Baseband Architecture Design for Future Wireless Base-Station Receivers

Sridhar Rajagopal

Thesis: Master of Science
Electrical and Computer Engineering
Rice University, Houston, Texas (May 2000)

RICE UNIVERSITY

Baseband Architecture Design for Future Wireless Base-Station Receivers

by

Sridhar Rajagopal

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Master of Science

Houston, Texas

May, 2000

ABSTRACT

Baseband Architecture Design for Future Wireless Base-Station Receivers

by

Sridhar Rajagopal

This thesis demonstrates the use of designing efficient algorithms and architectures to meet the real-time requirements of future wireless base-station receivers. Next generation receivers will require orders-of-magnitude performance improvements in order to provide support for features such as Multimedia, Quality-Of-Service and extremely high data rates. The sophisticated, compute-intensive algorithms proposed to integrate these features make their real-time implementation difficult on current Digital Signal Processor (DSP)-based receivers. A real-time implementation can be achieved by (1) making the algorithms computationally efficient, without significant loss in error rate performance, (2) task partitioning and (3) designing hardware to exploit available pipelining, parallelism and bit-level computations.

Multiuser Channel Estimation and Detection, two of the most compute-intensive baseband tasks in the receiver, are implemented on DSPs for performance evaluation. A reduced complexity iterative channel estimation scheme for slow fading channels is proposed for a fixed point, area-time efficient and real-time VLSI architecture. The multiuser detection algorithm is modified for a simple, pipelined structure. A General Purpose Processor (GPP) or DSP based architecture with reconfigurable support suited for different wireless communication standards is proposed and extensions are developed to accelerate the implementation of wireless communication algorithms.

Acknowledgments

I would like to acknowledge the support and guidance from my advisor, Dr. Joseph Cavallaro, whose suggestions and directions have a major influence on all aspects of my thesis. I would like to thank Dr. Behnaam Aazhang for his suggestions and guidance on the algorithms studied and implemented in this thesis and Dr. Dan Sorensen for his help on linear algebraic techniques for reducing the complexity of the algorithms. I would also like to thank Dr. Sarita Adve and Partha Ranganathan for several discussions and suggestions on implementation comparisons with general purpose processors. I'm grateful to Chaitali Sengupta and Suman Das for helping me in the understanding of Channel Estimation and Multiuser Detection algorithms and to Srikrishna Bhashyam for his help with the iterative Channel Estimation scheme. I'm also grateful to my office-mates, Gang Xu and Vishwas Sundaramurthy, for helping me with the software tools. I'm grateful to researchers from Nokia and Texas Instruments for their valuable comments and feedback during presentations. I would like to thank Dr. Jan Hewitt for her help in my thesis-writing and finally to all my friends, especially in ECE, for making the past two years at Rice, a wonderful and memorable experience.

This work was supported by Nokia, Texas Instruments, the Texas Advanced Technology Program under grant 1999-003604-080, and by NSF under grants NCR-9506681 and ANI-9979465.

Contents

Abstract ii

Acknowledgments iii

List of Illustrations vii

List of Tables ix

1 Introduction 1

1.1 Requirements for enhanced Base-Station Receivers 1

1.2 Thesis Contributions 2

1.3 Thesis Overview 4

2 Background 6

2.1 Wideband CDMA Communication Systems 6

2.2 The Base-Station Physical Layer 8

2.3 Multiuser Channel Estimation 10

2.3.1 Parameter Extraction Scheme 12

2.4 Multiuser Detection 13

2.4.1 Matched Filter Detector 14

2.4.2 Multistage Detector 14

3 DSP Implementation and Task Partitioning 17

3.1 Implementation Methodology 17

3.2 DSP Implementation and Comparisons 18

3.3 Task Decomposition 19

3.4 Simulations and Analysis 21

3.5 Meeting Real Time Constraints 24

4 Reduced-Complexity Channel Estimation and Detection 25

4.1 Iterative Channel Estimation 25

4.2 Estimation based on Precomputed Preamble 28

4.3 Pipelined Detection 29

4.4 Computational Savings 31

5 VLSI Architecture for Channel Estimation 33

5.1 Task Decomposition 33

5.2 Area-Constrained Architecture 36

5.3 Time-Constrained Architecture 37

5.4 Area-Time Efficient Architecture 40

5.5 Comparisons with DSPs 40

6 Architecture and Extensions for DSPs and GPPs 46

6.1 Features of Wireless Communication Algorithms 46

6.2 Architecture Design with Reconfigurable Support 49

6.2.1 Need for Reconfigurable Support 49

6.2.2 Architecture Design 49

6.3 Instruction Set Extensions for Wireless 52

6.3.1 Reference Architecture 52

6.3.2 Integer-Bit Multiplications 53

6.3.3 Bit-Bit Multiplications 55

7 Conclusions and Future Work 58

7.1 Conclusions 58

7.2 Future Work 59

7.2.1 Online Arithmetic 59

7.2.2 Multiprocessing with DSPs and FPGAs 59

Illustrations

1.1 Data Rates achieved by a typical DSP for a Base-Station Receiver . . 2

2.1 Frame structure for the Uplink Data and Control Channels 8

2.2 Uplink Communication between the Mobile and the Base-Station . . 9

2.3 Physical Layer of the Base-Station Receiver 10

2.4 Channel Model 11

2.5 Multiple Access Interference from Past and Future bits of users . . . 15

3.1 Data Rate Comparisons for a Matched Filter and Multiuser Detector on a single DSP 18

3.2 The Task Partition Graph for the Joint Estimation and Detection Algorithm 19

3.3 Further Pipelining & Parallelism in the Multistage Detection 22

3.4 Data Rates for various levels of Pipelining and Parallelism for A ($\mathbf{A}^H\mathbf{r}$) and B (block IV) 23

4.1 Error Rate Performance in Additive White Gaussian Noise Channel . 27

4.2 Error Rate Performance in a Fading Channel 28

4.3 Block Based Detection 30

4.4 Pipelined Detection 31

5.1 Task Decomposition of Multiuser Channel Estimation Algorithm . . . 34

5.2 Area-Constrained VLSI Architecture 35
5.3 Time-Constrained VLSI Architecture Block Diagram 37
5.4 Elements in the Auto-correlation Matrix Block 38
5.5 Elements in the Cross-correlation Matrix Block 39
5.6 Area-Time Efficient VLSI Architecture 41

6.1 Architecture Design with Reconfigurable Support 50
6.2 8 Integer - Bit Multiplications in Parallel 53
6.3 8 to 64 bit conversions 55
6.4 64 1-bit Multiplications in Parallel 56
6.5 8 8-bit Increments/Decrements in Parallel 57

Tables

2.1 Evolution of Wireless Communication Systems 7

2.2 Proposed Data Rates for Next Generation Communication Systems . 8

3.1 Cycle count and Complexity for different blocks 20

3.2 Cycle count for different optimization levels of blocks A ($\mathbf{A}^H\mathbf{r}$) and B (block IV) 21

4.1 Comparisons of Computational Savings 32

5.1 Hardware Requirements for an Area-Constrained Architecture 43

5.2 Hardware Requirements for a Time-Constrained Architecture 44

5.3 Hardware Requirements for Area-Time Efficient Architecture 45

5.4 Comparisons between Different Channel Estimation Architectures . . 45

6.1 Latencies for various operations 53

Chapter 1

Introduction

1.1 Requirements for enhanced Base-Station Receivers

Next generation cellular wireless communication receivers will need to support features such as extremely high data rates (up to 2 Mbps), Quality-Of-Service (QoS) and multimedia [1, 2]. The algorithms proposed by researchers to provide these features are extremely sophisticated and have high computational complexity. The current base-station receivers, which are typically built using DSP hardware, need orders-of-magnitude performance improvement to meet the real-time requirements of the algorithms proposed for next generation communication systems. Therefore, there is a need for better algorithms and architectures to meet the real-time requirements of the base-station receiver.

Figure 1.1 shows the data rate achieved by a Digital Signal Processor(DSP) implementation of a multiuser detector with varying number of users for a system targeted to support 128 Kbps per user. It can be seen that the performance achieved by just the detector implementation falls below the real-time requirements by a factor of 10. Hence, the proposed algorithms for the base-station receiver need to be modified for computationally efficient solutions, without significant loss in their Bit Error Rate (BER) performance. These modifications are achieved by applying linear algebra techniques such as iterative, sub-optimal schemes or exploiting the structure of the algorithms. These algorithms, typically working on matrix data sets, have significant

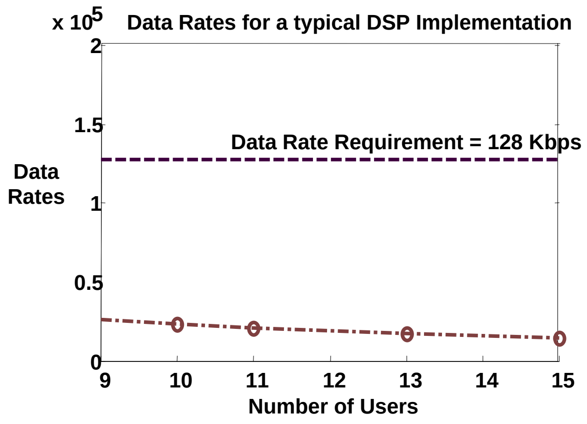

Figure 1.1 : Data Rates achieved by a typical DSP for a Base-Station Receiver

levels of parallelism. This inherent parallelism and the bit-level nature of the computations can be exploited to achieve real-time performance by an efficient architecture design. Modifying the algorithm for a computationally efficient solution and exploiting its parallelism using a suitable architecture design are needed to achieve real-time performance in future base-station receivers.

Channel Estimation and Detection, two major compute-intensive tasks in the physical layer (baseband) of the base-station receiver, are modified to make their implementation suitable for future wireless base-stations.

1.2 Thesis Contributions

The main contributions of this thesis include the design of computationally efficient algorithms for channel estimation and detection along with their hardware implementation. An in-depth analysis of the algorithms for estimation and detection is carried out for performance evaluation and seeking means for performance improvements. There has been previous work in computationally efficient algorithms for detection [3], but not for channel estimation. A computationally efficient algorithm for channel estimation is proposed and the detection algorithm is enhanced for a simpler hardware implementation.

Previous channel estimation schemes [4] required the use of matrix inversions to compute the estimate, which is compute-intensive and required the use of floating point arithmetic. An iterative, computationally-efficient channel estimation scheme [5] is developed, which has a simple-fixed point implementation and can be easily extended to provide tracking over fading channels. Another channel estimation scheme, which relies on pre-computation of the preamble, but has a faster implementation, is also developed.

A design methodology using task partitioning [6] is developed to make use of the inherent parallelism and bit level computations in the algorithms. The task partitioning methodology is applied to estimation and detection to achieve a real-time implementation. Further, a custom hardware solution is proposed for channel estimation as a study to see the effectiveness of a VLSI architecture [7] for the base-station receiver. A VLSI architecture is developed to use pipelining and parallelism effectively and take advantage of the bit level computations. The design space is explored for area-time tradeoffs and an area-time efficient solution which meets the real-time requirements is implemented.

With general-purpose processors and DSPs becoming more adaptive to multimedia applications, insights from the analysis of wireless communication algorithms are applied to enhance current general purpose processors (GPPs) and DSPs for wireless communications. An architecture with reconfigurable support and with wireless communication extensions is developed for accelerating wireless communication algorithms.

1.3 Thesis Overview

In this introductory chapter, the need for better wireless base-station receivers is stressed and the techniques to achieve real-time using computationally-effective algorithms and better architectures is shown. The key contributions of the thesis are also outlined.

The next chapter gives a background on the next generation communication systems. The terms 'Multiuser Channel Estimation' and 'Multiuser Detection' are explained and the previous work in this area described.

Chapter 3 discusses an implementation of the proposed algorithms on one of the recent DSP processors. A design methodology to achieve real-time performance by task decomposition is shown. The estimation and detection algorithms are decomposed to achieve real-time performance.

The modifications of the previously existing algorithms for multiuser channel estimation and the enhancements for a simpler pipelined detection scheme is detailed in Chapter 4. Two channel estimation modifications are developed. The first scheme uses an iterative method while the second makes use of prior knowledge of the preamble.

A VLSI architecture of channel estimation as a case study for seeking the effectiveness of a custom implementation for the base-station receiver is evaluated in Chapter 5. Area-time tradeoffs are made by exploring the design space for an area-time efficient real-time solution.

Chapter 6 shows the enhancements that could be done for DSPs and GPPs to accelerate their performance for wireless applications. A processor architecture with reconfigurable logic support and wireless communication extensions is developed.

Finally, the conclusions are presented and future directions such as the use of online arithmetic and multiprocessing on DSPs and Field Programmable Gate Arrays(FPGAs) are stated.

Chapter 2

Background

This chapter provides a background to the next generation communication standards and describes the algorithms for Multiuser Channel Estimation and Detection, two of the most compute-intensive baseband algorithms in the base-station receiver.

2.1 Wideband CDMA Communication Systems

Wideband Direct-Sequence Code Division Multiple Access (W-CDMA) is the emerging protocol [8] for wireless communications in the next generation (3G) communication systems. W-CDMA has been designed to provide support for features such as multimedia, high data rates (up to 2 Mbps), multi-rate services and Quality-Of-Service(QoS) in the existing wireless framework.

The first generation cellular communication systems using analog transmission for speech services were introduced in early 1980s. Several standards were developed in different countries such as AMPS (Advanced Mobile Phone Systems) in the US and NTT (Nippon Telephone and Telegraph) in Japan. Second generation systems using digital transmission were developed in the late 1980s. They offered higher spectrum efficiency, better data services and more advanced roaming than the first generation systems. GSM (Global System for Mobile Communications), which is a TDMA-based (Time Division Multiple Access) system in Europe, IS-136 (Dual Mode-AMPS) and IS-95 (CDMA) systems in the US are examples of second generation systems. The services offered by these systems cover speech and low data rates (9.6 Kbps). Though different multiple access schemes have been used in different standards of the first and second generation systems, the third generation systems have accepted W-CDMA as the form of multiple access communication.

Table 2.1 shows the evolution of wireless communication systems over the past two decades. T. Ojanpera and R. Prasad [9, 10] give a detailed description of the evolution of the standards and its specifications. As per 3GPP standards [11], the data transmission for W-CDMA is done in frames of $10ms$. The frame structure for transmission is as shown in Figure 2.1. The data is multiplexed with the control channel in a dual-channel QPSK (Quadrature Phase Shift Keying) form. The dedicated physical data channel (DPDCH) is QPSK-multiplexed with the dedicated physical control channel (DPCCH). The actual data transmission is done on the DPDCH while the DPCCH is used for sending control information. The control information consists of the Pilot bits, which are used for channel synchronization and estimation, the transmit power control (TPC) commands and an optional Transport Format Indicator (TFI), which contains the data rate information. The multi-rate is achieved by varying the spreading factor of the bits. The data rates achieved for a typical chip rate of 4.096 Mcps and varying spreading factors between 4 and 256 are as shown in Table 2.2.

