Efficient VLSI Architectures for Baseband Signal Processing in Wireless Base-Station Receivers*

Sridhar Rajagopal Srikrishna Bhashyam Joseph R. Cavallaro Behnaam Aazhang
Rice University
Center for Multimedia Communication
Department of Electrical and Computer Engineering MS-366
6100 Main St., Houston, TX 77005-1892.
{sridhar,skrishna,cavallar,aaz}@rice.edu

Abstract

A real-time VLSI architecture is designed for Multiuser Channel Estimation, one of the core baseband processing operations in wireless base-station receivers. Future wireless base-station receivers will need to use sophisticated algorithms to support extremely high data rates and multimedia. Several features in these algorithms that can help meet real-time requirements are not utilized effectively in DSPs. These features, such as bit level arithmetic and parallel structure, can be revealed and well exploited by task partitioning the algorithms. We modify the channel estimation algorithm for a reduced complexity fixed-point hardware implementation. We show the complexity and hardware required for three different area-time tradeoffs: an area-constrained, a time-constrained and an area-time efficient architecture. The area-constrained architecture achieves low data rates with minimum hardware, which may be used in 'picocell' base-stations. The time-constrained solution exploits the entire available parallelism and determines the maximum theoretical data rates. The area-time efficient architecture meets real-time requirements with minimum area overhead. The orders-of-magnitude difference between area and time constrained solutions reveals significant inherent parallelism in the algorithm. All VLSI solutions exhibit better time performance than a previous DSP implementation.

1. Introduction

Next generation wireless communication systems [11] have been designed to integrate features such as high data rates varying up to 2 Mbps, Quality-of-Service (QoS) guarantees and multimedia in the existing communication framework. This requires the implementation of highly sophisticated and complex algorithms in real-time. There is a strain on existing hardware resources to meet the requirements of these algorithms. Many algorithms proposed for next generation communication receivers, which are designed to give good performance in terms of error rates, are discarded for a direct implementation as they have high computational complexity. A typical DSP or general purpose processor implementation [5] is unable to fully exploit the parallelism and bit level computations available in these algorithms. Hence, there is a need to efficiently decompose these algorithms into different tasks and study their mapping on hardware to accelerate their implemen-

*This work was supported in part by Nokia Corporation, Texas Instruments Inc., the Texas Advanced Technology Program under grants 1997-03604-044 and 1999-003604-080, and by NSF under grants NCR-9506681 and ANI-9979465.

tation. The algorithms are also computationally expensive, involving subroutines such as matrix inversions, which require floating point accuracy.

We develop VLSI architectures for multiuser channel estimation, one of the most computationally challenging baseband tasks in the base-station receiver. There have been several hardware implementations for multiuser detection [4]. Most implementations either assume perfect channel estimation or assume single user estimation. This assumes that channel estimation can be done in real-time and the data rates are considered to be dependent only on the detector. However, many advanced channel estimation schemes [3, 6] have high computational complexity due to matrix inversions involved and cannot be performed in real-time and typically, simpler single-user sliding correlator structures are used for channel estimation [13]. Jointly performing multiuser channel estimation and detection is shown to have lower computational complexity and better error rate performance than performing estimation and detection separately. We modify the previous channel estimation algorithm [15], based on the maximum likelihood principle and develop an iterative scheme [12], which is computationally effective, suitable for a fixed point implementation and is equivalent to matrix inversion in terms of error rate performance.

Architectures for the mobile handset have similar algorithms for implementation. 'Blind' versions [16] of these algorithms are available for the mobile handset. In this case, the channel is synchronous and only a single user has to be detected. However, the architecture design considerations for the mobile handset has to be power-efficient [9] and this also needs to be accounted for in the design as a critical parameter. In this paper, we concentrate on the base-station receiver and neglect power issues in our considerations.

The organization of this paper is as follows. The next section provides an introduction to Multiuser channel estimation and its real-time requirements. The algorithm is modified for a reduced complexity fixed-point solution, without loss in error rate performance. Section 3 shows the task partitioning of the algorithm and the various area-time trade-offs. Area-constrained, time-constrained and area-time efficient architectures are presented. A comparison is also made with a previous DSP implementation. The conclusions and future directions are presented in Section 4.

2. Multiuser channel estimation

The next generation wireless communication systems [1] use a Wideband Code-Division Multiple Access (W-CDMA) scheme as the multiple access protocol for communication. This scheme uses spread spectrum signaling, where each active user uses a unique signature sequence (spreading code) to modulate the data bits. The base-station receives a 'mixed' version of signals of all active users after they travel through different paths in the channel. These channel paths induce different delays, attenuations and phase-shifts to their signals and the mobility of the users causes these parameters to change over time (called fading). Moreover, the signals from different users interfere with each other (Multiple Access Interference) adding to the Additive White Gaussian noise present in the channel. Multiuser channel estimation refers to the joint estimation of these unknown parameters for all users to mitigate these undesirable effects and accurately detect the received bits of different users.

