# An Efficient Parallel Architecture for Implementing LST Decoding in MIMO Systems

Amirhossein Alimohammad, *Student Member, IEEE*, Bruce F. Cockburn, *Member, IEEE*

***Abstract*— Recovering the symbols in a multiple-input multiple-output (MIMO) receiver is a computationally-intensive process. The layered space-time (LST) algorithms provide a reasonable tradeoff between complexity and performance. Commercial digital signal processors (DSPs) have become a key component in many high-volume products such as cellular telephones. As an alternative to power-hungry DSPs, we propose to use a moderately-parallel single-instruction stream, multiple-data stream (SIMD) co-processor architecture, called DSP-RAM, to implement an LST MIMO receiver that offers high performance with relatively low power consumption. For a typical indoor wireless environment, a 100-MHz DSP-RAM can potentially provide more than 10 times greater decoding throughput at the receiver of a (4,4) MIMO system compared to a conventional 720-MHz DSP. The DSP-RAM processor has been coded in a hardware description language (HDL) and synthesized for both available field-programmable gate arrays (FPGAs) and for a 0.18$-\mu m$ CMOS standard cell implementation.**

***Index Terms*— Layered space-time decoding, MIMO receiver, processor-in-memory, parallel processing.**

## I. Introduction

MULTIPLE-INPUT multiple-output (MIMO) systems have emerged as an attractive new paradigm for spectrum-efficient wireless communications in rich multipath fading environments [1]. The MIMO architecture can exploit diversity in both space and time to significantly increase system capacity as well as improve the quality (i.e. reduce the symbol error rate (SER)) of the wireless link in the presence of adverse propagation conditions, such as multipath fading and interference.

Due to the high aggregate link capacity, an important first challenge is to minimize the computational complexity of the decoding algorithm at the MIMO receiver. Among the published linear decoding techniques [1], the *layered space-time* (LST) algorithms [2], [3], [4], [5] employ a divide-and-conquer approach where, rather than jointly decoding the received symbols from all transmitter antennas, the receiver sequentially decodes one symbol at a time beginning, preferably, with the symbol with the highest signal-to-noise ratio (SNR). The LST decoding algorithms then predict and then subtract away the interference due to the most recently decoded symbol from the simultaneously received signals, and then proceed to decode the next symbol, and so on. The computational complexity of LST decoding is reported to be $\mathcal{O}(n^4)$ in the number $n$ of antennas for the zero-forcing (ZF) [3] and mininum mean-squared error (MMSE) LST algorithms, and $\mathcal{O}(n^3)$ for the square-root [6] and ordered QR-LST algorithms [7]. However, the running time of an algorithm depends on the number of hardware instructions that need to be executed, and this number depends on the architecture of the processor. Given a scalable parallel processor architecture, in particular, a key factor is the degree to which the algorithm can be parallelized and mapped efficiently onto the available processor resources.

The great computational demands of MIMO signal decoding can exceed the performance available from even high-end DSPs. Therefore a second practical challenge in MIMO systems is to develop low-power, but sufficiently high-performance, hardware to implement the receiver. Application-specific integrated circuits (ASICs) have been proposed to implement MIMO decoding algorithms [8], [9] and contemporary FPGAs have been used successfully to prototype MIMO testbeds [10], [11], [12]. However, in the published LST decoding algorithms, there is abundant inherent parallelism that has yet to be exploited to increase the symbol decoding throughput at the receiver. Therefore an alternative approach to using a faster conventional processor or an ASIC is to identify and exploit opportunities for parallel processing using a flexible parallel architecture to maximize the useful work that is accomplished in every clock cycle.

DSP-RAM is a moderately-parallel, scalable SIMD co-processor architecture for high-performance signal processing [13], [14]. In the *processor-in-memory* (PIM) architecture [15] of DSP-RAM, a linearly-interconnected array of simple *processing elements* (PEs) is integrated with associated local memories. Integrating the processors with the memories exposes the enormous data bandwidth between the two, and eliminates the bottleneck that otherwise occurs on an external bus between the memory chips and processor(s) in conventional architectures. The degree of parallelism (i.e., the number of PEs) provides a trade-off between including more transistors on a die to increase the throughput of a parallel algorithm, and running at a slower clock frequency to simplify the implementation and still meet the required processing performance. In addition, reductions in the core operating voltage might be possible. Since dynamic power consumption is proportional to the voltage squared [16], the possibility of reducing the core voltage would be especially attractive in power-constrained systems.

The rest of this paper is structured as follows: Section II reviews the LST MIMO architecture. The related decoding algorithms and the key characteristics that make them suitable for parallel realization are discussed. The parallel DSP-RAM architecture is presented in Section III. Section IV discusses

Manuscript received June 1, 2004; revised December 12, 2004. This work was supported by the Natural Sciences and Engineering Research Council of Canada under grant OGP0105567.

A. Alimohammad and B. F. Cockburn are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada.

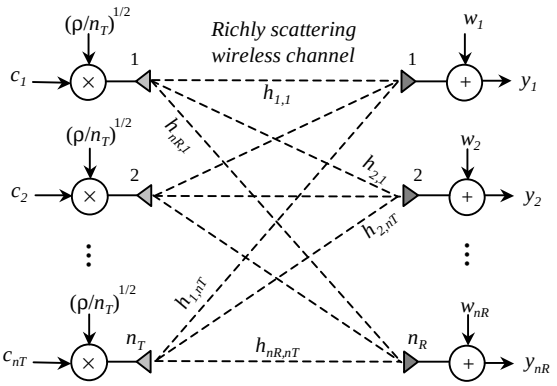



Fig. 1. An $(n_T, n_R)$ MIMO channel.

the mapping of the LST algorithm onto DSP-RAM. Section V describes the implementation of a MIMO receiver onto six different commercially available processors and an efficient realization of LST decoding onto the parallel DSP-RAM architecture. Finally, Section VI makes some concluding remarks.

## II. MIMO Systems and Decoding Techniques

Figure 1 illustrates the model of a MIMO channel between $n_T > 1$ transmitter antennas and $n_R > 1$ receiver antennas, which collectively will be called an $(n_T, n_R)$ *MIMO system*. The input/output relations over a MIMO channel can be represented using complex baseband vector notation as follows:

$$\mathbf{y} = \sqrt{\frac{\rho}{n_T}}\,\mathbf{H}\,\mathbf{c} + \mathbf{w} \tag{1}$$

where $\mathbf{c} = (c_1, c_2, \ldots, c_{n_T})^T$ denotes the vector of transmitted symbols (collectively called a *space-time* (ST) symbol) drawn from a finite complex signal constellation $\mathbb{Q}$, $\mathbf{y} = (y_1, y_2, \ldots, y_{n_R})^T$ denotes the corresponding vector of received symbols, and $\mathbf{w} = (w_1, w_2, \ldots, w_{n_R})^T$ denotes an *additive white Gaussian noise* (AWGN) vector. Vector $\mathbf{w}$ comprises statistically-independent, normally-distributed variables with equal variance. The random Rayleigh channel model in a flat-fading, richly scattering environment with no line-of-sight can be represented as an $n_R \times n_T$ matrix $\mathbf{H} = [\mathbf{h}_1, \ldots \mathbf{h}_{n_T}]$, where $\mathbf{h}_k$ is the column vector of complex transfer gains from the $k$-th transmitter antenna to all $n_R$ receiver antennas. Each entry $h_{ij}$ in $\mathbf{H}$ denotes the gain from transmitter antenna $j$ to receiver antenna $i$. The $h_{ij}$ are usually modeled as *independent identically-distributed* (i.i.d), complex Gaussian random variables of zero mean and variance 0.5 per dimension [1]. Finally, $\rho$ denotes the SNR per receiver antenna defined as the ratio of the total transmitted power per channel used divided by the per-component noise variance. The power launched by each transmitter is normalized by $n_T$ so that the total radiated power is constant and independent of $n_T$ for a fixed SNR.

