Truncated Online Arithmetic with Applications to Communication Systems

Sridhar Rajagopal, *Member*, *IEEE*, and Joseph R. Cavallaro, *Senior Member*, *IEEE*

Abstract—Truncation in digit-precision is a very important and common operation in embedded system design for bounding the required finite precision and for area-time-power savings. In this paper, we present the use of online arithmetic to provide truncated computations with communication systems as one of the applications. In contrast to truncation in conventional arithmetic, online arithmetic can truncate dynamically and produce both area and time benefits due to the digit-serial nature of computations. This is of great advantage in communication systems where the precision requirements can change dynamically with the environment. While truncation in conventional arithmetic can have significant truncation errors, especially when the output precision is less than the input precision, the redundancy and most significant digit first nature of online arithmetic restricts the truncation error to only the least significant digit of the truncated result. As an application that uses significant truncation in precision, a code matched filter detector for wireless systems is designed using truncated online arithmetic. The detector can provide both hard decisions and soft(er) decisions dynamically as well as interface with other conventional arithmetic circuits or act as a DSP coprocessor. Thus, optimized communication receivers with coexisting conventional arithmetic for saturation and online arithmetic for truncation can now be built. The truncated online arithmetic detector was also verified with a VLSI implementation in an AMI 0.5 μ MOSIS tiny chip process.

Index Terms—Dynamic truncation, finite precision, online arithmetic, communication systems.

1 Introduction

Communication system designs typically use finite (and, preferably, fixed-point) precision arithmetic to provide faster, simpler, and low power hardware. Truncation and saturation in digit-precision are two very important and common operations in such systems as they help to bound the finite precision while simultaneously providing area-time-power savings. Various designs have been proposed in conventional arithmetic to support truncation and saturation in units such as multipliers [1], [2]. These designs can provide savings in area by 25-35 percent and power dissipation by 29-40 percent [3]. Truncation is used when the Most Significant Bits (MSBs) of the result are important, while saturation is used when the Least Significant Bits (LSBs) of the result are needed. The precision requirements of the algorithms in communication systems are dependent on several dynamic factors, such as the instantaneous signal-to-noise ratio (SNR). Thus, if time, area, and power benefits were to be available by dynamic saturation and dynamic truncation, that would be of immense use to communication system design.

Dynamic saturation is a simpler problem to solve in conventional arithmetic for time and power benefits. This is because if fewer LSBs are needed, the input precision can be decreased for time and power benefits. For example, in [4], different regions of the multiplier are dynamically deactivated depending on the precision of the operands, giving up to 35 percent power savings. Current truncation schemes in conventional arithmetic do not provide the ability to modify the amount of truncation dynamically. The only related work known to us has been in switching between IEEE floating-point rounding and truncated floating-point arithmetic [5]. Even if dynamic truncation were possible by dynamically turning off parts of the circuit, truncation in conventional arithmetic is ineffective in providing throughput benefits as the MSBs are obtained only at the end of computations. Hence, in this paper, we focus only on truncated arithmetic. In particular, we explore the use of online arithmetic to provide dynamic truncation while simultaneously providing throughput benefits.

As can be seen from Fig. 1, truncation in conventional arithmetic is difficult as the most significant digits are computed only at the end due to the right-to-left flow of computations. Here, truncation is achieved by eliminating the hardware for calculating the lower significant digits while providing corrections for the error introduced. However, in online arithmetic [6], the operands, as well as the results, flow through the computations in a digit-by-digit manner, starting from the most significant digit (MSDF), providing a natural way of truncating the results of the computation without introducing significant error. Due to the digit serial nature of computations, we can provide dynamic truncation of digits while simultaneously providing throughput benefits. Though online arithmetic has been quite well-explored in the past, we are unaware of any past work that has exploited the MSDF computation nature of online arithmetic for providing dynamic truncation and its related trade-offs.

Dynamic truncation is not possible in conventional arithmetic due to the right-to-left flow of computations.

- *S. Rajagopal is with WiQuest Communications, Inc., 915 Enterprise Blvd., Suite 200, Allen, TX 75013. E-mail: sridhar.rajagopal@wiquest.com.*
- *J.R. Cavallaro is with the Electrical and Computer Engineering Department, Rice University, 6100 Main St. MS-380, Houston, TX 77251-1892. E-mail: cavallar@rice.edu.*

Manuscript received 21 Dec. 2004; revised 19 Jan. 2006; accepted 19 May 2006; published online 22 Aug. 2006.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0420-1204.

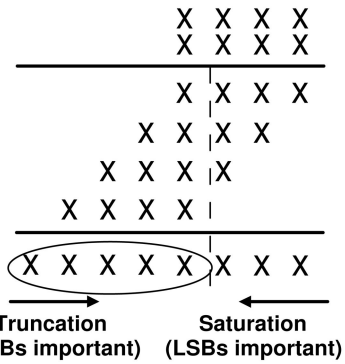

Fig. 1. Conventional truncated arithmetic for a 4-bit multiplication as an example. Truncated arithmetic is used when MSBs are important, while saturated arithmetic is used when LSBs are important.

Even static truncation in conventional arithmetic can have significant errors in the least significant bits of the truncated result and may need additional hardware to provide correction constants for error reduction [1], [3]. Also, the truncation error (without correction) increases with increased truncation, especially when the precision of the result is less than the input precision. Truncation in online arithmetic produces error only in the least significant digit used in the result due to the MSDF flow of computations and the redundancy in the number system. We quantify the error introduced in the least significant digit during truncation using online arithmetic and show its applicability and use in designing communication systems supporting dynamic truncation with online arithmetic. Thus, a wide variety of algorithms for communication systems requiring truncation in finite precision, such as FIR filters, FFT, vocoders, matched filters, and other sophisticated communication algorithms, such as multiuser detection [7], [8], [9], [10], can benefit from dynamic truncation and throughput improvements using online arithmetic [11], [12], [13].

The main contributions of our work are as follows: We present truncated online arithmetic and we show that it has significant potential for dynamic truncation with throughput benefits in designing finite precision embedded systems. We also quantify and limit the truncation error to the error in the least significant digit of the truncated result. We then design a code-matched filter detector as an example of truncated arithmetic for communication systems and show the effect of truncation on the design. We present the online arithmetic adders and multipliers used in the design with support for dynamic truncation and the ability to interface with other conventional arithmetic circuits, such as a DSP coprocessor. We finally discuss the implementation trade-offs against conventional arithmetic using truncation and present a VLSI implementation of the code matched filter detector in an AMI 0.5 μ process.

We have organized this paper as follows: The next section presents the background on the use of online arithmetic for truncation and a code matched filter detector for communication systems. Section 3 presents the effect of truncation error on the least significant digit and how it affects the bit error rate of the detector. Section 4 shows the radix-4 online adder and multiplier design used in the detector with support for dynamic truncation and interface to conventional arithmetic. In Section 5, the area-time trade-offs of truncation in conventional and online arithmetic based detector architectures are presented. Section 6 shows a VLSI design and implementation of the online arithmetic detector.

2 Background

Online arithmetic [6], [14], [15] has been shown to be very useful for many signal processing applications such as DCT, FFT, CORDIC, filtering, and matrix-based operations [11], [12], [16]. The advantages of online arithmetic have been to eliminate carry-propagate addition, reduce interconnection bandwidth between modules, and allow parallelism between several operations. With a serial data flow, online arithmetic can be pipelined to implement sophisticated algorithms. As carry-propagation is eliminated, online operations can be overlapped. Though online arithmetic has been shown to provide a speedup of $2\times - 16\times$ [17] for conventional numerical operations, this gain is reduced when conventional arithmetic systems are deeply pipelined. This is because the word-parallel logarithmic-time computations attain better throughput than the digit-serial online computations (though at the expense of larger area and higher latency). The other implementation trade-offs related to the applicability of online arithmetic are its need for a nonconventional number system, conversions to-and-from a conventional system [18], and the inherently serial nature of the operations.

