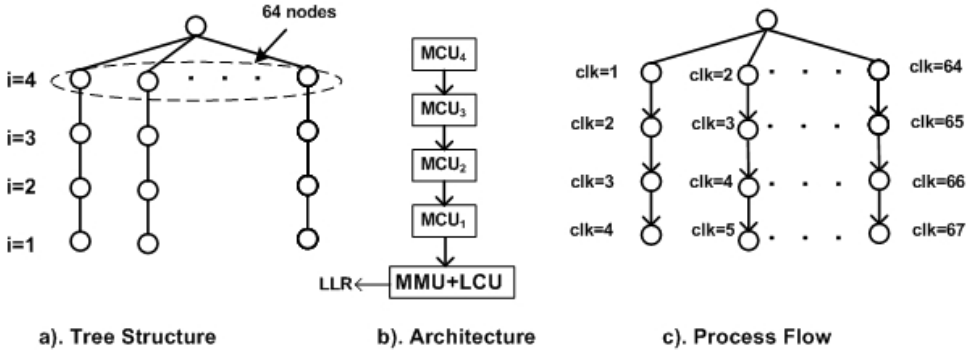

Fig. 48. Tree Structure and High Level Architecture/Process-Flow

Fig. 48 shows the high level architecture of the proposed decoder. It consists of an one dimensional systolic like array of MCUs. These units feed the Metric Management Unit(MMU), and the LLR Computation Unit(LCU). The MMU is same as the bit metric processor in Fig. 43 and LCU is very similar to LLR processor but does not support configuration.

Fig. 48 also shows the process flow of the detection process(assuming one MCU takes one cycle to process), it shows the sequence in which the nodes in the tree are processed. MCU_4 is being utilized for cycles from 1 to 64, MCU_3 from 2 to 65, and so on. Note that even though it takes 67 cycles to process one MIMO symbol, a new

MIMO symbol can be fed into the pipeline after(at MCU_4), and hence it effectively takes 64 cycles to process one MIMO symbol.

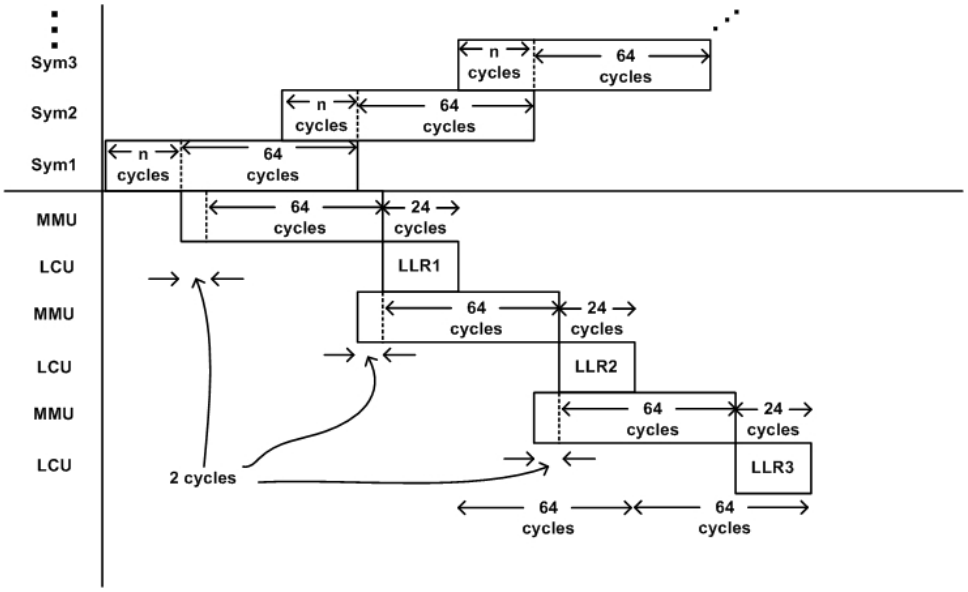

Fig. 49. Timing Diagram

Fig. 49 shows the timing details of the whole detector architecture. First n cycles are due to the pipelines introduced in the MCU array. All vector symbol Sym1,Sym2,... are processed by the array for a total of n+64 cycles. However, the array can start processing Sym2 after 64 cycles because all the top level nodes of the previous symbol would be processed within 64 cycles. Hence, baring initial latency of n cycles the total cycles needed per vector symbol is 64. The MMU operates with a latency of 2 cycles. Since the LCU operates serially it takes further 24 cycles(corresponding to 24 LLRs). The MMU takes 64 cycles to process one vector symbol and LCU takes 24, hence, the slower throughput of the LCU is effectively