As given in Equation (1), the signal at each receiver antenna is the superposition of transmitted symbols scaled by the channel gain and corrupted by AWGN. In the receiver, the $n_T$ components of a transmitted ST symbol $\mathbf{c}$ must be recovered from the received signal vector $\mathbf{y}$ and the previously-estimated channel matrix $\mathbf{H}$ [17]. If the multipath scattering is sufficiently rich, the transmitted signals are scattered slightly differently since they originate from different transmitter antennas and propagate over different paths. Consequently, if the MIMO system equations are sufficiently independent, a decoding algorithm at the receiver can recover the symbols despite the multitude of interferers.

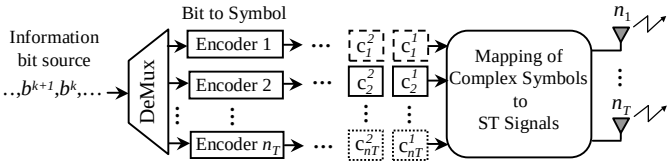



Fig. 2. LST transmitter architecture.

Several techniques have been proposed for recovering the symbols transmitted by $n_T$ antennas [2], [3], [4], [5], [6], [7], [18]. From estimation theory, the optimal decoding method with respect to the error rate is *maximum likelihood* (ML), where the receiver considers all possible ST symbols that could have been transmitted, i.e.,

$$\hat{\mathbf{c}} = \underset{\mathbf{c}\in\mathbb{Q}^{n_T}}{\arg\min} \|\mathbf{y} - \mathbf{H}\,\mathbf{c}\| \tag{2}$$

where the minimum is sought over all possible ST symbols $\mathbf{c} \in \mathbb{Q}^{n_T}$ and $\|\cdot\|$ denotes the Euclidean norm. Because ML decoding requires an exhaustive search over a typically large set $\mathbb{Q}^{n_T}$, its computational complexity can be high, and probably prohibitive when many antennas and/or high-order modulations are used. Even though the complexity of ML decoding is often too great, the algorithm clearly has potentially exploitable parallelism when calculating $\|\mathbf{y} - \mathbf{H}\,\mathbf{c}\|$ over all possible $\mathbf{c} \in \mathbb{Q}^{n_T}$. If the algorithm could be parallelized and mapped onto a parallel computer architecture with $n_P$ PEs, the throughput could be directly multiplied by $n_P$. Sphere decoding (SD) is a promising approach which attempts to prune the search space rather than searching over all possible points. However, SD is inherently an irregular algorithm [18] and mapping an algorithm that has a complicated control sequence onto a data-parallel processor decreases the efficiency [19]. Most practical systems use heuristic decoding techniques [8], [9].

The LST MIMO architecture [2], [3] introduces temporal and spatial diversity into the transmitted signals. A block diagram of a conventional single-user LST system is shown in Figure 2. The incoming information bits are denoted by $\{b^k\}$, where $k$ is a discrete time index. The high-rate data stream is demultiplexed into $n_T$ equal-capacity parallel substreams, called *layers*. Each layer is then encoded separately using the same constellation. If a block of information consists of $L$ space-time symbols, the output of the $n_T$ encoders can be represented by the following $(n_T \times L)$ ST codeword matrix $\mathbf{C}$:

$$\mathbf{C} = \begin{bmatrix} c_1^1 & c_1^2 & \ldots & c_1^L \\ \vdots & \vdots & \vdots & \vdots \\ c_{n_T}^1 & c_{n_T}^2 & \ldots & c_{n_T}^L \end{bmatrix} \tag{3}$$

Matrix $\mathbf{C}$ comprises the symbols that are transmitted by $n_T$ transmitter antennas at $L$ different time instants. The resulting ST signals drive identical transmitter pulse filters and the resulting analog baseband signals are modulated by a carrier and broadcast by $n_T$ antennas. All transmitter antennas are assumed to use the same constellation and to transmit data simultaneously using the same carrier frequency and symbol timing in the same frequency band. The capacity of the wireless channel under these conditions grows linearly with $\min(n_T, n_R)$ [1].

The LST receiver algorithm consists of two phases [3]: First, the channel matrix is estimated [20]. The channel state information is assumed to be known to the receiver but not to the transmitter [17]. Second, the received data signals from one ST symbol interval are processed to recover the $n_T$ transmitted complex symbols, $(c_1^j, \ldots, c_{n_T}^j)$, within a fixed number of symbol times after time $j$.

In a LST MIMO architecture, for a given ST symbol $\mathbf{c}$, a divide-and-conquer decoding strategy, called *nulling* and *cancellation*, has been proposed [2]. The decoding algorithm proceeds iteratively through the following three steps until all $n_T$ symbols are recovered.

Step 1) **Interference nulling:** Interference nulling tries to reduce the amount of interference towards $c_k$ by multiplying the received signal $\mathbf{y}$ by a *nulling vector* $\mathbf{g}_k$ [4]. Consequently, the symbol rate processing has a computational complexity of only $\mathcal{O}(n^2)$. In ZF-LST, $\mathbf{g}_k$ can be calculated using the $k$-th row of $\mathbf{G} = \underline{\mathbf{H}}_k^{\dagger}$ where the notation $\underline{\mathbf{H}}_k$ denotes the matrix obtained by zeroing column $k$ of $\mathbf{H}$ and the superscript $\dagger$ denotes the Moore-Penrose pseudo-inverse operator with a computational complexity of $\mathcal{O}(n^3)$ [21]. Since all components of a transmitted vector $\mathbf{c}$ are assumed to utilize the same constellation, the component $c_j$ with the lowest post-detection SNR will dominate the error performance of the detection process. It was shown in [2] that starting with the symbol (layer) with the strongest post-detection SNR and then proceeding successively to the symbol with the weakest SNR improves the performance remarkably [3]. This corresponds to choosing the row $\mathbf{g}_k$ of $\mathbf{G} = \underline{\mathbf{H}}_k^{\dagger}$ with the minimum norm, $k = \min_i \|\mathbf{g}_i\|^2$, and selecting the corresponding row as the nulling vector in the interference nulling step. Thus, the $k$-th element of $\mathbf{c}$ with the highest SNR is detected by $\hat{c}_k = \mathbf{g}_k\,\mathbf{y}$.

