ABSTRACT

JENKAL, RAVI S. Architectures and Design Methodology for Energy Efficient MIMO Decoders. (Under the direction of Professor William Rhett Davis).

This work focuses on the design and implementation aspects of Multi-Input Multi-Output (MIMO) decoders for multi-antenna communications. These decoders are used to determine, either optimally or sub-optimally, the bits encoded and transmitted over a wireless channel with more than one antenna. Present standards, such as 802.11n and 4G, call for systems with more than the present two antennas. Additionally, the need for future considerations of mobility along with lowered current limits of smaller technology nodes, calls for greater power awareness in the design of MIMO decoders.

The presence of multiple antennas brings with them a) an exponentially large space for a min-cost search for the solution and b) non-trivial VLSI requirements to deal with additional dimensions of the wireless channel. Additionally, the conditions under which a MIMO decoder is used would change in terms of the Signal to Noise Ratio (SNR) values. This requires considerations in the multiple axes for a decoder implementation: Power, Delay, throughput and algorithmic performance. Of the many options available, Sphere Decoding (SD) has become a popular implementation of MIMO detection due to its improved performance at lower hardware complexity in comparison with Maximum Likelihood methods for optimal algorithmic performance. ASIC implementations have proven the feasibility of this method but fail to effectively address the issue of energy efficiency (b/s/mW).

In this work, we investigate the architectural and design space of multi-antenna decoders. We show that systems that allow for tradeoffs along multiple axes are more likely to achieve energy optimality due to a changing usage environment. Multi-antenna systems are unique because they can exploit parallelism which could aid in amortizing the constraints on design. We design and implement improved architectures that exploit a combination of deeper pipelines and simple single-port read and write memories to increase the energy efficiency (bits/sec/mW) of the decoding process.

We also implement architectural modifications that increase the throughput of algorithmically optimal decoders. We use these improvements to make an argument for trading off power consumption with the final aim of improving the energy efficiency of decoding. Additionally, we also provide insights into the design of architectures that can handle an increased constellation size and increased antenna numbers in a power efficient manner.

In an effort to improve the throughput, we also provide a simple method of a block based algorithm using counters. VLSI implementation of all the architectures proposed provides the final measure of complexity in terms of power, area and throughput. The implementation of the proposed architectures in a 1.2V 90nm 8-metal IBM process demonstrates the effectiveness of the various methods proposed in reducing complexity and increasing energy efficiency.

Architectures and Design Methodology for Energy Efficient MIMO Decoders

by
Ravi Somnath Jenkal

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Engineering

Raleigh, North Carolina

2008

APPROVED BY:

Dr. Paul D. Franzon	Dr. Winser Alexander
Dr. Xun Liu	Dr. Ilse Ipsen

Dr. W. Rhett Davis
Chair of Advisory Committee

DEDICATION

To my parents Prema and Somnath Jenkal and my brother Vijey

BIOGRAPHY

Ravi Jenkal received his B.E. degree in Electronics and Communications from S. J. College of Engineering in 2002 and M.S. degree in Electrical and Computer Engineering from North Carolina State University, Raleigh, in 2004, respectively. He continued to work towards the Ph. D. degree in the MUSE group in the Department of Electrical and Computer Engineering at North Carolina State University. His research interests include the creation of novel architectural solutions for Multi-Antenna Systems-on-a-Chip (SoC), 3DIC and low-power and high- speed ASIC/SoC design methods. He is a student member of the Institute of Electrical and Electronics Engineers (IEEE) and Eta Kappa Nu (HKN).

ACKNOWLEDGMENTS

Above all, I would like to thank my parents Prema and Somnath Jenkal for being such wonderful sources of inspiration and love. They have shown me the way all through life and instilled in me the ideals of hard-work. They have shown me the means of finding joy in all I do and made sure that I am never lacking for strong ideals. I would also thank my brother, Vijey Jenkal, for being the wonderful role model that he is. I do not need to look beyond him to find a source of unending support and level headed advice. There is no greater support system that I can ask for.

I would like to thank Dr. William Rhett Davis for being a wonderful advisor and friend. His knowledge about design and tooling still astounds me. He has been an excellent source of ideas and conversation, both academic and otherwise. I would like to thank Dr. Paul Franzon for being an exceptional educator and a wonderful human being. His initiative in keeping me financially supported by a Mentor Graphics fellowship is something I will always remember. Dr. Davis and Dr. Franzon have provided all the necessary ingredients to ensure that I take a broader perspective in any problem I choose to tackle. Their knowledge in the field of design and their enthusiasm in dispensing it make them wonderful educators. It was a wonderful experience working under them as a teaching assistant as well.

I would also like to thank Dr. Xun Liu for being on my committee and providing an excellent insight into fascinating issues in Electronic Design Automation. I would like to thank Dr. Winder Alexander for being on my committee and showing me the meaning of grace and professionalism. I must thank him for pulling me out during the many times I tended to dig myself into a hole. I would particularly like to thank his feedback on my final dissertation. I would like to thank Dr. Ilse Ipsen for being on my committee and her feedback as well. I would like to thank Mentor Graphics in giving me a fellowship for my graduate studies.

The most important aspect of my work here has been my interaction with my fellow students. It has been a pleasure to interact with so many smart people. The people in 429 MRC (MUSE + ERL): Samson Melamed, Thorlindur Thorolfsson, Yongjin Choi, Bruce Su, Evan Erickson and Julie Oh, require particular mention and the lone ranger, Chris Mineo, in MRC 438. And then there is Mr Shanthamurthy Premswaroop and his company for lunches and racquetball. I really owe you guys a lot. I would also like to thank

some of the NCSU alumni: Hao Hua, Ambarish Sule, Ambrish Varma and Dhruba Chandra for being around as long as they have and helping me in my stay here at NCSU. The final acknowledgment must go to my roommates for the final years in graduate school: Mithun Acharya, Siddharth Savadatti and Anushree Mohan (almost a roommate). To come home to interact with them made it a easier to leave the pleasures of the lab in MRC. I would also like to thank my previous roommate Yathiraj Udupi.

I am deeply indebted to you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Multiple-Input-Multiple-Output(MIMO) techniques have promised much in their ability to provide increased data rates while not involving an equivalent increase in the spectral requirements. Their popularity has caused increased academic and industrial involvement in this field. Moreover, efforts are underway to standardize the inclusion of MIMO technology in the 802.11 standards and telecommunication standards under the aegis of the 3GPP working group.

At the core of the MIMO technique is the need for high complexity signal processing for the detection process to determine the transmitted vector at the receiver. The high complexity stems from the need to handle matrices of complex values at the receiver. Thus, to ensure the feasibility of the MIMO technique, these matrices need to be processed efficiently in VLSI.

1.1 MIMO: Perspectives

This section focuses on the motivation and reasoning for Multiple-Antenna systems. It also provides a high level view of the technology and its workings. It is important to view this technology keeping in mind the market and integrated circuit design based pressures.

1.1.1 Market Perspective

The wireless industry is perennially in need of techniques that utilize spectral resources better. Quite simply, there are three central driving forces for this [1]: a) There is an ever increasing demand for raw data rates to cater to emerging applications like video, gaming etc. b) There is a continual drive for the ubiquity of wireless devices to penetrate all areas of application c) The spectral resources available are not getting any cheaper.

The number of wireless data connections in North America[1] (and hence the average required data rates) has shown the trend seen in Fig. 1.1

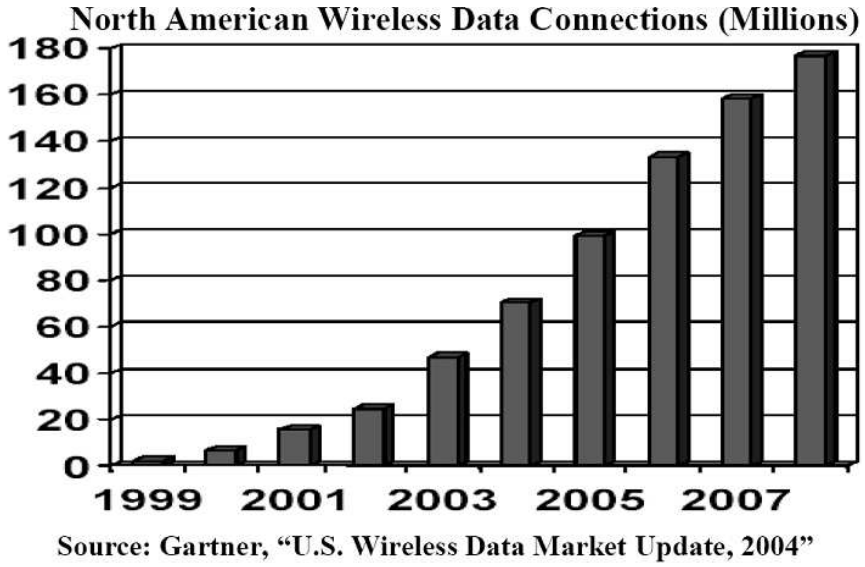


Figure 1.1: Number of wireless data connections in North America

Considering the above trends for data requirements, there is always a need for techniques that utilize spectrum better. A good measure of this is the **Spectral Efficiency** of a given technology expressed as the data rate supported, for a given Signal to Noise Ratio (SNR) specification, for a given bandwidth *i.e.* bits/sec/Hz. A higher spectral efficiency allows for greater data rates at a given bandwidth and hence for a constant spectral cost. It is in catering to this requirement that MIMO techniques have generated interest. MIMO systems promise an increase in capacity without a corresponding increase in bandwidth and

[1]Source: Gartner,"U.S. Wireless Data Market Update, 2004"

its concomitant cost.

Moreover, multi-antenna systems form an integral part of the 4G (or Beyond 3G) standard. The 4G standard calls for, among other things, high spectral efficiency, high data-rates and interoperability between standards [2]. This in turn calls for solutions that can span multiple constellations, multiple antenna numbers, multiple data rates etc. Achieving hardware solutions for these is an essential part of the move towards this new standard.

1.1.2 Technology Perspective

Quite simply, the term Multiple-Input Multiple-Output (MIMO) is derived with respect to the channel that passes data. In case of MIMO systems, there exist multiple antenna elements at both the transmit and receive sides. A typical setup is shown in Fig. 1.2. The ability of MIMO systems to successfully increase spectral efficiencies comes from two important factors:

- The Spatial Dimension: This dimension is introduced by virtue of the separation of the transmit and receive elements. This separation provides the possibility of a diversity (multiple independent copies received) previously not exploited.

- Independence by fading: The possibility of transmitting independent streams of data and receiving these by virtue of the independence of the fading on each stream. This independence is achieved through uncorrelated fading of the environment. Any correlation in fading would lead to a dependence between the channel coefficients of different receive-transmit antenna pairs. This adversely affects the ability to invert the channel and therefore the data rates that can be supported for a given algorithmic performance requirement.

In general, the algorithms for MIMO detection focus on the maximal exploitation of one of the two features. One feature is the provide an improved quality of transmission by exploiting the presence of more than one antenna and redundancy in transmission. The other feature is to send independent streams of data on each transmit antennas and hence increase the effective data rate. For high data-rate communications, the aim is to exploit the latter to the fullest in an aim to achieve maximum throughput. The set of algorithms that attempt to do this are called **Spatial Multiplexing (SM)** decoding algorithms [6].

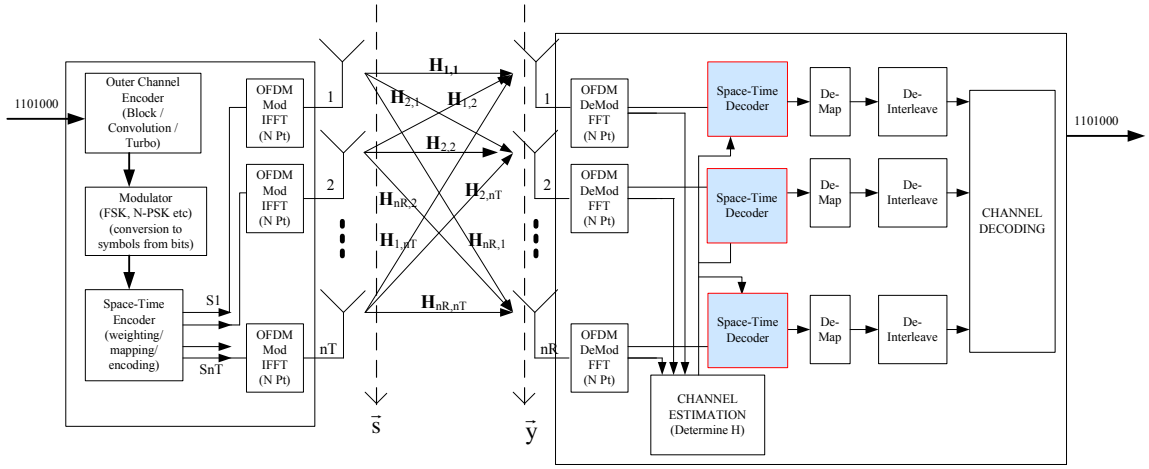


Figure 1.2: Typical MIMO System View

1.1.3 Integrated Circuits Perspective

- The SoC Paradigm: There is, presently, an unprecedented availability of die-area for use by circuit designers. This is in keeping with the predictions of Dr. Gordon Moore per Moore's law, which states that the number of transistors (components) in a given die area would double every 18 months. The availability of such immense die areas has allowed integration of multiple functional blocks on a single piece of silicon. Presently, it has become a viable prospect to include processors, memories, application specific blocks, peripherals, memory and bus controllers etc. on a single piece of silicon. All is not well in this respect though. Unfortunately, the rate of availability of transistors has not been matched by the **productivity** of the designers. This implies that rate of growth of the number of transistors that can be designed effectively has not kept pace with the rate of availability of silicon to be used. This, therefore, leads to the **Productivity Gap**. This lack of design productivity, and its subsequent effect on design cost, has been identified by ITRS [42] as the single greatest threat to the growth of the IC design industry. One way of dealing with this has been to throw more engineers at the design end. Another method of amortizing this problem is the introduction of the Intellectual Property concept. Here, the design problem moves from one of fundamental design to one of system-level design and integration. To

increase design productivity and have longer product life-cycles, there is a need for system components that are domain specific. This manifests itself, in the case of MIMO system implementations, in the form of multi-standard and multi-dimensional tradeoff supporting implementations. The multi-standard support, could be achieved both by hardware-based fine grained reconfigurability or software programmability-based course grained reconfigurability.

- Power constraints of design and technology: One of the major problems of scaling to smaller technology nodes(*i.e.* smaller transistor lengths) has been the worsening power issues in IC design. Being smaller, the transistors do consume reduced power but on the flip-side have higher transistor density and speed. The increase in speed and density manifests itself as an increased overall higher chip power consumption and, therefore, more generated heat. Heat, in turn, can cause chip failures by virtue of thermal runaway issues.

  Another important aspect of designing with a power perspective is the determination of optimality. This optimality is to be measured in terms of energy-efficiency (bits/sec/mW) for performing a computation. Focusing only on reducing power, for example in a portable system, might cause failure to reach throughput constraints. Conversely, focusing just on performance leads to increased power consumption. Therefore the aim must be to either a) maximize the energy efficiency for a given throughput constraint or b) maximize the computation for a given energy budget.

- VLSI non-triviality of MIMO implementations: The design of systems for MIMO decoding suffer from high complexity. This mainly arises from the need for matrix based computations and the need for complex arithmetic. The need for complex arithmetic manifests itself in a quadratic dependence of design size on the number of antennas [29]. For a given area constraint, the basic MIMO decoder algorithms suffer from very high complexity and to make matters worse suffer from the problem of scalability i.e. the rate of growth of complexity of the MIMO algorithms is greater than Moore's Law as seen[2] in Fig. 1.3. To further exacerbate the problem, the rate of growth of the battery capacity is extremely low. This is of particular importance to portable devices and hence necessitates a new look at the decoding algorithms. In essence for a wireless

[2]Taken from [1]

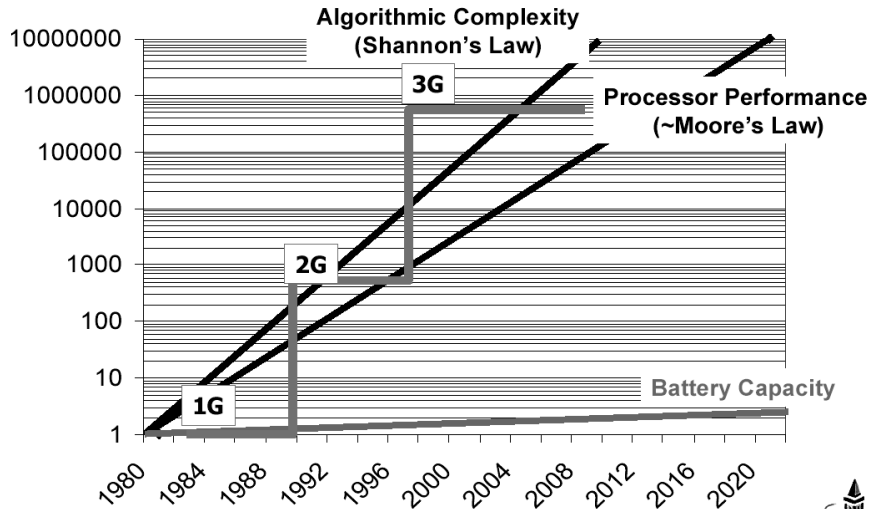


Figure 1.3: Scalability and Complexity of MIMO Algorithms [1]

product, such as the decoders we are building, the constraints of optimizing designs are multi-fold in energy efficiency, performance, area, and flexibility/reconfigurability.

1.2 Motivation and Contribution

1.2.1 Motivation for this work

The main motivating factors for this work are:

- **Power Awareness and minimization**: Present works in the design of MIMO decoders do not actively attempt to minimize power consumption. This is of primary importance given that the intended application of the algorithm is in portable devices and since present technologies are leading to increasing power densities.

- **Need for run time dynamic behavior**: Given that the environment and standards and applications for the MIMO systems tend to be varied, the requirements for the systems themselves would be different. Under such constraints we would need an architecture capable of handling a wide operational range in terms of power, speed and algorithmic performance.

- **Tradeoff conducive Sphere Decoder**: Also, to bring algorithmic performance tradeoffs into the picture, the depth-first search Sphere Decoding algorithm has shown itself to be ideal [31]. This is because it provides for an increased possibility for algorithmic tradeoffs over large ranges of algorithmic and architectural limits. There are multiple tradeoffs possible in MIMO algorithms with respect to algorithmic performance (BER), area, energy efficiency (bits/sec/mW) and speed. This is made clear in [31] where tradeoffs between speed, performance and area are considered for a block-based spherical decoding implementation. For a given environment or a given application requirement one of these can be traded off for the other. As an example, for low SNR conditions we would be better off going for a just a less complex zero-forcing V-BLAST solution over other methods [9].

- **Pipelinability as a tradeoff mechanism**: As seen in [40] pipelining provides a means of catering to multiple requirements (speed versus power) and a method for easy shift in design requirements. It has been stated that the sequential spherical decoding algorithm that is being used in this work is inherently non-pipelinable by virtue of its iterative data-dependant nature. We would like to provide a solution that enables active pipelining.

- **VLSI non-triviality of SD solutions (SoC and Market requirements)**: Sphere Decoding is still a task of non-trivial complexity and silicon requirements. Moreover, techniques such as Adaptive Modulation and Coding call for a reconfigurable and scalable solution, which, in turn, also leads to a lowering of the design costs. This is of central importance to increase the design productivity as identified by the ITRS [42]. There is therefore a need for a solution that scales well with the number of antennas and also provides support for different constellations while not compromising on power, area and performance.

The result of this sort of optimization and tradeoffs call for modifications of both the architecture and algorithm. To this end, there is a need for a methodology that allows for early analysis of the design and its tradeoffs at multiple levels of implementation.

1.2.2 Contributions

This work aims to proposes an architectural solution for Sequential Sphere Decoding that enables operation at a wide range of power, delay and environmental constraints. We make an initial analysis of the major sources of power consumption in the implementation of Sphere Decoders. We use these results to increase the energy efficiency of MIMO decoding implementations. Moreover, we analyze the scalability and re-usability of the Sphere Decoder architecture. The resultant contributions of this work are:

- The analysis of the relative power contribution of the processing elements of the spherical decoding process.
- The systematic exploration of the tradeoff space for Sphere Decoders in terms of power, area, delay, throughput and algorithmic performance. The information obtained from the above exploration is exploited to determine the axes of tradeoffs for different constraints. The sensitivity to different constraints helps provides a guide for a more directed optimization of the system.
- The creation of a framework that allows for a simple architectural exploration of the Sphere Decoder tradeoff space using SystemC. We use this framework to analyze the architectural space to determine solutions that would improve the throughput and reduce unnecessary computation.
- The analysis and implementation of architectural and algorithmic optimizations that allow for a drastic improvement of energy efficiency for optimal decoding with l^1 constraints. This is done using a combination of computational minimization in the datapath and and a smart traversal of the search l^1 space.
- A new architecture for energy efficient Spherical Decoding using a combination of deeper pipelining and simplified memory access schemes. This architecture has shown the ability to work at a clock frequency of 495 MHz(pre-place and route) and 390 MHz (post-place and route) and at a supply voltage of 1.2V (typical). The implementation supports throughputs of about 336 Mbps or 590 Mbps (datapath retiming) at an SNR of 20 dB. The BER performance at this operating point is about 10^{-3} based on simulations at 20 dB SNR for a 16-QAM constellation. The energy consumption at this operating point was found to be 23 mW at 6ns (pre-place and route).

- We provide insights on the ability of the architecture to work in a wider range of operating conditions. This ability enables the usage of this architecture in wider range of applications than before and makes it easier for the core to be integrated in a complex SoC.
- The implementation of an improved block-based algorithm for enabling run-time tradeoffs for improved BER performance under throughput constraints.
- Provision of insights into the constraints for scaling the number of antennas and the constellation size.
- The design, fabrication and testing of a self-testing and debuggable Spherical Decoding solution is explained. The architecture provides insights into the tradeoffs feasible with the scaling of voltage for different SNR constraints. The implementation was done using an IBM 90nm 8-metal CMOS process. The chip is one of the few functioning implementations of sphere decoders presently available.

1.3 Organization of document

The dissertation is organized as follows: Chapter 2 provides an overview of options available in the design of MIMO Decoders. We provide explain why Sphere Decoders are the minimum power solution in the power and throughput space. We also touch upon the reason for power awareness in system design and the methodology to be followed for optimality to be achieved. Chapter 3 provides an overview of high level decisions that need to be made for building Sphere Decoders. We also provide the detailed architecture of the implementation of choice. We also provide the final results of the chip designed using this base architecture and show the improvement in pipelined implementations. In Chapter 4, we formalize analysis of the sphere decoder space using results from the implemented architecture. We also propose a SystemC^{TM}based framework for analysis of the throughput-performance space using cycle accurate models of the decoder. We propose architectural modifications, what we call the **sorting** and **2-phase** modifications, in an attempt to minimize the search space for a 4x4 16-QAM system. Chapter 5 exploits the symmetry of the l^1 cost space to shrink the memory and datapath. Additionally, we look at the problems of implementing solutions that can scale with the number of antennas in both cycle time

and power consumption. We prove the ability of this improved architecture in providing a low-power solution. In Chapter 6, we show that the expanded search space with increased antenna and constellation sizes can only be traversed using algorithmically sub-optimal implementations when facing constraints of throughput. Additionally, the imposition of constancy in throughput and its ramifications in terms of performance is looked into. We propose a block based solution for decoding which is used to schedule cycle limits on the decoding process. Chapter 7 focuses on the design and implementation of a 1.2V 90nm 8-metal CMOS Time multiplexed Sphere Decoder. The implementation was done with features for increased visibility into the design using scan chains and the ability to load test vectors of different SNR values. The process of implementation from RTL to GDS2 is explained and the thoughts behind testing this implementation using a combination of a) sending in noisy test vectors with handshaking and b) use of scan chains for yield checks and debug is explained. Chapter 8 concludes this analysis and provides future directions that could be taken in the implementation of power aware systems for MIMO decoders.

Chapter 2

Literature Review and Related Work

This chapter sheds light on the progression of implementations of MIMO Decoder systems. A brief history of the technology is presented which shows the move towards the multi-antenna case from the initial single antenna based communications. Moreover, this should provide a feel for where the technology is progressing in terms of requirements and complexity. We shall be focusing specifically on Spatial Multiplexing (SM) Techniques and their implementations for the most part.

2.1 MIMO: History

Multiple-Input-Multiple-Output communication is born out of the presence of multiple antenna elements at both the transmission and reception sides of a communication channel. The channel between the transmit antennas and the receive antennas takes the form of a matrix. The signal path between each transmit and receive antenna pair is a channel coefficient in the matrix.

This technology was a result of the move towards the use of multiple antennas in communications. The number of antennas was increased to exploit a newer form of diversity called **spatial diversity**. **Diversity** refers to the reception of multiple **independent** copies of the same information which allows a more reliable determination of the original transmitted information. Spatial diversity comes from the separation between the antennas

transmitting signals and/or due to the separation of the antennas at the receiving side as well. The reduced probability of all paths present being lost to deep fading leads to a reduced loss of information and improved reliability.

Initially, techniques like **smart antennas** [6] provided multiple antennas at either the transmit or the receive side. These techniques initially provided for only receive diversity or transmit diversity using methods like Maximum Ratio Combining [27] or Space-Time-Block-Coding [27] [23] [6]. These methods were successful in providing diversity gain which improved the reliability of the communication channel and hence reduced the BER. **Diversity gain** is the ratio of the SNR with diversity to SNR without diversity. It is also defined as the slope of the BER curve in a log-scale. For the above multiple antenna cases, it is seen that if there are N independent paths for a given signal (*i.e.* there are N antennas), the maximum diversity gain is N. This is because, in theory, the average error rate decays as $1/SNR^N$ with N antennas as opposed to $1/SNR$ for the single antenna case [8]. At a fundamental level, these techniques used diversity to combat fading.

When multiple antennas were used **both** at the receive and transmit side it was determined that **Spatial Multiplexing Gain** is also obtainable. Spatial Multiplexing Gain implies, for M transmit antennas and N receive antennas, it is possible to recover $\min(M,N)$ different signals. This will be explained shortly. None of the other forms of diversity can promise such a gain. The implication of this gain is the possibility to transmit $\min(M,N)$ independent signals at the transmitter and successfully receive them. This leads to an increase in data rates in proportion with $\min(M,N)$ and also provides a maximum diversity gain of MN. **It must be stressed that the fundamental assumption for the above gains to be achieved is the independence of the fading between all transmit-receive antenna pairs**. In this case, fading turns from foe to a friend because of uncorrelated fading leads to a channel matrix with good condition which results, in turn, in the independence and separability of the incoming data streams.

A comparison of present schemes at the algorithmic level is presented very well in [27] and has been repeated in Table. 2.1

Table 2.1: Summary of Performance of Basic MIMO Algorithms [27]

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	$\leq M$	$\approx\leq \min(M,N)$

2.2 Survey of Present Implementations

We will now attempt to explain the connection between algorithmic and architectural decisions in the implementations of SM systems. Due to the multiple antennas between the transmitter and the receiver, we model the channel as a matrix, $\mathbf{H}$, of size $n_R \times n_T$. The signals at the transmitter and receiver are treated as complex (column) vectors, $\mathbf{s}$ (one symbol for each antenna) and $\mathbf{y}$ respectively. In all the cases considered $n_R \geq n_T$ where all the entries are complex. Additionally, the elements of H are Gaussian i.i.d variables. If H is $n_R \times n_T$, then the expected value of $log(cond(\mathbf{H}))$ is about $log(n_T)$ [66]. This suggests that the condition number of $\mathbf{H}$ is likely to be proportional to n_T. Hence H tends to be well-conditioned with respect to inversion, provided n_T is not too large. Thus, we can write:

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

where $\mathbf{n}$ is a size a $n_R \times 1$ column vector of noise associated with each receive antenna. The reception of $\mathbf{y}$, a $n_R \times 1$ vector, at the receiver necessitates the determination of the symbol vector $\mathbf{s}$ that was sent. The symbol vector is a $n_T \times 1$ vector. The matrix view for this process is shown in Fig. 2.1. The process of determination of $\mathbf{s}$ is called **Detection**. Successful detection requires the knowledge of $\mathbf{H}$ (the estimate of the channel matrix) which is manipulated, if needed, so as to be usable for the detection process. In all our analysis it is assumed that the channel $\mathbf{H}$ has been estimated correctly.

The interest in multiple-antenna configurations at both transmit and receive sides

$$\begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_{nR} \end{bmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} & . & . & h_{1,nT} \\ h_{2,1} & h_{2,2} & . & . & h_{2,nT} \\ . & . & . & . & . \\ . & . & . & . & . \\ h_{nR,1} & h_{nR,2} & . & . & h_{nR,nT} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ . \\ . \\ s_{nT} \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ . \\ . \\ n_{nR} \end{bmatrix}$$

Figure 2.1: The matrix model for a system with n_R receive and n_T transmit antennas

began with the theoretical proof of the concept independently by both Teletar [3] and Foschini [5](who also considered more practical aspects of multi-antenna implementations). They proved conclusively the possibility of linear increase in the capacity of MIMO systems with the minimum of the number of transmit and receive antennas. Since then, there have been attempts to build systems that could exploit the presence of multiple antennas.

At the highest level, the implementations of SM algorithms have followed the below sequence in chronological order:

- **Linear Equalization (LE)** based detection: A single solution is achieved by attempting a channel inversion and multiplication type operation. This method calls for channel inversion which is very susceptible to numerical error. Therefore, the results from this implementation were found to be severely sub-optimal BER performance, especially at lower SNR values. On the positive side, this implementation was very low power.

- **Ordered Successive Interference Cancellation (OSIC)**: This is a recursive LE based operation. The result of the detection of one symbol in the vector of symbols is considered interference for the symbols being decoded after it. The decision made on one symbol is used to aid the detection of other symbols by canceling its contribution to the values signal values at the other antennas. Though better than the LE method, this algorithm is still sub-optimal with increased implementation complexity and increased power consumption.

- **Maximum Likelihood Detection (MLD)**: This method exploited the feasibility of increased clock rates to perform an exhaustive search among all the candidate

symbol vectors for minimal cost vector. The aim was to determine the vector with the minimal Mean Square Error (MSE) or a similar minimum cost variant.

- **Square Root Algorithm**: Improves the complexity and numerical stability of the OSIC algorithm. Additionally, the square root algorithm also provided a better decoding solution but still remains sub-optimal.

- **Sphere Decoding (SD) (or Lattice Decoding (LD))**: This is an algorithmically optimal detection method. It constrains, if needed, 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.

2.2.1 Linear Equalization Based Detection:

An estimate of the transmitted data vector $\hat{\mathbf{s}}$ is calculated as $\hat{\mathbf{s}} = \mathbf{G}\mathbf{y}$, where $\mathbf{G}$ is an equalization matrix. The detected data vector is then obtained as $\overline{\mathbf{s}} = \text{Quant}\{\hat{\mathbf{s}}\}$, where *Quant.* represents the component-wise quantization according to the symbol alphabet used. To understand quantization better, let us take the example of a 16-QAM constellation seen in Fig. 2.2:

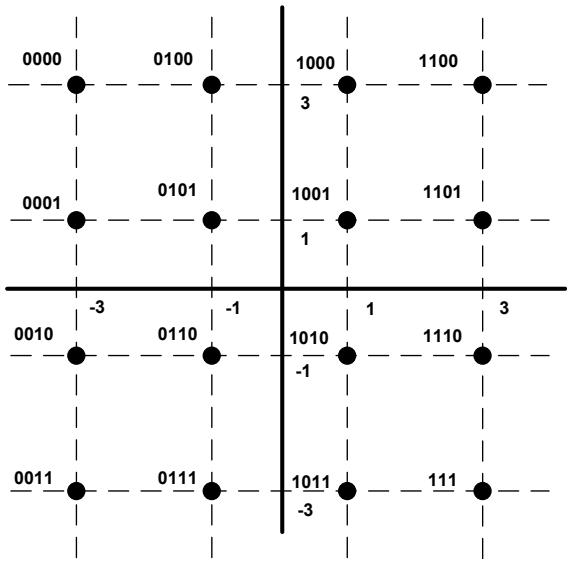


Figure 2.2: The 16-QAM constellation that is going to be used for the 4x4 decoder design

The $\mathbf{s}$ values in equation (2.1) are taken from this constellation. Message bits are taken 4 at a time and mapped to a symbol $\mathbf{s}$ from the above constellation using the mapping described in the Fig. 3.13. For example, 0111 would map to the symbol $-1-3i$. A quantizer function *Quant* performs the reversal of this process. The $\bar{\mathbf{s}}$ estimates are mapped onto the closest point the constellation space seen in Fig. 3.13.

The type of equalization matrix used enables classification into 2 popular implementations of this type of detection scheme: *Zero Forcing* and *MMSE*. Conceptually, these methods constitute channel inversion and vector multiplication of complex values.

The Zero-Forcing equalizer uses the pseudo-inverse of $\mathbf{H}$ which is given by

$$\mathbf{G} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H \tag{2.2}$$

The core implementation issue here is finding the inverse of the channel matrix. The algorithm has its major folly in not considering the noise in the system for its detection process. The resulting detection estimate is of the form

$$s_{ZF} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H\mathbf{y} = \mathbf{s} + \overline{w} \tag{2.3}$$

which basically corresponds to the data vector $\mathbf{s}$ corrupted by the transformed noise $\overline{\mathbf{w}}$.

The minimum mean-squared error (MMSE) equalizer is given by

$$\mathbf{G} = (\mathbf{H}^H\mathbf{H} + N_0 I)^{-1}\mathbf{H}^H \tag{2.4}$$

which minimizes the mean-squared error $\mathrm{E}\{||\mathbf{s_{ZF}} - \mathbf{s}||^2\}$. The major constraint in the design of such a system is the need for an accurate estimate of noise in the system to determine the value of N_0. If $\mathrm{N}_0 = 0$, then the MMSE algorithm becomes equivalent to the zero-forcing algorithm.

These algorithms have been found to be quite unusable for systems with low BER targets [6] [9] [30]. Moreover, stringent numerical requirements for matrix inversion algorithms, lead to hardware solutions with high circuit complexity, size and deeper critical paths [12].

2.2.2 Nulling and Canceling/VBLAST/OSIC

These iterative non-linear algorithms provide a middle path between linear equalization methods and the brute-force Maximum likelihood approach thereby providing a better BER performance at fair complexity. Nulling and Canceling uses a serial decision feedback approach *i.e.* an iterative methodology, to detect each layer separately. When a layer (a particular symbol in the vector of symbols transmitted) is detected, an estimate of the corresponding layer's contribution to the received vector is subtracted from the received signal and the result is then used to detect the next layer. Thus, once a layer has been detected it is considered as interference for the other layers and is hence canceled out. This is where this algorithm differs from the Zero-forcing approach where only nulling was performed. The nulling and cancellation of symbols continues iteratively until all the layers are detected. To detect a given layer the layers that have not been detected are nulled out or equalized. The choice of the equalization leads to ZF-VBLAST and MMSE-VBLAST as the most popular choices for the nulling-and-canceling approach [25] [26]. It was also found that the order of nulling and canceling affects the performance of the algorithm greatly.

Let us consider the case of ZF-VBLAST in greater detail to analyze the numerical complexity issues. MMSE-VBLAST differs from ZF-VBLAST in the nulling step. For the case shown below, H is a channel matrix of dimension $n_R \times n_T$ with $n_R \geq n_T$ as stated earlier.

In Algorithm 1, $H_{k_i}^{+}$ denotes the matrix obtained by zeroing columns k_1 , k_2 ... k_i of H and $^+$ denotes the Moore-Penrose pseudoinverse. Also, $(G_i)_j$ implies the choice of the j^{th} row of matrix G_i and y_i denotes the i^{th} element of the vector $\mathbf{y}$

The algorithm involves the recursive computation of the pseudo-inverse (also called the Moore-Penrose Matrix inverse) to perform the computation of the nulling vectors. The total number of iterations equals the number of transmit antennas present. The difference between MMSE-VBLAST and ZF-VBLAST lies in the computation of the nulling vector as discussed in the previous subsection.

Impressive spectral efficiencies have been seen during prototyping in the order of 20 - 40 b/s/Hz when using 8x12 configurations and SNRs of the order of 34 dB. These configurations are, however, highly impractical in real world situations. The performance of these systems are still sub-optimal given that they do not exploit the available diversity

Algorithm 1: **Successive Interference Cancellation**

Initialize
- i=1;
- $G_1 = H^+$ where H is the channel matrix. and H^+ represents the pseudo-inverse of the matrix
- Find the row in $G_1 = H^+$ with the minimum norm i.e. $k_1 = \text{argmin}_j ||(G_1)_j||^2$

while $i \leq NUMANT$ **do**
{until the time where all the antennas is dealt with}
- $w_{k_i} = (G_i)_{k_i}$, i.e. the k_{ith} row of the G matrix is taken as nulling vector
- $\hat{s}_{k_i} = w_{k_i}^T y_i$ i.e. the decision vector is created by simple matrix multiplication
- $\overline{s}_{k_i} = Quant(\hat{s}_{k_i})$; obtain the estimated symbol by quantization
- $y_{i+1} = y_i - \overline{s}_{k_i}(H)_{k_i}$; remove the effects of the decoded symbol from the received vector
- Replace the k_{ith} column of H with 0 i.e. cancellation is done, and determine pseudo-inverse, $G_{i+1} = H^+_{\overline{k_i}}$
- Among the non-zero rows of $G_{i+1} = H^+_{\overline{k_i}}$ find the row with the minimum norm i.e. $k_{i+1} = \text{argmin}_{j \notin \{k_1 \ldots k_i\}} ||(G_{i+1})_j||^2$
- $i = i + 1$

end

in the system effectively. Moreover, they also suffer from error propagation due to the fact that an error made in the determination of one symbol affects the other symbols as well. The matrix inversion requirement still remains a problem. But it is found that, for lower SNR values, the ZF-VBLAST and MMSE-VBLAST show performance comparable to the optimal case but this is lost as the SNR values go higher [9]. Additionally, other researchers have suggested that under certain environmental conditions, the VBLAST and zero-forcing algorithms are susceptible to numerical error [27]. Also, given that these algorithms tend to be relatively independent of the constellation size, they are of interest to higher order modulation systems [30].

2.2.3 Maximum Likelihood Detection

Maximum Likelihood detection is an approach that searches the space of possible solutions to find $\mathbf{x} = \arg\min_{\mathbf{x} \in X} \mathbf{J} = \|\mathbf{Hx} - \mathbf{y}\|^2$. If the constellation has 2^B constellation points then each symbol encodes B bits. Thus if there are n_T transmit antennas, each

transmit vector would be of length $B \times n_T$ bits. Hence, worst case, we need to cycle through all the $2^{B \times n_T}$ bit combinations (called candidates) of the transmitted symbol vector to constitute the space X. The governing equations for this algorithm follow:

1. Creation of the cost function Mean Square Error (MSE) (call it $\mathbf{J}$) for a given candidate vector that is being considered (say $\mathbf{x}$). MSE is given by

$$\mathbf{J} = \|\mathbf{Hx} - \mathbf{y}\|^2 \tag{2.5}$$

$$\mathbf{J} = \sum_{i=1}^{n_R} \left[|(\mathbf{H}x)_i - y_i|^2 \right] \tag{2.6}$$

2. If the system calls for it, the creation of the LLR (Log Likelihood Ratio) vector. This calls for the recognition of the minimal MSE value for each position in the candidate transmit vector for it being 1 and 0. Mathematically this equates to:

$$LLR(i) = min\ \mathbf{J}_{b_i(x)=0} - min\ \mathbf{J}_{b_i(x)=1} \quad \text{i = 1 to } (n_T \text{ * B}) \ \forall x \in X \tag{2.7}$$

   where X is the set of all the transmit vector candidates; b_i is the i^{th} in the symbol vector.