2.1. Real-time requirements

Data transmission in the next generation wireless systems [11] is done in frames of 10 ms. The data transmission can be done in variable rates depending on the spreading factor (N), as shown in Table 1. The table gives an example of the number of bits in a frame for spreading factors of

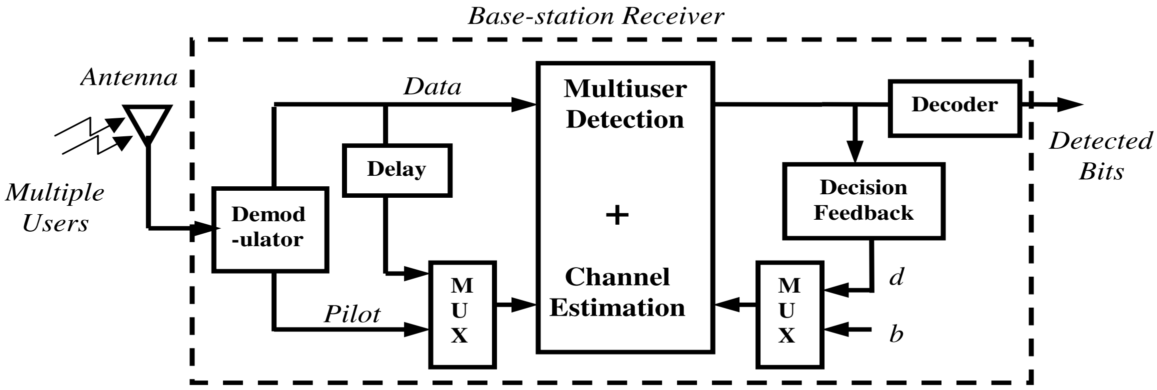

Figure 1. Simplified view of the base station receiver

4, 32 and 256 chips per data bit. We assume Binary Phase Shift Keying (BPSK) modulation. To support real-time, the number of bits detected per frame should be at the rate of transmission. We implement our design assuming a spreading factor of 32 chips per data bit. This implies that the real-time requirement of the joint estimation and detection scheme is to detect input data bits at a rate of 128 Kbps i.e. a bit of every user has to be estimated and detected in less than 7.8125 μs, assuming that the estimation and detection blocks will be pipelined. A spreading factor (N) of 32 can support 32 users (K) and we shall use $N = 32$ and $K = 32$ in our design specifications.

2.2. Channel model

The model for the received signal at the output of the channel [15] can be expressed as

$$\mathbf{r}_i \quad = \quad \mathbf{Y}\mathbf{b}_i + \eta_i \tag{1}$$

where $\mathbf{r}_i \in \mathbb{C}^N$ is the received signal vector due to the bits of all K asynchronous users, spread with a spreading factor N, $\mathbf{b}_i \in \mathbb{R}^{2K} = [b_{1,i-1}, b_{1,i}, \dots, b_{K,i-1}, b_{K,i}]^\top$ are the bits of K users to be detected, $\mathbf{Y} \in \mathbb{C}^{2K \times N}$ is the actual channel, containing information about the spreading codes, attenuation and delays from the various paths, η_i is the noise, which is assumed to be Additive White Gaussian Noise (AWGN) and i is the time index. The channel $\mathbf{Y}$ is to be estimated and used for accurately detecting the received data bits of different users.

2.3. Maximum likelihood channel estimation

The channel estimation and detection block in the base-station receiver is shown in Figure 1. The channel information is obtained by transmission of a pilot signal $\mathbf{b}$, which is a sequence of bits that is known at the receiver. The received pilot signal ($pilot$) is compared with the known bits to form an estimate of the channel. The decisions from the multiuser detection block $\mathbf{d}$ are fed back to the

Table 1. Proposed data rates for next generation communication systems

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

channel estimation block along with the received data bits ($data$), delayed by the time required for detection, for tracking the channel estimates when the pilot signal is absent.