Online arithmetic algorithms [6], [11] 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 online 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:

$$\begin{array}{rcccccccccc} \text{Input} & x & x_1 & x_2 & x_3 & x_4 & x_5 & \dots & & & \\ \text{Input} & y & y_1 & y_2 & y_3 & y_4 & y_5 & \dots & & & \\ \text{Output} & z & \leftarrow & \delta & \rightarrow & z_1 & z_2 & z_3 & z_4 & z_5 & \dots. \end{array}$$

In order to achieve MSDF operations, online algorithms need to use a redundant number system [19] for carry-free addition. The online representation of a number x is given by [6]

$$X_j = X_{j-1} + x_{j+\delta} r^{-\delta-j}$$

$$X_0 = \sum_{i=1}^{\delta} x_i r^{-i},$$

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 online delay. The digits x_i belong to a redundant digit set, $\{-\rho, \dots, -1, 0, 1, \dots, \rho\}$ (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 eliminate the need for on-the-fly conversion from nonredundant to redundant form and will allow the inputs to the system to be directly processed as if they were in redundant form for online operations. Although other values of ρ can be chosen to

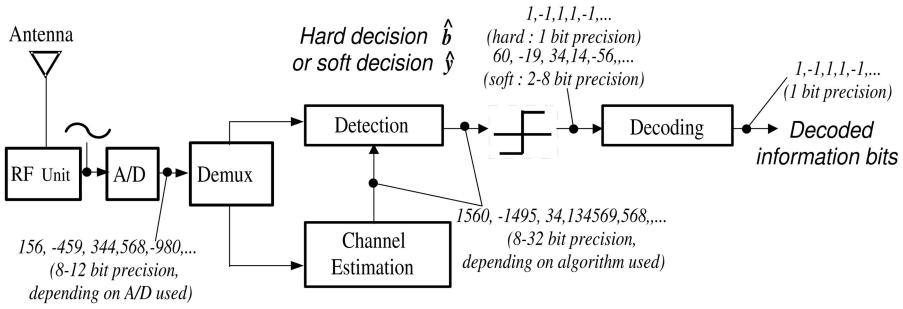

Fig. 2. Block diagram of a communication receiver. The detector may dynamically require either single-bit precision outputs (hard decisions) of the detected bits or higher precision outputs (soft decisions), depending on the received signal-to-noise ratio (SNR). Precision requirements first increase and then decrease during computations in the receiver chain. The decrease in precision may be obtained either by truncation or by saturation (depending on the algorithm and the designer).

simplify the operations, the choice of ρ does not limit the contributions of this paper. The choice of ρ was made based on simplifying the discussion in the paper and more benefits may be obtained with an alternate choice of ρ.

2.1 Communication System Example: Code Matched Filter Detector

Fig. 2 shows the main blocks in a digital communication receiver. The physical baseband layer in a 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 ± 1s, assuming a Binary Phase Shift Keying (BPSK) modulation system for simplicity. The precision requirements at different points in the receiver chain are shown in Fig. 2. We can observe that the precision requirements first increase and then decrease during computations in a receiver chain. The actual precision used depends on several factors such as the signal-to-noise ratio, the A/D converters used, and the algorithms implemented, with more advanced algorithms requiring greater amounts of precision to maintain numerical accuracy. This varying precision requirement makes communication systems a very suitable application for online arithmetic due to its digit-serial nature and the ability for dynamic truncation.

One of the most popular and simple code matched filters [20], [21] used in a Code Division Multiple Access (CDMA) system is mentioned as an example for truncation. The code matched filter is useful for both the base-station and the mobile handset and can be used as an initial estimate for more advanced handset schemes [22], [23], [24], [25]. Let $\mathbf{r}_i \in \mathbb{C}^N$ be the received signal and $\hat{\mathbf{A}} \in \mathbb{C}^{N \times K}$ be the true cross-correlation matrix. 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 $\mathbf{b}_i \in \{+\mathbf{1}, -\mathbf{1}\}^K$ be the bits of the K users to be detected ($K = 1$ for the mobile handset). Then, the system can be formulated as given below:

$$\mathbf{r}_i = \mathbf{A}\mathbf{b}_i + \eta_i, \tag{1}$$

where η is the Additive White Gaussian Noise (AWGN) in the system. The single user detector or the matched filter detector with hard decision outputs is shown to be:

$$\hat{\mathbf{b}}_i = sign(\hat{\mathbf{A}}^H \mathbf{r}_i), \tag{2}$$

where $\hat{\mathbf{b}}$ and $\hat{\mathbf{A}}$ refer to the estimate of the detected bit and the estimate of the channel, respectively (as opposed to the true value). Fig. 3 shows the architecture of a single user matched filter detector which exhibits an inner (dot) product structure for computations followed by sign-based testing. The subscript, p, refers to the pth user in the system. The received signal coming from the A/D converter typically has 8-bit or greater precision [26]. Depending on the algorithm implementation and the desired bit error rate performance, the precision requirements of the hardware processing the received signal are typically in the 8-32 bit precision range for finite precision implementations (see Fig. 2) such as those in [10], [27], [28], with more sophisticated algorithms requiring greater precision for operations such as division and matrix inversions. This implies 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. Online arithmetic, based on a signed digit number representation, provides Most Significant Digit First (MSDF) computation. Hence, the computations can stop after the first nonzero MSD (sign) is computed and additional computations for the successive digits are avoided. The need for back-conversion to a conventional number system is also not required as the sign of the digit represents the detected bit.

2.2 Truncation Using Online Arithmetic

Fig. 4 shows the suitability of online arithmetic for truncation. Fig. 4a shows the timing schedule of a truncated conventional arithmetic matched filter detector shown in Fig. 3 in the previous section. An implementation based on conventional arithmetic has throughput as a logarithmic function of the precision as both fast adders and multipliers can be built for logarithmic time computation, i.e., $t_{CONV-MF} \propto log(d)$ [29], [30], [31], where d is the bit-precision. The truncation error depends on the amount of correction support used. Fig. 4b shows the application of online arithmetic for a full precision matched filter

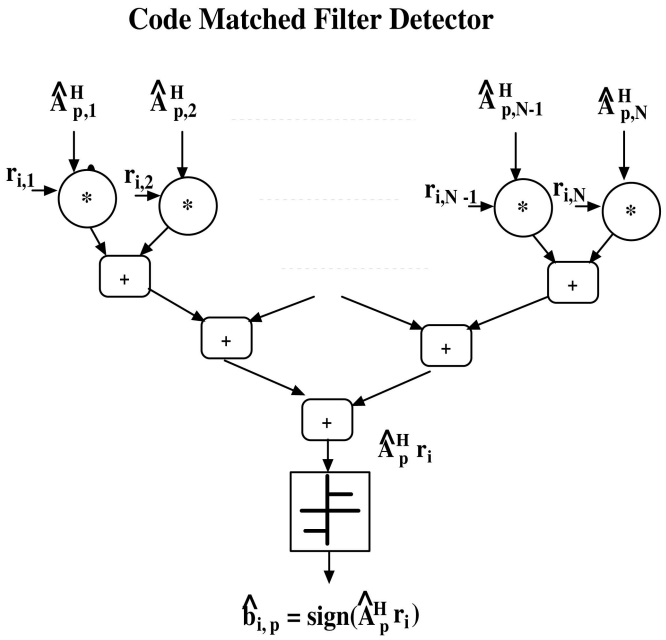

Fig. 3. A code matched filter detector for the $p\text{th}$ user in the system. The operations involve testing a dot product for its sign.

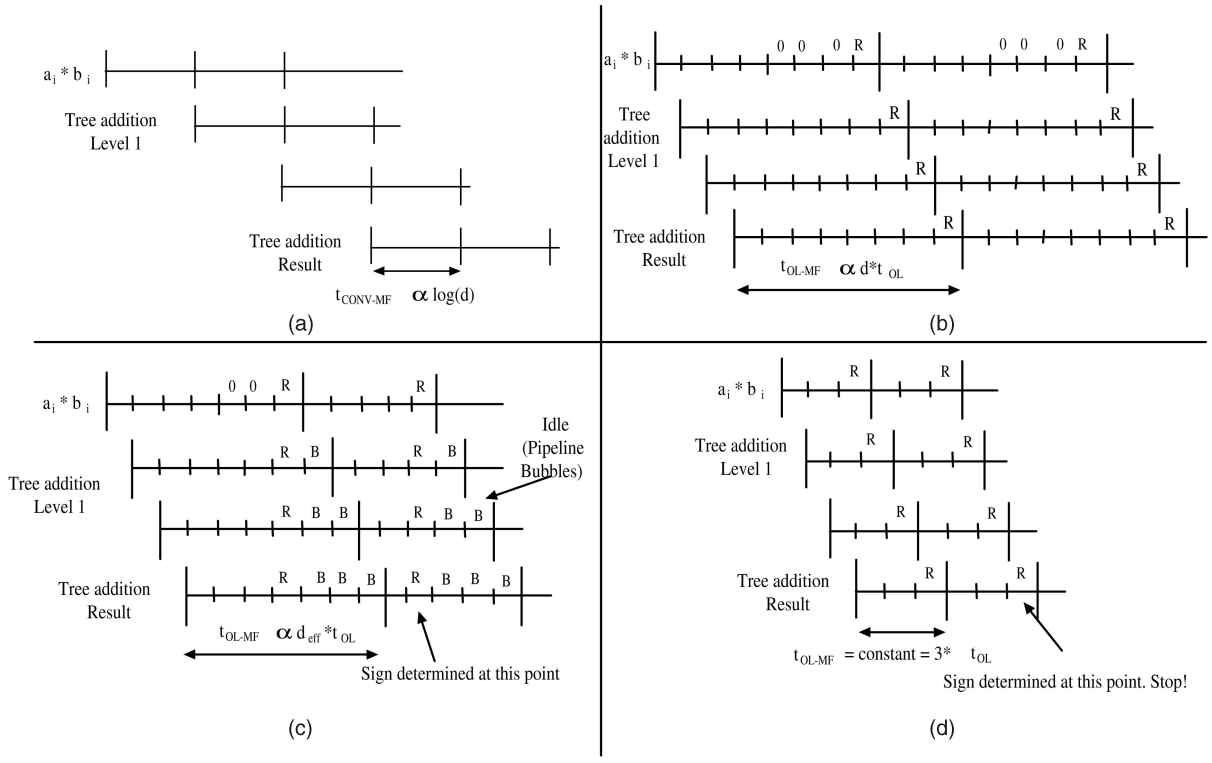

Fig. 4. Timing comparisons for truncation using different arithmetic techniques. Part (a) shows the time taken by a truncated conventional arithmetic circuit for a dot product. Part (b) shows the time taken by online arithmetic for calculating a dot product with full precision. Part (c) shows the time taken by a dynamically truncated online arithmetic detector without any truncation error. Part (d) shows the time taken by a truncated online arithmetic detector with truncation error. **R**: RESET to signify end of current number, **B**: pipeline bubble.