The most important advantage of this method lies in the absence of matrix inversion. All the computations are multiplicative and additive. Therefore, provided sufficient bits are provided, a minimum cost solution will be found which as that vector x which gives the minimum cost when equation (2.5) is evaluated. The absence of inversion is crucial for hardware implementations because of the absence of floating point implementations in almost all cases and the presence of integer arithmetic for all practical applications due to lower power and increased speed. Thus, a number of advantages come about due the type of computations that need to be done in case of ML decoding [21]: a) the lack of hardware costly operations like divisions, square roots or the use of CORDIC for matrix manipulations b) the absence of recursions which allows of aggressive pipelining and other architectural transformations. Additionally, there are a number of architectural optimizations feasible, like [4]:

- Processing of more than one $\mathbf{x}$ (candidate) at a time using parallel implementations.
- Use of $\mathbf{Hx}$ to create $-\mathbf{Hx}$ and therefore reduce computations.

- Use of approximate multiplication (shift and add based approach) for **Hx** since **x** is from a known constellation and hence can be approximated and stored in memory as a binary equivalent.
- simpler arithmetic leads to smaller dynamic range of the intermediate results [12].

On the flip side the issues with Maximum Likelihood algorithm implementations is the exponentially increasing complexity with the number of antennas and the constellation size. The exponentially increasing search space which necessitates massive pipelining and parallelization when applied to configurations greater than 4x4 QPSK [29] [15] [12]. This algorithm, though, is shown to be ideal for systems of low rates *i.e.* $\leq$ 8 bpcu (bits per channel used) [30].

2.2.4 Spherical (Lattice) Decoder

Spherical Decoding is an improvement over the brute-force maximum likelihood approach where attempts are made to reduce the search space for the optimal result. For the Sphere decoder the search space is reduced by means of early elimination of bad candidates combined with an incremental method of MSE cost calculation. The search for a min-cost solution starts at a certain point in space (called the center) and only considers those points at a certain "distance" (or radius r^2) from the center [30] [29].

The transmitted symbol vector is a combination of individual complex symbols from each antenna which derive the symbol values from a pre-decided constellation. It might so happen that these values might change as is the case for Adaptive Modulation and Coding (AMC) that is gaining popularity in the wireless world. But for the most part, these values would remain constant for a considerable amount of time. For example for the case we are considering $s = [s_1, s_1, s_2, .. \ s_{n_T}]^T$. The aim of the SD algorithm is to determine the final cost i.e. MSE, as a contribution of the costs of each level i.e. 1 to n_T. Thus ideally the cost criterion in (2.5) should be represented in the form

$$\mathbf{J} = \mathbf{T}_1(s^{(1)}); \text{ where} \tag{2.8}$$

$$\mathbf{T}_i(s^{(i)}) = \mathbf{T}_{i+1}(s^{(i+1)}) + \left|\mathbf{e}_i(s^{(i)})\right|^2; i = n_T \rightarrow 1 \tag{2.9}$$

In the above the $\mathbf{T}_{i+1}(s^{(i+1)})$ term accounts for the cumulative Partial Euclidean Distance (PED) at level (symbol) i from levels (symbols) $i+1$ and above in the symbol vector s , $\mathbf{e}_i(s^{(i)})$ is the distance increment (or the contribution to the cost) as we move from level $i+1$ to level i. Also, $s^{(i)} = [s^i, \ldots\ s^{i+1}, s^{n_T}]^T$ is one sequence of symbols till level i. Thus a full symbol vector would be $s^{(1)}$ which would be a candidate for the final solution. It should also be noted that each symbol in the vector $s^{(i)}$ has 2^B possible values which in this example is 16 (since we use 16-QAM). In the case of the SD algorithm this type of **incremental** cost computation is brought about by the use of QR decomposition of the $\mathbf{H}$ matrix to give $\mathbf{H} = \mathbf{QR}$. Here the $n_R \times n_T$ matrix $\mathbf{Q}$ has orthonormal columns (i.e.,$\mathbf{Q^H Q}$ $= I_{n_T}$), and the $\mathbf{R}$ is $n_T \times n_T$ upper triangular matrix. As a result of the orthonormal nature of the Q matrix, the min cost criterion modifies to become [30] (2.10) (2.11)

$$\text{For } \mathbf{J} = \|\hat{\mathbf{y}} - \mathbf{Rs}\|^2 ; \text{where } \hat{\mathbf{y}} = \mathbf{Q^H y} \text{ we get} \tag{2.10}$$

$$\left|\mathbf{e}_i(s^{(i)})\right| = |\hat{y}_i - \underbrace{\sum_{j=i+1}^{n_T} [R_{ij} s_j]}_{\text{path till that node}} - R_{ii} s_i|; i = n_T \rightarrow 1 \tag{2.11}$$

$$\mathbf{J} = \mathbf{T}_1(s^{(1)}); \tag{2.12}$$

In the above, all the multiplications and additions are in the complex domain. The result of such a cost calculation structure is the modification of the cost computation into a tree-search type flow as seen in Fig. 2.3. The level of the tree is determined by the number of the symbol/antenna (1 symbol per antenna) being detected at a given time. The tree search structure is because of the combination of 1) the calculation of the total cost for the entire symbol vector one symbol at a time and 2) the presence of multiple options, *i.e.* 2^B possible next nodes in the tree, for further move down the cost computation flow. The formulation of the problem as a tree search does not reduce the search space given that we are, at present, still going to visit all the possible nodes in the search space. An important aspect of this computation is that the cost increment in (2.11) is always going to be positive, which implies that costs only increase as we move down the depth of the tree. Therefore, for a given path being traversed in the tree, if a certain limiting criterion is breached by the cost at that node, all the lower nodes along that path would also breach

the cost. This is a very important feature that must be exploited for systems that search for minimum cost. This feature prunes the tree search at unfavorable using a criterion called the the *sphere constraint* or the *radius*.

$$\mathbf{T}_i(s^{(i)}) \leq radius^2 \tag{2.13}$$

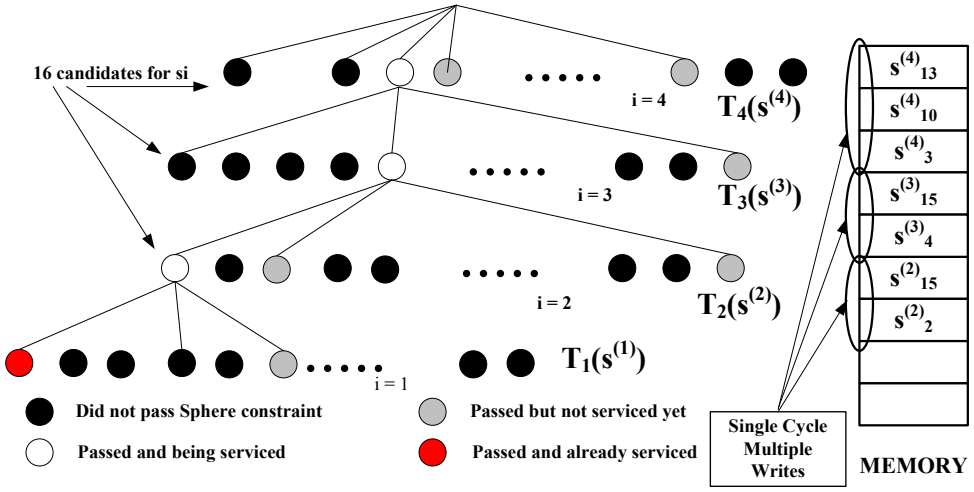


Figure 2.3: Status of nodes in tree at intermediate stage of search for a 4x4 16-QAM system

The topmost node of the tree corresponds to the center of the search. At level 1, the first symbol is considered using (2.13) with i = n_T. A node is kept for further consideration if the constraint is met and no longer considered (and hence all its children too) when the constraint is no longer met. The removal of nodes from further consideration reduces the search space. Thus, if say we reject 10 of the 16 nodes by proper radius constraints at the topmost level (i.e. 3) of the tree search in case of 16-QAM, we reduce the search space to $10 \times 16^3 = 40960$ out of $65K$ nodes from being considered. This enables the handling of constellations and antenna numbers previously thought to be prohibitive while allowing for either optimal or sub-optimal performance choices by virtue of the manipulation of the radius and the number of survivors. It is the traversal of this tree and the evaluation of the survivors that we focus on in this work.

The pseudo-algorithm for detection is presented as Algorithm 2 (for n_T levels and 4 bits per symbol). Please note that the term s_j is the j^{th} element of vector $s^{(i+1)}$ which

corresponds to the vector of symbols leading up to the current node of interest in the tree.

Algorithm 2: **Spherical Decoding Algorithm**

Initialize
- $i = n_T$;
- radius = INITRADIUS
- $memory\_addr = 0$
- $next\_node$ = RESETVALUES
- $current\_node$ = RESETVALUES
- $\mathbf{QR} = \mathbf{H}$: QR decomposition of channel matrix H of size $n_R \times n_T$
- $\hat{\mathbf{y}} = \mathbf{Q^H y}$
- $\mathbf{T}_{n_T+1}(s^{(n_T+1)}) = 0$

while *Nodes still present in search space* **do**
- {until the time where all the antennas are dealt with}
- Determine $2^B - 1$ cost increments at this level i
- $\left|\mathbf{e}_i(s^{(i)})\right|_k = |\hat{y}_i - \sum_{j=i+1}^{n_T}[R_{ij}s_j] - (R_{ii}s_i)_k| \; \forall k \in [0, 2^B - 1]$
- $\mathbf{T}_i(s^{(i)})_k = \mathbf{T}_{i+1}(s^{(i+1)}) + \left|\mathbf{e}_i(s^{(i)})\right|_k^2 \forall k \in [0, 2^B - 1]$
- Determine, $S[k]$, all survivors ($NumSurvivors \in [0, 2^B]$) based on Sphere Criterion i.e. $\mathbf{T}_i(s^{(i)})_k \leq radius^2 \; \forall k \in [0, 2^B - 1]$
- **if** $NumSurvivors > 1$ *and* $i \neq 1$ **then**
  - $\forall k : 1 \rightarrow (NumSurvivors - 1)$: Store MEMORY[$memory\_addr$] = [$i$, $T_i(s^{(i)})_k$ and $[(s_i)_k, s^{(i+1)}]$] for all other survivors
  - Final $memory\_addr = memory\_addr + NumSurvivors$ - 1
- **end**
- **if** *all nodes not finished* **then**
  - **if** $NumSurvivors \geq 1$ *and* $i \neq 1$ **then**
    - $level_{next} = i - 1$, $cost_{next} = T_i(s^{(i)})_0$, $symvect_{next} = [(s_i)_0, s^{(i+1)}]$
    - $i = level_{next}$; $T_{i+1}(s^{(i+1)}) = cost_{next}$; $s^(i+1) = symvect_{next}$
  - **else**
    - $[level_{next}, cost_{next}, symvect_{next}]$ = MEMORY[$memory\_addr$]
    - $i = level_{next}$; $T_{i+1}(s^{(i+1)}) = cost_{next}$; $s^(i+1) = symvect_{next}$
    - $memory\_addr = memory\_addr - 1$
  - **end**
- **end**

end
- Send Final decision to post processor like LLR units

At the core of this search is the determination of the cost function at a given level (16 nodes corresponding to 16 possibilities). The entire search over all the candidates at an intermediate stage of the search is shown in Fig. 2.3. Additionally, the details of node visits and processing is shown in Fig. 2.4

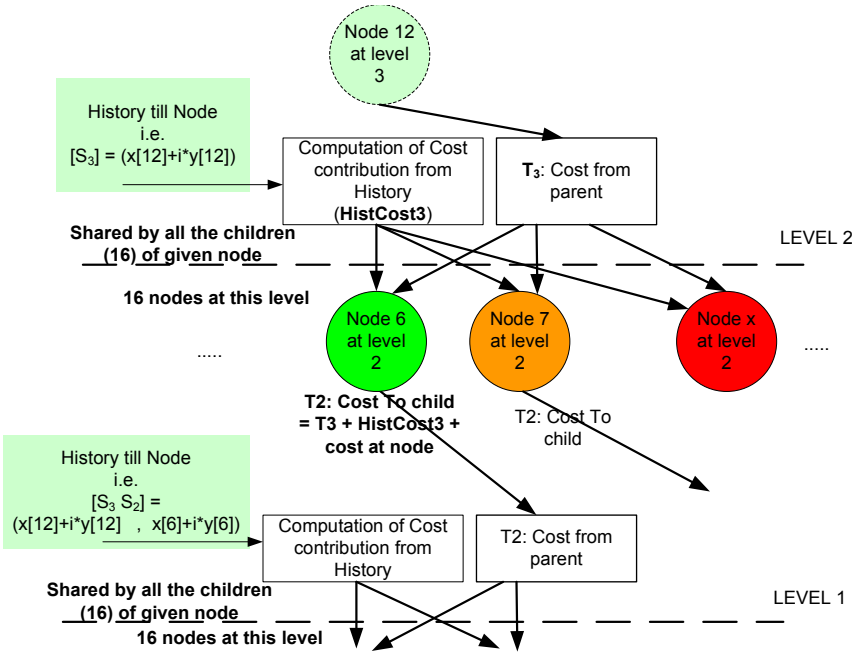


Figure 2.4: Computation performed during the processing of a given node in the tree search

This algorithm and its implementation is considered an ideal solution to the MIMO decoding problem. The implementation complexity is found to be significantly lower than the optimal ML solution [30] [29] [31] [32] [33] [34]. This algorithm still remains one of non-trivial VLSI complexity. Moreover, the SD algorithm in its sequential (Depth-first-search DFS) form has a variable throughput. This causes issues in some real-time applications. Attempts have been made to reduce the complexity of the SD algorithm by creation of variants such as the K-best [37] [35] [36] and VLSI for increased utilization of parallelism [33] [34]. The aim was to increase in the throughput of the system by use of parallelism or pipelining. The K-best approach has greater pipelinability but has a reduced BER performance especially in case of a soft-output requirement [32] and has lower average throughput. Moreover, the possibilities for tradeoffs at run time are limited in the case of K-best algorithms due to a hard-wiring of the design for a given K value. Moreover, it is seen that the Sequential SD algorithm has greater average throughput. The effect of parallelization has not been studied fully and thus no comment can be made on that front. Lately, attempts have been made to offset the variable throughput of the Sequential SD algorithm by virtue of using [31]

- Iteration Limit (Early Termination) based Sequential SD: Here the total number of nodes that are visited are fixed per symbol vector to be decoded.
- Block-based iteration limit Sequential SD: Here, the iteration limit is applied on a block of symbol vectors. The allocation of the iteration limit for a given symbol vector is based on the time taken by the previous symbol vectors in the block to reach a solution.

A study done by Burg *et. al* [30] showed that sequential SD outperformed K-best in most scenarios, providing higher throughput for a given SNR at lower silicon complexity. Of even greater importance is the ability to control the dynamic behavior of the algorithm and hence work with a changing environmental conditions.

In summary the Sequential SD algorithm has shown favorable characteristics in

1. Having greater average throughput that other decoding implementations.
2. Offering greater possibilities for run-time tradeoffs and dynamic behavior.
3. Providing close-to optimal performance if needed while not scaling in complexity like the ML algorithm.
4. Having simple operations in the cost computation which leads to a smaller dynamic range of the intermediate signals in the MSE cost computation. This manifests itself easier fixed-point or integer implementations and reduced bit widths of the datapath.

It is for these reasons that we shall be dealing exclusively with the depth-first search variant of Sphere Decoding. Present works have considered optimizations to reduce area and timing with a performance tradeoff [29–32, 43]. Consideration for power reduction and its inclusion in the tradeoffs have not been addressed.

A quick comparison of the popular methods for MIMO decoding is shown in Table. 2.2.

2.3 Power Aware Design

Many different and, in many cases, mutually destructive, forces exist in the design of present day systems. In particular the requirement for handling increased design

Table 2.2: Scalability and Complexity of MIMO Algorithms

Algorithm	BER Performance	Complexity Scaling with Antenna	Complexity Scaling with Constellation	Energy/decoded symbol vector	Throughput (4x4 16-QAM)	Application Domain
Linear Equalization	Sub-optimal	O($n_T \times n_R$) comp. mults	Independent	Low	Fixed $\approx$ 320 Mbps	Not-used(Low BER Perf.)
V-BLAST (SIC)	Sub-opt @ high SNR; $\approx$ opt. @ low SNR	O($n_T \times n_R$) comp. mults	Independent	Low	Fixed $\approx$ 320 Mbps	Large Constellations with relaxed BER perf. needs
Maximum Likelihood	Optimal; Works well with Soft-outputs	Exponential; O(2^{n_T}) comp. mult	O(2^B) comp mult.	High	Fixed $\approx$ 100 Mbps (max)	$<$ 4 Ant $<$ 16-QAM
Sphere Decoder	Optimal or Sub-optimal (user-controlled)	Exponential; O($n_T{}^2$) comp. mult per Symbol vector solution	O(2^B) comp adds per Symbol vector solution	Low-Mid (tradeoff with BER Perf. feasible)	50 - >500 (tradeoff with (BER Perf. feasible)	$>$ 4 Ant $>$ 16-QAM

complexity while attempting to keep the number of man hours the same, makes design particular difficult. It is this requirement for an increased productivity of a designer that lead to the move from full-custom solutions to semi-custom ASIC solutions. The improved productivity brings with it the need to tradeoff one or more points of optimality. Central among these would be Area (A), Delay (D) and Power (P) optimality of the implementation. The aim moved from having the best possible design to the best possible design "under considerations of time". Initial focus on area optimality for ASIC implementations have made way in recent times to a focus on power optimality. This largely has to do with the move down technology node which brings with it added pressures of needing current limitations, centrally, IR drop and heat effects. These effects have caused fundamental shifts in the way design is done. Power awareness has modified the way we look at circuit modeling, the stage of Electronic Design Automation (EDA) methodologies at which power is considered and architectural abstraction of power. Power aware designs are made much higher in the design flows and with the advent of means of encoding power intent using Universal Power Formats (UPF). Additionally, the advent of Systems-on-a-Chip (SoCs) have brought with the need for integration of blocks of disparate points of minimal power optimality. This is further worsened by the emergence of leakage as a serious implementation constraint.

The considerations of power for systems that are portable is different from non-battery operated systems. On one side, there are increasingly complex applications that portable SoCs need to support in the hope of wireless ubiquity. Additionally, wireless systems tend to have increasingly complex algorithms which need to be implemented in a

finite amount of time. In such systems, a purely high-speed view leads to an unnecessarily high power consumption. The final aim is always going to be the increase in the energy-efficiency so as to support a certain throughput while consuming a minimal energy. A fall in throughput would lead to an unacceptable fall in quality of service. This has very interesting ramifications for building wireless systems. Considerations need to be given to power optimality when making algorithmic, architectural and circuit decisions while considering throughput as well [54]. Though Application Specific Integrated Systems (ASICs) have

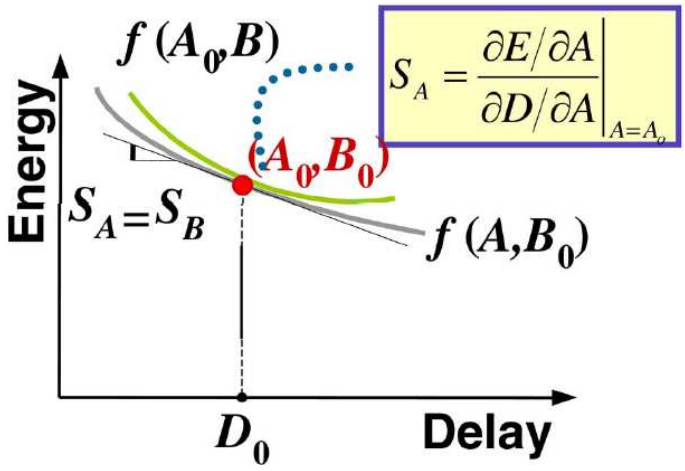


Figure 2.5: The multi-parameter optimization approach to circuit design [57]

the downfall of being sub-optimal in a Power-Area-Delay sense, improvements in EAD and fabrication have enabled a steady bridging of the performance gap. This holds particularly strongly for power [40]. Presence of multiple supply voltage and threshold voltage standard cells theoretically provide a large space for converge to the lowest power implementation. The problem lies in the usage of these cells effectively. Presently, inclusion of level converters in the EDA flow and the active usage of body bias allows designers to take greater advantage of the options available. The single greatest contributor to the increase in energy efficiency in the SoC view is the exploitation of clock gating to shut down sections of the chip that need not be active. One of the most important consequences of the emergence of leakage as a constraint is the move towards pipelining as the architecture of choice [40]. A properly designed pipeline enables the completion of a workload with the consumption of predominantly dynamic power. Multiple axes of tradeoffs like these bring about the

question of how the best solution can be determined.

What does optimality mean with power constraints? How does it change for signal processing system considerations [56] [55]? It is no longer feasible to just look at just a multiplicative minimum of constraints. Optimization is a multi-dimensional problem. Use of sensitivity analysis comes to the fore at this point. For example, the use of Lagrangian methods for finding global minimum as seen in Fig. 2.5. The minima in terms of Energy (E) and Delay (D) needs to be determined as a function of the many variables x_i (supply voltage, threshold voltage, logic family, architectural choices like pipeline depth, parallelism and pipelined numbers) and is achieved at the point where the sensitivity for each variable is the same i.e. the marginal cost in energy for the change in delay should be the same for all the variables. The problem of finding the optimum values of x_i takes the form shown in equation 2.14 where D_0 corresponds to the delay constraint and S is the Lagrangian multiplier.

$$F(x_i, S) = E(x_i) + S(D(x_i) - D_0) \tag{2.14}$$

The aim of the work here is to determine the minimum energy architectures for MIMO decoding using the above principles.

Chapter 3

Time multiplexed Sphere decoder

3.1 Introduction

To effectively design an energy efficient solution for Spherical Decoding, we need to first understand the options available for its implementation. In this chapter, we begin by going into the details of Sphere Decoder and its benefits over the previous implementations of MIMO decoders. We draw architectural conclusions from the above details and converge to the architecture of choice for an initial base implementation. We provide details of a base architecture that works in a 4x4 antenna configuration and 16-QAM constellation. The specialty of the architecture lies in the exploitation of system level parallelism and the pipelinability of computational resource in a time-multiplexed manner. This architecture is used to provide scalability in a power sense and simplifications that allow for relaxation of hardware constraints by allowing for critical operations to be performed sequentially. We then go through a combined Synopsys^{TM}, Mentor^{TM}and Cadence^{TM}tool flow for conversion of Verilog Register Transfer Level (RTL) system specification to an error free layout. The move to layout is performed with the aim of studying the loss of performance as we move to layout and to identify new constraints in design, if any, that need to be considered. We show that an architecture of this nature provides an ideal energy efficiency solution.

3.2 Sphere Decoder Fundamentals

3.2.1 Alternative Approaches to solution search

This algorithm and its implementation is considered an ideal solution to the MIMO decoding problem. This is based on the observation that the implementation complexity is found to be significantly lower than the optimal ML solution. An additional advantage of the SD algorithm in comparison with other spatial multiplexing algorithms is that, if needed, diversity gain can also be achieved in place of multiplexing gain (increase in data-rates) by pre-processing of the incoming data stream [8]. On the flip-side VLSI implementation still remains non-trivial. Moreover, the SD algorithm in its sequential (Depth-first-search DFS) form has a variable throughput causing concerns in its real-time applicability and increased complexity in interfacing to a design of this nature. As a result, attempts have been made to reduce the complexity of the SD algorithm by creation of variants such as the K-best, where a breadth first search is employed with survival of K best nodes at each level, [37] [35] [36]. The core architectural difference is captured in Fig. 3.1. Also, VLSI for increased utilization of parallelism [33] [34] have also been proposed. The essential focus of these works is the aim of achieving greater throughput using pipelining or the use of parallelism. The K-best approach has greater pipelinability but has a reduced BER performance especially in case of a soft-output requirement at smaller K values [32, 52] and has lower average throughput. Also, it is seen that the Sequential SD algorithm has greater average throughput. Additionally, the DFS algorithm allows for achieving optimal hard-output searches while K-best does not. Moreover, the possibilities for run-time tradeoffs (throughput/power/performance) at run time are limited in the case of K-best algorithms given due to a hard-wiring of the design for a given K value. [30] studied the fact that in most cases sequential SD outperformed K-best, providing higher throughput for a given SNR at lower silicon complexity. The effect of parallelization has not been studied fully and thus no comment can be made on that front. In addition, the K-best solutions proposed fail to scale well with the number of antenna given that the antenna scaling and constellation scaling decisions tend to have non-linear effects on the area (and hence power) and delay of the system [53]. To add to this, the K-best implementations lack a controllable search space which makes this a very inflexible and non-reconfigurable implementation. Lately, attempts have been made to offset the variable throughput of the Sequential SD algorithm

by virtue of using [31]

- Iteration Limit (Early Termination) based Sequential SD: Here the total number of nodes that are visited are fixed per symbol vector to be decoded.
- Block-based iteration limit Sequential SD: Here, the iteration limit is applied on a block of symbol vectors. The allocation of the iteration limit for a given symbol vector is based on the time taken by the previous symbol vectors in the block to reach a solution.

These techniques provide us further methods of enabling run-time tradeoffs in algorithm. It is for these reasons that we shall be dealing exclusively with the depth-first search variant of Sphere Decoding.

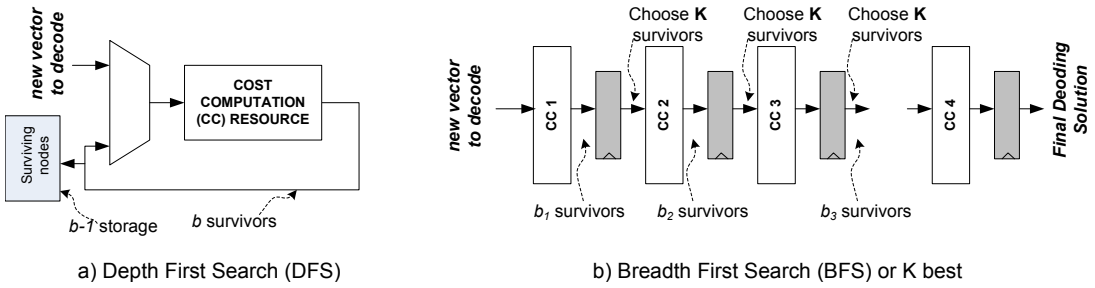


Figure 3.1: Fundamental architectural difference between DFS and K-best algorithm

Present works in the design of Spherical Decoders do not consider reduction of power consumption [29–32]. This is of primary importance given that the intended application of these systems is in portable devices and the shrinking device sizes have caused increased chip failure due to higher power-densities.. Another important aspect of designing with a power perspective is the means of determining optimality. Present works have dealt primarily with designs oriented towards high throughputs or area minimization. This optimality is to be measured in terms of energy-efficiency (bits/sec/mW) for performing a computation while meeting required performance constraints (BER @ SNR). The aim must be to either a) maximize the energy efficiency for a given throughput constraint or b) maximize the computation for a given energy budget. An additional axes for tradeoffs would be the need to search for optimality at the circuit level. An architecture that allows

for optimality to be achieved by virtue of a circuit corner would be a natural corollary to optimality at the algorithmic level. As seen in [40] pipelining provides a means of catering to multiple requirements (speed versus power) and a method for easy shift in design requirements. It has been stated that the sequential spherical decoding algorithm that is being used in this work is inherently non-pipelinable by virtue of its iterative data-dependent nature. In contradiction to this belief, we would like to provide a solution that enables use of active pipelining. Communication techniques such as Adaptive Modulation and Coding call for a reconfigurable and scalable solution. This approach also leads to a lowering of the design costs. This is of central importance to increase the design productivity as identified by the ITRS [42]. There is therefore a need for a solution that scales well with the number of antennas and also provides support for different constellations while not compromising on power, area and performance.

3.3 Architectural Perspective

In this section we aim to determine the architectural implications of the algorithm characteristics and draw some interesting inferences. Moreover, we propose a modified time-multiplexed architecture based implementation that allows for improved energy efficiency and trade-off analysis.

It is seen from the previous section that the tree-search results in a variable number of survivors per cost evaluation as per (2.13). The need for the storage of the nodes to enable later servicing therefore leads to an essential dichotomy in architecture as seen in Fig. 3.2 into a **Survivor Memory** and a **Processing Resource**(PR) for cost computation.

3.3.1 Multi-Context Processing of SD Algorithm

A fundamental issue in the present design of spherical decoders is the use of multi-ported memories to write the survivors of the *sphere criterion* to memory for later processing [29]. Moreover, the number of ports is chosen to cater to the average case and hence the design is over or under-utilized. The energy per access varies linearly with the number of ports [38] [39] while the area varies quadratically. Moreover, multi-ported structures are unsuitable for low V_{th} applications. Some interesting observations about the algorithm that could be exploited are:

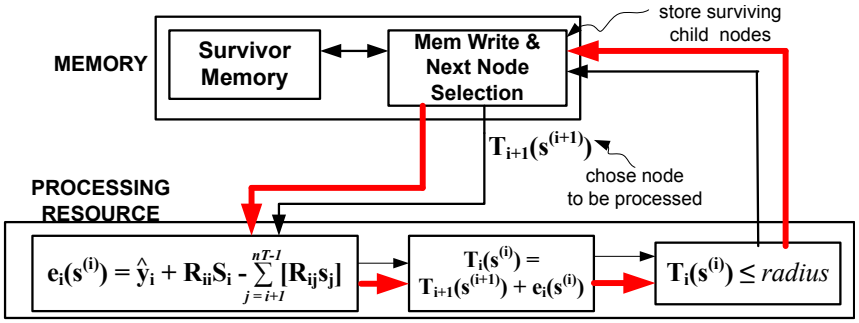


Figure 3.2: Dichotomy of Architecture for Spherical Decoding

a) **Repetitive Operation**: Each cost evaluation varies only in complexity on the basis of the level i being serviced and hence the number of multiplications to be performed. Two different phases of the cost computation exist in i)Shared Cost computation: $\hat{y}_i - \sum_{j=i+1}^{\mathbf{3}}[\mathbf{R}_{ij}\mathbf{s}_j]$, shared by all the children of a parent node, and ii)Per Child Cost computation: $= R_{ii}s_i$ which is child specific.

b) **Inherent Parallelism**: A multitude of paths exist that could be serviced in parallel. This parallelism can also be taken to the symbol vector level $\hat{\mathbf{y}}$ where multiple symbols vectors can be serviced in parallel.

The SD algorithm is inherently non-pipelinable because a) the next node to be serviced depends upon the result of the sphere criterion *i.e.* a control dependence exists and b) the final cost is the sum of the contributions from individual levels and hence there is a limit on the possible pipelining. But it is to be noted that the cost computation resources are control-free and hence deeply pipelinable within the context of the cost calculation of a single node. This feature is exploited in our architecture by combining a deeper pipeline with the servicing of multiple independent contexts. Among the choices of parallelism present, it would be judicious to attempt to exploit the symbol vector($\hat{y}$) level parallelism. This is because true independence exists at the symbol vector level since a given symbol vector can be retired and a new one fetched once it has been serviced. Note that this would not be possible while attempting to service multiple paths for a single symbol vector. In this work we aim to exploit this inherent parallelism of the algorithm in a multi-threaded paradigm to enable deeper pipelining without stalling.

To effectively exploit the above parallelism and the pipelinability of the cost computation, we modify the base architecture of Fig. 3.2 to handle multiple symbol vectors. This is done by the provision of independent symbol vector (contexts) memories that are used to store the survivors for each context. This draws from the concept of Multi-threading that allows for continued work in a pipelined scenario in case of stalls on one thread by virtue of the presence of other independent threads that can take over the utilization of the pipelined resource. This leads to a time multiplexed manner of resource usage.

Though this method leads to the overhead of management of the memories, we argue that this apparent increased complexity is compensated for by simpler memory accesses, achieved by serialization of the memory writes. This serialization is possible because a given memory has a controllable number of clock cycles before it is accessed again. Moreover, we also argue that the increase in the number of memories also allows for algorithmic modifications that would achieve greater decoding throughput thus resulting in an overall improvement in energy efficiency. Thus, the dichotomy of Fig. 3.2 takes the form shown in Fig. 3.3. As is apparent, the cost computation resource is pipelined into "P" (here 7) stages and "M" memories (0 through $M-1$). This figure shows the status of the system at an arbitrary instance of time where the numbers indicate the context (one of the M) being serviced presently. At every clock cycle, a certain number of nodes out of 2^Q possible costs survived the sphere criterion and make it out of the pipelined resource. These need to be stored in the appropriate memory and serviced later. The surviving nodes are indicated by green at the output of the cost computation resource and the progress of the writes to memory is shown as a stack of green. As can be seen, at the instant that the survivors corresponding to context 2 come into the memory interface to be written to memory, progress in writing to memory for context 1 is greater than that of context 0 which in turn is greater than context $M-1$ and so on.

Of particular importance when considering this architecture and its implications is the observation of the nature of data flow. This becomes crucial when determining the choices that would need to be made such as the stages that need to be pipelined. This is best described (for one symbol decoding process) by Fig. 3.4. It is seen that the datapath becomes wider after the Node Cost Unit given that all the children given node in the tree shall be evaluated at this point. After the cost is computed the entire set of "survivor" nodes is passed through to the memory for a) determining the next child node that will be

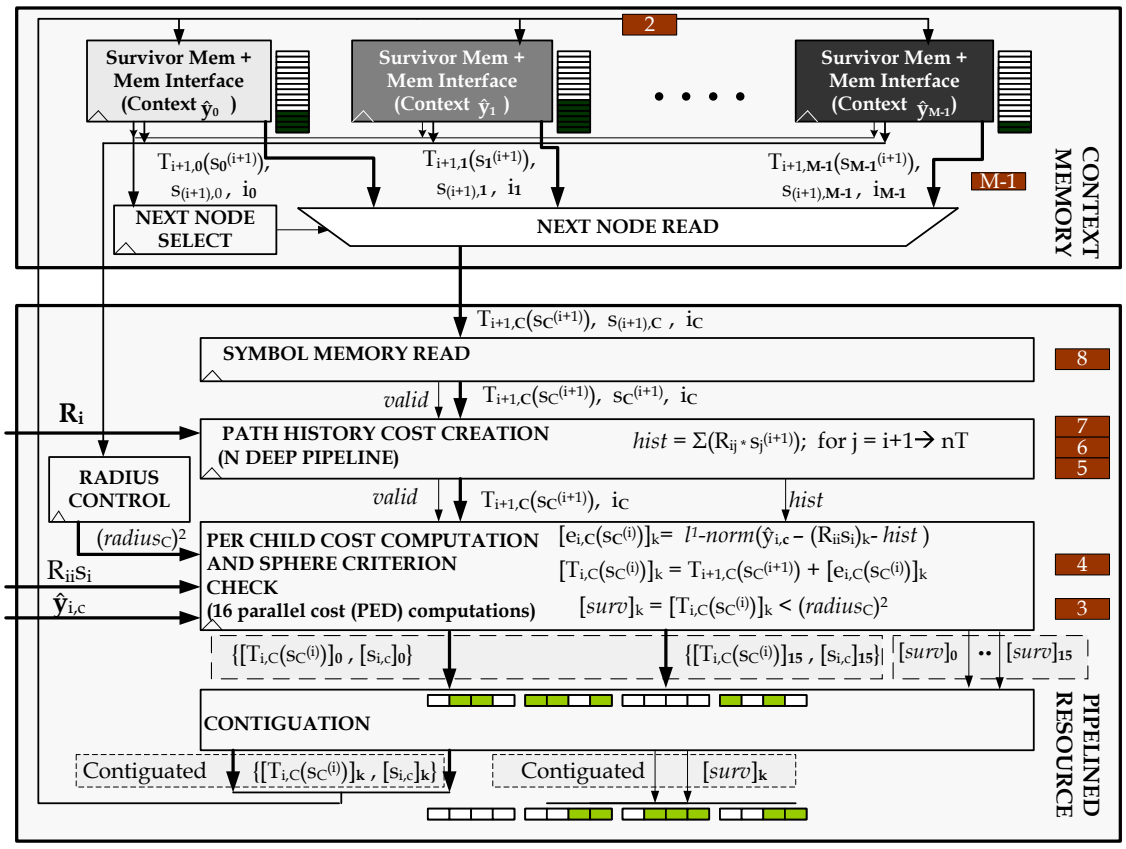


Figure 3.3: Architectural Dichotomoy for the Multi-Context Paradigm

processed and b) storing the rest of the survivors in the Context Memory. It is obvious that the number of survivors (**PtA** in the figure) plays a crucial role in determining the amount of time needed to decode the symbol vector given that it determines the number of future nodes that need to be processed by the sphere engine and the number of nodes to be written to memory. Moreover, it must also be noted that this count is directly influenced by the present minimum cost as this is the radius which the sphere constraint is subjected to (**PtB** in the figure). Thus, it is important to aim to achieve the minimum cost for a given symbol vector in the quickest manner without significantly affecting the power consumption of the system.

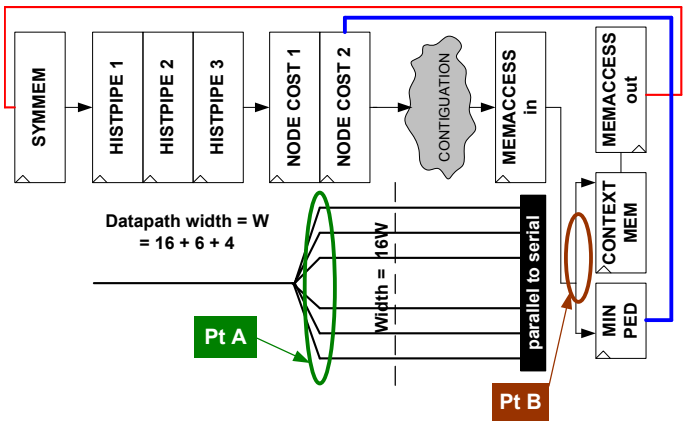


Figure 3.4: Architecture Control Points and Data Flow trends

3.4 Base Architecture: Details

The base time-multiplexed architecture provides the necessary blocks seen in Fig. 3.3 for the a) creation of the costs corresponding to the children of a parent node in the tree search b) the storage of the survivors in memory corresponding to each of the independent contexts and c) the selection of a new node to be processed in the tree from the "M" available choices.

3.4.1 Pipelined Resource

The pipelined cost computation resource utilizes the incoming node information from memory to compute the PEDs of all the children of the given node. The incoming node information from memory would be constituted by the quartet of $level$, $PEDin$, $threadID$ and $symbolindex$. Here, $level$ corresponds to the level, i in (2.9) in the tree search that the node corresponds to, the incoming cost $\mathbf{T}_{i+1}(s^{(i+1)})$ in (2.9), $threadID$ refers to the independent symbol vector context from which the node is taken and $index$ corresponds to the symbol index in the constellation of the incoming node *i.e.* s^{i+1} in $s^{(i+1)} = [s^{nT-1}, ... s^{i+1}]$ corresponding to (2.9).