The derivation of the joint estimation and detection algorithm is detailed in [15]. The multiuser channel estimation algorithm is based on the maximum likelihood principle, where the probability of received input given the transmitted bits is maximized. The computations that occur during the estimation phase [15] are:

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

$$\mathbf{R}_{bb} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{b}_i \mathbf{b}_i^T \tag{3}$$

where L is the length of the pilot sequence, $\mathbf{R}_{br} \in \mathbb{C}^{2K\times N}$ is the cross-correlation matrix between the synchronization bits $\mathbf{b}_i$ and the received signal $\mathbf{r}_i$ and $\mathbf{R}_{bb} \in \mathbb{R}^{2K\times 2K}$ is the auto-correlation matrix. The correlation matrices are averaged over a window of length L. The channel estimate can be obtained by solving

$$\mathbf{R}_{bb}\mathbf{Y} = \mathbf{R}_{br} \tag{4}$$

The channel estimate $\mathbf{Y}$ is fed to the detection block for detecting the unknown bits. The detected bits , $\mathbf{d}$, which are obtained at the detection stage, are fed back to the estimation block for tracking purposes for a fading channel and to the rest of the processing blocks in the base-station receiver.

It is difficult to maintain numerical stability for matrix inversion, using decomposition techniques such as QR or LU, with fixed-point. Also, tracking requires the rebuilding of the correlation matrices and computing the inverse every time. This is computationally inefficient as this implies a matrix inversion for every update. Hence, a better scheme is needed for meeting the real-time requirements.

2.4. Iterative scheme for channel estimation

A direct computation of the exact maximum likelihood channel estimate $\mathbf{Y}$ involves the computation of the correlation matrices $\mathbf{R}_{bb}$ and $\mathbf{R}_{br}$, and then the computation of $\mathbf{R}_{bb}^{-1}\mathbf{R}_{br}$ at the end of the preamble (pilot). The computation of the inverse at the end of the preamble is computationally expensive and delays the start of detection beyond the length of the preamble until the estimate has been computed and this delay limits the data rate. In our iterative algorithm, we approximate the maximum likelihood solution based on the following ideas.

1. The product $\mathbf{R}_{bb}^{-1}\mathbf{R}_{br}$ can be directly approximated using iterative algorithms such as the gradient descent algorithm [7].
2. The iterative algorithm can be modified to update the estimate as the preamble is being received rather than waiting till the end of the preamble. This means that the computation per bit can be reduced by spreading it over the entire preamble.

We present an iterative scheme based on the method of steepest descent for the matrix inversion. The channel estimate $\mathbf{Y}$ is updated iteratively every bit and is available immediately after the end of the pilot sequence. The updating of the estimate is done using the iterative scheme as shown

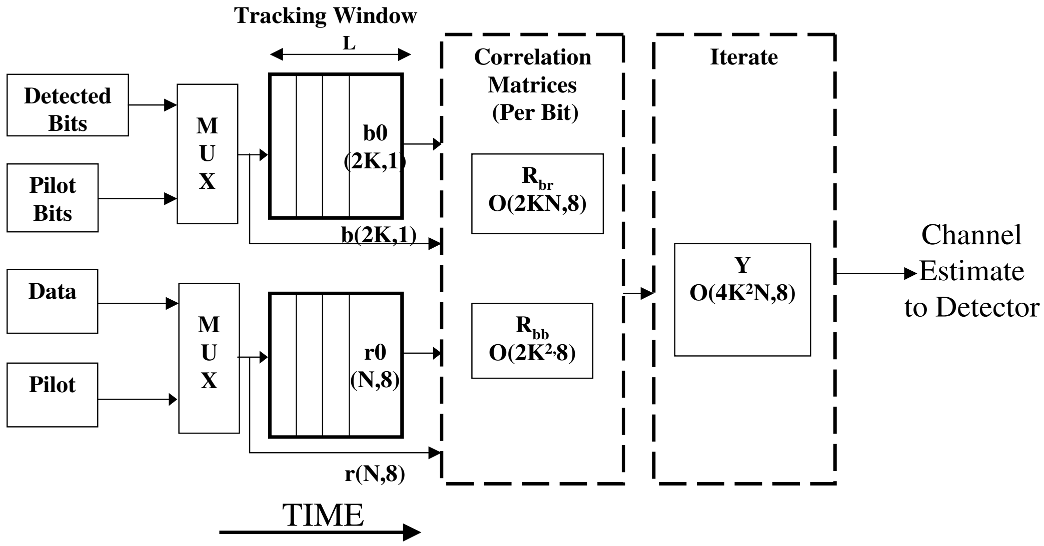

Figure 2. Task decomposition of multiuser channel estimation algorithm

below:

$$
\begin{aligned}
\mathbf{R}_{br} &= \mathbf{R}_{br} + \mathbf{b}_i \mathbf{r}_i^H - \mathbf{b}_{i-L} \mathbf{r}_{i-L}^H && (5) \\
\mathbf{R}_{bb} &= \mathbf{R}_{bb} + \mathbf{b}_i \mathbf{b}_i^T - \mathbf{b}_{i-L} \mathbf{b}_{i-L}^T && (6) \\
\mathbf{Y} &= \mathbf{Y} - \mu(\mathbf{R}_{bb} * \mathbf{Y} - \mathbf{R}_{br}) && (7)
\end{aligned}
$$

