On-line Arithmetic for Detection in Digital Communication Receivers

Sridhar Rajagopal and Joseph R. Cavallaro
Rice University
Department of Electrical and Computer Engineering
Houston TX 77005.
{*sridhar,cavallar*}@*rice.edu*

Abstract

This paper demonstrates the advantages of using on-line arithmetic for traditional and advanced detection algorithms for communication systems. Detection is one of the core computationally-intensive physical layer operations in a communication receiver and determines the communication data rates. Detection algorithms typically involve hard decisions (sign based testing) to find the sign of the transmitted information bit. This results in extraneous computations in a conventional number system as the sign is obtained only at the end due to the Least Significant Digit First (LSDF) nature of computations. On-line arithmetic, based on a signed digit number representation, provides Most Significant Digit First (MSDF) computation. Hence, the computations can stop after the first non-zero MSD (sign) is computed and additional computations for the successive digits can be avoided. Back-conversion to a conventional number system is not required as the sign of the digit represents the detected bit. A comparison of a radix-4 serial digit on-line multiuser detector with an 8-bit parallel conventional arithmetic multiuser detector shows a decrease in latency by 1.79X, a 3X increase in throughput, and possible savings in area.

1. Introduction

Next generation communication systems based on W-CDMA (Wideband Code Division Multiple Access) are being designed to increase data rates by orders-of-magnitude and enhance performance significantly in order to support real-time multimedia services [1]. However, the increased complexity required by the algorithms to support multimedia capabilities has not been supported by hardware to achieve real time implementations. Detection is one of the core baseband processing operations in a digital communication receiver and is used to find the transmitted bits from the source after it has been corrupted by noise and other interference present in the channel. The data rates that can be achieved in a communication system are dependent on the speed of detection; therefore, it is critical to accelerate detection to meet real-time performance requirements.

Several advanced schemes have been proposed for detection [12, 16] for future communication receivers that provide better performance in terms of lower bit error rates. Despite their sub-optimality, these algorithms have not been implemented in practical systems due to their high implementation complexity [14]. In this paper, we demonstrate the use of non-conventional redundant number representations using on-line arithmetic to accelerate the implementation of traditional as well as advanced detection algorithms to help meet real-time requirements for next generation communication systems.

The physical baseband layer in a digital communication receiver involves operations to detect and decode the transmitted information bits. Sophisticated algorithms for channel estimation, detection and decoding are applied on the receiver to determine the transmitted bits. Based on these high-precision operations, a *hard decision* (a sign-based test) is made on the transmitted bits for detection, which are typically ± 1's, assuming a Binary Phase Shift Keying (BPSK) system. Thus, higher precision algorithms are used to determine the sign of the result. This approach involves extraneous computations even for finite precision implementations as the sign is obtained only at the end of the computations. Figure 1 shows the main blocks in a digital communication receiver.

On-line arithmetic [5, 9, 11] has been developed and researched extensively over the past two decades. On-line arithmetic has been shown to be very useful for many signal processing applications such as DCT, FFT, CORDIC, filtering and matrix based operations [3, 7]. In conventional arithmetic, all digits of the result must be computed before the next operation can be started. However, in on-line arithmetic, the operands, as well as the results, flow through the computations in a digit-by-digit manner, starting from the most significant digit (MSDF). The advantages of on-

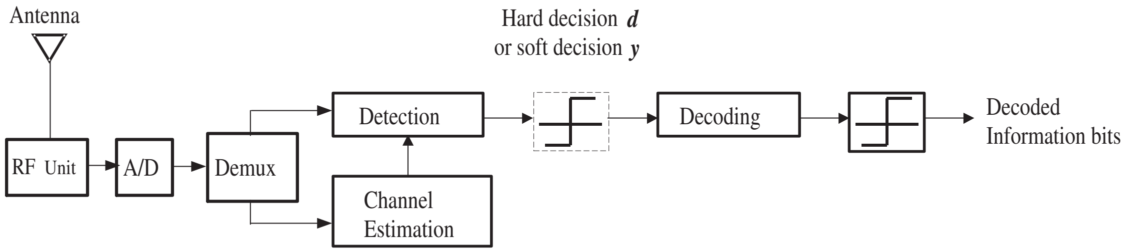

Figure 1. Block diagram of a digital communication receiver. The detector produces either single-bit precision outputs (hard decisions) of the detected bits or higher precision outputs (soft decisions). The final decision on the transmitted information is made at the decoder.

line arithmetic have been to eliminate carry-propagate addition, reduce interconnection bandwidth between modules and allow parallelism between several operations. With a serial data flow, on-line arithmetic can be pipelined to implement sophisticated algorithms. As carry-propagation is eliminated, on-line operations can be overlapped. On-line arithmetic has been shown to provide a speed-up of 2-16X [6] for conventional numerical operations. The implementation tradeoffs related to the applicability of on-line arithmetic are its need for a non-conventional number system, conversions to-and-from a conventional system [10], and the inherently serial nature of the operations.

