Arithmetic Acceleration Techniques for Wireless Communication Receivers

Suman Das   Sridhar Rajagopal   Chaitali Sengupta*   Joseph R. Cavallaro
Rice University
Center for Multimedia Communication
Department of Electrical and Computer Engineering MS-366
6100 Main St., Houston, TX 77005-1892.
{suman,sridhar,chaitali,cavallar}@rice.edu

Abstract

We develop techniques to accelerate the implementation of the next generation wireless communication algorithms in hardware. We discuss an implementation of a key computationally intensive baseband algorithm for joint multiuser channel estimation and detection for this purpose and study its real-time requirements. An analysis of the bottlenecks present in the algorithm is made. We present an acceleration technique using task decomposition to take advantage of the existing pipelining and parallelism flow in the algorithm. We show that an application specific system design with multiple processing elements is more effective than the conventional single processor approach as it can satisfy the high data rate requirements of the next generation wireless communication systems. Our analysis is done independent of the final mapping of the processing elements in hardware.

1. Introduction

The next generation wireless communication receivers are being designed to support enhanced features such as multimedia capabilities, higher data rates, Quality of Service (QoS) and multi-rate services in the existing wireless communication framework. Wideband Direct Sequence Code Division Multiple Access (W-CDMA) is the chosen multiple access protocol for these communication systems [1]. Support for high data rates [4, 5] such as 2 Mbps indoor, 384 Kbps pedestrian and 144 Kbps vehicular traffic is to be provided by the base-station receivers.

Several algorithms are being developed for the different functional blocks of the next generation base-station receiver. These algorithms have extremely high complexity and they put increased pressure on the next generation receivers, which have stringent time, power and size constraints.

One of the main computational bottlenecks in the base-station receiver is the estimation and detection of the transmitted bits from the received signal. Multiuser channel estimation and detection is being considered as part of the advanced receiver structures for next generation communication systems. For our evaluation purposes, we choose one of the designed computationally intensive algorithms for W-CDMA multiuser channel estimation and detection [7]. This algorithm is based on a joint multiuser estimation and detection technique which eliminates the need for conventional extraction of parameters in channel estimation and thus gives gains both in performance in terms of Bit Error Rates (BER) as well as computational complexity. This algorithm includes chip level despreading, which is one of the main bottlenecks in the receiver. An implementation of this algorithm on a Texas Instrument's C6x DSP Simulator[9], chosen as a example of current processor technology, does not meet real time constraints. We develop techniques to accelerate the implementation of this algorithm. A task decomposition of the algorithm is done to explore the data dependencies between the different tasks in the algorithm and take advantage of available pipelining and parallelism. The analysis also shows how frequently we can update the algorithm based on decision feedback.

The main contributions of this paper are twofold. First, we develop techniques to accelerate the implementation of the wireless communication algorithms on hardware, which is independent of the final hardware mapping of the algorithm. We show this specifically for the joint estimation and detection algorithm. Also, we show that an application specific approach of multiple processing elements is more effective than a single processor system in meeting the real-time constraints in the base-station receiver.

*Currently at Texas Instruments, Inc. Dallas, TX. This work was supported in part by Nokia Corporation, Texas Instruments Inc.,the Texas Advanced Technology Program under grant 1997-003604-044, and by NSF under grants NCR-9506681 and ANI-9979465

