Real-Time DSP Multiprocessor Implementation for Future Wireless Base-Station Receivers

Bryan Jones, Sridhar Rajagopal, and Joseph Cavallaro
ECE Department and Center for Multimedia Communication
Rice University, Houston, TX 77005
{bryan, sridhar, cavallar}@rice.edu

Abstract

The convergence of cellular phones, the Internet, and laptop computers into a single small, lightweight, wireless information appliance drives a need for a high data rate, low-power digital wireless communication link to enable the creation of such a device. A simulation environment supporting rapid prototyping is developed, and used to evaluate the real-time data-rate performance of these receivers implemented on a multiprocessor DSP board. Simulations of a multiprocessor implementation of joint multiuser channel estimation and detection algorithms is projected to achieve combined performance of 15.6 Kb/user/sec for 10 users. Performance gains over a single-processor implementation range from 5% for the three-user case to 69% for a 15 user case.

1 Introduction

In the past decade, the usage of cellular phone services have grown dramatically. Convergence between the traditional cellular phone, the Internet, and the laptop computer have driven a desire for a wireless device with e-mail, web browsing, live video capabilities [1]. The availability of compact digital replacements for their larger analog counterparts – such as digital cameras, camcorders, and MP3 players – give rise to the concept of combining all these devices into a wireless pocket-sized information appliance. At the same time, dissatisfaction with the talk time on today's cell phones drives a desire for longer battery life in these new devices.

Meeting these demands poses formidable challenges for today's wireless designer. To provide new forms of content, higher data rate communications systems must be developed. Proposals for the next generation of cell phones – such as the 3GPP group – propose data rates of 128 Kb/sec [2], compared to data rates of 19.2 Kb/sec specified by today's IS-95 standard [3]. To keep pace with an order of magnitude increase in data rate, the computational power of base stations must also increase by an similar amount. In section 3, the characteristics of a CDMA-based wireless system are discussed, and advanced base-station algorithms to distinguish these lower-power signals from background noise are presented.

As wireless systems grow more and more complex, a robust simulation environment in which the effects of algorithms and hardware can be modeled and comprehended has become a necessity. In order to deliver this new technology to an extremely competitive market, the design and simulation tools used must enable a rapid flow from algorithm prototyping to software and hardware implementation. In section 4, the develop-

ment of a simulation environment, and its integration with a DSP-based multiprocessor board is discussed. Finally, section 5 presents results from experiments performed with this architecture.

2 Related work

The field of multiuser wireless communications is replete with many algorithms and implementations. A DSP-based system employing partial parallel interference cancellation is discussed in [4], a less complex form of the parallel interference cancellation studied in this paper. Improvements in bit error rate performance over a single-user matched-filter detector are demonstrated using multiuser detection. A software radio architecture employing linear multiuser detection is presented in [15]. A Simulink alternative, SPW, is used to simulate the system and then implement it on a TI TMS320C4x/Xilinx FPGA hardware platform. However, the resulting hardware/software system is not integrated into the simulation environment, making debugging and real-time performance monitoring more difficult. The computational time spent performing a channel estimation is often overlooked when reporting real-time performance. The joint channel estimation and detection scheme presented in this paper addresses these oversights by reporting an overall channel estimation and detection performance.

3 System description

To enable the information appliance envisioned in the paper, a base station receiving signals from multiple users of the appliance must incorporate advanced channel estimation and detection algorithms. However, to meet high data-rate requirements, these algorithms must have limited computational demands, and efficiently map to hardware. To satisfy these requirements, a newly-developed joint channel estimation and detection scheme is selected. This scheme consists of an iterative maximum-likelihood based channel estimator [5]; by solving for the channel estimate iteratively, an expensive matrix inversion step is avoided when updating the channel estimate. For detection in the joint scheme, a pipelined version of the multistage parallel-interference cancellation detector [6], [7] is selected; it offers better performance than previous windowing detectors. A brief description of these algorithms follows.

Input from the antenna, after being demodulated into a baseband signal, is then passed through a chip-matched filter. This filter integrates the received signal for the period of one chip, outputs the resulting integration, and then resets its output to zero (dumps) in preparation for the next chip period. This is typically implemented digitally, by oversampling the incoming signal with an A/D converter, then summing the samples taken during each chip period. Both the channel estimator and detector use the output of the chip-matched filter.

