Implementation of Channel Estimation and Multiuser Detection Algorithms for W-CDMA on Digital Signal Processors

Sridhar Rajagopal, Gang Xu and Joseph R. Cavallaro
Center for Multimedia Communication
Department of Electrical and Computer Engineering
Rice University, Houston, TX 77251-1892.
{*sridhar,gxu,cavallar*}*@rice.edu*

ABSTRACT

Proposed algorithms for Third Generation W-CDMA communication systems have extremely high performance requirements. In this paper, we study the implementation issues involved for one of the proposed multiuser channel estimation and detection algorithms for base-stations in the uplink using TI's TMS320C6x DSP Evaluation Modules(EVM). It was found that these proposed algorithms for multiuser channel estimation and detection have different processing and precision requirements. While the detector can be implemented using the C6201 16-bit fixed point DSP, the proposed channel estimation algorithm may be more suitable for a floating point implementation using the C6701 floating point DSP. We study the effects of the specialized approximate instructions available on the C6701 DSP on channel estimation. Then, the advantage of multistep optimizations and use of assembly code is studied for both the algorithms. Memory issues involved in the implementation of these algorithms is also investigated. It was found that the data memory requirements for channel estimation for the chosen system parameters necessitates the use of external memory while the multistage detection algorithm could be placed in the available internal data memory. We finally discuss the current and future trends of DSPs and their utilization for such wireless communication applications.

1. INTRODUCTION

Wideband Direct-Sequence Code Division Multiple Access (W-CDMA) is the emerging protocol[1] for the next generation (3G) wireless communication systems. W-CDMA has been designed to add features such as multimedia capabilities, high data rates and multi-rate services to the existing wireless communication framework. The data rates proposed[2,3] are 2Mbps indoor, 384 Kbps pedestrian, and 144 Kbps vehicular. Several standards for third generation systems have been proposed and developed by different industrial committees in countries such as the U.S, Europe and Japan. All these standards have accepted CDMA in one form or another as the multiple access method for wireless communications.

One of the main bottlenecks in the base station receiver, both in terms of accuracy and speed, is the estimation and detection of the transmitted data from the received signal. Channel parameter estimation[4] is essential for finding the delays and fading amplitudes in the system. This information is fed to the multiuser detector and Rake receivers to detect the incoming data accurately. Channel estimation and detection processing[5] can done both on the mobile handset and the base station using DSPs or DSP cores with specialized blocks integrated with RF circuitry. Before the proposed advanced algorithms for W-CDMA can be integrated with the next-generation CDMA technology, they need to be tested and prototyped on such DSP processors to evaluate their performance. This is the main goal for our performance evaluation of multiuser channel estimation and detection algorithms on DSPs. The proposed channel estimation and detection algorithms have stringent time, power and size constraints and they put increased pressure on the existing resources and the available processor technology. As our analysis shows, the current trend in DSPs are showing encouraging signs in meeting the real-time requirements of these algorithms.

The organization of this paper is as follows. In section 2, we introduce the multiuser channel estimation and detection algorithms. In section 3, we describe the simulation methodology used for our performance evaluation. We also discuss the hardware used in our simulations. In section 4, we present the results obtained for both the algorithms on DSPs and their analysis. In section 5, we discuss the real-time requirements for channel estimation and detection for W-CDMA. We also discuss current and future trends in DSPs and their suitability for such wireless communication applications. We finally conclude our paper in section 6.