Year	**Generation**	**Multiple Access**	**Services**
1980	First	FDMA, TDMA	Speech
1990	Second	TDMA, CDMA	Speech, Low rate data
2000	Third	W-CDMA	Speech, High rate data, Multimedia

Table 2.1 : Evolution of Wireless Communication Systems

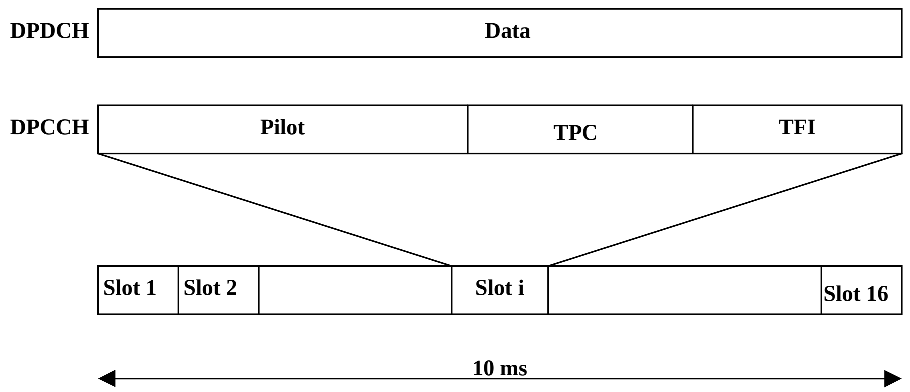

Figure 2.1 : Frame structure for the Uplink Data and Control Channels

2.2 The Base-Station Physical Layer

Uplink communication occurs when mobile users are sending data to the base-station. Figure 2.2 shows a scenario in which different users are communicating with the base-station. As the data from each mobile user is transmitted through the wireless channel, it experiences changes such as delays, multipath reflections, attenuation, interference from other users, fading due to the mobile velocity, and noise before it reaches the base-station receiver.

At the receiver, steps must be taken to correct these changes and to recover the

Spreading Factor (N)	**Bits Per Frame**	**Data Rates (Bits per second)**
4	10240	1 Mbps
32	1280	128 Kbps
256	160	16 Kbps

Table 2.2 : Proposed Data Rates for Next Generation Communication Systems

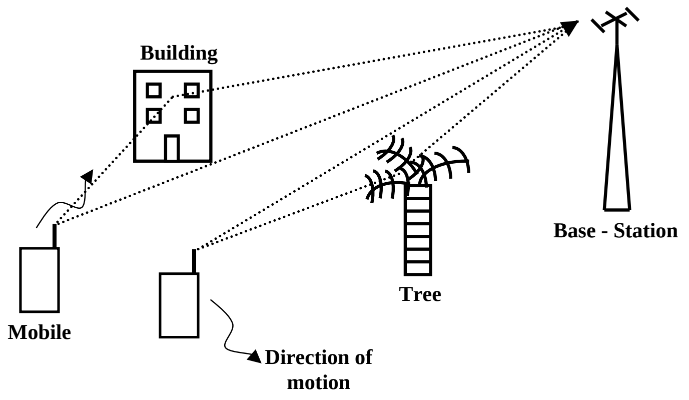

Figure 2.2 : Uplink Communication between the Mobile and the Base-Station

transmitted data accurately. Thus, the main baseband processing blocks at the receiver are the blocks for multiuser channel estimation, multiuser detection and decoding. For accurate detection, the bits, which are asynchronous due to the multipath delays, need to be synchronized and their amplitude variations due to fading and attenuation, need to be estimated. Multiuser Channel Estimation involves estimating and tracking the delays and amplitudes of the mobile users over the different multipaths. Multiuser detection involves canceling the interference from the other users to get better accuracy and bit error rate performance than single user detection, which treats the interference from other users as Additive White Gaussian Noise(AGWN) [12]. The coding applied to the bits at the mobile transmitter for better performance is then decoded with the help of the decoding block.

The block diagram of the physical layer of the uplink communication at the base-station is as shown in Figure 2.3. As per 3GPP standards [11], the QPSK signal received at the input of the base-station receiver is split into the Data and Pilot signals. The channel estimation block computes the channel amplitudes and delays and helps the multiuser detector in accurate detection of the received data. The

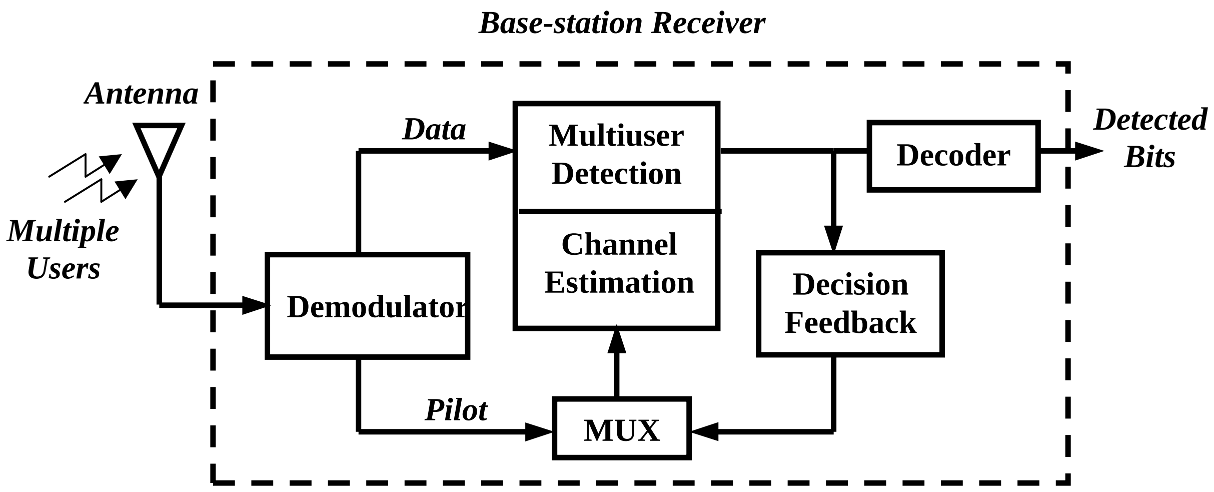

Figure 2.3 : Physical Layer of the Base-Station Receiver

multiuser detector removes the interference from other users and passes the detected bits to the decoder for further processing. As noted from Figure 2.1, the Pilot signal is not available for the entire duration of the slot. During this time, decision feedback [13] is used to update the channel estimation block. Decision feedback uses the detected bits for updating the estimates by acting as an extended pilot during the absence of the Pilot bits. A discussion of the existing multiuser channel estimation and detection schemes is given below.

2.3 Multiuser Channel Estimation

Multiuser Channel estimation refers to the joint estimation of amplitudes and delays for all active users. The discussion here refers to the maximum likelihood based channel estimation scheme [4]. A short Gold code of spreading factor 31 and a BPSK modulation system is assumed for convenience. In this model, it is assumed that the maximum delay spread between the multipaths is less than half a symbol period.

The analysis mentioned can be easily extended to Long Codes, as in the W-CDMA standards [11,14].

The channel model is as shown in Figure 2.4.

$$\mathbf{r}_i = \mathbf{A}_i\mathbf{b}_i + \eta_i \tag{2.1}$$

where $\mathbf{r}_i \in \mathbb{C}^N$ are the received bits of all K asynchronous users, spread with a spreading factor N, $\mathbf{b}_i \in \mathbb{R}^{2K} = [b_{1,i-1}, b_{1,i}, \dots, b_{K,i-1}, b_{K,i}]^\top$ are the bits of K users to be detected, $\mathbf{A}_i \in \mathbb{C}^{2K\times N}$ is the estimate of the channel containing information about the spreading codes, attenuation and delays from the various paths, η_i is the noise, which is assumed to be Gaussian (AGWN) and i is the time index. The aim of the estimation and detection process is to detect the bits $\mathbf{b}_i$ from the received signal $\mathbf{r}_i$. The computations that occur during the estimation phase [4,15] are

$$\mathbf{R}_{br} = \frac{1}{L}\sum_{i=1}^{L}\mathbf{b}_i\mathbf{r}_i^H \tag{2.2}$$

$$\mathbf{R}_{bb} = \frac{1}{L}\sum_{i=1}^{L}\mathbf{b}_i\mathbf{b}_i^H \tag{2.3}$$

where L is the length of the pilot sequence, $\mathbf{R}_{br} \in \mathbb{C}^{2K\times N}$ is the cross-correlation

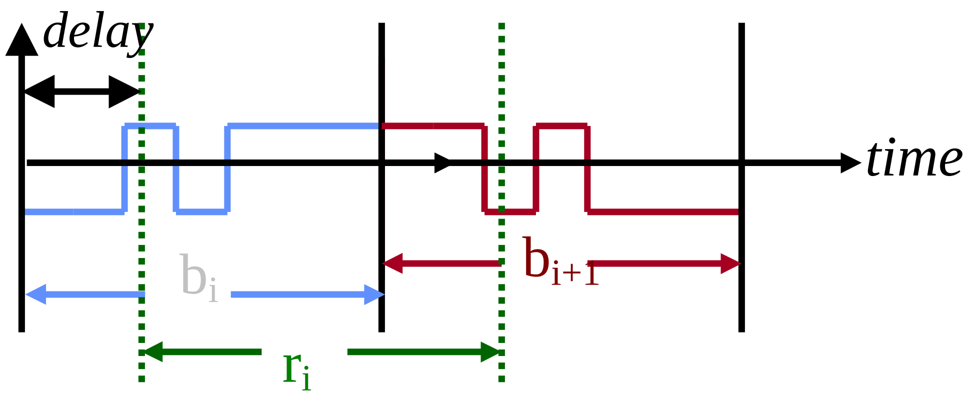

Figure 2.4 : Channel Model

matrix between the synchronization bits $\mathbf{b}_i$ and the received signal $\mathbf{r}_i$, and $\mathbf{R}_{bb} \in \mathbb{R}^{2K \times 2K}$ is the autocorrelation matrix. The channel estimate $\mathbf{A}_i$ can be obtained by solving

$$\mathbf{R}_{bb}\mathbf{A_i}^H = \mathbf{R}_{br} \tag{2.4}$$

The channel estimate in this matrix form can be directly fed to a multiuser detector in a joint estimation and detection scheme [4]. This results in computational savings as well as in error rate performance benefits.

2.3.1 Parameter Extraction Scheme

The channel estimates can also be extracted from the matrix and fed to the detector [16]. For this purpose, further computations are needed to extract the amplitudes and delays as shown below:

$$\begin{aligned}
\mathbf{R}_{rr} &= \frac{1}{L}\sum_{i=1}^{L} \mathbf{r}_i \mathbf{r}_i^H \\
\mathbf{K} &= \mathbf{R}_{rr} - \mathbf{Y}\mathbf{R}_{br}^H \\
\mathbf{z}_k^H &= (\mathbf{y}_{2k-1}^H \mathbf{K}^{-1}\mathbf{U}_k^R + \mathbf{y}_{2k}^H \mathbf{K}^{-1}\mathbf{U}_k^L) * (\mathbf{U}_k^{R'}\mathbf{K}^{-1}\mathbf{U}_k^R + \mathbf{U}_k^{L'}\mathbf{K}^{-1}\mathbf{U}_k^L)^{-1}
\end{aligned}$$

where $\mathbf{R}_{rr}$ is the autocorrelation of the observation vector, $\mathbf{K}$ is the noise covariance matrix, $\mathbf{U}$ is the matrix of codes which are known to the receiver, and $\mathbf{Z}$ is the channel impulse response matrix.

A least squares fit of $\mathbf{z}_k$ is performed to extract the strongest $\mathbf{P}$ paths. For each pair of adjacent coefficients of $\mathbf{z}_k$, we obtain local values of amplitudes and delays from the following optimization:

$$[w_q, \gamma_q] = \arg\min ||z_{k,q} - (1-\gamma)w||^2 + ||z_{k,q+1} - \gamma w||^2.$$

We then search for the global maxima to obtain the strongest path :

$$q = \arg\max |w_q|, \ \tau = (q + \gamma_q)T_c, \ w = w_q$$

where γ is the fractional part of the delay, q is the integer part of the delay, τ is the estimated delay, and w is the estimated amplitude. The estimated path is subtracted from $\mathbf{z}_k$ and the process is repeated to find the next strongest path until a specified number of paths have been identified. The estimated amplitudes and delays are then fed to a multiuser detector for accurate detection.

2.4 Multiuser Detection

Multiuser detection tries to cancel the interference from other users to improve the error rate performance, as opposed to single user detection using a matched filter [12]. The detection scheme discussed is the Differencing Multistage detection method [15, 17], which is based on the principle of Parallel Interference Cancellation. The channel estimate matrix can be fed directly into the multistage detector.

Dropping the subscript i for convenience, the matrix $\mathbf{A}_i$ can be rearranged into its odd and even columns $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{C}^{K\times N}$ which corresponds to the bits $\mathbf{b}_{i-1}$ and $\mathbf{b}_i$ in the estimate. In vector form, the received vector is

$$\mathbf{r}_i = [\mathbf{A}_0\mathbf{A}_1]\begin{bmatrix} b_{1,i-1} \\ \vdots \\ b_{K,i-1} \\ b_{1,i} \\ \vdots \\ b_{K,i} \end{bmatrix} + \eta_i \qquad (2.5)$$

2.4.1 Matched Filter Detector

The actual bits, $\mathbf{b}_i$, of the K users lie between the bits $\mathbf{r}_i$ and $\mathbf{r}_{i-1}$ bit boundaries. The matched filter detector does a correlation of the input bits with the received bits. Hence, the matched filter detector can be represented as

$$\mathbf{b}_i \quad = \quad \mathbf{A}1^H \mathbf{r}_{i-1} + \mathbf{A}0^H \mathbf{r}_i. \tag{2.6}$$

The multistage detector uses the matched filter to get an initial estimate of the bits and then subtracts the interference from other users.

2.4.2 Multistage Detector

The multistage detector performs parallel interference cancellation iteratively in stages, with the convergence of the bits increasing per stage. To subtract the interference, the interfering bits from other users have to be removed. The desired user's bits receives interference from the past or future overlapping symbols of different users because they are asynchronous. The effect of interference from the past and future symbols of users is as shown in Figure 2.5. To subtract interference from other users' future bits, a block based detection scheme is used. Detecting a block of bits simultaneously (multishot detection) can give performance gains [18]. In order to do multishot detection, the above model should be extended to include multiple bits. Let us consider D bits at a time ($i = 1, 2, \cdots, D$). So, we form the multishot received vector $\mathbf{r}$ of length ND by concatenating D $\mathbf{r}_i$-s ($i = 1, 2, \cdots, D$).