In this paper, we show that on-line arithmetic also has immense potential for use in detection in communication systems. We present on-line implementations for both traditional and advanced detection algorithms. Because communication systems are special-purpose applications, there is no need to maintain a conventional number representation. On-line arithmetic can be used in a communication receiver with almost the same numerical accuracy and significant performance acceleration in terms of speed and latency reduction. Also, a MSDF representation allows us to stop computations as soon as the first non-zero MSD (sign) has been calculated. This not only avoids the computations of the successive digits, but also eliminates the need for back-conversion to a conventional number system.

2. Background on on-line arithmetic

On-line arithmetic algorithms [5, 7] work in a digit-serial manner, producing the result in a MSDF fashion. To generate the first output digit, δ digits of the input are required. Thereafter, with each digit of the input, a new digit of the result can be obtained. The on-line delay δ is typically a small integer, e.g. 1 to 4. Since the outputs are produced serially, the algorithms can be pipelined with a latency of δ as shown below:

Input x	x_1	x_2	x_3	x_4	x_5	....			
Input y	y_1	y_2	y_3	y_4	y_5	....			
Output z	$\leftarrow$	δ	$\rightarrow$	z_1	z_2	z_3	z_4	z_5	....

In order to achieve MSDF operations, on-line algorithms need to use a redundant number system [2] for carry-free addition. The on-line representation of a number x is given by [5]

$$X_j = X_{j-1} + x_{j+\delta} r^{-\delta-j}$$
$$X_0 = \sum_{i=1}^{\delta} x_i r^{-i}$$

where X_j represents the value of the addends at step j, r is the radix of the redundant number system and δ is the on-line delay. The digits x_i belong to a redundant digit set, $\{$-ρ,.....,-1,0,1,.....,$\rho\}$ (assumed symmetric) where $r/2 \leq \rho \leq r-1$, represents the amount of redundancy in the number system. For our system, we shall assume $\rho = r - 1$ for maximum redundancy as this will allow the inputs to the system to be directly acceptable in redundant form for on-line operations. We choose a radix-4 system to reduce the on-line delay to $\delta = 1$ for multiplication and addition [5] as it requires the least number of gates [8]. In our algorithms, we shall assume fixed point inputs with 8-bit precision as it is shown to be sufficient for most detection implementations [14]. On-line arithmetic, being word-length independent, will demonstrate better performance benefits with higher precision. The operations that need to be performed for detection are on-line addition and multiplication, which are presented in the Appendix.

3. Detection algorithms implemented in a W-CDMA communication receiver

We first show the advantages of on-line arithmetic for a traditional and widely implemented matched filter detector [16] (or the single user detector) as an initial example. Then, we present one of the advanced multiuser detection algorithms proposed to achieve better performance in wireless communication receivers.

3.1. Synchronous matched filter detector

A synchronous detector [12, 16] implies that different users are transmitting their information to the receiver at the same time, i.e. all users are synchronized in time with respect to each other. This makes detection a simpler task as the delays of the different users need not be accounted for. This is a valid assumption for the downlink when a base-station is talking to a mobile receiver. A matched filter detector is also termed as a single user detector as it ignores the effect of interference due to the other users. The detection problem can be viewed as a least squares problem. Let $\mathbf{r}_i \in \mathbb{R}^N$ be the received signal and $\mathbf{A} \in \mathbb{R}^{N\times K}$ be the cross-correlation matrix obtained from channel estimation. N is the length of the spreading code (also known as the spreading gain or the spreading factor) and K is the number of users in the system. Let $\mathbf{d}_i \in \{+1, -1\}^K$ be the bits of the K users to be detected. Then, the system can be formulated as given below:

$$\mathbf{r}_i = \mathbf{A}\mathbf{d}_i + \eta_i. \tag{1}$$

where η is the Additive White Gaussian Noise (AWGN) in the system. So, a least-squares solution to the problem can be derived as

$$\mathbf{A}^H\mathbf{r}_i = \mathbf{A}^H\mathbf{A}\mathbf{d}_i + \mathbf{A}^H\eta_i \tag{2}$$

$$\mathbf{d}_i = sign((\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H(\mathbf{r}_i - \eta_i)). \tag{3}$$

Equation 3 represents the decorrelating detector, which is a multiuser detector. A simple approximation to the decorrelating detector can be obtained by assuming that the cross-correlations are zero (the received signals for each user are independent) and the $\mathbf{A}^H\mathbf{A}$ matrix is identity. This gives rise to the single user detector or the matched filter detector with hard decision outputs:

$$\mathbf{d}_i = sign(\mathbf{A}^H\mathbf{r}_i). \tag{4}$$

Figure 2 shows the architecture of a single user matched filter detector.

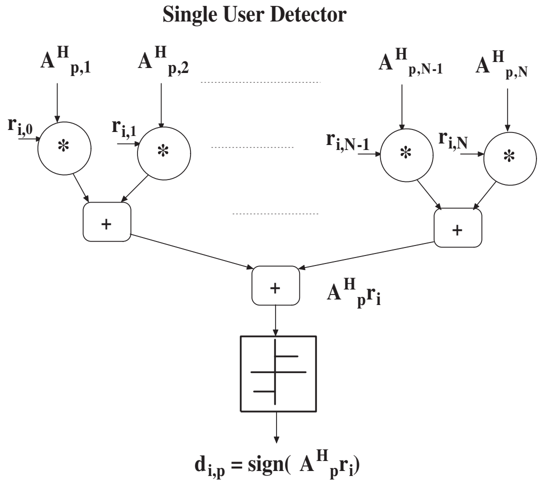

Figure 2. A single user matched filter detector. The figure shows the formation of an element $\mathbf{A}_p^H\mathbf{r}_i$ of the vector $\mathbf{A}^H\mathbf{r}_i$. The subscript 'p' refers to a row index of the matrix $\mathbf{A}_p^H$. See equation (4).