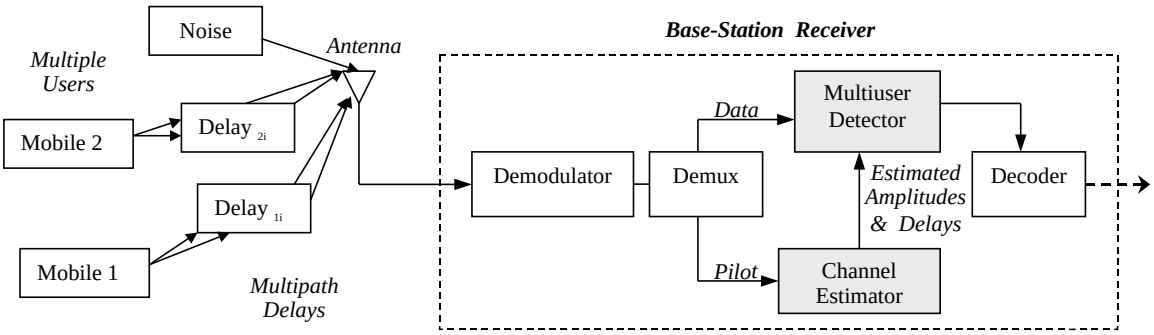


Figure 1. Simplified view of the Base Station Receiver for the Uplink

2. CHANNEL ESTIMATION AND DETECTION IN THE BASE STATION

The main blocks in a base station receiver are as shown in Figure 1. The channel is the wireless interface between the mobile user and the base station. Many undesirable effects such as interference from other users, delays from multiple paths, fading and noise occur on the signal as it passes through the channel. 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. One of the proposed methods for channel estimation is using the Maximum Likelihood method. There has been ongoing related research at Rice using the Maximum Likelihood algorithm for channel estimation.[6] This algorithm is designed to handle time variations in the system, multiple propagation paths, and large number of users with varying level of transmitting power.

In the uplink, since all users are transmitting information, each desired user experiences direct interference from other users (Multiple Access Interference or MAI). Also, signals from users near the base station tend to be stronger and overshadow the signals from users far away from the base station (near-far effect). The optimal multiuser detector was first proposed by Verdú,[7] but several sub-optimal schemes have been proposed to reduce the complexity of the algorithm. One of the most effective sub-optimal schemes, based on the principle of Parallel Interference Cancellation (PIC), was first proposed by Varanasi and Aazhang[8] at Rice. This scheme was the iterative multistage method where the inputs of one particular stage are the estimated bits of the previous stage. After interference cancellation, the new estimates, which should be closer to the transmitted bits, are output and fed into the next stage. Ideally at the last iteration stage, the output and the input should be identical if the algorithm converges. Further optimizations have been made on the algorithm, making use of the fact that as the iterations progress, the solution becomes more and more invariant, i.e. more and more elements in the output vector turn out to be the same as the elements in the input vector. The proposed detection algorithm is the Differencing Multistage algorithm,[9] which is based on the above principle.

2.1. System Model

Our proposed scheme for the uplink channel estimation makes use of a pilot or preamble (a sequence of bits known at the receiver), which is time-multiplexed with the data. As the signal passes through the channel, the channel causes changes in the preamble. By comparing the received bits with the known preamble, the channel parameters (delays and amplitudes) can be extracted from the signal. The channel parameters are assumed to remain static until the end of the frame. These estimates are then passed to the differencing multistage detector. The multistage detector does the interference cancellation stage by stage until the algorithm converges. The algorithms discussed here essentially consider multipath and multiuser effects but assume a static or a slow fading channel (for the duration of the frame) and a single sensor at the receiver. We also assume a short repeating spreading sequence system for our algorithms.

2.2. Maximum Likelihood Based Channel Estimation

The channel parameters have to be extracted from the information contained in the received pilot signal. Since the observation vector $\mathbf{r_i}$ depends on the channel whose a priori statistics are unknown, a maximum likelihood estimate of the channel is often used. In our problem, $\mathbf{r_i}$ is a function of the channel vectors $\mathbf{z_k}$, the noise covariance matrix $\mathbf{K}$, which is assumed unknown and the transmitted bits $\mathbf{b_i}$. We assume that the bits $\mathbf{b_i}$ are known. This is accomplished

in the acquisition phase by requiring that all users transmit training pilot sequences. The following steps occur in the maximum likelihood method :

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

where $\mathbf{R}_{rr}$ is the autocorrelation of the observation vector, $\mathbf{R}_{br}$ the cross- correlation between the observation vector and the preamble bits, $\mathbf{R}_{bb}$ the autocorrelation of the preamble bits, $\mathbf{K}$ the noise covariance matrix, $\mathbf{Y}$ the estimate of $\mathbf{UZ}$, $\mathbf{U}$ the matrix of codes and $\mathbf{Z}$ 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.

2.3. Differencing Multistage Detection Algorithm

A Matched Filter Bank is usually the first stage in the baseband signal detection. Most of the multiuser detection techniques use the output of the matched filter bank and the cross-correlation information of all users in the system. The technique of the matched filter bank is to use one matched filter per user. The data bits are passed through the matched filter bank to detect each user's signal. The matched output is then sent through the multiuser detector where the parallel interference cancellation is done. The matched filter output is given by

$$\mathbf{y} = \mathbf{RAd} + \eta$$

where $\mathbf{y}$ and $\mathbf{d}$ are the output of the matched filter bank and the transmitted user data bits, $\mathbf{R}$ is the cross correlation matrix of the synchronized spreading codes and $\mathbf{A}$ is a diagonal matrix containing the amplitudes of the users. The cross correlation matrix $\mathbf{R}$ can be split into three parts, i.e.

$$\begin{aligned}
\mathbf{R} &= \mathbf{D} + \mathbf{S} + \mathbf{S^T} \\
\mathbf{B} &= (\mathbf{S} + \mathbf{S^T})\mathbf{A}
\end{aligned}$$

where $\mathbf{D} = \mathbf{diag}(\mathbf{R}) = \mathbf{I}$ and $\mathbf{S}$ is the lower triangular part of matrix $\mathbf{R}$. $\mathbf{B}$ is used later in the explanation of the algorithm for notational convenience.

