EFFICIENT VLSI ARCHITECTURES FOR MULTIUSER CHANNEL ESTIMATION IN WIRELESS BASE-STATION RECEIVERS

Sridhar Rajagopal Srikrishna Bhashyam[†] Joseph R. Cavallaro
Behnaam Aazhang
Rice University
Department of Electrical and Computer Engineering
Houston TX 77005.
{sridhar,skrishna,cavallar,aaz}@rice.edu

Abstract

This paper presents a reduced-complexity, fixed-point algorithm and efficient real-time VLSI architectures for multiuser channel estimation, one of the core baseband processing operations in wireless base-station receivers for CDMA. Future wireless base-station receivers will need to use sophisticated algorithms to support extremely high data rates and multimedia. Current DSP implementations of these algorithms are unable to meet real-time requirements. However, there exists massive parallelism and bit level arithmetic present in these algorithms than can be revealed and efficiently implemented in a VLSI architecture. We *re-design* an existing channel estimation algorithm from an implementation perspective for a reduced complexity, fixed-point hardware implementation. Fixed point simulations are presented to evaluate the precision requirements of the algorithm. A dependence graph of the algorithm is presented and area-time trade-offs are developed. An area-constrained architecture achieves low data rates with minimum hardware, which may be used in pico-cell base-stations. A time-constrained solution exploits the entire available parallelism and determines the maximum theoretical data processing rates. An area-time efficient architecture meets real-time requirements with minimum area overhead.

Keywords : real-time implementation, multiuser channel estimation, VLSI, dependence graphs, DSP, W-CDMA, fixed-point.

This paper was presented in part at the 12th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP), July 2000, pp.173-184

[†]Currently with Qualcomm Inc., Campbell, CA.

I. Introduction

Next generation wireless communication systems [1] 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 not feasible for a direct implementation. This is due to their high computational complexity, involving subroutines such as matrix inversions, which require floating point accuracy. Also, a typical DSP or general purpose processor implementation [2] is unable to fully exploit the parallelism and bit level computations available in these algorithms. Hence, there is a need to *re-design* these algorithms and study their mapping on hardware to accelerate their implementation.

We develop a reduced complexity, fixed point algorithm and efficient 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 [3]. Most implementations either assume perfect channel estimation or assume single user estimation. However, many advanced channel estimation schemes [4, 5] have high computational complexity due to matrix inversions involved and cannot be performed in real-time. Hence, typically in current IS-95 based systems, simpler single-user sliding correlator structures are used for channel estimation [6]. We re-design a recently developed multiuser channel estimation algorithm [7, 8], based on the maximum likelihood principle and develop an iterative scheme [9], which is computationally effective, suitable for a fixed point implementation and is equivalent to matrix inversion in terms of error rate performance. We also present a fixed-point analysis for multiuser channel estimation to evaluate the precision requirements. We use fixed-point classes, developed using C++ for this purpose.

We also analyze a dependence graph (DG) of the algorithm in order to find various area-time trade-offs. Such an analysis is useful as many methods have been proposed for optimal area-time mappings based on DGs [10, 11]. In previous work [9], we had used task-partitioning to find three

different area-time tradeoffs. However, with DGs, a wide range of efficient area-time mappings can be designed. In this paper, we design area-time mappings for three different architectures : an area-constrained architecture, a time-constrained architecture and an area-time efficient architecture.

Architectures for the mobile handset have similar algorithms for implementation. 'Blind' versions [12] 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 architectural design considerations for the mobile handset includes power-efficiency [13] and this also needs to be accounted for in the design as a critical parameter. Though power savings are not as critical for a base-station as for a mobile receiver, we address methods to attain power savings as a discussion section in the end.

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 re-designed from an implementation perspective for a reduced complexity fixed-point solution, without loss in error rate performance. Fixed-point simulations for multiuser channel estimation is presented in Section 3. Section 4 shows the DG of the algorithm and explains the nodes in the DG in detail. The various area-time trade-offs based on the DG are shown in section 5. Area-constrained, time-constrained and area-time efficient architectures are presented. A comparison is also made with a previous DSP implementation. A discussion on obtaining power savings at the base-station is given in Section 6. Conclusions and future directions are presented in Section 7.

II. Multiuser channel estimation

Next generation wireless communication systems [14] use Wideband Code-Division Multiple Access (W-CDMA) as the multiple access protocol for communication. This scheme uses spread spectrum signaling, where each mobile user applies a unique signature sequence (spreading code) to modulate the data bits. The base-station receives a summation of the signals of all the 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. Algorithms for interference cancellation or multiuser detection require the use of such highly accurate estimates of the channel for proper detection.

The performance advantages of using multiuser channel estimation is shown in Figure 1. The top curve shows the bit error rate of a single user channel estimator (sliding correlator) followed by a single user matched filter detector for varying SNR. The next curve shows the performance of multiuser channel estimation with a single user matched filter detector. The bottom curve shows the performance of multiuser channel estimation with multiuser detection. From the figure, we can observe that multiuser channel estimation can give significant performance improvements in bit error rates, especially if used with multiuser detection. However, this performance improvement comes at a cost of increased computational complexity and increased precision requirements.

A. Real-time requirements

Data transmission in 3G wireless systems such as 3GPP or UTMS is possible at varying rates such as from 32 Kbps to 2 Mbps depending on the spreading factor (N) which varies from 256 (for vehicular traffic) to 4 (for indoor environments) respectively (for example, see [15]). The standards assume a chip rate of 4.096 Mcps and Quadrature Phase Shift Keying (QPSK) modulation (2 bits/symbol). We have assumed Binary Phase Shift Keying (BPSK) modulation (1 bit/symbol) in our work for simplicity. Hence, we target data rates in the range of 16 Kbps to 1 Mbps.

To illustrate the advantages of dependence graphs for area-time mappings, we implement our design assuming a spreading factor of 32 chips per data bit as an example. A spreading factor (N) of 32 can support 32 users (K) and we shall use $N = 32$ and $K = 32$ in our design specifications. 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. However, by different area-time mappings for varying spreading gains, we can target a wide range of real-time

requirements of multiuser channel estimation. Note that the reference to 3G systems is solely as an example to illustrate important system features such as the varying data rates which we seek to target and the use of training sequences for channel estimation.

Though we are targeting multiuser channel estimation for every received bit for data rates, this is done as a worst case implementation. Channel estimation updates can be done less frequently, depending on the amount of fading present in the channel. Slowly fading channels require lower channel estimation updates than fast fading channels.

B. Channel model

The model for the received signal at the output of the channel [7, 8] can be expressed as

$$\mathbf{r}_i \;=\; \mathbf{Y}\mathbf{b}_i + \mathbf{n}_i \tag{1}$$

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, $\mathbf{b}_i \in \mathbb{R}^{2K} = [b_{1,i-1}, b_{1,i}, \ldots, b_{K,i-1}, b_{K,i}]^\top$ 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, $\mathbf{n}_i$ 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.

C. Maximum likelihood channel estimation

The channel estimation and detection block in the base-station receiver is shown in Figure 2. 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 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 multiuser channel estimation algorithm is based on the maximum likelihood principle, where the probability of received input given the transmitted bits is maximized. A basic summary of the algorithm and its computational aspects are presented here. More details along with

comparisons with other schemes can be found in [8]. Consider L observations of the received vector $\mathbf{r}_1,\ \mathbf{r}_2,\ \ldots,\ \mathbf{r}_L$ corresponding to the known training bit vectors $\mathbf{b}_1,\ \mathbf{b}_2,\ \ldots,\ \mathbf{b}_L$. Given the knowledge of the training bits, the discretized received vectors $\mathbf{r}_1$, $\mathbf{r}_2$, ..., $\mathbf{r}_L$ are independent and each of them is Gaussian distributed. Thus, the likelihood function becomes

$$\mathrm{p}(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_L | \mathbf{Y}, \mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_L) = \frac{1}{\pi^{NL}} \exp\ \left\{ -\sum_{i=1}^{L} (\mathbf{r}_i - \mathbf{A}\mathbf{b}_i)^H (\mathbf{r}_i - \mathbf{A}\mathbf{b}_{\ i}) \right\}.$$

After eliminating the terms that do not contribute to the maximization, the log likelihood function becomes

$$\left\{ \sum_{i=1}^{L} (\mathbf{r}_i - \mathbf{Y}\mathbf{b}_i)^H (\mathbf{r}_i - \mathbf{A}\mathbf{b}_i) \right\}. \tag{2}$$

The estimate $\hat{\mathbf{Y}}$, that maximizes the log likelihood, satisfies the following equation:

$$\mathbf{R}_{bb}\hat{\mathbf{Y}} = \mathbf{R}_{br}. \tag{3}$$

The computations that occur during the estimation phase [7, 8] are:

$$\mathbf{R}_{br} = \frac{1}{L} \sum_{i=1}^{L} \mathbf{b}_i \mathbf{r}_i^H \tag{4}$$

$$\mathbf{R}_{bb} = \frac{1}{L} \sum_{i=1}^{L} \mathbf{b}_i \mathbf{b}_i^T \tag{5}$$

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 equation (3). 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.