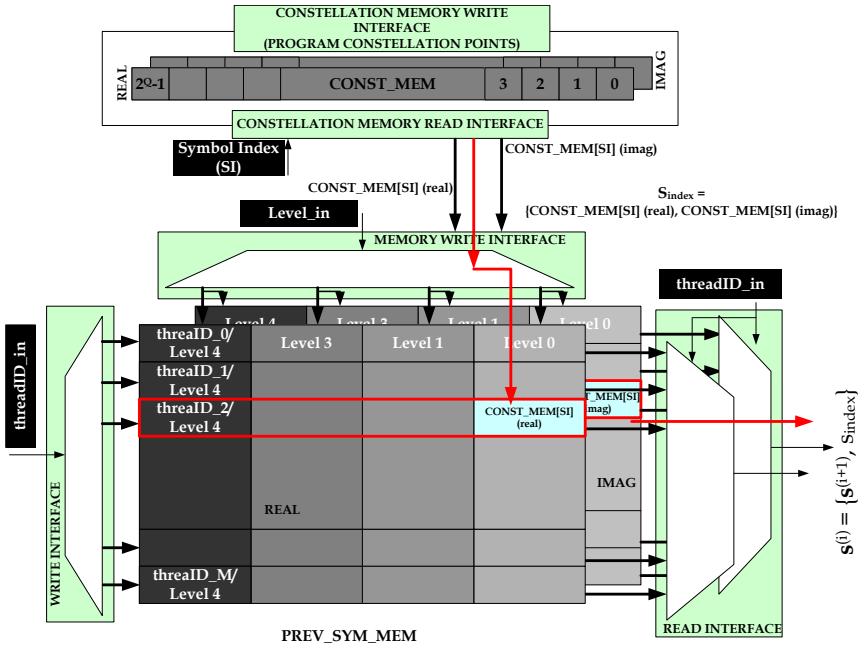


Figure 3.5: Details of Symbol Read Unit

Symbol Read Unit

The proposed architecture makes an optimization by noting that significant portions of the path in the tree, as we move from one node to another is shared in DFS. This is shown in Fig. 3.6 for the case where we have two searches at level = 1. We see that Path1 is followed by Path2 after there are no more nodes down Path1 at level = 1 that need servicing. It is seen that Path1 and Path2 share the P1 and P2 semi-paths. The same holds for when we move down any path down to the leaf nodes in a DFS manner. To exploit this characteristic, the storage of the entire symbol vector is avoided in memory and we work with symbol indices instead and work with just the last symbol index. The result is the reduction in memory requirements by reducing the memory word width (for each context memory) from $DataWidth + LevelWidth + 2 * (nT - 1) * SymbolWidth$ (the 2 comes from having to store both the real and imaginary symbols) to $DataWidth + LevelWidth + Q$ where Q refers to the number of bits per symbol. Thus, for a 16 bit data width, 3 bit level width and 6 bit symbol width we see each memory word reduce from (16+3+(2*3*6)) = 55 bits to (16+3+4) = 23 bits. On the flip side, this optimization does necessitate the

storage of $2 * SymbolWidth * M * (nT - 1)$ bits corresponding to the shared path as we go from one path to the next. This optimization proves of greater value with an increase in the number of antennas and for larger constellation sizes where the number of bits needed would increase in keeping with an increased susceptibility to noise. It is to be noted that the word size only changes with the constellation size by the modification of Q. This is used to create a full symbol vector for cost computation using the input index. The details are seen in Fig. 3.5.

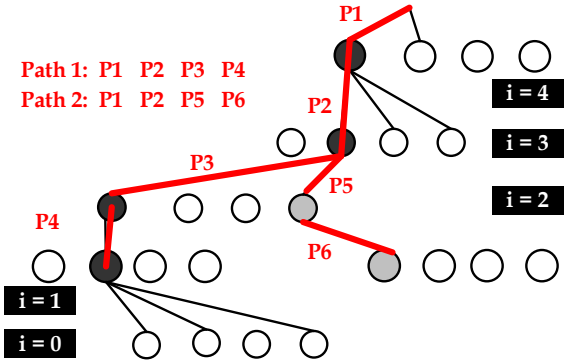


Figure 3.6: Path sharing argument for tree search

History Computation Unit

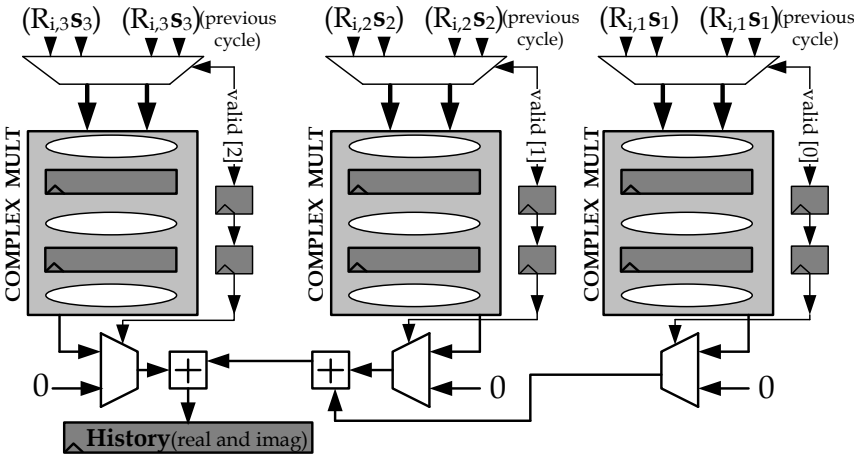


Figure 3.7: Details of History Computation Unit

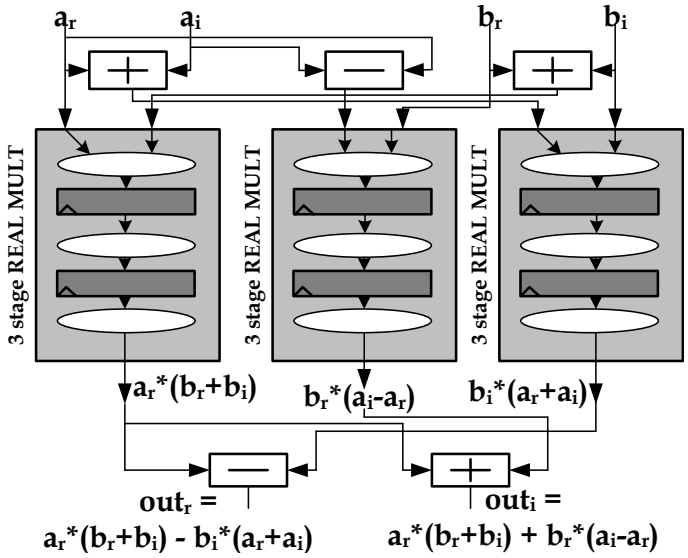


Figure 3.8: Details of Complex Multiplication

At this point, we have a complete symbol vector corresponding to the path up to and including the node under consideration. The aim at this point is to compute the cost contribution corresponding to the second term in the RHS of (2.11) for a given node which we will call $Hist$. This requires the correct row from the $\mathbf{R}$ matrix for computation. This is done by providing the level to a scratch-pad memory which provides the real and imaginary rows of the matrix to compute $Hist$. To prevent unnecessary switching, the $valid$ signal is used as seen in Fig. 3.7. This signal encodes the number of valid symbols in the vector of symbols coming in from the Symbol Memory Unit. A vector of symbols at level 4 (nT) has $valid = 000$, that at level 3 has $valid = 001$, that at level 2 $valid = 011$. What must be noted is the use of multi-staged multipliers to minimize the total delay. The use of pipelined multipliers leads to automated re-timing during synthesis for a balanced delay through the multiplier. The returns on this method are diminishing by virtue of an increased number of DFFs being inserted which lead to increased dynamic power and increased clock power. The total number of multipliers is reduced by optimizing complex multiplication $(a+ib)\times(c+id)$ from $(ac-bd)+i(ad+bc)$ to $(a*(c+d)-d*(a+b))+i(a*(c+d)+c*(b-a))$ with no loss in accuracy as seen in Fig. 3.8. Further optimizations might be feasible by the inclusion of partial product multipliers followed by adder trees.

Node Cost Computation

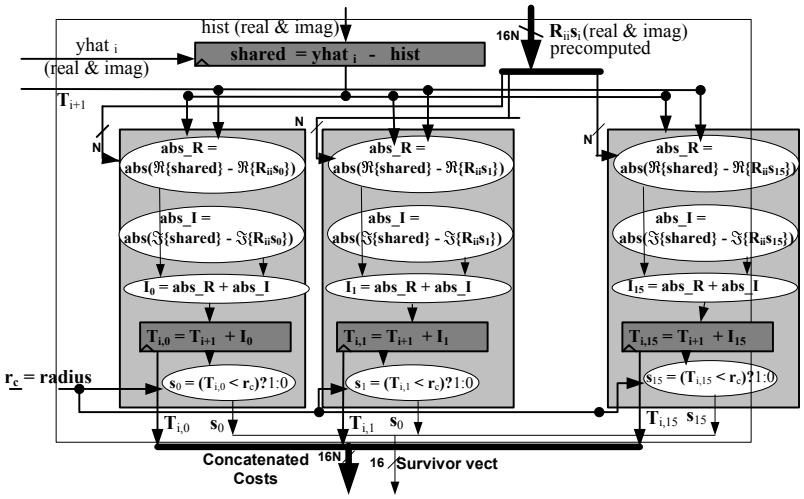


Figure 3.9: Details of Node Cost Computation Unit

The next step is to compute the 2^Q (here 16) PED values corresponding to all the children of the node under consideration at that point in the pipeline.This corresponds to the computation of the distance increment as seen in (2.11) and the outgoing PED as seen in (2.9). The History Cost resulting from the History Computation Unit is used with values from a y_hat memory to compute $Increment = \mathbf{e}_i(s^{(i)}) = \hat{y}_i - History - R_{ii}s_i$ for all 16 children of a given node. The y_hat memory stores $M \times nT$ $\hat{y}_i$ values indexed by $threadID$,$level$ which are precomputed and stored via preprocessing. The $shared = \hat{y}_i - History$ cost is first computed and registered to be shared by all 2^Q parallel Node Computation units that compute the Partial Euclidean Distances (PED)($T_{i,0} \rightarrow T_{i,15}$). The shared cost is then utilized to perform $(Increment)_k = shared - (R_{ii}s_i)_k$ (complex) and $PEDout = PEDin$ + norm($Increment_k$). Also, the sphere criterion (2.13) is used to determine a 16 bit survivor_vect that positionally represents whether a particular child satisfies the criterion or not. It is in the computation of the norm-based $Increment$ that decisions need to be made. An important consideration in the implementation of the cost increment computation is the choice of l^1-norm versus the l^∞-norm which as seen in [30] proves to be a valid alternative by providing increased algorithmic performance (1.0 dB) while reducing area by a small margin (16%) [30]. It is also important to note that both of these alternatives save an area

of 600% or more when compared to l^2-norm. We chose to use the l^1-norm which gives us and $PEDout = PEDin$ + abs(Re($Increment_k$)) + abs(Im($Increment_k$)) for each child. Power v/s Delay evaluation v/s algorithmic performance v/s nodes visited goes here. The details of this block are shown in Fig. 3.9.

Contiguation Unit

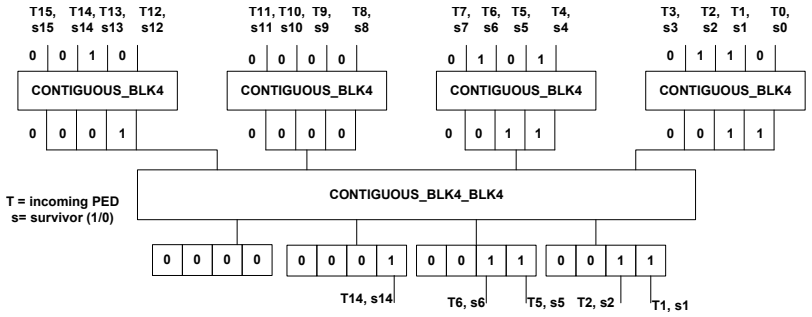


Figure 3.10: Details of Contiguation Method

The aim of this module is to re-arrange the nodes coming out of the node-cost unit to provide for a simpler parsing of survivors to be written to memory. This is done by the contiguation of the resultant PEDs from the Node Computation Unit in blocks of 4 PED values (referred to as *block*4s). Thus, if there were 16 outputs from the Node Computation Unit, we would first contiguate PEDs within blocks 0-3, 4-7, 8-11 and 12-15. The process of contiguation ensures that all the survivors are grouped at one end of the block (here rightmost). Moreover, to ensure that we have minimal overheads in dealing with memory writes, we ensure that the *block*4s are themselves contiguated. This is visually represented in Fig. 3.10 where we show the case when we have survivors at indices 14, 6, 2 and 1.

Radius Manipulation Unit

This unit manipulates the radius for the different tree searches being performed. The user can set the radius to a specific magnitude or infinity for ML operation. In addition, dynamic modification using "Radius Reduction" technique [31] is also allowed. Under the radius reduction technique, the initial radius is kept at infinity and hence all survivors are allowed to pass through until one minimum PED candidate is found at level = 1. This then

becomes the new radius until it is replaced by a new minimum, if at all.

3.4.2 Memory Subsystem

The role of the memory sub-system is to provide for the storage of the survivors of the tree search at each level for consideration at a later stage. Moreover, a node is selected at each clock cycle to be sent into the pipelined cost computation resource. To achieve this, the memory interface is sub-divided into a) "M" instances of context memory and their interfaces to read and write from the memory and b) Node Selection logic and its routing to the output. The top level combination of modules is as seen in Fig. 3.12:

Next Node Selector

This module determines which of the SV memories gets preference to use the pipeline in the next clock cycle based on the status of memory access unit. The choice made is then sent as an aggregate of $pipeline\_info = nextlevel$, $nextPEDin$, $nextindex$, $nextvalid$ in the next clock cycle, where $nextlevel$ = next level in the tree to be processed, $nextPEDin$ = incoming cost to the pipeline for a given node, $nextindex$ is the symbol index of the node and $nextvalid$ provides a pattern that represents the number of nodes that need to be considered for cost computation as talked about in the History Cost Computation description.

Next Node Read

Routes the right $pipeline\_info$ to the Next Node Selector to be sent out.

Memory Access Block

This block described in Fig. 3.11. It is seen that the incoming set of PEDs, assuming unsorted but contiguated, need to be written to memory for later consideration. The mapping of the incoming set of PEDs to a given memory is done by virtue of the $threadID$ which tells a given memory whether the set of values belongs to it or not. If there is a match, the Memory Access block corresponding to the $threadID$ memory will latch in the 2^Q PEDs and the associated symbol indices. It must be stated again, that we are not going to be dealing with full symbol vectors for storage. Instead, we will work with

the indices and re-create the full symbol vector in the Symbol Memory Unit. The incoming nodes are then stored one at a time to the a memory which is organized as $NodeInfo = PED + Level + SymbolIndex$. This allows for a much simper memory interface and also provides for profiling of the PED stream if needed. If the incoming stream is at the lowest level in the tree, then the minimum PED search kicks in and searches through the PED stream serially to determine the lowest cost. This lowest cost becomes the new radius for the given context. It must be noted, that if there are K survivors, then $K-1$ $NodeInfo$s would be written to memory and one be kept aside as the preferred node to be sent into the pipeline. The select unit sends a request to one of the Memory Access Blocks and sends in the relevant node information into the pipeline every cycle that there is no stall. A stall would be created if there are nodes left to be written to memory while a new node comes into the Memory Access block with more than one survivor.

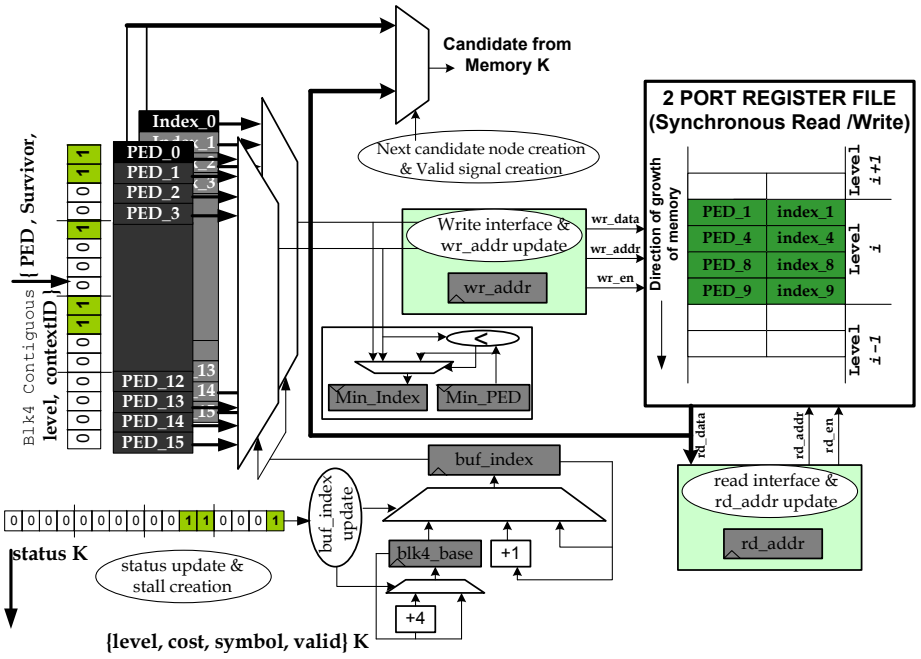


Figure 3.11: Memory organization and details of memory access block

A few immediate points of note that need to be brought out in this architecture in comparison with other works are:

- Deep critical path and no resource sharing in present architectures.

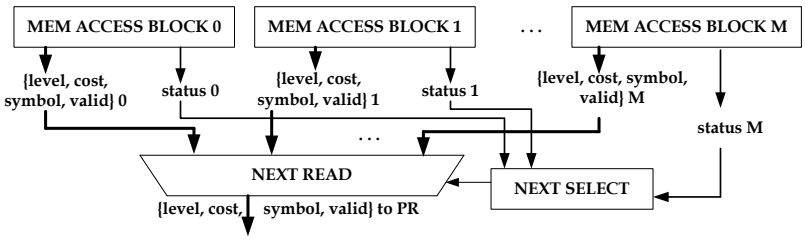


Figure 3.12: Memory organization and details of memory access block

- Multiported memories in present architectures by virtue of the lack of pipelining. This is needed to make sure writing of survivors to memory does not become the bottleneck in the system. The use of multi-ported memories brings with the problem of power hungry memories and interfaces which are tough to design. Also, multi-ported memories do not scale with technology. The use of Single Port Read and Single Port Write and serialized writes are worth exploring to.

3.5 Time-multiplexed SD: Base Architecture results

The fully optimal base architecture using radius reduction described above was implemented in an 8 metal 90nm 1.2V CMOS process. The implementation from Verilog to GDSII was implemented using a combination of Synopsys design compiler for synthesis and Cadence Encounter for Place and Route and Cadence DFII for physical verification using Mentor Graphics' Calibre. To provide for verification of the sphere decoder, the Sphere Decoder system was designed with a loadable test vector memory that will be used to feed the sphere decoder architecture and the interface to this memory. The methodology used for building this architecture and its evaluation is detailed in Chapter 7. Also, for this implementation, the constellation of choice is 16-QAM which is of the from shown in Fig 3.13

The evaluation of the architecture is done at both the physical design and algorithmic levels. In this section we also make observations with regards to the advantages and limitations of time multiplexing.

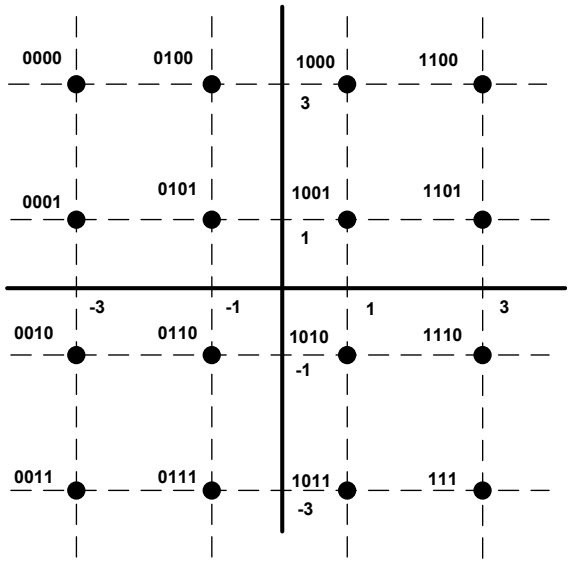


Figure 3.13: The 16-QAM constellation that is going to be used for the 4x4 decoder design

3.5.1 Algorithm Verification

To verify the correctness of the proposed architecture and the testing structures, we perform evaluations over 20 iterations of 10000 message bits each at variable SNR per receiver antenna and 4 x 4 configuration with a 16-QAM constellation. The channel is modeled as i.i.d. complex Gaussian with zero mean and unit variance. The noise is modeled as zero mean circularly symmetric white Gaussian noise. The variation of Bit-Error-Rate (BER) with SNR and the reduction in the decoding cycle time for the Radius Reduction is seen in Fig. 3.14. Note the reduction in completion time by orders of magnitude. The architecture provides for easy movement between the different algorithmic flavors by virtue of radius control. It is seen that the unconstrained Radius Reduction Technique (RReduce) exactly matches the Maximum Likelihood (ML) decoder performance.

It is important to note that the architecture can be evaluated at the architectural level independent of the algorithm being used. This is because the mathematical fabric for cost computation remains unchanged.

The summary of the design results for base Sphere Decoder is shown in 3.1.

The contribution of the memory sub-system to the Sphere Decoder results is seen in Table 3.2. At first glance, it is seen that the memory sub-system dominates the power performance of the decoder architecture. But, in this case, the results reported are worse

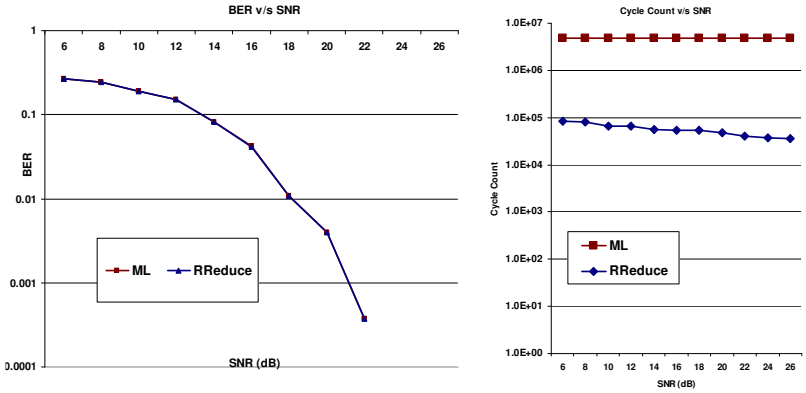


Figure 3.14: BER anc Cycle Count versus SNR for the RReduce versus Maximum Likelihood decoding for Hard Outputs

Table 3.1: Chip Results for Sphere Decoder

Modulation	16-QAM
Decoding Performance	Depth First SD (optimal)
Technology	0.09μm
Frequency	128/230 MHz (retimed)
Gate Count	123K + preproc (using 3.5 μm^2 90nm NAND2)
Throughput (Mbps)	Optimal: 14
BER @ 20 dB	Optimal:10^{-3}
Power	57.63 mW (1.2V)

than what we would generally expect by virtue of the use of DFF's for their creation. Another point of note is that the clock tree accounts for 4.97 mW of the 15.47 mW of the datapath power by virtue of the addition of the pipeline stages while accounting for 35.7 mW of the 41.1mW of the memory power. This results in the clock consuming of the order of 85% of the total power consumption from the memory. We will be dealing with the revision of these numbers by the use of commercial memory generators in the next section. An additional cause for increased power consumption is the use of severe constraints for clock tree synthesis. We have design the datapath for a clock un-certainty of 500 ps but have use a clock skew limit of 120ps for the clock tree synthesis to provide a 380 ps slack for signal integrity issues arrising from cross talk.

Table 3.2: Distribution of Area (10K μm^2) and Power (mW) of the memory + datapath

	Datapath	Mem Unit	Sphere
Area (10K μm^2)	13.079	40.35	53.43
Power Dynamic (mW)	15.470	41.1	56.82
Power Leakage (mW)	0.193	0.671	0.864

3.5.2 Use of Commercial Memories

The study upto this point used the standard cells which utilize scalable CMOS rules for building memory. We have since re-created the results using a commercial memory generator to create a 2 port register file / SRAM of relevant size for each of the context memories. Based on simulation results, the memory power consumption modifies from 41.8 mW to 17 mW bringing the power consumption of the decoder down to 32 mW and the area down from 530K μm^2 to 430K μm^2. Therefore the revised table for area and power distribution between datapath and memory is shown in Table 3.3.

Table 3.3: Revised Distribution of Area (10K μm^2) and Power @ 6ns (mW) of the memory + datapath

	Datapath	Mem Unit	Sphere
Area (10K μm^2)	13.079	30.58	43.659
Power Dynamic (mW)	15.470	17	32.47
Power Leakage (mW)	0.193	0.476	0.669

3.6 Conclusion

It is seen, in 3.1 that the move to an increased number of stages in the pipelined cost computation unit allows for a lowering of the power consumption. But we also see that the implementation suffers from a low throughput especially for the optimal decoding case. An optimal decoder is assured to reach the minimum cost criterion feasible for the numeric scheme used for cost computation. It is feasible to increase the throughput by forcibly constraining the search space by the use of a fixed radius but this drastically reduces performance especially at higher SNR values. This is seen in Fig 3.15.

To this end, there is a need to determine methods that would allow for an increased throughput. This manifests itself in the form of algorithmic/architectural modifications

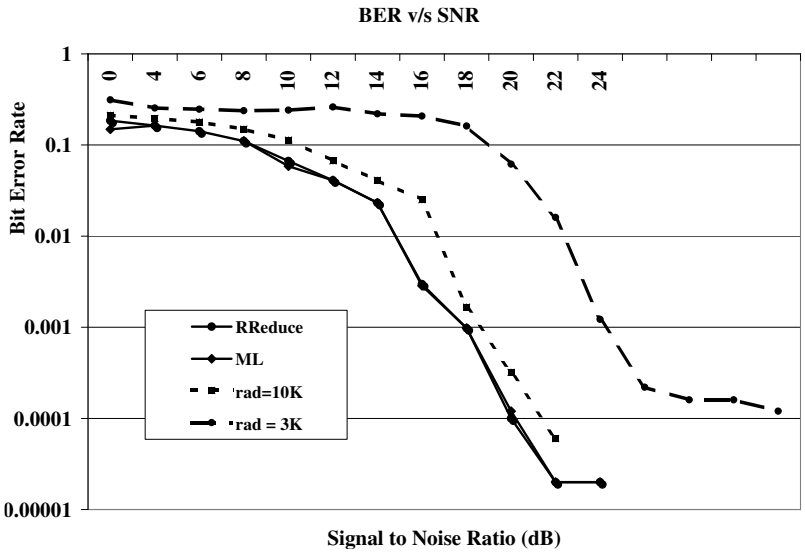


Figure 3.15: Symbol Error Rate versus SNR for different radius values

that minimize the number of nodes visited for a given symbol vector to be decoded. It is this improvement that we strive to cater to and implement in the next chapter while also quantifying the losses as we move down the SNR ranges.

Chapter 4

Throughput Improvements for Optimal Sphere Decoding

4.1 Introduction

As with all systems of this nature, multiple options exist for implementation at algorithmic, architectural and circuit levels. In the presence of such options, the determination of the best solution becomes a function of the constraints to which we optimize. To this end, we analyze the typical constraints that need to be kept in mind when designing Optimal Sphere Decoders. Optimal Decoding performance is always desirable for assuring quality of the wireless data transfer. A loss in performance tends to have a magnified effect as we move down the typical path in wireless system. This is especially true for soft outputs for channel decoding. We use the 4x4 antenna and 16-QAM configuration to observe and quantify the shift in optimality with architecture. Additionally, it is equally important to determine a good methodology for the analysis of the tradeoffs. A good methodology provides for a faster exploration of the algorithmic/architectural space. It is well known that benefits of choices decrease as we go from algorithmic to circuit levels. We create a SystemC^{TM}based evaluation methodology that aims to analyze the losses and benefits of different architectural considerations. The measurements are used to support further architectural improvements that allow us to achieve greater throughput while allowing for optimal algorithmic performance. To improve the throughput for the optimal case, we propose a two layered approach for search space minimization. These improved architectures,

what we call the **Sorting** and **2phase**, attempt to start the search close to the final solution and traverse the search space in order of minimum distance increment. Moreover, tradeoffs in the power-throughput-performance space are also explained so as to achieve optimality based on the usage environment.

4.2 Multi-antenna System Implementation Constraints

The design of wireless systems tend to have a unique set of constraints by virtue of the additional requirement of portability. Design of portable systems differ in their needs by virtue of a limited energy source *i.e.* the battery. This, at first glance, translates to a power minimization problem. But the problem loses its simplicity when we look at the entire process of design and usage of a typical decoder.

To begin with, we have constraints of **throughput**, which translates to the number of bits of information that needs to be processed by the decoder *i.e.* Mega bits per second(**Mbps**). Given that we are dealing with modulated systems this also translates to a certain Mega Symbols per sec where one symbol would encode Q bits using a constellation of 2^Q constellation points. There has been a constant growth in the throughput support needed by mobile systems. Present requirements for 4G systems come to the order of 1 Gbps [2] and beyond. An additional constraint in the throughput space lies in the lack of constancy of the throughput requirement. A present day SoCs would have multiple phases of operation. This might correspond to video streaming requirements at 100s Mbps to browsing of the Internet to speech at or just plain texting. When active in one of these modes, the necessary throughput from one of the blocks of the SoC, here the decoder, would be different. A typical SoC would also have multiple other blocks working at their respective rates. The overall optimality of the SoC would be best achieved by having each sub-block running at a certain rate so as to a) avoid latency issues and b) consume minimum energy for that workload. Therefore, it becomes important for the decoder to perform well at more than one throughput with a higher limit imposed at the high 100s of Mbps.

Power is going to the be performance limiter in any system due to reliability constraints. High currents bring with them the difficulty in delivery of due to IR drop. Additionally, heat limits for high activity regions of the die and Electro-migration which directly influence the Mean-Time-To-Failure. All of these issues get magnified as we move

the technology nodes because of the additional issues of variability and shrinking noise margins. All of these have resulted in the move from area minimizing design flows to power minimizing flows in recent times. Power forms a good minimization objective for systems of constant throughput. A constant amount of work, say bps or Instructions per sec, provides a direct indication of the overall system progress. In most communications systems, design tends to be looked at more in terms of **energy**. This is because there exists either a fixed energy source or the total number of clock cycles does not suffice as a good measure of progress. In such systems, the objective of interest would be a reduction in the total amount of energy consumed for a given workload, say a frame of data to be processed. This, therefore, directly works towards the minimization of the Energy per bit, denoted usually by nJ/b, as a constraint. This measure captures the algorithmic optimality in the energy sense as well. Algorithmic modifications result in reductions in the total number of cycles taken for a successful processing of data. These modifications can also allow for a smaller datapath if the right optimizations. In case of the Sphere decoders we are going to aim to achieve the highest possible energy efficiency (b/s/mW) while supporting a lower limit on the throughput *i.e.* bps limitation exists.

Though the **area** of the ASIC implementation of a core has taken a back-seat as a limiting criterion, this constraint can not be written off. The area of the core tends to have ramifications at multiple levels of the core and its usage. One the one hand, the area of implementation is directly in correlation with the total number of active devices and hence leakage of the design. As we move down the technology nodes present day SoCs have an ever increasing component of leakage power in the total logic power consumption [65]. It is clearly seen that the leakage power consumption of logic will consume about 50 % of the total logic power around 2012. Another trend that must be noted is the increasing number of Data Processing Engines (DPEs) or Accelerators as a significant portion of the controlling functionality is going to be moved to software. Thus, architectures that enable a smaller number of processing elements to be used for a given processing constraint would greatly improve battery life. From the CAD perspective a reduced area core tends to allow for simpler system integration. This is because processing elements such as the sphere decoder tend to be distributed as layout IP which needs to be integrated on a die for a SoC implementation. An additional aesthetic benefit of reduced area would be the ability to achieve small from factors.

The ever increasing costs and time for design and implementation of present day SoCs have interesting ramifications for a typical designer. The creation of a new core within a system brings with it the need for critical design cycles to be spent on RTL design, system integration, physical implementation and functional and physical verification. These design cycles contribute to the already critical design productivity gap. At one end, this can be catered to at either the level of functional **reconfigurability** at the level of the silicon implementation. This reconfigurability can then be used with software control to achieve the requisite dynamism to, in most cases, changing protocols and standards. One the other hand, a **reusabe** soft design IP goes a long way in helping with the problem of reducing CAD cycles. This is done by the provision of characterization information with the IP distribution which could be used by architectural analysis tools to converge on a solution of choice. This solution can then be sent through the layout phases with minimum fuss. On the other hand, pre-synthesized layout macros can also be stored for later integration along with verification IP. In case of the Sphere Decoder this re-configurability comes in the form of the need for the system to be able to cater to multiple a different number of antennas and a changing constellation. The changing constellation could either be a modification in the values of the constellation points in keeping with protocols that use Automated Modulation and Control (AMC).

An important additional constraint that has become popular is that of **adaptability**. Until recently, most systems designed to function at one supply voltage, one threshold voltage and one clock rate. With the combined progress of modeling, CAD and fabrication, use of multiple supply voltages and threshold voltages has becomes feasible. This has become important because the presence of the need to adapt to a varying workload. Each workload brings with it a new point of optimality. These optima can only be reached with the ability to reconverge as a function of the workload. In the case of the sphere decoder, it translates to the ability to work with a variable SNR and its concomitant throughput. The main question to be answered is if the throughput variable is a positive or a negative. An argument does exist for the exploitation of the variable throughput in a windowed manner to achieve a performance closer to optimal than otherwise feasible by employing sub-optimal methods.

The final criterion that forms a unique constraint for any wireless communication system is the choice in the **optimality** of the system. The optimality of the algorithm

being run has ramifications in terms of the system as a whole. For the sphere decoder, the optimality is particularly if the output is going to be used to drive channel decoders. A small reduction in performance would be magnified when sent through channel decoders. On the other hand, a case can be made for a reduction in optimality in favor of increased throughput if the protocol has support for techniques like repetition [53] which would benefit more by a larger number of symbol transmissions than a small number of optimally processed data symbols. The final say on the optimality comes from the type of data being transmitted. A transmission of a high quality image or security sensitive data would have different requirements when compared with a high data rate streaming video.

It is seen from the above discussion that the design and use of sphere decoders is a tradeoff exercise. But, irrespective of the tradeoffs that need to be done at run-time, we are always attempting to do the following for a given performance and throughput requirement:

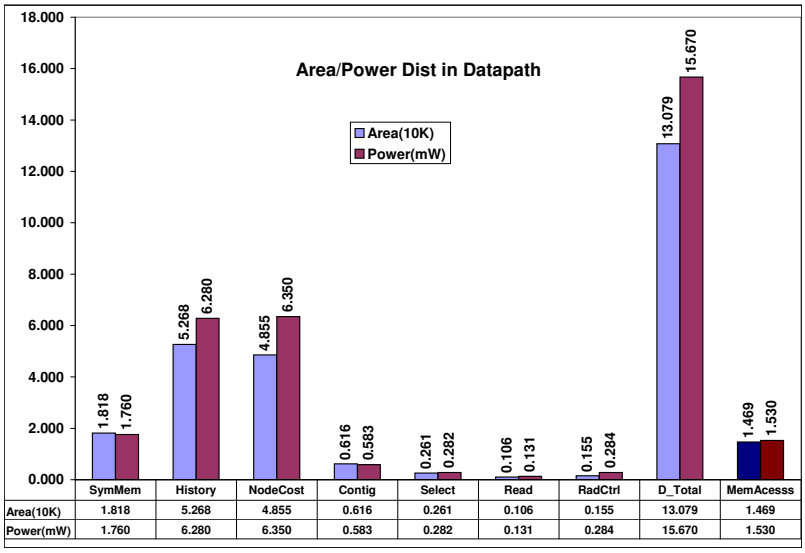
- Minimize the number of nodes to be visited.
- Minimize the average energy consumed per node.

Additionally, the aim of a decoder architecture would be to provide for the ability to converge to a point of optima as decided at run-time. Over and above these requirements, the aim would be to achieve a larger operating space at the circuit, architectural and protocol levels.

4.3 Evaluation of 4x4 Decoder Architecture

An initial analysis of Power, Area and delay was performed to provide an insight into the regions of greatest interest in the decoder architecture. The distribution of the area and power per stage of the pipeline is as seen in Fig. 4.1 for the core while the delay distribution with and without parasitics is seen in Fig. 4.2. The area and power contribution of the memory interface is also shown to provide an estimate of the cost of adding a context memory. Moreover, the delay limitation in accessing the memory is also shown to provide an understanding of the limits of the memory access while optimizing the datapath. It is to be noted that the total critical path delay throughput the datapath of the design would be computed as $[D_{symmem} + 3 \times D_{Hist1} + D_{Hist2} + D_{NodeCost1} + D_{NodeComp} + D_{Contig} + MAX(D_{Select}, D_{MemAcces}) + D_{Read}]$ which comes out to be **21.63** (16.48 ns without

parasitics). The datapath delay distribution shows fair balance. It is seen that most of the delay contribution comes from the Node Computation Block with its array of additions for PED computation where we perform $PEDout = PEDin$ + abs(Re$(shared - (R_{ii}s_i)_k)$) + abs(Im$(shared - (R_{ii}s_i)_k)$).



	SymMem	History	NodeCost	Contig	Select	Read	RadCtrl	D_Total	MemAcesss
Area(10K)	1.818	5.268	4.855	0.616	0.261	0.106	0.155	13.079	1.469
Power(mW)	1.760	6.280	6.350	0.583	0.282	0.131	0.284	15.670	1.530

Figure 4.1: Distribution of Area (10K μm^2) and Power (mW) per block of the Cost Computation Pipeline and Memory Access Unit

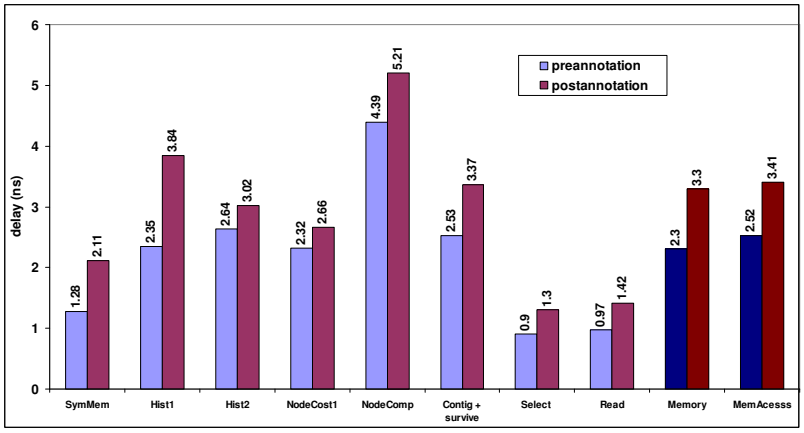


Figure 4.2: Distribution of Delay (ns) per block of the Cost Computation Pipeline and the Memory Access Unit

There exists a great deal of correlation between the distribution of the area and the power of the datapath. This is intuitively satisfying because, a power optimal solution

should be as active as possible. This allows for a lowered contribution leakage to total power consumption and hence leads to a greater possibility of having direct control on converge to the optimal operating point.

The analysis of the Sphere Decoder can be orthogonalized. This is because of the nature of timing in the system. While the datapath can be viewed as a constantly running computational resource, the memory provides for a means of controlling the run-time behavior of the decoding process. While the datapath activity is predominantly a function of the type of data being sent in, the number of cycles for which the memory is active is controlled by the number of values that need to be written to memory. Considering average behavior, each context memory with its associated memory interface behaves in relatively the same manner.

4.3.1 Delay Constrained Synthesis

It is of interest to see what results would be given for an architecture that be synthesized for minimum critical path delay. This is important to gage the increased area, if any, and the corresponding increase in the power. The comparison of the results in a per-block manner for the Base architecture for a clock of 6ns and 1ns is seen in Table 4.1 As

Table 4.1: Power @10ns and Area difference for timing constrained versus area constrained synthesis

Delay Constraint	Criterion	Datapath	MemUnit	Sphere
6ns	Area (sq. μm)	98943	291117	390061
1ns	Area (sq. μm)	123923	304097	428020
6ns	Power (mW)	6.44	7.574	14.014
1ns	Power (mW)	7.59	8.393	15.983

is seen, the constraints result in an increased area and power for the tighter delay constraint, but it is important to see the overall gain in the delay before any judgment can be passed about the result. In particular we focus on the datapath where the critical path exists. The resulting decrease in **total logic delay** is seen in Table 4.2. Note that at present we are ignoring any presence of pipelining for the datapath. Thus, we see a 46% reduction in the logic delay through datapath while increasing the energy consumption by about 14.5%. Consider the following cost equations for delay and energy:

Table 4.2: Total Logic delay reduction for timing constrained synthesis

Delay Constraint	Criterion	Total Delay Logic Delay(ns)
6ns	Total Logic Delay	16.48
1ns	Total Logic Delay	8.8463

$$E_{dynamic} = \alpha C_{switch} V_{dd}^2 \tag{4.1}$$

$$D_{crit} = \frac{V_{dd}}{(V_{dd} - v_{th})^{\beta}} \tag{4.2}$$

$$EDP_{dynamic} = \frac{\alpha C_{switch} V_{dd}^3}{(V_{dd} - v_{th})^{\beta}} \tag{4.3}$$

In the above β corresponds to a technology dependent velocity saturation index which is usually between 1 and 2. Let us consider a value of 2.0 for our first order analysis. If we were to first attempt to bring the delay for the 6ns constrained implementation down to the 1ns delay constrained implementation, the V_{dd} would need to be changed to the order of $2 \times V_{dd}$. This has the effect of approximately doubling the EDP. What we must note is that a lower value of β would only support our conjecture. This is clearly seen in Fig. 4.3 where we observe the expected reduction in slope, *i.e.* the need for increased supply scaling for a similar change in delay as β goes down. It is for this reason that we deal exclusively with delay constrained synthesis.