This scheme is suitable for tracking, which is shown by the removal of the oldest bit in the window of length L as the new bit is received. Tracking is simpler in this iterative scheme as the channel estimates and correlation matrices are updated iteratively. During the initial pilot phase, tracking is absent and the equations for correlation (equations 6, 7) reduce to the previous estimation scheme using inversion (equations 2, 3). Another advantage of this scheme is that it lends itself to a simple fixed-point solution, which was difficult to achieve using the previous inversion scheme. There are no divisions except for the multiplication by the convergence parameter, μ, which can be implemented as a right-shift, by making it a power of two. This can be done as the algorithm is not highly dependent on the exact value of μ. This algorithm shows good convergence behavior as $\mathbf{R}_{bb}$ is a symmetric, positive definite matrix and has a small condition number. The iterative scheme gives the same error rate performance as that of the original scheme and has been verified using simulations [12].

3. Task decomposition and VLSI architecture

The task partitioning of the algorithm into sub-blocks is carried out for pipelining and for utilizing the inherent parallelism present in the algorithm. We implement different mappings of the multiuser channel estimation algorithm to hardware to study the complexity and hardware requirement tradeoffs. We discuss an area-constrained architecture, a time-constrained architecture and an area-time efficient solution.

3.1. Task decomposition

The task decomposition of the algorithm is as shown in Figure 2. The blocks that are pipelined are shown on the horizontal time axis while the blocks that have coarse-grained parallelism are

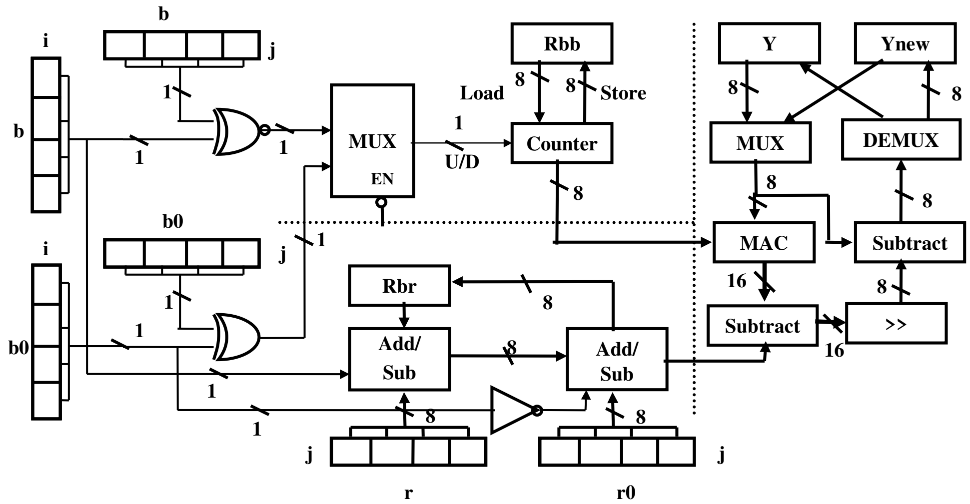

Figure 3. Area-constrained VLSI architecture

shown along the vertical axis. The figure shows that the correlation matrices can be formed in parallel and this can be pipelined with the iteration of the channel estimate matrix. The two multiplexers shown are for selecting between the known pilot and the received pilot signal during the training mode and the detected bits and the received data signal in the tracking phase. The tracking window is the history buffer and keeps the L most recent samples of the bits as well as the received signal. The sizes of the sub-blocks are shown along with their word lengths in Figure 2. The dynamic range of the input is dependent on Signal-to-Noise ratio (SNR), the Multiple Access Interference (MAI) and the number of users in the system. A detailed analysis is required to determine the word-length of the input. For our design, we assume that the received signal is quantized by an A/D converter to have a fixed precision word-length of 8 bits as a similar dynamic range analysis [8, 18] for detection shows the input range to be 8 bits. However, the analysis of the algorithm presented here is independent of the word-length. Also note that the blocks $\mathbf{r}$, $\mathbf{R}_{br}$ and $\mathbf{Y}$ are complex-valued while $\mathbf{b}$ and $\mathbf{R}_{bb}$ are real-valued. For the sake of convenience, we henceforth represent the current inputs $\mathbf{b}_i$, $\mathbf{r}_i$ as $\mathbf{b}$, $\mathbf{r}$ and $\mathbf{b}_{i-L}$, $\mathbf{r}_{i-L}$ as $\mathbf{b}0$, $\mathbf{r}0$.