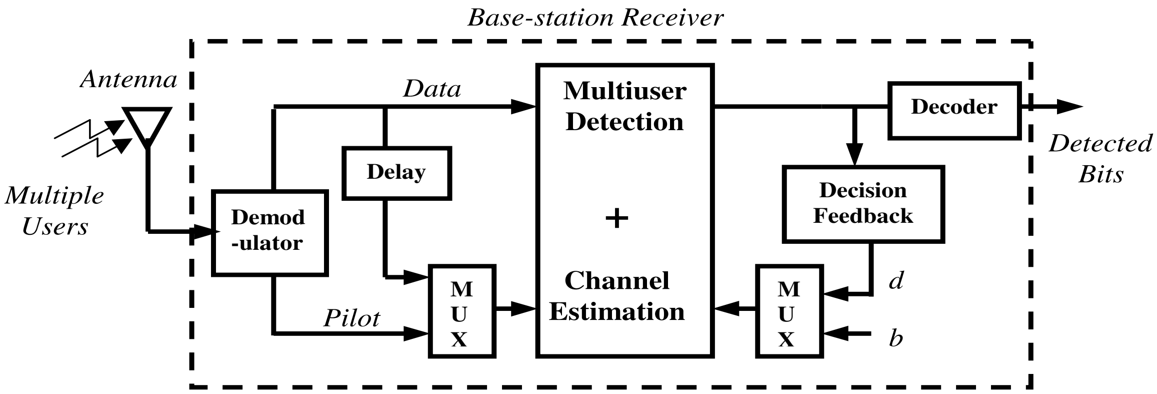

Figure 1. Simplified view of the Base Station Receiver

2. Joint Estimation and Detection

2.1. Channel Model

The channel estimation and detection block in the base-station receiver is shown in Figure 1. We assume an asynchronous Code Division Multiple Access (CDMA) based system with Binary Phase Shift Keying (BPSK) modulation, where the signal is multiplied with a short repeating spreading code before transmission. As the spread signal is sent through the channel, it experiences undesirable effects such as delays due to multiple paths, interference from other users, fading and noise. The detector needs to acquire synchronization with the input signal in order to correctly detect the incoming bit sequence. Hence, the parameters of the channel need to be estimated for proper detection. Channel estimation involves estimating and tracking the delays of each users' bits and the channel attenuation over the different paths. The channel estimation scheme uses the Maximum Likelihood principle [6] to estimate the channel parameters and directly feeds this information to the detector without actually extracting the channel parameters. The detection scheme is the Differencing Multistage Detection method [11], based on the principle of Parallel Interference Cancellation (PIC) [10]. The detector uses the information from the channel estimation block to remove the interference from other users.

The channel information is built on the basis of *a priori* information obtained by transmission of a pilot signal (b), which is a sequence of bits that is known at the receiver. The pilot signal received at the base-station (pilot), is compared with the known bits to form an estimate of the channel. The decisions from the multiuser detection block (d) are fed back to the channel estimation block along with the received data bits (data), delayed by the time required for detection, for tracking the algorithm when the pilot signal is absent.

2.2. Real-time Requirements

Data transmission in the next generation wireless systems [5] is done in frames of $10ms$. The data transmission can be done in variable rates depending on the spreading factors (SF), as shown in Table 1. The table gives an example of the number of bits in a frame for spreading factors of 4, 32 and 256. We assume BPSK modulation for our algorithm. To support real-time, the number of bits detected per frame should be at the rate of transmission. We choose a short Gold Code sequence of length 31 for our spreading (which matches nearest to the proposed spreading factor of 32). This implies that the real-time requirement of our joint estimation and detection scheme is to detect input data bits at a rate of 128 Kbps.

2.3. Computations Involved

The derivation of the joint estimation and detection algorithm is detailed in [7]. We include the algorithm here to explain the computational aspects involved without proof. The model for the channel can be expressed as

$$\mathbf{r_i} = \mathbf{A_i b_i} + \eta_\mathbf{i} \quad (1)$$

where $\mathbf{r_i} \in \mathbb{C}^N$ are the received bits of all K asynchronous users, spread with a spreading factor N, $\mathbf{b_i} \in \mathbb{R}^{2K} = [b_{1,i-1}, b_{1,i}, \ldots, b_{K,i-1}, b_{K,i}]^\top$ are the bits of K users to be

SF	Bits	Data Rates
4	10240	1 Mbps
32	1280	128 Kbps
256	160	16 Kbps

Table 1. Proposed Data Rates for Next Generation Communication Systems

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 computations that occur during the synchronization phase [7, 11] are:

$$\mathbf{R}_{br} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{b}_i \mathbf{r}_i^H \quad (2)$$

$$\mathbf{R}_{bb} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{b}_i \mathbf{b}_i^H \quad (3)$$