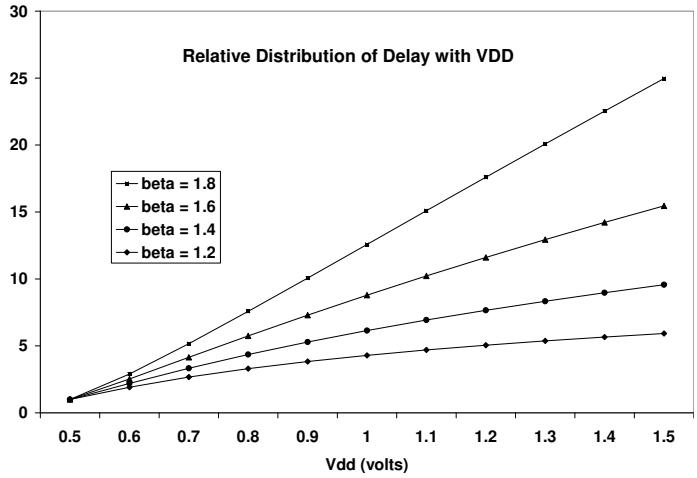


Figure 4.3: The relative scaling of delay with VDD for different velocity saturation indices

4.3.2 Bit-width sensitivity

Datapath Analysis: Pipelining of a design has both pluses and minuses. The inclusion of registers for pipelining does improve the highest possible clock cycle by virtue of a reduced critical path. But this improvement comes at the expense of an increased area and power by virtue of addition of flip-flops. Other than the direct contribution of capacitive switching internal to the flip-flop, an eye must also be on the problem of clock delivery. As seen in [57] the power minimal delivery of the clock is becoming a cause for great concern in present day systems. Additionally, the presence of multiple clock domains also provides greater challenges in the delivery of zero or controlled skew clocks. Therefore, the number of flip-flops must be closely monitored in the design. The expanding nature of the datapath in the decoder makes the flip-flop issue particularly serious in the NodeCost unit by virtue of the need for storing the results of all the PED calculations. The width of the datapath is fundamentally a function of the number of bits used to encode the analysis data. Fig 4.4 captures the variation of the area and delay of the base architecture with the size of the data bits. It is clearly seen that the flip flops do not form the predominant source of the area for the decoder. As a first order analysis for the datapath, we approximate that the area forms a good estimate of the power consumption of the datapath given that this is a high throughput system with most nets being continually exercised.

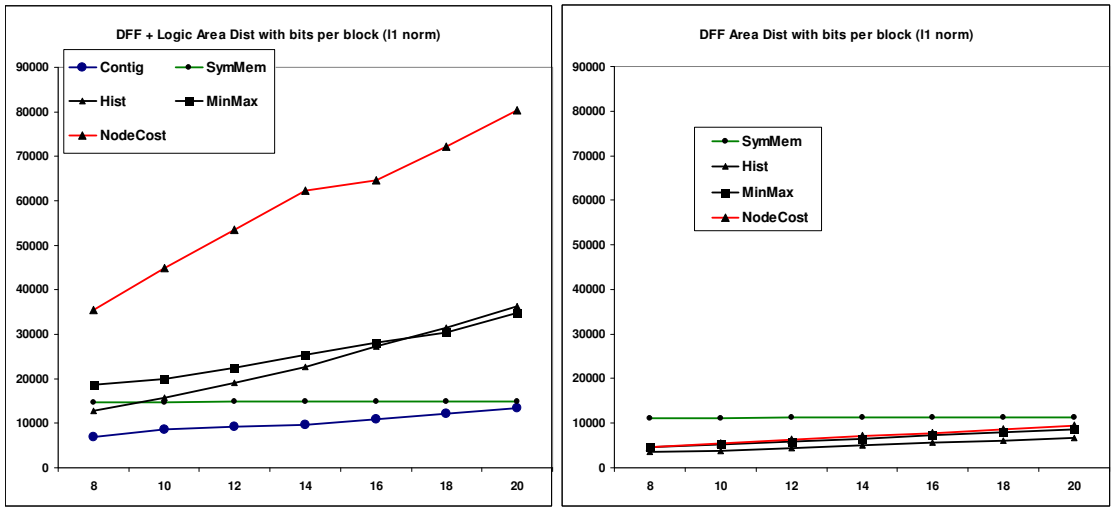


Figure 4.4: The scaling of the datapath with bit width for 4x4 architecture

Memory Analysis: The memory analysis is broken down into the analysis of the interface and the Register File IP. The scaling of both values with the number of bits is shown in Fig. 4.5

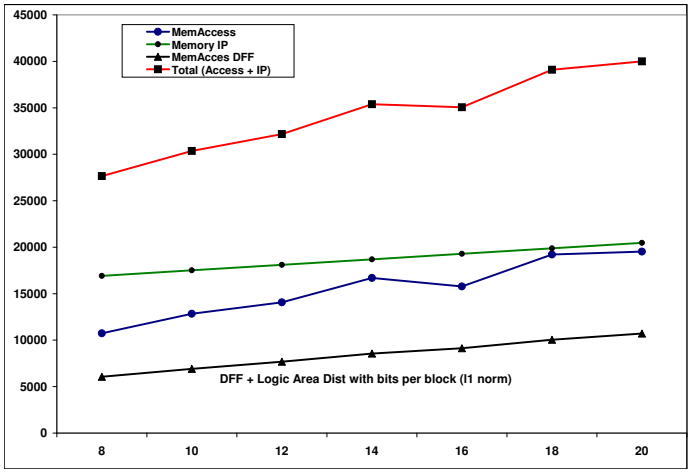


Figure 4.5: The scaling of the memory (interface and IP) with bit width for 4x4 architecture

Though a lower number of bits is desirable, the effects on the algorithmic performance can not be ignored. The modification of the BER with SNR for different bit sizes is shown in Fig. 4.6. As can be seen clearly, the increase in the number of bits brings about a corresponding approach to the ideal case. After a certain number of bits, here 16, the total the algorithmic benefits cease. This is due to the fact that this algorithm is a min-cost search based approach instead of a max search and hence permits a truncate or discard based approach to number handling. This approach will lead to no errors as long as overflows are handled correctly.

It is clear that the system performance degrades significantly when the number of bits is reduced below 16 bits. In most cases, the width of the registers is fixed by algorithm designers in keeping with known limits of signal and noise powers. It is, though, desirable to keep a circuit view while making these decisions due to the possibilities of trading off complexity for increased number of clock cycles as and when feasible.

4.3.3 SNR Sensitivity

The insensitivity of the Datapath to SNR memory accesses and SNR sensitivity. How much memory do we need? How many memories do we need? What port numbers do

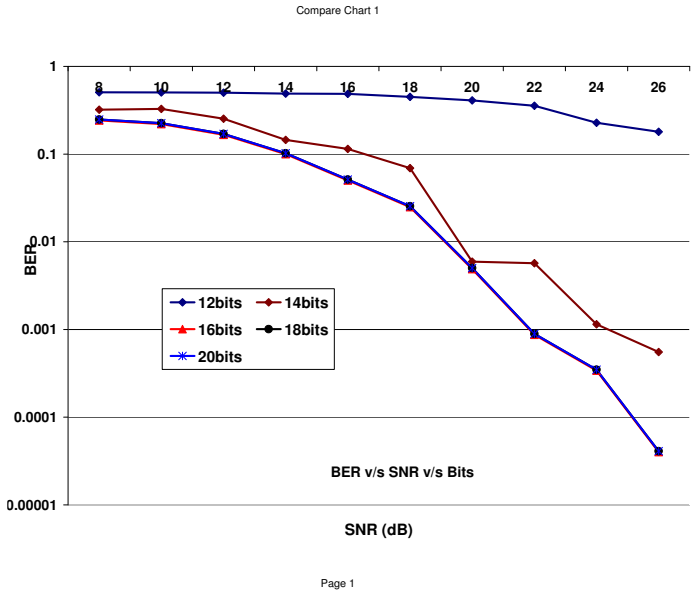


Figure 4.6: The scaling of the datapath with bit width for 4x4 architecture

we need? Evaluation of access patterns

To begin with, we must first determine the fidelity in terms of power of the core in different environment. This has been evaluated at max clock for different Antenna sizes and snr values as seen in Fig. 4.7

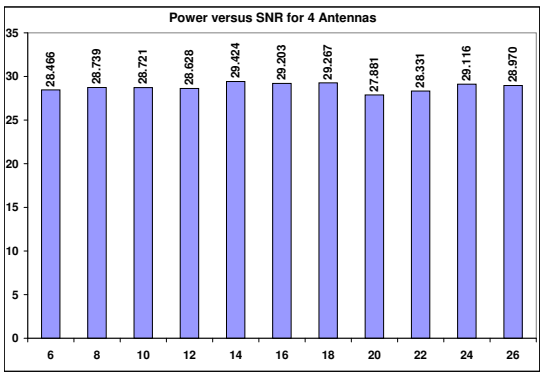


Figure 4.7: The relationship of SNR and Power for the SD core for the 4 Antenna case

As can be seen, this architecture shows a very steady behavior with respect to the environment given that the entire architecture does active work every clock cycle. This characteristic of the datapath further supports the search for an orthogonal datapath and

memory power analysis. Under this datapath characteristic, a great deal of the power reduction would involve system optimizations that would work out to minimization of node visits.

To determine the possibilities of an algorithmic modification and architectural optimizations to reduce the node visits, it would be useful to determine the patterns of survivors seen at each level of the SD search process. To this end, we have performed an evaluation of the average survivor patterns for different SNR values (200 K information bits each) for levels 1 though nT. For the 4 Antenna case we see the pattern seen in Fig. 4.8. In this case we analysis the number of survivors in the bins [1 .. 16]. We chose to ignore the 0 survivor case for now. As can be seen, the search breaks down as the signal to noise ratio begins to fall, especially at the higher levels. This implies that that LE initial radius does not provide a very good starting point for the sphere that contains the minimum cost vector. Thus, this would cause problems if we had limited the number of nodes that would survive at each level as is the case in K-best. But, we must also take heart from knowing that the survivor patterns show a very good tree pruning trend with higher SNRs. This is something that could be exploited to minimize the total cycle counts.

Thus an analysis of the data arrival at the memory boundary provides a good deal of information regarding the progress of the search for a solution. Additionally, a profiling of this nature also provides a good means of quantifying the improvement of one solution over the other. Finally, we must also note that a good deal of the efficiency of implementation depends upon the efficiency of tree pruning.

4.3.4 Discussion of 4x4 System Analysis

The base architecture explained above provides a good starting point for many further improvements that could be made at the architectural level of the Sphere Decoder. As stated earlier, the aim of the architecture should be to achieve the best feasible algorithmic performance at minimum energy consumption. This equates to the requirement for analysis of architecture that would allow for the analysis of improved convergence to a solution while, if needed, a greater momentary power consumption is required. It is our supposition that this is best achieved by focusing our energies on the a) process of manipulating the order of the costs that come out of the Node Cost Unit and b) working on achieving the minimum cost at the memory access block in conjunction with the order of costs coming in

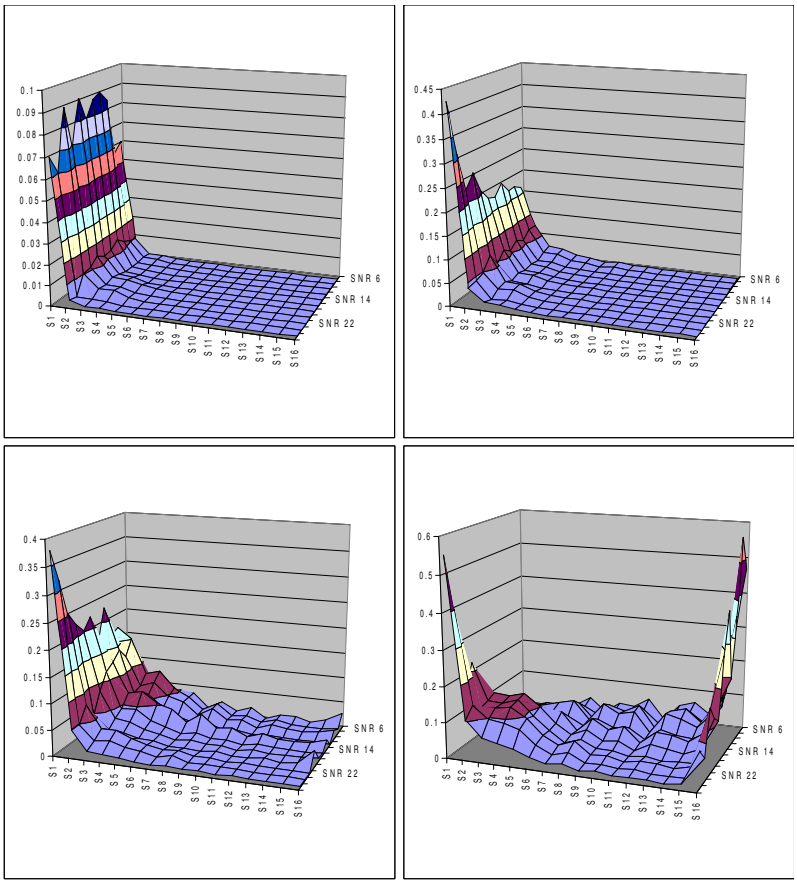


Figure 4.8: The relationship of SNR and survivor numbers for 4 Antenna Configuration

from the Contiguation block. A few issues and improvements that should be explored to achieve the above requirements are:

- It is seen that the use of Radius reduction in its nascent form leads to a great deal of initial setup time for reaching a minimum PED. This is because, all the nodes are allowed to survive. Therefore a great deal of benefit can be leveraged by investigation of means to achieve the minimum PED as early as possible.

- The use of single ported memories do not cause a stall criterion given that the number of survivors when LE solution is used as initial radius tends to be low. In this case, we need to make sure that the number of survivors is less than the number of contexts being serviced at a given time. This is generally the case [29].

For these reasons, we use a single port read and single port write memory interface that uses a serialized write to memory of the survivors of the sphere criterion. This allows for the use of serial search algorithms such as max and min searches, block-wise sorting etc. to be applied in a more scalable manner at the interface to memory. This serialization does not become the bottleneck as long as the total number of survivors is less than the number of memories (which is generally the case [29]). Also include the explanation for the stall number not being an issue when Radius Reduction is used when we have LE solution based method. We should see how things vary with SNR in that case though. Also mention the analysis of the memory.

Also, the Contiguation unit provides for reduced complexity in conjunction with the serialized writes to memory by limiting complexity to blocks of 4 nodes. There is a possibility of performing functions like sorting within the *block*4s which would combine well with the Memory Interface to enable quicker servicing of the surviving nodes that need to be written to memory. Results show that the Contiguation block has a very small delay through it and hence can be exploited in attempt to reduce the strain on memory.

4.4 Evaluation of technology for high level analysis

The evaluation of architectures at higher levels of representation requires confidence in the relative trends being maintained as we move down to layout. This criterion is very closely related to characteristics of the process in question. Area and power trends can be affected by process features like the number of metal layers, the sensitivity to process corners and kind of standard cells available for use. Standard cells from commercial flows tend to have a larger range in drive strengths available and also are better characterized. To analyze these physical characteristics, the base architecture from Chapter 3 was analyzed post layout. The parasitics were back-annotated using the Standard Parasitic Exchange Format (SPEF) from Encounter using FireandIce sign-off extraction. The switching activity back-annotated using Switching Activity Interchange Format (SAIF) files from Modelsim into Synopsys PrimeTime and PrimePower for timing and power analysis. The details of this methodology can be found in Chapter 7. It must be noted that the context memories were created using arrays of D Flip Flops for simplicity of Place and Route. The entire analysis was done with a delay constraint of 150 MHz.

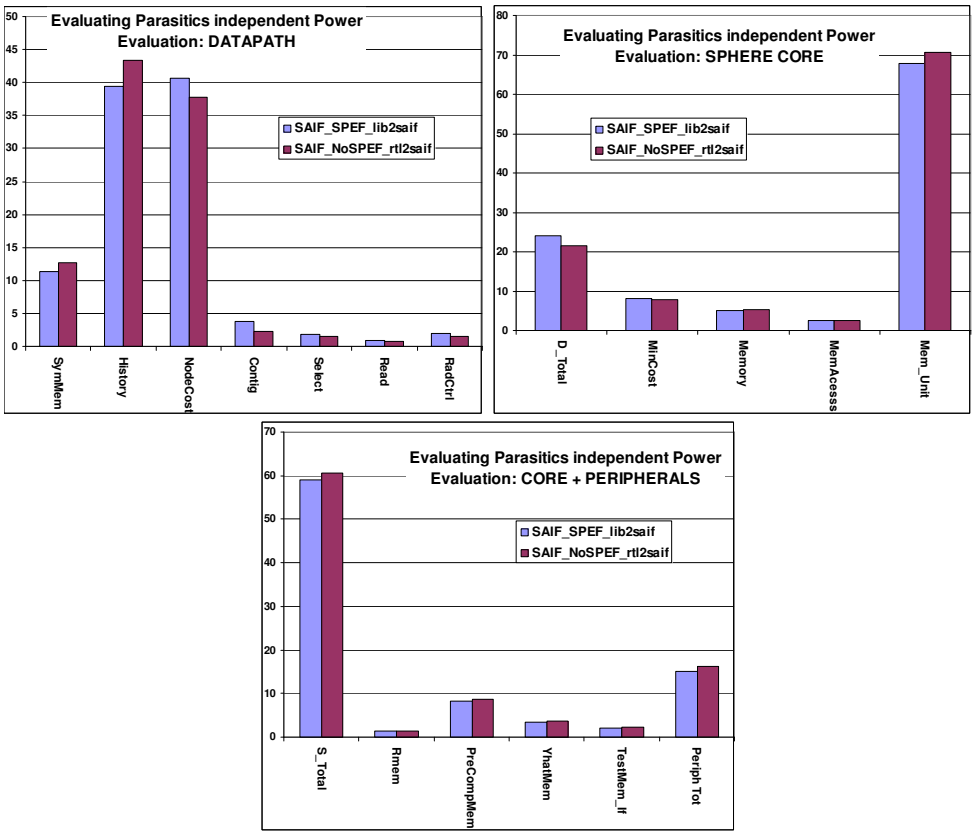


Figure 4.9: Evauating feasibility of only Switching Activity based analysis

Standard Parasitics Exchange Format (SPEF), used to represent the net parasitics in an ASIC, takes time to be generated. It would be advantageous to check if the evaluation of the architecture without SPEF will hold even after the SPEF is added. To this end, we performed a quick study of the relative distribution of the power for just the datapath, the datapath with the memory (*i.e.* the Sphere Decoder) and the Sphere Decoder with the peripheral circuitry for testing (*i.e.* a realistic environment). The breakup essentially maintains the separation of our analysis. Also, the datapath is a high activity block, the memory a low activity block and the peripheral block and even lower activity block. The results of this analysis are show in Fig 4.9. As seen, the relative distribution is retained in both the pre and post annotation power evaluations. This implies that a successful architectural evaluation and a corresponding decision holds, in case of the power behavior, after place and route. This result holds better for processes with higher number of metal

layers given that the distribution of area is retained and a closer relationship between synthesis results and place route results can be achieved. This, therefore, makes it feasible for the evaluation to be performed post synthesis with greater confidence.

4.5 SystemC^{TM}based exploration

The complexity of designs such as the sphere decoder make it difficult to make architectural decisions at the RT level. The fundamental constraint here is the amount of time taken to both design the architecture at the RT level of detail and the requisite simulation time for the description. The simulation time constraint comes about due to the level of detail in the descriptions of the system of interest. A description with comprehensive details of both the computation within different cores and the communication between the cores would have a large number of events that need to be monitored and responded to in case of state transitions. Over and above this, the constraints of concurrence required for hardware simulations makes it all the more difficult for the simulation times to scale well with the size of the design. To cater to this bottleneck it becomes necessary to move to higher **levels of abstraction**. SystemC^{TM} [58] is a C++ class-library that allows high-level modeling of hardware and software. The most popular use of SystemC^{TM}is to co-simulate hardware and software, encapsulating low-level models of the hardware and bus inside software system-calls, which is called Transaction-Level Modeling [59]. SystemC has gained some degree of acceptance in industry by virtue for system level exploration *i.e.* performing architectural exploration and verification of hardware software systems. This acceptance comes about by virtue of the following major pluses:

1. The language has explicit constructs for modeling concurrency of hardware systems while allowing for the ability to abstract away details if needed. This support is provided in the availability of

   (a) **Event Driven Kernel**. SystemC has an event driven simulation kernel much like the RT level simulators presently available. This allows for an increased confidence in correctness of the results from the SystemC description.

   (b) **Concurrency Modeling**: Processes like `SC_METHOD`, `SC_THREAD` and `SC_CTHREAD` enable simple modeling of concurrency and dataflow in system descriptions. In

most cases `SC_METHOD` is preferred because it represents a simple execution of a function call which returns after it is done. The `SC_THREAD` construct is provided to allow for stalling using `wait()` statements and hence requires thread management by the simulation kernel which might becomes a quite severe memory management issue if not controlled.

(c) **Combinational and Synchronous behavior**: The language comes with the ability to express explicit sensitivity of the above processes to either clock or signals using the `sensitive` construct. This allows for the ability to model both combinational and synchronous behavior.

(d) **Modular System Representation** The presence of `SC_MODULE` macros allows for the coding of functionally separate units which can be later instantiated to create a higher level design unit. This, therefore, allows for a hierarchical design capture which is a must for understanding and describing functional separation and also enhances debuggability. To further aid in the system description, the modules can be coded with either inputs (`sc_in`) and outputs (`sc_out`) of a certain data type. In addition to this, the ports of a module can be abstracted away by the creation of a port (`sc_port`) of an abstract data type (a packet class perhaps) which caters to a certain interface (`sc_interface`).

(e) **Multiple Data types** In typical signal processing systems, the convergence to a final working specification of an architecture requires the agreement of the algorithm developers. Most algorithm developers works in the floating point domain. It is important To ensure that the correctness of the algorithm is not lost when transitioning from a floating point representation to a fixed point or integer representation. This is aided in the language support for floating point, fixed point (`sc_fixed()`) or integer (`sc_int`, `sc_uint`, `sc_bit` etc) data types.

2. The final system description automatically becomes an executable specification which aids greatly in the verification process. The importance of this description is heightened by virtue of it, if needed, being cycle accurate. The cycle accuracy of a model makes it possible for its easy incorporation as a golden model for cycle by cycle checks of results which results in faster bug convergence and correctness [60].

3. Easy extension of previous versions to include new features by virtue of object oriented

coding of the architecture. This manifests itself in not just a re-usable method of coding but the creation of templates of blocks with abstract data types. Thus, the result is a simple method of packaging of a verification IP for a system which could also serve as source code for synthesis if tool support is present for it.

4. SystemC^{TM}is an open source which allows us to do away issues with licensing and the lack of re-usability of solutions from one platform to the next. The compilation of SystemC^{TM}code is easily done using standard GNU tools and hence allows for easy integration with any tool flow.

Effective utilization of the language is still being explored at many different fronts like verification and synthesis with CoCentric^{TM} [61]. But no real success has been reported in the ability to provide large scale SoC power results using SystemC. Work done with Ramsey Hourani and Winser Alexander was an effort to evaluate the ability of SystemC to provide prototyping and evaluation information for low level DSP design blocks [62] [63]. It was seen that the SystemC based framework provided power results that closely followed synthesis results when a sufficient characterization information is available. The power and area results using the above framework was based on the use of pre-characterized Library computation IP cores for evaluation of design choices. It is our supposition that this methodology holds good, at least in relative terms, for most highly active datapath units.

In the case of the present architecture, the datapath is an excellent candidate for analysis using simpler models for power and area as a function of arithmetic units. One of the important choices to be made in the representation is the level of abstraction. Best to know the effects of algorithmic and architectural choices and approximate the computational requirements first. This is because the decisions made here are far greater ramifications than the decisions made at the circuit level. It is to this end, that we attempt to make as many decisions as feasible at the architectural level using SystemC as our representation of choice. Before this can be done, it is imperative to analyze the loss of accuracy, if any, by moving to higher levels for architectural tradeoff analysis.

4.5.1 Search space minimization: Sorting and the 2-phase architecture

Adaptive Radius reduction is good but is only as good as the first estimate of the cost. It is seen that the Node Cost Unit creates 2^Q (here, $Q = 4$ for 16-QAM) cost increments corresponding to 2.11. The nodes that survive the sphere constraint need to be written to memory. This interaction is shown in Fig. 4.10. A 1 at the input of the contiguation unit implies that the partial euclidian distance (PED) C that is coming in at the position in the incoming input stream has satisfied the sphere criterion. Note that the memory is written in the the order of incoming survivor stream. The aim here is to make sure that we visit node in increasing order of PED's under the belief that the global minimum cost is most likely to be achieved by visiting the nodes in order of smallest PED *i.e.* expected minimum mean square error $\mathbf{J} = \min T_3(s^{(3)}) + \min T_2(s^{(2)}) + \min T_1(s^{(1)}) + \min T_0(s^{(0)})$. To determine the correctness of this, we built a SystemC model of the present architecture with cycle accurate modeling and interger number representations (in place of floating point numbers).

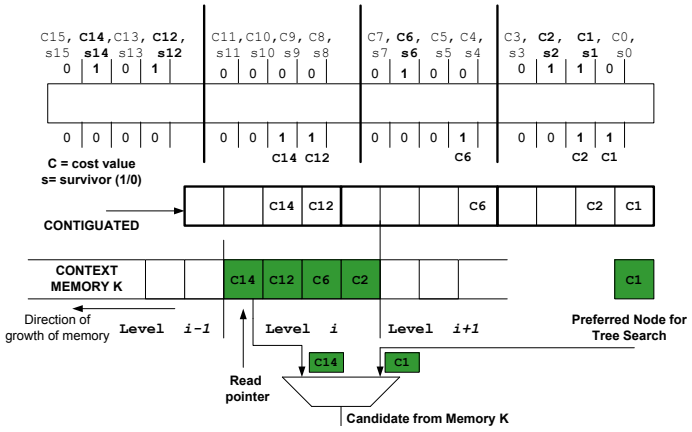


Figure 4.10: The Interaction between the contiguation block and memory in the present Architecture

The modification in contiguation is shown in Fig. 4.11. Note that $C_x < C_{x+1} < C_{x+2}$ and so on. CThe basic idea is to keep the minimum PED C_x at the bottom of the resulting array of PEDs and move the rest to the format shown in Fig. 4.11. The aim of this type of sorting is to enable the growth of the memory from highest to lowest PEDs for a given level in the tree. Thus, after the preferred node for a given level (minimum PED

node) is sent into the pipeline, the next node at the same level that will be send into the pipeline would be the next minimum PED i.e. C_{x+1}.

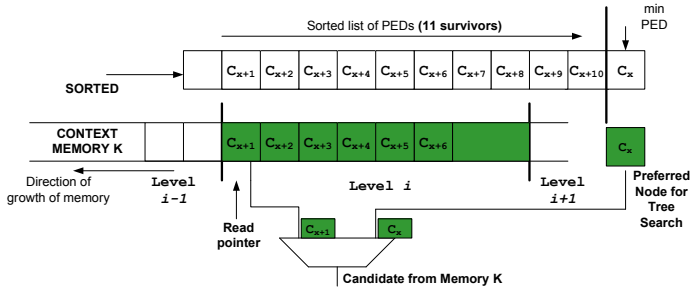


Figure 4.11: The sorting of PEDs that come out of the Node Cost Unit and the resulting memory interaction

It is seen that this technique, does provide improvement but only marginally in comparison with the basic Radius Reduction algorithm. The reason for this lies in the fact that all the nodes stored in memory continue to be sent into the pipeline only to fail to pass the sphere criterion. This causes a lot of wastage of power by virtue of having un-necessary computation of costs which are sure to fail the sphere criterion for most cases. To this end, we can perform aggressive pruning of the memory. This is feasible because of a few observations:

1. The memory growth, for a given level, is from max to min PED. The min PED for a given level is sent first into the pipeline. If a given PED, say C_p, fails to be less than the global minimum, all C_q with $q > p$ will also fail to pass the sphere constraint.

2. There is an explicit boundary between the PEDs stored for each level. This makes it easy to prune the memory by modification of the present read/write address to the memory to the correct level when all the nodes a level are pruned out.

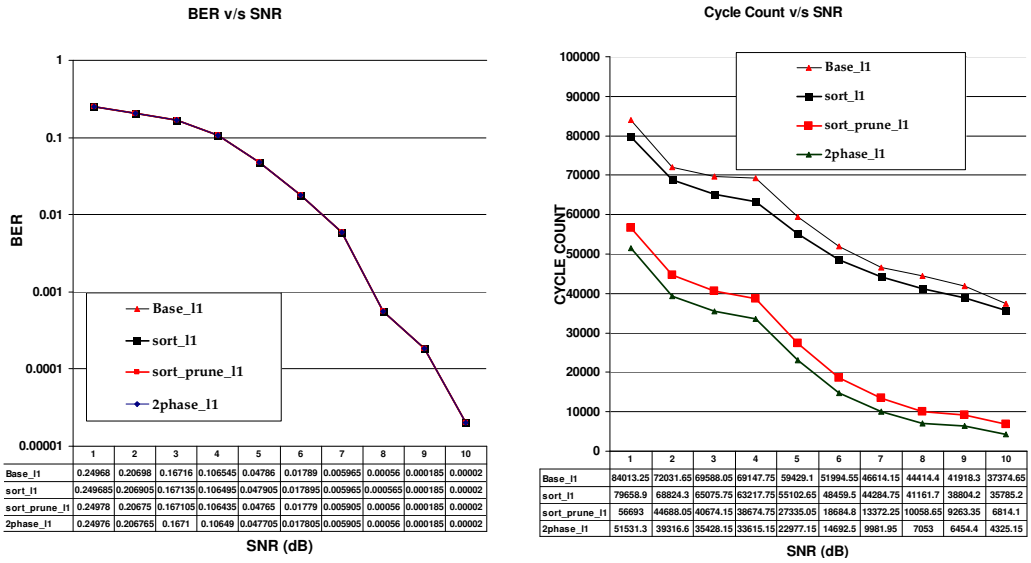
3. The pruning of the tree is best performed when

   (a) The minimum final cost (which becomes the radius for the sphere constraint) is updated. At this point all the nodes in memory with a PED greater than the minimum cost would need to be removed.

   (b) The memory is read. At this point, the PED of next node at the level that is read from is going to be the new lowest PED for that level. This is going to the

new constraint for pruning the tree at the same level.

The advantage of using this optimization is shown in Fig. 4.12. It is seen that the performance of the algorithm does not change from the basic Radius Reduction case *i.e.* the optimal case. We do see a drastic reduction in the cycles taken to decode the message sequence. This strengthens our case for focussing on the sorting of PEDs at the level of the contiguation block.



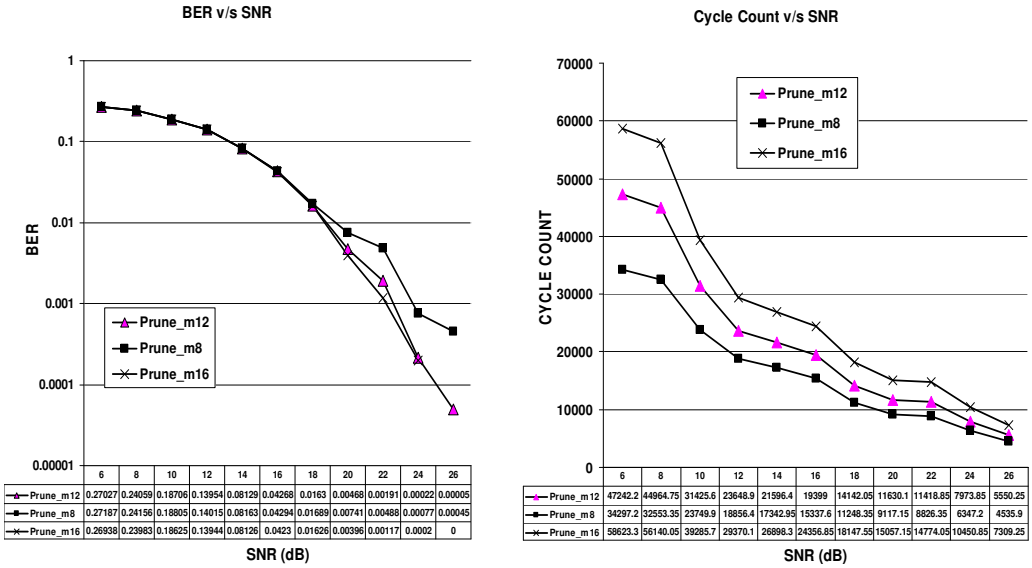
	1	2	3	4	5	6	7	8	9	10
Base_l1	0.24968	0.20698	0.16716	0.106545	0.04786	0.01789	0.005965	0.00056	0.000185	0.00002
sort_l1	0.249685	0.206905	0.167135	0.106495	0.047905	0.017895	0.005965	0.000565	0.000185	0.00002
sort_prune_l1	0.24978	0.20675	0.167105	0.106435	0.04765	0.01779	0.005905	0.00056	0.000185	0.00002
2phase_l1	0.24976	0.206765	0.1671	0.10649	0.047705	0.017805	0.005905	0.00056	0.000185	0.00002

	1	2	3	4	5	6	7	8	9	10
Base_l1	84013.25	72031.65	69588.05	69147.75	59429.1	51994.55	46614.15	44414.4	41918.3	37374.65
sort_l1	79658.9	68824.3	65075.75	63217.75	55102.65	48459.5	44284.75	41161.7	38804.2	35785.2
sort_prune_l1	56693	44688.05	40674.15	38674.75	27335.05	18684.8	13372.25	10058.65	9263.35	6814.1
2phase_l1	51531.3	39316.6	35428.15	33615.15	22977.15	14692.5	9981.95	7053	6454.4	4325.15

Figure 4.12: The BER and Decoding Cycle Count Performance for Base(Radius Reduce), Sorted PED and Prune methods for l^1 costs

An additional optimization that could be considered is the imposition of an upper limit on the number of survivors by the use of an explicit mask. We use this optimization in conjunction with the ideal sorting to set an upper limit on the performance of the system. Fig. 4.12 uses a mask of $m = 16$ which impilies all 16 of the nodes from the Node Cost Unit are allowed to survive. The resulting performance is shown in Fig. 4.13

The sorting architecture does well in reducing the total number of nodes visited in the search for the decoding solution. But the number of nodes visited would still be unnecessarily high. This is because a large number of initial nodes need to be visited before a min cost solution is found. This also implies that a large number of nodes will be written to memory and later pruned away which is a large overkill. As a solution we propose the 2 phase architecture. As the name suggests, we work in 2 phases. In Phase 1 we only allow



	6	8	10	12	14	16	18	20	22	24	26
Prune_m12	0.27027	0.24059	0.18706	0.13954	0.08129	0.04268	0.0163	0.00468	0.00191	0.00022	0.00005
Prune_m8	0.27187	0.24156	0.18805	0.14015	0.08163	0.04294	0.01689	0.00741	0.00488	0.00077	0.00045
Prune_m16	0.26938	0.23983	0.18625	0.13944	0.08126	0.0423	0.01626	0.00396	0.00117	0.0002	0

	6	8	10	12	14	16	18	20	22	24	26
Prune_m12	47242.2	44964.75	31425.6	23648.9	21596.4	19399	14142.05	11630.1	11418.85	7973.85	5550.25
Prune_m8	34297.2	32553.35	23749.9	18856.4	17342.95	15337.6	11248.35	9117.15	8826.35	6347.2	4535.9
Prune_m16	58623.3	56140.05	39285.7	29370.1	26898.3	24356.85	18147.55	15057.15	14774.05	10450.85	7309.25

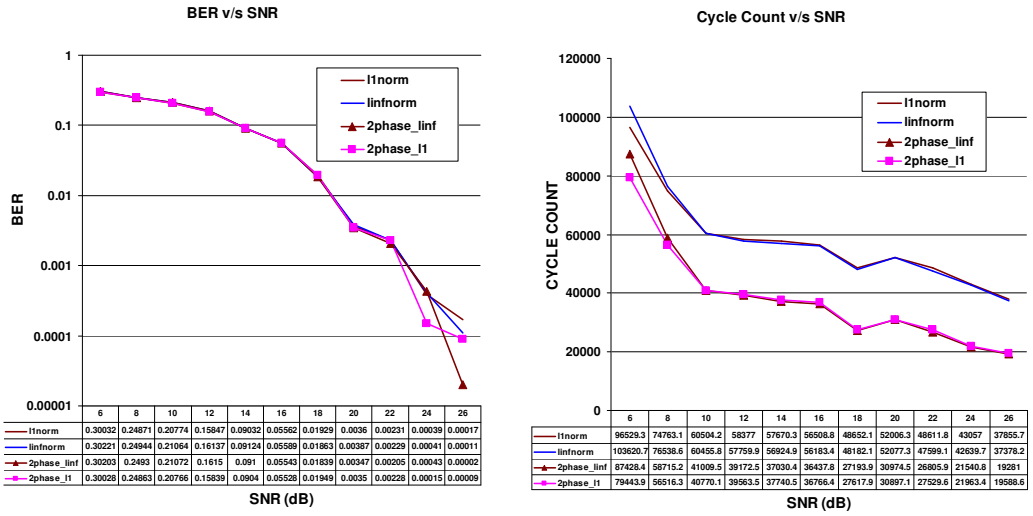
Figure 4.13: The BER and Decoding Cycle Count Performance for $mask = 8, 12, 16$ with Sorting and Pruning

the minimum cost from each level $(nT - 1) \rightarrow 0$ to survive. This leads to the creation of a Linear Equalization solution which forms the initial *radius* value. In Phase 2, we continue normal operation with sorting and pruning while search through the cost space. The improvements of this methodology are going to be discussed in the next sub-section in conjunction with the cost space of Sphere Decoders.

4.5.2 Cost Space for Decoder

Prior work in Sphere decoders has stated that the use of l^1 norm provides a small loss of performance when compared with l^2 norm [30] with of the order of 600% reduction in area. Additionally, l^1 norm also provides an improved performance when compared to the l^∞ for a minimal increase in combinational area and hence power consumption. It is important,though, to consider not just the area but the overall decoding time as well given that we are going to be aiming for a throughput constraint as well. To this end, we first determine the BER and cycle count performance of the l^1 and l^∞ costs for the creation of the output costs in the NodeCost unit. The variation of the Cycle Count and the BER performance with the SNR for the 2phase (Ver_4) design is showing in Fig. 4.14. As is seen, there is a little difference between the two cost computation methods with the l^1 norm

providing a marginally improved performance at higher SNR values.