implementation. As an example, a radix-4 system and 8-bit precision is assumed, giving 4 digits per input. Though the full precision implementation has a low latency, the throughput of the system for repeated operations is a linear function of the precision, i.e., $t_{OL-MF} \propto d * t_{OL}$, where t_{OL} is the online throughput time per digit. Note that one digit per computation is wasted as it signifies the end of the current computation precision (shown by $\text{RESET}(R)$ in Fig. 4, which clears all latches at the end of the computation). Also, additional 0s have to be inserted for full precision multiplication as the multiplier generates twice the output precision. It is this RESET signal (R) that is used to provide

the truncation dynamically. There is no truncation error here. However, as shown in Fig. 4c, the determination of the sign (the first nonzero MSD) can take variable time. This is because, in contrast with conventional arithmetic (both sign-magnitude and two's complement), online arithmetic does not always give the sign in the MSD. The sign is obtained only when the first nonzero digit is computed in an MSDF manner. As soon as the sign is detected, we can insert the RESET signal(R) for all the adder stages in the pipeline. We will then have idle stages in the pipeline (shown by B for pipeline bubbles) during which the hardware could stop the clock for low power. Since the time for obtaining the MSD is variable, the time taken is almost the same as part(b), i.e., $t_{OL-MF} \propto d_{eff} * t_{OL}$, where d_{eff} is the average digit precision needed to find the sign. The area used in the detector is also the same. The advantage of this scheme over the full precision scheme is due to increased throughput ($d_{eff} \leq d$) and the power savings during idle computations. This scheme also does not have any truncation error. Fig. 4d shows a truncated online arithmetic detector with constant truncation during which ($m = 2$) MSDs were declared to be sufficient to determine the sign along with truncation error. In this case, a higher throughput can be obtained along with savings in area. $t_{OL-MF} = constant = (m+1) * t_{OL}$. Note that, in both Fig. 4c and Fig. 4d, the truncation can actually eliminate the necessity of the entire input precision and can, hence, truncate both the input and output to provide throughput speedups while retaining numerical accuracy.

3 Effect of Truncation on Online Arithmetic

Online arithmetic can have multiple representations for the same number due to the use of a redundant number system. When the first m digits out of d digits are used for truncation, the mth digit may have an error as the $(m+1)$th digit may have the opposite sign to the mth digit, thereby changing the mth digit. For example, in radix-4 online arithmetic, if the number is $1\bar{2}13\bar{1}$ and we truncate it to $1\bar{2}$, i.e., $d = 5$, $m = 2$, the Least Significant Digit (LSD) is in error as the truncated number should have been $1\bar{1}$. Note that if the $(m+1)$th digit is 0, we need to investigate further digits for the first nonzero digit to check if the mth digit is in error. For example, if the number was $1\bar{2}03\bar{1}$ truncated to $1\bar{2}$, looking just at the third digit is not sufficient to tell whether the truncated number has an error in the LSD. It is, therefore, useful to know the average truncation error in the LSD to help determine the amount of truncation that may need to be performed to meet precision requirements.

3.1 Average Truncation Error Probability in the LSD

We now quantify the average truncation error probability in the LSD, assuming all the digits are equally probable. This is a reasonable assumption since, even if the numbers are small in general, the sign bits are equally probable. If both the mth digit and $(m+1)$th digit are not equal to 0, the truncation error probability in the LSD, e, can be given by

$$e_{(m,m+1\neq 0)} = \frac{q-2}{2(q-1)}, \tag{3}$$

where $q = 2r$ and r is the radix used in the implementation ($q = 2r$ because of the additional sign digit in the maximally redundant system). There are a total of $q-1$ digits possible in a maximally redundant radix r system and there are $\frac{(q-2)}{2}$ digits of the opposite sign, excluding 0. Now, if the $(m+1)$th digit is 0, we need to look at further digits and the probability of error in the LSD assuming the mth digit to be nonzero can be shown to be

$$e_{(m\neq 0,m+1=0)} = \frac{1}{2}\left(1 - \left(\frac{1}{q-1}\right)^{d-m-1}\right) \qquad (m < d). \tag{4}$$

For the case when $m = 0$, truncation error will occur when the sign of the first nonzero digit to the left of m is different from the sign of the first nonzero digit after m. For this case, the probability of error can be shown to be

$$e_{(m=0)} = \frac{1}{4}\left(1 - \left(\frac{1}{q-1}\right)^{d-m-1}\right)\left(1 - \left(\frac{1}{q-1}\right)^{m-1}\right) \qquad (m < d). \tag{5}$$

Hence, the average probability of truncation error in the LSD assuming all digits are equally likely is given by

$$\begin{aligned} e &= \frac{1}{q-1}e_{(m=0)} + \frac{q-2}{q-1}e_{(m\neq 0)} && (m < d) \\ e &= 0 && (m = d). \end{aligned} \tag{6}$$

Equation (6) tells us the probability that the LSD is in error if all digits are equally likely. We note that the probability depends on both the number of digits d in the full precision result and the number of digits m in the truncated result and the error keeps decreasing as we increase m. The maximum error in LSD converges to 0.5 for increasing precision and using higher radix number systems. Note that truncation of the output result may actually involve truncation of the input digits also. For example, in multiplication, if $m < d/2$, both the inputs as well as the outputs are truncated as in Fig. 4d.

3.2 Effect of Truncation on Detection

Fig. 5 shows a simple example to motivate the use of truncated online arithmetic for detection. Fig. 5a shows a simple sign-based detector for the received signal, which has been transmitted in an AWGN communication channel. The probability distribution of the received signal [32] is as shown in Fig. 5b(1). The sign of the received signal determines the bit that has been transmitted. Fig. 5b(2) and Fig. 5b(3) represent the time taken by conventional and online arithmetic implementations of the detector to find the sign, assuming an 8-bit precision and radix-4 online arithmetic modules. The y-axis is normalized to the time taken to find the first digit using online arithmetic. Conventional arithmetic, being logarithmic time with precision, will have a lower throughput than the online throughput per digit (roughly shown as 1.5 in the figure for illustrative purposes). The logarithmic time assumption of conventional arithmetic assumes no internal pipelining within the conventional arithmetic blocks, which can make truncated conventional arithmetic also have a constant time. However, that would be at a significant increase in area.

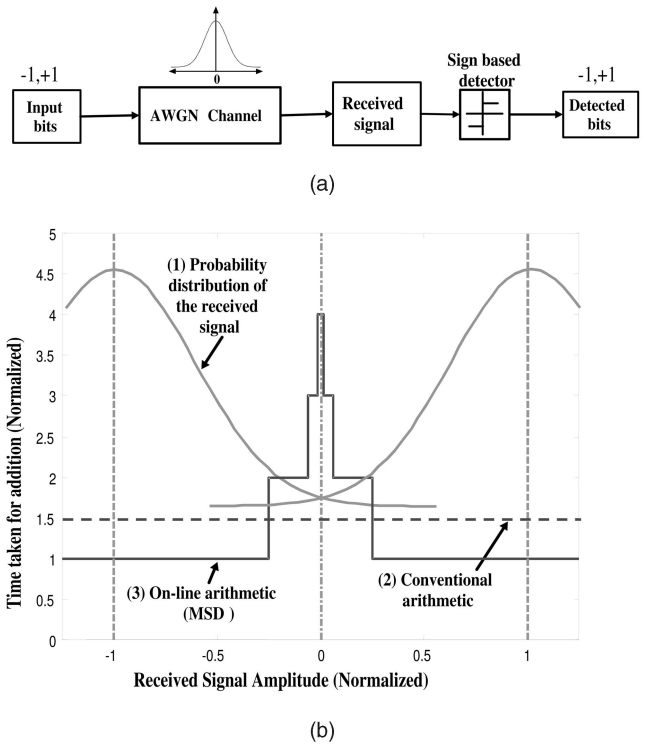

Fig. 5. Use of online arithmetic for detection. Part (a) shows a noisy communication channel through which a $+1$ or -1 is transmitted. Part (b) shows the time taken by conventional and online arithmetic to detect the sign of the received signal. To have performance benefits, the sign should be detected before the online arithmetic time exceeds the conventional arithmetic time. An 8-bit precision and a radix-4 online module are assumed.