To begin the matrix inversion, we have to wait till all the preamble bits are received and accumulated as in equations (4)-(5). This results in additional delay before the start of computing the estimates using equation (3). Tracking for a fading channel 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 new scheme, based on gradient descent, which begins the computations with the first received preamble bit and facilitates tracking for fading channels is presented in the next section.

The comparisons of our scheme with recursive least squares (RLS) has been shown in [16]. For AWGN, the ML approach of our scheme and the MMSE approach of RLS lead to the same solution and hence, have similar BER performance [16]. However, our scheme is more suited towards a real-time implementation than RLS. Matrix decomposition techniques such as QR have been used for updating the channel estimates for tracking using similar schemes based on RLS [17]. QR is also numerically stable even with fixed-point computations [18]. The QR decomposition can also be implemented efficiently in fixed-point using systolic arrays [19]. However, the cells in the array (especially, the boundary cells which compute the Givens rotation) [17, 19] have more computational complexity than the cells used in our iterative algorithm. Our proposed iterative algorithm uses only truncated multipliers and adders and does not require any special boundary cells. Hence, our proposed iterative algorithm has a lower hardware complexity than schemes based on RLS.

D. 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 [20].

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 below:

$$\mathbf{R}_{br}^{(t)} = \mathbf{R}_{br}^{(t-1)} + \mathbf{b}_i \mathbf{r}_i^H - \mathbf{b}_{i-L} \mathbf{r}_{i-L}^H \quad (6)$$

$$\mathbf{R}_{bb}^{(t)} = \mathbf{R}_{bb}^{(t-1)} + \mathbf{b}_i \mathbf{b}_i^T - \mathbf{b}_{i-L} \mathbf{b}_{i-L}^T \quad (7)$$

$$\mathbf{Y}^{(t)} = \mathbf{Y}^{(t-1)} - \mu(\mathbf{R}_{bb}^{(t)} * \mathbf{Y}^{(t-1)} - \mathbf{R}_{br}^{(t)}) \quad (8)$$

where t represents the iteration count. This scheme is suitable for tracking, which is shown by the removal of the oldest bit in the window of length L as the newest 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 (4), (5))*. 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.

Figure 3 shows the performance of both schemes, the iterative method and the previous scheme using matrix inversion for a static multipath environment. The Bit Error Rate (BER) is calculated using the channel estimates after the end of the pilot phase for two types of detectors, a Matched Filter Detector (MF) [21, 22] and a Multistage Multiuser Detector (MUD) [23, 24], based on the principle of Parallel Interference Cancellation (PIC). The simulations are carried out for a 15-user system using a pilot of length 128 bits (power of 2). The users are all transmitting at the same power over a static channel with 3 paths of relative strengths 1, 0.5 and 0.33. Although the detection algorithm can handle the near-far problem, we simulated the equal power scenario as it generates the worst case for multistage detection. The simulation was carried out for a detection

window length of 10. The value of μ for the iterative scheme was chosen to be 1/1024 (power of 2). Walsh codes of length 32 were used. From the simulations, it can be observed that the iterative scheme denoted by (ITER) in Figure 3 and Figure 4 essentially gives the same error rate performance as that of the original scheme.

The analysis of the system for a multipath fading channel with tracking is as shown in Figure 4. Here we see that the proposed tracking scheme based on the update equations (6)-(7) is able to effectively track the time-varying channel. The poor performance of the static channel assumption for this Rayleigh fading channel (with mobile velocity 10 km/h) at a carrier frequency of 1.8 GHz shows the importance of tracking. The simulation was done for 15 equal power users with a window length of 128 (and preamble length of 128). For faster fading, the window length needs to be decreased appropriately. The original channel estimation scheme requires a matrix inversion and matrix multiplication for every update while the iterative scheme reduces the complexity to a matrix multiplication per update.

III. Fixed-point implementation for multiuser channel estimation

Current CDMA-based systems such as IS-95 implement a single user channel estimator (sliding correlator) and a single user matched filter receiver. This can be implemented with 8-bit precision. However, for multiuser channel estimation, there is a cost of both increased complexity and increased precision requirements in order to obtain BER performance benefits. In this section, we discuss the effects of finite precision for multiuser estimation. For this purpose, we developed a model of the system in C++ using fixed-point "classes" in order to study the performance of the system with different precision requirements. The multiplications and addition operations were "over-loaded" so as to saturate if the available precision were to be exceeded. Instead of exploring a detailed theoretical analysis similar to [25], for simplicity, we choose to study the effects of finite precision on multiuser channel estimation based on performance using simulations.

Since the received signal amplitude depends on the number of users in the system, the number of multiple path reflections, the spreading gain and the signal-to-noise ratio, we developed a model

to first calculate the amount of precision required by the A/D converter.

$$\text{precision (in bits)} = \left\lceil \log_2(K * \sum_{p=1}^{P} \frac{1}{p} + 4 * 10^{\frac{-snr}{20}} * \frac{\sqrt{N}}{\sqrt{2}}) \right\rceil + 4 \tag{9}$$

Equation (9) was developed for quantization using the fact that the maximum amplitude of the received signal would be $K * \sum_{p=1}^{P} \frac{1}{p}$, where K is the number of users and P is the number of multipath reflections. The noise would be less than $4 * \sigma * \frac{\sqrt{N}}{\sqrt{2}}$ with a probability of more than 0.99, where σ is the variance of the noise and N is the spreading gain. Four more bits for additional precision are provided with one bit for the sign. This gives precisions in the range of 8-12 bits for different users and spreading gains which is possible with current A/D converters.

We present two simulation results for finite precision with different spreading gains. Figure 5 shows the bit error rate performance of multiuser channel estimation for a spreading gain of 16 with 8 users. Figure 6 shows the performance for a spreading gain of 32 with 15 users. In each case, we choose a preamble length of 128 and a μ of $1/1024$ (chosen to be smaller than the reciprocal of the largest eigenvalue of $\mathbf{R}_{bb}^{(i)}$ for all i in order to ensure convergence [26]). The fixed-point output of channel estimation was tested with floating point versions of the single user matched filter detector [21, 22] and a multiuser detector based on PIC [23, 24]. Floating point detection was used to ensure that the loss in performance was due to the effects of channel estimation alone. The effects of fixed-point detection is studied in [16].

Based on the simulations performed, we have made the following observations:

1. Multiuser channel estimation has higher precision requirements in general than single user channel estimation. Though lower precision implementations for multiuser channel estimation can be constructed for specific values of N and K, to obtain generality and flexibility, we need higher precision in the range of 13-16 bits for multiuser channel estimation as opposed to sliding correlator implementations.

2. We see that 13-16 bit precision multiuser channel estimation performs almost as well as floating point precision multiuser estimation. In fact, for $N = 16$ and $N = 32$, the performance begins to degrade only at 12-bit precision. It is also interesting to note from Figure 5 and

Figure 6 that as soon as the minimum precision (13-bit) is obtained, there are no significant benefits with additional precision for multiuser estimation.

3. The A/D quantization of the received chip-matched filter output does not require as much precision as required for the computations. Reasonable precision of 8-12 bits for A/D conversion is sufficient. For very high SNR, there could be some degradation due to the A/D quantization as the quantization noise could be significant compared to the background noise.