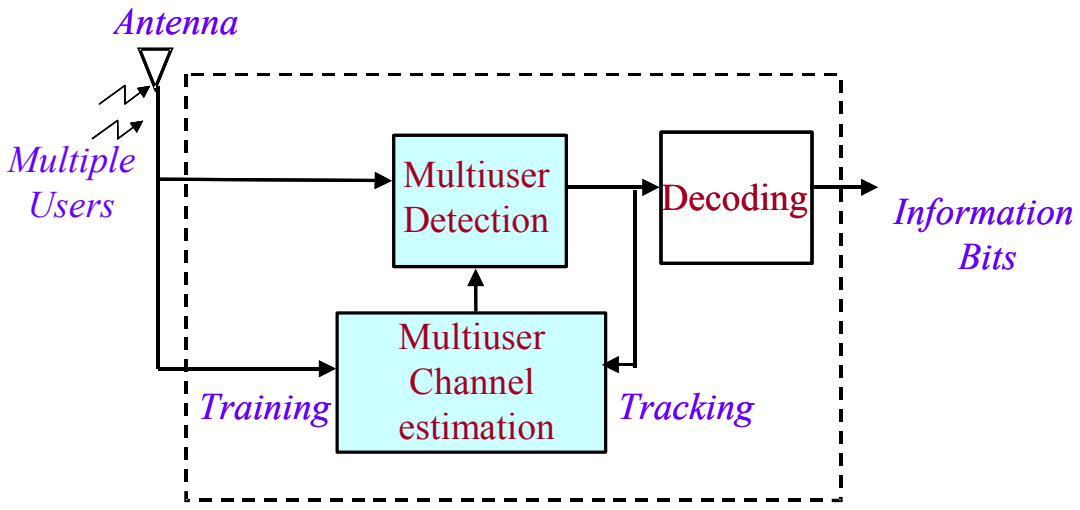

Figure 1: Base-station receiver

Channel estimation occurs in two phases: a pilot, or preamble phase, in which all users transmit a known sequence of bits to allow the receiver to form a channel estimate by correlating it with the chip-matched filter output; and a tracking phase, during which unknown (data) bits are transmitted by all users. To track the channel during this phase, bits output by the detector are fed back to the channel estimator, along with corresponding chip-matched filter outputs. These detected bits are used in the same way the known pilot bits are used to update the channel estimate. Algorithmically, the same computations are carried out during both the pilot phase and tracking phase; only the source of the data bits input to the channel estimator varies. Figure 1 gives the main computational blocks in the receiver. This paper concentrates on the processing that occurs in the multiuser channel estimation and multiuser detection blocks.

The estimation is carried out by first forming two matrices: an auto-correlation matrix R_{bb}, and a cross-correlation matrix R_{br}. The auto-correlation matrix, size $2K \times 2K$, captures the effects of each of the K users' bits on that user's signal as it moves through the channel; the cross-correlation matrix, size $N \times 2K$, captures the effects that each chip in one user's transmitted bit, where N specifies the number of chips in a bit, has on another user's bits. Next, the channel estimate, A, is formed by multiplying R_{bb} by R_{br}^{-1}. To reduce computational complexity, the matrix inversion of R_{br} is approximated using an iterative algorithm employing the method of steepest descent. A windowing method is used to subtract past bits and chips from the two matrices, and then add in new bits and chips as they arrive. The equations describing the iterative estimator are given below.

$$A^{(i)} = A^{(i-1)} - \mu(A^{(i-1)} * R_{bb}^{(i)} - R_{br}^{(i)})$$
$$R_{bb}^{(i)} = R_{bb}^{(i-1)} + b_L * b_L^T - b_0 * b_0^T$$
$$R_{br}^{(i)} = R_{br}^{(i-1)} + b_L * r_L^H - b_0 * r_0^H$$

The detection is accomplished using a multi-user parallel-interference cancellation (PIC) algorithm. First, the transpose of the channel estimate is multiplied by the incoming chips from the chip-matched filter. This process, code-matched filtering, produces an initial estimate of the bit transmitted by each user in the system. This initial estimate is used as a starting point for parallel interference cancellation, an iterative process that removes multiple-access interference (MAI) generated by other users from the received signal, then makes a more accurate estimate of each user's transmitted bit. The output from one iteration of the PIC is fed into the next iteration, until the resulting outputs have converged. The equations below describe code-matched filtering and PIC.