A typical architecture has window length L = 150, spreading gain N = 32 and the number of users

Table 2. Hardware requirements for an area-constrained architecture

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

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

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

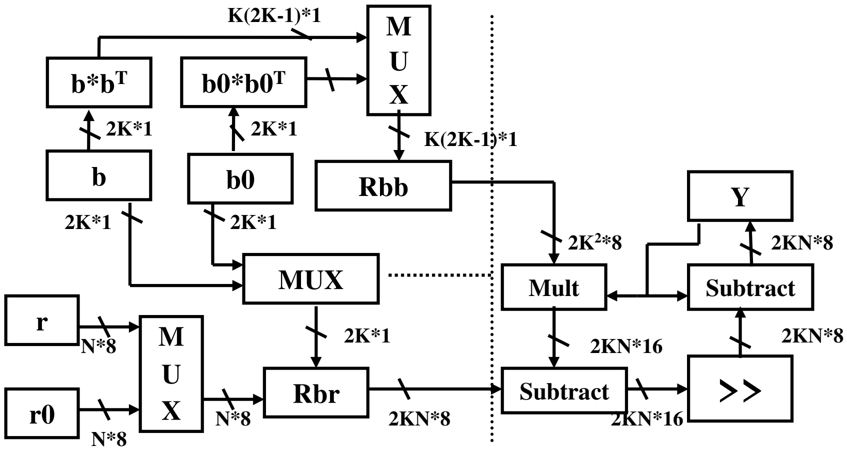

Figure 4. Time-constrained VLSI architecture block diagram

K = 32. All the architectures shown here assume a single-cycle 8-bit multiplication and addition. We assume that a Wallace or Dadda multiplier tree [17] is used for multiplication requiring $O(n^2)$ 1-bit Full Adders for a n-bit multiplication. Since the multiplication by μ in equation 7 results in truncation of the output and need not be highly accurate for numerical stability, a truncated multiplication using significantly less hardware [14] can be used. The delays of blocks such as multiplexers and gates are assumed to be included in the single-cycle delay. For an area estimate of the architectures, we consider the number of 1-bit Full Adder Cells in the design. We also assume all blocks can be pipelined effectively. It can be observed from Figure 2 that the bottleneck in the pipeline is the matrix multiplication $\mathbf{R}_{bb} * \mathbf{Y}$ in equation 7 and we shall concentrate on this part in our architectures.

3.2. Area-constrained architecture

An area-constrained architecture of the multiuser channel estimation scheme is as shown in Figure 3. The architecture is shown for computing only the real part of the channel estimate. Since there are no multiplications between two complex numbers, the architecture can be assumed to be replicated for the imaginary part. In this architecture, all matrix elements are computed an element at a time. The word lengths of the various blocks are as shown in Figure 3. The dotted lines indicate the parts corresponding to the equations 5-7. The left part shows the calculation of the auto-correlation and cross-correlation matrices (equations 5-6)whereas the right part shows the calculation of the iteration loop (equation 7).

To form the outer product update, we take advantage of the single bit nature of the data and replicate the bits $\mathbf{b}, \mathbf{b}0$ such that for forming the $(i, j)^{th}$ element of $\mathbf{R}_{bb}$, the i^{th} and j^{th} bit of b are XNORed (multiplication between +1 and -1 is an XNOR operation) and sent to a counter loaded with the previous value of $\mathbf{R}_{bb}$ which increments or decrements by one. The $(i, j)^{th}$ element of the outer product update $\mathbf{b}0 * \mathbf{b}0^T$ is calculated, negated and sent to the counter, which again increments or decrements by 1 (Up/Down). The multiplexer also has an enable signal such that the output is tristated during the pilot phase, when $\mathbf{b}0 * \mathbf{b}0^T$ (equation 6) is not computed. The matrix $\mathbf{R}_{bb}$ is then updated with a store signal.