After $\mathbf{M}$ iterations, it is likely that $\hat{\mathbf{d}}^{(l)} = \hat{\mathbf{d}}^{(l-1)}$, which reflects the convergence of the algorithm. So instead of dealing with each estimated bit vector $\hat{\mathbf{d}}^{(l)}$, we calculate the difference of the bits in two consecutive stages, i.e. the input of each stage becomes $\hat{\mathbf{x}}^{(l)} = \hat{\mathbf{d}}^{(l)} - \hat{\mathbf{d}}^{(l-1)} (j = 1, 2, \ldots, K)$, where $\hat{\mathbf{x}}$ is called the differencing vector. Significant savings in computations can be acheived as more and more elements in the vector $\hat{\mathbf{x}}^{(l)}$ tend to be zero after several iterations. Moreover, all the non-zero terms of $\hat{\mathbf{x}}^{(l)}$ equal to $+2$ or -2. This type of constant multiplication can

be implemented by arithmetic shifts, which will eliminate all multiplication operations. Further, the bit error rate (BER) in the system is not affected. The main steps in the algorithm[9] are as shown below:

$\hat{\mathbf{d}}^{(0)} = \text{sign}(\mathbf{y})$
for $k = 1$ to NK
  $z_k^{(1)} = y_k - \sum_{j=1}^{j=NK} B_{ij} \hat{d_j}^{(0)}$
end
$\hat{\mathbf{d}}^{(1)} = \text{sign}(\mathbf{z}^{(1)})$
for $l = 1$ to M
  $\hat{\mathbf{x}}^{(l)} = \hat{\mathbf{d}}^{(l)} - \hat{\mathbf{d}}^{(l-1)}$
  for $k = 1$ to NK
    $z_k^{(l+1)} = z_k^{(l)} - \sum_{j=1}^{j=NK} B_{ij} \hat{x}_j^{(l)}$
  end
  $\hat{\mathbf{d}}^{(l+1)} = \text{sign}(\mathbf{z}^{(l+1)})$
end

3. IMPLEMENTATION METHODOLOGY

We initially looked at a fixed point implementation for both the algorithms. However, the matrix inversions using the LU decomposition method along with squareroots and divisions in the proposed channel estimation algorithm made a fixed point implementation difficult. Hence, we decided to investigate a prototype floating point implementation for this algorithm and evaluate the use of the specialized single-cycle approximate instructions (Table 2) for this purpose in the C6701 DSP. We also investigated the wordlength issues for multistage detection[9] and found that the multistage algorithm can be suitably implemented using fixed point arithmetic. We study the performance of the channel estimation algorithm on the floating point TI TMS320C6701 EVM and that of the multistage detector on the fixed point TI TMS320C6201 EVM. Both the EVM boards have DSPs with 64 KB each of internal program and data memory, 256 KB of external SBSRAM (Synchronous Burst Static RAM) and 8 MB of external DRAM.

We use the TI Code Generation tools version 3.0 for the compiler and TI Code Composer Studio version 4.01 for profiling the code performance on the DSP. The execution time is calculated by setting profile points in the desired part of the code in Code Composer and setting the proper clock rates. A Pentium II 400 MHz machine serves as the host to the EVM. The Code Composer Studio uses the JTAG interface in the EVM for profiling the code.

Parameter	Notation
Preamble length	L
Number of Users	K
Number of Paths	P
Signal-to-Noise ratio	SNR
Signal-to-Interference-Noise ratio	SINR
Spreading gain	N
Window size	W
Number of Iterations	M

Table 1. Parameters for Channel Estimation and Multistage Detection

3.1. Algorithm Implementation

The parameters on which the channel estimation algorithm[6] and the multistage algorithm depend are shown in Table 1. The following parameters are used for our simulations for channel estimation as default :
N = 31, SINR = -10dB, SNR = 5dB, L = 150, K = 1 to 15, P = 3.
The main computation involved in the algorithm are matrix multiplications, matrix inversions, and matrix additions. The code was implemented in C. The algorithm used for inversion of matrices was the LU Decomposition method.[10] We use a simple Picksort method[10] for sorting and re-arranging arrays. Though not an optimal ($O(P^2)$) method, it is used as it is simple and P (the number of paths) is not large (three, in this case).