$$
\begin{aligned}
y &= A^T r \\
d &= sign(y) \\
L &= A_1^H A_0 \\
C &= A_0^H A_0 + A_1^H A_1 - diag(A_0^H A_0 + A_1^H A_1) \\
y_i^{(l)} &= y_i^{(0)} - L d_{i-1}^{(l)} - C d_i^{(l)} - L^H d_{i+1}^{(l)}
\end{aligned}
$$

4 Implementation

Two aspects of the implementation of the algorithms discussed in the previous section are covered. First, the a wireless simulation environment is presented. A methodology for interfacing this system with algorithms executing on a multi-DSP system is developed, and used to collect real-time performance data from the system. Second, the multi-processor board is introduced, and a methodology for implementing real-time measurements of execution time is detailed.

4.1 Wireless simulation environment and interface

Testing and development of the algorithms are carried out in a wireless simulation environment. Such an environment should produce two significant measures. First, the environment should provide an indication of the receiver's accuracy in the presence of transmissions from multiple users. Second, it should measure of the data rate the algorithm can sustain in a real-world environment. To provide a complete simulation environment, it must simulate the effects of multiple users as they transmit through a multipath fading channel. Finally, the environment should facilitate easy testing of the algorithms over a variety of conditions. Simulink, a graphical data-flow time-domain simulator based on the Matlab engine, allows the creation of such an environment. A block diagram, incorporating all elements of the communications system, such as multiple transmitters, the channel, and the receiver, is developed in Simulink (Figure 3). The block

Figure 2: RealSync blocks

diagram also contains an error counter, to measure the BER of the receiver. A dialog box and corresponding configuration script allows the user to easily vary system parameters, and test a variety of receivers. The system supports three detectors: a code-matched filter, a decorrelator, and a PIC detector. It provides one channel estimator, the maximum likelihood multiuser estimator discussed earlier.

However, the environment lacks a means to evaluate the real-time performance of the receiver algorithms on a DSP. To provide this ability, the Rice RealSync program is developed, providing a means to execute algorithms written for the Sundance multiprocessor board in real time, while also transparently interfacing to the Simulink graphical environment. RealSync incorporates both a standardized Simulink interface to C++ (an S-function [8]), and a set of C/C++ routines that allow the DSP to exchange data with a Windows host (the 3L WinServer [11]). Figure 2 shows two RealSync Simulink blocks; by simply connecting their inputs and outputs in a Simulink block diagram, RealSync automatically loads the DSPs with the appropriate code, then executes the algorithms running on the DSPs. Data provided to the input ports of the RealSync Simulink block, and results of the DSPs' execution, are exchanged with Simulink via the ports of the RealSync Simulink block, as shown in Figure 2.

One of the challenges involved in developing RealSync is in overcoming the master/slave paradigm inherent in both the S-function and WinServer interfaces. Both the S-function interface and the WinServer interface operate as the master in a master/slave system: as the master in the system, Simulink and the DSP send a message to the slave interface (S-function or WinServer) providing input data, and then expect output data to be received in return from the slave. In order to interface these two masters, a multithreaded approach is taken: two threads are created, one for each of the two masters. Semaphores are then used to synchronize data exchange between the two interfaces, as specified below:

1. Thread A receives data from its master.

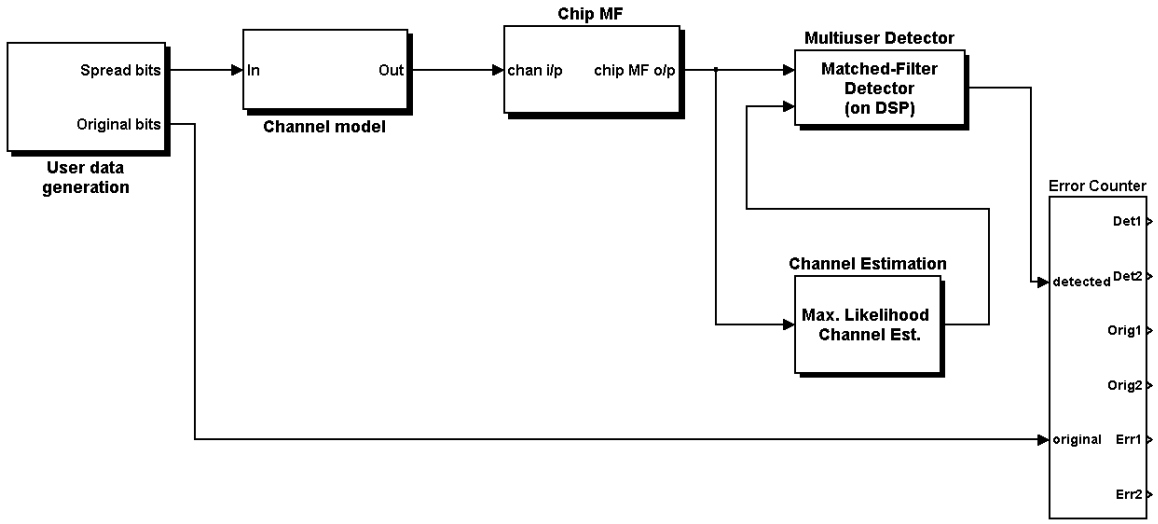

Figure 3: Simulink block diagram of CDMA system

2. Thread A signals thread B's "data available" semaphore.
3. Thread A waits for thread B to signal its "data available" semaphore.
4. Thread A returns the data now available from thread B to its master.

For example, Simulink passes data to RealSync's input port via the S-function interface. RealSync's Simulink thread then signals the DSP thread that input data is available. The DSP thread then transfers this data to the DSPs via WinServer interface. When the DSPs finish processing this data, they return it to the DSP thread, again via the WinServer. Having received the data, the DSP thread then signals the Simulink thread. Via the S-function interface, the Simulink thread then returns output data to Simulink. Simulation then continues within other blocks in the Simulink block diagram.

4.2 Multiprocessor implementation

In order to better achieve the high data rates required by an information appliance, channel estimation and detection must be carried out rapidly. To accomplish this, the channel estimation and detection tasks are partitioned among the two available DSPs, and an efficient interprocessor communications link is utilized. A brief description of the multiprocessor hardware/software environment is given, followed by a description of the channel estimation and detection algorithms. Finally, the multiprocessor implementation of the channel estimator and detector is described.

The Sundance, Inc. multiprocessing system [9] provides a modular approach to DSP-oriented processing. In our system, a SMT320 32-bit PCI carrier board provides for the connection of 4 TMS 320C4x-style TI modules (TIMs). Sundance offers a wide variety of TIMs; for our system, two SMT372 TIMs were chosen. Each module contains one TMS320C6701 DSP running at 167 MHz, 512 KB of SBSRAM, and 16 MB of SDRAM. For interprocessor communication, the SMT372 contains FPGAs that provide C4x-style comm-ports, accessible through the C67's external memory interface (EMIF). Two Sundance Digital Bus (SDB) ports, a proprietary high-speed point-to-point interconnect, are also interfaced to the C67's EMIF. Two additional TIMs, each containing a Xilinx Virtex 300 K gate FPGA, were also chosen. Figure 4 gives a block diagram of the multiprocessor system. The 3L Diamond real-time operating system (RTOS) [10], which provides multitasking, multiprocessor, and a host/processor interface, is used to speed development of the system. Additional drivers from Sundance and 3L enable the use of TI's Code Composer Studio for debugging. The 3L WinServer [11], a host-side interface to Microsoft Windows, is used to integrate the system into a wireless simulation environment executing on the host (see section 4.1). TI's C++ compiler and linker, combined with 3L's multiprocessor configurer, is used to generate multiprocessor-ready object code.