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

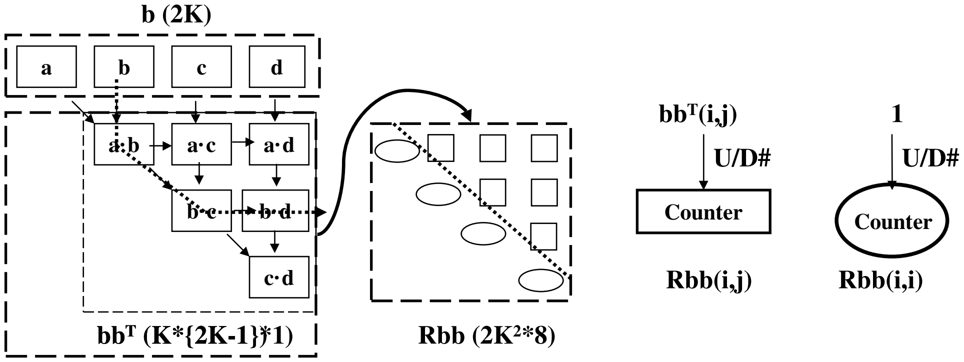

Figure 5. Elements in the auto-correlation matrix block

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

3.3. Time-constrained architecture

The block diagram of a time-constrained architecture is as shown in Figure 4. In this architecture, the available parallelism in the algorithm is exploited to the maximum extent. Hence, all the elements needed to perform a parallel multiplication are computed simultaneously and are pipelined. Now, the entire matrices $\mathbf{R}_{bb}$ and $\mathbf{Y}$ are multiplied using an array of multipliers. The entire product matrix is subtracted by the auto-correlation matrix, $\mathbf{R}_{br}$, shifted and a new channel estimate is formed. Thus, as the time taken by the other computations is pipelined with the time for the multiplication, the output can be formed every $\log_2(2K)$ or 6 cycles.

We exploit the bit-level arithmetic and parallel structure of the correlation matrices to form the correlation matrices simultaneously within a cycle. The sub-blocks for the formation of the auto-correlation matrix and cross-correlation matrix are shown in Figures 5 and 6. Since the auto-correlation matrix update is a symmetric matrix and all the diagonal elements are 1's ($\overline{a \oplus a} = 1$), we need to compute only the strictly upper triangular (or lower triangular) part of the auto-correlation matrix (Figure 5). Also, as the updates are all +1's or -1's, this can be obtained from a simple XNOR gate structure. The counters in the auto-correlation matrix are then updated based on the sign of the updates. Also, the elements in the cross-correlation update are $+\mathbf{r}$ or $-\mathbf{r}$ and hence, the vector $\mathbf{r}$ could be directly added or subtracted with every column of the auto-correlation matrix based on the sign of the bit vector $\mathbf{b}$. The hardware requirements for the time-constrained architecture are as shown in Table 3. We see that though the hardware requirements increase by an order of magnitude, the memory requirements decrease significantly as there is no need for storage and there is a high speedup in time obtained compared to the area-constrained architecture which shows the potential parallelism in the architecture. For a typical architecture , the number of Full Adder Cells required is 20,000,000. This is a far too aggressive solution and difficult to implement even with current silicon technology. However, this states the theoretical minimum time requirements by exploiting the available parallelism as $\log_2(2K)$ or 6 cycles, which is the time required to do the parallel multiplication and pipelining it with the other blocks. We require $2KN(2K-1)$ 16-bit adders for doing the recursive doubling in $\log_2(2K)$ time [adding $2K$ elements in $\log_2(2K)$ time requires $(2K-1)$ adders] and $2KN$ 16-bit adders for the subtraction following the multiplication.

Table 3. Hardware requirements for a time-constrained architecture

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

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

3.4. Area-Time Efficient Architecture

From comparing the above two architectures in Table 5, we see that the area-constrained architecture does not meet real-time requirements while the time-constrained architecture is highly aggressive in area. So, a tradeoff point in the design space needs to be found, which meets the real-time requirements with minimum additional area. This can be done by observing that the major part of the chip area is used the array of multipliers. Hence, instead of computing the entire matrix product in parallel, the product should be computed element by element by doing the inner product in parallel. This would imply $4K$ or 128 multipliers. If this was done row by row or column by column, it would require $4K^2$ or $4KN$ multipliers, requiring about 3600 multipliers, which may not be available just for channel estimation. Since the output is computed element-by-element, this would require $2KN$ or 2000 cycles for the complete channel estimate. The block diagram of the area-time efficient architecture is shown in Figure 7.

The hardware requirements for an efficient area-time architecture are as shown in Table 4. This design requires $2K$ Multipliers to compute an element every cycle and $(2K-1)$ 16-bit adders for recursive doubling. This design requires about 10,000 Full Adder Cells and finds the estimate in $2KN$ cycles.