Step 2) **Symbol decoding:** Symbol $c_k$ from the $k$-th transmitter antenna is estimated by mapping to the nearest symbol $c_k = \mathcal{Q}(\hat{c}_k)$ in the constellation, where $\mathcal{Q}$ is the quantization function appropriate to the constellation $\mathbb{Q}$ in use.

Step 3) **Symbol cancellation:** At this stage, the recovered symbol $c_k$ can be used to improve the estimate of the remaining $n_T - 1$ symbols that are yet to be recovered. The interference on the $n_T - 1$ other signals due to $c_k$ can be subtracted out from the received signal as $\mathbf{y}' = \mathbf{y} - c_k\mathbf{h}_k$. Thus, the number of signals remaining to be detected is reduced by one with each decoding step.

There are other published LST decoding algorithms with various computational complexities [1]. Different methods are used to calculate the nulling vectors. To improve the SER performance, especially for mid-range SNR values, the nulling vectors can be obtained using the MMSE algorithm [1]. MMSE nulling yields the nulling matrix $\mathbf{G} = (\mathbf{H}^H\mathbf{H} + 1/\rho\,\mathbf{I})^{-1}\mathbf{H}^H$. The computational complexity of this algorithm is similar to ZF-LST but the SNR $\rho$ has to be determined at the receiver. The square root algorithm [6] was proposed to calculate the nulling vectors using unitary transformations instead of repeatedly evaluating the pseudo-inverse of the reduced channel matrices. Even though the square root algorithm offers an order of magnitude reduction in the computational complexity compared to MMSE-LST decoding, it is shown in [5], [6] that for a typical number of antennas (e.g., between one to eight) the complexity of the algorithm in floating-point operations (FLOPs) is almost the same as for other LST decoding algorithms. The ordered QR decoding technique proposed in [7] uses the modified Gram-Schmidt (MGS) algorithm. In this approach, rather than explicitly calculating the nulling vectors, the channel matrix is decomposed and then optimal ordering is applied to the MGS algorithm to perform the LST decoding.

The required SER performance and decoding rate are two important decoding algorithm metrics. However, the microarchitecture of the target processor can greatly influence decoding algorithm running times so one must be careful to choose the most appropriate decoding algorithms for different processors. In either of the above LST decoding algorithms, there is much data-level parallelism that we propose to exploit using a moderately-parallel architecture. Even though both the square-root and ordered QR algorithms provide better numerical stability and less computational complexity than the MMSE-LST algorithm [6], [7], they exhibit load imbalance and hence a less efficient hardware utilization when they are mapped to a linear parallel architecture [21]. Therefore, we used MMSE-LST as the default decoding technique. However, our parallel implementation, described below, could be modified to accomodate any of the above decoding algorithms.

## III. A Parallel Architecture for Digital Signal Processing

In many indoor wireless environments in the presence of fading, high symbol rates imply that the recovery process will be computationally intensive. As an efficient alternative to using a single, high-performance processor to achieve real-time decoding, multiple simpler processors can be considered to exploit the data-level parallelism in the decoding algorithms. DSP-RAM is a processor-in-memory (PIM) SIMD processor consisting of a linear array of $n_P$ simple fixed-point PEs, as shown in Figure 3. Each PE consists of a data path, containing data registers and functional units, and a local memory. The DSP-RAM controller is a state machine that broadcasts micro-instructions to the PEs, exchanges data with the PEs (described later), and interfaces to the host processor. The DSP-RAM controller broadcasts one instruction stream and all PEs execute the same instructions in lock-step over multiple instances of data stored in their local memories. The DSP-RAM architecture is readily scalable from its HDL specification, and more processing elements can always be implemented as required to achieve any higher symbol decoding rate.

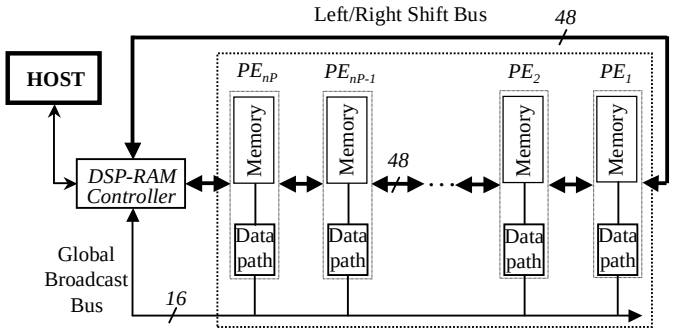



Fig. 3. DSP-RAM architecture.

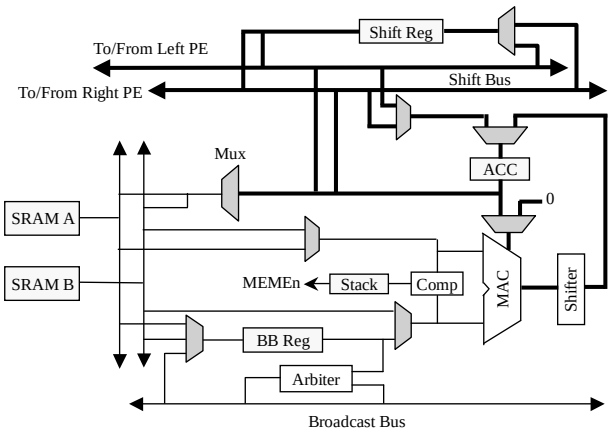



Fig. 4. Architecture of one processing element.

The DSP-RAM architecture provides an efficient processing platform for implementing many algorithms with data-level parallelism [13], [14]. If the algorithm scales well on the linear array of the parallel architecture, the processing throughput will be increased directly by increasing the degree of parallelism (i.e., the number of PEs). More PEs implies more transistors on the chip. However, the processor can often operate at a slower clock frequency and still meet the required processing performance. As well as lowering the clock frequency, one might also choose to lower the power supply voltage. Since power consumption is proportional to the square of the power supply [16], the possibility of reducing the voltage in DSP-RAM would be attractive for power-constrained processing platforms.

PIM-style architectures like DSP-RAM can offer the following advantages [15]:

- Internal memory accesses are usually much faster than external memory accesses.
- For high data rate applications, the restricted bandwidth to external memory tends increasingly to limit the overall performance. In the PIM-style architecture, the processor can directly-exploit the very large (e.g. 1024-bit) bus width at the sense-amplifiers of the internal memory blocks.
- Both the capacity and word width of custom on-chip memories is adjustable to any convenient value. There is no need to conform with standard memory configurations.
- System power consumption is reduced because fewer external memory accesses are generated. Such accesses consume significant power when driving the relatively high-capacitance of off-chip buses.