The following parameters are used for our simulations for multiuser detection as default:
N = 31, SINR = -10 dB, SNR = 6 dB, K = 12, W = 12, P = 1, M = 4.
The main computations involved in the algorithm are matrix - vector multiplication and additions. The code was implemented in C.

3.2. DSP Implementation : Channel Estimation

The algorithm was initially written in Matlab for a quick implementation and verification. After finding that a fixed point version may not be feasible, the code was then ported to general floating point C code. The code was first modified to remove file I/O for the DSP. Later, the functions were replaced with inline code for more aggressive optimization. Also, this leads to elimination of some of the temporary variables which helps to save precious internal memory space on the DSP. The C6701 DSP has assembly instructions which calculate the approximate inverse reciprocal and inverse square root. The use of these approximate instructions was investigated. By profiling the code, it was found that the critical part of the code consisted of matrix multiplications and dot product calculations. The use of assembly language subroutines was studied for the critical parts. The assembly subroutines are not only highly optimized code, but they also help in further reducing some temporary variables and loops. The assembly routines used were the routines for *matrix vector multiplication* and for *dot product calculation*. Both of these routines used were TI's floating point assembly benchmarks.[11]

The code was compiled[12] using the TI Code Generation Tools version 3.0 so as to place the variables and stack in the external SRAM[13] using the maximum optimization (-o3 -pm: file with program level optimization) while the code was placed in internal memory. A large memory model (-ml3) was used for our simulations due to the large data sizes of the chosen parameters used in our code.

3.3. DSP Implementation : Multistage Detection

The code was initially written in Matlab for a quick implementation and verification. The word length issues in the implementation of the algorithm were studied. The code was then ported to 16-bit fixed point C Code. We exploited several properties in the matrix structure to reduce the complexity.[9] The use of different optimization levels in our code was also investigated to see the benefits. It was found that dot product calculations were the critical part in the code. We investigated use of assembly subroutines for *dot product calculation*. We use TI's fixed point assembly benchmarks[14] for this purpose. This routine implements software pipelining, utilizing the entire eight functional units and helps the algorithm achieve a maximum performance of 2 MAC/cycle.

The code was compiled using the TI Code Generation Tools version 3.0 so as to place the variables and stack in internal memory as the entire data for the chosen parameters fits in the available 64 KB internal memory. We study the effect of different levels of optimization to observe the improvements with each of the compiler options. We use the default small memory model for the comparisons.

TMS320C6701 DSP	Cycles
FP add instruction	1
FP multiply instruction	1
Approx. FP divide instruction	1
FP divide function	28
Approx. FP reciprocal square root instruction	1
FP squareroot function	34

Table 2. Instruction Cycles for C6701 DSP instructions

4. SIMULATION RESULTS AND ANALYSIS

4.1. Impact of Specialized Instructions for Channel Estimation

The initial code was written using the square root and division routines provided in the math library. It was observed that these routines use the approximate specialized instructions available on the C6701 DSP as a initial seed and then use 2 or 3 iterations of Newton-Raphson's method for convergence. We can see from Table 2, that the functions for floating point reciprocals and square-root in the math library take 28 and 34 cycles[15] respectively, compared to the single cycle approximate instructions for reciprocal squareroots and division. The approximate instructions are supposed to be accurate in the exponent and atleast 8 bits in the mantissa. Hence, we decided to investigate the effects of using just the hardware instructions on the accuracy and the execution time of the code. The execution time for the base floating point code versus the code using specialized instructions for square root and reciprocal is shown in Figure 2.

From the figure, we observe approximately 10% improvement in execution time performance due to the use of the approximate specialized instructions. This benefit is due to the elimination of the iterations for convergence. We find that the use of these hardware instructions affect the accuracy of the algorithm to the order of 0.3%. The performance benefit is only 10% because the square roots and inversions are used only in the parameter extraction part, which is not the critical code in the program. As can be seen, the use of assembly for the critical part of the code has a much greater impact on the execution time, improving the performance by 100%. A close look at the algorithm reveals that if there were additional instructions and hardware support for complex arithmetic, a greater performance benefit would be observed.

4.2. Optimization levels for Channel Estimation

We also show the performance benefits obtained by going to different types of optimizations[16] for the floating point code. This is shown in Figure 3(a) by considering the case for 15 users and using -o3 compiler optimization. As can be seen from the figure, we achieve 1.08X improvement by using specialized instructions only for square roots and reciprocals. We can see that the execution time reduces by 2.34X by using highly optimized assembly subroutines for matrix multiplications and dot products, which is the critical part in the code. Also, the assembly subroutines help in eliminating some of the loops and intermediate variables. It may be possible to improve the performance more by coding completely in assembly. Thus, we can see an overall improvement of 2.52X by modifying the original code to suit the DSP.