3.2. Asynchronous multiuser detection

In the uplink, when multiple mobile users are communicating with the base-station, the desired user's bits receive interference from the past or future overlapping symbols of other users along with their current symbols because they are asynchronous. This is as shown in Figure 3. Multiuser detection cancels the interference from other users to improve the error rate performance compared to traditional single user detection using only a matched filter [16]. Here, we assume that the delays of the users are coarse-synchronized within one symbol duration. We consider multistage detection [15, 18], based on the principle of Parallel Interference Cancellation (PIC). This scheme cancels the interference from the other users successively in stages and is shown to have computational complexity quadratic with the number of users.

The different delays and phase-shifts make the received signal $\mathbf{r}_i$ and the channel estimates complex numbers. For an asynchronous system with BPSK modulation, the channel estimate can be arranged as $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{C}^{N\times K}$ which corresponds to partial correlation information for the successive bit vectors $\mathbf{d}_{i-1}, \mathbf{d}_i \in \{+1, -1\}^K$, which are to be detected. In vector form, the received signal is

$$\mathbf{r}_i = [\mathbf{A}_0\mathbf{A}_1]\begin{bmatrix} d_{1,i-1} \\ \vdots \\ d_{K,i-1} \\ d_{1,i} \\ \vdots \\ d_{K,i} \end{bmatrix} + \eta_i. \tag{5}$$

3.2.1. Asynchronous matched filter detection

The bits, $\mathbf{d}_i$, of the K users to be detected lie between the received signal $\mathbf{r}_i$ and $\mathbf{r}_{i-1}$ boundaries. The matched

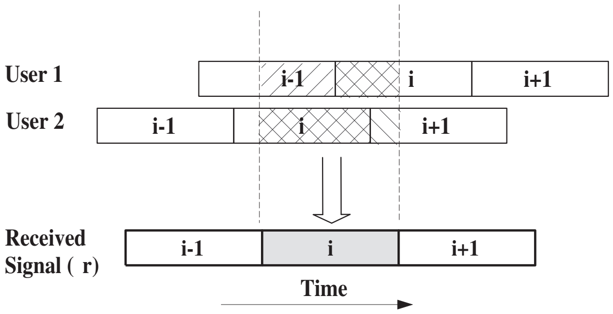

Figure 3. Multiple access interference. This figure shows the interference due to asynchronous delays of the different users. The received signal contains the sum of past, current and future bits (each shaded differently) of all users.

filter detector [12, 16] does a single user detection for each user, neglecting the effects of other users. The synchronous matched filter detector (equation (4)) can be extended to the asynchronous case, producing both soft decisions $\mathbf{y}_i$ and hard decisions $\mathbf{d}_i$.

$$\begin{aligned} \mathbf{y}_i &= \Re[\mathbf{A}_1^H \mathbf{r}_{i-1} + \mathbf{A}_0^H \mathbf{r}_i] \quad (6) \\ \mathbf{d}_i &= sign(\mathbf{y}_i). \end{aligned}$$

We see that the asynchronous nature of users increases the complexity of matched filter detection. Comparing equations (4) and (6), we can see an increase in complexity for the asynchronous matched filter due to the increase in addition.

3.2.2. Multistage parallel interference cancellation

The multistage detector [13, 15, 18] uses the soft decisions $\mathbf{y}_i$ of the matched filter to get an initial estimate of the bits and then subtracts the interference from all other users. The multistage detector performs parallel interference cancellation iteratively in stages.

$$\begin{aligned} \mathbf{y}_i^{(l)} &= \mathbf{y}_i^{(0)} - \mathbf{L}\mathbf{d}_{i-1}^{(l-1)} - \mathbf{C}\mathbf{d}_i^{(l-1)} - \mathbf{L}^T\mathbf{d}_{i+1}^{(l-1)} \quad (7) \\ \mathbf{d}_i{}^{(l)} &= sign(\mathbf{y}_i{}^{(l)}). \quad (8) \end{aligned}$$

Equation (7) may be thought of as subtracting the interference from the past bit of users, who have more delay, and the future bits of the users, who have less delay than the desired user (Refer to Figure 3 and equation (5)). The left matrix $\mathbf{L} \in \mathbb{R}^{K\times K}$, stands for the partial correlation between the past bits of the interfering users and the desired user, the right matrix $\mathbf{R} = \mathbf{L}^T$, stands for the partial correlation between the future bits of the interfering users and