The architecture of one PE in DSP-RAM is shown in Figure 4. Each PE stores data in its own local memory, which we assumed to be partitioned into two banks, labeled SRAM A and B. Therefore two operands can be fetched simultaneously from memory. We assumed a word width of 16 bits but this can be easily adjusted. The core of each PE is a pipelined multiplier accumulator (MAC). Two 16-bit operands can be multiplied and the 32-bit product can be added to the content of the 48-bit accumulator (ACC) register through the adder/subtractor. Use of extended precision (48-bit) inner product accumulation reduces the rounding error and allows for more accumulation steps without risk of overflow. Since the MAC is in the critical path, the number of pipeline stages in the MAC can be adjusted to achieve different throughputs. The shifter can perform a logical shift, an arithmetic shift or no shift before the MAC output is loaded into the ACC register. As an example, to execute the `MAC A,B` instruction two 16-bit operands are read from the memory banks and then multiplied in the MAC unit. Since the adder is enabled and the shifter is disabled by the controller, the 32-bit product is summed into the 48-bit accumulator value and stored. The result can then be kept in the accumulator, written back into the local memories, or written onto the shift bus to move it into the neighbor's PE shift register. To perform the division and square-root operations, rapidly-converging iterative algorithms based on multiplication can be used [22]. To perform division, the Goldschmidt algorithm takes advantage of the pipelined multiplier to permit division in $\lceil \log_2 16 \rceil = 4$ iterations [22].

It is possible to temporarily exclude processors in the DSP-RAM from executing an instruction depending on certain logical conditions. A comparator and stack are provided in each PE to support if-then-else conditional control flows. The depth of the hardware stack is configurable to allow different nesting depths in the program. The MEMEn signal enables the local memories in each PE. The arbiter module provides a global minimum compare operation (assuming a wired-AND implementation of the global broadcast bus) among all the PEs to speed up the merging of local results from the PEs into a single global result.

A simple communication network includes local left and right shift busses between adjacent PEs and a global broadcast bus. Data can be transferred into a PE in three ways: First, data can be transferred over the 16-bit wired-AND global broadcast bus from one data source (the host processor or one or more PEs) to all PEs. When the same constant needs to be stored into each PE, the broadcast bus can be used to broadcast the value into each PE's *broadcast bus register* (BBReg). For example, loading the $n_R \times n_T$ complex channel gain coefficients into the PEs local memories requires only $2n_R\, n_T + p$ clock cycles, where $p$ is the number of pipeline stages between the broadcast bus and the local memory banks of the PE. Second, data can be sent to the left or right nearest-neighboring PEs over the 48-bit left/right shift bus, shown in bold arrows in Figure 4. The shift bus can be used

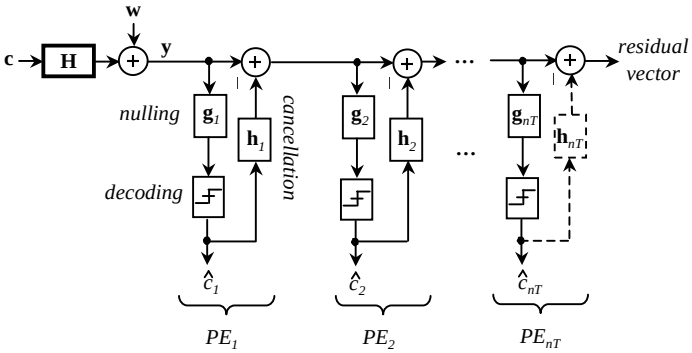



Fig. 5. Dataflow diagram of the symbol decoding process for one ST symbol.

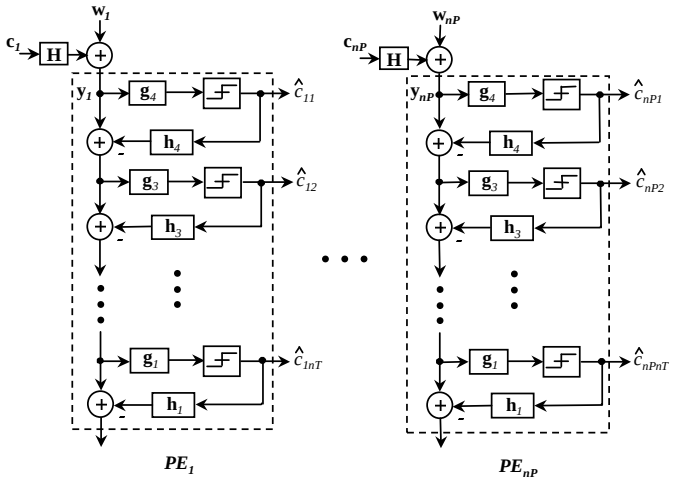



Fig. 6. Dataflow diagram of the symbol decoding process for $n_P$ ST symbols.

to shift three 16-bit operands between PEs. Thus incoming ST symbols can be loaded efficiently into the corresponding PE memories. Note that the operations of writing data into and reading data from the DSP-RAM can be interleaved. For example, previously decoded data may be shifted out to the host processor at the same time that a current block of data is being decoded. Since the I/O path over the shift bus uses only the 48-bit shift register and the shift bus, the memory is free during the shifting of incoming data and, therefore, significant power savings can be realized over a sequential memory load that requires a memory access for each data value [13]. Consequently, shifting of data can occur in parallel with the processing of local data in each PE. Finally, data can be loaded by memory write operations from the DSP-RAM controller or host processor into any locations in the two local memories. This requires one write instruction for every single data word and therefore is relatively slow.

## IV. Mapping LST Decoding onto DSP-RAM

One of the key decisions when implementing LST decoding on DSP-RAM is the mapping of subtasks onto the moderate number of PEs. Decomposing the decoding procedure into finer subtasks and distributing them among the PEs for parallel execution will affect the degree of concurrency and, consequently, the overall throughput. Moreover, since inter-PE communication delay often significantly affects the decoding throughput [23], if we map the calculation onto the PEs in such a way that inter-PE communication is minimized, we can generally expect higher performance and lower dynamic power consumption. Therefore the aim of the mapping is twofold: First, balancing the load among the PEs and increasing the resource utilization by uniformly distributing the decoding of the received ST symbols onto the available PEs. Second, according to the interconnection topology of the DSP-RAM, the decoding algorithm should be mapped onto the PEs to minimize the communication overhead.