3.5. Comparisons with DSPs

An architecture comparison of the different VLSI architectures with a DSP is evaluated in this section. Though DSPs and general purpose processors with MMX-enhanced instruction sets can exploit byte-length parallelism, they are inefficient for bit level parallelism. Storage of bits on such a processor is either inefficient as it is stored as bytes or a large overhead is involved in packing and unpacking these bits. Also, the compiler may not take advantage of the fact that most multiplications are with bits and replace them with additions or subtractions. Using a control structure instead also limits the utilization of available parallelism. Also, formation of bit-level matrix updates as seen in the different VLSI architectures is much more effective and simpler to build in hardware with XNOR gates, giving $O(1)$ performance with $O(K^2)$ or 1000 XNOR gates, while it may take $O(K^2)$ or 1000 cycles on a DSP and takes $O(K^2)$ or 1K bytes in memory.

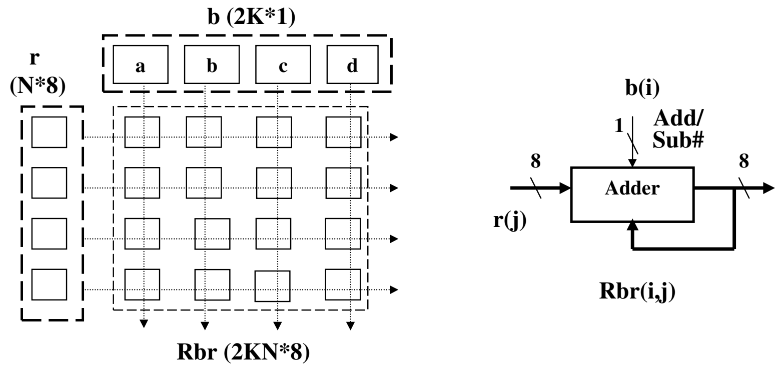

Figure 6. Elements in the cross-correlation matrix block

Assuming a 500 MHz clock for the VLSI architectures, the projected time required to compute the channel estimate along with the hardware required for 32 users and a spreading code of length 32 is as shown in Table 5. This is compared with the implementation of the previously existing algorithm (equation 4), on a TI TMS320C6701 Evaluation Module, operating at 166 MHz. The DSP implementation of the Multiuser Channel estimation algorithm using the previously existing schemes is shown to require 763401 cycles [5], which corresponds to 4.56 ms for 15 users. Assuming that the channel estimate is updated for every block of 10 bits, and extending it linearly to 32 users, this corresponds to a time requirement of 0.97 ms or 1.02 Kbps.

The inherent parallelism present in the algorithm can be seen from the ratio of time taken for computation by the area-constrained and time-constrained architectures. The area estimates are compared using the number of Full Adder Cells needed in the design, as shown in Table 5. The time difference between the DSP and the VLSI architectures is due to the improvements in the algorithm modifications and the fact that the bit-level and byte-level parallelism are not exploited on the DSPs and the additional memory references. The difference in the processor speed does not play a major role in the time differences. We can observe that the area-constrained architecture does not satisfy real-time constraints of 7.8125 μs while the time-constrained architecture is far too aggressive. The area-time efficient architecture meets the next generation real-time constraints by designing the area-time tradeoff in 4 μs, which is twice the target data rate of 128 Kbps. Hence, the clock speed could be reduced by half to 250 MHz for power efficiency. From Table 4, it is seen that the time required is directly proportional to the number of users (K) in the system and the spreading factor (N), which are also dependent on each other as seen from Table 1. Hence, the system design also meets real-time requirements for various data rates, such as 1 Mbps for 4 users with a spreading factor of 4.

4. Summary and Future Directions

We show that a custom VLSI architecture for baseband signal processing in a wireless base-station receiver can be extremely efficient in meeting real-time requirements of the receiver. We discuss three different architectural mappings of multiuser channel estimation, one of the core baseband signal processing algorithms in the receiver. We develop a fixed-point, computationally effective version of the algorithm for a real-time VLSI architecture. The area-constrained architecture with minimum hardware could be mapped on 100K gate FPGAs as it requires only 248 adder cells and 16 KB memory and used in low data rate picocell base-stations. The time-constrained architecture is used to identify the parallelism in the algorithm and to establish the maximum the-