BER	6	8	10	12	14	16	18	20	22	24	26
l1norm	0.30032	0.24871	0.20774	0.15847	0.09032	0.05562	0.01929	0.0036	0.00231	0.00039	0.00017
linfnorm	0.30221	0.24944	0.21064	0.16137	0.09124	0.05589	0.01863	0.00387	0.00229	0.00041	0.00011
2phase_linf	0.30203	0.2493	0.21072	0.1615	0.091	0.05543	0.01839	0.00347	0.00205	0.00043	0.00002
2phase_l1	0.30028	0.24863	0.20766	0.15839	0.0904	0.05528	0.01949	0.0035	0.00228	0.00015	0.00009

CYCLE COUNT	6	8	10	12	14	16	18	20	22	24	26
l1norm	96529.3	74763.1	60504.2	58377	57670.3	56508.8	48652.1	52006.3	48611.8	43057	37855.7
linfnorm	103620.7	76538.6	60455.8	57759.9	56924.9	56183.4	48182.1	52077.3	47599.1	42639.7	37378.2
2phase_linf	87428.4	58715.2	41009.5	39172.5	37030.4	36437.8	27193.9	30974.5	26805.9	21540.8	19281
2phase_l1	79443.9	56516.3	40770.1	39563.5	37740.5	36766.4	27617.9	30897.1	27529.6	21963.4	19588.6

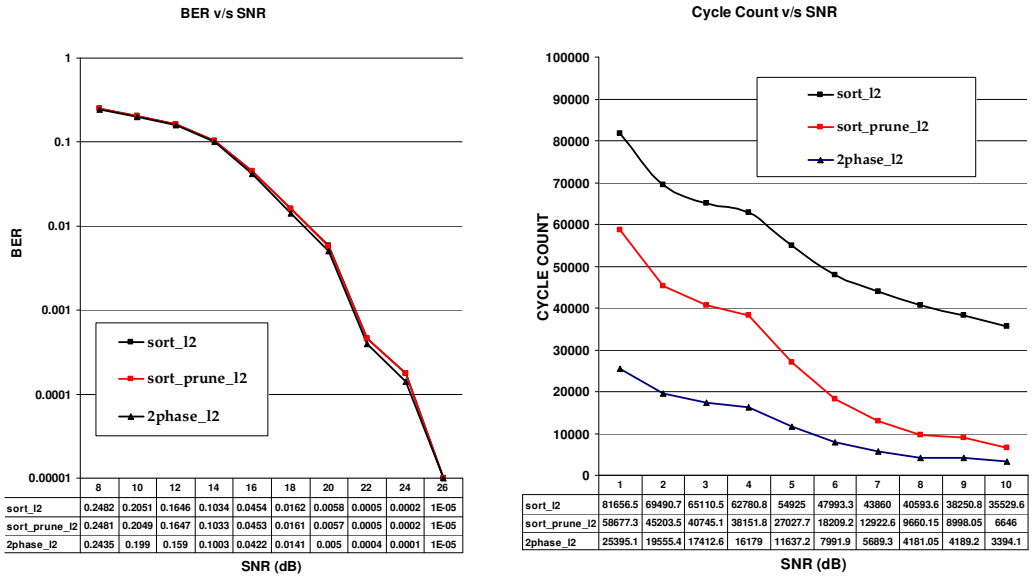
Figure 4.14: BER and Cycle Count variation of 2phase architecture with SNR

The more important tradeoff that needs to be analyzed closely are the characteristics of the l^2 norm. The formulation of the cost takes the form shown in 4.4 for a complex distance of $\mathbf{x}$.

$$(PED)_{l^2} = abs(\Re(\mathbf{x}))^2 + abs(\Im(\mathbf{x}))^2 \tag{4.4}$$

The behavior of the l^2 system to the sorting and pruning constraints discussed in the previous section is shown in Fig. 4.15. In comparison with the l^1 system shown in Fig. 4.12 we see a far greater effect of the 2phase mechanism. This is justified by virtue of the presence of a greater distance between points in the cost space and hence a greater effectiveness in pruning of the search. This is particularly applicable in the lower SNR ranges where the l^1 cost space proves to be insufficiently spaced to provide a good pruning solution.

This cost increment calls for the implementation of two squaring functions which, though fast, would still consume significant combinational energy. To this end, it is worthwhile to explore alternatives in terms of the cost such as the approximate l^2 cost function



	8	10	12	14	16	18	20	22	24	26
sort_l2	0.2482	0.2051	0.1646	0.1034	0.0454	0.0162	0.0058	0.0005	0.0002	1E-05
sort_prune_l2	0.2481	0.2049	0.1647	0.1033	0.0453	0.0161	0.0057	0.0005	0.0002	1E-05
2phase_l2	0.2435	0.199	0.159	0.1003	0.0422	0.0141	0.005	0.0004	0.0001	1E-05

	1	2	3	4	5	6	7	8	9	10
sort_l2	81656.5	69490.7	65110.5	62780.8	54925	47993.3	43860	40593.6	38250.8	35529.6
sort_prune_l2	58677.3	45203.5	40745.1	38151.8	27027.7	18209.2	12922.6	9660.15	8998.05	6646
2phase_l2	25395.1	19555.4	17412.6	16179	11637.2	7991.9	5689.3	4181.05	4189.2	3394.1

Figure 4.15: The BER and Decoding Cycle Count Performance for Base(Radius Reduce), Sorted PED, Prune and 2phase methods for l^2 costs

shown in 4.5

$$(PED)_{approxl^2} = (max(abs(\Re(\mathbf{x})), abs(\Im(\mathbf{x}))) + min(abs(\Re(\mathbf{x})), abs(\Im(\mathbf{x})))/2)^2 \quad (4.5)$$

The approximate l^2 norm consumes a reduced area due to the need for just one squaring operation. But we must still keep in mind the change in the critical path delay by virtue of the need for the implementation of the compare and addition functions. As seen in Fig. 4.16 this cost function proves to be a very good alternative and provides negligible loss in performance in comparison with the l^2 solution. More importantly, we see a definitely improvement in the number of cycles with more than a halving of the number of cycles required for decoding when compared with the l^1 solution. Additionally, the BER performance is also improved.

It is, therefore, important to consider the l^2 space for our analysis especially in case of low SNR applications.

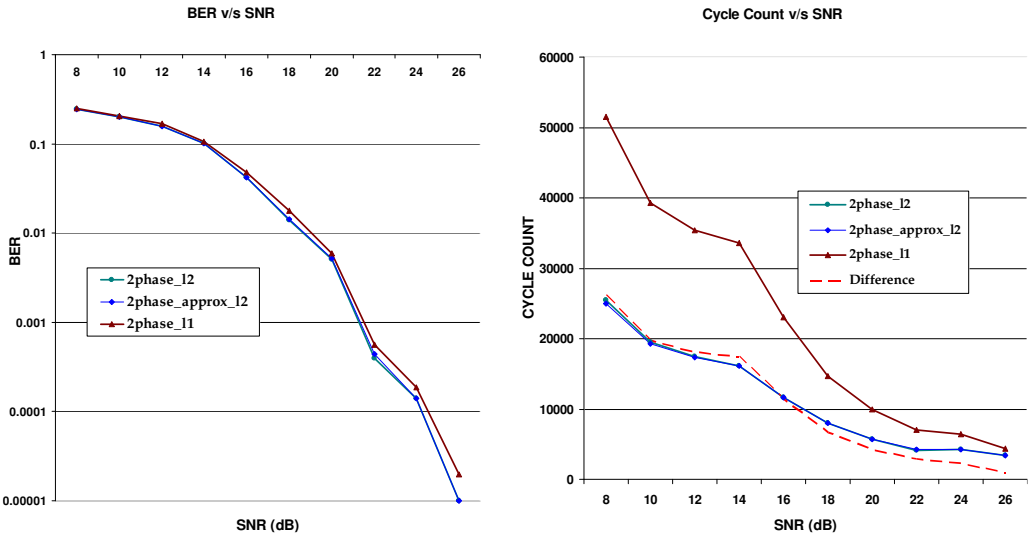


Figure 4.16: Comparison of optimal solutions in the l^1 and l^2 cost spaces

4.6 Implementation aspects of Improved System

In an effort to sort the nodes coming out of the NodeCost unit in a scalable manner, we implementation a sorting implementation as seen in Fig. 4.17

The idea here is to sort the blocks of 4 nodes coming out of the contiguation block. This is done in the MINMAX_4 block which takes in a contiguated input block of 4 nodes and generates a sorted result using a set of MINMAX blocks. The MINMAX block takes in two input PED values $in1$ and $in2$ along with the corresponding index values ind_1 and ind_2 and survivor bits $surv\_1$ and $surv\_2$. This block implements, output $l = in2$ if (($in2 < in1$) && $surv\_2 == 1$); else $l = in1$. The reverse applies for the output g.

Thus, for the 16QAM case above, we have 4 MINMAX_4 blocks which sort the contents of a block of 4 nodes perfectly. The 4 block of 4 nodes (sized 4W each) are themselves sorted using a combination of the MINMAX_4_4 units. The MINMAX_4_4 unit performs the same functionality as the MINMAX unit except for the fact that it takes into consideration only the minima of each group of 4 for making it sorting decisions *i.e.* $l_{4W} = in2$ if (($in2_{4W}[0] < in1_{4W}[0]$) && $surv\_2[0] == 1$); else $l_{4W} = in1_{4W}$. By doing this, we finally have the lowest node of the 16 incoming nodes at $out(4W)_{min}[0]$. It must be noted that this sorting is not yet a very scalable solution given that a move to a 64-QAM

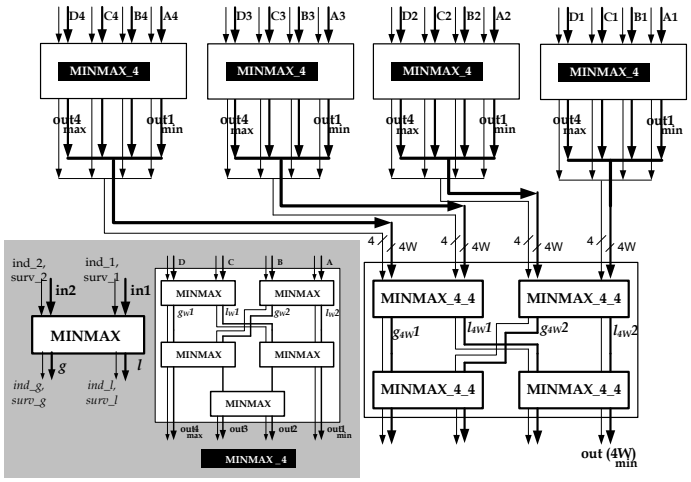


Figure 4.17: Implementation of rudimentary sorting of nodes (16-QAM)

constellation would necessitate the addition of another layer of sorting to deal with the sorting of the 16 nodes presently worked on. The important tradeoff in implementing this block lies in determining the granularity of the sorted blocks. In this case we have blocks of 4 nodes, lets call each $sort\_4$, within which we have all the nodes sorted perfectly, i.e. $sort\_4[i][0] < sort\_4[i][1] < sort\_4[i][2] < sort\_4[i][3]$. These blocks $sort\_4[0] \rightarrow sort\_4[3]$ themselves are sorted on the basis of value of the minima in each block of 4. This, of course, does not imply that $sort\_4[x][j] < sort\_4[y][j]$. But we must note that if we have the group $sort\_4$ coming from points from the constellation sufficiently far apart from each other, then we can assume that the possibility of having a min-cost solution coming from the minimum to maximum within a $sort\_4$ block becomes progressively improbable. We will attempt to show this experimentally in the next few sections. With this argument, it is seen that the natural grouping granularity for the full sorting block is 4 given that, in the Cartesian constellation view, the determination of 4 mutually exclusive nodes, boils down to chosing nodes from the 4 quadrants of the constellation space.

Even without the smarts of selective grouping of nodes, this implementation leads to a strong improvement by virtue of going down the path of the minimum PED at each level of the tree. This boils down to having the ability to finding the first minimum cost, which becomes the first sphere constraint, as a sum of the minimum at each level. To effectively exploit the advantage of this method, it is necessary to modify the radius reduction method

of setting the sphere constraint. This is because, using the initial method of setting the radius to infinity and only modifying it when we get a mincost solution at level 0, we are going to have a lot of nodes at multiple levels that would be written to memory with PED values greater than the first sphere constraint. This is effectively avoided by running a 2-phase mechanism. In phase 1 only one node is allowed to survive as we progress from level $nT - 1$ down to 0. This would allow for the creation of an initial radius in nT clock cycles which would be the initial sphere constraint. In phase 2, we work in normal operation with the rejection of the nodes already processed in the first phase. This achieves a far greater effectiveness in pruning of the tree.

The resulting improvement in throughput performance with no sacrifice in the SNR performance can be seen in the Fig. 4.18

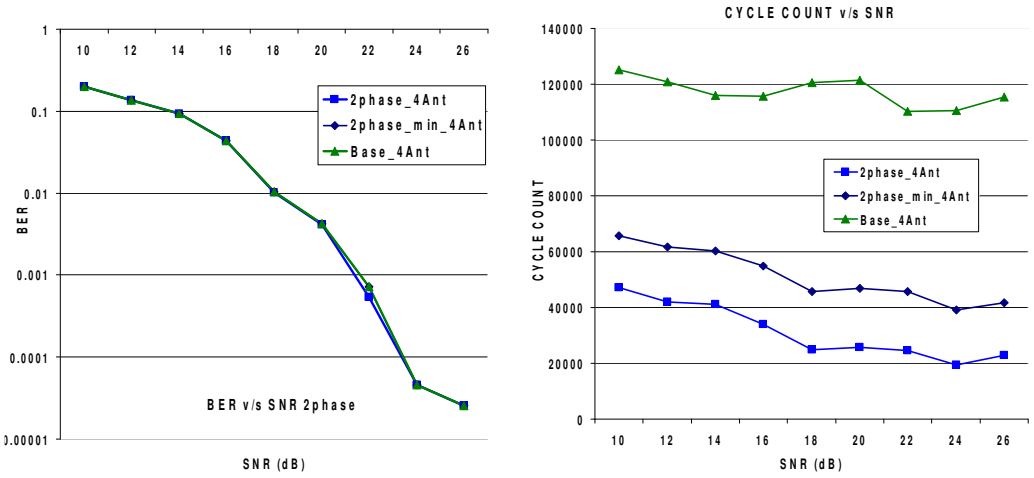


Figure 4.18: BER and Cycle Count variation of min and 2phase architectures

The sorting case the pipelined solution is a good one given that the decrease in the critical path more than makes up for the increase in the power additionally we now have a new model of computation that allows us addition axes for increasing the efficiency of the decoding process.

It is seen that there is a fairly consistent improvement in performance for all levels of the SNR values. But this results should also be viewed with causion because the improvement shown is dependent upon the distance between the initial radius set in phase 1 and the PED at a given level of the tree search. This distance becomes larger with a

larger number of antennas and hence there will be an increased number of nodes surviving at the higher levels of the tree search. This will be dealt with in the next section. For the 4x4 16QAM case we see that that there is a consistent improvement as seen in Table 4.3

Table 4.3: Comparison of percentage improvement in performance between the base algorithm, Min and 2phase variations

snr	10	12	14	16	18	20	22	24	26
min/base	47	48	47	52	62	61	58	64	63
2phase/min	28	32	31	37	45	44	46	49	44

The implementation results of this architecture for the three cost functions is shown in Table 4.4

Table 4.4: Power (mW) @10ns clock, Area (sqμ m), Critical Path (ns) delay and total logic delay for different Architectures

Arch	P_{mem}	P_{data}	P_{total}	A_{mem}	A_{data}	A_{total}	D_{crit}	D_{logic}
$Base$	7.59	8.393	15.983	304.097 K	123.923 K	428.02 K	2.11	8.8463
l^1	9.18	8.49	17.67	310.136 K	135.633 K	445.769 K	2.11	11.2507
$approx - l^2$	14.6	14.7	29.3	439.983 K	243.711 K	683.694 K	2.74	12.7584
l^2	14.6	15.2	29.8	439.983 K	281.239 K	721.22 K	2.41	12.3548

It is seen from the above that the ASIC implementation as we move from l^1 to l^2 comes with about a 65 % increase in power consumption. The major part of this comes from the increase in the size of the adders in the NodeCost unit and the increased bit widths needed to cater to the l^2 norm. Unfortunately, this also brings about an increase in the power consumption of the registers in the datapath especially in the MinMax and the NodeCost units.

With the inclusion of the results from Synthesis, the variation of throughputs with this architecture (without place and route considerations) would be as seen in Fig. 4.19

The comparison with previous Sphere Decoders is shown in 4.5 for the case where ML-like optimality is achieved. The power numbers are reported at maximum throughput performance. It should be noted that Ver_1 corresponds to the fabricated chip with DFF memories. Ver_2 uses commercial SRAM memories for the same design and Ver_3 is an improvement in the datapath where scalable sorting of nodes is performed to ensure that the path down the minimum cost node is searched first. Ver_4 improves upon Ver_3 by performing a 2 phase operation where an initial radius is set by getting a zero-forcing

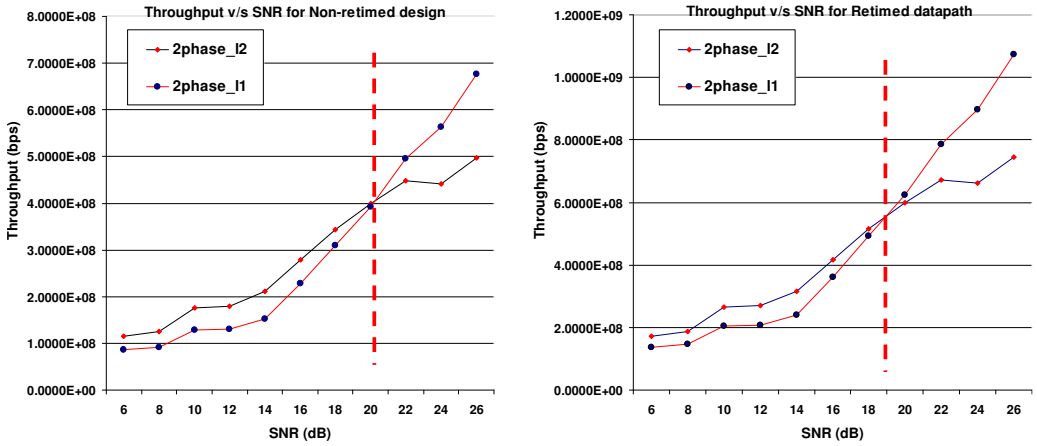


Figure 4.19: Throughput versus SNR for l^1 and l^2 implementations for both non-retimed and re-timed synthesis results

solution in nT cycles. This is done to allow for faster pruning of the search space. The next step is to provide for a true scalable sorting assisted by the aggressive pruning of the memory to ensure that the nodes that are sure to fail in the datapath are not even sent into the pipeline. This is enumerated as Ver_5.

Table 4.5: Comparison of Optimal Sphere Decoders

Ref	Constellation	Mbps	mW	nj/b
Ver_1	16QAM	14	61	4.35
Ver_2	16QAM	14	35	2.5
Ver_3	16QAM	38.5	37	0.96
Ver_4	16QAM	87	43	0.49
Ver_5	16QAM	400Mbps	88mW (@2.01ns)	< 0.22

It is seen that the architectural modifications provide a progressive improvement of the nj/b or the b/sec/mW performance.

One of the more important results that should be noted is that the highest energy efficiency architecture of choice changes with the SNR value. Based on the argument of voltage scaling while meeting a throughput constraint, the choice of cost computation method, *i.e.* l^1 versus l^2 takes the form shown in Table 4.6

Table 4.6: Cost Criterion of choice at different SNR ranges

snr	6 - 18	20 - 26
min energy cost	l^2	l^1

4.6.1 Time-multiplexing: benefits and limitations

One of the fundamental requirements for evaluating this architecture is to determine the benefits and limitations of the number of stages in the pipeline. The addition of a pipeline stage brings with it an expected reduction in the critical path delay where the benefit is greatest if the division of the critical path is uniform. On the flip side, the addition of pipeline stages brings with it more sinks that need to be driven by the clock tree and hence a greater number of buffers for meeting skew limitations. Also, we have added registers that contribute logic power consumption via internal switching. Another constraint for this time-multiplexed approach is the need for an increased number of memories for context state preservation. To evaluate the advantages of pipelining in the first order, we assume that the critical path [$Crit$] can be re-timed into blocks of equal distribution [D=$Crti/M$], and that the overall power consumed can be computed as $CoreP+(P+1)*MemP_{stg} + P*ClkP_{stg}$. Here P is the number of stages, $CoreP$ is the energy of the cost computation core, $ClkP_{stg}$ is the power consumption increment (WORSTCASE) for each stage and $MemP_{stg}$ is the power contribution from a single context memory. For a high activity and pre-dominantly control independant constant throughput datapath such as this, a reasonable assumption is that the $CoreP$ value would not vary greatly with the addition of pipeline stages [64].

The effect of adding a pipeline stage is going to manifest itself mostly in terms of a modified capacitance to be driven at the block boundaries and an increased contribution of the register internal power consumption. At a 6ns clock, on evaluation for a 8 stage pipelined datapath we note that the registers themselves take 1.44 mW for the cost computation portion and 0.165 mW per memory interface for registers at one memory interface (1.463 mW total) which store the incoming stream of nodes for parallel to serial conversion. On furher evaluation, this comes out to be of the order of 1.7 μW per bit for the D Flip Flops (10.2 fJ per bit per access) in the datapath. It must be noted that the power consumption is strongly dependent upon the location of the pipeline stage given that the datapath widens after the NodeCost unit given that we move from working with one node to a stream of

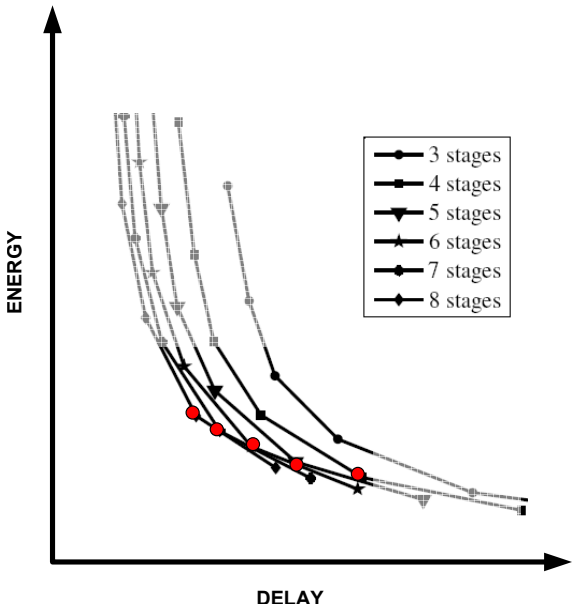


Figure 4.20: Energy Delay Curve with number of stages in datapath

nodes. Also, on analysis of the clock tree we see a 7 stage H-tree with (4, 7, 16, 46, 134, 459, 52) buffers for level 1 through 7. It can be safely assumed that the datapath registers would be sinks for levels 5, 6 or 7 which on careful analysis gave us an average power consumption per buffer of 30.22, 33.45 and 33.89 μW thus showing a uniformity of the clock tree power consumption at lower levels. Is there an argument to be made here about the increase in the number of stages $\rightarrow$ increase in the number of sinks $\rightarrow$ increase in the number of buffers $\rightarrow$ increase in the power consumption? My understanding of how this is going to have to be argued is that the effect of a certain number of sinks reflects itself in an increase in the number of last stage buffers (assuming constant sized buffers) followed by a progressively reduced number of stages in the lower levels of the clock tree. Therefore, 256 new sinks would lead to, say 5 new level 7 buffers (given that we observe about 52 sinks per level 7 buffer), followed by 2 new level 6 buffers and 1 level 5 buffer and so on. Given that they are all assumed to be the same sized buffer, the total power contribution is the total number of new buffers added. The trend therefore would take the form shown in Fig. 4.20

4.7 Conclusions

This chapter captures constraints that a Spherical Decoder implementation would be subjected to. Using a 4x4 and 16-QAM system as the working example, we use a SystemC^{TM}implementation to evaluate the benefits of 2 optimizations that aim to improve the throughput while trading off the energy consumption per node access. We use the information provided to guide us in the search for an architectural choices for different throughput constraints. The benefits of the sorting architecture in conjunction with the 2 phase mechanism are shown with improved throughput performance from 88% to 134% as we move from SNR of 6 to 26 dB. Additionally, we also use first order principles to determine the optimal number of stages in the pipeline for a minimum ED requirement. Also, we provide reasons for the use of different cost computation metrics, namely, the l^1, l^2 and $approx - l^2$, at different throughput constraints. The ideal solution, based on our analysis, is to use the l^1 cost criterion for higher SNR ranges (20 dB and above) while using l^2 for lower SNR ranges (below 20 dB). We utilize all of these analyses in the seach for the best possible optimal solution in the 4x4 and 16-QAM space. One of the important drawbacks of this architecture is the increased requirements for memory.

Most architectures, tend to be optimized for a certain constellation and antenna size. This, though, is a drawback when we consider that the number of antennas for which we design could change. This argument holds for the constellation size as well. It is important, therefore, to analyze the effects of the changes along these axes. It is also important to determine architectural modifications that might lead to a smaller increase Power Area and Delay with the number of antennas and constellation size . We will be doing this in the next chapter.

Chapter 5

Improving the Scalability of Sphere Decoders

5.1 Introduction

Present works have predominantly focused their energies in the design of systems in the 4x4 domain. This is in keeping with the presence of standards like 802.11n which limit the number of antennas to 4 on both the transmit and receive side. But, it is important to consider the future scaling of antenna to ensure that the present systems are not discarded as technology progresses. Additionally, the need for higher spectral efficiencies have also led to the move to larger constellations. This, of course, further compounds the problems of design energy efficient architectures. To this end we work on analyzing our solution on both these fronts. We propose a l^1 based solution that is relatively constellation independent and show that the increase in area and power is very small as the number of antennas is increased. Additionally, this architecture is relatively independent of the number of points in the constellation.

5.2 Scalability issues in present architecture

The scaling of the antenna has ramifications on both the datapath and memory. On the memory side, the memory needs to be organized as $(nT - 1) \times 2^Q$ with each entry being of size $datawidth + levelwidth + indexwidth$.The Table 5.1 provides the necessary

sizing of the a memory for different constellation and antenna numbers. It must be noted that this is for one context memory and would be multiplied by the number of context memories present.

Table 5.1: The size of one context memory in terms of the number of antenna, constellation size and the required level width (l), data width (dw and index width i) for best case l^1 solution from previous chapter

Antenna (A)	Constellation (2^Q)	l	dw	i	Memory Org	Memory Area
4Ant	16-QAM	2	16	4	48×22	173.597 K
6Ant	16-QAM	3	16	4	80×23	222.342 K
8Ant	16-QAM	4	18	4	112×24	268.371 K
4Ant	64-QAM	2	16	6	192×24	483.648 K
6Ant	64-QAM	3	16	6	320×25	790.521 K
8Ant	64-QAM	4	18	6	448×26	1062.395 K

The datapath calls for a larger number of multiplications for the creation of the history cost computation. This is by virtue of a larger possible cost vector of symbols as seen in equation 5.1.

$$\sum_{j=i+1}^{nT\text{-}1} [R_{ij} s_j] \tag{5.1}$$

This scaling brings about the need for larger number of adders and hence a possible greater bit-width to determine the history cost by combining the results of the individual multipliers. On the other hand, the scaling of the constellation brings about the need for an increased number of parallel cost computations for creation of the PED values for the 64 children of a node in the tree. Given that we know the area of one such unit, we can make a quick estimate of the area requirements for the datapath and memory. The scaling trend of the area with the number of antennas and constellation is as seen in Table 5.2. Of particular interest is the nature of scaling of the datapath given that this is where the critical path lies and hence would require greater optimization.

Is is seen, the scaling of the constellation is of greater criticality than the scaling in the number of antennas when memory is considered. Of even greater importance is the need for the storage of the incoming $2^Q \times dw$ (1024 bits for a 64-QAM system with 16 bit wide datapath) values in a set of registers which would consume a good deal of energy per clock edge. Additionally, this presents a much larger load on Clock Tree Synthesis during place and route.

Table 5.2: Area Scaling with the number of antenna and constellation size for the best case l^1 solution proposed in the previous chapter

Antennas	Constellation	Datapath Area	Memory Area	Sphere
4Ant	16-QAM	123.923 K	304.097 K	428.02 K
6Ant	16-QAM	164.259 K	310.137 K	474.396 K
8Ant	16-QAM	178.560 K	404.910 K	583.471 K
4Ant	64-QAM	292.927 K	629.508 K	922.435 K
6Ant	64-QAM	333.263 K	997.483 K	1330.746 K
8Ant	64-QAM	347.564 K	1246.509 K	1594.073 K

In addition to this area trend, the change in the number of cycles for computation is seen in Fig. 5.2 for the 2phase architecture for different antenna numbers. It is greatly satisfying to see that we are going to have a small loss of performance at higher SNR values. This is a crucial point to be noted when considering the need for an optimal architecture which works at multiple SNR values.

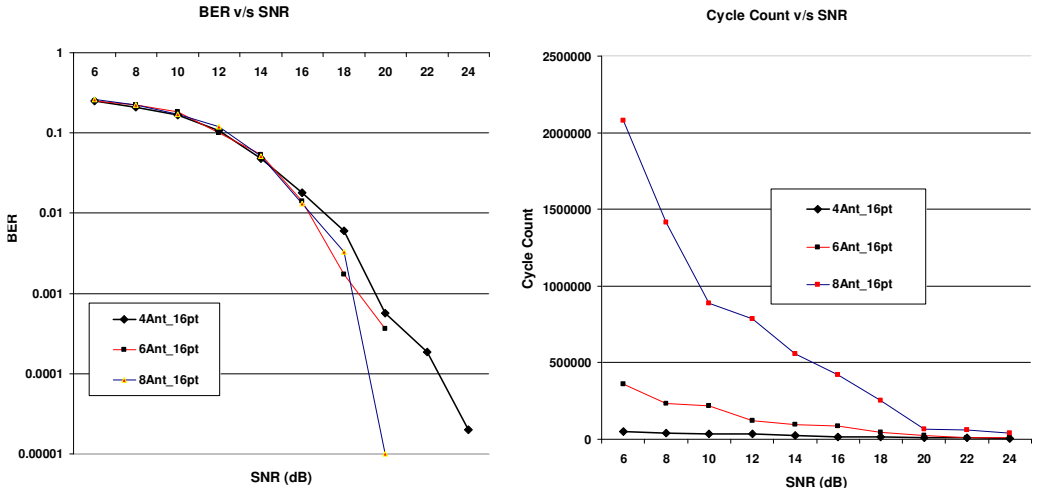


Figure 5.1: The performance versus cycle time variation for 2phase with different antenna sizes

As can be seen, the effectiveness of the 2 phase architecture goes down as the number of antennas goes up. So as to diagnose this it would be useful to re-do the survivor profile analysis for the 6 and 8 antenna case. Fig. 5.2 shows us the variation of the number of survivors versus the level and the SNR. The trends follow the 4 antenna case to some extent except for an overall increase in the number of survivors. This is because of the fact that the individual level intermediate cost $T_i(s^{(i)})$ is going to be much smaller than the

initial radius by virtue of an increased number of added costs (1 per level) leading to the initial radius determination.

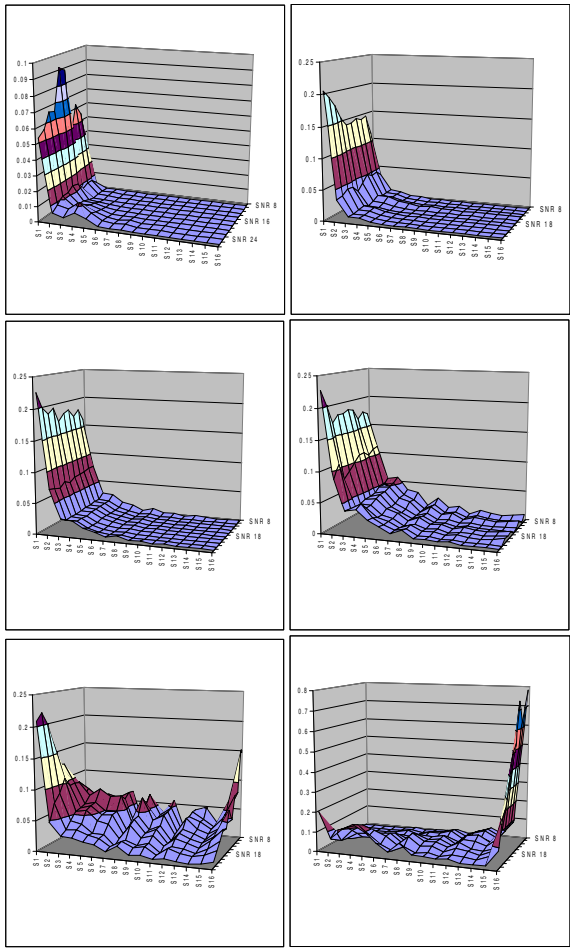


Figure 5.2: The relationship of SNR and survivor numbers for 6 Antenna Configuration

It must be concluded that there is a need for improved search methods, to keep pace with an increased number of antennas. It would be worthwhile to consider the optimization of the core with the SNR of the environment.

5.2.1 Constellation Scaling

The bigger issue in is going to be in dealing with an increased constellation size. We already see an increasing presence of the sorting mechanism in the area and power reports for the 4 Antenna case. To further exacerbate the problem, the writing of up to

64 solutions at a given time to memory becomes a bottleneck. The only solution to the increased number of possible survivors is the need for a larger width of writes to memory and complicated read mechanisms for selection of the correct node to be sent to the cost computation resource. It is to this end, that we consider alternatives to the architecture already presented. The l^1-norm cost characteristics show a very interesting trend in the distribution of their real and imaginary cost contributions. This is as seen in Fig. 5.4 for the 64-QAM case. This characteristic can be effectively exploited in minimizing the number of additions for the PED cost calculation as well as the effort needed in sorting the resulting PEDs from the NodeCost Unit.

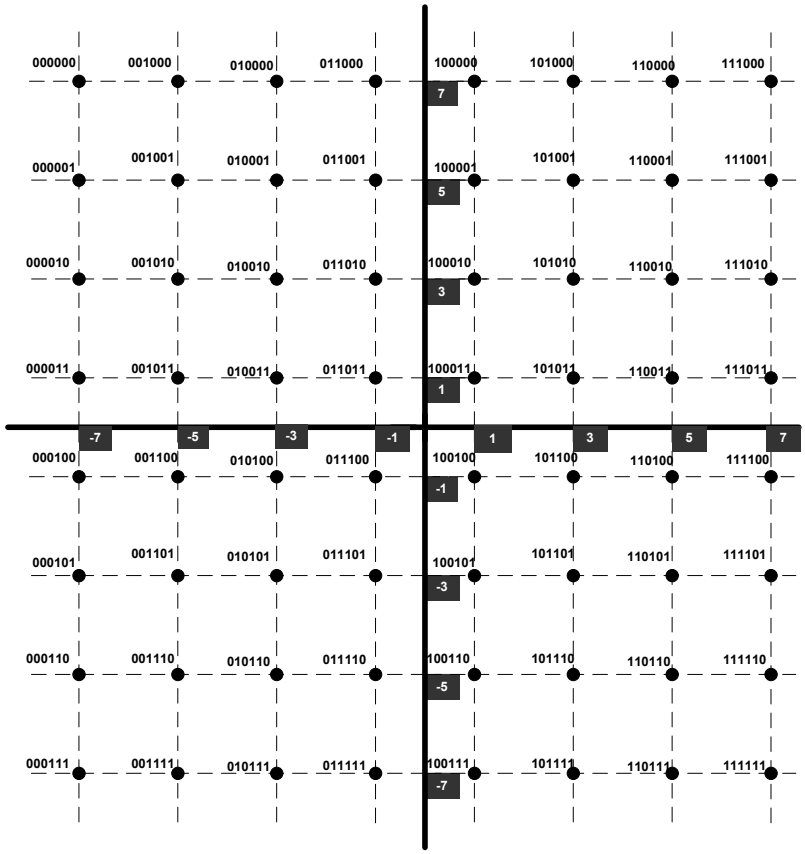


Figure 5.3: The 64-QAM constellation that will be used for the constellation experiments

As can be seen, a *costline* is formed by the set of points in a search space that share the same l^1 norm cost value. This, of course, is not the optimal solution for decoding given that the l^2 would have a better cost distribution which leads to a more unique traversal of

cost from min to max. But, we argue that the l^2 cost space for a 64-QAM system would be prohibitively large to sort and traverse. It is to this end, that we focus out energies on the l^1 cost. The constellation that is going to be adopted for the constellation experiments is shown in Fig. 5.3

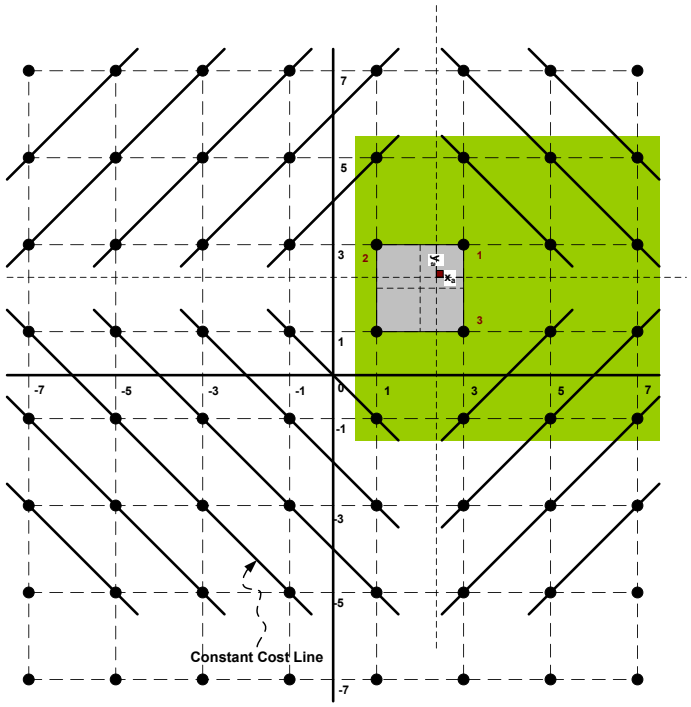


Figure 5.4: Distribution of the real and imaginary contributions to the l^1-norm cost of nodes

All decisions on the order of node visits are made on the basis of the offset values formed by the result of $\hat{y}_i - R_{ii}s_i$ for a given level. Again, it is to be noted that there exist as many $R_{ii}s_i$ values as there are constellation points *i.e.* 2^Q. The result is the need to traverse this space in the order defined by the l^2 norms given that the l1 norms have the constant cost lines shown in the figure below. Though the l^1 cost values can be retained, the l^2 cost decisions would be the ones that decide which points on the constant cost lines are going to be sent into the cost computation first a min cost first manner. We will have to move from a lower cost line to a larger cost line as we traverse the solution space.