"hidden". This fact is shown on negative Y-axis in Fig. 49. Overall, the whole detector outputs 24 LLRs every 64 cycles. Thus, the throughput of the architecture is given by: $\theta = \frac{24}{64} freq$, where $freq$ is the operational frequency of the architecture.

3. Node Pruning to Lower the Energy Consumption

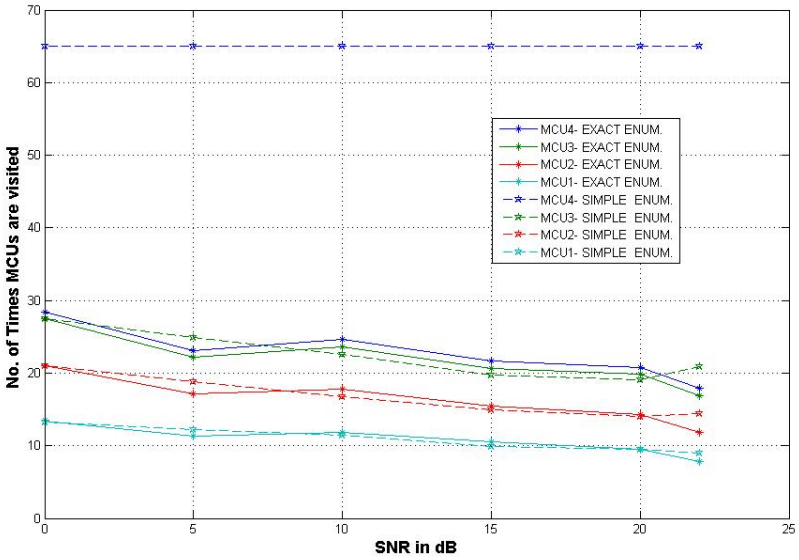

Fig. 50. Node Pruning Behavior for l^2 with SNR

As mentioned earlier, sphere decoder reduces the search complexity by updating the radius value whenever a leaf node is reached. We apply same concept to our detector, except that we use (d_a + clip) as radius. This way we can preclude(via clock gating) some MCUs from carrying out computations, thereby reducing energy consumption.