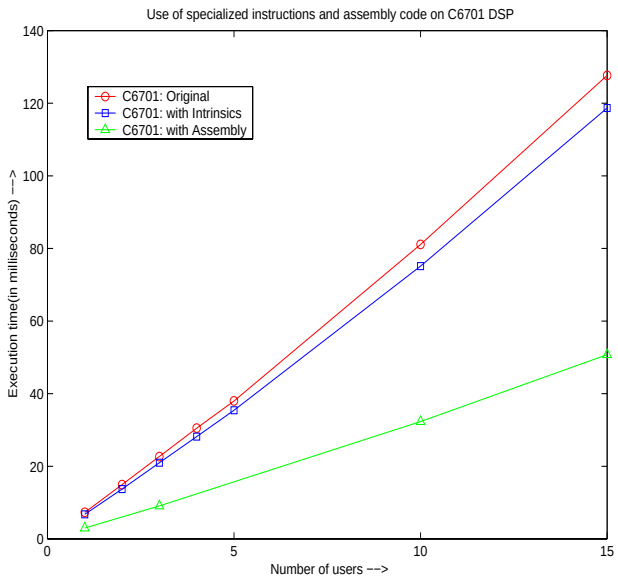
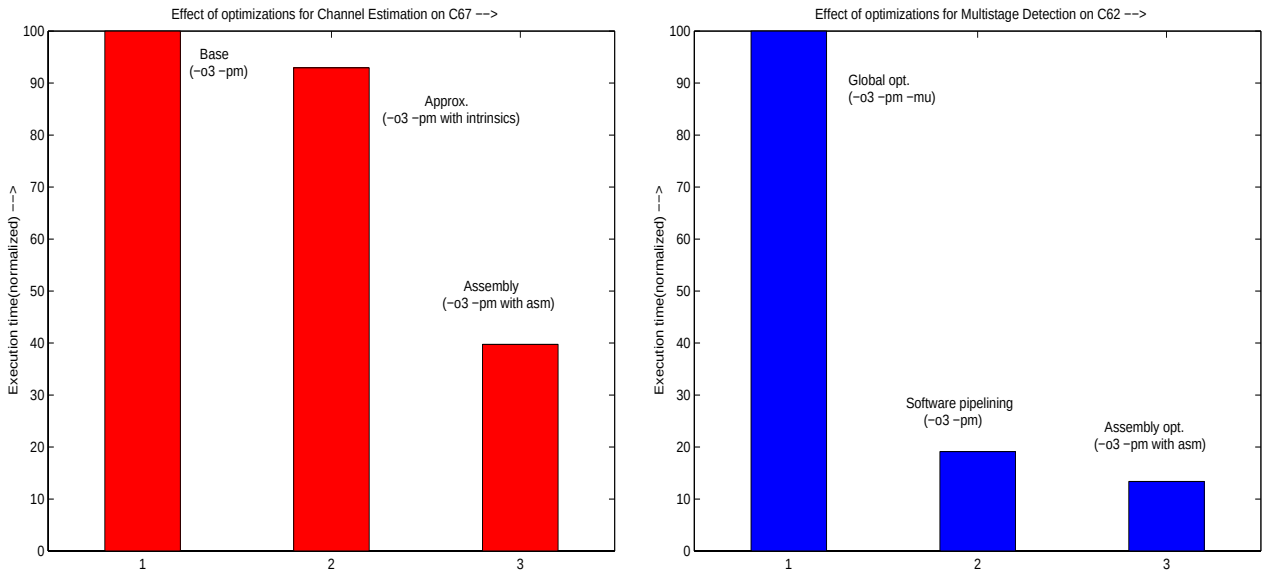


Figure 2. Impact of Specialized instructions and Assembly code on execution time



a. Execution time for Channel Estimation

b. Execution time for Multistage Detection

Figure 3. Optimization effects for Channel Estimation and Multistage Detection on C6701 and C6201 respectively