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

$$\mathbf{R}_{bb}\mathbf{A_i}^{\mathbf{H}} = \mathbf{R}_{br} \quad (4)$$

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,

$$\mathbf{r}_i = [\mathbf{A}_0|\mathbf{A}_1] \begin{bmatrix} b_{1,i-1} \\ \vdots \\ b_{K,i-1} \\ b_{1,i} \\ \vdots \\ b_{K,i} \end{bmatrix} + \eta_i . \quad (5)$$

It has been shown that detecting a block of bits simultaneously (multishot detection) can give performance gains [11]. Also, multishot detection is near-far resistant as it accounts for the interference from both the overlapping symbols of the interfering users. 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 ($i = 1, 2, \cdots, D$).

$$\mathbf{r} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & 0 & 0 \\ 0 & \mathbf{A}_0 & \mathbf{A}_1 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \mathbf{A}_0 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} b_{1,1} \\ \vdots \\ b_{K,1} \\ \vdots \\ b_{1,D} \\ \vdots \\ b_{K,D} \end{bmatrix} + \eta \quad (6)$$

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 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 is given by

$$\mathbf{y}^{(0)} = Re[\mathbf{A^H r}] \quad (7)$$

$$\mathbf{d}^{(0)} = sign(\mathbf{y}) \quad (8)$$

$$\mathbf{y}^{(1)} = \mathbf{y}^{(0)} - Re[\mathbf{A^H A} - \mathbf{S}]d^{(0)} \quad (9)$$

$$\mathbf{d}^{(1)} = sign(\mathbf{y}^{(1)}) \quad (10)$$

where $y^{(1)}$ and $d^{(1)}$ are the soft and hard decisions after the first stage of the joint detector and $S \in \mathbb{R}^{KD \times KD}$ is the diagonal elements in $\mathbf{A^H A}$. These computations are iterated $l = 1, 2, \cdots, M$ where M is the maximum number of iterations.

$$\mathbf{x}^{(l)} = \mathbf{d}^{(l)} - \mathbf{d}^{(l-1)} \quad (11)$$

$$\mathbf{y}^{(l+1)} = \mathbf{y}^{(l)} - Re[\mathbf{A^H A} - \mathbf{S}]x^{(l)} \quad (12)$$

$$\mathbf{d}^{(l+1)} = sign(\mathbf{y}^{(l+1)}) \quad (13)$$

The structure of $\mathbf{A^H A} \in \mathbb{C}^{\mathbf{KD \times KD}}$ is as shown

$$\begin{bmatrix} \mathbf{A_0^H A_1} & \mathbf{A_0}^{\mathbf{H}}\mathbf{A_1} & 0 & 0 \\ \mathbf{A_1}^{\mathbf{H}}\mathbf{A_0} & \mathbf{A_0}^{\mathbf{H}}\mathbf{A_0} + \mathbf{A_1}^{\mathbf{H}}\mathbf{A_1} & \mathbf{A_0}^{\mathbf{H}}\mathbf{A_1} & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \mathbf{A_1}^{\mathbf{H}}\mathbf{A_0} & \mathbf{A_1}^{\mathbf{H}}\mathbf{A_1} \end{bmatrix} \quad (14)$$

The hard decisions, $\mathbf{d}$, which are made at the end of the final stage, are fed back to the synchronization block and to the rest of the processing blocks in the receiver.