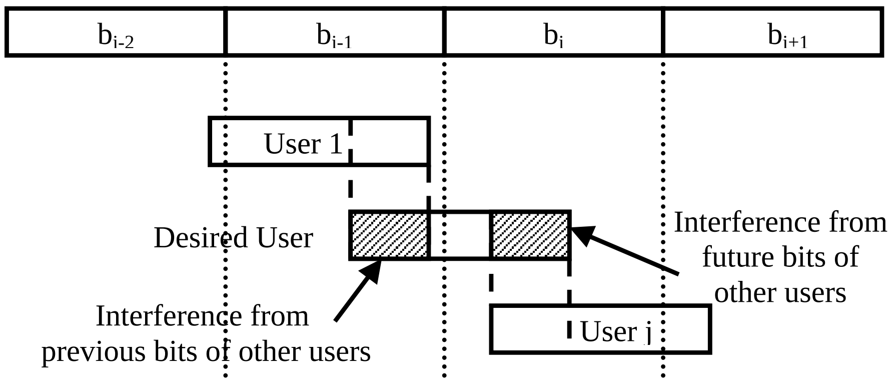
b_{i-2}
b_{i-1}
b_i
b_{i+1}
User 1
Desired User
Interference from future bits of other users
Interference from previous bits of other users
User j

Figure 2.5 : Multiple Access Interference from Past and Future bits of users

$$\mathbf{r} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & 0 & 0 \\ 0 & \mathbf{A}_0 & \mathbf{A}_1 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \mathbf{A}_0 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} b_{1,1} \\ \vdots \\ b_{K,1} \\ \vdots \\ b_{1,D} \\ \vdots \\ b_{K,D} \end{bmatrix} + \eta \qquad (2.7)$$

Let $\mathbf{A} \in \mathbb{C}^{ND \times KD}$ represent the multishot channel estimate matrix. We now proceed to the detection part of the algorithm after the formation of $\mathbf{A^H A}$ using the $\mathbf{A}$ matrix. The multistage multiuser detector needs initial estimates of the bits for performing the detection iteratively. The initial soft decision outputs $\mathbf{y}^{(0)} \in \mathbb{C}^{KD}$ and hard decision outputs $\mathbf{d}^{(0)} \in \mathbb{R}^{KD}$ of the detector are obtained from a matched filter as

$$\begin{aligned} \mathbf{y}^{(0)} &= Re[\mathbf{A^H r}] && (2.8)\\ \mathbf{d}^{(0)} &= sign(\mathbf{y}) && (2.9)\\ \mathbf{y}^{(1)} &= \mathbf{y}^{(0)} - Re[\mathbf{A^H A} - \mathbf{S}]\mathbf{d}^{(0)} && (2.10)\\ \mathbf{d}^{(1)} &= sign(\mathbf{y}^{(1)}) && (2.11) \end{aligned}$$

where $\mathbf{y}^{(1)}$ and $\mathbf{d}^{(1)}$ are the soft and hard decisions after the first stage of the joint detector and $\mathbf{S} \in \mathbb{R}^{KD \times KD}$ is the diagonal elements in $\mathbf{A^H A}$. The differencing method [15] is applied to take advantage of the convergence behavior of the iterations:

$$\begin{aligned} \mathbf{x}^{(l)} &= \mathbf{d}^{(l)} - \mathbf{d}^{(l-1)} && (2.12)\\ \mathbf{y}^{(l+1)} &= \mathbf{y}^{(l)} - Re[\mathbf{A^H A} - \mathbf{S}]\mathbf{x}^{(l)} && (2.13)\\ \mathbf{d}^{(l+1)} &= sign(\mathbf{y}^{(l+1)}) && (2.14) \end{aligned}$$

These computations are iterated $l = 1, 2, \cdots, M$ where M is the maximum number of iterations. Instead of performing hard decisions, the soft decisions could be forwarded to the decoding stage in a joint detection and decoding scheme [19] for better performance. The structure of $\mathbf{A^H A} \in \mathbb{C}^{KD \times KD}$ is as shown:

$$\begin{bmatrix} \mathbf{A_0^H A_1} & \mathbf{A_0}^{\mathbf{H}}\mathbf{A_1} & 0 & 0 \\ \mathbf{A_1}^{\mathbf{H}}\mathbf{A_0} & \mathbf{A_0}^{\mathbf{H}}\mathbf{A_0} + \mathbf{A_1}^{\mathbf{H}}\mathbf{A_1} & \mathbf{A_0}^{\mathbf{H}}\mathbf{A_1} & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \mathbf{A_1}^{\mathbf{H}}\mathbf{A_0} & \mathbf{A_0}^{\mathbf{H}}\mathbf{A_0} + \mathbf{A_1}^{\mathbf{H}}\mathbf{A_1} \end{bmatrix} \qquad (2.15)$$

The hard decisions, $\mathbf{d}$, which are made at the end of the final stage, are fed back to the estimation block in the decision feedback mode for tracking in the absence of the pilot signal and to the rest of the processing blocks in the receiver.

Chapter 3

DSP Implementation and Task Partitioning

This chapter discusses the implementation of the channel estimation and multiuser detection algorithms on current generation C6x TI DSP processors and shows how real-time performance can be achieved by task-partitioning the algorithms on multiple processors.

3.1 Implementation Methodology

The channel estimation and detection algorithms are implemented on a TI C6x DSP [20] with a TI TMS320C6701 (C67) floating point processor. This processor is taken as an example of the current generation processor technology for our analysis. The C67 [21] is one of the recent DSPs from TI, which has a high-performance VLIW (Very Long Instruction Word) architecture and has been proposed for wireless base-stations. It has a 32-bit architecture with 8 functional units, consisting of 2 multipliers, 4 ALUs and 2 Load/Store Units. It has hardware support for IEEE single and double precision floating point instructions and can produce 2 Multiply and Accumulate's (MAC) per cycle. The algorithms to be evaluated are written in a memory-efficient manner using the 'C' programming language so as to avoid transposes and to utilize inplace computations. The entire code and data segments fit in the internal memory of the DSP. In this initial implementation, the LU decomposition [22, 26] is used to calculate the matrix inversions. The TI C Compiler ver 3.0 [23] is used to generate the

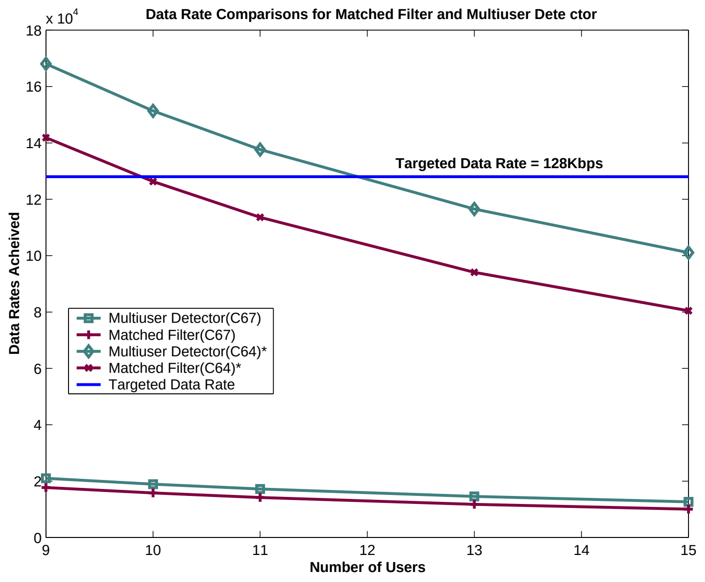

Figure 3.1 : Data Rate Comparisons for a Matched Filter and Multiuser Detector on a single DSP

assembly code for the DSP. The highest possible compiler optimizations recommended by TI [24,25] are used. The optimizations perform software pipelining, loop unrolling and other program level optimizations to exploit the available fine-grain parallelism available in the VLIW architecture. The structure and sparseness of the various matrices are also accounted for in the implementation.

3.2 DSP Implementation and Comparisons

The DSP implementation of the algorithm is shown in Figure 3.1. The graph shows the data rates achieved on a C67 processor for a matched filter and 3-stage multistage detector for varying number of users. It can be observed that the matched filter detector achieves a data rate of 18.8 Kbps for each of the 15 users as compared to 10.7 Kbps for the multistage detector. Thus, a 3-stage multistage detector reduces the data rate by half, but it provides significant error rate performance benefits [15].

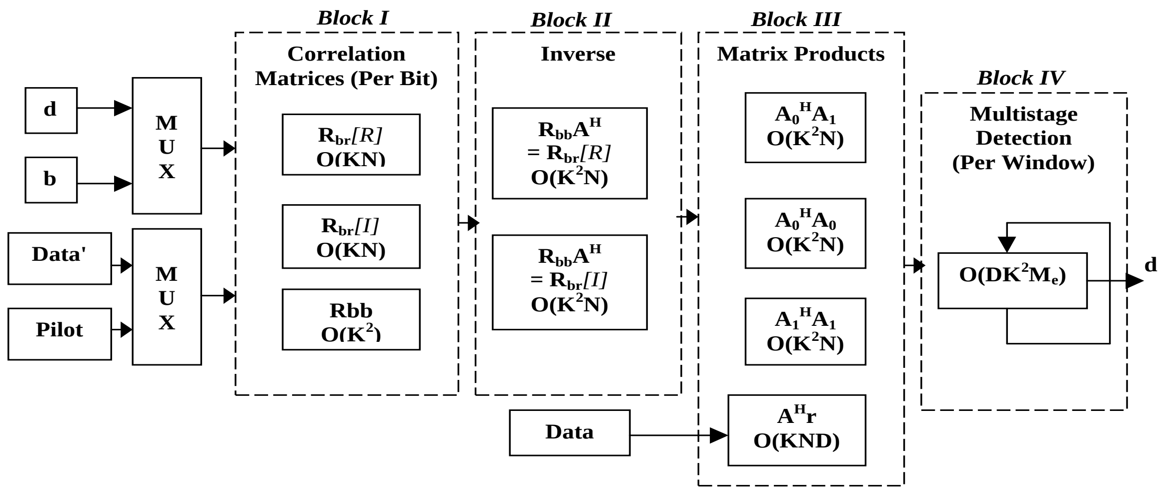

Figure 3.2 : The Task Partition Graph for the Joint Estimation and Detection Algorithm

However, both fall far short of the targeted 128 Kbps for each user in the system. Even the recently announced fixed TI processor, the C64x, which is projected to have an 8x performance improvement (neglecting the fixed-floating conversion), does not meet the targeted rate for 15 users. Also, note that detection is just one of the processing blocks in the receiver and that other compute-intensive blocks such as multiuser channel estimation and decoding, which are currently assumed to be pipelined on different processors, need to be implemented. Hence, there is a need for orders-of-magnitude performance improvements in DSP technology in order to come up with a single DSP-based base-station receiver.

3.3 Task Decomposition

The sequential implementation of the entire algorithm on the DSP does not meet real-time constraints as seen from Figure 3.1. The achieved data rates for just the detection block implementation, assuming a single stage iteration of the multistage

Block	Complexity	Cycle count
Correlation Matrices	$KN + K^2$	27957
Inverse	K^2N	763401
$A_0^H A_1, A_0^H A_0, A_1^H A_1$	K^2N	124205
$A^H r$	KND	132723
per bit	KN	13272
Multistage(1st stage)	DK^2	33669
per bit	K^2	3367

Table 3.1 : Cycle count and Complexity for different blocks

detector, show the data requirements falling short by a factor of 6. So, a task decomposition of the algorithm is carried out to find the data dependencies and to identify all available sources of pipelining and parallelism. A coarse-grained pipelined-parallel task decomposition of the joint estimation and detection algorithm, detailed in Chapter 2, is as shown in Figure 3.2. The input to the channel estimation block to the left is either the known pilot bits (b) and the received pilot bits (Pilot) or the previously detected data bits (d) and the received data bits, delayed by the time required for detection (Data'). The dotted blocks (I-IV) represent pipelined operations whereas the blocks inside a dotted block represent operations that can be done in parallel.

Block I shows both the correlation matrices, $\mathbf{R}_{br}$ and $\mathbf{R}_{bb}$, (Chapter 2, equation 2.2) which can be computed in parallel. These are outer product computations. Also, both the real and imaginary parts of $\mathbf{R}_{br}$ can be computed independently. Block II shows the inverse of $\mathbf{R}_{bb}$ (Chapter 2, equation 2.4), which is calculated by using a LU Decomposition. Block III shows the computation of the different matrix products required for forming the multishot channel estimate. The matrix product, $\mathbf{A}_1{}^H\mathbf{A}_0$

Optimization	Cycle count
A + B	13272 + 3367$\times M_e$
A B	max(13272,3367$\times M_e$)
(Pl A) B	3367$\times M_e$
(Pl A) (Pp B)	3367
(Pl A) (PlPp B)	885

Table 3.2 : Cycle count for different optimization levels of blocks A ($\mathbf{A}^H\mathbf{r}$) and B (block IV)

is not computed as it is $(\mathbf{A}_0{}^H\mathbf{A}_1)^H$. Block III also includes the computation of the Matched Filter (chapter 2, equation 2.8), as it can be done in parallel with the above operations. The iterative loop of the Multistage Detection (Chapter 2, equation 2.12-2.14) is shown as a single Block IV.

The input data bits are streaming in continuously to the receiver, which has to ensure that the received data stream is being continuously processed so as to meet the real-time constraints. However, the channel estimation can be updated less frequently so as to meet with the requirements of the detection. (A slow fading channel model is assumed). The parts of multiuser detection which depend on the input data are the calculation of the Matched Filter $\mathbf{A}^H\mathbf{r}$, and the multistage detection loop. An order complexity analysis is also done on the algorithm to find the bottlenecks in each block.