We define a *thread* to be a recovery process for a ST symbol that was transmitted in the same symbol time by $n_T$ transmitter antennas. As illustrated in Figure 5, a thread involves $n_T$ nulling steps, $n_T$ decoding steps and $n_T - 1$ cancellation steps. The recovery process can be completed entirely within a single symbol period or can be pipelined over a fixed number of symbol periods. For instance, as shown in Figure 5, a thread can be pipelined across several PEs (similar to the mapping used in [24]). Since all PEs execute the same stream of SIMD instructions, the last redundant cancellation step, shown with dotted lines in Figure 5, must also be performed. In addition, due to the data dependency in the iterative three-step decoding algorithm, the communication penalty can be high. This particular mapping involves at most $n_T$ PEs, implying that the maximum speed-up would be only $n_T$. Therefore, in this mapping the load cannot be uniformly distributed among only a moderate number (e.g. 64) of PEs. A more efficient approach, shown in Figure 6, takes advantage of the distributed memory architecture of the processor and maps a decoding thread to a dedicated PE. Thus the recovery of a ST symbol $\mathbf{y}^k$, where $k$ is the $k$-th vector in a codeword matrix, can be performed by PE$_k$ in the DSP-RAM. In this mapping the last additional cancellation step is not required. The mapping minimizes the inter-PE communication and decouples the number of PEs in DSP-RAM from the number of transmitter/receiver antennas. Therefore, the number of PEs can be scaled based on the desired decoding rate. The greater the number of PEs, the greater the degree of parallelism and the higher the decoding throughput.

The decoding process starts by loading the channel matrix into each PE's local memory using the broadcast bus. For a Rayleigh block-fading channel model, the channel characteristics are assumed to be fixed over constant-sized $L$ data blocks. Therefore the channel gains need to be loaded only once at the beginning of each block. In the next step, the nulling vector calculation can differ depending on the decoding algorithm. The MMSE nulling vectors can be efficiently obtained on DSP-RAM. First, the pseudo-inverse of the deflated channel matrix $\underline{\mathbf{H}}$, namely $\underline{\mathbf{H}}^{\dagger} = (\underline{\mathbf{H}}^H \underline{\mathbf{H}})^{-1} \underline{\mathbf{H}}^H$, has to be calculated $n_T$ times at the beginning of each block. Since the dimensions of $\mathbf{H}$ are typically small, to obtain higher accuracy, we have chosen the direct approach [25] instead of the iterative method to invert the matrix. There are several published techniques for calculating the matrix pseudo-inverse using direct routines [23]. In addition to the strong influence of the DSP-RAM microarchitecture on the complexity of the parallelized pseudo-

inverse algorithm, the structure of the channel matrix $\mathbf{H}$ may also influence the choice of algorithm. Since the channel matrix in practice does not fall into any of the special cases, mapping Cramer's inverse method over $n_R \times n_T$ PEs gives an efficient parallel realization on DSP-RAM. This technique, which is also used in Intel's MMX library [26], requires only one division operation. To calculate the pseudo-inverse of $n_T$ deflated channel matrices, $\underline{\mathbf{H}}$, with the rank ranging from $n_T$ down to 1, $n_T$ iterations are required. In iteration $k$, the $k^2$ cofactors of $\underline{\mathbf{H}}^H\underline{\mathbf{H}}$ can be calculated concurrently in complex arithmetic using a single instruction stream over $k^2$ PEs. Then cofactors are passed among the PEs and each PE calculates the $\underline{\mathbf{H}}^{\dagger} = \left[\frac{1}{det(\underline{\mathbf{H}}^H\underline{\mathbf{H}})}\, adj\,(\underline{\mathbf{H}}^H\underline{\mathbf{H}})\right]\underline{\mathbf{H}}^H$, where $det$ and $adj$ denote the determinant and the adjoint matrix of $\underline{\mathbf{H}}^H\underline{\mathbf{H}}$, respectively. Note that assuming a block-fading channel model, the nulling vectors need to be calculated only once using $n_T$ instances of pseudo-inversion of the deflated channel matrix $\underline{\mathbf{H}}$ at the beginning of each received block.

The process of loading the received ST symbols into the DSP-RAM can be overlapped with the nulling vector calculation. A block of received information is first divided into sub-blocks of size $n_T \times n_P$, where $n_P \ll L$, and the ST symbols of each sub-block are loaded consecutively using the left/right shift bus into the corresponding PE's memory. Since the size $n_P$ of a sub-block is typically much smaller than the block size $L$ for the block-fading wireless channel model, this approach reduces the initial latency of the decoding process considerably.

After loading $n_P$ ST symbols into the PEs' local memories and calculating the nulling vectors, each PE concurrently initializes the iterative three-step decoding process and decodes one received ST symbol. Therefore the time to process $n_P$ threads is equivalent to the processing time of one single thread, hence the throughput is directly increased by a factor of $n_P$. When the parallel decoding is finished, the resulting symbols are shifted out from each PE into the host processor while the next set of $n_P$ signals is shifted into the PEs' local memories. Therefore, the write and read operations are also overlapped. This process of decoding the received ST signals is continued until all of the received symbols in a block are recovered. Decoding of the next block can begin as soon as the channel gain coefficients are estimated at the receiver.

## V. Implementation of a MIMO Receiver

To determine the efficiency and compare the performance of the parallel implementation of LST decoding on the moderately-parallel DSP-RAM architecture with conventional implementations on contemporary DSPs, a (4,4) MIMO system with 16-QAM modulation was modelled for six different processors. The processor specifications are summarized in Table I. The ARM7TDMI is an embedded RISC processor with very low power consumption on a small die size that does not have any DSP-specific features [27]. The SA-110 uses architectural enhancements beyond the original ARM processor to execute at rates far exceeding those of the ARM7TDMI. The PXA255 processor from Intel Corp. is a 32-bit super-pipelined 16-bit SIMD processor intended to enhance audio/video decoder performance [28]. The VLIW-based TMS320C6416T-720 from Texas Instruments Inc. is a high-performance signal processor that contains two identical fixed-point data-paths [29]. The TMS320C6713-225 is a family of 32-bit floating-point DSPs that target applications such as 3-D graphics, radar and speech recognition. Analog Devices' ADSP-TS203 is optimized for demanding multiprocessor DSP applications such as communication infrastructure [30]. This processor supports both fixed-point and floating-point computations.

TABLE I
PROCESSOR SPECIFICATIONS

| Processor | Data Width | Instr. Width | Clock Freq. (*MHz*) | Core Volt. (*V*) | Core Power (*mW*) | Proc. Tech. ($\mu m$) |
|---|---|---|---|---|---|---|
| ARM7TDMI | 16 | 16/32 | 133 | 1.08 | 332 | 0.13 |
| SA-110 | 32 | 16 | 233 | 2.0 | 1000 | 0.35 |
| PXA255 | 16/32 | 16/32 | 400 | 1.65 | 2598 | 0.18 |
| TMSC6416T | 8/16 | 32 | 720 | 1.2 | 2147 | 0.13 |
| TMSC6713 | 32 | 32 | 225 | 1.26 | 1386 | 0.13 |
| ADSP-TS203 | 32 | 32 | 500 | 1.05 | 2700 | 0.13 |

TABLE II
DSP-RAM PE IMPLEMENTATION CHARACTERISTICS

| Device | Clock Freq. (*MHz*) | Core Volt. (*V*) | Core Power (*mW*) | Proc. Tech. ($\mu m$) |
|---|---|---|---|---|
| Virtex E | 98 | 1.8 | 970.24 | 0.18 |
| VirtexII | 120 | 1.5 | 968.48 | 0.13 |
| VirtexII Pro | 130 | 1.5 | 1057.99 | 0.13 |