To maximize the efficiency of the implementation of the channel estimation and detection routines, the routines are hand-coded in C++ and optimized based on feedback from TI's assembly optimizer. A C6000 on-board timer is used to measure execution time of the algorithms, and provide immediate feedback after running the program. Because the timers only require CPU cycles to start and stop the timer, they add minimal

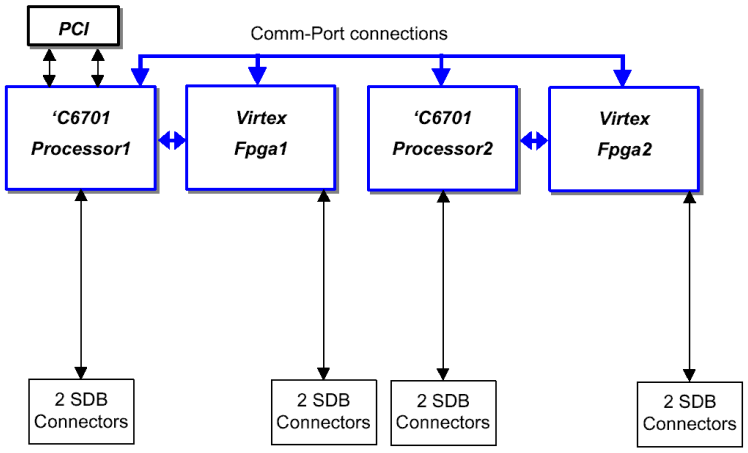

Figure 4: Sundance multiprocessor board block diagram

overhead to the actual execution time. The number of iterations for all loops in the code is known at compile time, allowing the optimizer to produce more efficient code.

5 Experimental results

The RealSync program allows both the development and verification of algorithms in Simulink and enables their real-time execution on the Sundance multiprocessor DSP platform. Execution speeds of each matrix multiply are measured using the C67's on-chip timer, and results reported back to the Matlab/Simulink environment via RealSync. In the same way, interprocessor communication speed is measured and recorded. Simulation parameters calculated in the Matlab/Simulink environment, such as the bit error rate, are also available for analysis. Bit error rate simulations of the joint channel estimation and detection scheme are presented in [5], [7] and are not repeated here.

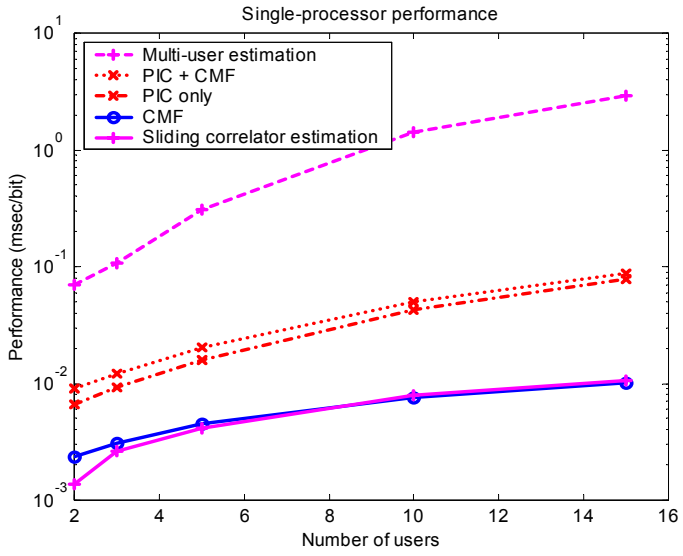

Figure 5: Single-processor performance

A set of Matlab m-files are developed to vary simulation parameters (number of users, detector type, single/multi-processor configuration) and then to capture the resulting simulation results. All simulations are carried out using a static AWGN channel, with SNR=6dB, and a preamble sequence of 150 bits followed by 1000 bits of random data. Gold codes are used to spread the incoming data to N=31 chips/bit. The channel estimator runs only during the preamble period; the detector runs only during the data phase. The parallel-interference cancellation (PIC) detector is implemented using 4 stages of interference cancellation.

Single-processor simulations (Figure 5) show execution times for two channel estimators and two detectors. The two estimators profiled are the iterative multiuser channel estimator discussed earlier, and the sliding correlator. The execution time for the sliding correlator estimator is measured as the time taken to compute R_{br} while calculating the multiuser channel estimate. Two detectors are presented, the code-matched filtering (CMF) detector, and the parallel interference cancellation detector. The PIC detector depends on output from the code-matched filter; however, note that its overall execution time (labeled PIC plus CMF on the graph) is dominated by the PIC operation. Note that the channel estimator is almost an order of magnitude slower (~0.1 ms/bit, or 10 Kbit/sec, for 2 users) than the PIC detector.