IV. Dependence graph for multiuser channel estimation

A. Design specifications

The dynamic range of the input $\mathbf{r}$ 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, $\mathbf{r}$, is quantized by an A/D converter to have a fixed precision word-length of 8-12 bits, depending on the spreading gain and the SNR as discussed in the previous section. We assume an implementation on 16-bit architectures as we have shown them to have sufficient fixed-point range. The area and time requirements do not vary significantly with precision as they are more dependent on the spreading gain. 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 $\mathbf{b}_0$, $\mathbf{r}_0$.

As an example, we choose a typical architecture as having spreading gain $N = 32$ and the number of users $K = 32$ to target 128 Kbps/user. All architecture designs assume a single-cycle 16-bit multiplication and addition. We justify this assumption to the following reasons. First, as seen in Figure 10, the node for matrix multiplication in a systolic array needs to do a multiplication and addition in the same amount of time and it is difficult to pipeline the multiplication further. Also, we envision that such dependence graph mappings could be mapped to programmable processors using mapping techniques as in [27], that may have single cycle multiplication and addition. This assumption also helps in simplifying the DG mapping and helps us with DSP comparisons.

We assume that a Wallace or Dadda multiplier tree [28] is used for multiplication requiring $O(n^2)$ 1-bit Full Adders (FA) for an n-bit multiplication. Since the multiplication by μ in equation (8) results in truncation of the output and need not be highly accurate for numerical stability, a truncated multiplication using significantly less hardware [29] can be used. Such truncated multiplier schemes can offer 25-35% savings in the area of the multiplier, depending on the precision requirements. For an area estimate of the architectures, we consider the number of 1-bit FA cells in the design. We also assume that other blocks in a communication system such as those for detection and decoding can be pipelined in a similar fashion with multiuser estimation.

B. Dependence graph

Dependence graph (DG) analysis [10] is a tool for mapping parallel algorithms into array structures. A DG is used to show the data dependencies between the operations. By mapping from a DG design to a Signal Flow Graph (SFG) representation to parallel hardware, different VLSI architectures with various area-time tradeoffs can be achieved. We choose to design a DG for the multiuser channel estimation algorithm as there have been many methods proposed for efficient mapping of algorithms [10] and area-time optimizations using dependence matrices [11] developed for parallel processing. A systolic nature of the interconnections is used as it helps in avoiding *broadcast data*, and eliminates problems such as clock-skew, fault susceptibility and peak-power present in global interconnections [10]. A DG of the algorithm, based on equations (6)-(8), is as shown in Figure 7. The figure shows the DGs for each of the sub-blocks of the algorithm. The block arrows represent interconnections from all elements in a plane to the subsequent block (*for clarity*). The regular arrows represent a systolic interconnection while the dotted lines represent independent processing elements with no interconnections. The DG has been aligned such that the arrows represent the direction of data flow. The sequence of operations in the algorithm is ordered on the DG. The shaded portions in the DG are individual elements and non-systolic.

Step 1 refers to the DG for $\mathbf{R}_{bb}$ along the (i,j)-plane. This is then fed to the (i,j)-face of the matrix multiplication DG cube $\mathbf{R}_{bb} * \mathbf{Y}^{(t-1)}$. The previous value of the channel estimate $\mathbf{Y}^{(t-1)}$ is applied to the (i,k)-face of the cube. The output $\mathbf{R}_{bb} * \mathbf{Y}^{(t-1)}$ is then produced on the (j,k)-face of the cube in step 2. At the same time, the cross-correlation matrix $\mathbf{R}_{br}$ is also computed in the

(j,k)-plane. The loop of steps 2,3,4 of the DG represent the computations present in equation (8). It can be observed from Figure 7 that the bottleneck in the DG is the matrix multiplication $\mathbf{R}_{bb} * \mathbf{Y}^{(t-1)}$ in equation (8) and we shall concentrate on this part in our architectures. The nodes of the DG are explained below.

1. *Node for auto-correlation*

The structure of the $\mathbf{R}_{bb}$ autocorrelation node is shown in Figure 8. This figure shows the nodes of the three different processing elements in the autocorrelation block along with their wordlengths. Since $\mathbf{R}_{bb}$ is symmetric, we need to compute only a triangular part of the matrix. Figure 8(a) shows the nodes for $i \leq j \leq 2K-2$. The node consists of a XNOR gate to compute the multiplication (*multiplication between +1 and -1 is an XNOR operation*). This is then sent to an UP/DOWN counter, loaded with the previous value of $\mathbf{R}_{bb}$ which increments or decrements by one based on the XNOR gate output (different for $\mathbf{b}$ and $\mathbf{b}_0$). The output $\mathbf{R}_{bb}(i,j)$ is then mapped on the (i,j) face of the cube for the matrix multiplication. Figure 8(b) shows the nodes for $i = j = 2K$. The counter is always incremented as all the diagonal elements of $\mathbf{R}_{bb}$ are 1's ($\overline{a \oplus a} = 1$). Figure 8(c) shows the nodes for $i = j = 2K-1$. The inputs cross-over at this node so that just the j-axis input is able to serve both the i, j nodes.

2. *Node for cross-correlation*

The structure of the $\mathbf{R}_{br}$ autocorrelation node is shown in Figure 9. This figure shows the nodes of the three different processing elements in the cross-correlation block along with their wordlengths. The node consists of an adder/subtracter block which updates the stored value of $\mathbf{R}_{br}$, based on the value of the input $\mathbf{r}$ and the sign of $\mathbf{b}$.

3. *Node for matrix multiplication*

The structure of the matrix multiplication node [10] is shown in Figure 10. Matrix-matrix multiplication results in a DG cube, with inputs, being applied at two faces and the output emerging from the third face. The input operands are sent along the j and k axes and the dot product is

computed along the i-axis. Note that though we expect a 24-bit output at the multiplier, the result is truncated to 16-bit precision.

4. *Other processing elements*

The two processing elements shown with shaded blocks in Figure 7 are non-systolic in the sense that they do not have any *transmittent* data and are independent elements. Figure 11 shows the detailed nodes in these two processing elements.

V. Mapping to VLSI architectures

In this section, we explore different area-time tradeoffs, based on the dependency graph. We saw the matrix-matrix multiplication in Figure 7 was the bottleneck and we concentrate on it for different mappings. We observe the area and time requirements for three different mappings : an area-constrained, a time constrained and an area-time efficient mapping. A comparison with a DSP implementation is also made in this section.

A. Area-constrained architecture

In order to obtain an area-constrained architecture, we seek to minimize the number of processing elements, required in the DG of Figure 7. A mapping for an area-constrained architecture of multiuser channel estimation is as shown in Figure 12. Each basic node is implemented just once. The iterations in order to compute the elements for the entire array are shown by loops around the node. The location of the loop represents the direction in which an iteration is folded on a single element. 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 matrix-matrix multiplication cube of the DG in Figure 7 is compressed in all three directions requiring only a single multiplier and taking $4K^2N$ or 128,000 cycles (*the time taken to compute the entire matrix*). The hardware requirements for an area-constrained architecture are as

shown in Table 1. The design requires an 8-bit counter, a 16-bit multiplier and four 16-bit adders. 128,000 clock cycles takes 262 μs on a clock rate of 500 MHz, giving us a data rate of 3.81 Kbps. Thus, we see that the area-constrained architecture fails to meet real-time requirements.

B. Time-constrained architecture

Theoretically, each node of the DG of Figure 7 can be mapped to a separate processing element for a time-constrained architecture. However, since we choose to make the DG systolic to avoid broadcast data, the output takes $2K$ or 64 cycles to compute the dot product. However, if the data is broadcast and the dot product per output element is computed using a tree structure, the output can be formed every $\log_2(2K) + 1$ or 7 cycles. This is because each element of a $N * N$ product matrix can be computed in $\log_2(N) + 1$ time using N^3 multipliers and using a tree structure to compute the inner products [30], in a time-constrained architecture. The hardware requirements for the time-constrained architecture are as shown in Table 2. 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 FA cells required is 40,000,000. This is a highly aggressive solution for today's silicon technology and it is infeasible to devote so many FA cells just for channel estimation, which is only a part of the complete receiver. However, this states the theoretical minimum time requirements by exploiting the available parallelism as $\log_2(2K)+1$ or 7 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 [30] 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. 7 cycles takes 14 ns on a 500 MHz clock, which translates into a data rate of 71.43 Mbps. Thus, we see that the time-constrained architecture does meet real-time but with a significantly large area overhead.