3. Task Decomposition and Implementation

The algorithm is implemented on a TI C6x DSP simulator [9], assuming 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 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 algorithm was written in C. The algorithm was written in a memory-efficient manner so as to avoid transposes and uses inplace computations. The entire code and data fits in the internal memory of the DSP. In this initial implementation, the LU decomposition was used to calculate (4). We use the TI C Compiler ver 3.0 to generate the assembly code for the DSP. The highest possible compiler optimizations recommended by TI [8] were used. The optimizations perform software pipelining, loop unrolling and other program level optimizations to exploit

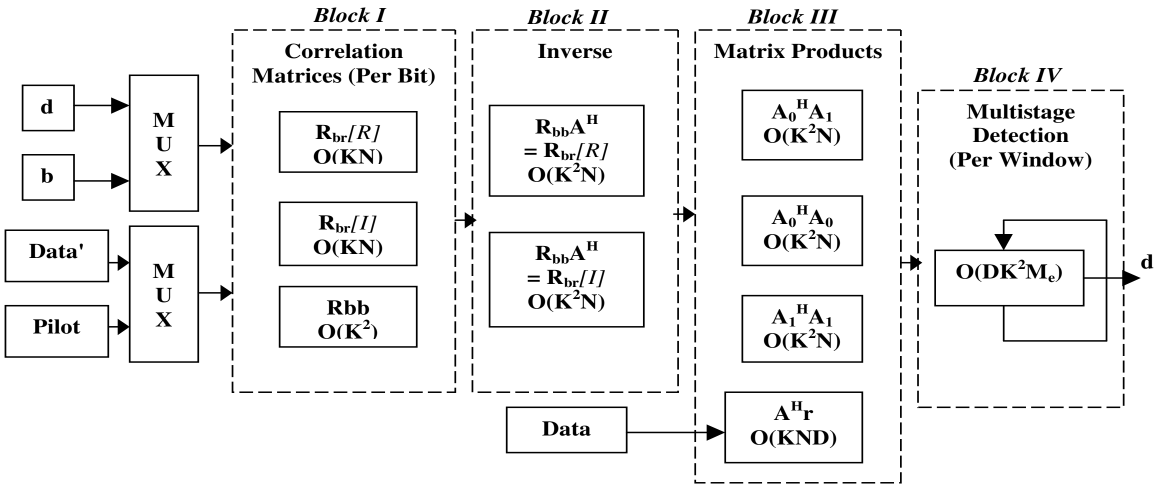

Figure 2. The Task Partition Graph for the Joint Estimation and Detection Algorithm

the available fine-grain parallelism available in the VLIW architecture. The structure and sparseness of the various matrices were also accounted for in the implementation.

3.1. Task Decomposition

The sequential implementation of the entire algorithm on the DSP does not meet real-time constraints. In fact, the achieved data rates for just the detection block implementation, assuming a single stage iteration, shows 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 is as shown in Figure 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 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.

Equations 2,3 are shown in block I as both $\mathbf{R}_{br}$ and $\mathbf{R}_{bb}$ 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. Equation 4 is represented in block II, where the inverse of $\mathbf{R}_{bb}$ is calculated by using a LU Decomposition. The block III shows the computation of equation 14. $\mathbf{A_1}^H\mathbf{A_0}$ is not computed as it is $(\mathbf{A_0}^H\mathbf{A_1})^H$. Block III also includes the computation of the Multiuser detection part (7), as it can be done in parallel with the above operations. The iterative loop of the Multistage Detection (8-13) is shown as a single block IV.

The input data bits are streaming in continuously in 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. (We neglect the effect of channel estimation on bit error rates for this purpose). The parts of multiuser detection which depend on the input data are the calculation of $\mathbf{A}^H\mathbf{r}$, as in (7), and the multistage detection loop (8-13). An order complexity analysis was also done on the algorithm to find the bottlenecks in each block.

Block	Complexity	Cycle count
Correlation Matrices	$KN + K^2$	27957
Inverse	K^2N	763401
$A_0^HA_1, A_0^HA_0, A_1^HA_1$	K^2N	124205
A^Hr	KND	132723
per bit	KN	13272
Multistage(1st stage)	DK^2	33669
per bit	K^2	3367

Table 2. Cycle count and Complexity for different blocks

3.2. Simulations and Analysis

An in-depth profiling of the various blocks was carried out using the clock function in the C6x DSP. The cycle count for the various blocks is as shown in Table 2 for 15 users and a detection window of length 12. Assuming a 250 MHz processor, a data rate requirement of 128 Kbps implies the available number of cycles per bit is 1953 for real-time detection. The successive stages in the Multistage detector take significantly less time than the first stage. Hence, let us assume the effective number of stages

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

Table 3. Cycle count for different optimization levels of blocks A ($\mathbf{A}^H\mathbf{r}$) and B (block IV)

as 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 $\mathbf{A}^H\mathbf{r}$. 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 $3367 * M_e + 13272$ cycles. Therefore, we need to search for more fine grain parallelism from the above task partition graph.