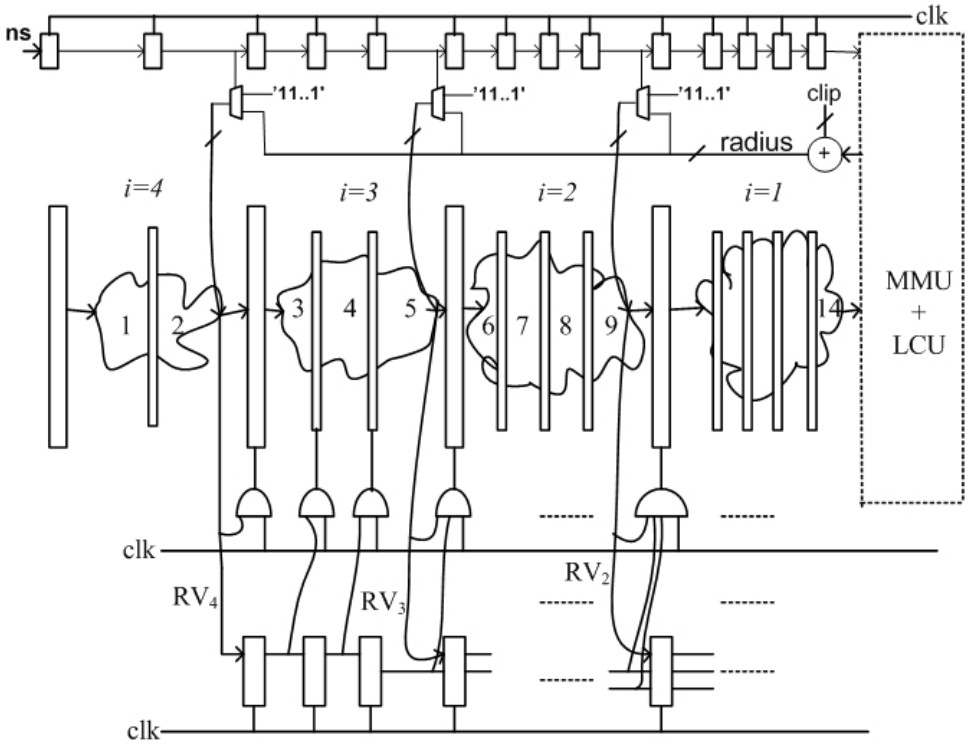

Fig. 51. Clock Gating Details

Pruning is usually most aggressive when the top level nodes in the FSD tree are processed in increasing order of their PEDs. One way to do this efficiently, is by enumerating them as described in [52] or in [44]. However, approaches in [44]-[52] are not conducive for pipelining because of the inherent loop that occur in the hardware realization of the procedure. Furthermore,they need 8 MCU_4 and a 8-way compare select unit(using approach of [52]), and 9 MCU_1 9-way compare select unit(if we use the approach of [44]), which is a huge hardware overhead. Hence, we propose a suboptimal approach to carry out enumeration. Finding the exact enumeration

on real(or imaginary) axis can be implemented using counters generating a zig-zag pattern. In our approach, we first find the zig-zag pattern the symbols on real axis and imaginary axis. We then keep the real part constant while we pick imaginary part per the pattern until the column corresponding to the real part is exhausted. We then pick the next real part in the pattern and keep it constant while we traverse its column. We do this until all the QAM-64 points are visited. Fig. 50 shows the pruning behavior for l^2 norm with clip=3. It can be seen that the pruning performance of the exact and simplified enumeration schemes is similar except for MCU_4, this is because the PEDs at the top level are not strictly increasing.Hence, we cannot use early termination and have to process all 64 nodes. Note that, the nodes lower down the tree are more computationally complex(and more energy consuming).

In hardware, node pruning can be achieved by clock gating as shown in Fig. 51. d_a from MMU is the current best metric, which is added to clip to get the radius. to distinguish between current vector symbol and the next one we use "ns" bit to drive the value of radius to a very large value('111..1'), this is to preclude the radius of older vector symbol to interfere. Each combinational cloud consists of MCU_i and a comparator to check for radius violations (RVs). RVs are basically the clock gating signals that propagate along the pipeline as shown. Note that, by doing clock gating we have introduced a loop in the MCU array. However, this loop can be run at a high speed since it has a two operand (7 and 3 bits each) adder and a 2-to-1 MUX (this delay comes to about 0.8ns based on our synthesis results). Fig. 52 shows the energy consumption per detected bit as a function of SNR. It shows the energy consumption profile with and without pruning.

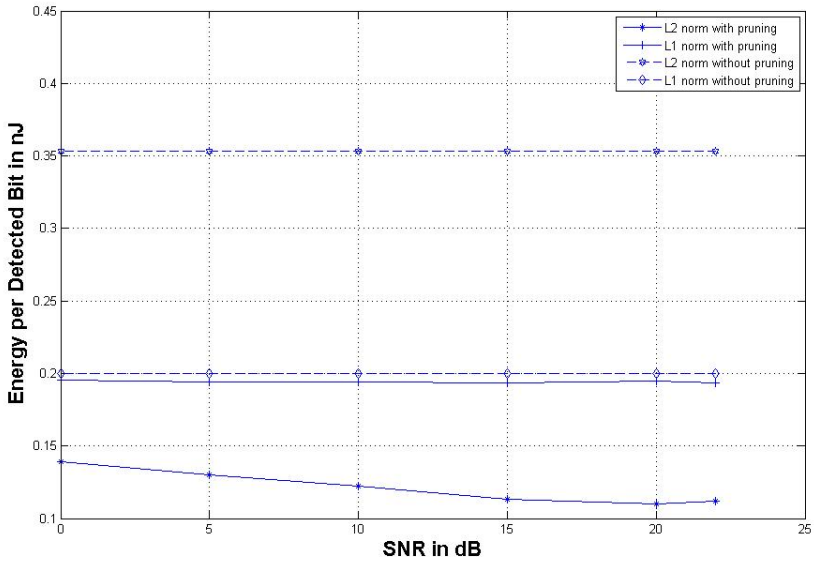

Fig. 52. Energy per Detected Bit vs SNR for l^2,l^1 Norms With and Without Pruning