C. Area-Time efficient architecture

From comparing the above two architectures in Table 4, 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 area overhead. This can be done by observing that the major part of the chip area is used for the array of multipliers. Hence, instead of computing the entire matrix product in parallel, the product should be computed element by element by starting N computations in parallel. This would imply $2N$ or 64 multipliers. The mapping for an area-time efficient architecture is shown in Figure 13. Here, an array of N multipliers along the k-axis is shown. From the figure, we can observe that the output is computed N elements every $2K$ cycles. Thus, this would require $4K^2$ or 4000 cycles for the complete $2KN$ channel estimate.

The hardware requirements for an efficient area-time architecture are as shown in Table 3. This design requires N multipliers to compute N elements every $2K$ cycles and N 16-bit adders. This design requires about 20,000 FA cells and finds the estimate in $4K^2$ cycles. 4000 cycles implies 8 μs on a 500 MHz clock, giving a data rate of 128 Kbps. Thus, the area-time efficient architecture meets real-time with minimum area overhead.

From compressing the DG in Figure 7 in different directions and axes, we can obtain several other intermediate mappings. When the data processing requirements change due to different spreading gains or in order to target other data rates, a different intermediate real-time efficient architecture needs to be obtained. For example, if the target data rates double, a new area-time efficient architecture can be achieved by adding another row of N-multipliers to Figure 13. However, we need to ensure that all the processor computation and communication rates are satisfied. A more formal method of finding real-time architectures is by using dependence matrices and finding area-time optimizations or by developing a compiler that does the scheduling, as in [11, 27]. However, the process becomes difficult when the algorithms become complicated and more global optimizations need to be performed. We are investigating this further as future work.

D. 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. 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.

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 4. This is compared with a DSP implementation of the revised algorithm on a TI TMS320C6701 DSP at 167 MHz. The channel estimation DSP implementation takes 50 ms for all 32 users or 0.64 Kbps/user. The poor performance of the DSP is due to its inability to exploit parallelism and inefficient bit storage. 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 FA cells needed in the design, as shown in Table 4. 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 for 8 μs.

VI. Power savings

Though power issues are not critical from a base-station perspective, there are several techniques that could be applied to the iterative multiuser estimation scheme that could result in power savings. Apart from the basic reduction in power due to a VLSI implementation, it has been shown in [29, 31] that the use of truncated multiplication can have a savings of 25-35% in hardware and 29-40% savings in the power dissipated by the multipliers, depending on the precision. Since most of the area in the multiuser channel estimation design is consumed by multipliers, this could result in significant power savings. Also, on-line arithmetic techniques [32, 33] based on digit-serial computations can be applied to multiuser channel estimation to provide significant savings in area and power due to digit-serial hardware. Techniques such as disabling parts of the filter taps that

are not needed in case of better BER performance than necessary as in [13] for multiuser detection could also be considered for multiuser channel estimation. For example, the iterative updates for multiuser channel estimation could be stopped when performance is better than necessary. The reduction in computations in turn relates to power savings.

VII. Conclusions

We present a reduced complexity algorithm for multiuser channel estimation, which has been re-designed from an implementation perspective. The algorithm is massively parallel and is suitable for a fixed-point VLSI implementation. We show that there is no loss in the algorithm performance due to this re-design of the algorithm. Simulations show that 13-16 bit precision is sufficient for producing close to floating point performance. A dependence graph of the channel estimation algorithm is presented to evaluate different area-time tradeoffs. We present mappings for three area-time tradeoffs: an area-constrained architecture, a time-constrained architecture and an area-time efficient architecture. An area-constrained architecture achieves low data rates with minimum hardware, which may be used in pico-cell base-stations. A time-constrained solution exploits the entire available parallelism and determines the maximum theoretical data processing rates. An area-time efficient architecture meets real-time requirements with minimum area overhead. A wide range of real-time architectures can be designed with different area-time mappings using DGs. Thus, by re-designing algorithms from an implementation perspective and mapping them to real-time architectures using DGs, VLSI solutions for other processing blocks such as detection and decoding can also be built for a complete real-time ASIC baseband receiver.

Acknowledgments

We are extremely grateful to the reviewers for their comments, which has helped in significant modifications and improvements on this paper. We are grateful to Parthasarathy Ranganathan for his comments and feedback. This work was supported in part by Nokia Corporation, Texas

Instruments Inc., the Texas Advanced Technology Program under grant 1999-003604-080, and by NSF under grant ANI-9979465.

References

[1] M. Zeng, A. Annamalai, and V. K. Bhargava, "Recent advances in cellular wireless communications," *IEEE Communications Magazine*, vol. 37, no. 9, pp. 128–138, September 1999.

[2] S. Das, S. Rajagopal, C. Sengupta, and J. R. Cavallaro, "Arithmetic acceleration techniques for wireless communication receivers," in 33^{rd} *Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, October 1999, pp. 1469–1474.

[3] 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.

[4] S. E. Bensley and B. Aazhang, "Maximum likelihood synchronization of a single user for CDMA communication systems," *IEEE Transactions on Communications*, vol. 46, no. 3, pp. 392–399, March 1998.

[5] E. G. Strom, S. Parkvall, S. L. Miller, and B. E. Ottersen, "DS-CDMA synchronization in time-varying fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 8, pp. 1636–1642, October 1996.

[6] R. L. Pickholtz, D. L. Schilling, and L. B. Milstein, "Theory of spread-spectrum communications - A tutorial," *IEEE Transactions on Communications*, vol. 30, no. 5, pp. 855–884, May 1982.

[7] C. Sengupta, S. Das, J. R. Cavallaro, and B. Aazhang, "Efficient multiuser receivers for CDMA systems," in *IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, September 1999, pp. 1461–1465.

[8] C. Sengupta, J. R. Cavallaro, and B. Aazhang, "On multipath channel estimation for CDMA using multiple sensors," *IEEE Transactions on Communications*, vol. 49, no. 3, pp. 543–553, March 2001.

[9] S. Rajagopal, S. Bhashyam, J. R. Cavallaro, and B. Aazhang, "Efficient VLSI architectures for baseband signal processing in wireless base-station receivers," in 12^{th} *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, Boston, MA, July 2000, pp. 173–184.

[10] S. Y. Kung, *VLSI Array Processors*, Information and System Sciences. Prentice Hall, Englewood Cliffs, NJ, first edition, 1988.

[11] D. I. Moldovan, *Parallel Processing - From Applications to Systems*, Morgan Kaufmann, San Mateo, CA, first edition, 1993.

[12] M. Torlak and G. Xu, "Blind multiuser channel estimation in asynchronous CDMA systems," in *IEEE Transactions on Signal Processing*, January 1997, vol. 45, pp. 137–147.

[13] T. Long and N. R. Shanbhag, "Low-power CDMA multiuser receiver architectures," in *IEEE Workshop on Signal Processing Systems*, Taipei, Taiwan, October 1999, pp. 493–502.

[14] F. Adachi, M. Sawahashi, and H. Suda, "Wideband DS- CDMA for next-generation mobile communication systems," *IEEE Communications Magazine*, vol. 36, no. 9, pp. 56–69, September 1998.

[15] "Third Generation Partnership Project," http:// www.3gpp.org.

[16] S. Rajagopal, S. Bhashyam, J. R. Cavallaro, and B. Aazhang, "Real-time algorithms and architectures for multiuser channel estimation and Detection in Wireless Base-station Receivers," *Accepted to IEEE Journal in Selected Areas in Communications (JSAC) (also available as Rice University Tech. Report TREE0102)*, 2001.

[17] S. Haykin, *Adaptive Filter Theory*, Information and System Sciences Series. Prentice Hall, 3rd edition, 1996.

[18] P. S. R. Diniz and M. G. Siqueira, "Fixed-point error analysis of the QR-recursive least square algorithm," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 42, no. 5, pp. 334–348, May 1995.