Table 3 shows the advantages of various levels of parallelism(Pl) and pipelining(Pp). Let A refer to the calculation of $\mathbf{A}^H\mathbf{r}$ in block III and B to block IV. Let (A + B Sequential) be the present solution obtained. If A and B were pipelined (A B), the required computation becomes the maximum of A and B. Next, $A^H r$ can be done for each user in parallel as each row of A^H corresponds to a user, reducing the time to 885 cycles. This puts the bottleneck to block B (Pl(A) B). Hence, block B is also unrolled into different stages. The first stage now has the most complexity, it becomes the new bottleneck, needing 3367 cycles ((Pl A) (Pp B)). It has been shown [11] that each successive stages in B requires less computation than the previous stage. Hence, fewer or less powerful processing elements need to be used to these stages. Each stage can also be split into multiple processing elements in a manner similar to A. This reduces the cycles needed to 225, putting the bottleneck back to A (Pl(A) PlPp(B)). A and B after this step are shown in Figure 3.

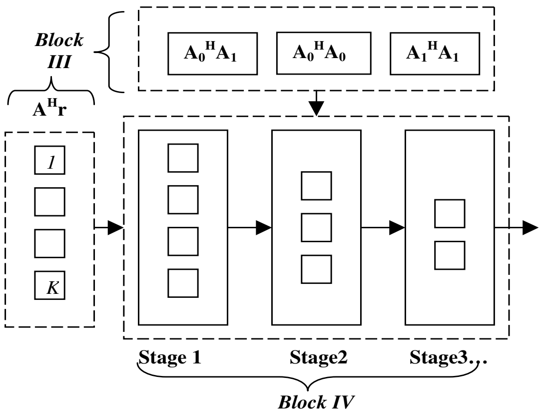

Figure 3. Further Pipelining & Parallelism in the Multistage Detection

4. Meeting Real Time Constraints

The data rates which can be met with different levels of pipelining and parallelism are as shown in Figure 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 lower 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, the bottleneck shifts back to 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.

Judging from the time requirements for the block I and block II, 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.

5. Hardware Mapping

The above analysis with task partitioning is independent of the final hardware mapping of the processing elements. We assume that the processing elements in the critical part (block IV and $\mathbf{A}^H\mathbf{r}$) are equivalent to the functional units in a C67 for that particular operation because the C67 is used as the basis for our timing results. The processing elements could be mapped to different architectures such as a single ASIC or multiple processors or a combination of a processor with an ASIC or FPGA. The mapping could also be done such as to have a DSP core with some coprocessor structures for critical parts. Also, if there exists many processing elements in parallel where a single element dominates the computation, such as block III where the time taken by $\mathbf{A}^H r$ takes the same time as the other 3 matrix products taken together (see Table 2), all those processing elements could be mapped to a single processor. Thus, the load between elements that have idle times could be dis-

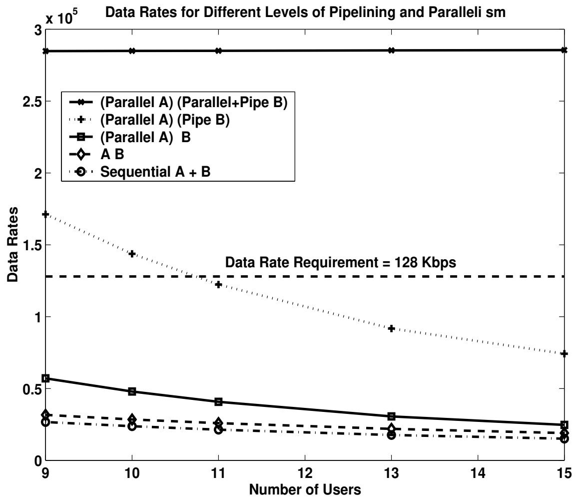

Figure 4. Data Rates for various levels of Pipelining and Parallelism for A ($\mathrm{A}^H\mathrm{r}$) and B (block IV)

tributed to other processing elements.