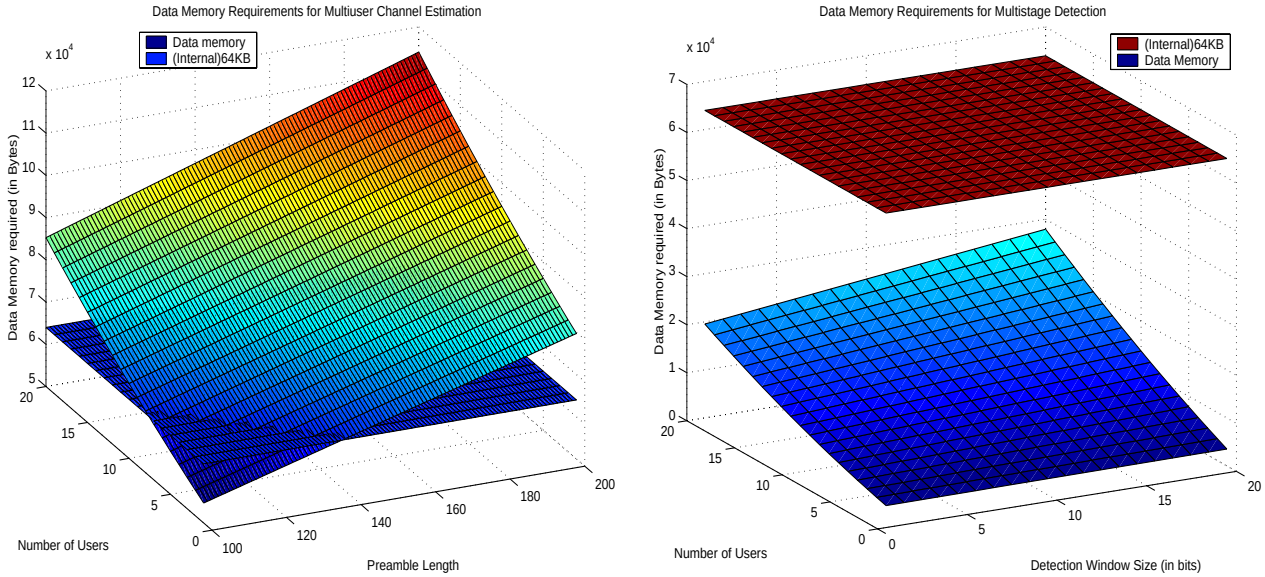
4.3. Optimization levels for Multistage Detection

Figure 3(b) shows the performance benefits obtained by the different optimization methods work for the multistage detection algorithm. The fully optimized code (-o3 -pm with assembly) executes 7.47X faster than the global register optimization (-o3 -pm -mu). Due to the lack of functional unit utilization information, we looked at the assembly code to find out the functional unit usage. The analysis using the assembly code could be done in this case as the code was easier to analyze than the channel estimation code. We saw from the assembly code that global register optimization (-o3 -pm -mu) without software pipelining has roughly one functional unit usage per instruction, which means no parallelism. The software pipelining optimization (-o3 -pm) approaches approximately five functional units usage per instruction. After inserting inline assembly code (-o3 -pm with assembly), we almost achieve the maximum functional unit usage rate i.e. eight per instruction.

4.4. Memory requirements for Channel Estimation

DSPs have very stringent memory requirements and the programmer has to manually perform memory allocation based on the size of the data. If the data does not fit in the 64 KB internal data memory of the chip, the programmer has to place it in external memory. The C6701 EVM has 256 KB external SRAM and 8 MB external DRAM. The memory requirements for the DSP is as shown in Figure 4(a). This is an approximate analysis and has been found by calculating the array sizes based on the chosen parameters. It can be seen that for the chosen parameter values, the data sizes for channel estimation do not fit in the internal 64 KB on-chip data memory and hence, have to be placed in external SRAM. This type of analysis is extremely useful in deciding the optimum parameter sizes such that all data can be placed in internal memory. In this case, we see that optimum allocation is between $L = 130$, $K = 1$ and $L = 100$, $K = 6$, where L is the preamble length and K is the number of users. This is a conservative estimate as the data memory may be used for other purposes such as the stack. However, for good performance, we need a preamble of length 150 and which can support more users. So, all data has to be placed in external 256 KB SBSRAM.

We have observed a significant loss of performance when data is placed in external memory, especially for kernel benchmarks such as dot product and FIR filter. This is due to the slower accesses to external memory. We have observed a factor of 4-5 improvement in these benchmarks when data is placed in internal memory. DSP programmers need to be very cautious about memory usage as it is a scarce resource. Hence, the recent trend in DSP processors to have greater levels of internal memory will be useful in this application. Also, the internal memory could be made of DRAM instead of SRAM. This would increase the memory latencies and complexity but would help accomodate



a. Channel Estimation b. Multistage Detection

Figure 4. Memory Requirements for Channel Estimation and Multistage Detection

larger memory on-chip. Also, significant power savings would be achieved as DRAMs consume significantly less amounts of power.