F. Discussion

The FER plot for the proposed detector is shown in Fig. 53. We see that at FER of 1% the proposed detector gains almost 2dB wrt to the COSIC hard detector. Use of l^1 norm causes the FER to degrade by about 0.4dB.

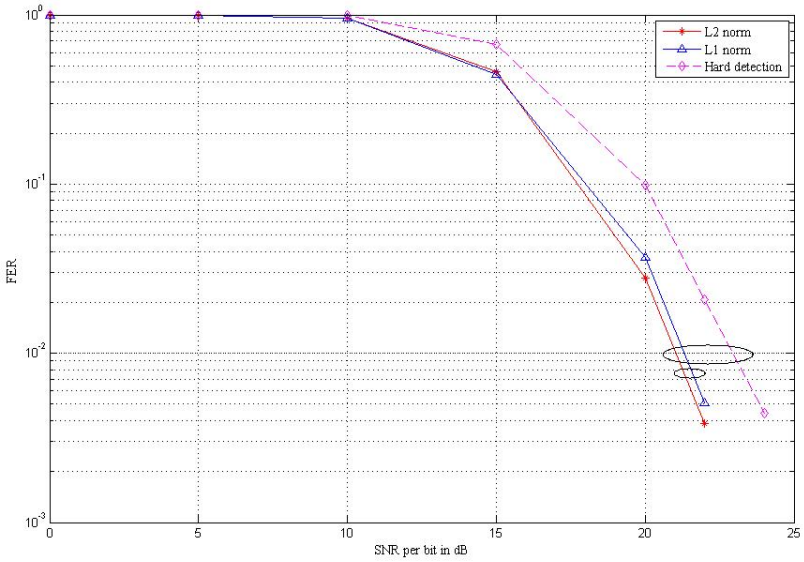

Fig. 53. FER Performance of Proposed Scheme

l^2 norm also gives better energy efficient after incorporating pruning as can be seen from Fig. 52. Use of l^2 norm, however, requires about 73% gate count more than l^1 norm (Table 8). The minimum delay in MMU is 1.5ns because of the loop, and minimum delay in MCU array is 0.8ns. We can introduce pipelines to get as close as possible to 1.5ns. Let p_i denote the number of pipelines in level i. For l^1 norm we chose p_i=7,7,6,3 for i=1,2,3,4, and for l^2 norm we chose p_i=9,9,8,4 for i=1,2,3,4. This was done in order to get close to the bottleneck of 1.5ns imposed by the loop in the MMU. The reduction in loop delay is possible because of the clipping that reduces the size of cmpsel and M1 in Fig. 43.

The RTL coding was done using Verilog HDL. Nangate 45nm CMOS standard

cell library was used for the design flow. Synopsys Design Compiler was used to synthesize the gate level net-list and to get power, area, and delay estimates.

Table 8. Synthesis Results and Comparisons

	l^1	l^2	[59]	[52]
Gate Equivalent	19.1K	33.1K	280K	70K
Power Consumption at 20dB(mW)	44.8	23.6	94	114
Energy per Bit at 20dB(nJ)	0.19	0.11	N.A	0.61
Frequency(MHz)	613.5	574.7	270	500
Throughput(Mbps)	230	215	8.57	187.5
SNR Gain wrt to hard detection	1.6dB	2dB	N.A	N.A
Tech. Library	45nm	45nm	130nm	45nm