[19] K. J. Raghunath and K. K. Parhi, "Finite-precision error analysis of QRD-RLS and STAR-RLS adaptive filters," *IEEE Transactions on Signal Processing*, vol. 45, no. 5, pp. 1193–1209, May 1997.

[20] G. H. Golub and C. F. VanLoan, *Matrix Computations*, chapter 10, pp. 520–521, John Hopkins University Press, third edition, 1996.

[21] S. Moshavi, "Multi-user detection for DS-CDMA communications," *IEEE Communications Magazine*, pp. 124–136, October 1996.

[22] S. Verdú, "Minimum probability of error for asynchronous gaussian multiple-access channels," *IEEE Transactions on Information Theory*, vol. IT-32, no. 1, pp. 85–96, 1986.

[23] 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, April 1990.

[24] S. Rajagopal and J. R. Cavallaro, "A bit-streaming pipelined multiuser detector for wireless communications," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Sydney, Australia, May 2001, vol. 4, pp. 128–131.

[25] R. D. Gitlin and S. B. Weinstein, "On the required tap-weight precision for digitally implemented, adaptive, mean-squared equalizers," *The Bell System Technical Journal*, vol. 58, no. 2, pp. 301–321, February 1979.

[26] S. Bhashyam and B. Aazhang, "Multiuser channel estimation for long code CDMA systems," in *IEEE Wireless Communication and Networking Conference (WCNC)*, Chicago, IL, September 2000, pp. 664–669.

[27] S. S. Bhattacharyya, R. Leupers, and P. Marwedel, "Software synthesis and code generation for signal processing systems," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 47, no. 9, pp. 849–875, September 2000.

[28] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI design: A Systems Perspective*, chapter 8, Addison-Wesley, second edition, 1993.

[29] M. J. Schulte and E. E. Swartzlander, "Truncated multiplication with correction constant," in *Workshop on VLSI Signal Processing*, Veldhoven, Netherlands, October 1993, vol. VI, pp. 388–396.

[30] J. J. Modi, *Parallel Algorithms and Matrix Computation*, Oxford University Press, 1988.

[31] M. J. Schulte, J. E. Stine, and J. G. Jansen, "Reduced power dissipation through truncated multiplication," in *IEEE Alessandro Volta Memorial Workshop on Low Power Design*, Como, Italy, March 1999, pp. 61–69.

[32] M. D. Ercegovac, "On-line arithmetic: An overview," in *Real Time Signal Processing VII, SPIE*, San Diego, CA, August 1984, vol. 495, pp. 86–93.

[33] S. Rajagopal and J. R. Cavallaro, "On-line arithmetic for detection in digital communication receivers," in 15^{th} *IEEE International Symposium on Computer Arithmetic (ARITH-15)*, Vail, CO, June 2001, pp. 257–265.

Table 1: Hardware requirements for an area-constrained architecture

Blocks	**Quantity**	**FA cells**	**Complex**	**Total**
Counter	1*8	8	-	8
Multiplier	1*16	256	*2	512
Adders	$4 * 16$	64	*2	128
Total FA cells				648
Total Time (Cycles)			$4K^2N$	128,000

Table 2: Hardware requirements for a time-constrained architecture

Blocks	**Quantity**	**FA cells**	**Complex**	**Total**
Counter	$2K^2 * 8$	$16K^2$	-	$16K^2$
Multipliers	$4K^2N * 16$	$1024K^2N$	*2	$2048K^2N$
Adders	$4KN * 16+$	$64KN$	*2	$128KN+$
	$4K^2N * 16$	$+64K^2N$		$128K^2N$
Total FA cells N=K=32				40,000,000
Total Time(Cycles)			$\log_2(2K) + 1$	7

Table 3: Hardware requirements for area-time efficient architecture

Blocks	**Quantity**	**FA cells**	**Complex**	**Total**
Counter	1*8	8	-	8
Multipliers	N*16	$256N$	*2	$512N$
Adders	$4 * N * 16$	$64N$	*2	$128N$
Total FA cells N=K=32				20,000
Total Time(Cycles)			$4K^2$	4,000

Table 4: Comparisons between different architectures

Architecture	**FA cells**	**Data Rates/User**
Area-Constrained	648	3.81 Kbps
Time-Constrained	40,000,000	71.43 Mbps
Area-Time	20,000	128 Kbps
TMS320C6701 DSP	-	0.64 Kbps
Real-Time Requirements		128Kbps