4.5. Memory requirements for Multistage Detection

The multistage detection algorithm completely fits in internal memory for the chosen data ranges. This is shown in figure 4(b). However, we see that the variation is more with the number of users. So, such an analysis is useful in finding the parameters which have the maximum effect on the data memory and hence, it could help in deciding optimal memory allocation.

5. MEETING REAL-TIME REQUIREMENTS

In this section, we discuss the real time requirements for channel estimation and multistage detection.

5.1. Real-time requirements for Channel Estimation

In the proposed W-CDMA standards, transmission is done using frames of 10 ms duration. Though the standards are not yet fixed about the estimation schemes, we feel that a modification of the proposed scheme will most likely be used in the Random Access Channel for the uplink, based on a slotted-ALOHA approach.[17,18] The random access burst consists of 2 parts, a *preamble* part of 1 ms followed by a *message* part of 10 ms. Between the preamble and the message, there is an idle time of 0.25 ms. The idle time allows for the detection of the preamble sequence and subsequent on-line processing of the message part. Hence, we would ideally like to complete the channel estimation in 0.25 ms. However, if the input data stream could be buffered, the estimation and detection could be done later. The real-time requirements will depend on the latency that the application can tolerate.

5.2. Real-time requirements for Multiuser Detection

As mentioned in the introduction, proposed W-CDMA systems have a data rate requirements of 2Mbps indoor, 384 Kbps pedestrian, and 144 Kbps vehicular. The complexity of the multistage algorithm[9] is approximately $3MK^2N$.

Using assembly code optimization, the C6201 DSP ensures about $\eta = 2$MAC per clock cycle in the kernel.[15] Since C6201 operates at a clock cycle of $T_c = 5ns$, we can obtain a processing speed of about 150Kb/s/user for a $K = 15$, $M = 4$ system.

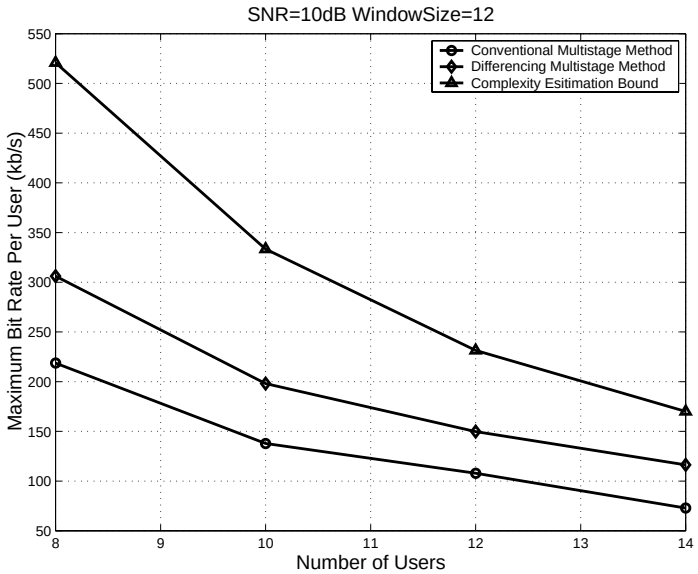


Figure 5. The real-time processing speed of C6201 fixed point DSP

This rate is obtained by the relation below:

$$R = \frac{1}{3T_c MK^2/\eta} = \frac{1}{3 \times 5ns \times 4 \times 15^2/2} \approx 150Kb/s/user \tag{1}$$

This shows that a maximum of 15 users can be supported at a data rate of 150 Kb/s/user. Thus, greater advances are required, both in algorithms and architectures for wireless communications, to handle the next generation W-CDMA requirements. Figure 5 shows the complexity estimation bound and the actual profiling result by C6201. In a 12-user system, the differencing multistage detector can reach up to 150Kb/s/user, while the conventional multistage detector[8] can only process at 100Kb/s/user. This is due to the recoding techniques and the computational savings used in the differencing algorithm. The speed curve does not approach the theoretical bound because many other overhead operations such as memory accesses are involved.