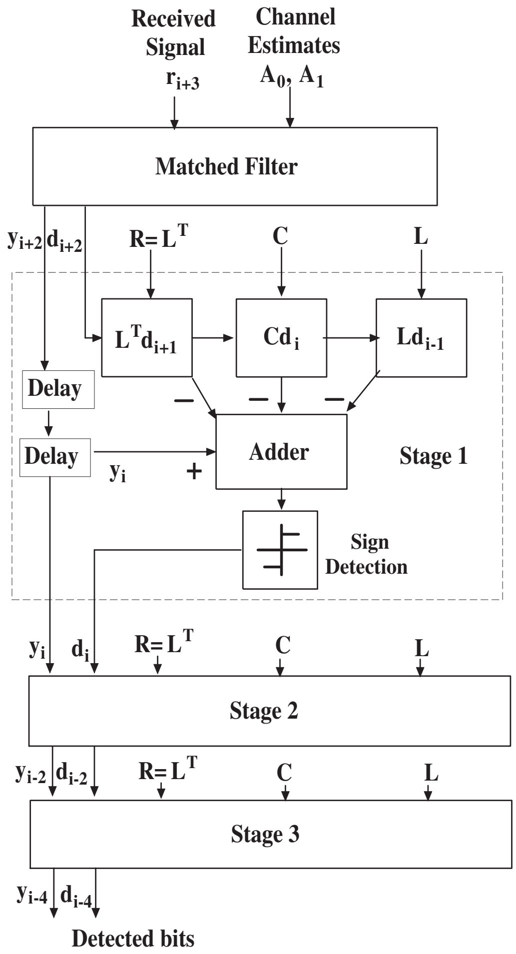

Figure 4. Pipelined multiuser detector architecture. The asynchronous matched filter provides the initial estimates for PIC. A 3-stage PIC detector is shown. The first stage of the detector is expanded in detail.

the desired user. The center matrix $\mathbf{C} \in \mathbb{R}^{K\times K}$, is the correlation of the current bits of interfering users and the diagonal elements are zeros since only the interference from other users, represented by the non-diagonal elements, is canceled. The lower index, i, represents time, while the upper index, l, represents the iterations. We assume $\mathbf{L}$ and $\mathbf{C}$ to be pre-computed as part of channel estimation.

Figure 4 shows the architecture of the multistage multiuser detector [13]. The detector takes in the soft decisions, $\mathbf{y}_i$, and the hard decisions, $\mathbf{d}_i$, from the asynchronous matched filter. The partial correlation matrices $\mathbf{L}, \mathbf{C}$ (channel estimates) are also loaded. The detector subtracts the interference iteratively in stages from the matched filter output. The first stage of PIC needs $\mathbf{y}_i$, $\mathbf{d}_i$, and $\mathbf{d}_{i+1}$ from the

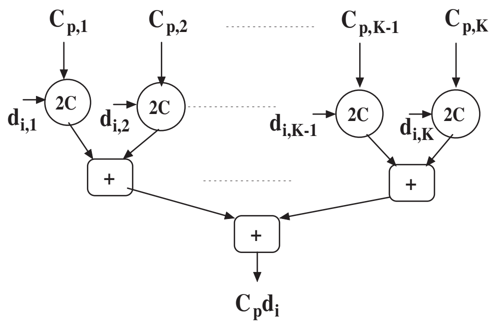

Figure 5. PIC stages of the multiuser detector. The figure shows the formation of an element $\mathbf{C}_p\mathbf{d}_i$ of the vector $\mathbf{C}\mathbf{d}_i$. The subscript 'p' refers to a row index of the matrix $\mathbf{C}$ while 'i' refers to the time index of the detected data vector $\mathbf{d}$.

matched filter to start its computation (equation (7)). This represents a latency of 2 matched filter operations before the first stage can begin its computations. Similarly, each stage of the detector sends the match-filtered estimate and the updated hard decisions to the next stage of the detector with a latency of 2 operations. Three stages of PIC are shown to be sufficient for convergence of multistage detection [18]. The detected bits are then sent to the decoder for retrieving the transmitted information.

Each stage of the multiuser detector uses only adders because multiplication by single bits can be reduced to addition and subtraction. In order to form the various vectors such as $\mathbf{C}d_i$ in equation (7), we can use an adder tree as shown in Figure 5. The multiplication by the bits can be substituted by a 2's complement representation in case the bit is -1 (shown as $2C$ with a circle in the figure) and left unchanged when the bit is +1.

4. Implementation of on-line single user and multiuser detectors

This section compares the advantages of implementing detection using on-line arithmetic. The performance benefits for the traditional single user matched filter detector and for a more advanced, multiuser detector are presented in this section.

4.1. Synchronous matched filter detector

As seen from Figure 2, the computation performed in a matched filter is the sign of a matrix-vector multiplication. For a minimum time implementation, each element of the output vector $\mathbf{d}$ can be computed in parallel in $\log_2(N) + 2$ time, where N represents the length of the spreading code used. A conventional implementation of the detector has a latency time $(\log_2(N) + 2) * \log_2(d) * t_{conv}$ where $\log_2(d) * t_{conv}$ is the time taken to compute d-bit addition and $2 * \log_2(d) * t_{conv}$ for d-bit multiplication. This is assuming a carry look-ahead adder and a Wallace tree multiplier, which computes the result in approximately $O(\log_2(d))$ time [17]. A hard-decision is finally made in the end, and all the successive digits in the output can be disregarded. This latency time also represents the throughput of the conventional arithmetic detector as internal operations are not pipelined.