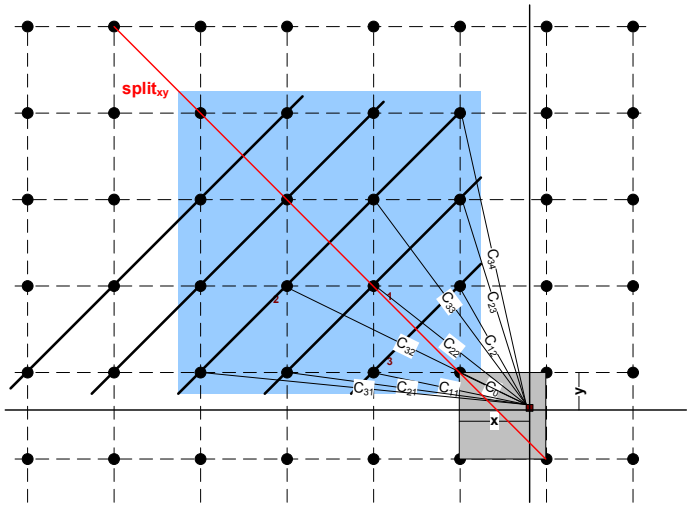


Figure 5.5: Example node distribution for Case a: single quadrant evaluation

5.2.2 Case A: *l1* guided decisions for one quadrant

The sign value of the x and y offsets are of the same sign i.e. the point under consideration is outside the scaled constellation map defined by the pre-computed $R_{ii}s_i$ space. Let us look at the cost values from min to max: The costs that would be computed in the NodeCost by virtue of the use of the sorted offset based method would be $C0$, $C11$, $C22$ and $C12$. Let us look at $C11$, $C22$ and $C12$ in greater detail using l^2 norm.

$$\text{X oriented } C11 = (x+K)^2 + y^2 = x^2 + y^2 + 2Kx + K^2; \tag{5.2}$$

$$\text{Y oriented } C12 = x^2 + (y+K)^2 = x^2 + y^2 + 2Ky + K^2; \tag{5.3}$$

Therefore, on simplification, $C11 < C12$ iff $x < y$. Thus, at first glance, across splitxy, the decision in favor of the x-oriented of y-oriented point is done using the initial offset in x or y. Let us now consider the next constant costline to confirm this:

$$\text{X oriented } C21 = (x+2K)^2 + y^2 = x^2 + y^2 + 4Kx + 4K^2; \tag{5.4}$$

$$\text{Diagonally oriented } C22 = (x+K)^2 + (y+K)^2 = x^2 + y^2 + 2Kx + 2Ky + 2K^2; \tag{5.5}$$

$$\text{Y oriented} C23 = (y+2K)^2 + x^2 = x^2 + y^2 + 4Ky + 4K^2; \tag{5.6}$$

Therefore, on simplification we automatically see that the $C22$ case is the min given that $x + y$ is always going to be less than $2(x/y) + K$ since $x < Kandy < K$. The decision between $C21$ and $C23$ follows the same logic as the previous case i.e. $C21 < C23$ iff $x < y$.

At the next level, using the same logic, we immediately see that iff $x < y$ $C31 < C34$ and $C32 < C33$. Moreover, $C31/C34$ is always going to be less than $C32/C33$. Hence, if $x < y$, the min to max would be $C32 < C33 < C31 < C34$.

We have a generic rule for the growth in one direction which bases itself off of the offset coming in to determine the minimum to the maximum costs along a given line of constant cost. Also, the x and y under consideration would be different for different for cases B and C where would have more constant cost planes in more than one direction and things might get hairy when we have more than one quadrant that needs to be considered for candidates.

5.2.3 Case B: 2 Quadrant Case

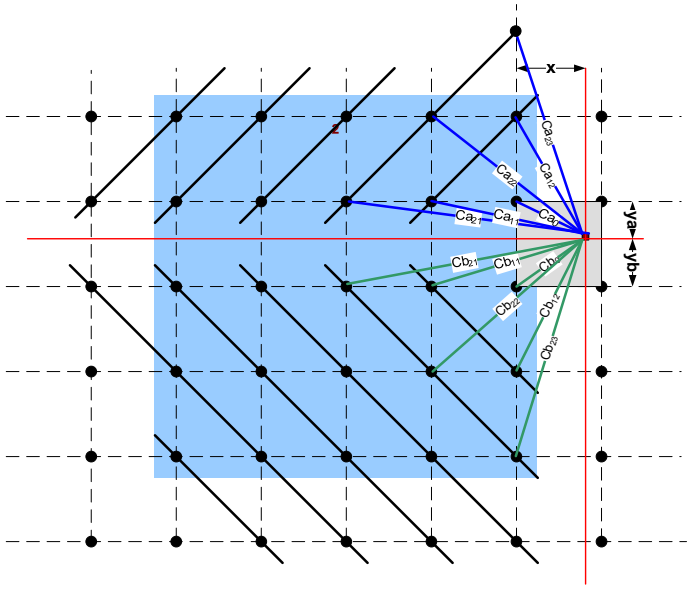


Figure 5.6: Example node distribution for Case b: double quadrant evaluation

The analysis done previously would hold good within each quadrant of analysis

i.e. Ca or Cb. Therefore, given that $x > ya$ and $x > yb$ (here) we can conclude, from our previous analysis, the following progression of cost (min - max)$Ca0 < Ca12 < Ca11 < Ca22 < Ca23 < Ca21$ and $Cb0 < Cb11 < Cb12 < Cb22 < Cb21 < Cb23$

The question lies in the choice between the costs at similar cost levels in the 2 streams. Let us begin by considering $Cb11, Cb12, Ca11$ and $Ca12$. $Cb12 = x^2 + yb^2 + K^2 + 2Kyb$;$Cb11 = x^2 + yb^2 + K^2 + 2Kx$ $Ca11 = x^2 + ya^2 + K^2 + 2Kx$; $Ca12 = x^2 + ya^2 + K^2 + 2Kya$

After removing the common terms (x2 + K2) we get,

$$Cb12 = yb2 + 2Kyb; Cb11 = yb2 + 2Kx; \tag{5.7}$$

$$Ca11 = ya2 + 2Kx; Ca12 = ya2 + 2Kya \tag{5.8}$$

The choices between the costs in the same region of constant line growth (a/b) follows the same trend as the previous cases where the decision is made by virtue of the relative magnitudes of x v/s ya and x v/s yb . The issue lies in the determination of the relationship of the costs between the regions that things get complicated. A few cases with variable relationships between x, ya and yb might provide some light on the decision making process.

Thus, greater though needs to be put in when traversing the nodes in sorted order. To put it simply, the regions of operation (given their symmetry for the ya and yb axis) would follow the zone rules shown below:

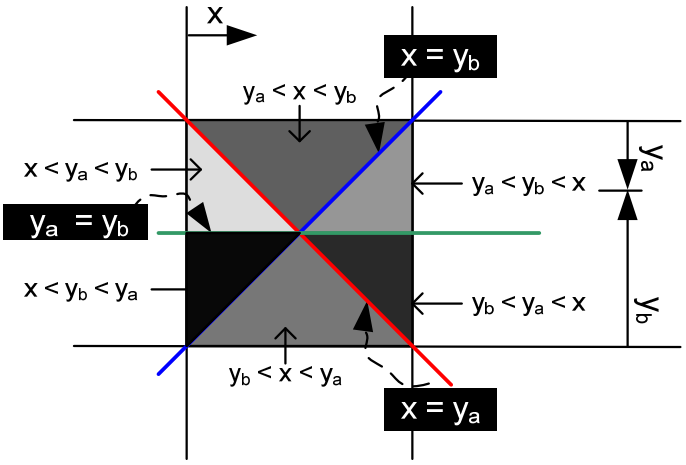


Figure 5.7: Generic rules for assignment of minimum based on offset

5.3 Implementation of l^2 guided l^1 architecture

Simple l^1 constant cost line solution with l^2 awareness.Here we do not concern ourselves with the choices between the confusing l^2 based cost decisions between a,b,c and d quadrants. We are just interested in traversing the l^1 cost space in the order of increasing costs with the use of some smarts for determining the order of costs going into the pipeline. These smarts, based on previous experience, is going to be based on the x or y oriented nature of each quadrant of growth. We have learned from the exercise from the previous section that the direction of lower PED is in keeping with the orientation of the node A

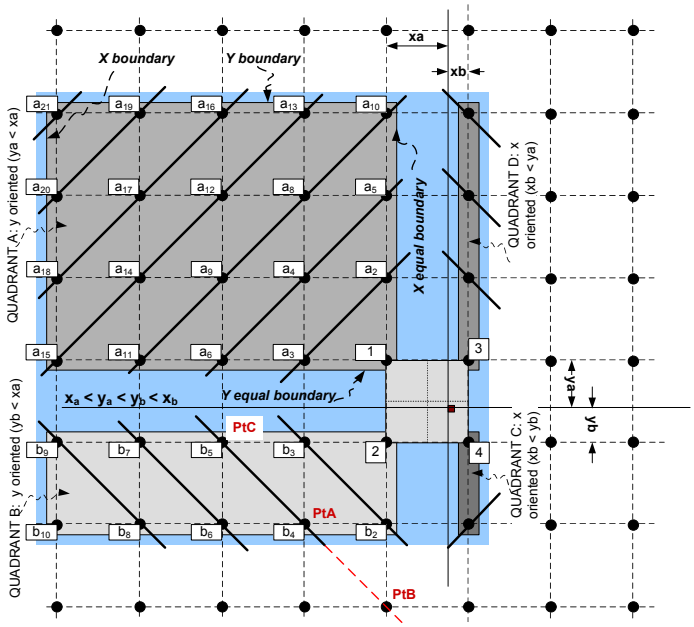


Figure 5.8: Example distribution of nodes into quadrants for a search at a given level in the tree

typical search space for the sphere decoder at a given level in the tree would take the form shown in the Fig. 5.3. Let us define a few terms in the context of this example to help us evaluate the solution we are proposing: a) we divide the search space into four ***quadrants*** A, B, C and D in keeping with the direction of growth of the search from its respective origin b) each *quadrant* has a progressively expanding from the origin where, due to the nature of the l^1 norm consists of a set of constant ***costlines*** which share the same l^1 PED value c) each candidate on a *costline* would be called a ***node*** which has a fixed unique index

and PED associated with it and d) when we are done with a given *costline* we move to a new *costline* at a different ***costlevel*** in the same quadrant. It can be observed that the number of nodes for a given costlevel increases linearly as we move from one costlevel to the next. Also, the point with respect to which we are sorting the search space is called the ***evaluation point***. To capture the essence of the search, we can note that the quadrants are constrained by boundaries which coincide with a) the end of the search space in the x and y directions which do not change and b) the limits imposed by the real or imaginary value of the offset being equal for the node. These boundaries along with the sphere constraint and cycle count restrictions, if any, form the limit on the search in any given quadrant and hence the entire search on the whole.

Let us consider Quadrant B. If there was no boundary to the search space, when we transition from one level to the next would be to go in the y-direction and choose PtB and then x (PtC) given that this is a y-oriented quadrant. But the bounded nature of the space, forces the transition to PtC after PtA given that PtB falls outside the valid search space. These sorts of decisions add to complexity of determining the sequence of nodes in a time / area optimal manner.

To begin with, we note that the search for the solution in the l^1 norm case can be performed with the presence of information corresponding to the 4 closest points to the evaluation point in the search space. There might be fewer required for the case when the evaluation point is outside the search space. The information needed to perform the search is:

1. *PEDin[0-3]*: the costs corresponding to the 4 base nodes:
2. *xoryin[0-3]*: the orientation of the nodes
3. *ind_realin[0-3]*: the real indices of the base nodes
4. *ind_imagin[0-3]*: the imaginary indices of the base nodes
5. *quadin[0-3]*: quadrant information for the base nodes
6. *survin[0-3]*: 0 =¿ incoming PED does not satisfy sphere criterion

In addition to this we would need to keep track of whether the search space corresponding to a quadrant has run out of candidates or not. This could happen if the sphere criterion

is breached or the search space has reached its boundaries. This status is captured using *finished[0-3]*. Using these inputs and assuming *surv[0] != 0*, the basic algorithm to obtain the next node at a given level would take the form shown in algorithm3:

Algorithm 3: **proclevel()** Algorithm for next node evaluation for a given level

Input: All the incoming node information listed above with *finished[0-3]*,
minPED: minimum final cost for the current tree search
Output: Output *PEDout* and *indexout* for that level
Let, i = *quadrant* pointer
clevel[i] = *costlevel* pointer for quadrant i; **Initialize**
for $i \leftarrow 0$ **to** 3 **do**
  finished[i] = *survin[i]*; *PED[i]* = *PEDin[i]*; *xory[i]* = *xoryin[i]*; *i_real[i]* = *ind_realin[i]*; *i_imag[i]* = *ind_imagin[i]*
  quad[i] = *quadin[i]*
  clevel[i] = 0;
end
i=0;
while *finished!= 0* **do**
  {until the case where the entire level is done}
  PEDout = *PED[i]*
  indexout = i_real[i]$\times NUMANT$ + i_imag[i]
  if ***BOUNDARY****(i,j)* **then**
    UPDATE(*i_real[i]*, *i_imag[i]*, *xory[i]*, *quad[i]*)
  else
    if ***VALID****(i,minPED,PED[i])* **then**
      PED[i] = *PED[i]* + *K*
      clevel[i] = *clevel[i]* + 1
    else
      finished[i] = 0
    end
    i = ***NEXTQUAD***(*finished*) i ∈ [0,3]
    UPDATE(*i_real[i]*, *i_imag[i]*, *xory[i]*, *quad[i]*)
  end
end

Validity of a given quadrant implies the presence of further possibilities either at the same level of next level. When validity is 0 it implies there is no reason to attempt any further progress in that quadrant. The most important issue to deal with is the recognition of the BOUNDARIES AND ORDER for a given constant line. This is where the complexity lies. In the above, the complexity is captured in the functions ***VALID***, ***BOUNDARY***,

UPDATE and ***NEXTQUAD***. The ***VALID*** function needs to determine if the feasibility of further growth in a quadrant for which we need to consider the *minPED* and the current *PED[i]* for that quadrant. The ***BOUNDARY*** function determines presence of further candidates in the same *costlevel* for a given quadrant. The ***NEXTQUAD*** determines the next valid quadrant when the current quadrant is done which requires the *finished* vector to evaluate the candidates. It is interesting to note that the next node might have to be from the same quadrant if all the other nodes are no longer valid. Finally, the ***UPDATE*** function will have to create the next real and imaginary index values based on the current index, the x or y orientation and the current quadrant. The current quadrant comes into the picture by virtue of its influence on the direction of growth of the node search.

All of the above subroutines can be boiled down to the determination of the boundaries. This is because, per the above algorithm, we progress along a given *costline* until the boundary is breached and then move to the next *costlevel* for that quadrant. This directly influences the node that needs to be sent in increasing order of cost. This is best described in the Fig. 5.3:

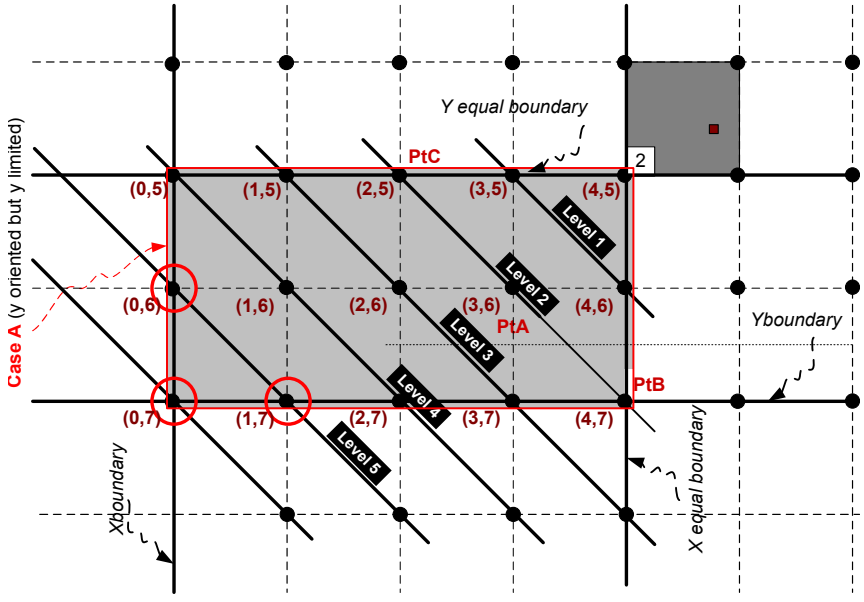


Figure 5.9: Example Node distribution in a quadrant with annotated real and imaginary indices

The entries in red correspond to the real and imaginary indices of the node in

question. As can be seen, the origin for this quadrant is at (4,5). Thus, assuming this is a y-oriented quadrant, we have the following progression of nodes: *clevel 0: (4,5); clevel 1: (4,6), (3,5); clevel 2: (3,6), (4,7), (2,5); clevel 3: (3,7), (2,6), (1,5); clevel 4: (2,7), (1,6), (0,5), clevel 5: (1,7), (0,6), clevel 6: (0,7).* Thus we see that we experience nothing but the *x-equal* and *y-equal* boundaries for the first three *costlevel*s but have to deal with the *yboundary* for the next 2 while we have both *xboundary* and *yboundary* after that. We know we are done with this level, ignoring the sphere constraint, when we hit the corner (0,7) for this quadrant. This is somethings that we will need to note and exploit when implementing the logic for the algorithm.

The first major modification that comes into effect is the modification of the node-cost unit to produce just the 4 base costs and the relevant quadrant and orientation information. The modified implementation takes the form shown below:

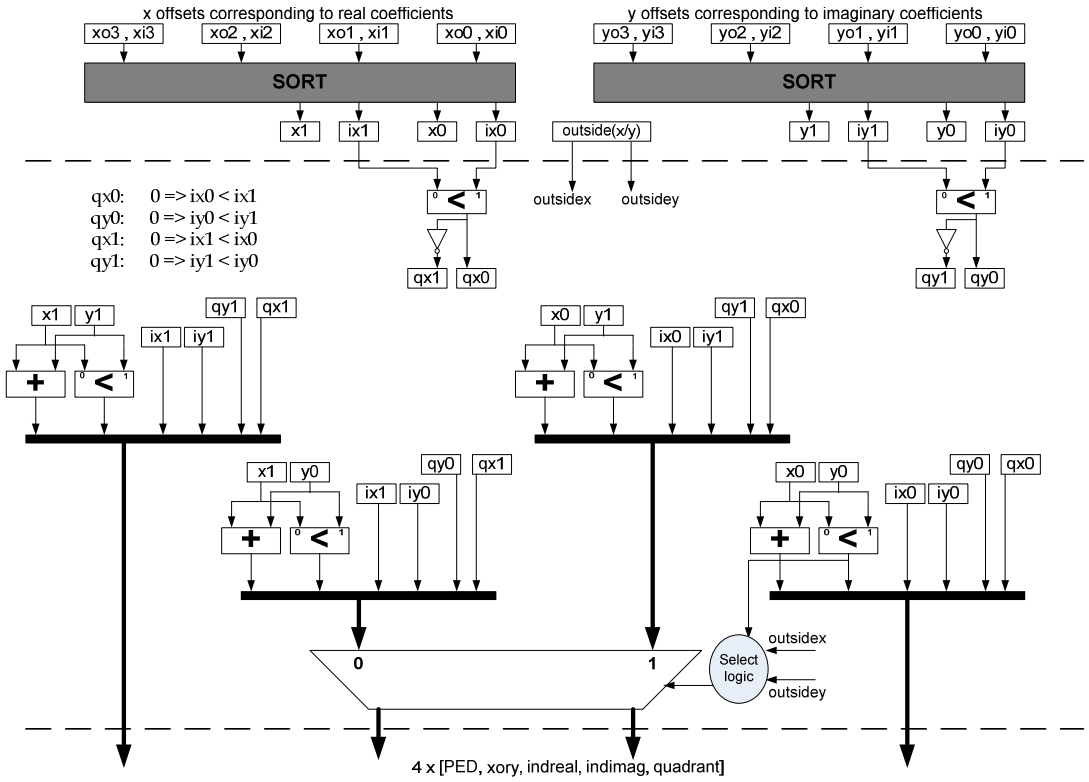


Figure 5.10: Architecture for scalable NodeCost Unit requiring independent processing of real and imaginary axes

The output of this modification are the 4 closest PED values to the *evaluation point i.e.* 1, 2, 3 and 4 in 5.3 as well as the directional preference of each quadrant encoded in the *xory[3:0]* and the corresponding real and imaginary index values of the 4 nodes in keeping with the above algorithm. The next step lies in dealing with the incoming information at the relevant memory interface. The above structure can be simplified to remove the need for sorting. This will be dealt with in the next section.

To begin, we need to get the larger picture here. The algorithm for processing a given search space shown above corresponds to a given level. For a system with $NUMANT$ nodes we will have to deal with the arrival of information corresponding to progressively lower levels in the tree search. Thus from the perspective of the entire decoding process, we need to be able perform the algorithm shown below. In this case, we use *context* to represent the combination of *(xory, PED, ind_(real/imag),quad,surv)[0-3]* to come up with the algorithm shown in algorithm 4

What we can note from the above algorithm is the need for memory processing in two cases

1. When a node needs to be sent into the cost computation resource *i.e.* when *enable* goes high for that memory. At this time, the next valid node from this memory is sent out which corresponds to the current active context *currPED, currlevel, currindex*. Along with this, the **proclevel()** function is called to update the status of the search at the current level *currlevel*. This results in the creation of the next valid node for the current level, if any exist. If there are no valid nodes left at the current level then a search for the next valid (higher) level is performed using the ***VALIDLEVEL*** to update the current node.

2. When a new processed *context* comes in from the cost computation resource. If there are any survivors in the incoming *context*, the current context at the relevant memory interface is pushed into the context stack *mem*. Along with this, the incoming context is made the current context. It is important to note that this method is in keeping with the depth first search approach since we are going to attempt to move down the tree search to lower nodes until we reach the leaf of the tree. Thus, any incoming context with a survivor is going to be the source for the next node for processing.

To enable **proclevel()** to be implemented, we begin by dividing the search within

Algorithm 4: **processmem** Algorithm for creating next node for a given memory

Input: *enable*: seeking memory read; *context*
Output: Output *PEDout*, *indexout*, *levelout*
Let, *incomplete[0-(NUMANT-1)]* = to determine completion of level processing
currPED = curr PED candidate
currlevel = curr level candidate
currindex = curr index candidate
currquadinfo = information for quadrant processing at *currlevel*
mem[0-(NUMANT-1)] = storage for level information **Initialize**
for $i \leftarrow 0$ **to** $NUMANT$ **do**
  incomplete[i] = 1
end
currPED = 0
currlevel = NUMANT
currindex = 0
while *incomplete!= 0* **do**
  {until the case where the entire context is done}
  if *incoming context* **then**
    if $surv_{context}$*!=0* **then**
      if $level_{context}$ *!=1* **then**
        mem[currlevel] = currlevel, currPED, currindex, currquadinfo
        proclevel:Initialize(*context*)
        currlevel = currlevel-1
      else
        minPED = $PED_{context}[0]$
      end
    end
  end
  if *enable* **then**
    PEDout, levelout, indexout=currPED, currlevel, currindex
    currPED, currindex, incomplete= **proclevel**(*currquadinfo*, *minPED*)
    if *incomplete[currlevel] == 0* **then**
      the level is completely processed
      currlevel, currPED, currindex, currquadinfo =
      VALIDLEVEL(mem, incomplete)
    end
  end
end

a given quadrant along the xy diagonal. This in keeping with Fig. 5.3 where we note that the progression along a given quadrant takes the form of growth along the real and imaginary dimensions when we travel along a given *costline* in search of nodes to send out. The choice between the real and imaginary candidates along a given line is made alternatively as *real* $\rightarrow$ *imag* $\rightarrow$ *real* $\rightarrow$ *imag* $\rightarrow$ *real* and so on. The choice on starting with *real* or *imag* in this progression is made based on whether the quadrant is x or y oriented. In keeping with this decision we can maintain the progession for a given quadrant in two base candidates *BaseX* and *BaseY*. The overall implementation of the algorithm for the processing of a single level to create the next set of valid nodes and flags takes the form shown in Fig. 5.11. The implementation shown performs the relevant computation to create the next node in both the X and Y directions of growth along a cost line for current level being processed. It also provides for the determination of movement from one quadrant to the next at a given *clevel* in keeping with Algorithm3

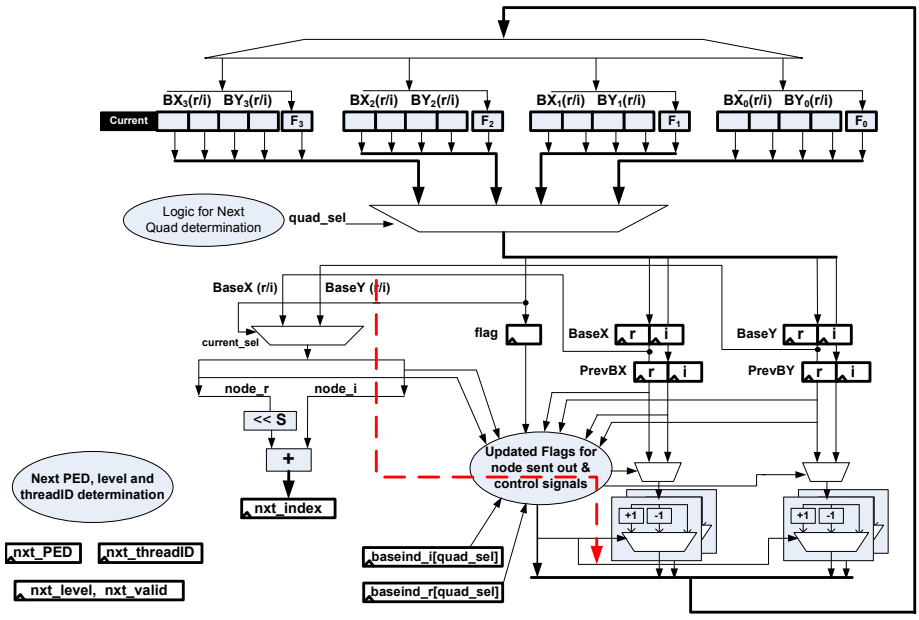


Figure 5.11: View of system organization for processing of a quadrant in keeping with algorithm 3

All the relevant information for encoding the status of the search at a given quadrant of the current level is stored in the form shown in Fig. 5.3 to create the current quadrant status register. 4 of these status registers form the requisite information for encoding the

status of the current level. There would be $nT - 2$ such storage elements required to back up the current level as shown in Algorithm4 thus leading to an organization of the memory to be as seen in Fig. 5.3

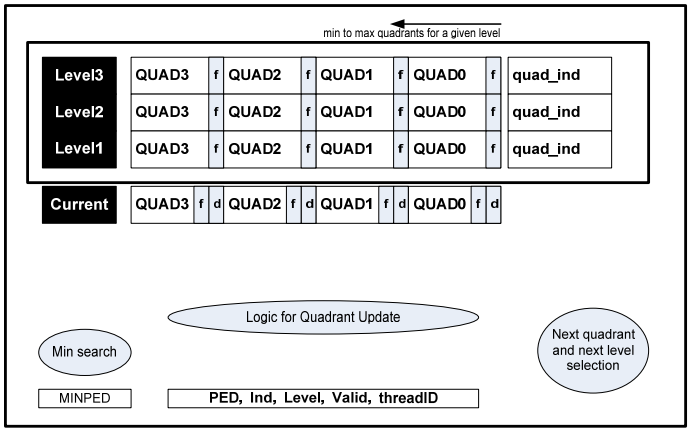


Figure 5.12: Organization of memory subsystem to deal with arrival of nodes at a lower level

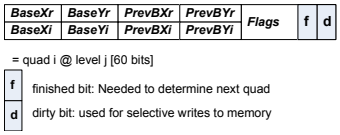


Figure 5.13: Register Organization for encoding status of a single quadrant search

5.4 Evaluation of Scalable Architecture

We evaluate this architecture first by viewing the improvements seen in the 4x4 antenna configuration case with 16-QAM constellation. The BER performance and the cycle count values for this scalable implementation is seen in Fig. 5.14

It is clear from the above analysis that this architecture provides the expected optimal l^1 solution while not sacrificing cycle counts. To provide a realistic throughput number comparison, results of this architecture in comparison with the best case l^1 architecture in the previous section is shown in Table 5.3.

There is an increase in the overall datapath delay by because of the logic for

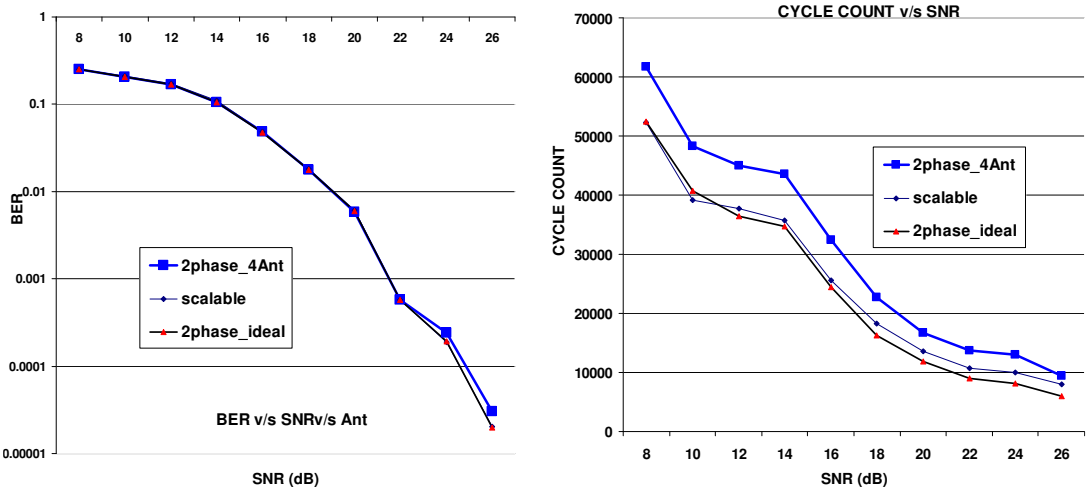


Figure 5.14: The BER performance and Cycle count characteristics of scalable architecture for 4x4 16-QAM decoding

Table 5.3: Power @ 10ns clock(mW) and Area (sq μm) and Critical Path delay comparisons for Scalable solution in 4x4 configuration

Arch	P_{mem}	P_{data}	P_{sphere}	A_{mem}	A_{data}	A_{sphere}	D_{crit}	D_{logic}
$l1$	9.18	8.49	17.67	310.136 K	135.633 K	447.856 K	2.11	11.2507
Current	4.84	6.949	11.789	259.533 K	72.686 K	335.377 K	2.114	11.9799

the creation of the 4 base offsets: 2 in the real and 2 in the imaginary axes. Additional contributions come from the need for creation of special flags like the *quadrant*, *xory* along with the need for manipulation of the order of the nodes to ensure a min-first approach to node visits. The cycle count requirements for this algorithm are seen in Fig. 5.14. It is clear that the use of the l^2 based decisions get us closer to the ideal solution seen in SystemC without the need for extremely complicated sorting requirements. It is of greater importance to determine the scalability of the architecture in terms of the antenna and constellation size.

5.4.1 Antenna and Constellation size sensitivity

This architecture is almost completely constellation insensitive. This is because we no longer need the requirement of sorting 2^Q nodes. We reduce the complexity to the identification of the lowest PED nodes in both the real and imaginary axes. This problem

Table 5.4: Area Scaling with the number of antenna and constellation size for scalable architecture presented

Antennas	Constellation	Datapath Area	Memory Area	Sphere
4Ant	16-QAM	72.686 K	259.533 K	335.377 K
4Ant	64-QAM	95.474 K	259.533 K	355.007 K
6Ant	16-QAM	98.594 K	367.451 K	466.045 K
6Ant	64-QAM	121.382 K	367.451 K	488.833 K
8Ant	16-QAM	118.756 K	454.390 K	573.146 K
8Ant	64-QAM	141.544 K	454.390 K	595.934 K

is a fairly simple one given that we follow these basic rules:

1. If the *evaluation point* is outside the search space in either the real or the imaginary axes, the offsets would either be all 1's or all 0's. This implies that there exist only one direction of growth for the respective axis. Therefore, this requires only one base node value.

2. For the case when the *evaluation point* in question is within the boundaries of the real and imaginary axis limits, the lowest 2 nodes in the search space correspond to the ones at either side of a change of sign. Therefore, if the offsets for the real axis for a 64-QAM constellation are (from left to right): $-(Base1+2K), -(Base1+K), -(Base1), +(Base2), +(Base2+K), +(Base2+2K), +(Base2+3K), +(Base2+4K)$, the lowest two nodes would be $Base1$ and $Base2$ at real indices 2 and 3. This follows from the sign of the offsets being (1,1,1,0,0,0,0,0).

The Memory is completely independent of the constellation size. This is where the true advantage of this architecture makes itself evident. The memory does scale with the number of antennas. The summary of the change in area for different constellation and antenna sizes are shown in Table 5.4

We see that the increase in the area and hence the power of the architecture with the number of antennas follows the same trends as previous architectures due to the need for a larger number of complex multiplications. The true benefits of this architecture come in the larger constellation spaces. It is seen in comparison wth table 5.1 that the the scaling of the area of the memory IP is reduced a great deal in case of the architecture. Additional benefits exist at the interface given that we are no longer dealing with 2^Q values coming in. Instead we are always going to have 4 values coming in that need to be dealt with.

Table 5.5: Power Scaling with the number of antenna and constellation size for scalable architecture presented

Antennas	Constellation	$P_{datapath}$	P_{memory}	P_{total}
4Ant	16-QAM	4.84	6.949	11.78
4Ant	64-QAM	6.475	6.949	13.424
6Ant	16-QAM	6.552	9.838	16.39
6Ant	64-QAM	8.187	9.838	18.025
8Ant	16-QAM	7.683	12.166	19.849
8Ant	64-QAM	9.318	12.166	21.484

The reduced register area goes a long way in reducing both the area, the switching power and the clock tree design constraints. On the datapath side, the increased requirement for sorting the constellation points makes for an extremely non-scalable architecture as described before. The variation of Sphere Decoder power with the number of antennas and constellation is seen in Table 5.5

5.5 Conclusion

It is seen that the unique configuration of the l^1 cost space allows for a far greater optimization of the datapath and memory. These characteristics can be utilized to reduce the power consumption of the sphere decoder by about 70% from the best case l^1 solution for 4x4 system while still not losing out on optimal decoding preformance in the l^1 space. These benefits would be much larger for increased antenna sizes and even greater for increased constellation sizes. On the flip side, the increase in the anteanna and constellation scaling has a far greater impact on the throughput of the decoding process because of the increase in the search space. Pruning in its traditional form fails to scale very well as well due to a larger number of survivors at the higher levels of the tree seach. This translates to a larger number of total nodes that need to be visited. Therefore, it is important to consider sub-optimal solutons that would work not just on increasing the throughput but work on attaining a constant throughput as well. We aim to propose a simple enhancement to the design which would allow for this to be achieved.

Chapter 6

Decoder architecture under throughput constraints

One of the issues with the Sphere Decoder is the fact that the throughput of the system is variable. This is due to the fact that different symbol vectors take different cycle times to finish given that the solution is dependent on the channel characteristics and SNR. This is made clear in the the Fig. 6.1 where we note the finish times for different SNR values and for different iterations. From a system perspective this is undesirable because a variable throughput leads to the lack of predictability in system timing. A lack of predictability leads to the inability to achieve an optimal design at the level of system integration. This proves to be particularly true for the case where this Sphere Decoder would be an accelerator. It is therefore important to investigate the feasibility of constraining the cycles for analysis while not sacrificing performance greatly. As a possible solution, we propose a block based algorithm that provides user control in the allocation of block level limits on the decoding time.