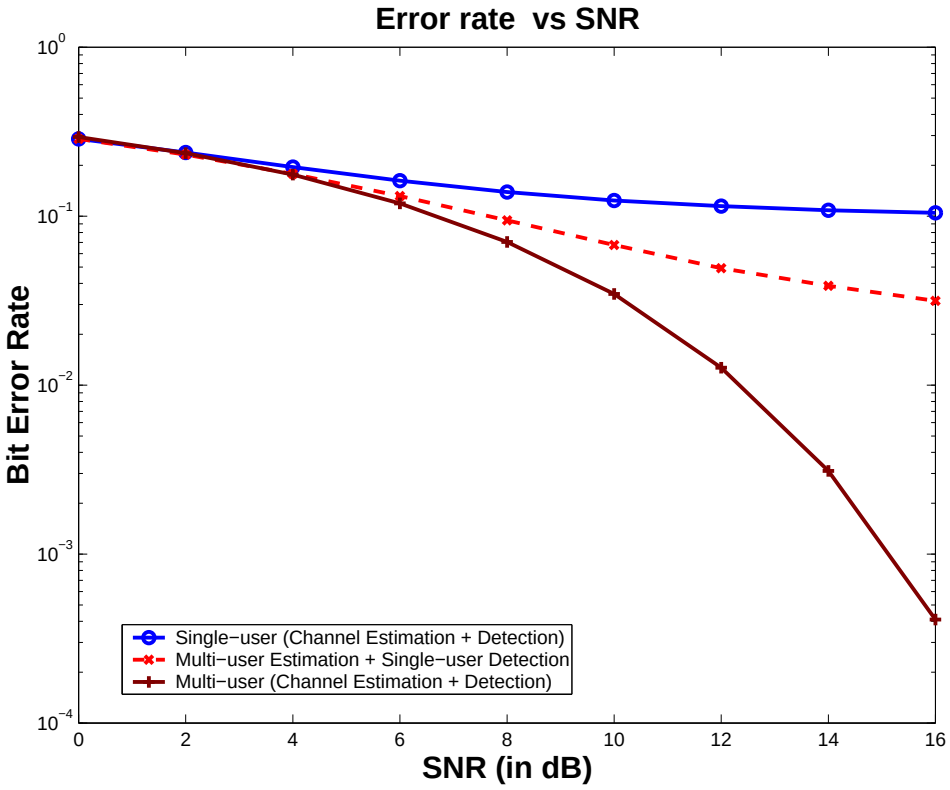

Figure 1: Performance benefits of multiuser channel estimation

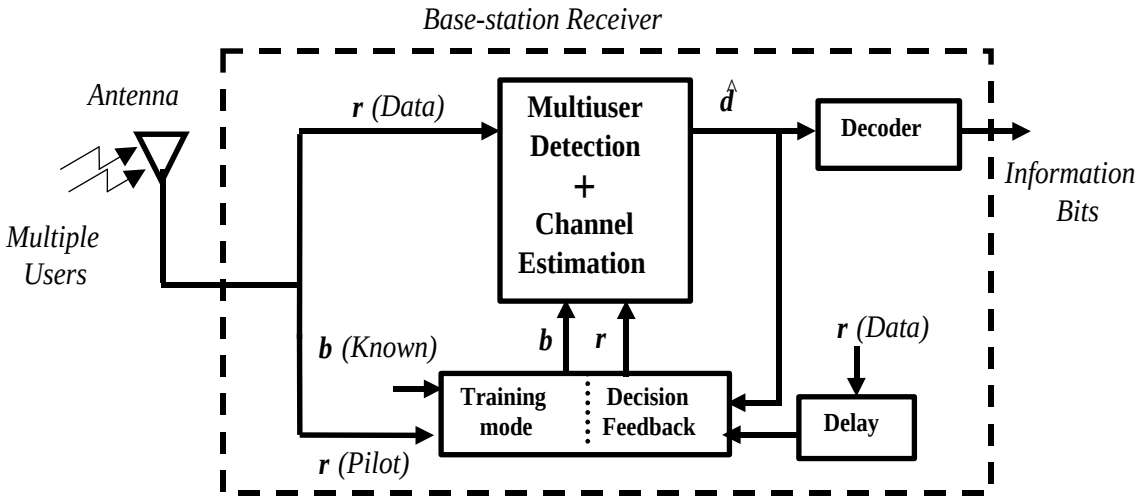

Figure 2: Simplified view of the base station receiver

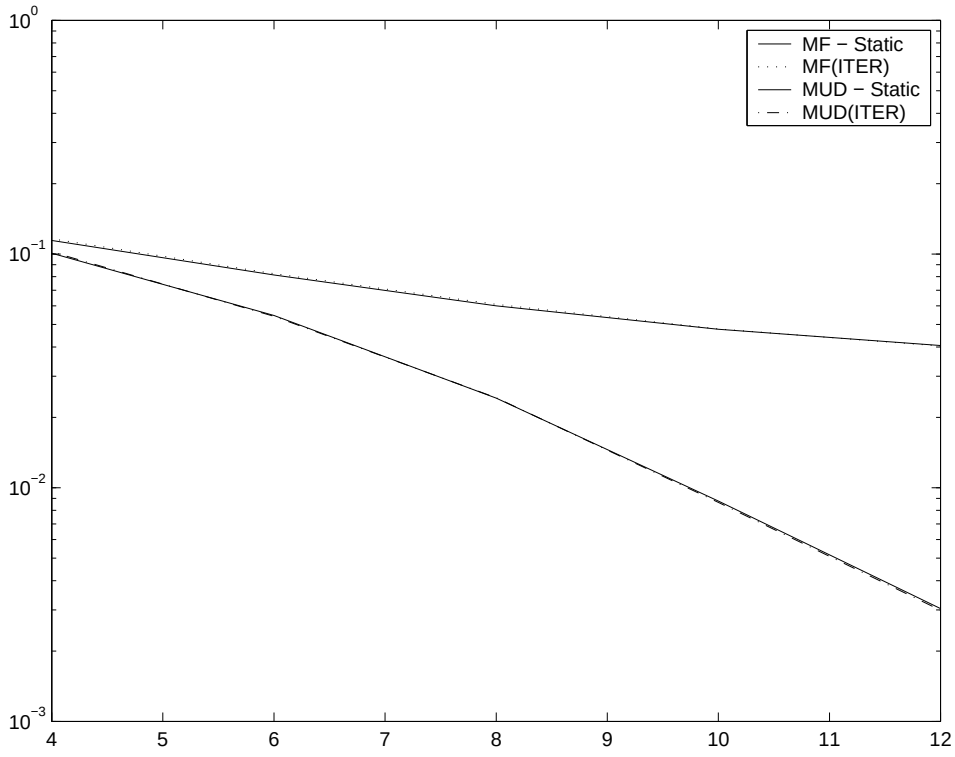

Figure 3: Error rate performance for a static multipath channel. This figure shows the error performance for two detectors, a matched filter detector and a multiuser detector for both channel estimation schemes: the inversion based scheme and the iterative scheme. The simulations were done for 15 users at equal powers with a spreading gain of 32. The channel is assumed to be static with AWGN and 3 paths. A pilot sequence of 128 bits was used initially to obtain the channel estimates. A detection window of 10 bits was used for the multiuser detector.

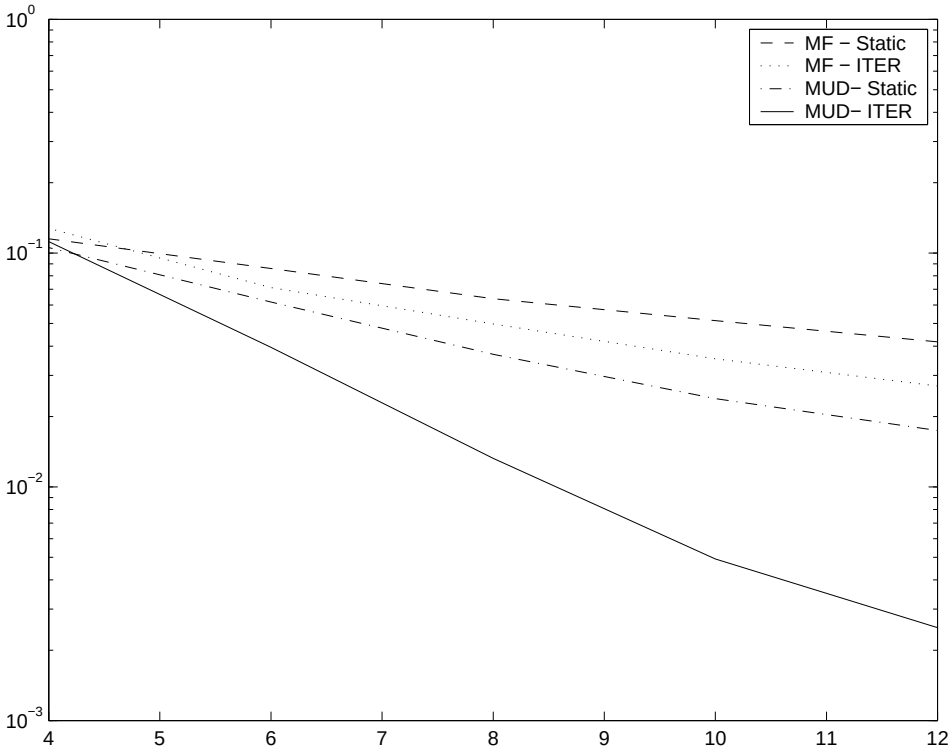

Figure 4: Error rate performance in a multipath fading channel. This figure shows the error performance of both estimation schemes in the presence of slow fading at 10 km/h mobile velocity at a carrier frequency of 1.8 GHz. The matrix inversion based scheme assumes a static channel and is not updated with decision feedback, while the iterative scheme is updated every bit. The convergence parameter, μ, is chosen as 1/1024. Other parameters are the same as for Figure 3.

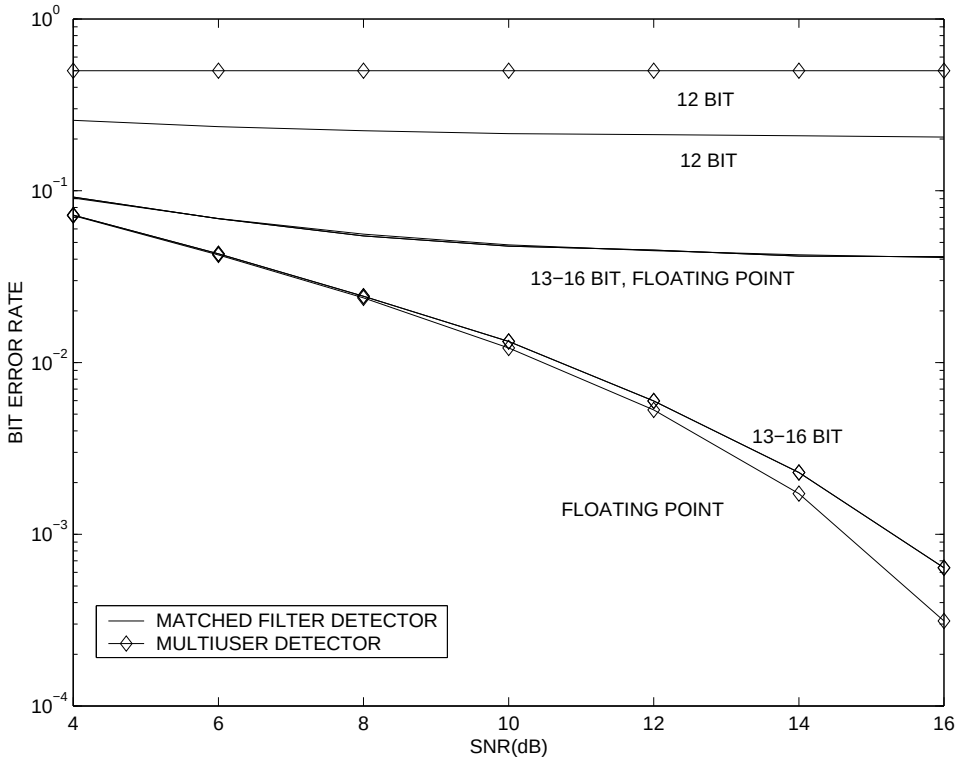

Figure 5: Fixed point error rate performance of multiuser channel estimation for N = 16, K = 8. The figure shows the effect of quantization of multiuser channel estimation on a single user matched filter receiver and a multiuser detector.