G. Summary

A configurable accelerator architecture is presented for detecting Spatially Multiplexed (SM) data in a high order (4x4) multiple-input-multiple-output (MIMO) wireless systems. It is a customized architecture that provides soft values to the error control codes, with systolic-like data and control flow. The detector is able to switch between three different modulation schemes (QPSK, 16-QAM, and 64-QAM) without any configuration latency. RTL synthesis results indicate that a detector core uses only about 18 Kilo Gate Equivalent (KGE) in addition to around 1800 storage flip-flops. Multiple detector cores can be stacked to achieve very high throughput in a multi-carrier MIMO system. Moreover, the performance of the detector in terms of the error rate performance is excellent. The processor is very well suited for baseband processing at wireless base-stations and mobile handsets alike. A novel high speed systolic MIMO detector architecture and its ASIC implementation estimate is

presented in this paper. I lower complexity algorithm is derived based on observation from LORD and COSIC algorithms. We see that use of multiplier-less l^1 does not necessarily improve energy efficiency. This is due to more aggressive pruning when using l^2 norm. By using multiple detectors operating concurrently the throughput scales linearly with linear increase in hardware. This detector is highly suitable for MIMO-OFDM systems which offer inherent parallelism and require very high throughputs.

CHAPTER VI

CONCLUSION

In this research we have taken an approach, wherein, algorithms and hardware architectures are considered jointly. This approach has resulted in efficient detector architectures. Specifically the detector architecture, 1) has high throughput, 2) has systolic data flow, 3) is configurable to support multiple modes of operation, 4) and maintains excellent algorithmic performance (close to optimal for hard detection). We also evaluated different detector architectures under a variety of conditions.

The architectural evaluation in terms of their efficiencies is presented in Chapter III. The sequential architecture achieves a throughput of 151.18/144.34Mbps, and detector core occupies an area of 1288.33/3248.6 μm^2 (for 16/64-QAM). The Staggered architecture achieves a throughput of 319.16/221.33 Mbps at an area cost of 2143.83/4804.35 μm^2. The parallel architecture provides the fastest throughput of 646.46Mbps (for 16-QAM), but at an area cost of 10350.88 μm^2. The second implementation achieves significantly higher throughput than the sequential approach, and half the throughput of the parallel approach (using an area of much less than half of the parallel implementation). We also see that the efficiency of the architectures reduces when we attempt to achieve constant throughput using BET scheme.

Chapter IV presents architecture for a configurable hard output detector. The detector is capable of supporting QPSK, 16-QAM and 64-QAM modulation scheme for 2x2,3x3 and 4x4 MIMO systems. It has a many qualities of a systolic architecture and can be easily modified to support much higher throughput. Moreover, the configuration can be done during runtime and incurs no latency. This attribute is especially useful for wireless systems employing OFDM technology. The algorithmic performance of the detector is close to optimal. This chapter also presents a design

case study for detector design under baseband timing limit imposed by 802.11n standard. We have presented two designs that are optimized for power consumption and area for a given constraint.

In Chapter V, we present VLSI architecture for configurable soft output detector. We have developed a deeply pipelined systolic like architecture that provide high quality soft estimates at high throughput. The architecture supports detection of all three modulation scheme mentioned above. An advanced version of this architecture is then developed that supports algorithmic enhancements that improve the error rate performance by up to 1.5dB. Furthermore, a lower complexity algorithm and its hardware implementation is developed. This algorithm has complexity that is 25% of the LORD algorithm. The detector achieves a sustained throughput of 215Mbps for 64-QAM constellation and still gains almost 2dB over hard ML detection. Owing to its fixed throughput multiple detector cores can operate concurrently to efficiently extract parallelism inherent in a multi-carrier MIMO system. The overall throughput of this multi-core system scales linearly with each additional core.

Our research efforts have culminated in hardware solutions for hard and soft output detection. The soft output in our work is based on a non-iterative principle. More advanced receivers use iterative soft detection. Here the soft information is exchanged between the outer decoder and the detector. This leads to much improved error rate performance. Viability of ideas from this work need to be examined for iterative MIMO receivers. In particular, the idea of systolic like data-flow can be extended to include information exchange between the decoder and the detector. The impact of algorithmic modifications to enable this needs to be studied in detailed.

REFERENCES

[1] S. Cherry, "Edholm's law of bandwidth," *IEEE Signal Proc. Magazine*, vol. 41, no. 7, pp. 58-60, July 2004.