Hence, we will retain this logarithmic time assumption in the rest of the paper, but we will also compare the associated area consumption. The figure shows that the online arithmetic scheme can take variable time to detect the sign, while the conventional arithmetic detector finds the sign in constant time (assuming constant precision for both schemes). Hence, we will have performance gains as long as the expected time to detect the sign (the first nonzero MSD) is less than that of conventional arithmetic for the same precision. In the next section, we show that we need not wait for the worst-case execution time for sign detection in communication systems as the system performance does not degrade significantly by including the error as another noise source in the communication system. From Fig. 5b, it is clear that the larger the radix used for the online detection scheme, the faster it is to determine the sign. Hence, a higher radix system is preferable for our design. However, higher radix systems have overhead in terms of area and delay. We choose a radix-4 system as a trade-off point because 1) it allows greater control on the amount of truncation, 2) it covers most of the area under the probability density curve for the first digit (0.25), and 3) it reduces the online delay, δ, to 1 for multiplication and addition [6], [33].

In detection due to variation in SNR, as seen from Fig. 5b(1), the MSD has a higher probability of being nonzero than being zero. Hence, the truncation error equation (6), which gave the effect on truncation on the

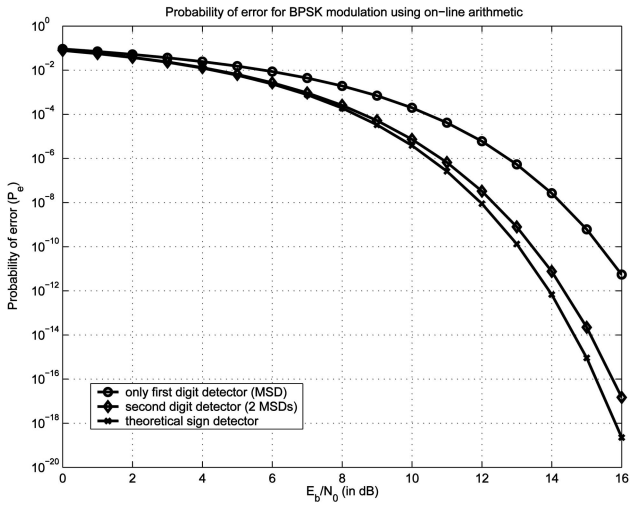

Fig. 6. Probability of error versus SNR for truncated detection. The figure shows the increase in probability of error assuming only the first digit or first two digits are used for detection in a BPSK system.

LSD for uniform probability of input digits, is not applicable in this detector example. Hence, in this case, a separate truncation error analysis needs to be performed as in the following subsection.

3.3 Truncation Error Probability for Sign Detection

The truncation error analysis for detection can be performed similar to the error analysis in (6), where all digits were equally likely. However, the truncation error analysis now depends on the probability density function as well as the signal to noise ratio (SNR) of the system. This is shown in Fig. 6. The probability of error P_{e-OPT} for an optimal detector for BPSK [32] is given by

$$P_{e-OPT} = Q\left(\sqrt{\frac{2 * E_b}{N_0}}\right), \tag{7}$$

where $Q(\cdot)$ is the Q-function [32], E_b is the energy of the transmitted bit, $N_0 = 2 * \sigma^2$ is the noise energy, and σ^2 is the variance of the noise. However, if we detect only the m MSDs, then the probability of error P_{e-OL} for the online detector for BPSK can be shown to be

$$\begin{aligned} P_{e-OL} = 0.5 * \Bigg(& Q\left(\left(1 + \frac{1}{r * m}\right) * \sqrt{\frac{2 * E_b}{N_0}}\right) \\ & + Q\left(\left(1 - \frac{1}{r * m}\right) * \sqrt{\frac{2 * E_b}{N_0}}\right)\Bigg), \end{aligned} \tag{8}$$

where r is the radix used in the online implementation. We can see from (8) that, with increasing the radix r or the number of digits to be detected m, we get increasingly closer to the optimal detector in (7). Fig. 6 shows the probability of error for a radix-4 number system for the first and second most significant digits ($r = 4$ and $m = 1, 2$) and compares it with the optimal detector. We can see around a $2\ dB$ drop in performance assuming only the MSD is used for detection. This may still be acceptable for higher throughput as there could be a decoder following this

detector to compensate for the bit error rate degradation. However, an additional digit makes the performance of the online detector close to the optimal detector. Hence, we will consider both a 1 as well as a 2 MSD truncated online detector in this paper for comparison purposes instead of a variable throughput, full precision, online detector architecture as presented in our earlier work [13]. Comparisons with conventional truncation are not shown here as it is not possible to truncate to such a low precision using conventional arithmetic without significant errors.

Fig. 6 assumes a simple transmission system without any CDMA spreading code applied on top of it ($N = 1$). The error rate curve for the CDMA matched filter detector of (2) is similar except that the Signal to Noise Ratio (SNR), E_b/N_0, is scaled by the spreading gain N (giving $10 * \log_{10}(N)$ dB better performance for the same SNR). This is assuming perfect knowledge of the channel estimates of the user. With proper channel estimation for fading channels, a closed form expression for the probability of error is no longer available, but performance similar to Fig. 6 can be expected. Thus, we have shown and quantified the trade-off between throughput speedup due to truncation and the probability of error for the code-matched filter detector.

4 Online Adders and Multipliers with Dynamic Truncation Support

This section describes the design of a radix-4 online adder and multiplier used in the detectors with dynamic truncation support. Many radix-4 online multiplier implementations in the literature assume one of the inputs as constant [12], [34], which can simplify the multiplier design. We present a design of a generalized online multiplier assuming both inputs as variable, which makes the multiplier design more complicated. Although generalized online multipliers have been presented in the literature in theory, we discuss the implementation in detail to provide area and throughput comparisons with conventional arithmetic. The multiplier is also enhanced to accept inputs in a conventional number system so that the detector can directly interface with a conventional arithmetic circuit, if needed. Theoretical estimates of area and throughput of the radix-4 adder and multiplier are determined in order to understand the area-throughput trade-offs in truncation in conventional versus online arithmetic.

4.1 Radix-4 Online Adder

The steps involved in fixed point online addition [6], [35] of two inputs, x and y, giving an output z are:

Initialization: $P_{-1} = z_{-2} = z_{-1} = 0$
Recurrence
for $j = 0, 1, \ldots, m + 1$ *do*

$$\begin{array}{rcl} W_j & = & r * P_{j-1} + r^{-\delta} * (x_j + y_j), \\ z_{j-1} & = & select(W_j), \\ P_j & = & W_j - z_{j-1}, \end{array} \quad (9)$$

where W is the residual and m is the number of digits in the inputs. The selection is done by rounding by using a discretization algorithm (DIS) [35] that performs the rounding based on the first few MSDs. The selection function is given by:

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

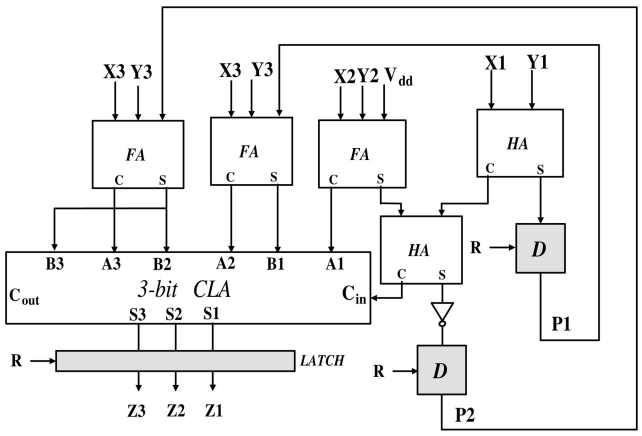

Fig. 7. A radix-4 online adder with dynamic truncation support.