3.4 Simulations and Analysis

An in-depth profiling of the various blocks is carried out using the clock function in the C6x DSP. The cycle count for the various blocks is as shown in Table 3.1 for 15

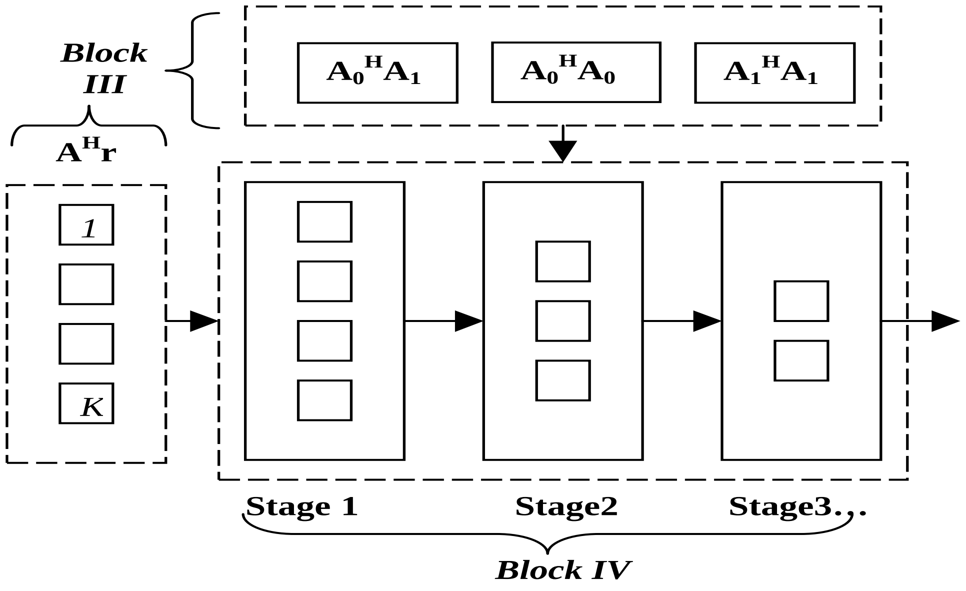

Figure 3.3 : Further Pipelining & Parallelism in the Multistage Detection

users and a detection window of length 12. Assuming a 250 MHz processor, a data rate requirement of 128 Kbps implies that the available number of cycles per bit is 1953 cycles for real-time detection. The successive stages in the Multistage Detector take significantly less time than the first stage. Hence, let the effective number of stages be M_e where $M_e \leq M$. The time required for block IV for all the M_e stages can exceed the time required to calculate the matched filter $\mathbf{A}^H\mathbf{r}$. Also, the first and last K bits in each window are ignored due to edge effects and have to be recalculated. The task partition graph at this level is unable to match the real time constraints as the present solution still requires $13272 + 3367 \times M_e$ cycles. Therefore, more fine grain parallelism from the above task partition graph needs to be explored.

Table 3.2 shows the advantages of various levels of parallelism(Pl) and pipelining(Pp). Let Task A refer to the calculation of the matched filter $\mathbf{A}^H\mathbf{r}$ in Block III and Task B to Block IV. Let (A + B Sequential) be the present solution obtained.

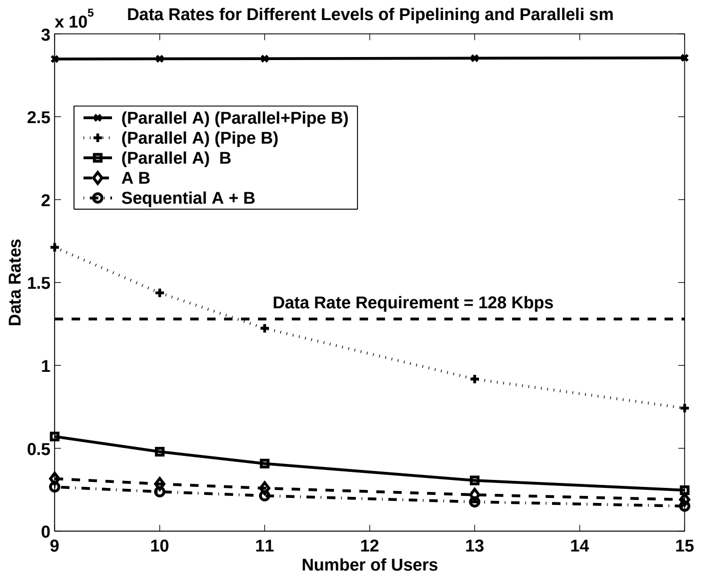

Figure 3.4 : Data Rates for various levels of Pipelining and Parallelism for A ($\mathbf{A}^H\mathbf{r}$) and B (block IV)

If Tasks A and B were pipelined (A B), the required computation becomes the maximum of task A and B. Next, the matched filter $\mathbf{A^H r}$ can be done for each user in parallel as each row of $\mathbf{A^H}$ corresponds to a user, reducing the time in (Pl A) to 885 cycles. This puts the bottleneck to block B of (Pl(A) B) case. Hence, Block B is also unrolled and pipelined into different stages. The first parallel interference cancellation stage now has the most complexity and becomes the new bottleneck, needing 3367 cycles ((Pl A) (Pp B)). It has been shown [15] that each successive stage in B requires less computation than the previous stage. Hence, fewer or less powerful processing elements need to be used for these stages. Each stage can also be split into multiple processing elements in a manner similar to Block A. This reduces the cycles needed to 225, putting the bottleneck back to Task A in (Pl(A) PlPp(B)). Tasks A and B after this final step are shown in Figure 3.3.

3.5 Meeting Real Time Constraints

The data rates which can be met with different levels of pipelining and parallelism are shown in Figure 3.4. The figure shows the variation in the achieved data rates with the number of users. We assume that the effective number of stages of the multistage detector is 3 ($M_e = 3$). As the level of pipelining and parallelism increases, we observe an increase in the data rates. The data rates from (Parallel A)(Pipe B) satisfies the requirements for fewer number of users (≤ 10) as it is limited by the complexity of the first stage which is $O(K^2)$. By having K processing elements for the first stage of Block IV, the bottleneck shifts back to Task A ((Parallel A)(Parallel + Pipe B)), which is of order $O(N)$ and hence, the data rate achieved is independent of the number of users. Note however, that this is because the number of processors is dependent on the number of users.

Judging from the time requirements for Block I and Block II in channel estimation, we can update block II once in 27 updates to Block I. The frequency of updates is determined by the amount of error that can be tolerated in the detection. If the updates are not frequent enough to keep up with the fading of the channel, the performance of the system will degrade in terms of the bit error rate. More frequent updates of once in 14 bits can be achieved by again further partitioning the matrix inverse into 2 separate tasks. Here, the key idea is to use the amount of parallelism necessary to satisfy the bit error rate tolerance levels. Alternate methods could also be used for computing the inverse to reduce the complexity and make more updates feasible. These are discussed in the next chapter.

Chapter 4

Reduced-Complexity Channel Estimation and Detection

The previous chapter discussed means to achieve real-time using task partitioning. However, the amount of processors needed to achieve real-time was $O(K^2)$, which is quite large. Hence, a direct implementation of the algorithms shown in Chapter 2 is not feasible. This chapter discusses modifications to the algorithms for channel estimation and detection to make them computationally efficient and suitable for a hardware implementation.

4.1 Iterative Channel Estimation

A direct computation of the exact Maximum Likelihood channel estimate $\mathbf{Y}$ involves the computation of the correlation matrices $\mathbf{R}_{bb}$ and $\mathbf{R}_{br}$, and then the computation of $\mathbf{R}_{bb}^{-1}\mathbf{R}_{br}$ at the end of the preamble (Chapter 2, equation 2.4). The computation of the inverse at the end of the preamble is computationally expensive and delays the start of detection beyond the length of the preamble until the estimate has been computed and this delay limits the information rate. In the iterative algorithm, the Maximum Likelihood solution is approximated based on the following ideas:

1. The product $\mathbf{R}_{bb}^{-1}\mathbf{R}_{br}$ can be directly approximated using iterative algorithms such as the conjugate gradient algorithm [26].
2. The iterative algorithm is modified to update the estimate as the preamble is

being received rather than waiting until the end of the preamble. This means that the computation per bit is reduced by spreading it over the entire preamble.

An iterative scheme based on the method of gradient descent (a variation of Richardson's method) [26] for the matrix inversion is presented. In this scheme, the channel estimate $\mathbf{Y}$ is updated iteratively every bit and hence is available immediately after the end of the pilot sequence. Here, the updating of the estimate can be done every bit using the iterative scheme as shown:

$$\mathbf{R}_{br} = \mathbf{R}_{br} + \mathbf{b}_i \mathbf{r}_i^H - \mathbf{b}_{i-L} \mathbf{r}_{i-L}^H \tag{4.1}$$

$$\mathbf{R}_{bb} = \mathbf{R}_{bb} + \mathbf{b}_i \mathbf{b}_i^T - \mathbf{b}_{i-L} \mathbf{b}_{i-L}^T \tag{4.2}$$

$$\mathbf{Y} = \mathbf{Y} - \mu(\mathbf{R}_{bb} * \mathbf{Y} - \mathbf{R}_{br}) \tag{4.3}$$

This scheme is suitable for tracking, which is shown by the removal of the oldest bit in the window of length L as the new bit is received. Tracking is simpler in this iterative scheme because the channel estimates and correlation matrices are updated iteratively. During the initial pilot phase, tracking is absent and the equations for correlation reduce to the equations in the previously shown estimation scheme in Chapter 2 (equation 2.2-2.4), using matrix inversion. The iterative algorithm approximates the maximum likelihood solution as the preamble is being received. As the l^{th} preamble observation is received, it tries to evaluate the maximum likelihood solution given l observations. The accuracy of the iterative estimate can be improved by increasing the number of iterations during each bit. In our simulations, we perform only one iteration per preamble bit as this is sufficient for the reasonable simulation parameters chosen. The algorithm shows good convergence behavior since $\mathbf{R_{bb}}$ is a symmetric positive definite matrix and has a small condition number [26]. The

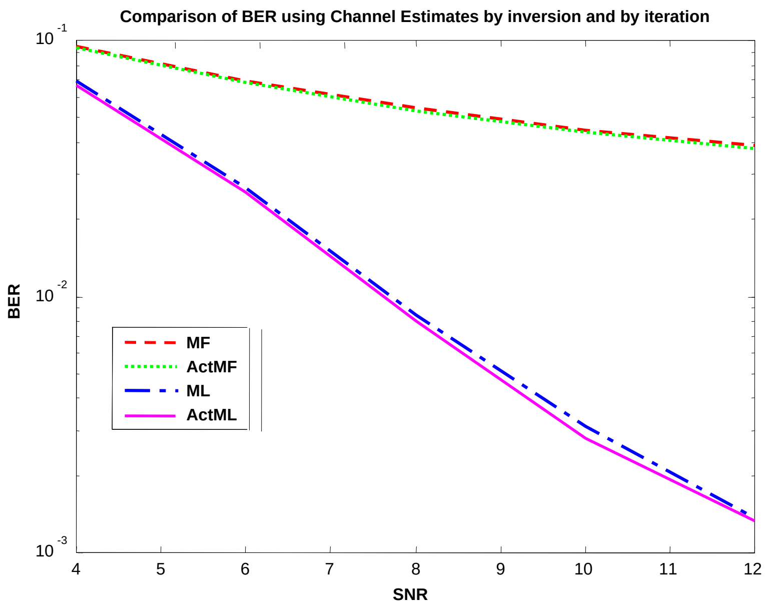

Figure 4.1 : Error Rate Performance in Additive White Gaussian Noise Channel

parameter , μ, should be chosen such that it is smaller than the largest eigenvalue of the auto-correlation matrix. A detailed analysis of this scheme is presented in a similar context for long codes in [14].

Figure 4.1 shows the performance of both schemes in an AWGN environment after the end of the pilot phase against two types of detectors, a Matched Filter Detector(MF) and a Differencing Multistage Detector(ML) [15]. The simulations are carried out for a preamble of length 150 bits, having 3 paths, for 15 users, all transmitting at the same power and for a detection window length of 12, with 10000 bits per user. The value of μ for the iterative scheme was chosen to be 0.0001. From the simulations, it can be observed that the iterative scheme gives almost the same error rate performance as that of the original scheme(ACT) but yet, has reduced complexity due to spreading the computations over the length of the preamble.

The analysis of the system for a fading channel with tracking is shown in Figure 4.2. Here it can be seen that the proposed tracking scheme based on the iterative

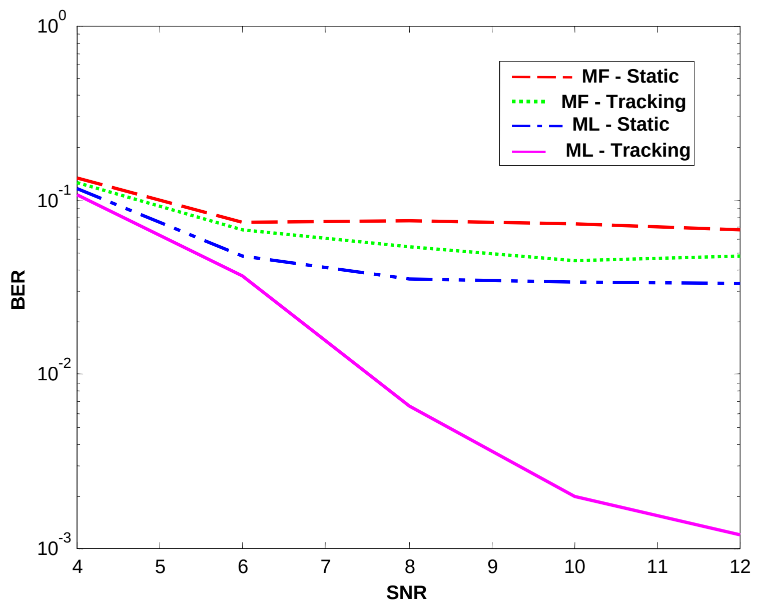

Figure 4.2 : Error Rate Performance in a Fading Channel

scheme is able to effectively track the time-varying channel. The poor performance of the static channel assumption for this Rayleigh fading channel (of Doppler spread 10 Hz) shows the importance of tracking. The simulation is done using 1000 bits per user for 15 users with equal power, for a Rayleigh fading channel with a Doppler of 10Hz.

4.2 Estimation based on Precomputed Preamble

Significant computational savings can be achieved if the autocorrelation matrix and its inverse (Chapter 2, equation 2.2-2.4) are precomputed as follows:

$$\mathbf{R}_{bb} = \sum_{i=1}^{L} \mathbf{b}_i \mathbf{b}_i^T \tag{4.4}$$

$$\mathbf{c}_i = \mathbf{R}_{bb}^{-1} \mathbf{b}_i \tag{4.5}$$

$$\mathbf{Y} = \mathbf{Y} + \mathbf{c}_i \mathbf{r}_i. \tag{4.6}$$

Thus, assuming prior computation of the autocorrelation matrix $\mathbf{R}_{bb}$ and its inverse $\mathbf{R}_{bb}^{-1}$, the coefficients $\mathbf{c}_i$ can also be precomputed and stored in memory as a lookup table. The channel estimate $\mathbf{Y}$ can be updated immediately on the reception of the received signal $\mathbf{r}_i$. Thus, a matrix multiplication can be reduced to an outer product update, thereby reducing the complexity. This scheme can be used only during the initial preamble phase as it requires previous computation of the autocorrelation matrix.

4.3 Pipelined Detection

The block-based multishot detection scheme in chapter 2(section 2.4.2) was proposed as it requires computation of future incoming bits of users, which are not available. This results in taking a window of (D+2) bits and using it to detect D bits as the edge bits are not detected accurately due to windowing effects. Thus, there are 2 additional computations per block and per iteration that are not used. Also, such a block-based implementation needs a windowing strategy and has to wait until all the bits in the window are ready for computation. This is as shown in Figure 4.3 for a detection window of length 10. It can be observed that the detection is done in blocks of 12 bits, and the edge bits are thrown away and recalculated in the next iteration.