The regular architecture of FPGAs is a convenient platform for prototyping the linear DSP-RAM architecture. Contemporary FPGAs integrate megabytes of memory with multiple millions of equivalent logic gates arranged in a two-dimensional array of configurable logic slices [31]. A 64-PE DSP-RAM system was synthesized from a Verilog HDL description to various FPGAs. Each PE contained 512 bytes of local memory for implementations on the Virtex-E XCV3200E-7-FG1156, Virtex-II XC2V8000-5-FF1517 and Virtex-II Pro XC2VP125-5-FF1704 FPGAs. Table II shows the clock frequency and core power consumption metrics for one processing element. Table III shows the resource utilization of the implementations of the DSP-RAM PE on the different FPGAs. For the FPGA implementations, single-port Block RAMs (BRAMs) were synthesized and the maximum pipelined multipliers were implemented in look-up tables (LUTs). Since there are typically alternative arithmetic circuit implementations with different maximum clock rates, areas and power dissipation values, one may choose different cells from the FPGA vendor's component library to meet the target application requirements. The same DSP-RAM design was also synthesized for a 0.18-$\mu m$ TSMC CMOS technology standard cell implementation [32]. Figure 7 shows the layout of the 20.65 $mm^2$ 64-PE DSP-RAM chip. The estimated core power consumption is 621-mW when the DSP-RAM operates at 100-MHz.

To develop the LST MIMO receiver algorithm, we used the following three-step design flow:

TABLE III
RESOURCE UTILIZATION BY DSP-RAM PE IMPLEMENTATION

| Device | 4-input LUTs | Slice FFs | Slices | BRAMs |
|---|---|---|---|---|
| Virtex E | 1220 (0.0187%) | 619 (0.0095%) | 713 (0.0219%) | 2 (0.0096%) |
| VirtexII | 1186 (0.0127%) | 620 (0.0066%) | 684 (0.0146%) | 2 (0.0119%) |
| VirtexII Pro | 1133 (0.0119%) | 616 (0.0055%) | 666 (0.0119%) | 2 (0.0035%) |

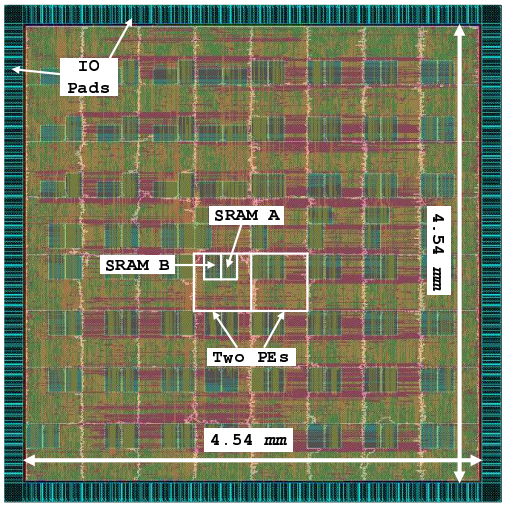



Fig. 7. Chip Plot of a 64-PE DSP-RAM in 0.18-$\mu m$ CMOS.

(1) MATLAB programs that model different LST decoding algorithms for an $(n_T, n_R)$ MIMO system were written and verified in simulation. The variance of the AWGN noise vector was normalized by

$$\sigma_w^2 = \frac{n_T \times E_{sav}}{2 \log_2 q} \times 10^{-\rho_{\mathrm{dB}}/10}, \tag{4}$$

which is the noise energy per bit. The average energy per bit was thus normalized to 1. Here $\log_2 q$ is the number of transmitted bits per constellation point, $\rho_{\mathrm{dB}}$ is the SNR in dB, and $E_{sav} = 2(q-1)/3$ is the mean symbol energy of the $q$-QAM constellation. Figure 8 plots the SER versus SNR simulation results for five different LST decoding algorithms for a (4,4) MIMO system that utilizes 16-QAM modulation over a flat Rayleigh fading channel. The ordered QR method requires one order of magnitude less computational complexity than the MMSE-LST decoding algorithm, but it has degraded SER performance due to the sub-optimal ordering employed in the MGS calculation.

(2) Floating-point and fixed-point versions of the LST decoding algorithm were developed in C++, and these implementations were optimized and verified for six target processors. Considering the time and accuracy objectives, the LST decoding algorithm was expressed using complex arithmetic. Differences in the computer architecture, available programming languages, and compiler quality can make large differences in the way one implements a decoding algorithm. A parallel implementation of the LST decoding algorithm was developed

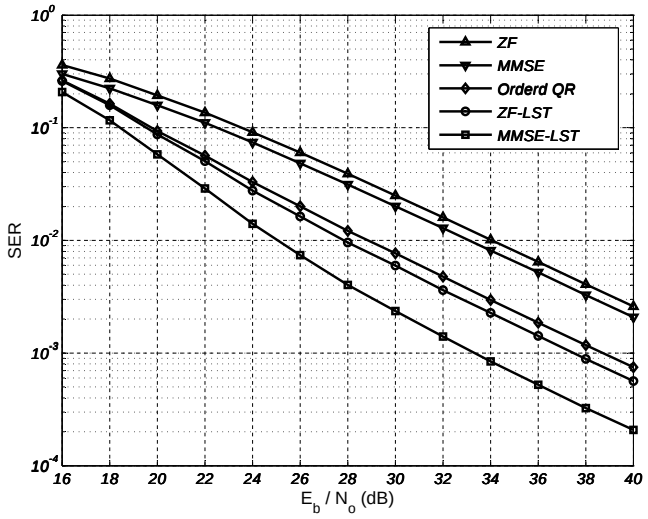



Fig. 8. Symbol error rate vs. SNR for ZF, MMSE and three different LST algorithms, for a (4,4) MIMO system utilizing 16-QAM modulation over a Rayleigh flat-fading channel.

on a DSP-RAM C++ emulator. Our emulator provides a debugging environment for a fixed-point implementation of the algorithm on the parallel DSP-RAM architecture and reports the exact clock cycle count required to execute a program, including the I/O and inter-PE communication cycles. This count can be used to choose an appropriate clock frequency for the DSP-RAM to achieve real-time decoding. During implementation of the parallel LST decoding algorithm for the emulator, the developer is entirely responsible for controlling and efficiently utilizing the available functional units and for avoiding any possible structural hazards. The compiled code produced from a high-level language program implementation of LST decoding can be less efficient than expertly optimized assembly language code. Similarly, our optimized micro-coded implementation of LST decoding on DSP-RAM probably has an efficiency advantage over compiled code even when, as we did, all optimization features of the compiler enabled. Also, the PIM-style architecture of DSP-RAM requires many fewer clock cycles to access the on-chip memory banks than those required by conventional processors for off-chip memory accesses.