The benefits of on-line arithmetic stem from computing the sign of the result (the first non-zero MSD) in a MSDF manner. As soon as the first non-zero digit is computed, all succeeding operations at that point can be stopped. The on-line implementation performs the same operation in latency time $(\log_2(N)+1)*t_{OL}+t_{stop}$, where t_{OL} is the time taken for a single digit on-line addition or multiplication and t_{stop} is the approximate time taken for the first non-zero digit of the result to appear.

The timing schedule of operations for the single user detector is as shown in Figure 6. The latency and the throughput of the conventional implementation is $(\log_2(N) + 2) * \log_2(d) * t_{conv}$. The $N*$ represents the N multiplications occurring in parallel, $N/2+$ the next $N/2$ parallel additions and so on. The tree-based nature of computations causes the latency of the on-line scheme to have a delay of $(\log_2(N) + 1) * t_{OL}$ for the first digit of the result to appear. However, the first digit may be a zero and may not represent the sign digit. The time for the sign digit to appear depends on the difference in magnitude and sign of the operands. If the operands have opposite signs and nearly equal magnitude (which could happen when the detected signal is close to the decision region, due to low SNR (Signal to Noise Ratio) or large interference from other users), the first few MSDs will be zeros, and the sign of the result will be known only after calculation of the succeeding digits. Hence, $0 \leq t_{stop} < m * t_{OL}$. As soon as the sign digit of the result is computed, the computations are stopped. Hence, the overall latency for the operations is $(\log_2(N) + 1) * t_{OL} + t_{stop}$. The number of digits m for the operands in a radix-r representation is given by $m = d/log_2(r)$, where d is the bit-precision. Hence, a complete calculation of the full precision of the result takes time $m * t_{OL}$.

Assuming a typical spreading code length of $N = 32$ and $d = 8$-bit precision, the number of digits for the operands in a radix-4 representation is $m = 4$. We assume that the on-line implementation time t_{OL} per digit takes approximately twice the time of conventional implementation t_{conv} per bit [7]. Hence, $t_{OL} \approx 2$, as we choose the basic unit of conventional addition time, $t_{conv} = 1$. The

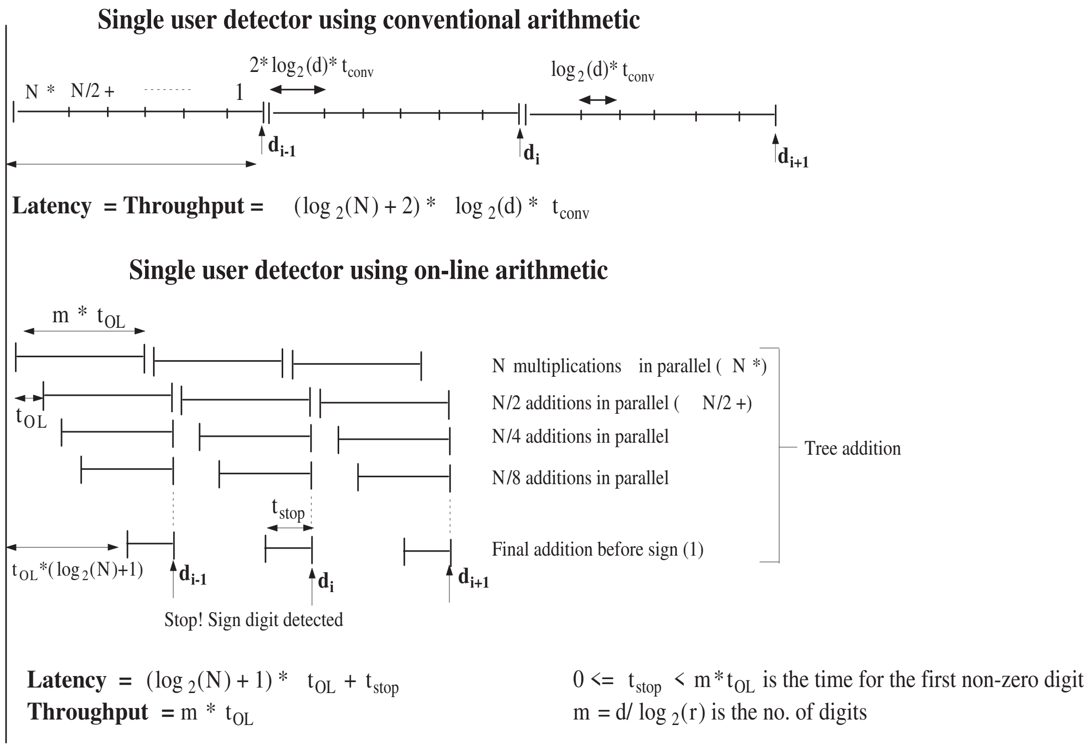

Figure 6. Timing schedule for the synchronous matched filter detector. The figure shows the latency reduction and speedup in throughput for the on-line arithmetic detector.

time taken for t_{stop} depends on the magnitude and sign of the operands. The values of t_{stop} now range between $0 \leq t_{stop} < 4 * 2 = 8$. We assume that, on an average, $t_{stop} = 2$ for an 8-bit implementation. This gives a latency of 21 cycles for the conventional case and 14 cycles for the on-line case. The throughput also improves to 8 cycles for on-line arithmetic from 21 cycles for conventional arithmetic. The benefits of on-line arithmetic increase with increase in N. The latency reduction and the speedup in the throughput for the single user detector due to an on-line arithmetic implementation is given in Table 1.