However, the stages in the multistage detector can be efficiently pipelined to avoid edge computations and to work on a bit streaming basis. This can be done due to the block Toeplitz nature of the matrix $\mathbf{A^H A}$ as seen in chapter 2, equation 2.15. The

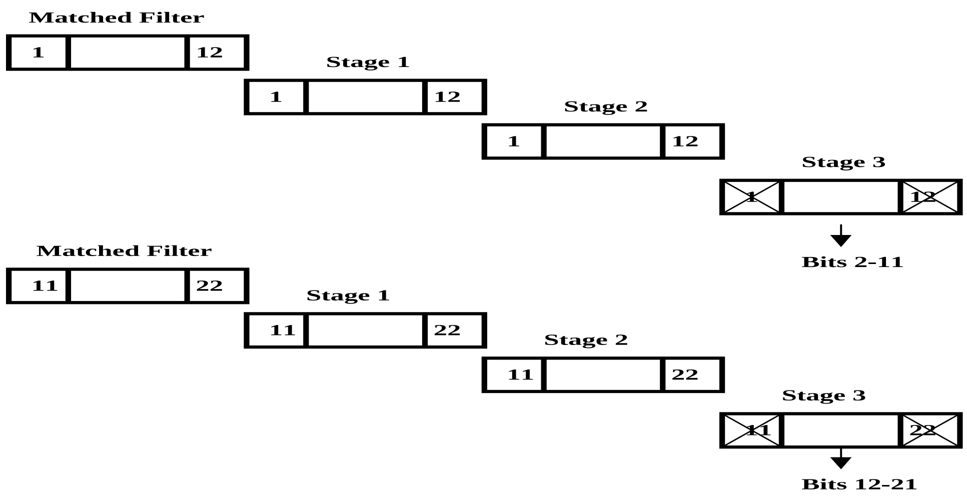

Figure 4.3 : Block Based Detection

computations performed on the intermediate bits reduce to

$$\mathbf{L} = \mathbf{A1^H A0} \tag{4.7}$$

$$\mathbf{C} = (\mathbf{A0^H A0} + \mathbf{A1^H A1}) - diag(diag(\mathbf{A0^H A0} + \mathbf{A1^H A1})) \tag{4.8}$$

$$\mathbf{y}_i = \mathbf{y}_i - \mathbf{L}\mathbf{b}_{i-1} - \mathbf{C}\mathbf{b}_i - \mathbf{L}^H\mathbf{b}_{i+1} \tag{4.9}$$

This equation may be thought of subtracting the interference from the past bit of users, who have more delay, and the future bits of the users, who have less delay than the desired user. The left matrix **L**, stands for the partial correlation between the past bits of the interfering users and the desired user, the right matrix $\mathbf{L^H}$, stands for the partial correlation between the future bits of the interfering users and the desired user. The center matrix **C**, is the correlation of the current bits of interfering users and the diagonal elements are made zeros to avoid self-cancellation. Thus, the similarity of the above equation (4.9) to the model chosen for output of the matched filter [27] is as shown:

$$\mathbf{y}_i = \mathbf{R}(1)Ab_{i-1} + \mathbf{R}(0)Ab_i + \mathbf{R}(-1)Ab_{i+1} + \mathbf{w}_i \tag{4.10}$$

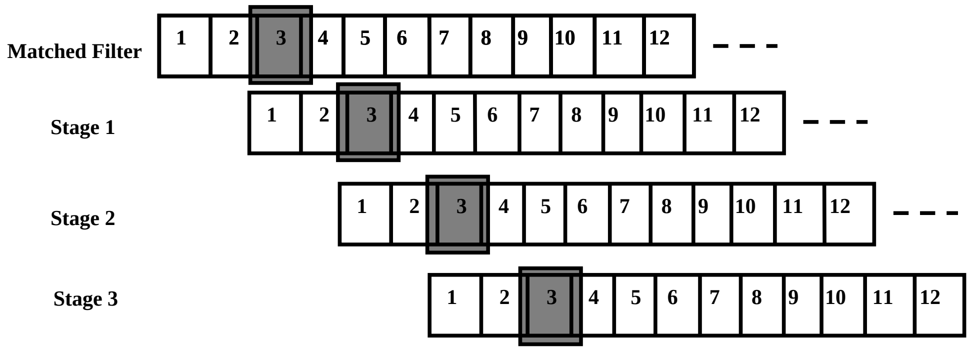

Figure 4.4 : Pipelined Detection

where **R** is the correlation matrix, **A** is a diagonal amplitude matrix, **w** is the noise vector and **b** is the symbol vector.

The detection can now be pipelined as shown in Figure 4.4. An example using bit 3 of the detector is shown. An initial estimate of the received signal is done using a matched filter detector, which depends only on the current and the past received bits. The stages of the multiuser detector need bits 2 and 4 of all users to cancel the interference for bit 3. Hence, the first stage can cancel the interference only after bits 2 and 4 estimates of the matched filter is available. The other stages have a similar structure. Hence, while bit 3 is being estimated from the final stage, the matched filter is estimating bit 9, the first stage bit 7 and the second stage bit 5. There are no edge bit computations in this scheme and hence, they can be avoided and recalculated in the next iteration as the window progresses.

4.4 Computational Savings

The schemes discussed in the previous sections have no degradation in error rate performance. The computational advantages of the newly proposed schemes in this

Blocks	**Precomputed Preamble**	**Original**	**Iterative**
Channel Estimation	Yes	$O(4K^2N)$	$O(2KN)$
	No	$O(6K^3 + 4K^2N)$	$O(4K^2N)$
Multiuser Detection (per D bits)		$O(DNK + 3(D+2)K^2)$	$O(DNK + 3DK^2)$

Table 4.1 : Comparisons of Computational Savings

chapter over the previous schemes in Chapter 2 are shown in Table 4.1.

The original channel estimation scheme with the precomputed matrix inverse needs to do a matrix multiplication $O(4K^2N)$ to obtain the channel estimate. The channel estimation scheme needs only a matrix rank 1 update $O(2KN)$ due to its iterative nature. If the channel estimation is used for fading channels, where the inverse cannot be precomputed, both the matrix inversion as well as the matrix multiplication needs to be done $O(6K^3 + 4K^2N)$ while estimation using the gradient method requires only a multiplication $O(4K^2N)$. For comparing the detection schemes, we assume that a window of D bits need to be detected. For every window, we save $O(6K^2)$ computations, assuming a 3-stage detector as the edge bits do not need to be calculated.

Chapter 5

VLSI Architecture for Channel Estimation

The task partitioning of the channel estimation algorithm into sub-blocks is carried out for pipelining and for utilizing the inherent parallelism present. Different mappings of the multiuser channel estimation algorithm to hardware are implemented to study the complexity and hardware requirement tradeoffs. A serial architecture, with minimum hardware requirements, a parallel solution in minimum time and an area-time efficient solution are discussed. An area-constrained architecture is a tradeoff for minimizing the area with increased computational time, which may be suitable for 'picocell' applications with lower data rates. A time-constrained architecture is used to evaluate the potential parallelism in the algorithm and find maximum theoretical data rates. The area-time efficient architecture meets the real-time requirements with minimum area overhead.

5.1 Task Decomposition

The task decomposition of the channel estimation algorithm is shown in Figure 5.1. The blocks that are pipelined are shown on the horizontal time axis while the blocks that have coarse-grained parallelism are shown along the vertical axis. The figure shows that the correlation matrices in Chapter 4 (equation 4.1-4.2) can be formed in parallel and the correlation can be pipelined with the iteration of the channel estimate matrix (equation 4.3). The two multiplexers shown are for selecting between the

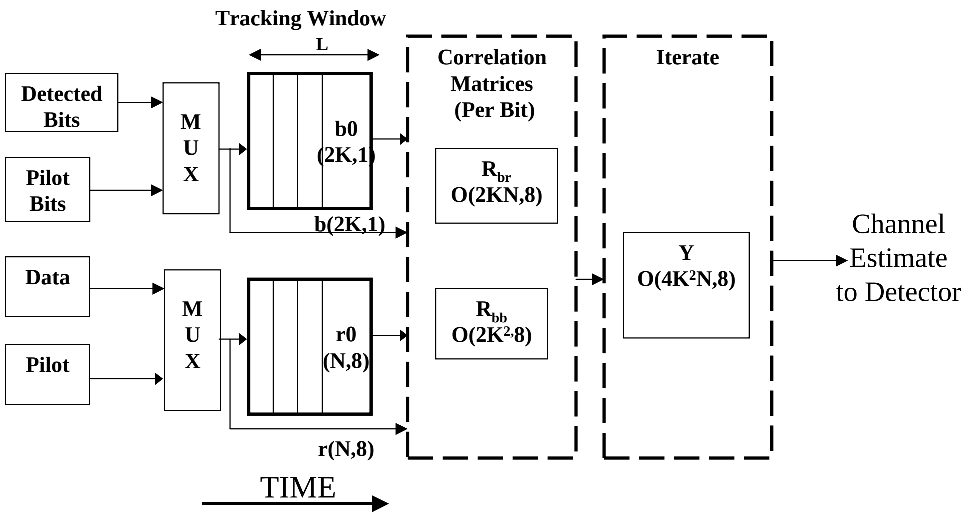

Figure 5.1 : Task Decomposition of Multiuser Channel Estimation Algorithm

known pilot and the received pilot signal during the training mode and the detected bits and the received data signal in the tracking phase. The tracking window is the history buffer and keeps the L most recent samples of the bits as well as the received signal. The sizes of the sub-blocks are shown along with their word lengths in the figure. The dynamic range of the input is dependent on Signal-to-Noise ratio (SNR), the Multiple Access Interference (MAI) and the number of users in the system. A detailed analysis is required to determine the word-length of the input. It is assumed that the received signal is quantized by an A/D converter to have a fixed precision word-length of 8 bits as a similar dynamic range analysis [15, 28] for detection shows the input range to be 8 bits. However, the analysis of the algorithm presented here is independent of the word-length. Also, note that the blocks $\mathbf{r}$, $\mathbf{R_{br}}$ and $\mathbf{Y}$ are complex-valued while $\mathbf{b}$ and $\mathbf{R_{bb}}$ are real-valued. For the sake of convenience, the current inputs $\mathbf{b_i}$, $\mathbf{r_i}$ can be represented as $\mathbf{b}$, $\mathbf{r}$ and $\mathbf{b_{i-L}}$, $\mathbf{r_{i-L}}$ as $\mathbf{b}0$, $\mathbf{r}0$.

A typical architecture has window length L = 150, spreading gain N = 32 and the number of users K = 32. For all the architectures shown here, we assume that the

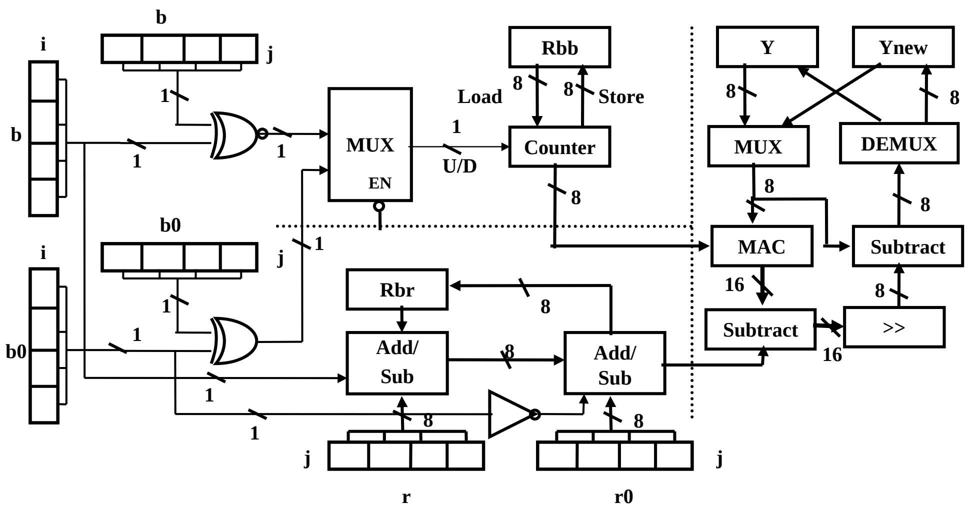

Figure 5.2 : Area-Constrained VLSI Architecture

unit of time in cycles is the time required for an 8-bit multiplication and addition. We assume that a Wallace or Dadda multiplier tree [29] is used for multiplication requiring $O(n^2)$ 1-bit Full Adders for a n-bit multiplication. Since the multiplication by μ in the iteration loop (Chapter 4, equation 4.3) results in truncation of the output and need not be highly accurate for numerical stability, a truncated multiplication using significantly less hardware [30] can be used. The delays of blocks such as multiplexers and gates are assumed to be included in the single-cycle delay. For an area estimate of the architectures, equivalent of the number of 1-bit Full Adder Cells in the design are considered. We assume all blocks can be pipelined effectively. It can be observed from Figure 5.1 that the bottleneck in the pipeline is the matrix-matrix multiplication between $\mathbf{R}_{bb}$ and $\mathbf{Y}$, and this will be the focus in the proposed architectures.

5.2 Area-Constrained Architecture

An area-constrained architecture of the multiuser channel estimation scheme is as shown in Figure 5.2. The architecture shown computes only the real part of the channel estimate. Since there are no multiplications between two complex numbers, the architecture can be assumed to be replicated for the imaginary part. In this architecture, all matrix elements are computed an element at a time. The word lengths of the various blocks are as shown in Figure 5.2. The dotted lines indicate the separation between the auto-correlation, cross-correlation and the iteration loop (Chapter4, equation 4.1-4.3). The left part shows the calculation of the auto-correlation and cross-correlation matrices whereas the right part shows the calculation of the iteration loop.

To form the outer product update, we take advantage of the single bit nature of the data and replicate the bits $\mathbf{b}, \mathbf{b0}$ such that for forming the $(i,j)^{th}$ element of $\mathbf{R_{bb}}$, the i^{th} and j^{th} bit of $\mathbf{b}$ are EX-NORed (multiplication between +1 and -1 is an EX-NOR operation) and sent to a counter loaded with the previous value of $\mathbf{R_{bb}}$ which increments or decrements by one. The $(i,j)^{th}$ element of the outer product update (Chapter 4, equation 4.1) $\mathbf{b0} * \mathbf{b0^T}$ is calculated, negated and sent to the counter, which again increments or decrements by 1 (Up/Down). The multiplexer also has an enable signal such that the output is tristated during the pilot phase, when $\mathbf{b0} * \mathbf{b0^T}$ is not computed. The matrix $\mathbf{R_{bb}}$ is then updated with a store signal.