Since the online adder interfaces with the output of an online multiplier and as an input to other online adders, we design the adder to be a general online adder, taking in redundant operands and producing a redundant result. Dynamic truncation is provided by using a RESET signal(R) to clear the latches, whenever necessary. The design of the online adder is shown in Fig. 7. The inputs are shown as $X[3:1]$ and $Y[3:1]$, with higher indices representing the more significant bits. The inputs are taken to be in two's complement radix-4 maximally redundant form (which is also the form in which the online multiplier produces its output). The residual is stored in Carry-Save (CS) form to avoid carry propagation delay (shown as WS and WC for residual sum and carry). The design includes two half adders (HAs), three full adders (FAs), five latches to store the residual and the output, and a 3-bit Carry Lookahead adder (CLA) for the digit selection. This circuit has approximately a throughput of eight gate delays and an area of 70 gates.

$$t_{OL-ADD} = 8, \quad (10)$$

$$A_{OL-ADD} = 70. \quad (11)$$

For comparison purposes, we have assumed the following:

Half Adder: Area = 2 gates, Delay = 1 gate
Full Adder: Area = 5 gates, Delay = 2 gates
3-bit CLA: Area = 21 gates, Delay = 5 gates
Latch: Area = 6 gates

We assume the time delay for a d-bit CLA adder with a fan-in of 4 as

$$t_{CLA}(d) = \lceil 4 * \log_4(d) \rceil + 1, \quad (12)$$

as in [29], [30], and the area in gates can be calculated as

$$A_{CLA}(d) = 4 * d + \left\lfloor \frac{14 * d}{3} \right\rfloor + g, \quad (13)$$

where g is a constant between 2 and 11 depending on $mod(d, 4)$. In the rest of the paper, wherever the time and area are dependent on the precision d, we will emphasize it by using it as an argument in the equations. This is to clearly distinguish the conventional implementations from online implementations whose area or time requirements may be independent of the bit-precision. It is important to note that the area and throughput per digit of the online adder (11), (10) are both independent of the digit precision.

4.2 Radix-4 Online Multiplier

The steps involved in online multiplication [6], [35] are:

Initialization: $P_{-1} = z_0 = X_0 = Y_0 = 0$
Recurrence *for* $j = 1, \ldots, m$ *do*

$$\begin{aligned} W_j &= r * P_{j-1} + < X_{j-1}, x_j > * y_j + Y_{j-1} * x_j, \\ Y_j &= < Y_{j-1}, y_j >, \\ z_j &= select(W_j), \\ P_j &= W_j - z_j, \end{aligned} \tag{14}$$

where the selection function is the same as in online addition.

The signed digit by vector multiplication in online multiplication [35] needs a multiplier having as one of its inputs a signed digit and as another a vector whose length keeps increasing with the number of digits (see (14)). This signed-digit by vector multiplication is difficult to implement as the multiplication time becomes dependent on the digit-precision and the advantage of precision-free online multiplication time is lost. However, in the radix-2 case, the signed digit multiplication reduces to multiplication by ± 1 and, hence, is easy to implement with lower area requirements. As we saw from Fig. 5, higher radix numbers are preferable in order to detect the sign as soon as possible, making us prefer a radix-4 number system as a better trade-off for throughput. Radix-2 multipliers [36], [37] can still be used if area is a more important constraint than throughput for the detector design.

Radix-4 online multipliers have also been designed for filter design problems and Discrete Cosine Transforms (DCT) [12], [34], where one of the inputs to the multiplier is considered constant. This is a reasonable assumption for filters as the coefficients were not time-varying and the signed digit by vector multiplication is again avoided. However, since both the inputs in the detector are variable, we need to implement a general Signed Digit by Vector Multiplier (SDVM). The SDVM needed is a two's complement multiplier of size $d \times 3$. In order to make the multiplication time precision-independent, we produce the outputs of the SDVM in CS form [12]. The multiplier outputs in CS form are then added together with the residual using a $6:2$ compressor (parallel counter) [29], [31]. We then use the MSDs in the residual for the selection function as in [12], [34] to attain online time independent of the precision.

Fig. 8 shows the implementation of a d-bit radix-4 multiplier in two's complement representation. Since the online multiplier may interface to a conventional arithmetic circuit (see Fig. 2 and Fig. 3), we design the multiplier to take in inputs in conventional arithmetic two's complement

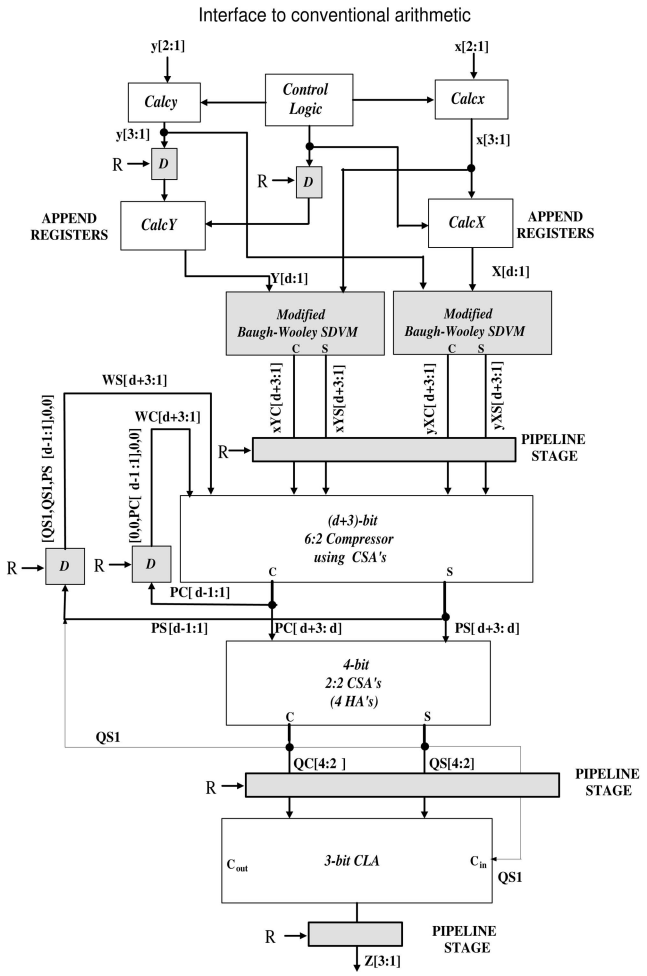

Fig. 8. A radix-4 online multiplier with conventional arithmetic inputs and dynamic truncation support.

representation. The inputs x and y shown could represent the input, r_i, from the A/D converter or the input, A_p, from the channel estimator with a parallel to serial (P/S) converter. We internally add the appropriate sign digit to the two's complement number in our circuit without any additional time overhead to make the number redundant. The $Calcx$ and $Calcy$ blocks take 2 bits from each of the inputs in the conventional arithmetic system in an MSDF fashion and convert them to a redundant number system. To do so, we note that all positive two's complement conventional numbers are valid in the redundant number system with a 0 as the sign digit for all digits. All negative two's complement numbers are valid with a 1 as the sign for the first digit and 0s for the successive digits. The $CalcX$ and $CalcY$ blocks append the digits generated to form the vector for the SDVM as in (14).

The SDVM is designed based on the modified Baugh-Wooley multiplier scheme [29], [38]. In a general $d \times d$ bit multiplication, the Baugh-Wooley method does not require any increase in the maximum column height. But, since the operands to the multiplier's input do not have the same precision ($d \times 3$), an additional stage of CSAs is needed to add the 1s at the bottom of the tree, increasing the delay of the partial product accumulation by one stage of CSAs. Since the multiplier outputs are now combined with the

residual addition, the number of inputs to be added becomes six. Hence, a $6:2$ compressor is used for addition. Again, to make the online multiplication time independent of precision, the selection function uses only the most significant 4 bits to determine the result, as shown in Fig. 8. Dynamic truncation is done by using the RESET signal (R) to clear all latches appropriately.

The need for an SDVM and a $6:2$ compressor makes the online time requirements for multiplication (17 gate delays) greater than that for addition (eight gate delays). Since the online multiplier output feeds the online adders, this will leave the adder starved for data. Hence, in order to match the delays of the adder and multiplier, we pipeline the design further as shown by the dotted lines in Fig. 8. This modified design has a delay of seven gates, equal to the delay of the compressor and the HAs. The pipeline stages increase the online delay, δ, of the multiplier from 1 to 3, resulting in higher throughput at the expense of larger latency. The complete online multiplier uses 913-2,929 gates for 8-32 bit precision. This large area requirement, even for a digit-serial multiplier, is due to the need for an SDVM and the large number of latches required to pipeline the design. If this area requirement exceeds the detector architecture specifications, then a radix-2 multiplier can be used at the expense of throughput. The estimates of area and throughput requirements for the radix-4 online multiplier are given below:

$$t_{OL-MUL} = 2*3+1 = 7, \tag{15}$$

$$A_{SDVM}(d) = 2*(d*5+d*2+3*d), \tag{16}$$

$$A_{COMPRESS-OL}(d) = 4*(d+3)*5, \tag{17}$$

$$A_{LATCHES}(d) = 5*\left(2*\left(\frac{d}{2}-1\right)+6*(d+3)+6+3\right), \tag{18}$$

$$A_{OL-MUL}(d) = A_{SDVM}(d) + A_{COMPRESS-OL}(d) + A_{LATCHES}(d) + A_{CONTROL}(d) + 4*2+21, \tag{19}$$

where t_{OL-MUL} and $A_{OL-MUL}(d)$ are the online multiplication time and area, respectively. The two SDVMs having a combined area of $A_{SDVM}(d)$ consist of d FA cells, d HA cells, and $3*d$ partial product generator gates. The $6:2$ compressor with area $A_{COMPRESS-OL}(d)$ consists of four $(d+3)$-bit FA cells. The 3-bit CLA takes 21 FA cells, the four HAs take two gates each, and $A_{LATCHES}(d)$ is the area required by the latches in order to pipeline the design sufficiently enough to reduce the online delay to 7. $A_{CONTROL}(d)$ is the area required for the additional control logic required. It is important to note that, while the area of the online multiplier is dependent on the precision (19), the throughput per digit of the online multiplier (15) is precision-independent.