These results suggest an intuitive multiprocessor task partitioning: execute the estimator on one processor, and the detector on the second processor. This task partitioning is carried out, using comm-ports exchange data between processors. Execution times, and interprocessor overhead, are measured. Figure 6 shows the outcome of this partitioning. Incoming data from the channel arrives after chip-matched filtering at the first processor. The first processor sends this chip-matched filter output to the second processor, where detection takes place. The detector then sends its detected bits back to the first

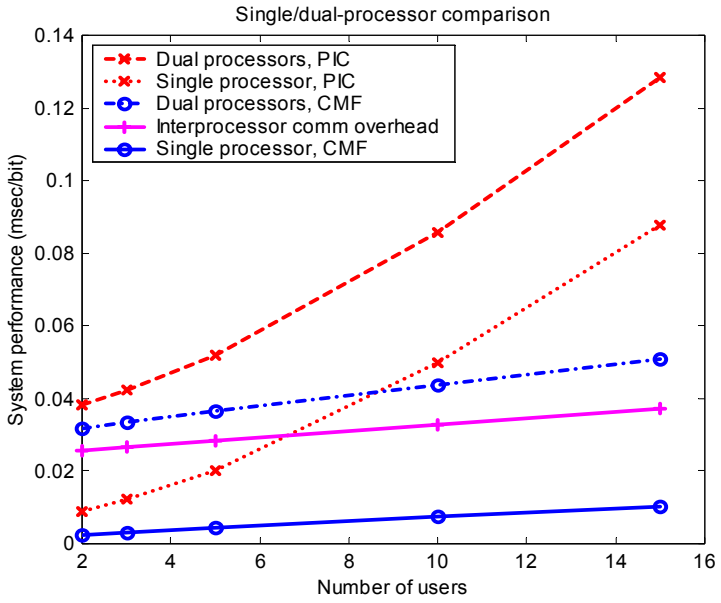

Figure 6: Single/dual-processor performance comparison

processor. The channel estimator can then use these detected bits for tracking. In addition, the first processor owns the only communications channel with the host, and must relay detected bits back to the host on behalf of the detector running on the second processor for BER analysis. Finally, the first processor carries out its iterative channel estimation update.

Communication overhead, currently implemented by blocking both processors until all data is transferred, proves to be a large overhead for the code-matched filtering operation. It is less significant for the PIC detector, particularly as the number of users increases. These results suggest means to improve the system performance, particularly when a fading channel is present, forcing the channel estimator during the data phase in addition to the preamble phase. Because of the high cost of multiuser channel estimation, we propose reducing the frequency of iterative channel estimate updates. In a multiprocessor system, the maximum rate achievable is max{execution time over all processors} + communication overhead. Choosing the detector to limit system performance, the estimator updates can be run every M^{th} bit, where M is chosen based on estimator/detector execution times.

Based on measured interprocessor communication speeds of 5.0 MB/sec and measured channel estimation and detection speeds, the following values of M are chosen to maximize processor utilization; ideally, both the estimator and detector processors need not wait on their partner to finish a computation. The interprocessor (IP) communication required, in bytes transferred between the two DSPs, is also given.