The other assumptions include ideal communication overhead, no restriction on the number of processing elements available and the feasibility of designing such an application specific system. The number of processing elements are dependent on the number of users (K), which is variable. Allocating elements for the maximum number of users may not lead to optimum utilization. Hence, reconfigurable architectures supporting varying number of users should also be considered.

6. Future Work

The dynamic range requirements for the joint estimation and detection algorithm are being analysed. In the initial version, the algorithm is implemented in floating point due to the possible loss in precision involved in LU decomposition. From the analysis of the differencing multistage detector[11], we expect a precision range of less than 24 bits. A fixed point implementation with a lower precision range could benefit from the VLIW and SIMD type of fine grained parallelism shown in recent DSP and general purpose architectures. Also, matrix oriented architectures [3], such as a vector processor with SIMD, showing 2 levels of parallelism could be beneficial to these applications. Another idea is to explore special systems to take advantage of the complex arithmetic data involved, such as using redundant complex number systems (RCNS) for a ASIC architecture [2].

7. Summary

We develop acceleration techniques to implement key computationally intensive baseband algorithms in hardware. The joint multiuser channel estimation and detection algorithm is considered for this purpose. A detailed task partition of this algorithm along with its complexity analysis is shown with the help of a C6x DSP simulator. The available parallelism and pipelined tasks in the algorithm are exploited to satisfy the real-time constraints. We discuss mapping issues of the task partitions in hardware. Such an application specific design with multiple processing elements is more effective than a single processor in meeting the real time requirements of next generation communication systems.

References

[1] F. Adachi, M. Sawahashi, and H. Suda. Wideband DS-CDMA for Next-Generation Mobile Communication Systems. *IEEE Communications Magazine*, 36(9):56–69, September 1998.

[2] T. Aoki, Y. Ohi, and T. Higuchi. Redundant Complex Number Arithmetic For High-Speed Signal Processing. *IEEE Workshop on VLSI Signal Processing VIII*, pages 523–532, October 1995.

[3] J. Corbal, M. Valero, and R. Espasa. Exploiting a New Level of DLP in Multimedia Applications. In *MICRO, International Symposium on Microarchitecture*, November 1999.

[4] E. Dahlman, B. Gudmundson, M. Nilsson, and J. Scold. UMTS/IMT-2000 Based on W-CDMA. *IEEE Communications Magazine*, 36(9):70–80, September 1998.

[5] M.Zeng, A.Annamalai, and V. K. Bhargava. Recent advances in Cellular Wireless Communications. *IEEE Communications Magazine*, 37(9):128–138, September 1999.

[6] C. Sengupta, J. Cavallaro, and B. 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)*, pages 1406–1410, September 1998.

[7] C. Sengupta, S. Das, J. Cavallaro, and B. Aazhang. Efficient Multiuser Receivers for CDMA Systems. In *IEEE Wireless Communications and Networking Conference, New Orleans, LA*, pages 1461–1465, September 1999.

[8] TI. C6000 Compiler Optimization Tutorial. http://www.ti.com/sc/docs/tools/dsp/ccstutor.htm.

[9] TI. *TMS320C6x Optimizing C Compiler*, chapter 6. Texas Instruments, February 1998.

[10] M. K. Varanasi and B. Aazhang. Multistage detection in asynchronous Code-Division Multiple -Access communications. *IEEE Transactions on Communications*, 38(4):509–519, Apr. 1990.

[11] G. Xu and J. Cavallaro. Real-time Implementation of Multistage Algorithm for Next Generation Wideband CDMA Systems. In *Advanced Signal Processing Algorithms, Architectures, and Implementations IX, SPIE*, 1999.