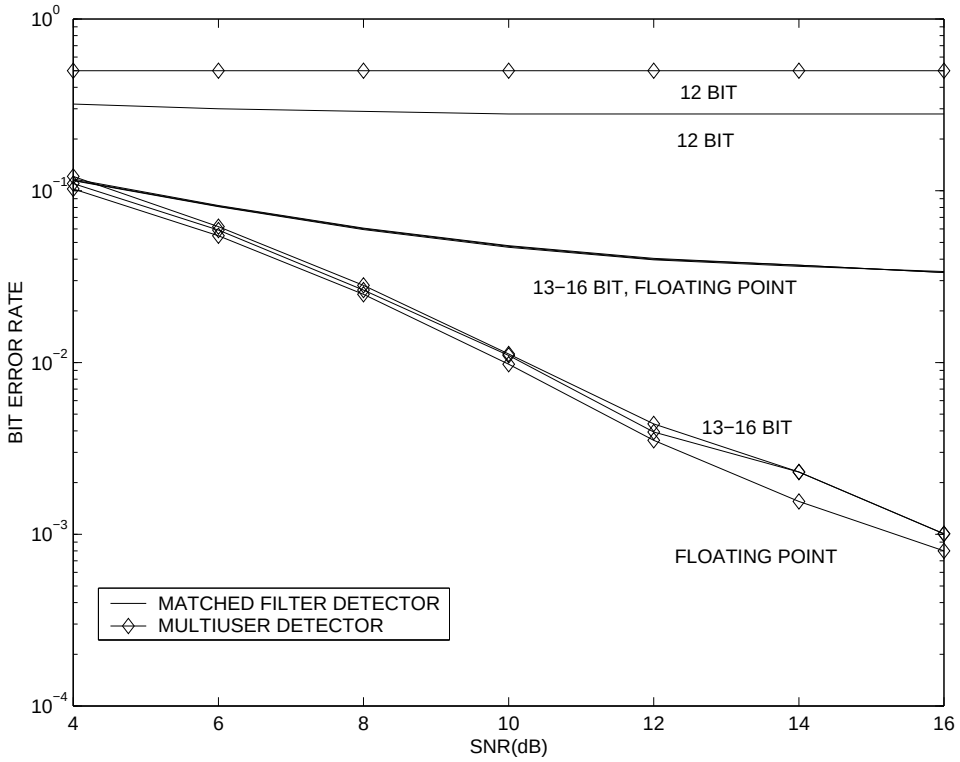

Figure 6: Fixed point error rate performance of multiuser channel estimation for N = 32, K = 15. The figure shows the effect of quantization of multiuser channel estimation on a single user matched filter receiver and a multiuser detector.

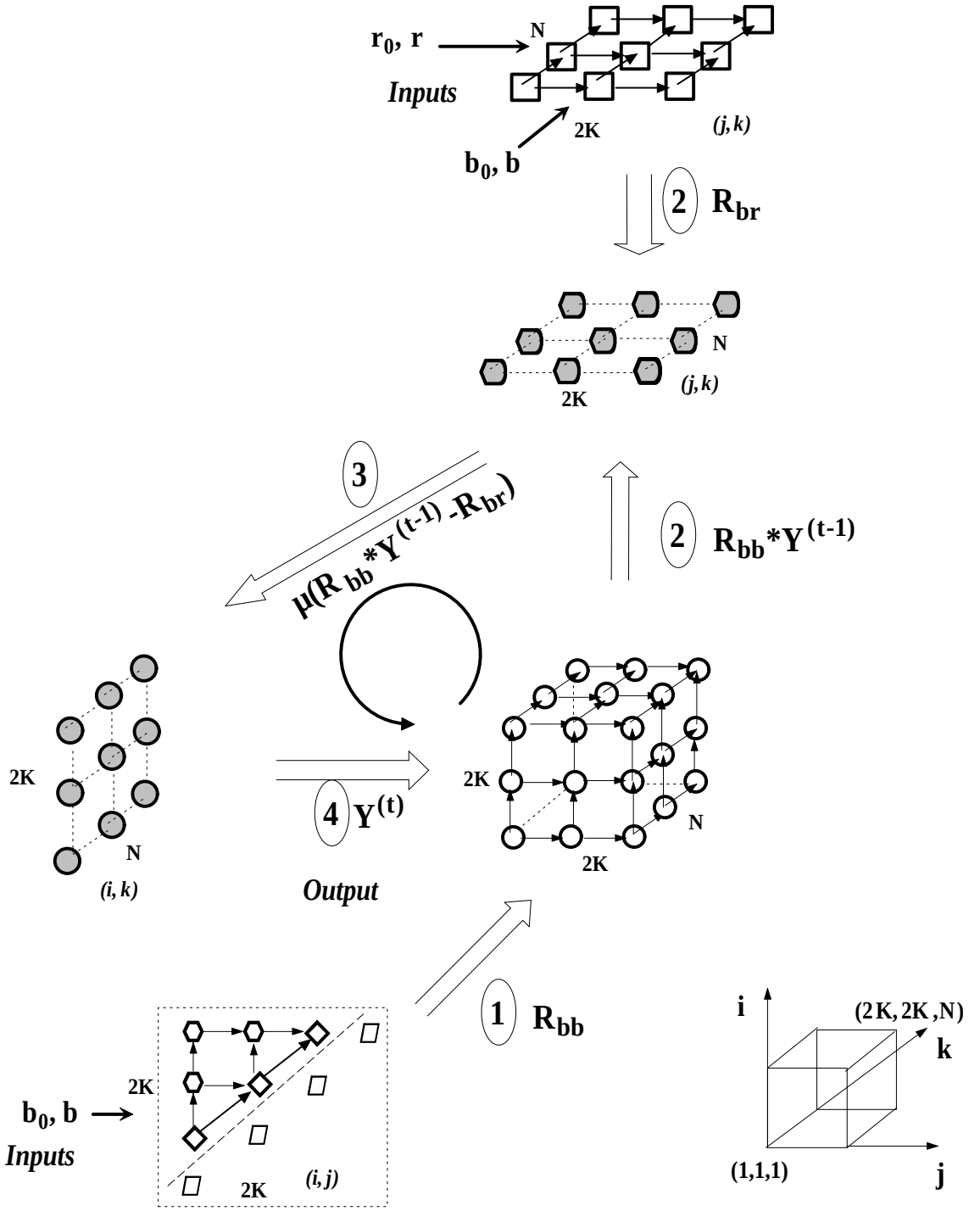

Figure 7: Dependence graph for multiuser channel estimation. This figure shows the DG's for each part of the algorithm. The block arrows represent interconnections from all elements in a plane to the subsequent block. The normal arrows represent a systolic interconnection while the dotted lines represent independent processing elements with no interconnections. The loop represents equation (8)

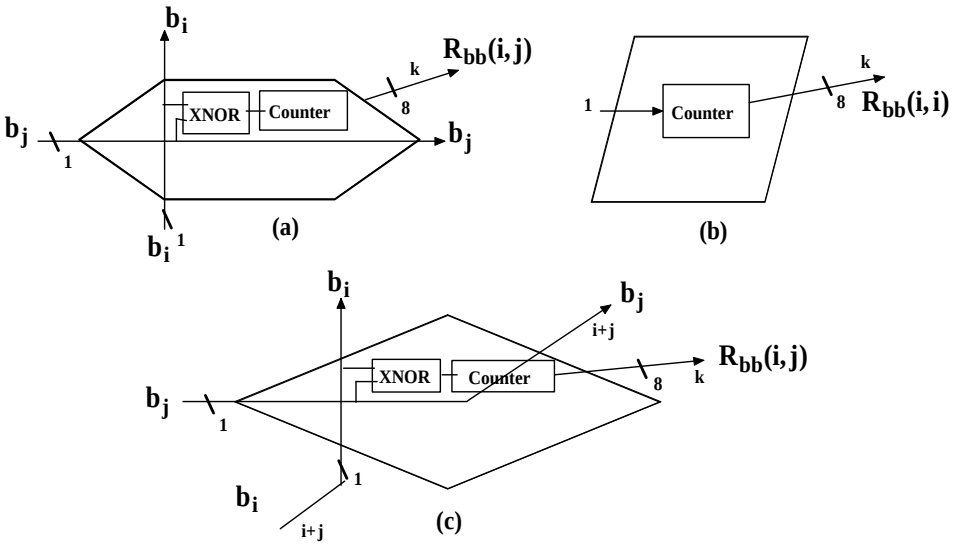

Figure 8: Elements in the auto-correlation block. This figure shows the nodes of the three different processing elements in the auto-correlation block.