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

[3] EETimes, MIMO's multi-dimensional approach multiplies capacity.[Online] Available:http://www.eetimes.com/design/other/4012570/MIMO-s-multi-dimensional-approach-multiplies-capacity. Accessed Sept. 2011.

[4] M. Costa, "Writing on dirty paper," *IEEE Trans. Inform. Theory*, vol. 29, pp. 439 441, May 1983.

[5] R. G. Gallager, Low Density Parity Check Codes, Monograph, M.I.T. Press, 1963.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-correcting Coding and Decoding: Turbo-codes," in *Proc. IEEE Int. Conf. on Commun.*, pp. 1064-1070, May 1993.

[7] J. M. Rabaey,"Low-power silicon architectures for wireless communications," *Asia and South Pacifc Design Automation Conference*, Japan, pp.377-380, Jan. 2000.

[8] B.Cerato, "Design of digital architectures for telecommunication systems with special emphasys in integrated circuits for MIMO-OFDM wireless channels," Ph.D. dissertion, Politecnico di Torino, Italy, Feb. 2007.

[9] R.S. Jenkal, "Architectures and Design Methodology for Energy Efficient MIMO Decoders," Ph.D. dissertion, North Carolina State University, North Carolina, 2008.

[10] P. Bhagawat, S, Sasidharan, S. Das, G. Choi, and S. Khatri,"VLSI Implementation of a Staggered Sphere Decoder Design for MIMO Detection," *Forty Fifth Annual Allerton Conference*, Illinois, 2007.

[11] P. Bhagawat, R. Dash, and G. Choi,"Architecture for reconfigurable MIMO detector and its FPGA implementation," *Proc. IEEE International Conference on Electronics, Circuits, and Systems*, Malta, 2008.

[12] P. Bhagawat, R. Dash, and G. Choi,"Array like runtime reconfigurable MIMO detectors for 802.11n WLAN: A design case study", *Proc. IEEE Asia and South Pacific Design Automation Conference*, Japan, 2009.

[13] R. Heath, S. Sandh, and A.J. Paulraj, "Space-Time Block Codes versus Space-Time Trellis Codes," *Proc. IEEE International Conference on Communications*, Helsinki, June 2001.

[14] L. Zheng, and D.N.C. Tse, "Diversity and multiplexing: a fundamental tradeoff in multiple-antenna channels," *IEEE Transactions on Information Theory*, vol. 49, no. 5, pp 1073- 1096, May 2003.

[15] K.Su, "Space-Time Coding: From Fundamentals to the Future," First year report submitted for admission to candidacy for the degree of Doctor of Philosoph, University of Cambridge, Cambridge, 2003.

[16] A.Burg, "VLSI circuits for MIMO communication cystems," Ph.D dissertation, Swiss Federal Institute of Technology Zurich, Zurich, 2006.

[17] W. Wolniansky, G. Foschini, G. Golden, and R. Valenzuala, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. IEEE ISSSE*, pp.295-300, Sept. 1998.

[18] U. Fincke, and M. Pohst , "Improved methods for calculating vectors of short length in a lattice, including complexity analysis," Math. Compute., vol.44, pp463-471, April 1985.

[19] K.Wong, C.Tsui, R.Cheng, and M.Waiho, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," *Proc. IEEE ISCAS*, Vol. 3, pp273-276.

[20] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner,and H. Bolcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid State Circuits*, vol.40, pp 1566-1577, July 2005.

[21] Z. Guo, and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Sel. Areas in Commun.*, Volume 24, Issue 3, March 2006, pp 491-503.

[22] A.Burg, M.Borgmann, M.Wenk, C.Studer, and H. Bolcskei, "Advanced Receiver algorithms for MIMO wireless communication," *Proc. Design Automation and Test in Europe Conference*, vol 1. March 2006.

[23] L. Barbero, and J. Thompson, "Rapid Prototyping of a Fixed-Throughput Sphere Decoder for MIMO Systems," *Proc. IEEE International Conf. on Commun.*, Istanbul, Jun. 2006.