6.1 Designing the Block Based Decoding Solution

One of the solutions to this issue is the explicit constraining of the number of cycles allocated to each symbol vector. This method is called the Early Termination methodology. Let us call this cycle count limit per Symbol Vector D. We attempt to determine the variation of the Algorithmic performance and the cycle count of this implementation with

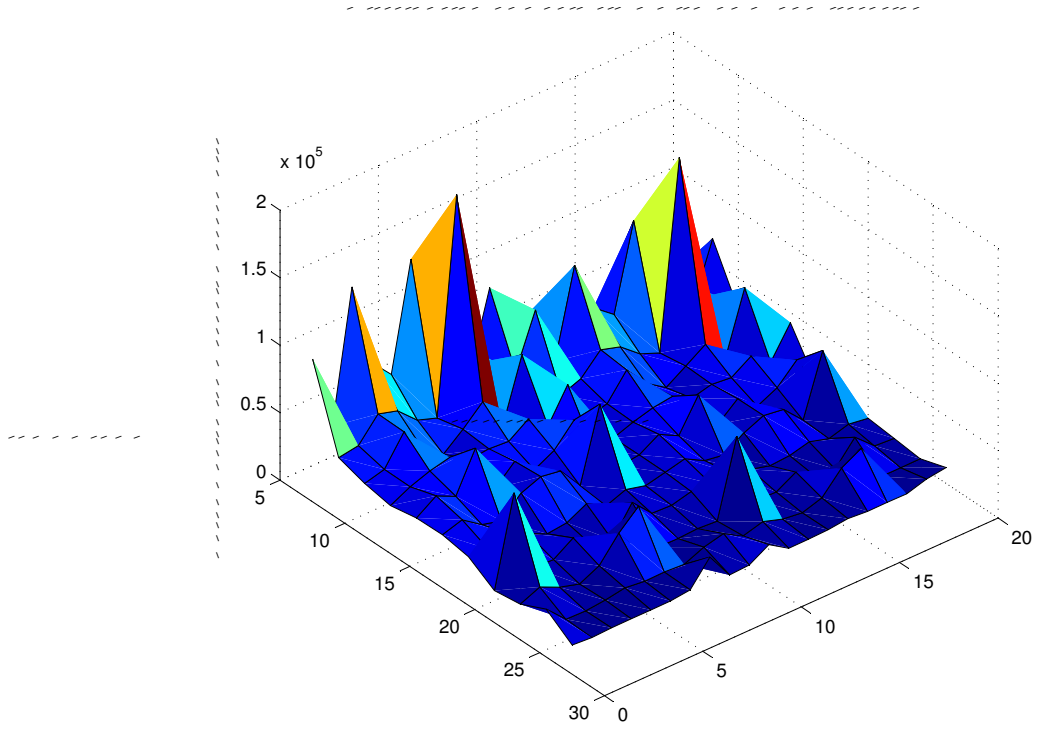
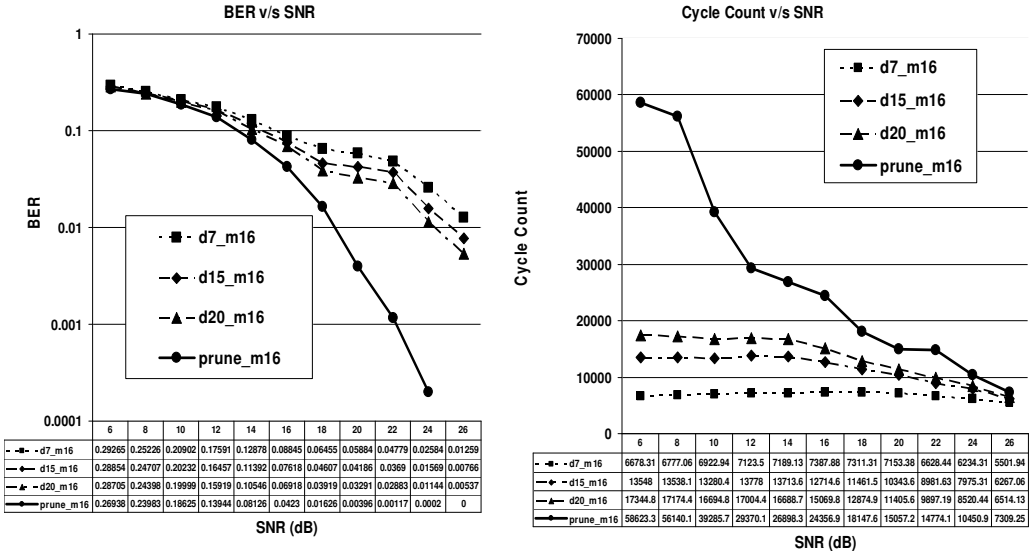


Figure 6.1: The distribution of the Cycle Time over SNR and Iteration number for the pruning algorithm

SNR. This is shown in Fig. 6.2

We also note that this variation does not vary much with the mask given that the run time limits on performance are imposed more by virtue of the termination limit rather than the survivor count. This is shown clearly in Fig. 6.3. On the positive side this methodology seen by the modification of cycle count variability to the form shown in Fig. 6.4

From the above analysis it is seen that a limit imposed on the processing of just one node fails to perform optimally. The ability to keep the throughput constant is also absent due to the early finishing of certain symbol vectors. The creation of a modified block-based algorithm that allows for a smarter decision on the iteration limits for SD. It has been shown in [31] [48] that a block-based approach with run-time constraints SD provides algorithmic performance that closely matches the unconstrained SD implementation (MLD). The advantage in using this approach is the ability to place a bound on the number of cycles that will be consumed in decoding a block of data. This also brings with it the ability to



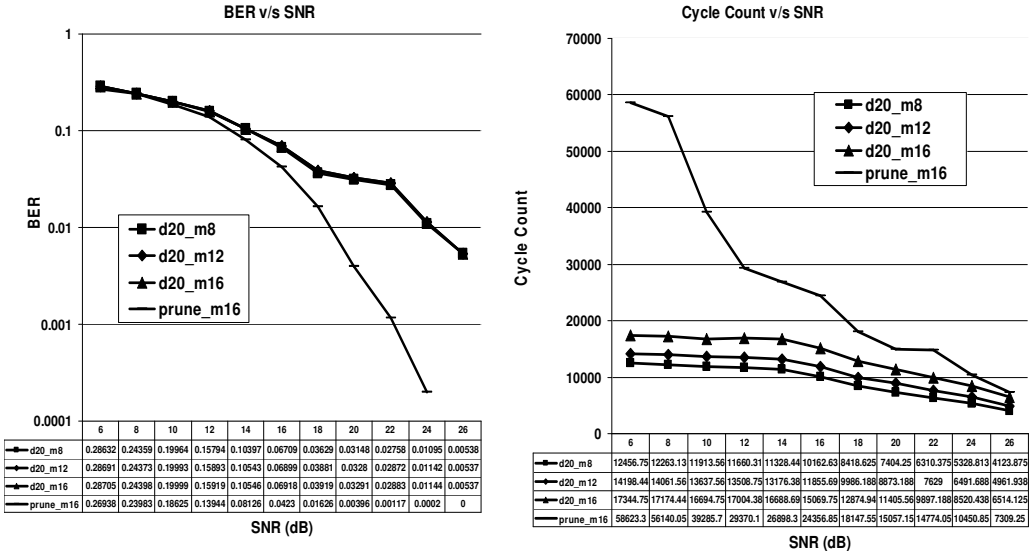
SNR (dB)	6	8	10	12	14	16	18	20	22	24	26
d7_m16	0.29265	0.25226	0.20902	0.17591	0.12878	0.08845	0.06455	0.05884	0.04779	0.02584	0.01259
d15_m16	0.28854	0.24707	0.20232	0.16457	0.11392	0.07618	0.04607	0.04186	0.0369	0.01569	0.00766
d20_m16	0.28705	0.24398	0.19999	0.15919	0.10546	0.06918	0.03919	0.03291	0.02883	0.01144	0.00537
prune_m16	0.26938	0.23983	0.18625	0.13944	0.08126	0.0423	0.01626	0.00396	0.00117	0.0002	0

SNR (dB)	6	8	10	12	14	16	18	20	22	24	26
d7_m16	6678.31	6777.06	6922.94	7123.5	7189.13	7387.88	7311.31	7153.38	6628.44	6234.31	5501.94
d15_m16	13548	13538.1	13280.4	13778	13713.6	12714.6	11461.5	10343.6	8981.63	7975.31	6267.06
d20_m16	17344.8	17174.4	16694.8	17004.4	16688.7	15069.8	12874.9	11405.6	9897.19	8520.44	6514.13
prune_m16	58623.3	56140.1	39285.7	29370.1	26898.3	24356.9	18147.6	15057.2	14774.1	10450.9	7309.25

Figure 6.2: The BER and Cycle Count performance for termination limits of D = 7, 15, 20

place constraints on the hardware requirements both within the system and in interfacing to it when used as an IP. An IP with a rate-limited IO behavior makes for a much simpler process of system integration. The decision on the block size could be as simple as $N \times D_{avg}$ which would allow for a better average behavior with say $D_{avg} = D_{max}$, where N is the block size and D_{max} is the maximum number of nodes that can be visited for a given symbol vector so as to meet a given throughput constraint. For example, if we have a 20M symbols/sec limit and an operating frequency of 200 MHz, we have a $Dmax$ of 10 (assuming we spend 1 clock cycle per node, which is the case here). This does not really do much in terms of improvement given that the number of cycles needed to decode a symbol vector vary around a mean. As an alternative, the allocation of the cycles per decoding could be performed by spreading the total cycles over a block dynamically over many decodings where one might take greater than the average case while another might take take less than the average case. Overall, the average behavior is maintained. For such a case, the allocation of the cycle count (node visitation) limit for a given symbol vector n in a block is per (with M_T = number of transmit antennas):

$$D_{max}(n) = ND_{avg} - \sum_{i=1}^{n-1} D(i) - (N-n)M_T \tag{6.1}$$



	6	8	10	12	14	16	18	20	22	24	26
d20_m8	0.28632	0.24359	0.19964	0.15794	0.10397	0.06709	0.03629	0.03148	0.02758	0.01095	0.00538
d20_m12	0.28691	0.24373	0.19993	0.15893	0.10543	0.06899	0.03881	0.0328	0.02872	0.01142	0.00537
d20_m16	0.28705	0.24398	0.19999	0.15919	0.10546	0.06918	0.03919	0.03291	0.02883	0.01144	0.00537
prune_m16	0.26938	0.23983	0.18625	0.13944	0.08126	0.0423	0.01626	0.00396	0.00117	0.0002	0

	6	8	10	12	14	16	18	20	22	24	26
d20_m8	12456.75	12263.13	11913.56	11660.31	11328.44	10162.63	8418.625	7404.25	6310.375	5328.813	4123.875
d20_m12	14198.44	14061.56	13637.56	13508.75	13176.38	11855.69	9986.188	8873.188	7629	6491.688	4961.938
d20_m16	17344.75	17174.44	16694.75	17004.38	16688.69	15069.75	12874.94	11405.56	9897.188	8520.438	6514.125
prune_m16	58623.3	56140.05	39285.7	29370.1	26898.3	24356.85	18147.55	15057.15	14774.05	10450.85	7309.25

Figure 6.3: The BER and Cycle Count performance for termination limits of D = 20 and $mask = 8, 12, 16$

This constraint has the visibility of just one vector and hence makes a decision based on just that data and the dynamics of its predecessors. In the case of my multi-context approach to design, the visibility of this algorithm is equal to the number of contexts and hence we can make a more judicious decision. This should be able to allow for a more predictable throughput at the level of the block while minimally degrading the SNR performance.

The issues with the present architecture is that the one node having a Linear Equalization (LE) solution in M_T clock cycles is not valid any more. When a node enters the decoding process, it only has an LE solution when after $P \times M_T$ clock cycles, where P is the number of contexts that are being processed in the pipelined architecture (presently stands at 9). But we can still run the block based algorithm with a minor modification by identifying the occurance of the criterion for "**forced completion**". The criterion is the presence of $(N - n)$ un-decoded symbol vectors with only $(N - n) \times M_T$ clock cycles available. At this point,

- The nodes that have already been decoded to the lowest level would be kicked out in favor of the new symbol vectors coming in.
- The nodes that have not reached the lowest levels would be allowed to go just to

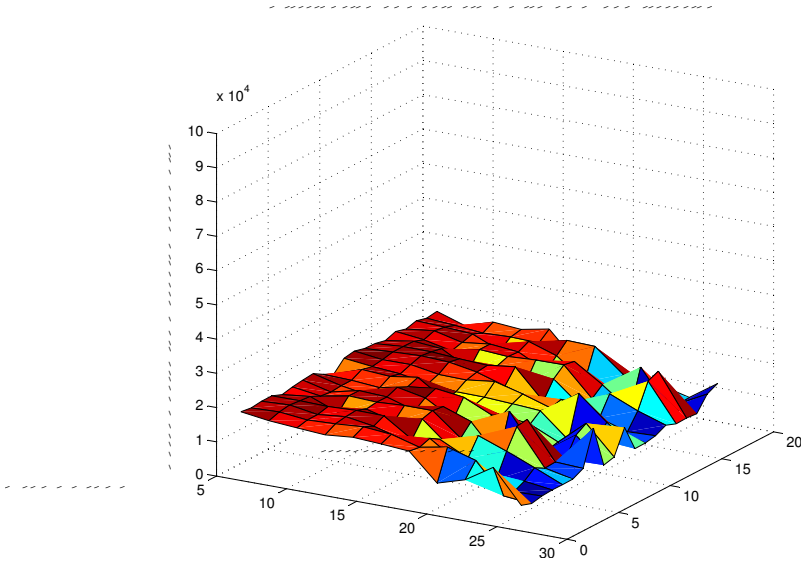


Figure 6.4: The Cycle Count Distribution for for termination limits of D = 20 and $mask$ = 16 in case of the Early Termination methodology

the lowest level i.e. level 1 and get a LE solution. We can also prevent un-necessary computation by allowing the mask to allow just one result of computation at the NodeCost unit to go through and survive.

Thus the idea is to just let things go un-interrupted till we reach the forced completion state at which point we move to a minimalistic mode where only the minimum resources and LE solution mode. The issue therefore is the determine, correctly, the occurance of the forced completion state. In (6.1), the $(N-n)M_T$ constraint does not consider the number of cycles already spent in decoding the symbol vectors in the pipeline. The number of cycles needed to get LE solutions for all the unfinished nodes in the block would be

$$C = ((N-n) \times M_T) - \left[(P \times M_T) - \sum_{(i=1)}^{P} \chi(i) \right] \tag{6.2}$$

where, if $Nvisit_i =$ number of nodes visited by i^{th} context,

$$\chi(i) = 0 \text{ if } Nvisit_i \geq M_T$$
$$= M_T - Nvisit_i \text{ if } Nvisit_i < M_T \tag{6.3}$$

The last term in (6.2) $\sum_{(i=1)}^{P} \chi(i)$ counts the exact number of cycles required to find the LE solution for the Symbol Vectors in the pipeline. Therefore, we can allow for normal operation until the cycles spent decoding the present block reaches

$$DecLimit = N \times D_{avg} - ((N-n) \times M_T) + \left[(P \times M_T) - \sum_{(i=1)}^{P} \chi(i) \right] \tag{6.4}$$

$$= \underbrace{N \times (D_{avg} - M_T) + (P \times M_T)}_{\text{Initial Constant}} + \underbrace{n \times M_T}_{\text{added each time a SV is decoded}} - \underbrace{\sum_{(i=1)}^{P} \chi(i)}_{\text{\# cycles required to get LE soln for SVs in pipeline}} \tag{6.5}$$

Another approach: The number of cycles needed to get LE solutions for all the unfinished nodes in the block would be

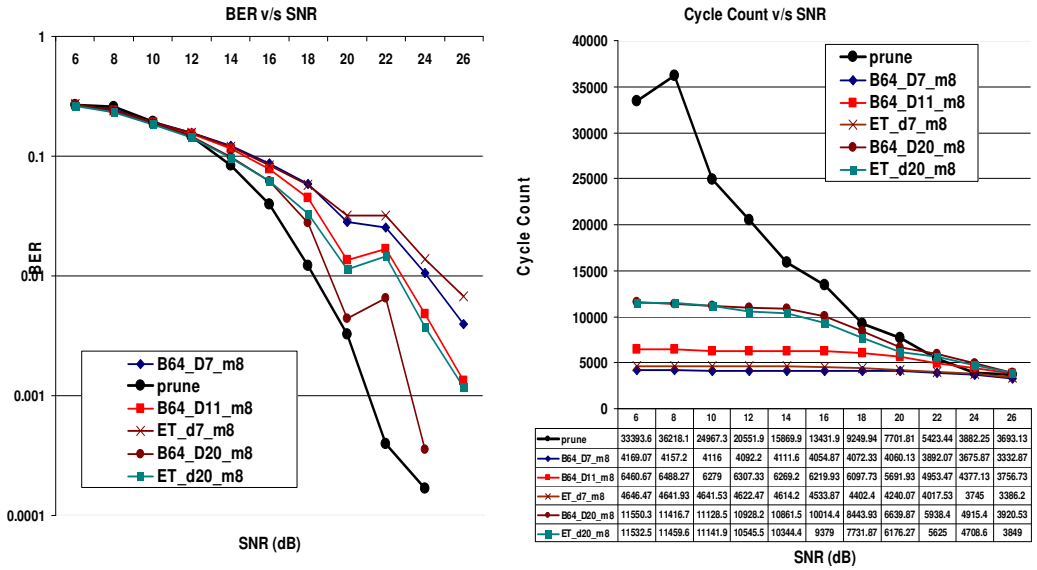
$$C = ((N-n) \times M_T) - \left[\sum_{(i=1)}^{P} \varphi(i) \right] \text{ where, if} \tag{6.6}$$

$$Nvisit_i = \text{ number of nodes visited by } i^{th} \text{ context,}$$

$$\varphi(i) = M_T \text{ if } Nvisit_i \geq M_T$$

$$= Nvisit_i \text{ if } Nvisit_i < M_T \tag{6.7}$$

The last term in (6.6) $\sum_{(i=1)}^{P} \varphi(i)$ counts the exact number of cycles already consumed by the pipeline in an attmept to reach the LE solution. The count for a given context saturates at M_T. Therefore, we can allow for normal operation until the cycles spent decoding the



	6	8	10	12	14	16	18	20	22	24	26
prune	33393.6	36218.1	24967.3	20551.9	15869.9	13431.9	9249.94	7701.81	5423.44	3882.25	3693.13
B64_D7_m8	4169.07	4157.2	4116	4092.2	4111.6	4054.87	4072.33	4060.13	3892.07	3675.87	3332.87
B64_D11_m8	6460.67	6488.27	6279	6307.33	6269.2	6219.93	6097.73	5691.93	4953.47	4377.13	3756.73
ET_d7_m8	4646.47	4641.93	4641.53	4622.47	4614.2	4533.87	4402.4	4240.07	4017.53	3745	3386.2
B64_D20_m8	11550.3	11416.7	11128.5	10928.2	10861.5	10014.4	8443.93	6639.87	5938.4	4915.4	3920.53
ET_d20_m8	11532.5	11459.6	11141.9	10545.5	10344.4	9379	7731.87	6176.27	5625	4708.6	3849

Figure 6.5: The BER and Cycle Count performance of Block Based Methodology

present block reaches

$$DecLimit = N \times D_{avg} - ((N-n) \times M_T) + \left[\sum_{(i=1)}^{P} \varphi(i)\right] \tag{6.8}$$

$$= \underbrace{N \times (D_{avg} - M_T)}_{\text{Initial Constant}} + \underbrace{n \times M_T}_{\text{added each time a SV is decoded}} + \underbrace{\sum_{(i=1)}^{P} \varphi(i)}_{\#\text{ cycles aready spent from minimum quota to get LE soln for SVs in pipeline}} \tag{6.9}$$

We use option two due to ease of implementation. On the use of this method, we attempt to determine the performance of the algorithm with run time contraints imposed by having the block size N = 64 and varying the average allocated cycles per Symbol Vector Decoding to be Davg = 7,11,20. The BER and cycle time performance is as shown in Fig. 6.5 in comparison with the Pruning methodology and early termination. The variation of the cycle count with SNR and iteration number is shown in Fig. 6.6 for $N = 64$ and $D_{avg} =$

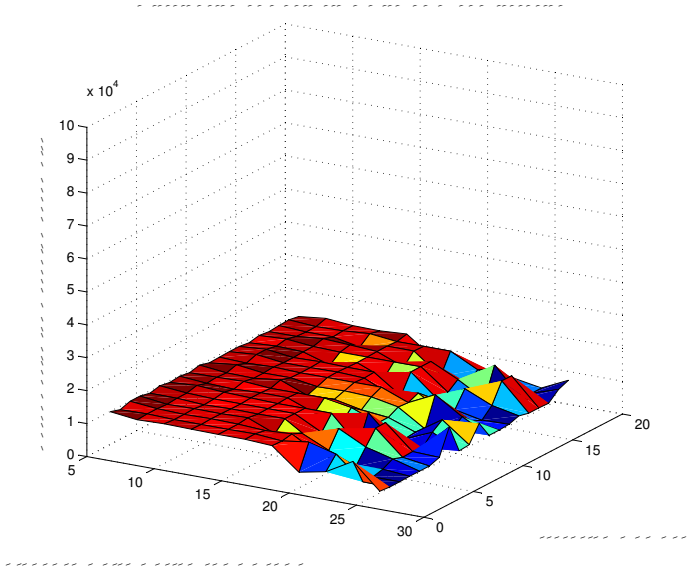


Figure 6.6: The Cycle Count Distribution for for termination limits of $N = 64$, $Davg = 20$ and $mask = 16$ in case of the Block based methodology

20 and $mask = 16$. As can be seen, the variation of the completion times for the decoding of 10000 bits is a lot smoother and hence easier to integrate into a SoC.

It is seen that this architectural modification allows for greater control of the throughput characteristics of the algorithm. The ideal configuration of the block limits N and $Davg$ require a greater understanding of the environment of usage. These control values are easily programmed into registers and can be modified by the controlling unit. This allows for the imposition of constraints only when needed. If a optimal decoding performance is called for, the mode can be disabled. There is no power, area or performance hit given that these options are imposed by the addition of simple control at the memory access unit.

6.2 Conclusion

This chapter provides an initial take on the creation of a block based solution for the time-multiplexed architecture. The aim of this block based architecture is to provide a simple means of controlling the total number of clock cycles assigned to the decoding of a block of symbol vectors. The architecture proposed allows the controlling element, say the software in a SoC, to program a few counters that allocates different limits on the

clock cycles for decoding. It is shown that the values needed for these counters is dependent upon the noise values and the throughput constraint on the system. Using the right counter values allows for a close to algorithmically optimal performance.

Chapter 7

Chip Design of Time Multiplexed Sphere Decoder

The time-multiplexed Sphere Decoder was implemented using a 8-metal 90nm IBM process. The aim of this design was to determine the feasibility of this architecture and to provide numbers for the analysis of the limits of voltage scaling. Additionally, the move to 90nm would provide additional information on the issues that would creep up as we move down the technology node for a wireless implementation such as this. To this end, we implemented the SD architecture with provisions for realistic testing and debuggability of the architecture in case of unpredictable behaviour.

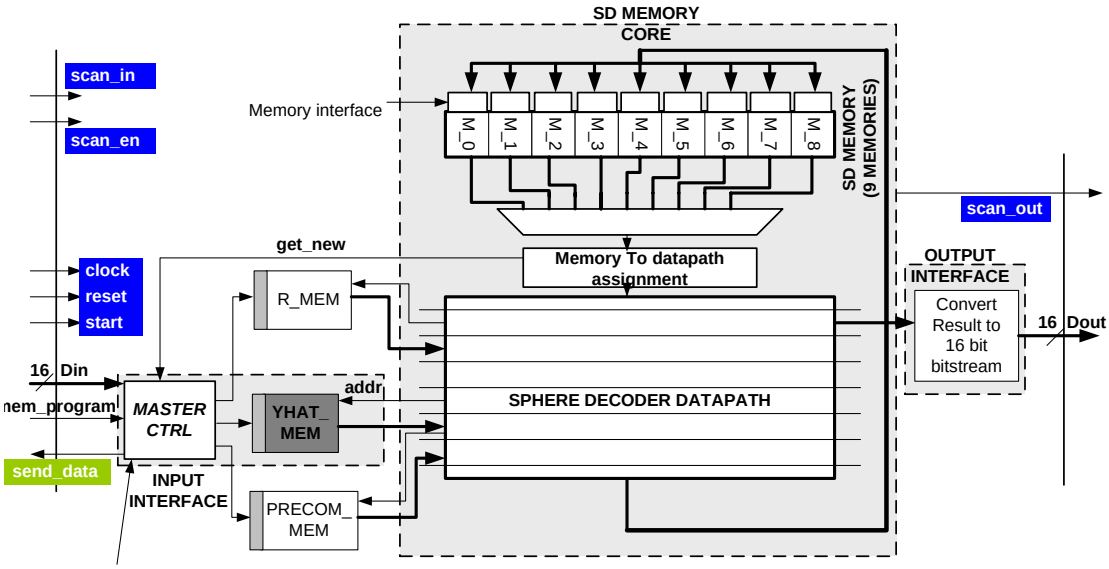
7.1 Chip Design with Testing

To enable testing of the SD design under realistic conditions, we designed an environment which provided inputs to the SD core using vectors loaded into memory. The top level block digram of this thought is shown in Fig.7.1. The vectors were loaded from the input to the chip (`din`) using handshaking involving two signals : `mem_program` and `send_data`. The `mem_program` signal is used to enable the operation of the controller at the input interface. The input interface, in turn, uses the `get_new`, which signifies the completion of a decoding, to create the `send_data` signal which is used by the data provider to assert new data at the `din` bus. The requirement for the loading of the vectors results in the need for two phases of operation of the Chip a) the **initialization phase** where time

is spent on loading the required testing memories and b) the **runtime phase** during which the decoder performs its actual functioning and hence will seek new data values when a decoding cycle is completed. This is visually represented in Fig. 7.2. The initialization phase is broken down into the loading of the Precompute Memory, the R memory and the initial yhat memory. The Precompute memory provides the values that will be used to create the $RiiSi$ signals for the NodeCost Unit, the R memory is used to store the result of the QR decomposition matrix for the channel realization and yhat memory is used to store the $\hat{y} = Q^H y$ vectors where y represents the incoming symbol vector at the receive antenna after the ADC. The SD implementation here uses 9 context memories for the decoding process. This, therefore, results in the scenario at Point A in Fig. 7.2 which corresponds to the need for 9 initial loads from memory corresponding to the 9 context memories present. The point B in the figure represents the scenario where a successful decoding has been performed and hence a request for a new symbol vector to be decoded is sent.

To enable a greater visibility into the Chip a scan chained design was implemented. The coverage for this scan chain involves the entire register chain of the datapath. Additionally, the set of registers that form the current inputs for cost computation are also scan chained. With scan chains, the ability to re-create a snapshot of the entire memory of interest also comes with the ability to insert a known state into the system. Of course, this requires us to be sure that the scan chain does yield and work correctly. The scan chain is particularly susceptible to hold issues by virtue of extremely short paths from the output to the input. This must be remembered when analyzing the correctness of the synthesis result. All paths with scan chain registers should be analyzed for minimum delay at the fastest corners.

The fully annotated final layout view is shown in Fig. 7.4. The final die area was 2.72 mm by 2.82 mm which includes guard rings for easy of dicing the die. The core area of the fabricated design is 2.5 mm by 2.4 mm. The layout shows the breakup of the die area into the two main sub-designs a) the datapath only section and b) the full sphere decoder with test vector memories. The datapath only section consists of the cost computation resource with scan chains for loading the environment for the computation to be performed. This was a contingency plan for the analysis of the die in case the larger design did not yield. The final die area for the Sphere Decoder came out to be 1.44 mm^2 with a significant portion of that area being spent on the analysis memories and their



INITIALIZATION: (CHRONOLOGICALLY)
- Perform Initialization of the following using `send_data` **and** `mem_program` **for handshaking and using** `Din` **to send in data**
`Precomp_Mem (4 x (16 x 2) x Din)*` → `R_Mem(3 x (3 x 2) x Din)` → `yhat_mem (16 x (4 x 2) x Din)`

RUN TIME OPERATION:
- Every time get_new goes high (implying that a decoding is finished):
- Program `yhat_mem (4 x 2 x Din)` **with new values using** `send_data, mem_program` **and** `Din`

***16 x 2 x Din => 16 x 2 clock cycles being spent on reading Din values**

Figure 7.1: Block diagram of SD Architecture with test and debug

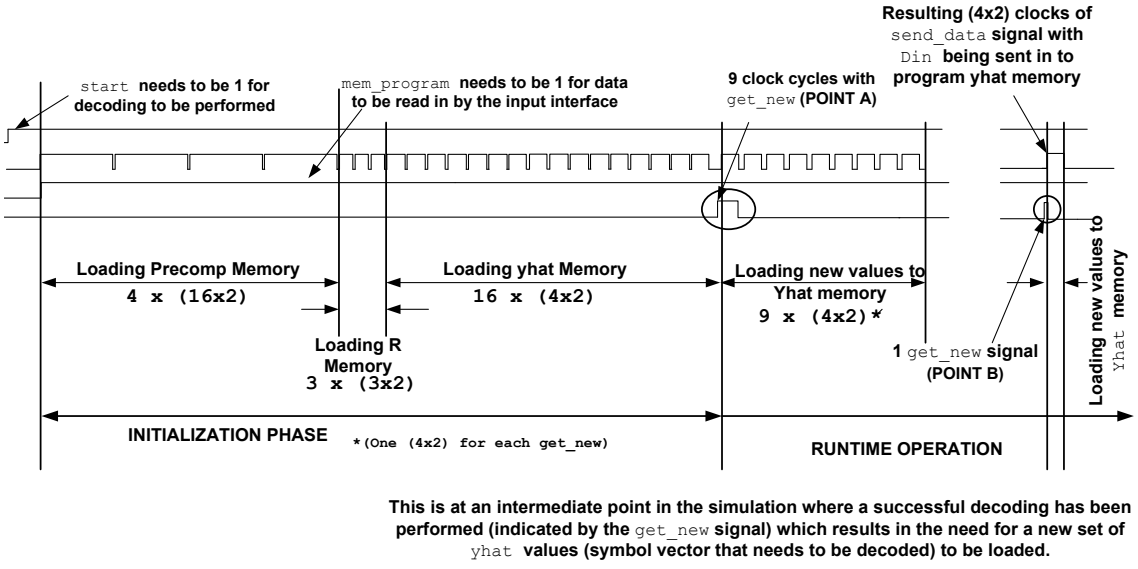


Figure 7.2: Protocol for chip initialization and normal run

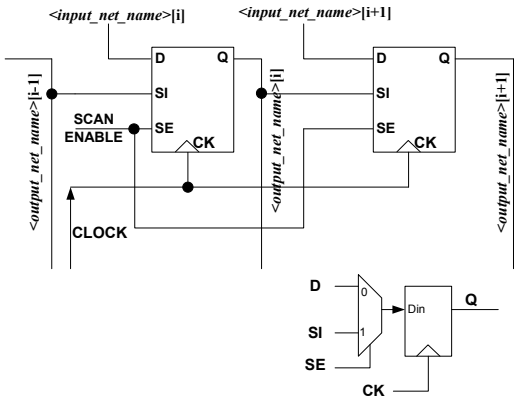


Figure 7.3: Scan Chain inserted into design for increased visibility and register control

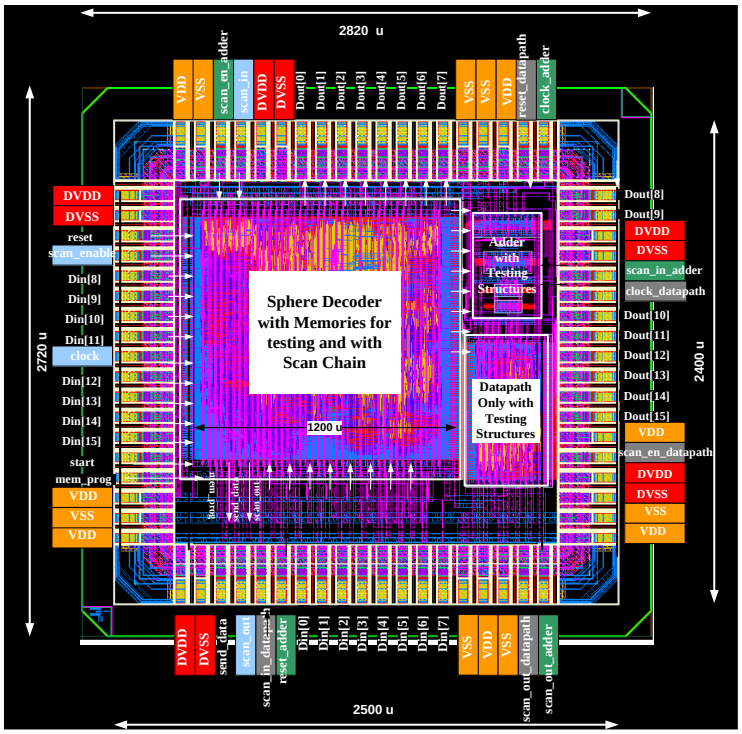


Figure 7.4: Annotated Chip Layout from DFII

interface. This of course makes for a very realistic design given that similar constraints for interfacing with the SD core would exist even for an SoC. The distribution of the input and output nodes was done so as to provide ease of bonding for "sign of life" testing or probing of pads if necessary. The feasibility of probing a pad vanishes greatly with the proximity of the pads required for analysis of a design. To this end, pads for simple testing were placed in different corners of the die.

The annotated photograph of the die received after fabrication is shown in Fig. 7.5. The sections corresponding to the full system and the datapath only section are clearly shown.

After ensuring that the VDD, VSS, DVDD and DVSS signals were not shorted, testing for sign of life was done using 5 inputs: `reset, scan_in, scan_enable, clock, start` and 1 output: `scan_out`. The bonding diagram to a 40 pin DIP package is created as shown in Fig. 7.6. We also bond up the secondary datapath only portion of the chip as

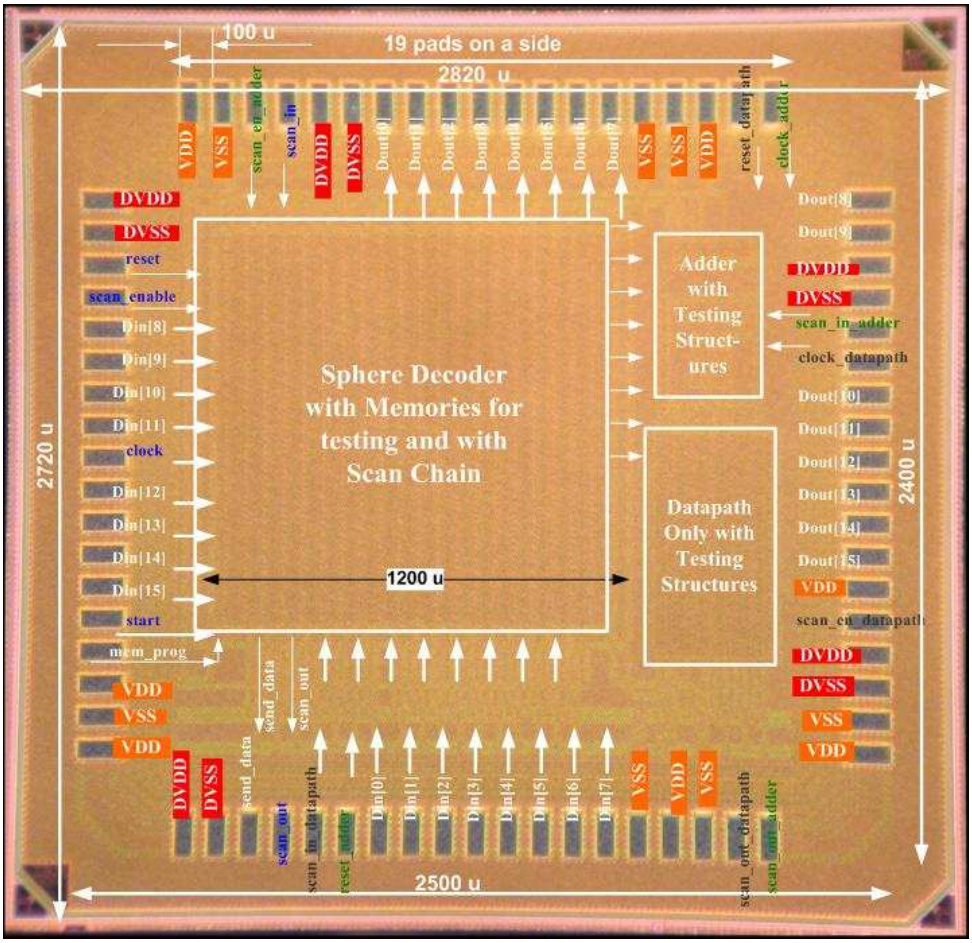


Figure 7.5: Die Photograph of 90nm Sphere Decoder

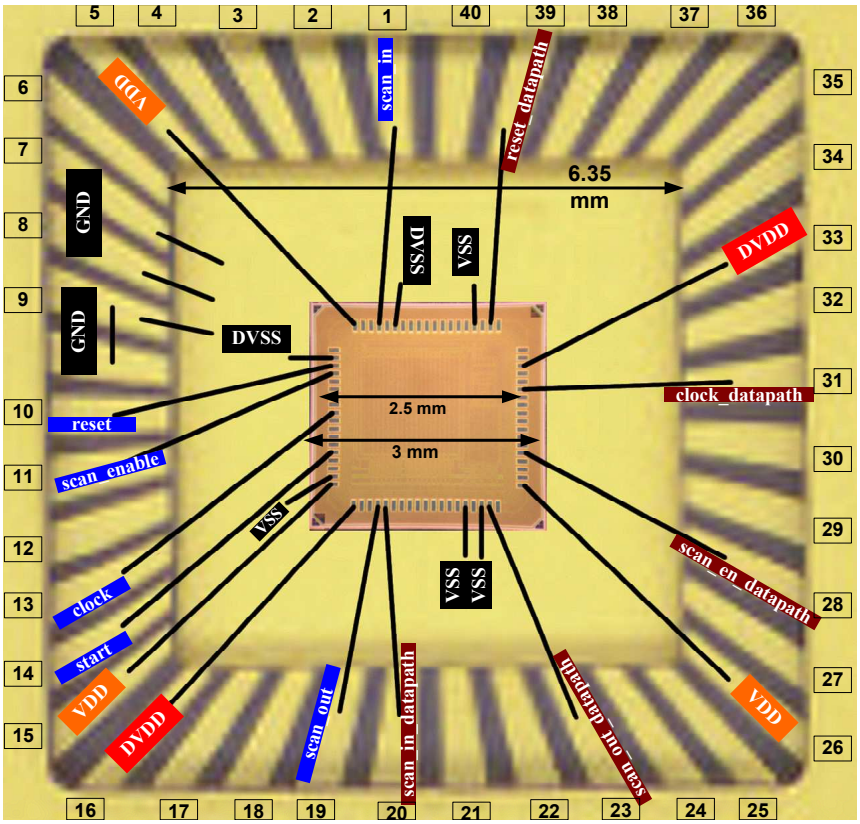


Figure 7.6: Gold bonding plan for sign of life testing using scan chains

a contingency to failure of the Sphere Decoder. The `DVDD` and `VDD` pads were drawn out to package inputs as well to provide isolation of chip and core power connections. The `DVSS` (pad ground) and `VSS` (core ground) were shorted to the cavity which in turn is connected internally to the pin number 10 (not numbered in Fig. 7.6).

A 2 in. $\times$ 4 in. board compatible with 40 pin dip package of choice was design and fabricated. The credit for this board design goes to Samson Melamed from our group. The board was integrated with the package, two 10 μF bipolar capacitors, two 10 μF ceramics capacitors, the requisite header pins and two SMA connectors for the two clocks: `clock_datapath` and `clock_sphere`. The multiple voltage levels and the use of a separate plane for signal routing led to the need for a four layer board. The final integration of all the components resulted in the testing system which takes the form Fig. 7.7. Most of the credit goes to Samson Melamed again.

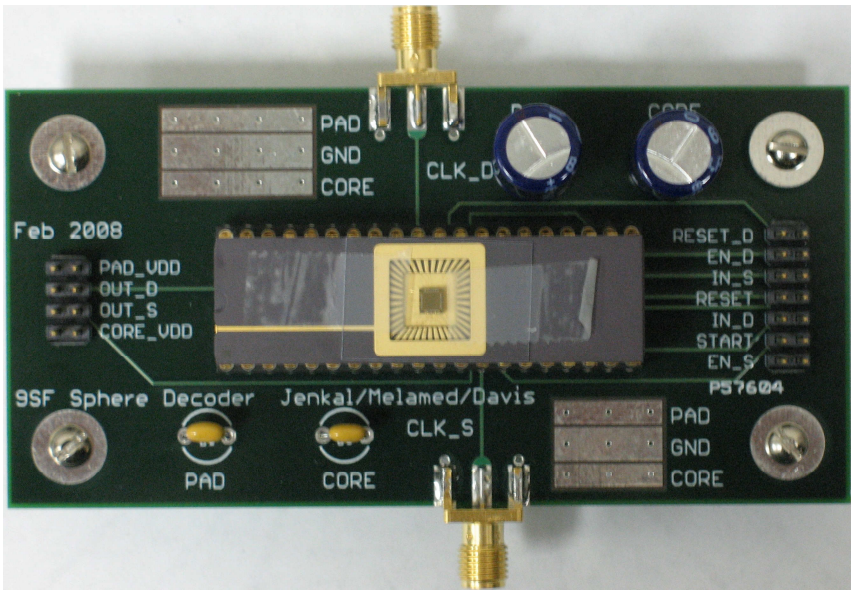


Figure 7.7: PCB designed for sign of life testing with die mounted on 40 pin Dual Inline Package (DIP)

To determine final correctness, a scan pattern was sent in using the scan_in pin and the clock while asserting `scan_en` and `start` and deasserting `reset`. The pattern sent in was seen at the `scan_out` pin after the correct 1854 clock cycles for the scan chain. This testing was done at 50 KHz.

Having passed the sign of life test, the next step was the creation of the full test board. The problem gets far more complicated by the need for a larger number of inputs and outputs to be considered. We can not longer work with DIP packages because of the presence of large parasitics values. Additionally, there is a reduced tolerance for parasitics from bond wires and traces on the board. To do away with the need for a package the decision was made to bond straight to the traces on the board. To reduce the complexity of this process, the traces end in a staggered configuration with line ends of greater width than the traces themselves. The final view of the bonding process is shown in Fig. 7.9.

The board designed for testing the full operation of the sphere decoder takes the form shown in Fig. 7.10. The number of inputs required for testing full operation goes up to include the data input values. We continue to include the scan pins in our testing strategy

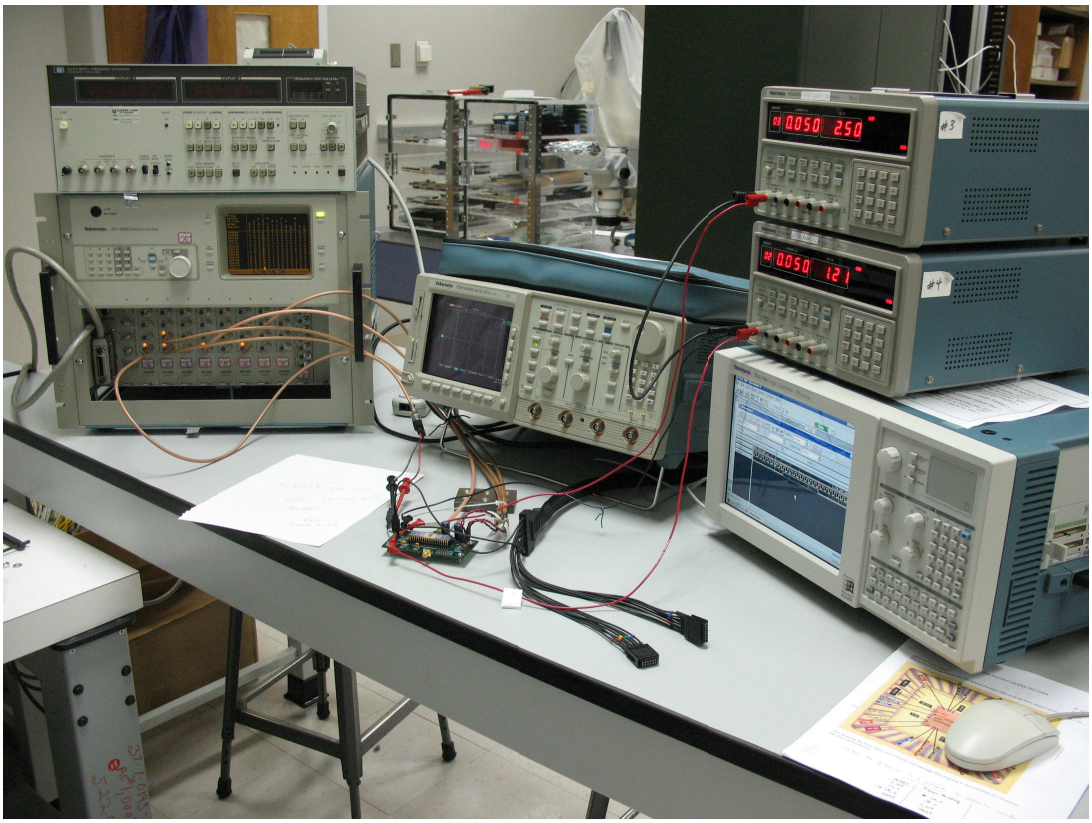


Figure 7.8: Instrumentation chain for chip testing

to enable debuggability. The scan chains can be kicked in to enable reconstruction of the datapath memory elements which should provide a good deal of understanding about the source of the error if needed. The board shows the separation of the inputs and the outputs of the chip. This is done to avoid having to cross wires over one another when testing. Things can get very difficult to handle when dealing with the usage of the instruments for testing which are both big and require rigid SMA cables for connectivity. This could lead to a lot of stress on the board.