Table IV gives the number of clock cycles required to decode received ST signals and the number of clock cycles required to calculate the nulling vectors for a $4 \times 4$ channel matrix. The reason that the number of clock cycles to decode 64 ST signals does not increase linearly with the number of ST signals is that most of the DSPs utilize some degree of parallelism in their instruction set architecture. For example, the ADSP-TS203 DSP provides a parallel core that can execute eight 16-bit MACs with 40-bit accumulation in one clock cycle. Also, the TMS320C6416T DSP contains two identical fixed-point data-paths. DSP-RAM can provide a much greater degree of parallelism (e.g., $n_P = 64$) than these DSPs and, therefore, the data-parallel LST decoding algorithm can scale efficiently on the linear architecture of DSP-RAM. Thus the number of clock cycles to decode 64 ST symbols is not much more than the number required to decode one received signal

TABLE IV
CYCLE COUNTS TO DECODE RECEIVED ST SIGNALS AND CALCULATE NULLING VECTORS OF A $4 \times 4$ CHANNEL MATRIX

| Processor | Decode One ST Symbol | Decode 64 ST Symbols | Calc. Nulling Vectors |
|---|---|---|---|
| ARM7TDMI | 4,798 | 206,396 | 43,641 |
| SA-110 | 3,124 | 146,698 | 33,701 |
| PXA255 | 3,732 | 155,602 | 22,698 |
| TMSC6416T | 1,812 | 87,714 | 25,340 |
| TMSC6713 | 1,814 | 90,386 | 17,572 |
| ADSP-TS203 | 1,555 | 70,455 | 13,216 |
| DSP-RAM | 1,078 | 1,174 | 12,742 |

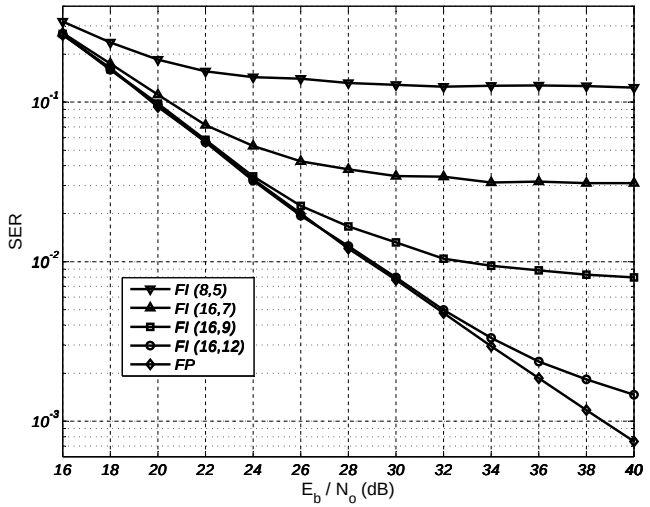



Fig. 9. SER vs. SNR for floating-point (FP) and four fixed-point number representations.

and hence the throughput is increased by a factor of $64$. To calculate the 4 nulling vectors of a $4 \times 4$ channel matrix, one must perform four pseudo-inversions of the reduced channel matrices, with the rank reducing from 4 to 1. For a DSP-RAM implementation, the pseudo-inverse operation can be performed efficiently in parallel using the mapping described in Section IV over 16 PEs. However, since the required degree of parallelism is only $n_R \times n_T$, the number of clock cycles required to calculate 4 nulling vectors in a DSP-RAM implementation is close to the number required by the slightly parallel ADSP-TS203 DSP.

(3) Once the design was completed on the emulator, test vectors were designed and used as the stimulus to the HDL model. After the algorithm was verified in simulation, it was synthesized for the target FPGA. Figure 9 plots the SER versus SNR results of the ordered QR-LST decoding for the various number representations of the MIMO system reported in Figure 8. The product word length and sum word length were set to 32 and 48, respectively, for all implementations. Note that a 12-bit fraction field, labeled as $FI(16, 12)$ in the figure, achieves almost the same SER performance as the floating-point implementations up to a SNR of 32 dB.

Consider a (4,4) MIMO system that exploits spatial multiplexing using an LST architecture and 16-QAM modulation.

TABLE V
DECODING THROUGHPUT OF THE LST DECODER IMPLEMENTATIONS FOR A (4,4) MIMO SYSTEM UTILIZING 16-QAM MODULATION, WITH THE BLOCK-FADING CHANNEL MODEL ASSUMPTION, AND $T_b = 100$ *ms*

| Processor | Clock Freq. (*MHz*) | Max. Decoding Throughput (*Mb/s*) |
|---|---|---|
| ARM7TDMI | 133 | 0.64 |
| SA-110 | 233 | 1.58 |
| PXA255 | 400 | 2.56 |
| TMSC6416T | 720 | 8.2 |
| TMSC6713 | 225 | 2.48 |
| ADSP-TS203 | 500 | 7.09 |
| DSP-RAM | 100 | 85.07 |

We will make the common assumption of symbol-synchronous receiver sampling and ideal timing recovery. Also, for a typical indoor wireless environment with a maximum Doppler frequency of $f_m = 3$ *Hz*, the coherence time of the channel can be calculated as $T_c = 0.423/f_m$ [33]. Assuming a block-fading wireless channel, where the channel response is almost invariant during the coherence time of the channel, the block duration can be chosen to be $T_b = 100$ ms. Let $k_n$ denote the number of clock cycles required to compute $n_T$ nulling vectors. To achieve real-time decoding, $\frac{k_n+(L_d \times k_d)}{f_p}$ should be less than or equal to $T_b$, where $k_d$ is the number of clock cycles required to decode one received ST signal, $L_d$ is the number of ST data symbols in a block of length $L$, and $f_p$ is the clock frequency of the receiver signal processor. Assume that each block is divided into a set of 64 ST symbols frames. Since most conventional processors utilize some degree of parallelism in their instruction set architecture, we introduce $k_{d64}$ to denote the number of clock cycles required to decode one frame of 64 ST symbols. Therefore, the number of ST symbols in a block can be written as:

$$L_d = \frac{K - k_n}{k_{d64}} \times 64 \tag{5}$$

where $K = T_b \times f_p$ is the total clock cycle budget for $T_b$ periods. Equation 5 shows that in addition to the clock frequency of the signal processor, the efficiency of the processor when executing the LST decoding algorithm has a great influence on decoding performance. To maximize the decoding throughput, the denominator of Equation 5 should be minimized. An efficient way to achieve this goal is to map the decoding algorithm over an array of PEs that operate concurrently. The results in Table V show that a 64-PE DSP-RAM can decode at more than 10 times the bit rate of a high-performance conventional DSP processor. The higher throughput is achieved even though DSP-RAM's assumed 100-MHz clock frequency is much smaller than that of the 720-MHz TMSC6416T DSP.