5 Conventional and Online Truncated Implementations of a Matched Filter Detector

We now compare the effects of truncation using different implementation schemes for a code matched filter detector. We discuss the trade-offs in area and delay due to truncation between a simple conventional arithmetic implementation, a conventional arithmetic implementation using carry-save adders, and an online arithmetic implementation. These are shown in Fig. 9. Both the conventional and online arithmetic schemes assume no pipelining between the adders and multipliers and the conventional truncated arithmetic implementations assume no internal pipelining. However, the area comparisons are also made to put things in perspective.

5.1 Truncated Conventional Arithmetic Matched Filter

A straightforward implementation of a truncated matched filter [24] is shown in Fig. 9a. The dot product is implemented in a tree structure for a parallel implementation. The multipliers form the first level of the tree and the adders form the rest of the levels. For a length-N dot product, there are N multipliers at the top of the tree, $N/2$ adders at the next level, $N/4$ adders at the next level, and so on. If the multipliers are implemented using Dadda tree multipliers [29], [39], the delay of the multiplier depends on the precision d.

This detector is used as a base case comparison point with other detector structures based on conventional and online arithmetic.

5.2 Truncated Conventional Arithmetic Matched Filter Using Carry-Save Adders

Further reductions in delay can be obtained by reducing the carry-propagation delay overhead using CSAs. Similar schemes using CSAs for dot-product type computations have been proposed in [2], [40], [41]. We explore this architecture as another trade-off point for comparison purposes with a time-constrained online arithmetic architecture. The Dadda multiplier no longer does the CLA at the bottom of the multiplier tree, but, instead, passes on its output in a CS form, as in Fig. 9b. The CS outputs of all the N multipliers are then added together with a $2*N:2$ compressor. The CS output of this compressor is then fed to a wider CLA for computing the final result. The time taken for the matched filter using carry-save adders is the time taken for the multiplier without the CLA, the time taken by the compressor, and the time for the final CLA. The area and throughput estimates of the truncated matched filter using CSAs are compared with the other schemes in the next section.

5.3 Truncated Online Arithmetic Matched Filter

In contrast to the conventional arithmetic implementations, the throughput of the online arithmetic-based matched filter detector (shown in Fig. 9c) depends only on the number of most significant digits m needed and is independent of the total precision of the input and the number of stages in the adder-multiplier chain as it is highly pipelined. The area

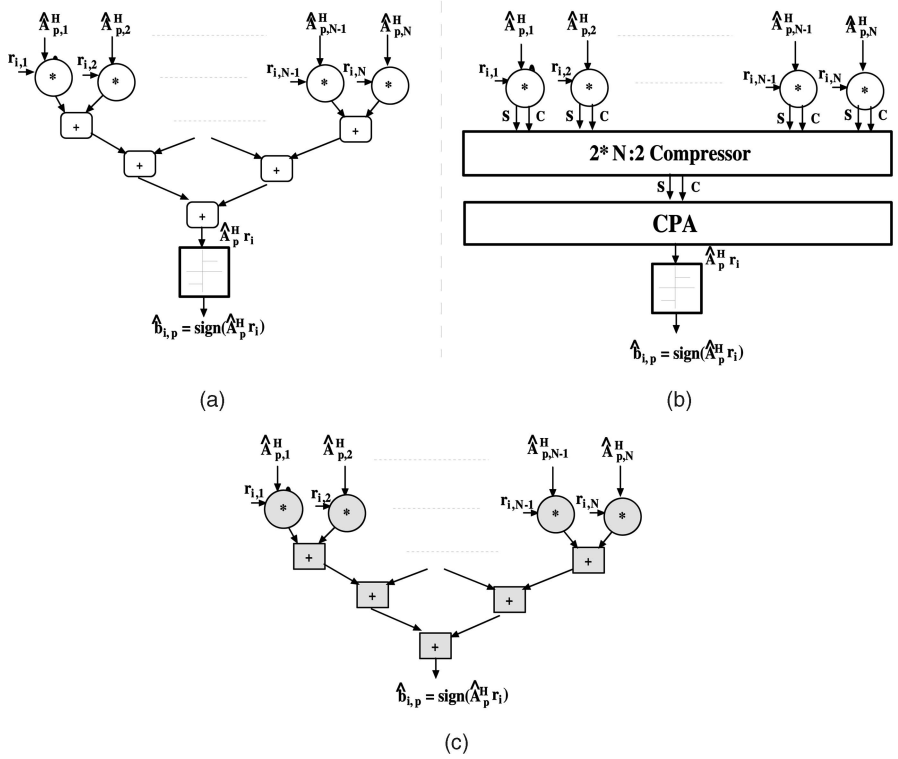

Fig. 9. Code matched filter with different arithmetic schemes. Part (a) shows a simple implementation using truncated conventional arithmetic. Part (b) shows a truncated conventional implementation using CSAs. Part (c) shows a truncated online arithmetic implementation.

and throughput comparisons of the truncated online arithmetic-based matched filter with the conventional arithmetic schemes are now presented in the section below. Note that the truncated arithmetic matched filter assumes full precision intermediate calculations and truncation to 1 bit output. Even if intermediate calculations were not done at full precision for the truncated conventional arithmetic multiplier, the throughput estimates do not change for the truncated multiplier since the critical path does not change. The area estimates might be lowered, depending on the amount of truncation needed to reliably get the sign.

5.4 Area-Time Trade-Offs and Results

Fig. 10 presents the theoretical throughput comparisons of the conventional arithmetic matched filter detector, the conventional arithmetic matched filter detector using carry-save adders, and a radix-4 online arithmetic-based matched filter detector. The x-axis represents the input precision in bits. The online arithmetic numbers use digits instead of bits and, since we use a radix-4 system, each digit in online arithmetic is equal to 2 bits in the conventional arithmetic system. Hence, the truncated MSD detector starts at an input precision of 2 bits, while the truncated 2 MSD detector, which needs at least 4 input bits to produce 2 MSDs as output, starts at an input precision of 4 bits. We can see that both the conventional arithmetic detectors have logarithmic increase in gate delays with precision, while the online detector has a linear increase in gate delay and is, hence, slower for a full precision implementation. The truncated detector schemes with 1 and 2 MSDs, however, perform much better as they have a constant throughput independent of the digit precision (see (10), (15)). The conventional arithmetic implementation using carry save adders also outperforms the simple conventional arithmetic implementation by limiting the carry propagation delay.

Fig. 11 presents the corresponding theoretical area comparisons for the detectors. It can be observed from the figure that the truncated conventional arithmetic matched filter detector using carry-save adders has better time performance than a straightforward implementation with nearly the same area. Full precision online arithmetic, on the other hand, consumes more area than the conventional schemes for low precision due to the use of a redundant

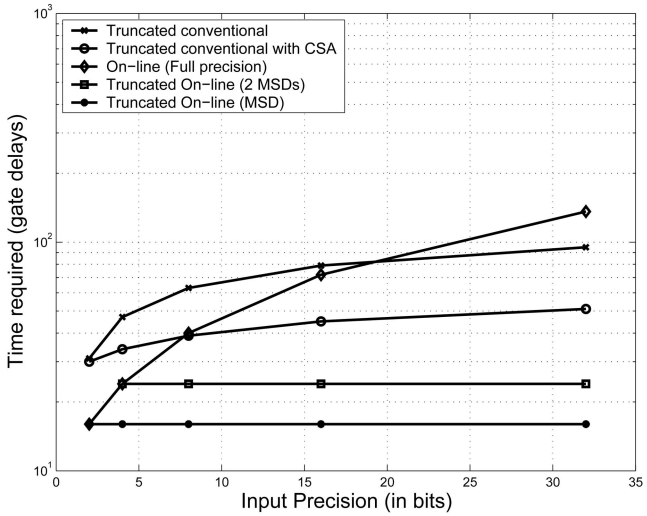

Fig. 10. Theoretical throughput estimates for the truncated arithmetic detectors in terms of gate delays.