A MAC (Multiply and Accumulate) unit is used to compute the inner product of the matrix multiplication (Chapter 4, equation 4.3) $\mathbf{R_{bb}} * \mathbf{Y}$. If we design a MAC unit such that the multiplication and addition are pipelined with the other blocks in the figure, computing an element of $\mathbf{R_{bb}} * \mathbf{Y}$ takes $2K$ or 64 cycles. The corresponding element of $\mathbf{R_{br}}$ is updated similarly with an adder. The multiplication by μ is then

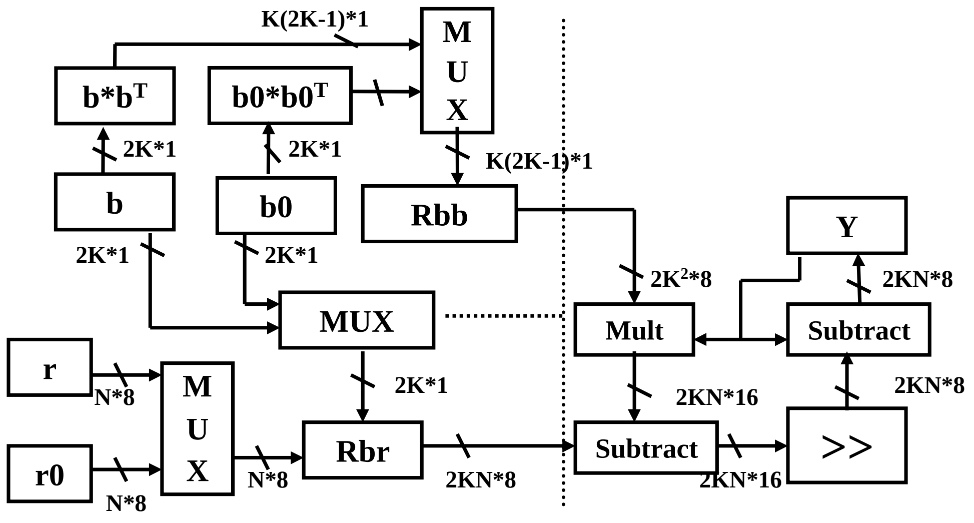

Figure 5.3 : Time-Constrained VLSI Architecture Block Diagram

carried out with the help of a right shift and the new $(i, j)^{th}$ element of $\mathbf{Y}$ comes out of the pipeline every $2K$ cycles. The MUX-DEMUX circuit loads from $\mathbf{Y}$ and stores in $\mathbf{Y}_{new}$ for every $4K^2N$ or 128,000 cycles (the time taken to compute the entire matrix) and then switches. The hardware requirements for an area-constrained architecture are as shown in Table 5.1. The design requires an 8-bit counter, an 8-bit multiplier, three 8-bit adders and two 16-bit adders (for the MAC and the subtraction by $\mathbf{R_{br}}$), about 112,000 bits of memory and $4K^2N$ cycles.

5.3 Time-Constrained Architecture

The block diagram of a time-constrained architecture is as shown in Figure 5.3. In this architecture, the available parallelism in the algorithm is exploited to the maximum extent. Hence, all the elements needed to perform a parallel multiplication are computed simultaneously and are pipelined. In this case, the entire matrices $\mathbf{R_{bb}}$ and $\mathbf{Y}$ are multiplied by using an array of multipliers. The entire product matrix is

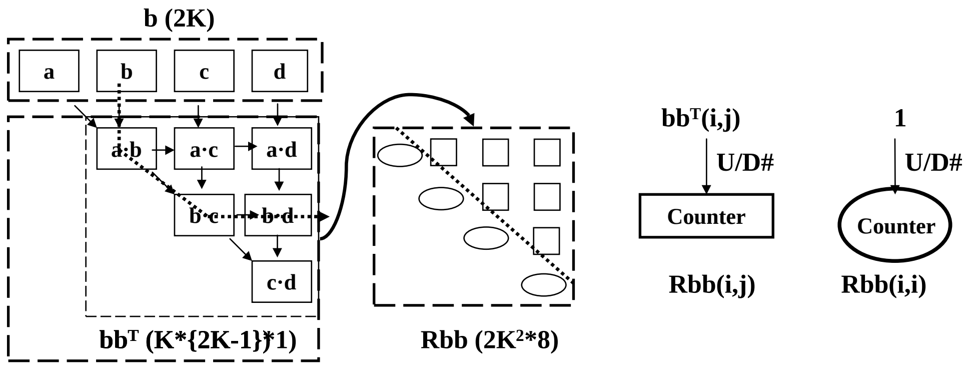

Figure 5.4 : Elements in the Auto-correlation Matrix Block

subtracted from the auto-correlation matrix, $\mathbf{R_{br}}$, shifted and a new channel estimate is formed. Thus, as the time taken by the other computations is pipelined with the time for the multiplication, the output can be formed every $\log_2(2K)$ or 6 cycles.

The bit-level arithmetic and parallel structure of the correlation matrices are exploited to form the correlation matrices simultaneously within a cycle. The sub-blocks for the formation of the auto-correlation matrix and cross-correlation matrix are shown in Figure 5.4 and Figure 5.5. Since the auto-correlation matrix update is a symmetric matrix and all the diagonal elements are 1's (a EX-NOR a = 1), only the strictly upper triangular (or lower triangular) part of the auto-correlation matrix (Figure 5.4) needs to be computed. Also, as the updates are all +1's or -1's, the bit multiplications can be obtained from a simple EX-NOR gate structure. The counters in the auto-correlation matrix are then updated based on the sign of the updates. Also, the elements in the cross-correlation update are $+r$ or $-r$, and hence the vector $\mathbf{r}$ could be directly added or subtracted with every column of the auto-correlation matrix based on the sign of the bit vector $\mathbf{b}$. The hardware requirements for the time-constrained architecture are as shown in Table 5.2. Though the hardware re-

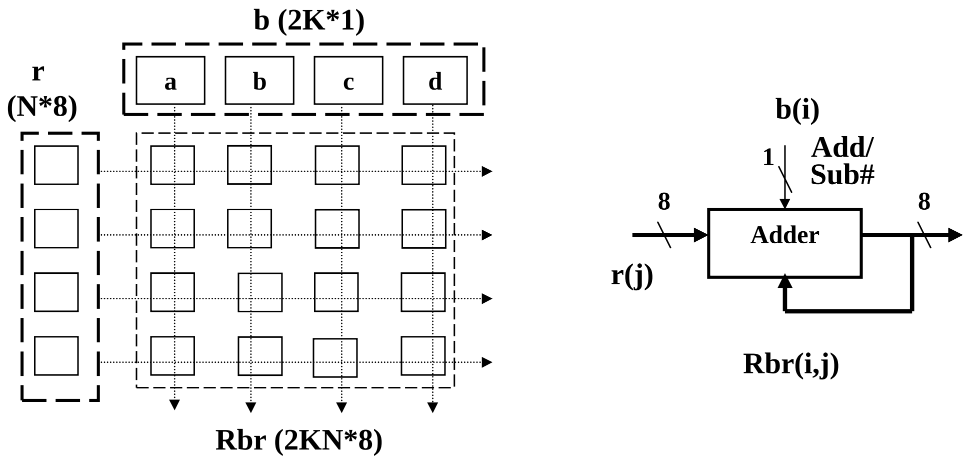

Figure 5.5 : Elements in the Cross-correlation Matrix Block

quirements increase by 5 orders of magnitude, the memory requirements decrease as there is no need for storage and there is a significant speedup in time (5 orders) obtained compared to the area-constrained architecture which shows the potential parallelism in the architecture. For a typical implementation, the number of Full Adder Cells required is 20,000,000. This is a far too aggressive solution and difficult to implement even with current silicon technology. However, the architecture states the theoretical minimum time requirements (maximum data rates achieved) by exploiting the available parallelism as $\log_2(2K)$ or 6 cycles, which is the time required to do the parallel multiplication and pipelining it with the other blocks. $2KN(2K-1)$ 16-bit adders are required for doing the recursive doubling in $\log_2(2K)$ time [adding $2K$ elements in $\log_2(2K)$ time requires $(2K-1)$ adders] and $2KN$ 16-bit adders for the subtraction following the multiplication.

5.4 Area-Time Efficient Architecture

From comparing the above two architectures in Table 5.4, the area-constrained architecture design does not meet real-time requirements while the time-constrained architecture is highly aggressive in area. So, a tradeoff point in the design space needs to be found, which meets the real-time requirements with minimum additional area. This can be done by observing that the major part of the chip area calculated is used by the array of multipliers. Hence, instead of computing the entire matrix product (Chapter 4, equation 4.3) in parallel, the product should be computed element by element by doing the inner product in parallel. This would imply $4K$ or 128 multipliers. If this was done row by row or column by column, it would require $4K^2$ or $4KN$ multipliers, requiring about 3600 multipliers, which may not be available just for channel estimation. Since the output is computed element-by-element, this would require $2KN$ or 2000 cycles for the complete channel estimate. The block diagram of the area-time efficient architecture is shown in Figure 5.6.

The hardware requirements for an efficient area-time architecture are as shown in Table 5.3. This design (real-part) requires $2K$ Multipliers to compute an element every cycle and $(2K - 1)$ 16-bit adders for recursive doubling. This design requires about 10,000 Full Adder Cells and finds the estimate in $2KN$ cycles.

5.5 Comparisons with DSPs

An architecture comparison of the different VLSI architectures with a DSP is evaluated in this section. Though DSPs and GPPs with VLIW architectures and MMX-like instruction sets can exploit byte-length parallelism, they are less efficient while working with bit level parallelism. Storage of these bits on such a processor is either

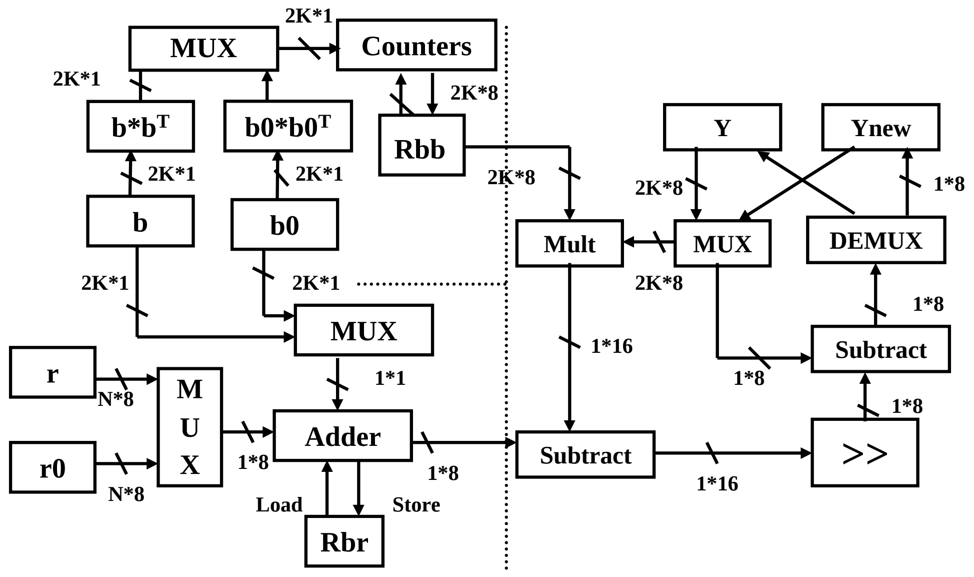

Figure 5.6 : Area-Time Efficient VLSI Architecture

inefficient as it is stored as bytes or clearly it represents a large overhead involved in packing and unpacking these bits. Also, the compiler may not take advantage of the fact that most of the multiplications are with bits and replace them with additions. Using a control structure instead also limits the utilization of available parallelism. Also, formation of bit-level matrix updates as seen in the different VLSI architectures is much more effective and simpler to build in hardware with EX-NOR gates, giving $O(1)$ performance with $O(K^2)$ or 1000 EX-NOR gates, while it may take $O(K^2)$ or 1000 cycles on a DSP and takes $O(K^2)$ or 1K bytes in memory.

Assuming a 500 MHz clock for the VLSI architectures, the projected time required to compute the channel estimate along with the hardware required for 32 users and a spreading code of length 32 is as shown in Table 5.4. This is compared with the implementation of the previously existing algorithm in Chapter 2, on a TI TMS320C6701 Evaluation Module, operating at 166 MHz. The DSP implementation of the Multiuser Channel estimation algorithm using the previously existing schemes is shown to require 763401 cycles (Chapter 3, Table 3.1), which corresponds to 4.56 ms for 15 users. Assuming that the channel estimate is updated for every block of 10 bits, and extending it linearly to 32 users, this corresponds to a time requirement of 0.97 ms or 1.02 Kbps. This is shown in Table 5.4.

The inherent parallelism present in the algorithm can be seen from the ratio of time taken for computation by the area-constrained and time-constrained architectures. The area estimates are compared using the number of Full Adder Cells needed in the design, as shown in Table 5.4. The time difference between the DSP and the VLSI architectures is due to the improvements in the algorithm modifications and the fact that the bit-level and byte-level parallelism are not exploited on the DSPs and the additional memory references. The difference in the processor speed does not play a major role in the time differences. We can observe that the area-constrained architecture does not satisfy real-time constraints of 7.8125 μs while the time-constrained architecture is far too aggressive. The area-time efficient architecture meets the next generation real-time constraints by designing the area-time tradeoff in 4 μs, which is close to the target data rate of 128 Kbps. From Table 5.3, it is seen that the time required is directly proportional to the number of users (K) in the system and the spreading factor (N), which are also dependent on each other as seen from Table 2.2. Hence, the system design also meets real-time requirements for various data rates, such as 1 Mbps for 4 users with a spreading factor of 4.

Blocks	**Quantity**	**Full Adder Cells**	**Complex**	**Total**
Counter	1*8	8	-	8
Multiplier	1*8	64	*2	128
Adders	$3 * 8 + 2 * 16$	56	*2	112
Total Full Adder Cells				248

Elements	**Memory/Reg Usage**	**Complex**	**Total**
b,b0	$4K * 1$	-	$8K$
r,r0	$N * 8$	*2	$32N$
R_{bb}	$2K^2 * 8$	-	$16K^2$
$R_{br}, Y, Ynew$	$2KN * 8$	*2	$96KN$
Net Memory Reqd. (in Bits) N=K=32			112,000

Total Time (Cycles)	$4K^2N$	128,000

Table 5.1 : Hardware Requirements for an Area-Constrained Architecture

Blocks	**Quantity**	**Full Adder Cells**	**Complex**	**Total**
Counter	$2K^2 * 8$	$16K^2$	-	$16K^2$
Multipliers	$4K^2N * 8$	$256K^2N$	*2	$512K^2N$
Adders	$2KN * 16 + 2KN * 8$	$48KN$	*2	$96KN+$
	$+4K^2N * 16$	$+64K^2N$		$128K^2N$
Total Full Adder Cells N=K=32				20,000,000

Elements	**Memory/Reg Usage**	**Complex**	**Total**
b,b0	$2K * 1$	-	$4K$
r,r0	$N * 8$	*2	$32N$
Y	$2KN * 8$	*2	$32KN$
Net Memory Reqd. (in Bits) N=K=32			32,000
Total Time(Cycles)		$\log_2(2K)$	6

Table 5.2 : Hardware Requirements for a Time-Constrained Architecture

Blocks	**Quantity**	**Full Adder Cells**	**Complex**	**Total**
Counter	$2K$*8	$16K$	-	$16K$
Multipliers	$2K$*8	$128K$	*2	$256K$
Adders	$2K*16+1*16+2*8$	$32K+32$	*2	$64K+64$
Total Full Adder Cells N=K=32				10,000
Elements		**Memory/Reg Usage**	**Complex**	**Total**
b,b0		$4K*1$	-	$8K$
r,r0		$N*8$	*2	$32N$
R_{bb}		$2K^2*8$	-	$16K^2$
$R_{br}, Y, Ynew$		$2KN*8$	*2	$96KN$
Net Memory Reqd. (in Bits) N=K=32				112,000
Total Time(Cycles)			$2KN$	2,000

Table 5.3 : Hardware Requirements for Area-Time Efficient Architecture

Architecture	**Full Adder Cells**	**Memory (Bytes)**	**Time**	**Data Rates**
Area-Constrained	248	16 KB	0.262 ms	3.81 Kbps
Time-Constrained	20,000,000	4 KB	12 ns	83.33 Mbps
Area-Time	10,000	16 KB	4 μs	256 Kbps
TMS320C6701 DSP	-	128 KB	0.97 ms	1.02 Kbps
Real-Time Requirements			7.8125 μs	128Kbps

Table 5.4 : Comparisons between Different Channel Estimation Architectures

Chapter 6

Architecture and Extensions for DSPs and GPPs

The DSP and the VLSI implementations of the algorithms in the previous chapters are used to find extensions for DSPs and GPPs in order to accelerate their performance for wireless communication algorithms. In this chapter, a new processor architecture with reconfigurable logic support is developed and extensions are suggested, which would greatly enhance DSPs and GPPs for wireless applications.