7.2 Input generation and output checking for board tests

One of the challenges in the testing of this chip is the creation and assertion of stimulus into the Board for checking. To simplify the process, we created a perl script to

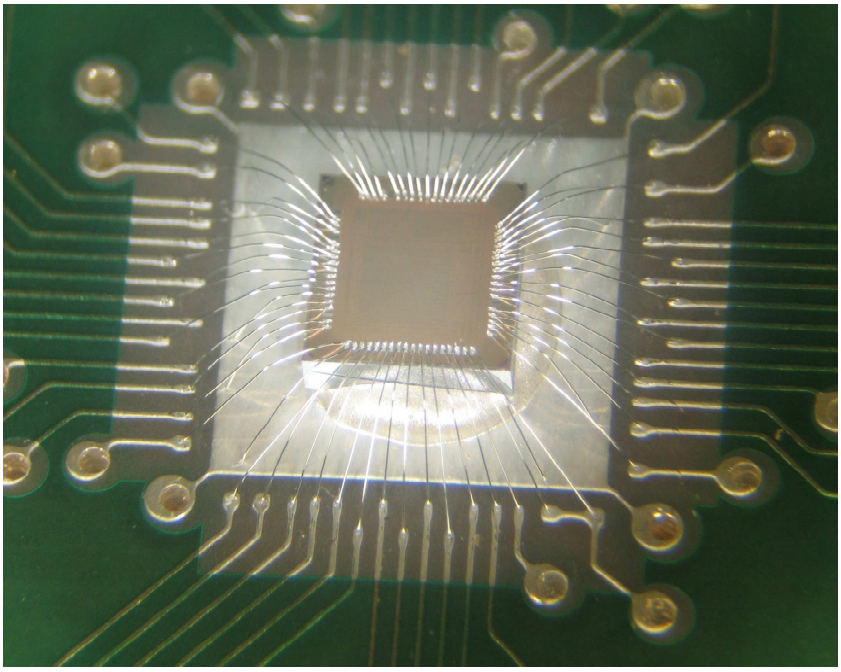


Figure 7.9: Wedge-bonding for performance measurement of Chip (Note the bonding to traces on PCB)

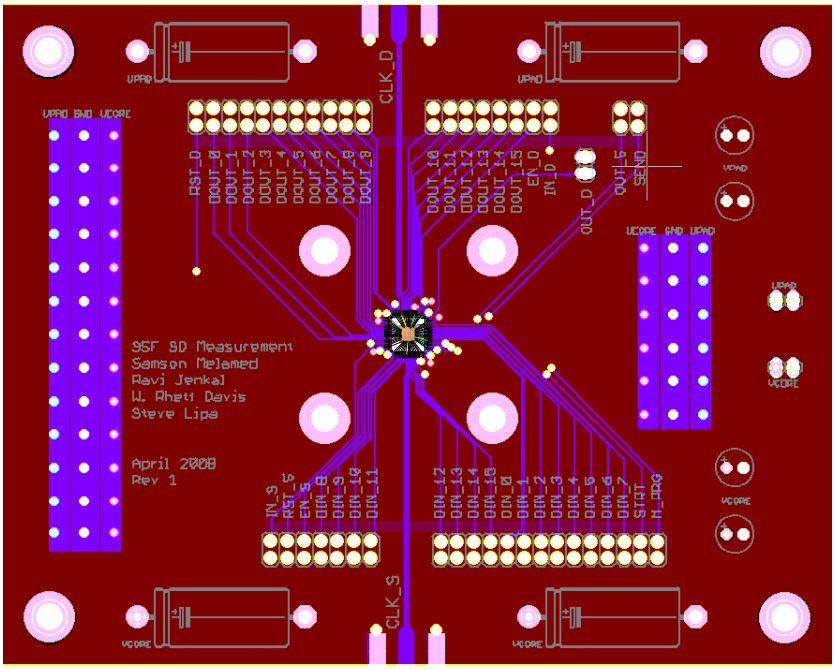


Figure 7.10: Full layout of PCB for performance measurement

parse the result of a Modelsim simulation with the requisite time values and data values for a clock edge based input for the chip. The inputs to the chip were modified to create a bit stream for each of the 22 inputs to the chip *i.e.* `reset, scan_in, scan_enable, clock, start, mem_prog` and `Din[15:0]`. The signals were then converted into a HFS 9000 Digital 36 Channel Pattern Generator compliant file. In this file, each of the 22 inputs were assigned to a particular channel of the HFS along with the creation of an oversampled clock that goes to the TLA 704 Logic Analyzer. For most cases this oversampling was done at 4. The TLA was also used to observe the output from the chip and `send_data` signal. This provided all the necessary information to determine the correctness of the decoding process in a visual manner.

Successful decoding was observed for the chip at 50 KHz initially. On increasing the clock rate we observed the correctness of the output upto 4 MHz. Beyond this point, the system decoded in a deterministic but incorrect manner. To determine the reason for this we created a perl script for the re-creation of the datapath register chain. This allowed us to look into the status of the system at any intermediate point in the decoding process. On analysis of the scan chain at intermediate points in the system beyond 4 MHz, we observed that the datapath was unable to perform accurate computation especially in the NodeCost unit. We attribute this to the absence of a sign-off level IR analysis using VoltageStorm^{TM}. Additionally, we believe that there should have been more on chip decoupling included to cater for current surges along the power and ground rails.

7.3 MIMO Analysis Flow

The MIMO Analysis flow we have developed is shown in Fig. 7.11. The entire flow is stitched together using perl. The flow begins with the ability to create SNR sensitive data of any length and corresponding to a constellation of choice using MATLAB. We usually create 20 iterations of 10000 bits each for SNRs from 6 to 30 for our analysis. This tends to capture the more important range of operation. In addition with the creation of the noisy vectors, the expected result is also created. This is necessary to determine the total number of errors in the decoding process.

These vectors form the basic testing requirements for checking the performance characteristics of the architecture of choice. At this point, a choice needs to be made. A

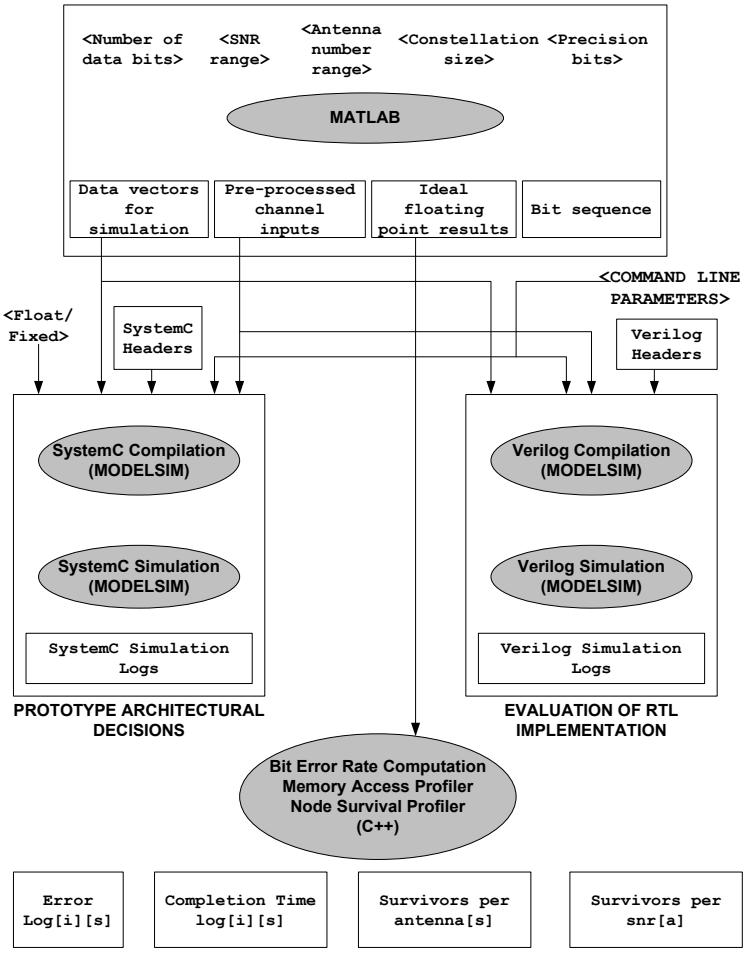


Figure 7.11: Methodology for Architectural Exploration of MIMO Designs

prototyping analysis in SystemC can be run to determine the characteristics of the architecture of choice. Alternatively, if all the necessary prototyping has been performed and the final architecture has been coded in Verilog, we can perform the requisite analysis on the coded system. This is the architectural exploration stage. Using a combination of perl for massaging the output logs and C++ for number crunching, the typical checks that are going to be done at this stage are:

1. BER checker: This C++ binary checker utilizes the result from either SystemC or RTL along with the floating point expected values to determine the total number of errors in the design. The requirements is, of course, is the need for a certain format for the display of the result of the decoding process.

2. Completion time checker: This is a simple grep to determine the total number of cycles taken to finish

3. Survivor Profiler: the display logs are parsed to create the requisite entries into the survivor profiler. The profiler bins the total number of survivors at each level as a function of the SNR. Another profiler bins the survivors for different SNRs as a function of the level. This provides valuable information regarding the efficiency of the decoding process.

4. Memory Profiler: This C++ profiler is used to determine the access patterns to memory in each of the context memories to determine if there are obvious trends that can be observed which could be exploited.

5. PED tracker: This is also a C++ function that runs on the logs to determine the progression of PED values for a given symbol vector before the final mincost is reached. This is a very useful debugging tool when used to compare the RTL implementation with the golder SystemC model. It also provides a easy means of comparison with ML estimates given that the ML final min-cost value will not be different from the SD final mincost value.

In addition to the above analysis, we can also perform bit level analysis by controlling the headers for the simulation. Any architecture can be chosen for checks by the use of command line options. The typical Command line parameters for analysis are:

1. Version: The version of the architecture to be analyzed
2. Type: This is used to determine if RTL or SystemC analysis is being performed
3. SNR range: the start and end values of the SNR range of interest
4. Iteration range : The start and end points of the iteration being performed. Each iteration will have 10000 bits to be decoded for each SNR value in the SNR option.
5. Number of Antennas: The total number of Antennas for the simulation
6. Number of contexts: This corresponds to the total number of context memories that will be used for a given simulation
7. Number of constellation points: For now, this is going to be 16 / 64 corresponding to 16-QAM or 64-QAM.
8. Operation: This corresponds to either: Simulation, Synthesis or Power

The most important choice in the above parameters is operation type. The Synthesis option creates the netlist for the design of choice using Synopsys Design Compiler. The clock used for this is 1ns given that we are going to be using a delay constrained synthesis result. The power results could either be a) using rtl2saif and running a simple modelsim simulation b) using a prototype value which uses trial route for SPEF creation or c) a final power evaluation after a full place and route is done and using a sign-off level SPEF value.

7.4 Chip Design Methodology

The chip design methodology followed for the implementation followed a typical Semi-custom flow. The synthesis flow (RTL to Netlist) is typical except for the need to incorporate Memory IP. It is important to note that the chip that was taped out did not use the Memory IP. This requires appropriate link and target library manipulation along with the search path manipulations. Additionally, a set_dont_touch attribute was applied on the higher drive clock buffers and inverters. This was done because the higher drive cells have U shaped metals in them that might cause complications in place and route.

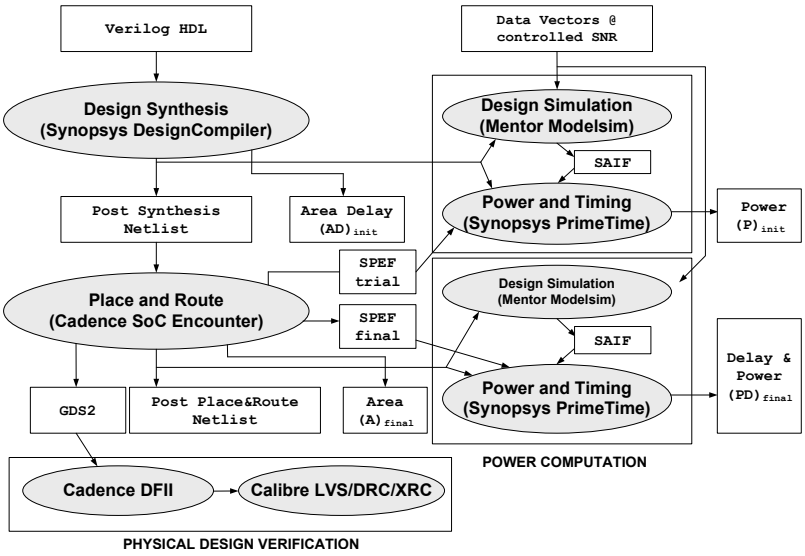


Figure 7.12: High level view of CAD flow for design and analysis

Additionally, the correct pads from the GPIO library of arm were included at the input and output side to ensure that the buffers are of the correct size at the interface. To ensure that there are no hold issues we provide a clock uncertainty of 400 ps and perform hold fixing at the fast corner of the process. The overall methodology for the movement from RTL to GDS2 is shown in Fig. 7.12

The resulting netlist from Synthesis captures the design intent using minimal logic for a given constraint. We have used a delay constraint for later designs but an area constraint for the Tapeout. The resulting netlist needs to be converted into an error free layout while ensuring that the inclusion of real capacitive and resistive values for the nets do not cause the performance to fall greatly. The toolflow for the place and route methodology used is shown in Fig. 7.13. This is done using Cadence Encounter

Of particular importance in the place and route flow is the need for Clock Tree Synthesis (CTS). A close eye must be kept on the constraints for this process. Too stringent a constraint would cause the inclusion of too many buffers which would cause an enormous increase in the power. We use a 120ps skew requirement. To ensure sign-off quality place and route parasitics, we use the Fire&Ice flow with a noise library database creation shown in Fig. 7.14. The use of this library ensures a more accurate modeling of both the nets

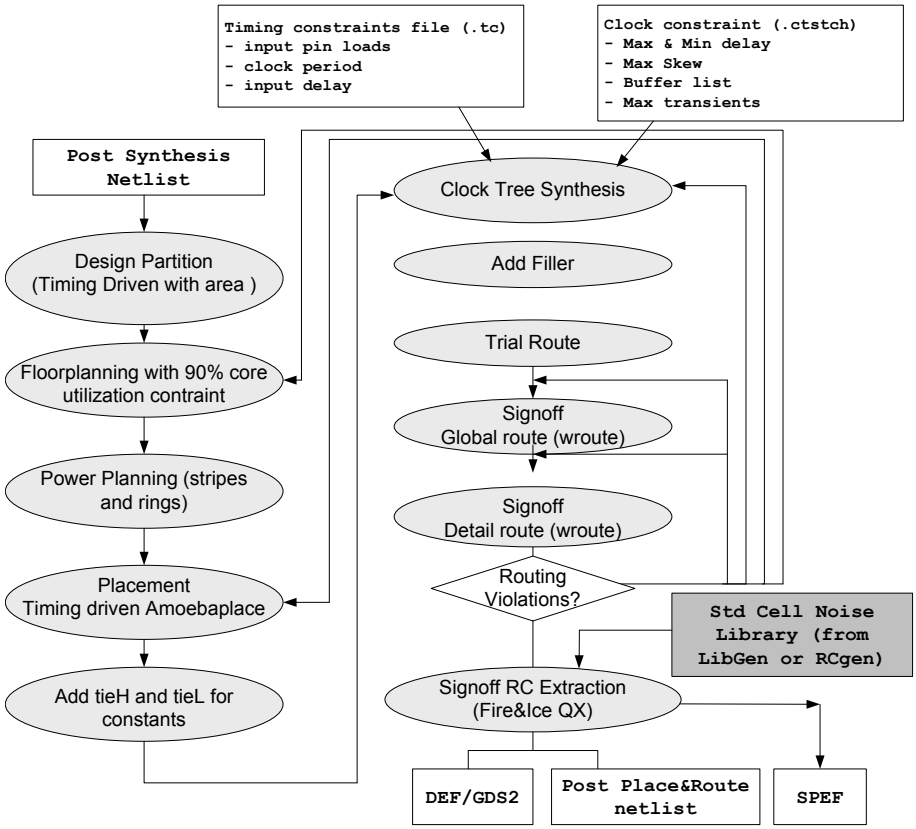


Figure 7.13: Place and route flow Sphere Decoder ASIC

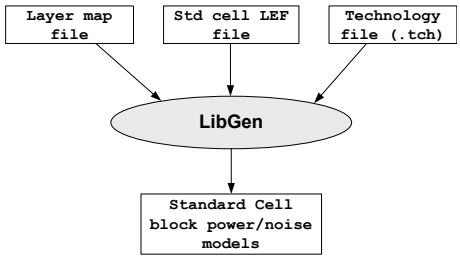


Figure 7.14: Creation of parasitic database file for Fire&Ice using `LibGen`

connecting standard cells and the internal wiring within the standard cells as well.

The final correctness of the result from Cadence Encounter place and route is established by moving the the finished GDS to DFII and running DRC and LVS using Mentor Graphics Calibre.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

The aim of this work was to provide a energy efficient approach to the design of MIMO Decoders. To this end, we analyzed these systems in the domains of both power and throughput while striving to achieve optimal decoding performance. We analyzed and implemented a set of solutions that aim to provide an energy efficient solution and showed that the optimal architecture changes with differing throughput requirements. We also provided the means of invoking an optional algorithmic sub-optimality if needed by the user. The final conclusion of the analysis was the implementation of a novel time-multiplexed architecture for Sphere Decoders. This combined deeper pipelining of a continuously processing cost resource with the creation of memories for processing the results of the computation. This architecture allowed for performing operations that would allow for an increase in throughput without having to sacrifice the optimality of the decoding process. The creation of a chip using these principles resulted in a low power implementation that consumes only 43.659 mW of power at a 6.7 ns clock period. This architecture supported a throughput of 14 Mbps for a optimal decoding performance.

We explored the algorithmic, architectural and circuit spaces using a combined SystemC and Verilog framework. This enabled the convergence to the minimum power architecture of choice based on the constraints of throughput and algorithmic optimality. The study of post layout values provided sufficient correlation to increase the trust in a high level analysis such as this. We use a fast cycle-accurate model of the system in SystemC

to determine throughput bottlenecks and converge to the minimum power configurations for different cost computational choices. We evaluated these options in terms of both algorithmic performance and throughput limitations so as to determine the architecture to be used. Based on our analysis we prove the need for the use of *l2* or its approximate to satisfy high throughput requirements at lower Signal to Noise ratios. We also analyze the effects of increasing constellation sizes and antenna values with the aim of providing solution that scale with these parameters. To deal with these constraints, we exploit the symmetry of the l^1 cost space. The resulting l^1 architecture has a small increase in power and delay with increased antennas when compared to the previous algorithmically optimal implementation. Moreover, this architecture is almost independent of the number of constellation points. The use of this architecture allowed for a further reduction in power consumption without sacrificing throughput performance at medium to high SNR values (18-26 dB). Post synthesis measurements brought the power consumption down to 23 mW at 6 ns.

In an effort to allow user control of throughput, we also provide insights into the tradeoff of throughput and decoding performance. A solution was proposed by the imposition of block-based limits during run-time by the user. These architectural modifications allow for operations close to the Gbps range at low to medium SNRs.

The time-multiplexed Sphere Decoder ASIC implementation using an 8-metal 90nm IBM is described. The creation of this chip was aimed at determining the advantages of voltage scaling using realistic measurement. We also included simple methods to increase the visibility and debuggability of the system while providing SNR sensitive data to the decoder. The chip was successfully tested for sign of life and performed successful decoding. Thus, the feasibility of this architecture was demonstrated using a combination of automatic and manual tool flows.

The combination of the above architectural improvements provide confidence in the ability of Sphere Decoders to provide flexibility and throughput support while maintaining a low power consumption. This work demonstrates the ability of these decoders to more than keep up with the needs of the future generation of wireless standards where throughputs in the high hundreds of Mbps are required at a sufficiently high quality of service. Also, we prove that the use of highly pipelined designs with multiple distributed memories allows us to achieve higher Energy Efficiency in these designs.

The key contributions of this work are:

1. Insight into the axes of tradeoffs for the implementation of Sphere Decoders
2. Methodology for design and analysis of Sphere Decoders using a combination of SystemC and RTL
3. The provision of data and analysis that shows that the optimal use of pipelining provides the lowest power implementation for a throughput constraint
4. Novel Block based algorithm for trading off sub-optimality in the interest of increasing throughput
5. Provision of insights into the constraints when facing the need for scaling the number of antennas and the constellation size.
6. New Architectures for Sphere Decoders that provides the ability to decode over wide range of throughput constraints in an energy efficient manner
7. Lowest Energy per bit performance of optimal decoding architectures
8. Functional 90nm CMOS implementation with test and debug structures to determine tradeoffs of Energy Efficiency versus SNR. We also provide details about the methodology followed for the design and testing of this system.

A comparison of our work with the State of the Art is seen in table 8.1

We compared out result with the best available implementations for both 16-QAM and 64-QAM constellation spaces. All the architectures were confirmed to be algorithmically optimal or close to optimal. The architectures were compared in terms of the clock rate where we see that our architecture achieves the highest values by virtue of deeper pipelines in the datapath. Additionally, the cycle count requirements are also lower except for [30] which is sub-optimal in decoding. The effects of the increased constellation sizes on the throughput cause the fall of the throughput in case of [52] to 14 Mbps at a low power of 80mW. Moreover, we also note that we achieve an extremely high b/s/mW for the 64-QAM case by virtue of an architecture that is insensitive to the constellation size thereby providing the ability to achieve a greater of operation for either throughput or power constraints. It is clear from the table that we have achieved the highest b/s/mW or lower nJ/b performance while not sacrificing algorithmic optimality for both the 16-QAM and 64-QAM case.

Table 8.1: Comparison of architecture with state of the art

Category	[30]	[4]	[43]	[52]	This Work
Modulation	16-QAM	QPSK	16-QAM	64-QAM	16-QAM
					64-QAM
Decoding	DFS (sub-opt)	ML (opt)	K-best SD (sub-opt)	K-best SD (sub-opt)	DFS (opt./sub opt.)
Technology	0.25u	0.18u	0.35u	0.13u	0.09u
Frequency (F)	71 MHz	123 MHz	100 MHz	270 MHz	476/711MHz (ret)
$F_{placeroute}$					362/540 MHz
Gate Count	50K + preproc	685K	91K + preproc	230K	140K + preproc ($3.5\ \mu m^2$ 90nm NAND2)
Throughput (T) @20dB @20dB	169 Mbps	28.8 Mbps	52 Mbps	14 Mbps	opt: 336/590Mbps(ret)(16-QAM) opt: 184/235Mbps(ret)(64-QAM) sub-opt:1Gbps (ret)(16-QAM)
$T_{placeroute}$					256/463Mbps(ret)(16-QAM) 141/180Mbps(ret)(64-QAM)
BER @ 20 dB					Opt.:10^{-3} Sub-Opt.10^{-2}
Power(mW) (P)	473(2.5V)	N/A	626(2.8V)	80(1.2V)	83.6(16-QAM)/95.26(64-QAM)
$P_{placeroute}$(mW)					100.32(16-QAM)/114.312(64-QAM)
$P_{100n,1.2V}$ (mW)	139.68	N/A	146.25	70	100.32(16-QAM)/114.312(64-QAM)
$T_{100n,1.2V}$ (Mbps)	422.5	N/A	218.75	18.2	463(16-QAM)/180(64-QAM)
nJ/b	2.8	N/A	12.03	2.3	0.17(16-QAM)/0.485(64-QAM)
$nJ/b_{100n,1.2V}$	0.33	N/A	0.67	3.84	0.21(16-QAM)/0.63(64-QAM)

8.2 Future Work

The primary focus of this work was to implement algorithmically optimal decoders. We have touched upon the use of block-based coding as a means of providing control over throughput while sacrificing algorithmic optimality. It would be very interesting to investigate further along this axis to determine a better criterion for assignment of a symbol vector to the pipeline resource for a given clock cycle. It would be ideal to assign symbol vector to resource as a function of the closeness to completion of a given decoding process. Sensitivity to criteria like quickness of convergence to current min-cost solution, the number of survivors still left to be analyzed etc would be ideal measures of the closeness to completion. A second important design aspect that needs to be considered is the requirement for soft-outputs. Increasing number of mobile systems have begun to adopt channel coding to achieve increased quality of service. Initial results on the implementation aspects of Sphere Decoders with soft outputs have been reported by Burg et. al [48]. It would be very interesting to determine how the energy efficiency of our architecture can be leveraged

to make more judicious decisions at the system level.

Another area that needs to be investigated in greater depth is the creation of a multi-core architecture. This architecture would involve the usage of multiple pipelined cost computation resource units with shared memories. Architectures of this type would improve the energy efficiency of decoding by increasing the throughput that cane be supported and would require a revaluation of the points of lowest power.

The use of multiple cores of the type we propose would best benefit a SoC with multiple possible application domains. There is, therefore, a case for investigating the tradeoffs involved in the evaluation and implementation of the proposed core, either one or many, in a SoC platform. This requires a combined software-hardware view with the cores acting as a sea of hardware accelerators. The evaluation would require the running of a realistic application and the evaluation of benefits of application specific instruction set improvements, software reconfiguration of the cores using control registers and the use of power management to shut-down and restart different cores based on necessity.

Another useful improvement that needs a formal investigation is the feasibility of a adaptive core which modifies its datapath and control protocols based on the environment. This sort of run-time change of flavor would help in minimizing the overall amount of unused computation. This kind of adaptability reduces the degree of control required from the software side and reduced the overhead of core maintenance. A combined algorithmic flavor modification and hardware reconfiguration might lead to lowest feasible power consumption for a given micro-architectural configuration.

Bibliography

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

[2] M. B. Steer, "Beyond 3G", *EEE Microwave Magazine*, February 2007.

[3] Emre Telatar, I. "Capacity of Multi-antenna Gaussian Channels", *Technical Memorandum*, Bell Laboratories, Lucent Technologies, October 1995.

[4] D. Garrett, G.K. Woodward, L. Davis, C. Nicol, "A 28.8 Mb/s 4x4 MIMO 3G CDMA receiver for frequency selective channels", *IEEE Journal of Solid-State Circuits*. vol. 40, Issue 1, page(s) 320 - 330, Jan. 2005.

[5] Gerard J. Foschini " Layered space-time architecture for wireless communication in a fading environment when using multiple antennas", *Bell Labs Technical Journal*, Vol. 1, Issue 2, pp. 41-59, September 1996.

[6] D. Gesbert, M. Shafi, Da-shan Shiu, P.J. Smith and A. Naguib "From Theory to Practice: An Overview of MIMO Space-Time Coded Wireless Systems", *IEEE Journal on Selected Areas in Communications*, vol 1, issue 3, pp 281 - 302, April 2003.

[7] A. Burr, "MIMO Wireless Systems: Principles, Potential, Problems and Prospects", COST273 Workshop in Helsinki, May 2002.

[8] Lizhong Zheng and D.N.C. Tse, "Diversity and multiplexing: a fundamental tradeoff in multiple-antenna channels", *IEEE Transactions on Information Theory*, Vol. 49, Issue. 5, page(s) 1073- 1096, May 2003.

[9] A. Adjoudani, E. Beck, A. Burg, G. Djuknic, T. Gvoth, D. Haessig, S. Manji, M. Milbrodt, M. Rupp, D. Samardzija, A. Siegel, T. Sizer II, C. Tran, S. Walker, S. A.

Wilkus and P. Wolniansky, "Prototype Experience for MIMO BLAST over Third Generation Wireless System", *IEEE Journal on Selected Areas in Communication, April 2003*, Vol. 21, Issue. 3, Pages(s) 440-451.

[10] Patrick Murphy, Feifei Lou, Ashutosh Sabharwal and Patrick Frantz, "An FPGA Based Rapid Prototyping Platform for MIMO Systems", *Asilomar Conference on Signals, Systems, and Computers, 2003*. Volume 1, 9-12 Nov. 2003 Page(s):900 - 904 Vol.1

[11] W. Xiang, D. Waters, T. Pratt, and J. R. Barry "Implementation and Experimental Results of a Three-Transmitter Three-Receiver OFDM/BLAST Testbed", *IEEE Wireless Communications Magazine*, Vol. 42, No. 12, Pages(s) 88-95, December 2004.

[12] A. Burg, D. Garrett, "VLSI Implementation of MIMO Detection", Chapter in *Space-Time Wireless Systems: From Array Processing to MIMO Communications*. Cambridge University Press, 2005.

[13] Zhan Guo and Peter Nilsson, "A VLSI Implementation of MIMO Detection for Future Wireless Communications", *14th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'03)*, September 7-10, 2003.

[14] Zhan Guo and Peter Nilsson, "A Low-Complexity VLSI Architecture for Square Root MIMO Detection", *IASTED International Conference Circuits, Signals, and Systems (CSS'03)*, May 19-21, 2003.

[15] A. Burg, N. Felber and W. Fichtner, "A 50 Mbps 4x4 Maximum Likelihood Decoder for Multiple-Input Multiple-Output Systems with QPSK Modulation", *IEEE International Conference on Electronics, Circuits and Systems, 2003*. Volume 1, 14-17 Dec. 2003 Page(s):332 - 335 Vol.1

[16] A. Burg, E. Beck, M. Rupp, D. Perels, N. Felber, and W. Fichtner, "FPGA implementation of a MIMO receiver front-end for UMTS", *International Zurich Seminar on Broadband Communication, IZS'02*, 19-21 Feb. 2002 Page(s):8-1 - 8-6.

[17] "Practical Aspects of Multiple Architectures for HSDPA" 3GPP Tech. Rep. TSGR1 16 (00)1219.

[18] H. Artes, D. Seethaler, and F. Hlawatsch, "Efficient Detection Algorithms for MIMO Channels: A Geometrical Approach to Approximate ML Detection", *IEEE Transactions on Signal Processing, Nov 2003*, Volume 51, Issue 11, Nov 2003 Page(s):2808 - 2820.

[19] Ernst Aschbacher, Sebastian Caban, Christian Mehlfuhrer, Georg Maier, , and Markus Rupp, "Design of a Flexible and Scalable 4x4 MIMO Testbed" *11th IEEE Digital Signal Processing Workshop*, 1-4 Aug. 2004, Taos Ski Valley, NM, USA.

[20] Inaki Berenguer and Xiaodong Wang, "Coding and Signal Processing for MIMO Communications- A Primer", *Journal of Computer Science and Technology*, Volume 18, Issue 5, Pages(s) 689-702, Nov. 2003

[21] D. C. Garrett, L. M. Davis, and G. K. Woodward. "19.2 mbit/s 4 x 4 BLAST/MIMO detector with soft ML outputs", *ELECTRONICS LETTERS, 2003*.

[22] Zhan Guo and Peter Nilsson, "A V-BLAST Detector Based on Square Root Algorithm", *14th IEEE International Symposium on Personal, Indoor and Mobile Radio Communication, 2003*.

[23] Robert W. Heath Jr, S. Sandh, and A.J. Paulraj, "Space-Time Block Codes versus Space-Time Trellis Codes", *IEEE International Conference on Communications, June 2001*, Volume 4, 11-14 June 2001 Page(s):1132 - 1136 vol.4.

[24] M. Rupp, M. Guillaud and S. Das, "On MIMO Decoding Algorithms for UMTS" *35th Asilomar Conference on Signals, Systems and Computers, 2001*, Volume 2, 4-7 Nov. 2001 Page(s):975 - 979 vol.2.

[25] P.W. Wolniansky, G.J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: an Architecture for Realizing Very High Data Rates over the Rich-Scattering Wireless Channel", *IEEE 1998 URSI International Symposium on Signals, Systems, and Electronics*, 29 Sept.-2 Oct. 1998 Page(s):295 - 300.

[26] P.W. Wolniansky, G.J. Foschini, G. D. Golden, and R. A. Valenzuela, "Detection Algorithm and Initial Laboratory Results using the V-BLAST Space-Time Communication Architecture", *Electronics Letters*, Vol. 35, Sept 1999.

[27] Karen Su. "Space-Time Coding: From Fundamentals to the Future".

[28] R. van Nee, A. van Zelst and G. Awater, "Maximum Likelihood Decoding in a Space Division Multiplexing System", *IEEE Vehicular Technology Conference (VTC), 2000*.

[29] D. Garrett, L. Davis, S. ten Brink, B. Hochwald, G. Knagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding", *IEEE Journal of Solid-State Circuits*. Volume 39, Issue 9, Sept. 2004 Page(s):1544 - 1552.

[30] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm", *IEEE Journal of Solid-State Circuits*, Volume 40, Issue 7, July 2005 Page(s):1566 - 1577.

[31] Andreas Burg, Moritz Borgmann, Markus Wenk, Christoph Studer, and Helmut Bölcskei., "Advanced receiver algorithms for MIMO wireless communications", *Design, Automation, and Test in Europe (DATE)*, Volume 1, 6-10 March 2006.

[32] A. Burg, M. Borgmann, C. Simon, M. Wenk, M. Zellweger, W. Fichtner, "Performance tradeoffs in the VLSI implementation of the sphere decoding algorithm", *Fifth IEE International Conference on 3G Mobile Communication Technologies, 2004. 3G 2004.*, 2004 Page(s):93 - 97.

[33] G. Knagge, G. Woodward, S. Weller, B. Ninness, "An optimised parallel tree search for multiuser detection with VLSI implementation strategy", *IEEE Global Telecommunications Conference, 2004. GLOBECOM '04.* , Volume 4, 29 Nov.-3 Dec. 2004 Page(s):2440 - 2444 Vol.4.

[34] B. Widdup, G. Woodward, G. Knagge, " A highly-parallel VLSI architecture for a list sphere detector", *IEEE International Conference on Communications, 2004.* , Volume 5, 20-24 June 2004 Page(s):2720 - 2725 Vol.5.

[35] Kwan-wai Wong, Chi-ying Tsui, Cheng, R.S.-K., Wai-ho Mow " A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels", *IEEE International Symposium on Circuits and Systems* vol.3, no.pp. III-273- III-276 vol.3, 2002

[36] Z. Guo, P. Nilsson, " VLSI implementation issues of lattice decoders for MIMO systems", *IEEE International Symposium on Circuits and Systems* vol.4, no.pp. IV- 477-80 Vol.4, 23-26 May 2004

[37] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W Fichtner, " K-best MIMO detection VLSI architectures achieving up to 424 Mbps throughput", *IEEE International Symposium on Circuits and Systems* May 22, 2006.

[38] Wen-Tsong Shiue, S. Tadas, C. Chakrabarti, "Low power multi-module, multi-port memory design for embedded systems", *IEEE Workshop on Signal Processing Systems, 2000. SiPS 2000.* , 11-13 Oct. 2000 Page(s):529 - 538.

[39] Wen-Tsong Shiue, "Energy efficient memory assignment", *The 2001 IEEE International Symposium on Circuits and Systems*, May 2001 Page(s):415 - 418 vol. 5.

[40] D.G. Chinnery and K. Keutzer, "Closing the Power Gap between ASIC and Custom", *42nd Design Automation Conference, June 2005*, 13-17 June 2005 Page(s):275 - 280.

[41] The OSU Standard Cell Library, `http://www.ece.iit.edu/~jgrad/osucells/`.

[42] International Technology Roadmap for Semiconductors- 2005 Edition, `http://www.itrs.net/Common/2005ITRS/Design2005.pdf`.

[43] Zhan Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection", *IEEE Journal on Selected Areas in Communications*, vol.24, no.3, pp. 491- 503, March 2006.

[44] Zhan Guo and P. Nilsson, "A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems", *IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication, 2004.*, vol.1, no.pp. 65- 68 Vol.1, 31 May-2 June 2004.

[45] Norm Jouppi et al, CACTI 4.0, HP-Compaq Western Research Lab, `http://www.hpl.hp.com/personal/Norman_Jouppi/cacti4.html`.

[46] V. Zyuban and P. Kogge, "The energy complexity of register files", *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 305-310, 10-12 Aug 1998.

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

GLSVLSI '07: Proceedings of the 17th great lakes symposium on Great lakes symposium on VLSI, March 11-13, 2007.

[48] C. Studer, A. Burg, and H. Bölcskei, "Soft-output sphere decoding: Algorithms and VLSI implementation", *IEEE Journal on Selected Areas in Communications*, Vol. 26, No. 2, pp. 290-300, Feb. 2008.

[49] C. Studer, M. Wenk, A. Burg, and H. Bölcskei, " Soft-Output Sphere Decoding: Performance and Implementation Aspects", *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, Oct.-Nov. 2006.

[50] M. Wenk, M. Zellweger, A. Burg, N. Felber, W. Fichtner " K-best MIMO detection VLSI architectures achieving up to 424 Mbps", *Proc. of IEEE Int. Symp. on Circuits and Systems (ISCAS)*, 2006.

[51] M. Wenk, A. Burg, M. Zellweger, C Studer, W. Fichtner " VLSI Implementation of the List Sphere Algorithm", *24th Norchip Conference, 2006.*, Nov. 2006.

[52] S. Chen, T. Zhang, and Y. Xin, "Relaxed K-best MIMO Signal Detector Design and VLSI Implementation", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, March 2007.

[53] C.-H. Yang and D. Markovic, "A Flexible VLSI Architecture for Extracting Diversity and Spatial Multiplexing Gains in MIMO Channels," *Proc. IEEE International Conference on Communications*, May 2008.

[54] A.P. Chandrakasan and R.W. Brodersen, "Minimizing power consumption in digital CMOS circuits", *Proceedings of the IEEE*, April 1995.

[55] Dejan Markovic, Vladimir Stojanovic, Borivoje Nikolic, Mark Horowitz, Robert W. Brodersen, "Methods for true energy-performance optimization", *IEEE Journal of Solid-State Circuits*, August 2004.

[56] D. Markovic, B. Nikolic, and R.W. Brodersen, "Power and Area Minimization for Multidimensional Signal Processing", *IEEE Journal of Solid-State Circuits*, April 2007.

[57] Benton H. Calhoun, Yu Cao, Xin Li, Ken Mai, Lawrence T. Pileggi, Rob A. Rutenbar, and Kenneth L. Shepard, "Digital Circuit Design Challenges and Opportunities in

the Era of Nanoscale CMOS" *Proceedings of the IEEE (Special Issue on Integrated Electronics: Beyond Moore's Law)*, February 2008.

[58] SystemC^{TM}, `http://www.systemc.org`

[59] T. Grötker, S. Liao, G. Martin, and S. Swan, "System Design with SystemC", Kluwer, Boston, 2002.

[60] Chris Spear, "SystemVerilog for Verification: A Guide to Learning the Testbench Language Features", Springer, NY, 2006.

[61] CoCentric^{TM}from Synopsys^{TM}, `http://www.synopsys.com`

[62] Ramsey Hourani, Ravi Jenkal , W. Rhett Davis, Winser Alexander "Automated Design Space Exploration for DSP Applications" *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology* May 2008.

[63] Ramsey Hourani, Ravi Jenkal , W. Rhett Davis, Winser Alexander, "Automated Architectural Exploration for Signal Processing Algorithms", *In IEEE Workshop on Signal Processing Systems (SiPS'06)*, Banff, AB, Canada, October 2- 4, 2006.

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

[65] "International Technology Roadmap for Semi-conductors, System Drivers Chapter", `http://www.itrs.net/Links/2007ITRS/2007_Chapters/2007_SystemDrivers.pdf`

[66] Alan Edelman, "Eigenvalues and condition numbers of random matrices", SIAM Journal on Matrix Analysis and Applications Vol. 9, 1988, pp. 543-560