[24] C.Hess, M.Wenk, A.Burg, P.Luethi, C.Studer, N.Felber, and W.Fitchtner, "Reduced-complexity mimo detector with close-to ML error rate performance,"

Proc. of the 17th Great lakes Symposium on VLSI, 2007.

[25] B Hochwald, and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. on Commun.*, Volume 51, Issue 3, March 2003 pp389 - 399.

[26] B. Hassibi, and H. Vikalo, "On the sphere decoding algorithm. Part I: The expected complexity," *IEEE Trans. Signal. Process.*, vol. 53, no. 8, pp. 28062818, Aug. 2005.

[27] J. Jalden, and B. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Trans. Signal Process.*, vol. 53, no. 4, pp. 14741484, Apr. 2005.

[28] S. Mondal, A. Altawil, and K.Salama "Architectural Optimizations for Low Power K-Best MIMO Detectors," *IEEE Trans. Vehicular Tech.*, vol. 58, no. 7, pp. 14741484, Sept. 2009.

[29] S. Mondal, A. Altawil, C.Shen, and K.Salama "Design and Implementation of Sort Free K-best Sphere Decoder," *IEEE Trans. VLSI*, vol.PP, no.99, pp.1-5, Nov. 2009.

[30] K. Amiri, C. Dick, R. Rao, and J. R. Cavallaro, "Novel Sort-free Detector with Modified Real-valued Decomposition (M-RVD) Ordering in MIMO Systems," *Proc. IEEE GLOBECOM Conference*, 2008, New Orleans, LA.

[31] J. Jalden, L.G Barbero, B.Ottersten, and J. Thompson,"Full Diversity Detection in MIMO Systems with a Fixed-Complexity Sphere Decoder," *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol.3, no., pp.III-49-III-52, 15-20 April 2007.

[32] A. Burg, N. Felber, and W. Fichtner, "A 50 Mbps 4x4 maximum likelihood decoder for multiple-input multiple output systems with QPSK modulation," *Proc. IEEE Int. Conf Electron., Circuits, Syst.*, vol.1, 2003, pp.332-335.

[33] P. Bhagawat, G. Choi, "Staggered Sphere Decoder," Technical Report TAMU-ECE-2007-13, March 2007.

[34] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," *Proc. of IEEE Custom Integrated Circuit Conference*, June 2000, pp201-204.

[35] L. Nagel, "SPICE: A Computer Program to Simulate Computer Circuits," University of California, Berkeley UCB/ERL Memo M520, May 1995.

[36] P. McGeer, A. Saldanha, R. Brayton, and A. Vincentelli, "Delay Models and Exact Timing Analysis," in Logic Synthesis and Optimization, Kluwer Academic Publishers, 1993. pp 167-189.

[37] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," UCB/ERL M92/41, Univ. of California, Berkeley, CA 94720, May 1992.

[38] B. Gamache, Z Pfeffer, and S. Khatri "A Fast Ternary CAM Design for IP Networking Applications," *12th International Conference on Computer Communications and Networks*, Dallas, TX, October 2003.

[39] E. Menendez, M Dumezie, R. Garg, and S. Khatri, "CMOS Comparators for High-Speed and Low-Power Applications," *Proc. IEEE International Conference on Computer Design*, Oct 1-4, 2006, San Jose, CA.

[40] S. Das, and S. Khatri, "A Timing-Driven Hybrid-Compression Algorithm for Faster Sum-of-Products," *IASTED Fifth International Conference on Circuits, Signals and Systems*2007, July 2-4, Banff, Alberta.

[41] P. Kogge, and H. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Transactions on Computers*, Vol. C-22, Number 8, pp 783-91, 1973.

[42] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," IEEE Journal Solid State Circuits, vol.40, pp 1566-1577, July 2005.

[43] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," IEEE Journal on Selected Areas in Communications, Volume 24, Issue 3, March 2006, pp 491-503.

[44] H. Wang, J.P. Delahaye,P. Leray,and J.Palicot,"Managing dynamic reconfiguration on MIMO Decoder," *Parallel and Distributed Processing Symposium*,pp. 26-30 March 2007.