5.3. Current and Future Trends in DSPs for Wireless Communications

The current trends of DSPs for wireless communication applications have shown highly encouraging signs. The recently announced C6203 DSP has 512 KB of internal data memory and 384 KB of internal program memory. This is almost 8 times the amount of internal memory present in the C6x01. This will definitely help in applications with large data and program sizes such as the proposed channel estimation algorithm. The effect of DMA controllers and caches may also be very useful in applications that need to access external memory. For example, the C6x11 has 4 KB each L1 program and data caches and a 64 KB L2 cache. The recent trends in DSPs to go to higher clock speeds (C6203 - 250 MHz) and low voltages (C5402 - 1.2V) will also be beneficial for wireless communication applications, where power efficiency and speed is very critical. We have also seen the advantage of integrating specialized blocks such that the Viterbi Decoder in C54x in wireless communication applications. Hence, chip makers are suggesting such approaches[19] of a DSP core with additional specialized blocks for handling the performance requirements of next generation W-CDMA. The compiler support for DSPs are also improving with the new version 3.0 of TI's C compiler and Code Composer Studio. Our suggestions for OS support in the compiler to help in memory allocation has also been acknowledged by TI for their future versions of Code Composer Studio.

It is suggested[20] that even a futuristic billion-transistor-in-a-chip processor may not be able to satisfy personal mobile computing requirements, if most of the chip area is devoted to caches. More chip area should be devoted to internal memory and functional units to make a processor suitable for wireless applications. The use of a Vector IRAM for future processors is also suggested. The future for processor technology for wireless applications looks exciting and an interesting fusion of the above processor technologies may be seen in the near future.

6. CONCLUSIONS

The implementation of one of the proposed methods for multiuser channel estimation and detection for W-CDMA on DSPs was studied. It was found that these algorithms have different requirements in terms of complexity and precision issues. A feasibility analysis showed that the channel estimation algorithm may be better suited for a floating point implementation while the multistage detection algorithm could be performed using a fixed point implementation. An analysis of the memory requirements for the algorithms showed that the 64 KB internal memory of the C6701 was not sufficient for the chosen parameter sizes while the multistage detection algorithm fits in the available 64 KB internal memory of the C6201. The effect of specialized instructions on the C6701 was also studied for channel estimation. It was found that these instructions would be very useful in algorithms that involve a large number of square roots and divisions. It was also found that additional instructions to support complex arithmetic would also benefit wireless communication applications. The recent trends in DSPs to have increased internal memory, faster clock speeds, additional specialized blocks and instructions, better compilers and lower voltages all seem to greatly enhance the use of DSPs for the next generation wireless communication systems.

REFERENCES

1. 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.
2. 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.
3. Tero Ojanpera and Ramjee Prasad, "An Overview of Air Interface Multiple Access for IMT-2000/UMTS," Tech. Rep., Nokia Research Center, Delft University of Technology, September 1998.
4. Andreas Polydoros and Savo Glisic, *Code Division Multiple Access Communications*, pp. 225–266, Kluwer Academic Publishers, 1995.
5. Zoran Kostic and Selvarajan Seetharaman, "Digital Signal Processors in Cellular Radio Communication," *IEEE Communications Magazine*, pp. 22–35, December 1997.
6. Chaitali Sengupta, *Algorithms and Architectures for Channel Estimation in Wireless CDMA Communication Systems*, Ph.D. thesis, Rice University, Houston, TX, December 1998.
7. 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.
8. 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, Apr. 1990.
9. Gang Xu, "Implementation Issues of multiuser detection in CDMA Communication Systems," M.S. thesis, Rice University, Houston, TX, May 1999.
10. Willam Press, Brian Flannery, Saul Teukolsky, and William Vetterling, *Numerical Recipes in C*, Cambridge University Press, 1991.
11. TI, "TMS320C67X Assembly Benchmarks at Texas Instruments," http://www.ti.com/sc/docs/products/dsp/c6000/67bench.htm.
12. TI, *TMS320C6x Optimizing C Compiler : User's Guide*, TI, February 1998.
13. TI, *TMS320C6x Evaluation Module : Reference Guide*, TI, February 1998.
14. TI, "TMS320C62X Assembly Benchmarks at Texas Instruments," http://www.ti.com/sc/docs/products/dsp/c6000/62bench.htm.
15. TI, *TMS320C62x/C67x CPU and Instruction Set : Reference Guide*, TI, March 1998.
16. TI, *TMS320C62x/C67X : Programmer's Guide*, TI, February 1998.
17. Riaz Esmailzadeh and Maria Gustafsson, "A New Slotted ALOHA Based Random Access Method for CDMA Systems," in *ICUPC*, October 1997.
18. ETSI, "Submission of Proposed Radio Transmission Technologies," Tech. Rep., http://www.etsi.org/smg/UTRA/utra.pdf, 1998.
19. Anthony Cataldo, "Chip makers sketch plans for 3G Cell Phones," http://www.eet.com/story/OEG19990416S0026, April 1999.
20. Christoforos E. Kozyrakis and David A. Patterson, "A New Direction for Computer Architecture Research," *IEEE Computer*, pp. 24–32, November 1998.