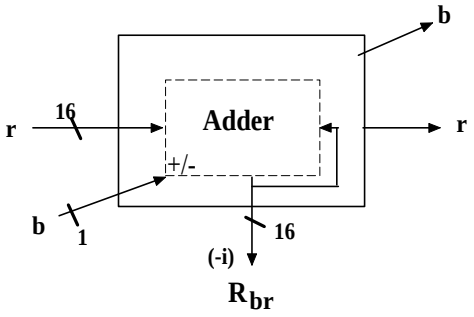

Figure 9: Elements in the cross-correlation block of equation (6)

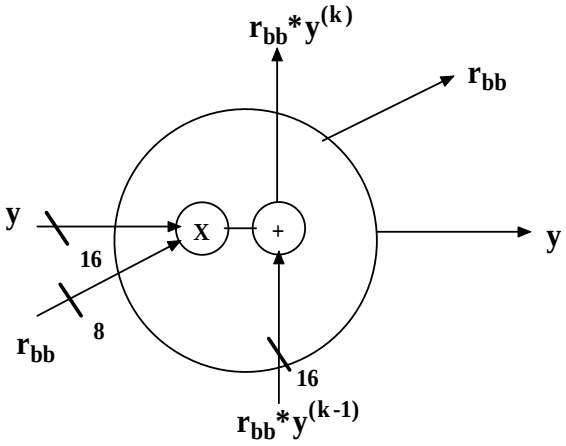

Figure 10: Node for matrix multiplication in equation (8)

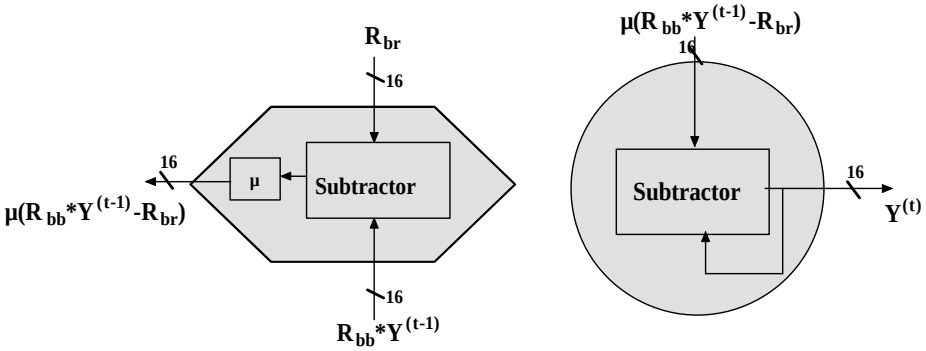

Figure 11: Nodes for non-systolic processing elements

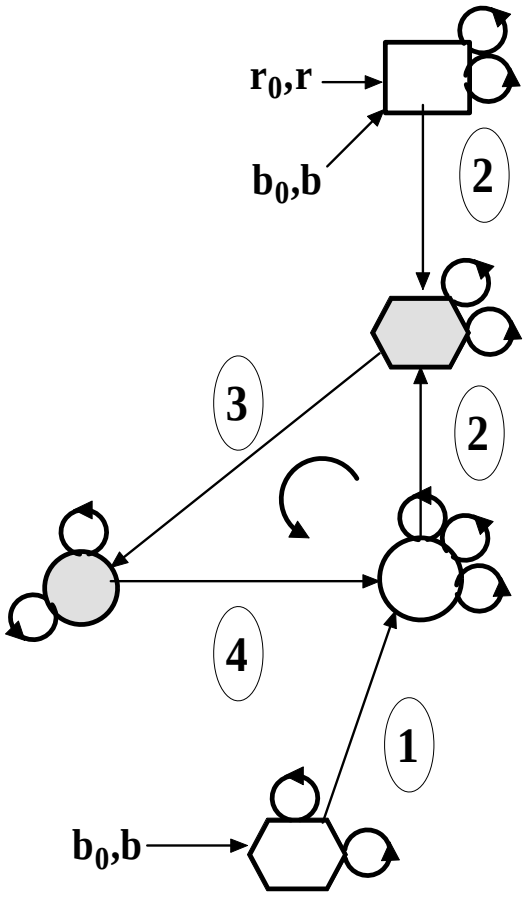

Figure 12: Area-constrained VLSI architecture mapping. This figure shows an area-constrained mapping for multiuser estimation. Each basic node is implemented just once. The iterations in order to compute the elements for the entire array are shown by loops around the node. The location of the loop represents the direction in which an iteration is folded on a single element.

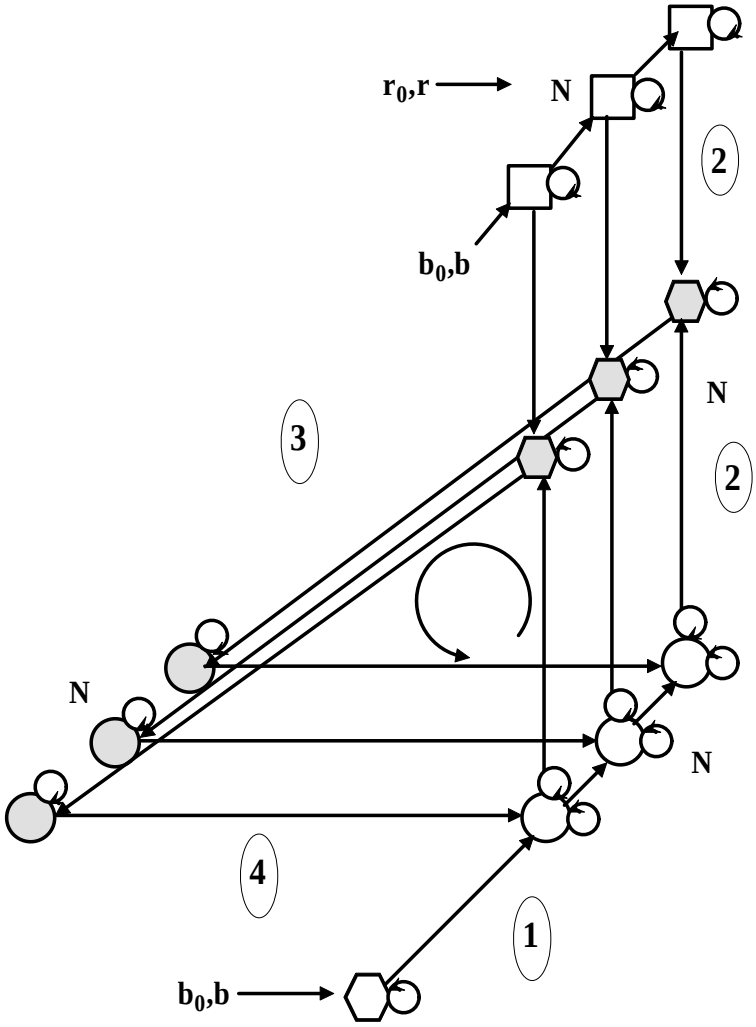

Figure 13: Area-time efficient VLSI architecture mapping. This figure shows an area-time efficient architecture for multiuser estimation. Each basic node is replicated N times. The iterations in order to compute the elements for the entire array are shown by loops around the node. The location of the loop represents the direction in which an iteration is folded on a single element.