4.2. Advanced multiuser detection

The timing schedule for multiuser detection is as shown in Figure 7. The figure shows the time steps involved during the computation of the asynchronous matched filter (equation (6)) and three stages ($S = 3$) of parallel interference cancellation (equations (7)-(8)). The figure shows the pipelining of the stages of the multiuser detector in the conventional arithmetic case and the pipelining of both the PIC stages as well as within the stages for the on-line implementation. In order to compute the new $\mathbf{d}_i$, the first stage of the detector must wait until $\mathbf{d}_i$, $\mathbf{y}_i$ and $\mathbf{d}_{i+1}$ of the matched filter are available, as can be seen from equations (7)-(8). For the conventional arithmetic implementation, since the hard decision $\mathbf{d}_{i+1}$ is available only at the end of the computation, the PIC stage needs to wait for 2 conventional matched filter operations t_{CMF} before it can start its computation, where $t_{CMF} = (\log_2(N) + 3) * \log_2(d) * t_{conv}$. Similarly, each stage of the PIC takes 2 conventional PIC operations t_{CPIC} to start the new $\mathbf{d}_i$, where $t_{CPIC} = (\log_2(K)+3)*\log_2(d)*t_{conv}$ as shown in Figure 7. Hence, the overall system latency to generate the result is given by $(2*S-1)*t_{CPIC}+2*t_{CMF}$, where S is the number of PIC stages. Also, as the stages are pipelined, the throughput of the conventional detector is t_{CMF} as typically, the spreading factor is greater than the number of users ($N \geq K$).

For the on-line multiuser detector, the first stage of the PIC can begin its computation as soon as the sign $\mathbf{d}_{i+1}$ is obtained from the matched filter, without waiting for the completion of the digits to $\mathbf{y}_{i+1}$. Thus, the first stage can begin its computation immediately after t_{stop} of the $(i+1)^{th}$ computation. Similarly, each stage of the PIC can begin computing $\mathbf{d}_i$ as soon as the $\mathbf{d}_{i+1}$ of the previous stage has finished. Thus, starting after t_{stop} helps to reduce the overall latency of $t_{MF} + m * S * t_{OL} + S * t_{PIC}$. Again, the throughput for the on-line detector is $m * t_{OL}$ as it was for