The reason for multimedia support in recent GPPs and DSPs such as the Sun UltraSPARC, Pentium MMX, TI TMS320C64x is to accelerate implementation of image and video processing algorithms. The acceleration was obtained by observing the fact that multimedia data is typically 8-bit wide (pixels) and the 64-bit wide databus and ALU could perform 8 operations on multimedia data in parallel by using SIMD (Single Instruction Multiple Data) parallelism. The recent explosion in wireless communications and their potential implementation in GPPs and DSPs behooves us to seek performance acceleration extensions for wireless communications by exploiting the potential for bit level arithmetic.

6.1 Features of Wireless Communication Algorithms

The implementation of the channel estimation and detection algorithms reveals the following features in algorithms for wireless communications and explains the limitations of current DSP and GPP architectures for these algorithms.

- **Bit Level Computations**

  A common feature among various algorithms for communication and as seen in the channel estimation and detection algorithms is the computations on bit-level data. Even other applications such as Huffman encoding in MPEG-4 requires computations on bit-level accesses. Current GPPs and DSPs which cannot operate directly on bit-level data need to do packing and unpacking to store the bits in memory; operate bit-level data in byte-wide ALUs and use 8-bit registers to operate data in the register files of the processor. Hence, bit-level support is needed in all parts of the processor: bit-level accesses to memory, registers that can process bit streams efficiently and ALUs which recognize operations on bits. Also, note that multiplications by bits can be replaced by additions if the bit is +1 or subtraction if the bit is -1. However, replacing them with a control if-else structure restricts the parallelism that can be exploited on DSPs and GPPs, especially if they have a VLIW architecture. Hence, proper hardware or software support is also needed for finding multiplications for bits and replacing them with additions and subtractions.

- **Matrix Based Operations**

  Another common feature in most wireless communication and image processing algorithms is that the operations performed on matrices. This implies that massive parallelism is available and memory-intensive operations. Current DSP and GPP implementations are unable to exploit the entire parallelism available in the algorithms and hence, unable to meet the MIPS or FLOPS requirements. Since operations on matrices are memory-intensive, the memory should be made insensitive to the strides of the data access, especially if matrix transpositions need to be calculated as accessing memory in column major order could potentially mean hits to the same bank. Hence, a stride insensitive memory system with a large bandwidth is needed.

- **Complex-Valued Arithmetic**

  Operations on complex-valued arithmetic is also common in many wireless and DSP algorithms. There have been DSP and GPP instructions for aligning and shuffling the real and imaginary parts [31] in computations such as the FFT. However, typically the support for complex arithmetic is done in software and the real and imaginary parts are computed in parallel in hardware.

- **Approximate Computations**

  There exists a class of arithmetic techniques that deal with the implementation of adders and multipliers based on the precision required to compute them. It has been shown [30] that typically savings by half the area and time delay can achieved for a multiplier if only the most significant digits need to be precise. This is particularly suited in DSP and wireless algorithms, which are iterative based and need the output of the multiplier to be fed back to the input, by throwing away the least significant bit values. An example for this would be the channel estimation scheme, which uses an iterative technique to compute the channel estimate. This truncated multiplier could be added to a GPP or DSP core and applications, which need not need higher precision, could use this multiplier to perform 2 multiplies in the time taken by the normal multiplier of the GPP or DSP core.

6.2 Architecture Design with Reconfigurable Support

6.2.1 Need for Reconfigurable Support

There have been different standards proposed for different environments for mobile communications, such as wireless LAN using OFDM (Orthogonal Frequency Division Multiplexing) in the office environment, W-CDMA in the mobile environment and Home Networks using Bluetooth. The RENE project at Rice [32] addresses the issue of developing a network interface, which will integrate all these standards and provide uninterrupted services between environments. Different services such as voice and multimedia have different protocols such as H.723 for voice and MPEG-4 for multimedia. Also, different environments may require different types of channel coding such as convolutional codes for the indoor environment and turbo codes for the mobile. Hence, algorithms also need to be reconfigured according to the standards, the services used and the environment. A reconfigurable architecture, which has a fast configuration time to switch between the standards and algorithms and deal with the initial protocols, is an efficient way to provide the network interface.

6.2.2 Architecture Design

The architecture design of the processor along with the reconfigurable support is as shown in Figure 6.1. This architecture design is adapted from the GARP architecture at University of California, Berkeley [33], though for a different application. The RF Unit converts the received signal to baseband and sends it to the processor. The RF unit is implemented as a PCMCIA network interface card and can be used for all the standards. The reconfigurable logic is on the same die as the processor core and acts as a co-processor to the main DSP or GPP core.

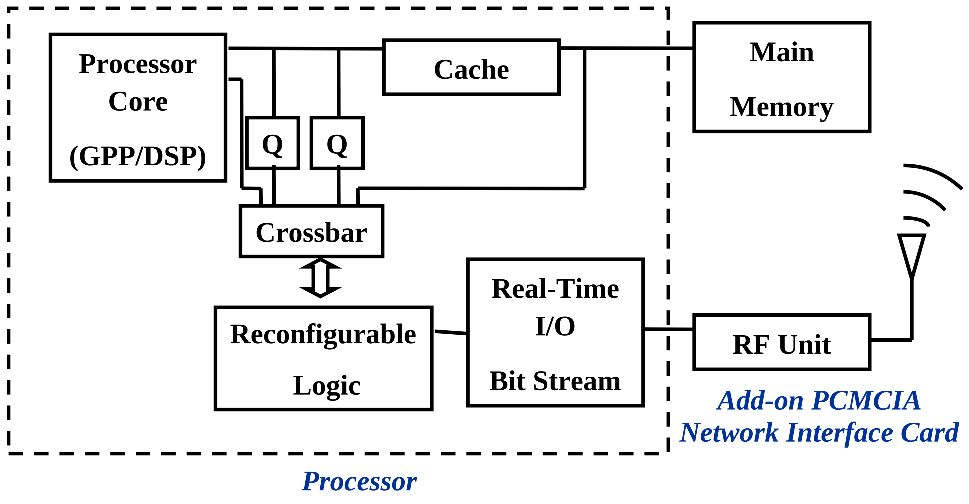

Figure 6.1 : Architecture Design with Reconfigurable Support

- **Reconfigurable Hardware**

The organization of the reconfigurable hardware is similar to the one proposed in GARP [33], except that it has also been optimized for fast data transfer between the architecture and the RF Unit to provide support for extremely high data rates (up to 2 Mbps). It is used primarily for fast I/O transfer and to process initial bit level computations which can be done more efficiently on the reconfigurable hardware. The reconfigurable hardware has a large datapath to memory to minimize load times and for fast data transfer. The reconfigurable hardware also has configuration caches, which stores the most recently displaced configurations. The configuration caches store the information for different environments. Up to 4 full size configurations can be stored and can be reprogrammed in 5 cycles. This fast reconfiguration time helps to provide seamless and uninterrupted service during transfers from one environment to another.

The reconfigurable hardware can execute independently, once configured and has access to the same memory system as the processor core, and hence there is no overhead involved in transferring data from the reconfigurable logic to main memory. When idle, the reconfigurable logic can transfer data to the processor and can be reconfigured via co-processor move instructions.

- **Memory Interface**

  The memory interface is designed to have a common access to the L1 data cache and the main memory to both the processor core and the reconfigurable logic. This avoids overhead in communication between the reconfigurable logic block and the memory via the processor. Memory Prefetch Queues are added for prefetching sequential data, especially by noting the fact that lot of operations are being performed on matrices. This allows for memory read-aheads and delayed write-backs.

- **Permutation Based Interleaved Memory**

  Since the operations are memory-intensive, a high memory bandwidth is required. The matrix-based operations, which involve both row and column major order access during transpositions. Also, the strides of the data access depend on the size of the matrices. To ensure that the stride pattern does not have multiple bank hits to a single bank, the system designed has to be stride-insensitive. It has been shown [34] that a permutation based interleaved (PBI) memory system can essentially randomize accesses and provide a throughput greater than 95%.

6.3 Instruction Set Extensions for Wireless

In this section, instruction set extensions based on bit level computations are proposed to accelerate the implementation of wireless communication algorithms. The multiuser detection and decoding algorithms involve integer-bit multiplications while algorithms for channel estimation involve bit-bit multiplications for outer product updates involved in auto-correlation. The extensions proposed here can also be used for bit level computations in similar algorithms for speech, video and image processing.

The extensions proposed involve no modifications to the architecture, except for the ALU. A few special purpose 8-bit registers with bit level access are also needed.

6.3.1 Reference Architecture

For comparison purposes, a general purpose processor with multimedia extensions such as the UltraSPARC with the VIS instruction set [35] or the Intel MMX Architecture [36]. Since different architectures have various types of ISA multimedia extensions, we assume a 64-bit architecture that supports 8-bit multiplications and additions. Specifically, we assume that we can perform eight 8-bit additions with 8-bit results and four 8-bit multiplications with 16-bit results simultaneously using SIMD parallelism. The latencies of the addition and multiplication operations and memory references, which are used in the following examples, are as shown in Table 6.1. All data is assumed to be in the internal registers for convenience in comparisons.

We now propose bit level extensions to the above instruction set architecture to accelerate wireless communication algorithms.

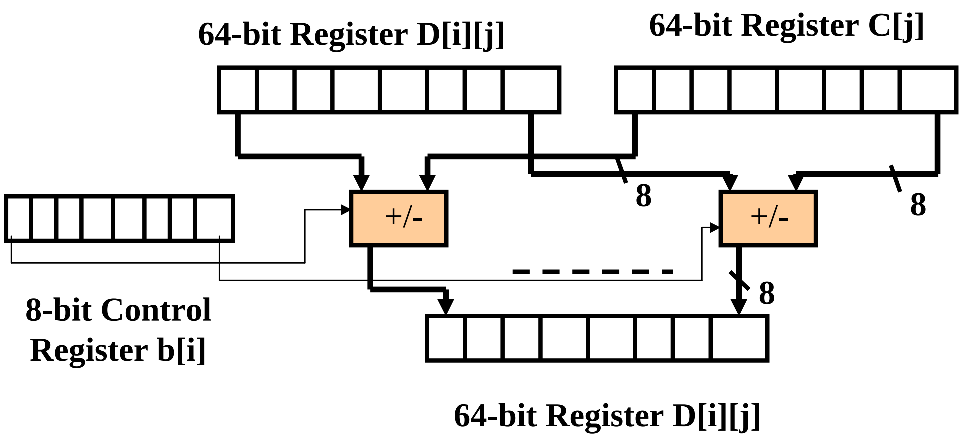

Figure 6.2 : 8 Integer - Bit Multiplications in Parallel

6.3.2 Integer-Bit Multiplications

The instruction set extension for performing integer-bit multiplications is as shown in Figure 6.2. Integer - bit multiplications are common in operations such as cross-correlation. This can be illustrated with the help of an example for cross-correlation. Consider

$$\begin{aligned} for..i &= 1:8 \\ for..j &= 1:8 \\ D[i][j] &= D[i][j] + b[i] * r[j] \end{aligned}$$

Operations	Latency
Four 8-bit Multiply	3
Eight 8-bit Add	1
Eight 8-bit Inc./Dec.	1

Table 6.1 : Latencies for various operations

where D and r are short integers(8 bits) and b is bit-wide (+1 or -1 represented as 1 or 0). If the bit b[i] is replicated and stored in the special purpose 8-bit register, 8 operations can be performed in parallel in a single cycle, based on the sign of the bit b[i]. This is in contrast to current DSP or GPP architectures without this extensions, where this operation would be a branch for addition or subtraction based on the value of the bit b[i] and limits exploiting the available parallelism. Note that the operation has been made more general to accommodate for multiplications such as b[j]*r[j], and hence the single bit b[i] has been replicated in this example. Thus, the branching can be avoided and 8 operations can be performed in parallel in a single cycle. This also helps in removing overheads involved in packing and unpacking of bits in the operation, as can be seen for the b[j]*r[j] case, where the bits can be utilized directly. Thus, the multiplication by bit vector b is done without any additional cycles as if it were absent compared to normal SIMD parallelism.

Since there are 64 multiplications and 64 additions in the above equation for cross-correlation, there are effectively 16 multiplication operations and 8 addition operations, as 4 multiplications and 8 additions can be done in parallel. This gives the cycle count as 16*3 + 8*1 = 54 cycles on an usual SIMD machine. However, this requires only 8*1 = 8 cycles using the proposed instruction as the bit multiplications can be avoided.