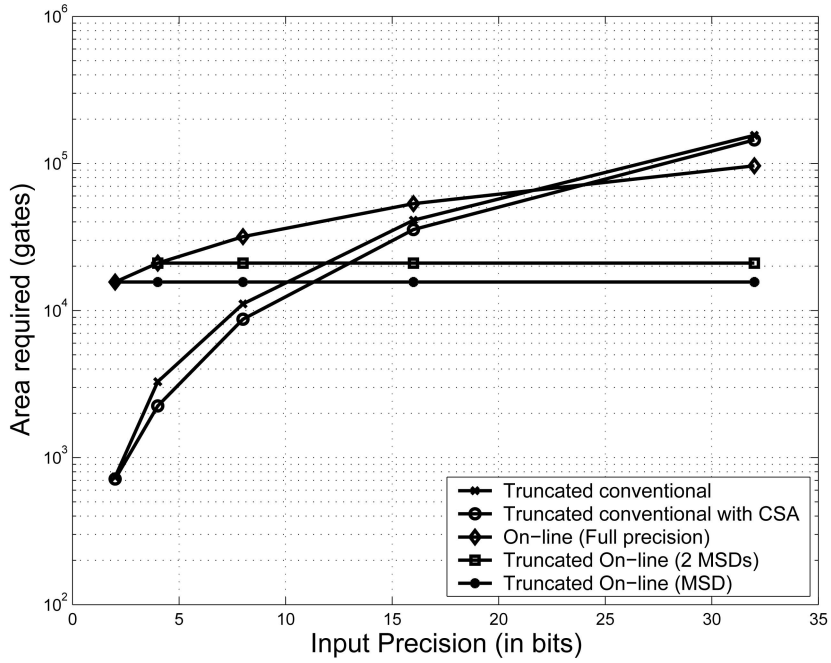

Fig. 11. Theoretical area estimates for truncated arithmetic detectors in terms of logic gates.

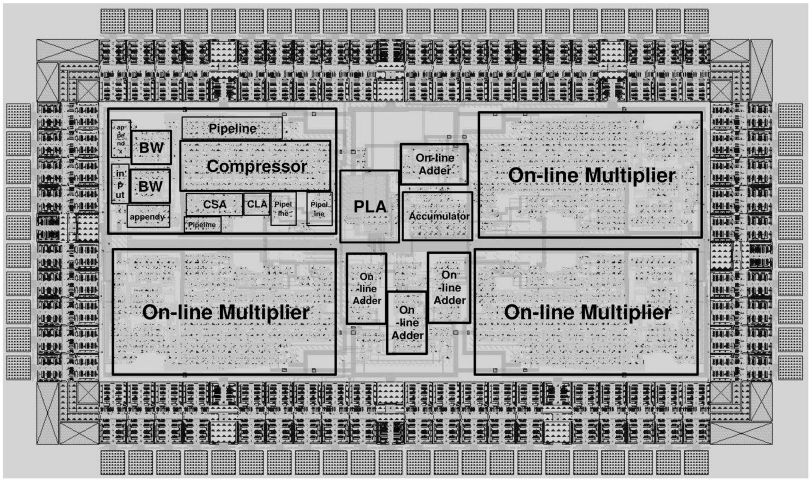

Fig. 12. VLSI prototype implementation of code matched filter using truncated online arithmetic.

number system, but soon starts giving benefits in area for higher precision (≥ 16 bits) due to its digit-serial nature. Though the area required by the online adders is constant with precision (11), the online multiplier requires more area with increasing precision (19). The truncated online arithmetic schemes again give additional gains in area as the multiplier area also becomes constant if the minimum number of truncation digits is known. Thus, the two MSD detector area estimates assume that the detector can provide both 1 and 2 MSDs, while the 1 MSD detector area estimates assume that the detector can only provide the MSD.

6 VLSI Implementation

VLSI implementations of the online and conventional arithmetic design were carried out to test the theoretical estimate curves and to test and verify the design.

A prototype implementation of the radix-4 online code matched filter chip with dynamic truncation was done at Rice by [42] as part of the VLSI design course using a MOSIS Tiny Chip 0.5 μ AMI process. The online arithmetic chip implementation is shown in Fig. 12. The chip consists of four online multipliers and adders and accepts only up to two MSDs as inputs and provides dynamic truncation for 1-4 digits of the output. Each chip can do a dot product of length 4 and several chips can be connected in parallel for processing higher spreading code lengths. The throughput estimates of the design are valid for all spreading code lengths due to pipelining of the adders and multipliers. The first online multiplier is expanded to show its subcomponents. BW represents the Baugh-Wooley multiplier. Another VLSI implementation [43] uses conventional arithmetic and truncated multipliers with constant correction as a proof-of-concept comparison. Though the online

TABLE 1
Area-Time-Power Estimates of the Radix-4 Online Arithmetic Code-Matched Filter Detector

Blocks	Delay (in ns)	Area (Transistors)	Power at 5V (in mW)
Half-adder	0.80	18	0.0978
Full-adder	2.04	38	0.1369
3-bit CLA	3.32	120	0.3497
Latch	-	18	0.1066
PLA	-	985	10.9040
On-line adder	7.17	394	0.8765
On-line Multiplier (non-pipelined)	16.42	2062	6.3001
On-line Multiplier (pipelined)	12.33	2962	10.4609
Total (without padframe)	12.33	15229	45.4331
Total (with padframe)	12.33	19917	-

arithmetic circuits were substantially larger than the truncated conventional arithmetic circuits in area for the chosen 8-bit output precision (which seemed to agree with the theoretical estimates), exploiting parallelism for conventional arithmetic was more difficult due to I/O pin limitations. Since truncation using conventional arithmetic has been well-studied [1], we only provide implementation details for the truncated online arithmetic code matched filter implementation below. Further information and comparisons can also be obtained from the Rice VLSI signal processing research page [44].

Table 1 shows the area, time, and power consumptions of the basic blocks used in the detector implementation. Note that the numbers are from a prototype implementation and should only be used as a relative comparison between the different blocks in the detector design. The chip die size was $2.2 \times 4.4mm^2$ and used around 15,000 transistors to implement four radix-4 truncated online multipliers and four radix-4 online adders. The delay and power estimates were obtained by a Spice analysis for a 0.5 μ AMI process.

7 Conclusions

We have presented truncated online arithmetic with applications to communication systems. We show that online arithmetic can support dynamic truncation while simultaneously providing area and throughput benefits. The benefits of online arithmetic are dependent on the amount of truncation; the larger the difference between the full precision and the truncated result, the greater the benefits of online arithmetic are. In general, if the output precision is greater than 8-16 bits and significant truncation of the result is required, online arithmetic should be used for both area and throughput benefits. We also show the effect of truncation error on the least significant digit of the truncated result for online arithmetic. In contrast, conventional arithmetic systems do not support dynamic truncation, have minor throughput benefits, and can have significant errors in the truncated results. Radix-4 online arithmetic adders and multipliers were built in VLSI and they provided support for dynamic truncation and the ability to interface with other conventional arithmetic circuits. Thus, communication systems with conventional arithmetic for saturation and truncated online arithmetic for truncation can coexist in an optimized design for throughput and power benefits.

Acknowledgments

The authors are grateful to the students at Rice University who helped in implementing and testing the truncated conventional and truncated online arithmetic designs—Predrag Radosavljevic, Manik Gadhiok, Nils Bagge, Noah Deneau, Chad Cook, Richa Dubey, and Amy Lin. This work was presented in part at the IEEE International Symposium on Computer Arithmetic in Vail, Colorado, in 2001. This work was supported by Nokia, Texas Instruments, and by the US National Science Foundation under grants ANI-9979465 and EIA-0224458.

References

[1] M.J. Schulte and E.E. Swartzlander, "Truncated Multiplication with Correction Constant," *Proc. Workshop VLSI Signal Processing*, vol. VI, pp. 388-396, Oct. 1993.

[2] P.I. Balzola, M.J. Schulte, J. Ruan, J. Glossner, and E. Hokenek, "Design Alternatives for Parallel Saturating Multioperand Adders," *Proc. IEEE Int'l Conf. Computer Design (ICCD)*, pp. 172-177, 2001.

[3] M.J. Schulte, J.E. Stine, and J.G. Jansen, "Reduced Power Dissipation through Truncated Multiplication," *Proc. IEEE Alessandro Volta Memorial Workshop Low Power Design*, pp. 61-69, Mar. 1999

[4] Z. Huang and M.D. Ercegovac, "Two-Dimensional Signal Gating for Low-Power Array Multiplier Design," *Proc. IEEE Int'l Conf. Circuits and Systems*, vol. 1, pp. 489-492, May 2002.

[5] K.E. Wires, M.J. Schulte, and J.E. Stine, "Combined IEEE Compliant and Truncated Floating Point Multipliers for Reduced Power Dissipation," *Proc. IEEE Int'l Conf. Computer Design (ICCD)*, pp. 497-500, Sept. 2001.

[6] M.D. Ercegovac, "Online Arithmetic: An Overview," *Proc. Real Time Signal Processing VII, SPIE*, pp. 86-93, Aug. 1984.

[7] E.G. Walters and M.J. Schulte, "Design Tradeoffs Using Truncated Multipliers in FIR Filter Implementations," *Proc. SPIE: Advanced Signal Processing Algorithms, Architectures, and Implementations*, July 2002.

[8] S. Oraintara, Y.J. Chen, and T.Q. Nguyen, "Integer Fast Fourier Transform," *IEEE Trans. Signal Processing*, vol. 50, no. 3, pp. 607-618, Mar. 2002.