[45] H. Wang, P. Leray, and J.Palicot,, "A Reconfigurable Architecture for MIMO Square Root Decoder," Lecture Notes in Computer Science, Reconfigurable Computing: Architectures and Applications,August 03, 2006.

[46] X. Huang,C.Liang, and J.Ma, "System Architecture and Implementation of MIMO Sphere Decoders on FPGA," *IEEE Trans. on VLSI Systems*, Vol 16, No.2, pp. 188-197, Jan.2008.

[47] S. Yazdi, and R.Kwasniewski,"Challenges in the Design of Next Generation

WLAN Terminals," *Canadian Conference on Electrical and Computer Engineering*, pp. 1483-1486,April.2007.

[48] D. Garrett,L. Davis,S. ten Brink B. Hochwald, and G.Knagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE Journal of Solid State Circuits*.vol.39, pp. 1544-1552, Sept. 2004.

[49] R. Jenkal, and W. Rhett Davis, "An Architecture for Energy Efficient Sphere Decoding," *International Symposium on Low Power Electronics and Design*, Portland, Oregon, Aug 27-29, 2007.

[50] C. Studer,A.Burg, and H.Bolcskei,"Soft-output sphere decoding: algorithms and VLSI implementation," *IEEE J. on Sel. Areas in Commun.*,vol.26, no.2, pp.290-300, February 2008.

[51] R. Wang,and G. Giannakis "Approaching MIMO channel capacity with reduced-complexity soft sphere decoding," *Proc. of IEEE Wireless Communications and Networking Conf.*, vol. 3, Mar. 2004, pp.16201625.

[52] P.Bhagawat, R.Dash,and G.Choi, "Dynamically Reconfigurable Soft Output MIMO Detector," *Proc. IEEE Conference on Computer Design*, Oct.2008.

[53] D.Wu,J.Eilert, and D.Liu,"Implementation of a High-Speed MIMO Soft-Output Symbol Detector for Software Defined Radio," *J. of Sig. Proc. Systems*, May, 2009.

[54] M.Siti, and M.P.Fitz, "A Novel Soft-Output Layered Orthogonal Lattice Detector for Multiple Antenna Communications," *Proc. IEEE International Conference Communications*, 2006. ICC '06.

[55] C.Michalke,E.Zimmermann, and G.Fettweis,"Linear Mimo Receivers vs. Tree Search Detection: A Performance Comparison Overview," *Proc. IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*,pp.1-7, Sept.2006.

[56] A.Tomasoni,M.Siti,M.Ferrari, and S.Bellini, "Turbo-LORD: A MAP-Approaching Soft-Input Soft-Output Detector for Iterative MIMO Receivers," *Proc. Global Telecommun. Conf.*,pp.3504-3508, 2007.

[57] C. Michalke, E. Zimmermann, and G. Fettweis, "Linear Mimo Receivers vs. Tree Search Detection: A Performance Comparison Overview," *IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications* ,pp.1-7, Sept. 2006.

[58] P.Bhagawat,R.Dash, and G.Choi,"Systolic like soft-detection architecture for 4x4 64-QAM MIMO system," *Proc. Design Automation and Test in Europe Conference*,DATE '09.

[59] S. Chen, T. Zhang, and Y. Xin, "Relaxed K-best MIMO Signal Detector Design and VLSI Implementation," *IEEE Transactions on VLSI Systems*, vol. 15, issue 3, pp. 328-337, March 2007.

VITA

Pankaj Bhagawat received his Ph.D. in computer engineering from Texas A&M University(TAMU) in December 2011. Prior he earned his M.S in electrical engineering from TAMU and B.E. degree from National Institute of Technology (NIT) in Tiruchirapally,India. He has worked as a teaching assistant at TAMU for a variety of undergraduate courses. His research interests lie at the confluence of hardware design and signal processing algorithms. His Doctoral work takes a joint approach to design efficient hardware for wireless systems. He was with Qualcomm Inc. in 2005. Pankaj can be reached at pankaj11@gmail.com and at: Department of Electrical and Computer Engineering

Texas A&M University

214 Zachry Engineering Center

TAMU 3128

College-Station, TX 77843-3128

The typist for this thesis was Pankaj Bhagawat.