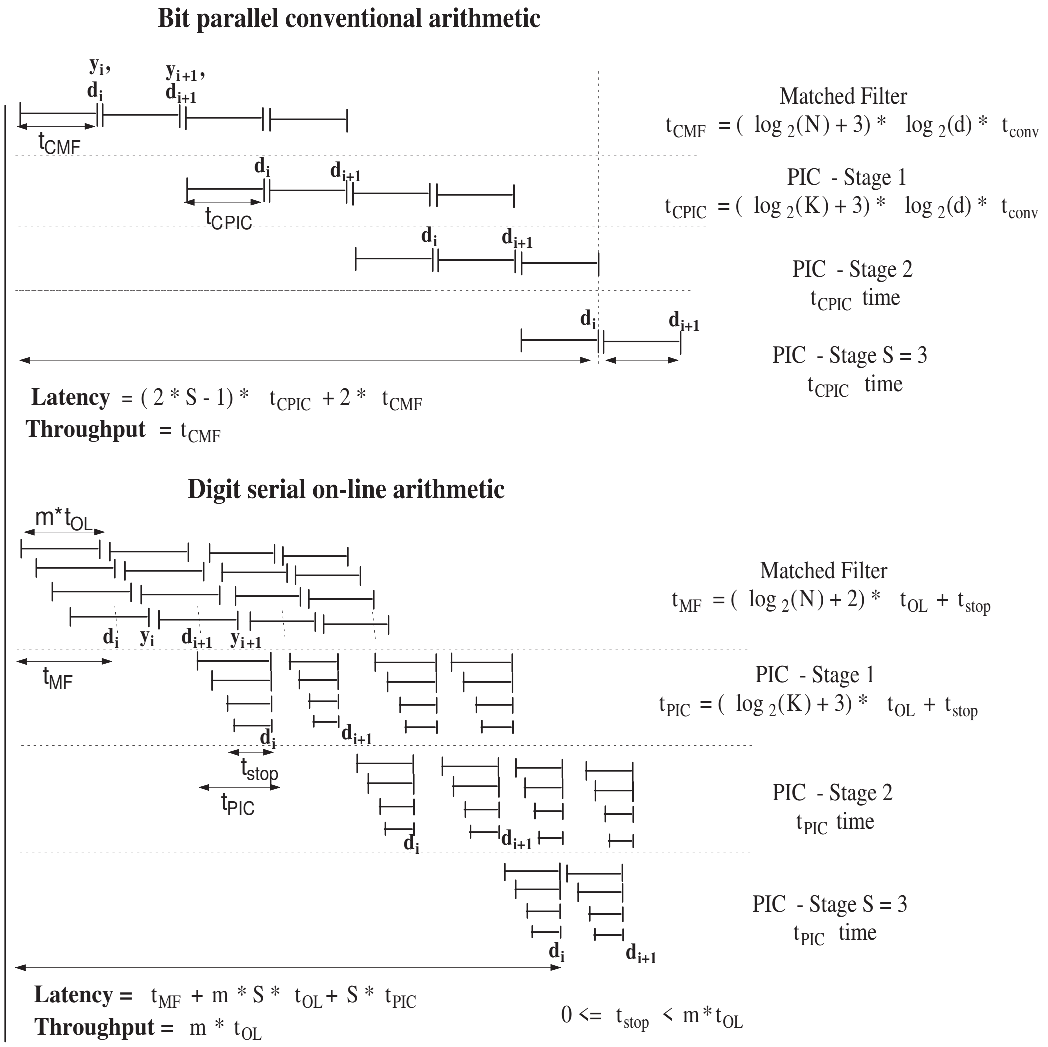

Figure 7. Timing schedule for multiuser detection. This figure shows the latency reduction and speedup in throughput for the on-line arithmetic detector.

the synchronous detector (Figure 6).

Assuming a three-stage detector ($S = 3$), we get a latency of 168 cycles and a throughput of 24 cycles for the conventional implementation. The on-line implementation has a reduced latency of 94 cycles and a throughput of 8 cycles. The speed advantages and comparisons with the synchronous single user detector are shown in Table 1. We can see that using on-line arithmetic achieves a 1.79X reduction in latency and a 3X speedup over the conventional detector implementation. Hence, the benefits of on-line arithmetic are greater with advanced detection algorithms, providing significant decrease in latency and increase in speed.

Savings in area are also possible due to the digit serial nature of on-line computations as we use only a single-digit radix-4 on-line adder and multiplier as opposed to a d-bit fully parallel conventional adder and multiplier for each element in the tree-based computations of the detection algorithms.

5. Extensions to higher modulation schemes

The algorithms for detection presented in this paper assumed BPSK transmission, which is currently used in IS-95 based CDMA systems. However, on-line arithmetic can also be used for QPSK (Quadrature Phase Shift Keying) proposed for future W-CDMA systems. For QPSK, the sign of the real as well as the imaginary component of the received signal is used to form the decision region in a manner

Table 1. Latency reduction and throughput speedup for different detection schemes. $N = K = 32,\ d = 8,\ S = 3,\ r = 4,\ t_{OL} = 2,\ t_{conv} = 1, t_{stop} = 2.$

Detector		Conventional	On-line	Speedup
Single User	Latency	$(\log_2(N) + 2)*$ $\log_2(d) * t_{conv} = 21$	$(\log_2(N) + 1)*$ $t_{OL} + t_{stop} = 14$	1.5
	Throughput	$(\log_2(N) + 2)*$ $\log_2(d) * t_{conv} = 21$	$m * t_{OL} = 8$	2.625
Multi-user	Latency	$(2 * S - 1) * t_{CPIC}$ $+2 * t_{CMF} = 168$	$t_{MF} + S * t_{PIC}$ $+m * S * t_{OL} = 94$	1.79
	Throughput	$t_{CMF} = 24$	$m * t_{OL} = 8$	3

similar to the BPSK scheme. On-line arithmetic can also be used in M-ary QAM (Quadrature Amplitude Modulation) proposed for wireless LAN (Local Area Network). This is because the decision regions in a square QAM are also squares. The knowledge of the square that the received signal will lie in can be obtained from the first few MSDs and the successive operations can be stopped. The closer the detected signal is to the decision region, the longer it will take to obtain the sign as the MSDs will be zeros. Thus, the throughput of the detector depends on the proximity of the detected signal with the decision region. Going to higher modulation schemes also implies that greater bit-precision is needed at the receiver. This need further motivates the use of on-line arithmetic for detection as the throughput is now independent of the bit-precision.

For higher M-PSK systems, the analysis becomes complicated as the decision becomes angle-based. However, it may be possible to work in polar coordinates and use on-line arithmetic. On-line algorithms for complex number arithmetic [11] using redundant complex number systems (RCNS) may also be utilized for these higher modulation schemes. The benefits of on-line arithmetic may reduce if interleaving or block computations requiring large latencies are performed on the received signal. We are investigating the case for higher modulation schemes as future work.

6. Summary and future work

This paper shows the potential benefits of on-line arithmetic for sign-based testing operations in a digital communication system. Specifically, we present the advantages of using on-line arithmetic for traditional and advanced detection algorithms for digital communication systems. A comparison of a single digit on-line multiuser detector to an 8-bit precision conventional multiuser detector shows a reduction in latency by 1.79X, a speedup of 3X in the throughput and possible savings in area. The overhead of returning back to a conventional number representation is also not required. However, the throughput of the detector is not constant as it depends on the proximity of the detected signal to the decision region (i.e. depends on the number of zeros in the MSDs).

The blocks preceding detection and decoding in Figure 1 can also be implemented using on-line arithmetic to form an entire communication receiver using on-line arithmetic. The inputs generated serially from the A/D converter are also inherently MSDF and hence, the on-line algorithms can increase the overall speed by overlapping computations with the conversion. Other signal processing applications involving sign-based computations can also benefit from an on-line arithmetic approach.