[9] J. Markel and A. Gray Jr., "Fixed-Point Truncation Arithmetic Implementation of a Linear Prediction Autocorrelation Vocoder," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 22, no. 4, pp. 273-282, Aug. 1974.

[10] 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," *IEEE Trans. Wireless Comm.*, vol. 1, no. 3, pp. 468-479, July 2002.

[11] M.D. Ercegovac and T. Lang, "Online Arithmetic: A Design Methodology and Applications in Digital Signal Processing," *Proc. VLSI Signal Processing III*, pp. 252-263, Nov. 1988.

[12] J. Bruguera and T. Lang, "2-D DCT Using Online Arithmetic," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 5, pp. 3275-3278, May 1995.

[13] S. Rajagopal and J.R. Cavallaro, "Online Arithmetic for Detection in Digital Communication Receivers," *Proc. 15th IEEE Int'l Symp. Computer Arithmetic (ARITH-15)*, pp. 257-265, June 2001.

[14] T. Lynch and M.J. Schulte, "A High Radix Online Arithmetic for Credible and Accurate Computing," *J. Universal Computer Science*, vol. 1, no. 7, pp. 439-453, July 1995.

[15] R. McIlhenny and M.D. Ercegovac, "Online Algorithms for Complex Number Arithmetic," *32nd Asilomar Conf. Signals, Systems, and Computers*, pp. 172-176, Oct. 1998.

[16] N.D. Hemkumar and J.R. Cavallaro, "Redundant and Online CORDIC for Unitary Transformations," *IEEE Trans. Computers,* special issue on computer arithmetic, vol. 43, no. 8, pp. 941-954, Aug. 1994.
[17] M.D. Ercegovac and A.L. Grnarov, "On the Performance of Online Arithmetic," *Proc. Int'l Conf. Parallel Processing,* pp. 55-62, Aug. 1980.
[18] M.D. Ercegovac and T. Lang, "On-the-Fly Conversion of Redundant into Conventional Representations," *IEEE Trans. Computers,* vol. 36, no. 7, pp. 895-897, July 1987.
[19] A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Trans. Electronic Computers,* vol. 10, no. 3, pp. 389-400, Sept. 1961.
[20] S. Moshavi, "Multi-User Detection for DS-CDMA Communications," *IEEE Comm. Magazine,* pp. 124-136, Oct. 1996.
[21] S. Verdú, *Multiuser Detection.* Cambridge Univ. Press, 1998.
[22] M. Honig, U. Madhow, and S. Verdú, "Blind Adaptive Multiuser Detection," *IEEE Trans. Information Theory,* vol. 41, no. 4, pp. 944-960, July 1995.
[23] J. Chen, G. Shou, and C. Zhou, "High-Speed Low-Power Complex Matched Filter for W-CDMA: Algorithm and VLSI Architecture," *IEICE Trans. Fundamentals,* vol. E83-A, no. 1, pp. 150-157, Jan. 2000.
[24] K. Chapman, P. Hardy, A. Miller, and M. George, "CDMA Matched Filter Implementation in Virtex Devices," Xilinx Application Note XAPP212(v1. 1), Jan. 2001.
[25] T. Long and N.R. Shanbhag, "Low-Power CDMA Multiuser Receiver Architectures," *Proc. IEEE Workshop Signal Processing Systems,* pp. 493-502, Oct. 1999.
[26] R.H. Walden, "Performance Trends in Analog-to-Digital Converters," *IEEE Comm. Magazine,* vol. 37, no. 2, pp. 96-101, Feb. 1999.
[27] I. Seskar and N.B. Mandayam, "A Software Radio Architecture for Linear Multiuser Detection," *IEEE J. Selected Areas in Comm.,* vol. 17, no. 5, pp. 814-823, May 1999.
[28] N. Zhang, A. Poon, D. Tse, R. Brodersen, and S. Verdú, "Trade-Offs of Performance and Single Chip Implementation of Indoor Wireless Multi-Access Receivers," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC),* vol. 1, pp. 226-230, Sept. 1999.
[29] B. Parhami, *Computer Arithmetic—Algorithms and Hardware Designs.* Oxford Univ. Press, 2000.
[30] S. Waser and M.J. Flynn, *Introduction to Arithmetic for Digital System Designers.* CBS College Publishing, 1982.
[31] I. Koren, *Computer Arithmetic Algorithms.* Prentice Hall, 1993.
[32] J.G. Proakis and M. Salehi, *Communication Systems Engineering.* Prentice Hall, 1994.
[33] A. Gorji-Sinaki and M.D. Ercegovac, "Design of a Digit-Slice Online Arithmetic Unit," *Proc. Fifth IEEE Symp. Computer Arithmetic,* pp. 72-80, May 1981.
[34] J.S. Fernando and M.D. Ercegovac, "Conventional and Online Arithmetic Designs for High-Speed Recursive Digital Filters," *Proc. Fifth IEEE Workshop VLSI Signal Processing,* pp. 81-90, Oct. 1992.
[35] M.D. Ercegovac, "A General Hardware-Oriented Method for Evaluation of Functions and Computations in a Digital Computer," *IEEE Trans. Computers,* vol. 26, no. 7, pp. 667-680, July 1977.
[36] A. Guyot, Y. Herreros, and J-M. Muller, "JANUS: An Online Multiplier/Divider for Manipulating Large Numbers," *Proc. Ninth IEEE Symp. Computer Arithmetic,* pp. 106-111, Sept. 1989.
[37] A.F. Tenca, M.D. Ercegovac, and M.E. Louie, "Fast Online Multiplication Units Using LSA Organization," *Proc. Advanced Signal Processing Algorithms, Architectures, and Implementations IX, SPIE,* pp. 74-83, July 1999.
[38] C.R. Baugh and B.A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm," *IEEE Trans. Computers,* vol. 22, no. 12, pp. 1045-1047, Dec. 1973.
[39] K'A.C. Bickerstaff, E.E. Swartzlander, and M.J. Schulte, "Analysis of Column Compression Multipliers," *Proc. IEEE Int'l Symp. Computer Arithmetic,* pp. 33-39, June 2001.
[40] G. Wang and M. Tull, "The Implementation of an Efficient and High-Speed Inner-Product Processor," *Proc. 35th Asilomar Conf. Signals, Systems, and Computers,* vol. 2, pp. 1362-1366, Oct. 2001.
[41] N. Azakova, R. Sung, N. Durdle, M. Margala, and J. Lamoureux, "Fast and Low-Power Inner Product Processor," *Proc. 2001 IEEE Int'l Symp. Circuits and Systems (ISCAS),* vol. 4, pp. 646-649, May 2001.
[42] P. Radosavljevic, M. Gadhiok, and N. Bagge, "Online Arithmetic Matched Filter Detector," Dec. 2002.
[43] C. Cook, N. Deneau, R. Dubey, and A. Lin, "Conventional Arithmetic Matched Filter Detector," Dec. 2002.
[44] "VLSI Signal Processing Research at Rice University," http://www.ece.rice.edu/~cavallar/vlsi, 2006.

Sridhar Rajagopal (S'98, M'04) received the MS and PhD degrees in electrical and computer engineering from Rice University, Houston, Texas, in 2000 and 2004, respectively, and the BE degree (Hons. with Distinction) in electronics engineering from VJTI, Mumbai University, India, in 1998. He is currently a member of the technical staff at WiQuest Communications, working on Ultra Wide Band (UWB) communication systems. His research interests are in wireless communications, VLSI signal processing, computer arithmetic, and computer architecture. He is a member of the IEEE.

Joseph R. Cavallaro (S'78, M'82, SM'05) received the BS degree from the University of Pennsylvania, Philadelphia, in 1981, the MS degree from Princeton University, Princeton, New Jersey, in 1982, and the PhD degree from Cornell University, Ithaca, New York, in 1988, all in electrical engineering. From 1981 to 1983, he was with AT&T Bell Laboratories, Holmdel, New Jersey. In 1988, he joined the faculty of Rice University, Houston, Texas, where he is currently a professor of electrical and computer engineering. His research interests include computer arithmetic, VLSI design and microlithography, and DSP and VLSI architectures for applications in wireless communications. During the 1996-1997 academic year, he served at the US National Science Foundation (NSF) as director of the Prototyping Tools and Methodology Program in the Computer (CISE) Directorate. During 2005, he was a Nokia Foundation Fellow and a visiting professor with the Centre for Wireless Communications at the University of Oulu, Finland. He is currently the associate director of the Center for Multimedia Communication at Rice University. Dr. Cavallaro is a recipient of the NSF Research Initiation Award and is a member of Tau Beta Pi, Eta Kappa Nu, ACM, and a senior member of the IEEE. He is an IEEE Computer Society Distinguished Lecturer, 2004-2006, and was cochair of the 2004 Signal Processing for Communications Symposium at the IEEE Global Communications Conference and general cochair of the 2004 IEEE 15th International Conference on Application-specific Systems, Architectures and Processors (ASAP).