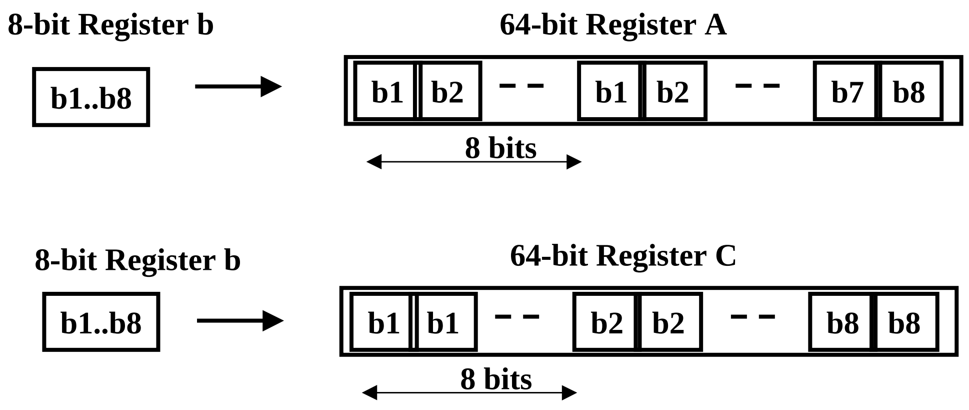

Figure 6.3 : 8 to 64 bit conversions

6.3.3 Bit-Bit Multiplications

Bit-Bit multiplications are commonly used in auto-correlation of matrices. Let us illustrate this with the help of another example. Consider

$$\begin{aligned} for..i &= 1:8 \\ for..j &= 1:8 \\ D[i][j] &= D[i][j] + b[i] * b[j] \end{aligned}$$

where D is a short int (8 bit) and b is bit-wide. To make optimum use of the hardware and to make the operations more general and RISC-like, the computations are separated into 3 parts: packing bits efficiently in a 64-bit register, 64 1 bit-bit multiplications, incrementing or decrementing the integer matrix based on the bit value.

1. **Packing bits efficiently in a 64-bit register**

   To perform the auto-correlation and obtain 64 1-bit products b[i]*b[j], the 8 1-bit sequence b is arranged as shown in Figure 6.3. All b[i]'s needed in the

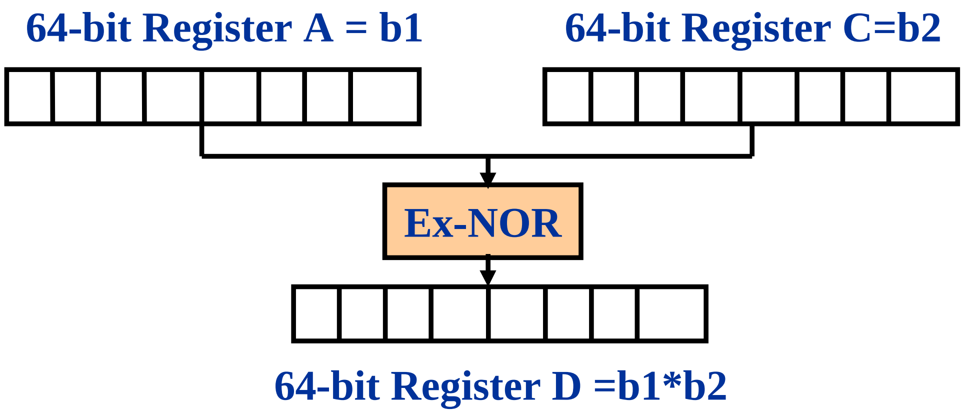

Figure 6.4 : 64 1-bit Multiplications in Parallel

product b[i]*b[j] are stored in 1 64-bit register and all b[j]'s are stored in another 64-bit register.

2. **64 1 Bit-Bit Multiplications in Parallel**

   Multiplication between bits, which are +1 and -1 is essentially an EX-NOR operation, thus the entire product matrix b[i]*b[j] can be computed in a single cycle with a single EX-NOR operation. Current processors have instructions for EX-OR and NOT and hence, these instructions could be used or another instruction could be built for EX-NOR. Note that the operation is made general such that any 2 64 1-bit numbers can be multiplied as shown in Figure 6.4.

3. **Incrementing or Decrementing based on the bit value**

   The auto-correlation matrix can be updated with 8 operations in parallel in a single cycle as shown in Figure 6.5. Thus, the entire matrix can be updated in 8 cycles. Again, the instruction has been generalized so that any eight 8-bit numbers can be incremented or decremented based on the value of 8 1-bits.

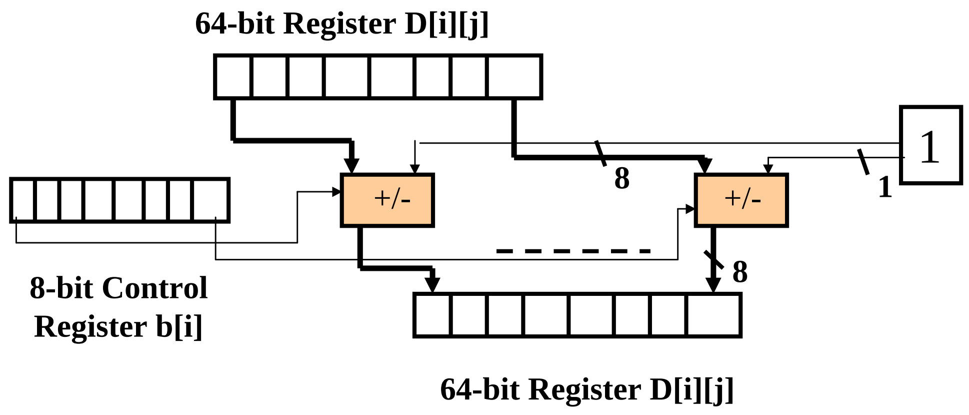

Figure 6.5 : 8 8-bit Increments/Decrements in Parallel

Thus, the entire auto-correlation consisting of 64 1-bit multiplies and 64 8-bit adds or subtracts can be achieved in 1+1+8 = 10 cycles (1 for conversion, 1 for EX-NOR and 8 for addition/subtraction).

The instruction set extensions show that significant savings in performance for algorithms for wireless communications can be obtained by supporting bit level computations. The proposed instructions do not require significant changes to the ALU design and are easy to support. They are also generalized to be useful to a wide variety of signal processing and communication applications that use bit-level computations. For measuring accurate performance improvements, the proposed changes need to be incorporated in a simulator and tested on a wide range of algorithms.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis demonstrates the use of computationally efficient algorithm and architecture design in order to meet real-time requirements for future wireless base-station receivers. Two of the main compute-intensive algorithms for the baseband layer of the base-station, Multiuser channel estimation and detection, are implemented on DSPs for performance evaluation. A design methodology using task partitioning is shown to meet the real-time requirements. However, this requires $O(K^2)$ elements. Hence, the multiuser channel estimation and detection algorithms are also modified for a more compute-effective solution, without compromising on the error rate performance of the algorithms. An area-time efficient VLSI architecture of the channel estimation scheme is designed to meet real-time requirements with minimum area overhead. Different area-time tradeoffs are investigated for channel estimation. An area-constrained architecture is designed that can be implemented on a FPGA. A time-constrained architecture design shows the extent of parallelism in channel estimation and determines the maximum data rates that can be achieved. An general purpose or DSP-based architecture design with reconfigurable support is developed to support different wireless communication standards. Several enhancements for general purpose processors and DSPs are proposed to accelerate their performance for wireless communication algorithms.

7.2 Future Work

7.2.1 Online Arithmetic

Online arithmetic is an unconventional technique for arithmetic intensive operations, where the computations are overlapped with the digit-by-digit communication of the operands and the results. The computations are carried out on a Most Significant Digit First (MSDF) mode of operations, where the MSB is computed ahead of the other bits. This requires the use of redundancy-based number systems in order to implement the operations online [37].

The advantages of using online techniques in wireless communication applications stems from the fact that the final goal of the computations is just to find whether the bit of the user is +1 or -1. If an online technique is applied in these algorithms, the operations on all subsequent bits after the MSB can be avoided, resulting in huge performance improvements. However, this requires the use of conversion between the normal arithmetic number system and a redundancy-based number system and the overhead and gains achieved need to be studied.

7.2.2 Multiprocessing with DSPs and FPGAs

As task partitioning is shown to be effective in meeting real-time requirements for these massively parallel algorithms, an implementation of these algorithms on multiple DSPs and FPGAs is being studied. The idea is to use the DSPs for the core operations such as multiplications and additions and to use the FPGAs for the bit-level support and operations.

Bibliography

[1] Erik Dahlman, Bjorn Gudmundson, Mats Nilsson, and Johan Scold, "UMTS/IMT-2000 Based on W-CDMA," *IEEE Communications Magazine*, vol. 36, no. 9, pp. 70–80, September 1998.

[2] M.Zeng, A.Annamalai, and Vijay K. Bhargava, "Recent advances in Cellular Wireless Communications," *IEEE Communications Magazine*, vol. 37, no. 9, pp. 128–138, September 1999.

[3] Suman Das, Joseph R. Cavallaro, and Behnaam Aazhang, "Computationally Efficient Multiuser Detectors," in *Eighth IEEE International Conference on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Helsinki, Finland, September 1997, pp. 62–67.

[4] Chaitali Sengupta, Suman Das, Joseph R. Cavallaro, and Behnaam Aazhang, "Efficient Multiuser Receivers for CDMA Systems," in *IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, September 1999, pp. 1461–1465.

[5] Sridhar Rajagopal, Srikrishna Bhashyam, Joseph R. Cavallaro, and Behnaam Aazhang, "VLSI Architectures for Multiuser Channel Estimation in W-CDMA Communication Systems," Tech. Rep. TREE0003 http://www.ece.rice.edu/~sridhar/research/tree0003.ps, Rice University, March 2000.

[6] Suman Das, Sridhar Rajagopal, Chaitali Sengupta, and Joseph R. Cavallaro, "Arithmetic Acceleration Techniques for Wireless Communication Receivers," in *33rd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, October 1999, pp. 1469–1474.

[7] Sridhar Rajagopal, Srikrishna Bhashyam, Joseph R. Cavallaro, and Behnaam Aazhang, "Efficient VLSI Architectures for Baseband Signal Processing in Wireless Base-Station Receivers," in *To appear in IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, Boston, MA, July 2000.

[8] Fumiyuki Adachi, Mamoru Sawahashi, and Hirohito Suda, "Wideband DS-CDMA for Next-Generation Mobile Communication Systems," *IEEE Communications Magazine*, vol. 36, no. 9, pp. 56–69, September 1998.

[9] Tero Ojanpera and Ramjee Prasad, Eds., *Wideband CDMA for Third Generation Mobile Communications*, Artech House Publishers, 1998.

[10] Tero Ojanpera and Ramjee Prasad, "An Overview of Air Interface Multiple Access for IMT-2000/UMTS," in *IEEE Communication Magazine*, vol. 36, pp. 82–95. September 1998.

[11] "Third Generation Partnership Project," http://www.3gpp.org.

[12] Sergio Verdú, "Minimum Probability of error for asynchronous Gaussian Multiple-access Channels," *IEEE Transactions on Information Theory*, vol. IT-32, no. 1, pp. 85–96, 1986.

[13] Fumiyuki Adachi, "BER Analysis of 2PSK, 4PSK, and 16QAM with Decision Feedback Channel Estimation in Frequency-Selective Slow Rayleigh Fading," *IEEE Transactions on Vehicular Technology*, vol. 48, no. 5, pp. 1563–1572, September 1999.

[14] Srikrishna Bhashyam and Behnaam Aazhang, "Multiuser Channel Estimation for Long Code CDMA Systems," Submitted for Publication for Wireless Communication and Networking Conference (WCNC), September 2000.

[15] Gang Xu and Joseph R. Cavallaro, "Real-time Implementation of Multistage Algorithm for Next Generation Wideband CDMA Systems," in *Advanced Signal Processing Algorithms, Architectures, and Implementations IX, SPIE*, Denver, CO, July 1999.

[16] Chaitali Sengupta, Joseph R. Cavallaro, and Behnaam Aazhang, "Maximum Likelihood Multipath Channel Parameter Estimation in CDMA Systems using Antenna Arrays," in *9th IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Boston, MA, September 1998, pp. 1406–1410.

[17] Mahesh K. Varanasi and Behnaam Aazhang, "Multistage detection in asynchronous Code-Division Multiple -Access communications," *IEEE Transactions on Communications*, vol. 38, no. 4, pp. 509–519, April 1990.

[18] Shimon Moshavi, "Multi-User Detection for DS-CDMA Communications," *IEEE Communications Magazine*, pp. 124–136, October 1996.

[19] Suman Das, Elza Erkip, and Behnaam Aazhang, "Computationally Efficient Iterative Multiuser Detection and Decoding," in *Thirty Second Annual Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, November 1998, pp. 631–634.

[20] Texas Instruments, *TMS320C6x Evaluation Module : Reference Guide*, TI, February 1998.

[21] Texas Instruments, *TMS320C62x/C67x CPU and Instruction Set : Reference Guide*, TI, March 1998.

[22] Willam Press, Brian Flannery, Saul Teukolsky, and William Vetterling, *Numerical Recipes in C*, Cambridge University Press, 1991.

[23] Texas Instruments, *TMS320C6x Optimizing C Compiler : User's Guide*, TI, February 1998.

[24] Texas Instruments, *TMS320C62x/C67X : Programmer's Guide*, TI, February 1998.

[25] Texas Instruments, "C6000 Compiler Optimization Tutorial," http:// www.ti.com/ sc/docs/tools/dsp/ccstutor.htm.

[26] Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, chapter 10, pp. 520–521, John Hopkins University Press, third edition, 1996.

[27] Markku J. Juntti and Behnaam Aazhang, "Linear Finite Memory-Length Multiuser Detectors," in *IEEE Global Communication Conference (GlobeCom)*, Singapore, November 1995, pp. 126–130.

[28] Seehyun Kim, Ki-Il Kum, and Wonyong Sung, "Fixed-Point Optimization Utility for C and C++ Based Digital Signal Processing Programs," in *IEEE Transactions on Circuits and Systems-II:Analog and Digital Signal Processing*, November 1998, vol. 45, pp. 1455–1464.

[29] Neil H.E. Weste and Kamran Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*, chapter 8, Addison-Wesley, second edition, 1993.

[30] Michael J. Schulte and Earl E. Swartzlander, "Truncated Multiplication with Correction Constant," in *Workshop on VLSI Signal Processing*, Veldhoven, Netherlands, October 1993, vol. VI, pp. 388–396.

[31] Craig Hansen, "MicroUnity's MediaProcessor Architecture," *IEEE Micro*, pp. 34–41, August 1996.

[32] "RENE: Seamless Multitier Wireless Networks for Multimedia Applications," Rice University, http://cmc.rice.edu/research/RENE.

[33] T.C.Callahan, J.R.Hauser, and J.Wawrzynek, "The GARP Architecture and C Compiler," *IEEE Computer*, pp. 62–69, April 2000.

[34] G.S.Sohi, “High-Bandwidth Interleaved Memory for Vector Processors - A Simulation Study,” *IEEE Transactions on Computers*, vol. 42, no. 1, pp. 34–44, January 1993.

[35] Marc Trembley, J. Micheal O’Connor, Venkatesh Narayan, and Liang He, “VIS Speeds New Media Processing,” *IEEE Micro*, p. August, 10-20 1996.

[36] Alex Peleg and Uri Weiser, “MMX Technology Extension to the Intel Architecture,” *IEEE Micro*, pp. 42–50, August 1996.

[37] M.D.Ercegovac and T.Lang, “Fast Arithmetic For Recursive Computations,” in *Workshop on VLSI Signal Processing*, Napa Valley, CA, October 1992, vol. V, pp. 14–28.