We are currently investigating a complete physical layer implementation of the communication receiver with estimation, detection and decoding chain of blocks using on-line arithmetic. The amount of precision required for the algorithms is different at intermediate stages of the blocks and is also dependent on the channel conditions and the number of users in the system and hence, may vary with time. An on-line implementation can be extremely beneficial in this scenario due to the digit-serial nature of computation and hence, the precision can be made programmable. Thus, a system with high communication data rates and latency reduction can be designed using on-line arithmetic for digital communication receivers.

7. Appendix

The appendix is used to describe the fixed point on-line addition and multiplication schemes used for detection.

7.1. On-line addition

The steps involved in fixed point on-line addition [4, 5] of 2 inputs a and b, giving an output c are:

Initialization: $P_{-1} = c_{-2} = c_{-1} = 0$

Recurrence

for j = 0,1,..... ,m+1 do

$$\begin{aligned} W_j &= r * P_{j-1} + r^{-\delta} * (a_j + b_j) \\ c_{j-1} &= select(W_j) \\ P_j &= W_j - c_{j-1} \end{aligned}$$

where the selection is done by rounding. A discretization algorithm (DIS) [4] performs the rounding based on the first few MSDs. The selection function is given by:

$$\begin{aligned} select(W_j) &= sign(W_j)\lfloor |W_j| + 1/2 \rfloor \quad \mathit{if}\ |W_j| \leq \rho \\ &= sign(W_j)\lfloor |W_j| \rfloor \qquad otherwise. \end{aligned}$$

7.2. On-line multiplication

The steps involved in on-line multiplication [4, 5] are:

Initialization: $P_{-1} = c_0 = A_0 = B_0 = 0$

Recurrence

for j = 1,..... ,m do

$$\begin{aligned} W_j &= r * P_{j-1} + < A_{j-1}, a_j > * b_j + B_{j-1} * a_j \\ B_j &= < B_{j-1}, b_j > \\ c_j &= select(W_j) \\ P_j &= W_j - c_j \end{aligned}$$

where the selection function is the same as in on-line addition.

8. Acknowledgments

We are extremely grateful to the reviewers, Dr. Behnaam Aazhang and Dr. Tracy M. Volz for their comments and guidance on this paper. 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] F. Adachi, M. Sawahashi, and H. Suda. Wideband DS-CDMA for next-generation mobile communication systems. *IEEE Communications Magazine*, 36(9):56–69, September 1998.

[2] A. Avizienis. Signed-Digit number representations for fast parallel arithmetic. *IRE Transactions on Electronic Computers*, EC-10(3):389–400, September 1961.

[3] J. Bruguera and T. Lang. 2-D DCT using on-line arithmetic. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 3275–3278, Detroit, MI, May 1995.

[4] M. D. Ercegovac. A general hardware-oriented method for evaluation of functions and computations in a digital computer. *IEEE Transactions on Computers*, 26(7):667–680, July 1977.

[5] M. D. Ercegovac. On-line arithmetic: An overview. In *Real Time Signal Processing VII, SPIE*, volume 495, pages 86–93, San Diego, CA, August 1984.

[6] M. D. Ercegovac and A. L. Grnarov. On the performance of on-line arithmetic. In *International Conference on Parallel Processing*, pages 55–62, Harbor Springs, MI, August 1980.

[7] M. D. Ercegovac and T. Lang. On-line arithmetic: A design methodology and applications in digital signal processing. *VLSI Signal Processing III*, pages 252–263, November 1988. IEEE Press.

[8] A. Gorji-Sinaki and M. D. Ercegovac. Design of a digit-slice on-line arithmetic unit. In 5^{th} *IEEE Symposium on Computer Arithmetic*, pages 72–80, Ann Arbor, MI, May 1981.

[9] T. Lynch and M. J. Schulte. A high radix on-line arithmetic for credible and accurate computing. *Journal of Universal Computer Science*, 1(7):439–453, July 1995.

[10] M. D. Ercegovac and T. Lang. On-the-Fly conversion of redundant into conventional representations. *IEEE Transactions on Computers*, C-36(7):895–897, July 1987.

[11] R. McIlhenny and M. D. Ercegovac. On-line algorithms for complex number arithmetic. In 32^{nd} *Asilomar Conference on Signals, Systems and Computers*, pages 172–176, Pacific Grove, CA, October 1998.

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

[13] 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.

[14] I. Seskar and N. B. Mandayam. A software radio architecture for linear multiuser detection. *IEEE Journal on Selected Areas in Communication*, 17(5):814–823, May 1999.

[15] M. K. Varanasi and B. Aazhang. Multistage detection in asynchronous code-division multiple-access communications. *IEEE Transactions on Communications*, 38(4):509–519, April 1990.

[16] S. Verdú. *Multiuser detection*. Cambridge University Press, 1998.

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

[18] G. Xu and J. R. Cavallaro. Real-time implementation of multistage algorithm for next generation wideband CDMA systems. In *Advanced Signal Processing Algorithms, Architectures, and Implementations IX, SPIE*, volume 3807, pages 62–73, Denver, CO, July 1999.