## VI. CONCLUSIONS

A PIM-style moderately-parallel architecture, called DSP-RAM, has been evaluated as an alternative to commercial DSPs and has been synthesized to implement parallel LST

MIMO receiver algorithms. Since the structure of LST decoding algorithms scales very well on the linear array of PEs in DSP-RAM, the data throughput can be readily increased by using more PEs. The performance results demonstrate that a 64-PE 100-MHz DSP-RAM can potentially provide more than 10 times greater decoding throughput in a typical indoor environment compared to a high-performance 720-MHz DSP processor. The significant speed-up of the parallel DSP-RAM architecture can be exploited to permit a lower operating clock frequency and/or a lower operating voltage, which would have the further benefit of lowering the power consumption.

## REFERENCES


[1] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge University Press, 2003.

[2] G. J. Foschini and M. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," in *Proc. of Wireless Personal Communications*, March 1998, pp. 311–335.

[3] G. Golden *et al*., "Detection algorithm and initial laboratory results using the V-BLAST space-time communication architecture," *Electronic Letters*, vol. 35, no. 1, pp. 14–15, January 1999.

[4] S. Loyka and F. Gagnon, "Performance analysis of the V-BLAST algorithm: an analytical approach," *IEEE Transactions on Wireless Communications*, vol. 3, no. 4, pp. 1326–1337, July 2004.

[5] J. Benesty, Y. Huang, and J. Chen, "A fast recursive algorithm for optimum sequential signal detection in a BLAST system," *IEEE Transactions on Signal Processing*, vol. 51, no. 7, pp. 1722–1730, July 2003.

[6] B. Hassibi, "An efficient square-root algorithm for BLAST," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 1999, pp. 737–740.

[7] D. Wubben *et al*., "Efficient algorithm for detecting layered space-time codes," in *4th International ITG Conference on Source and Channel Coding*, January 2002, pp. 399–405.

[8] Z. Guo and P. Nilsson, "A low-complexity VLSI architecture for square root MIMO detection," in *IASTED Circuits, Signals and Systems*, May 2003.

[9] D. Garrett *et al*., "A 28.8 Mb/s 4x4 MIMO 3G high-speed downlink packet access receiver with normalized least mean square equalization," in *IEEE International Solid-State Circuits Conference*, February 2004, pp. 420–536.

[10] A. Adjoudani *et al*., "Prototype experience for MIMO BLAST over third-generation wireless system," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 440–451, April 2003.

[11] P. Murphy *et al*., "An FPGA based rapid prototyping platform for MIMO systems," in *Thrity-Seventh Asilomar Conference on Signals, Systems and Computers*, November 2003, pp. 900–904.

[12] A. Burg *et al*., "FPGA implementation of a MIMO receiver front-end for the UMTS downlink," in *International Zurich Seminar on Broadband Communications*, February 2002, pp. 8–1–8–6.

[13] S. J. Dillen and B. F. Cockburn, "Parallel filtering and thresholding of images on the SIMD DSP-RAM architecture," in *Proc. of IEEE Canadian Conference of Electrical and Computer Engineering*, May 2002, pp. 995–1000.

[14] B. S. Kwan, B. F. Cockburn, and D. G. Elliott, "Implementation of DSP-RAM: An architecture for parallel digital signal processing," in *IEEE Canadian Conference on Electrical and Computer Engineering*, May 2001, pp. 341–346.

[15] M. Gokhale, B. Holmes, and K. Iobst, "Processing in memory: The Terasys massively parallel PIM array," *Computer*, vol. 28, no. 4, pp. 23–31, April 1995.

[16] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, April 1992.

[17] T. L. Marzetta, "BLAST Training: Estimating channel characteristics for high capacity space-time wireless," in *Proc. of the 37th Annual Allerton Conference on Communication, Control, and Computing*, 1999, pp. 958–966.

[18] B. Hassibi and H. Vikalo, "On the expected complexity of sphere decoding," in *Proc. of Conference on Signals, Systems and Computers*, 2001, pp. 1051–1055.

[19] J. Hennessy and D. A. Patterson, *Computer Architecture A Quantitative Approach*. Morgan Kaufmann Publishers, 2003.

[20] D. Samardzija and N. Mandayam, "Pilot-assisted estimation of MIMO fading channel response and achievable data rates," *IEEE Transactions on Signal Processing*, vol. 51, no. 11, pp. 2882–2890, November 2003.

[21] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Johns Hopkins University Press, 1996.

[22] M. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann, 2004.

[23] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation : Numerical Methods*. Prentice-Hall, 1997.

[24] Z. Guo and P. Nilsson, "A VLSI implementation of MIMO detection for future wireless communications," in *IEEE Proceedings on Indoor and Mobile Radio Communications*, 2003, pp. 2852–2856.

[25] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C Example Book : The Art of Scientific Computing*. Cambridge University Press, 1992.

[26] *Intel AP-928: Streaming SIMD Extensions*, Intel, 1999.

[27] *ARM7TDMI Technical Reference Manual*, ARM, 2001.

[28] *Intel PXA255 Processor: Developer's Manual*, Intel, 2004.

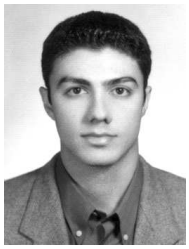
[29] *TMS320C6000 CPU and Instruction Set Reference Guide*, Texas Instruments, 2000.

[30] *TigerSHARC Embedded Processor*, Analog Devices Inc., 2003.

[31] *Virtex-II Pro™ Platform FPGAs: Functional Description*, January 2003.

[32] *TSMC 0.18-$\mu$m process technology*, TSMC, 2003.

[33] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 2002.

**Amirhossein Alimohammad** (S'01) received his B.Sc. in Computer Engineering from the University of Isfahan and his M.Sc. in Computer Architecture from the Department of Electrical and Computer Engineering at the University of Tehran, in 1995 and 1998, respectively. He was employed by Informatics Services Corporation (ISC) between 1998-2000 while he was a Ph.D. student at the Sharif University of Technology. In 2001, Amir was with the Electronic and Microelectronic Department at the University of Ulm, Atmel Germany GmbH, and Get2Chip GmbH in Munich, Germany. He is currently working toward his Ph.D. degree in the Department of Electrical and Computer Engineering at the University of Alberta, Canada. His research interests include logic-enhanced memory architectures, parallel signal processing for wireless applications, reconfigurable computing, and digital system testing.

**Bruce F. Cockburn** (S'86–M'90) received the B.Sc. degree in engineering physics from Queen's University, Kingston, ON, Canada in 1981, and the M.Math. and Ph.D. degrees in computer science from the University of Waterloo, Waterloo, ON, Canada, in 1985 and 1990, respectively. He is currently an Associate Professor in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. He is also a Scientist at Telecommunications Research Laboratories, Edmonton. His research interests include VLSI design and test, parallel signal processing, iterative decoders, and computer architecture. Dr. Cockburn is a member of the IEEE Computer Society, the IEEE Communications Society, the IEEE Solid-State Circuits Society, and the Association for Computing Machinery. He is also registered as a Professional Engineer in the Province of Alberta, Canada.