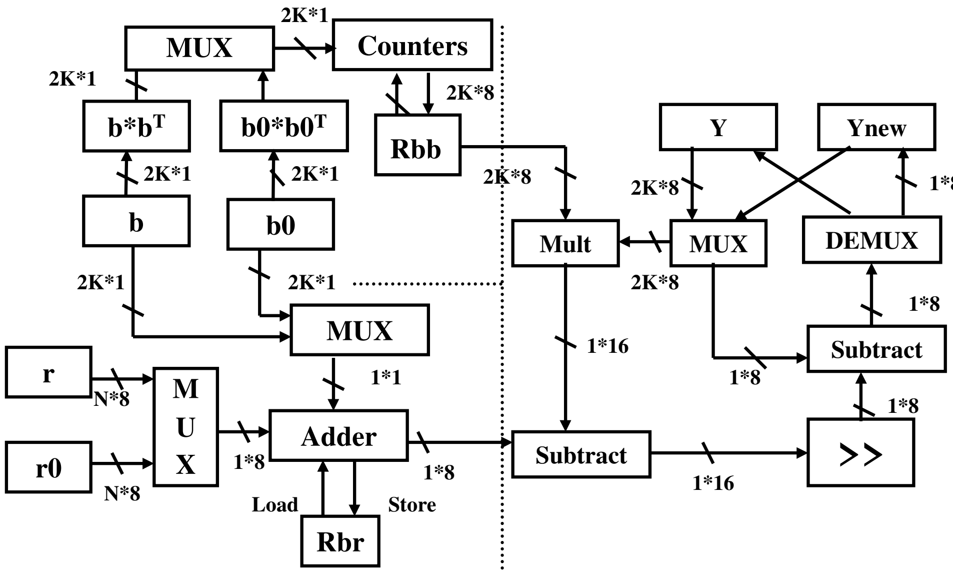

Figure 7. Area-time efficient VLSI architecture

Table 4. Hardware requirements for area-time efficient architecture

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

oretical data rates. The area-time efficient architecture design meets the real-time requirements of the receiver with minimum area overhead. The orders-of-magnitude difference between the area and time-constrained solutions show significant inherent parallelism in the algorithm. The VLSI architectures are able to better exploit the bit level computations and the available parallelism in the algorithm to achieve a real-time solution as compared to a DSP implementation.

A custom VLSI solution would also help in meeting the desired low power requirements of the mobile handset for 'blind' versions of the algorithm. A complex number representation [2] could be used to accelerate the complex-valued computations. The recursive nature of the algorithm could also be used to develop an online and Most Significant Digit First (MSDF) implementation [10]. VLSI solutions for other blocks in the baseband receiver, such as detection and decoding, are also being built for a complete real-time ASIC architecture for a wireless baseband receiver.

Table 5. Comparisons between different architectures

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

References

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

[2] Takafumi Aoki, Yuji Ohi, and Tatsuo Higuchi. Redundant Complex Number Arithmetic For High-Speed Signal Processing. In *IEEE Workshop on VLSI Signal Processing VIII*, pages 523–532, Sakai, Japan, October 1995.

[3] Stephen E. Bensley and Behnaam Aazhang. Maximum likelihood synchronization of a single user for CDMA communication systems. *IEEE Transactions on Communications*, 46(3):392–399, March 1998.

[4] Neiyer S. Correal, R. Micheal Buehrer, and Brian D. Woerner. A DSP-Based DS-CDMA Multiuser Receiver Employing Partial Parallel Interference Cancellation. *IEEE Journal on Selected Areas in Communications*, 17(4):613–630, April 1999.

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

[6] E.G.Strom, S.Parkvall, S.L.Miller, and B.E.Ottersen. DS-CDMA syncronization in time-varying fading channels. *IEEE Journal on Selected Areas in Communication*, 14(8):1636–1642, October 1996.

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

[8] Seehyun Kim, Ki-Il Kum, and Wonyong Sung. Fixed-Point Optimization Utility for C and C++ Based Digital Signal Processing Programs. *IEEE Transactions on Circuits and Systems-II:Analog and Digital Signal Processing*, 45(11):1455–1464, November 1998.

[9] Tao Long and Naresh R. Shanbhag. Low-Power CDMA Multiuser Receiver Architectures. In *IEEE Workshop on Signal Processing Systems*, Taipei, Taiwan, October 1999.

[10] M.D.Ercegovac and T.Lang. Fast Arithmetic For Recursive Computations. In *Workshop on VLSI Signal Processing*, volume V, pages 14–28, Napa Valley, CA, October 1992.

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

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

[13] R.L.Pickholtz, D.L.Schilling, and L.B.Milstein. Theory of spread-spectrum communications- A Tutorial. *IEEE Transactions on Communications*, 30(5):855–884, May 1982.

[14] Michael J. Schulte and Earl E. Swartzlander. Truncated Multiplication with Correction Constant. In *Workshop on VLSI Signal Processing*, volume VI, pages 388–396, Veldhoven, Netherlands, October 1993.

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

[16] Murat Torlak and Guanghan Xu. Blind Multiuser Channel Estimation in Asynchronous CDMA Systems. *IEEE Transactions on Signal Processing*, 45(1):137–147, January 1997.

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

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