# users	M	IP overhead (bytes)
2	8	178
3	10	197
5	17	196
10	30	205
15	35	228

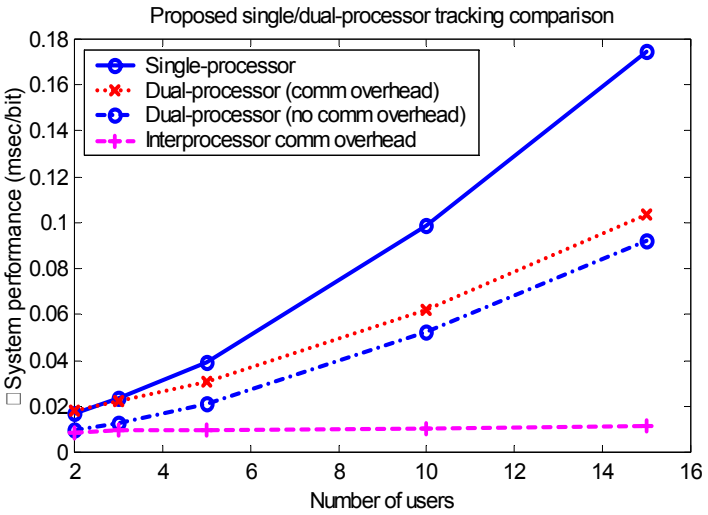

Figure 7: Tracking comparison

A graph is created by extrapolating measured values from the system. The proposed system runs the channel estimator on one processor, and the detector on the second processor. A conservative interprocessor communication speedup of 4 (20 MB/sec) is assumed, which can be implemented using a DMA engine to enable rapid non-blocking transfers between processors. Figure 7 demonstrates a significant speedup possible under the proposed tracking scheme. The speedups illustrated in the figure above range from 5% for the three-user case to 69% for the 15-user case, with a 59% speedup for the 10 user case.

6 Conclusions and future work

RealSync, a methodology for merging the flexibility and data analysis capabilities of a Simulink-based wireless simulation environment with real-time DSP execution is presented. This methodology is used to evaluate performance of channel estimation and detection algorithms in both a single and multi-processor environment. An analysis of simulation results demonstrates a projected performance improvement of 59% in the multi-processor system.

In the future, the integration of portions of the algorithm into FPGAs can provide additional performance gains [12]. Additional work in implementing the proposed tracking scheme will demonstrate the feasibility of the multi-processing system [13], [14]. A fixed-point implementation, as in [15], will increase system performance. Converting the present system to fixed-point would significantly reduce communication overhead, and allow the use of the faster C64 DSP.

References:

[1] David J. Goodman, "The Wireless Internet: Promises and Challenges," *IEEE Computer*, vol. 33, no. 7, pp. 36-41, July 2000.

[2] E. SMG2. The ETSI UMTS Terrestrial Radio Access (UTRA) ITU-R RTT Candidate Submission. Technical report, European Telecommunication Standard Institution (ETSI), May 1998.

[3] J. Lee and L. Miller. CDMA Systems Engineering Handbook, Artech House, Boston, pg. 341.

[4] N. S. Correal, R. M. Buehrer, and B. D. Woerner, "A DSP-Based DS-CDMA Multiuser Receiver Employing Partial Parallel Interference Cancellation," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 613-630, April 1999.

[5] 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, 1999.

[6] M. K. Varanasi and B. Aazhang, "Multistage detection in asynchronous Code-Division Multiple-Access communications," *IEEE Transactions on Communications*, vol. 38, no. 4, pp. 509-519, Apr. 1990.

[7] S. Rajagopal, S. Bhashyam, J. Cavallaro, and B. Aazhang, "Efficient Algorithms and Architectures for Multiuser Channel Estimation and Detection in Wireless Base-Station Receivers", to be submitted to *IEEE Journal on Selected Areas in Communications*, www.ece.rice.edu/~sridhar/research/asap.pdf.

[8] Mathworks, Inc. Writing S-Functions, version 3. http://www.mathworks.com/access/helpdesk/help/pdf_doc/simulink/sfunctions.pdf.
[9] Sundance Inc, http://www.sundance.com.
[10] 3L, Inc. Diamond, C6x Edition Version 1.3 User Guide, http://www.3l.com/c6x/index.htm.
[11] 3L, Inc. 32-Bit Windows Server, Version 2.0, http://www.3l.com/winser/index.htm.
[12] A. Gatherer, T. Stetzler, M. McMahan, and E. Auslander, "DSP-Based Architectures for Mobile Communication: Past, Present and Future," *IEEE Communications Magazine*, vol. 38, no. 1, pp 84-90, January 2000.
[13] P. D. Hoang and J. M. Rabaey, "Scheduling of DSP Programs onto Multiprocessors for Maximum Throughput," *IEEE Transactions on Signal Processing*, vol. 41, no. 6, pp 2225-2234, June 1993.
[14] C. Sengupta, K. Kota, and J. Cavallaro, "Parallel algorithms and architectures for subspace based channel estimation for CDMA communication systems," *Proceedings of SPIE Conference on Advanced Signal Processing Algorithms, Architectures, and Implementations VI*, vol. 2846, pp 412-423, August 1996.
[15] I. P. Seskar and N.B. Mandayam, "A software radio architecture for linear multiuser detection," *IEEE Journal on Selected Areas in Communications*, vol. 17, issue 5, pp 814-823, May 1999.