Design of an On-Line Multiply-Add Module for Recursive Digital Filters *

R. H. Brackert Jr., M. D. Ercegovac+, and A. N. Willson Jr.

Electrical Engineering Department
+ Computer Science Department

School of Engineering and Applied Science
University of California, Los Angeles
Los Angeles, CA 90024

Abstract

This paper describes an on-line multiply-add module that enables high filter sampling rates when used to implement the direct form II second-order filter structure [1], [2]. Important characteristics of on-line arithmetic [3] are that it produces results most significant digit first, and that its digit cycle time is independent of the data wordlength. These features not only enable high-speed filtering, but also allow the elimination of all nonlinear oscillations in the filter without affecting the sampling rate, and effectively eliminate scaling of the filter's input data. Presented is the derivation of the on-line multiply-add algorithm, and its hardware design using a 1.5μ CMOS standard cell library. Also described is a method for eliminating nonlinear oscillations by increasing the filter's working precision.

1 Introduction

Design of fast modules for fixed-point recursive digital filtering is strongly affected by two factors. First, the latency between successive recurrence evaluations limits the sampling rate [4]. Various techniques have been suggested to reduce this latency by exploiting various forms of parallelism and pipelining in the context of conventional arithmetic algorithms [5], [6], [7], [8]. We believe these methods have either slower sampling rates than our method, or can be used to supplement our method to provide even higher sampling rates. It should be noted that these schemes may have disadvantages (some quite severe) that generally have not been fully analyzed or, in some cases, not even addressed.

The second fundamental problem in the implementation of recursive filters is the handling of nonlinear oscillations without affecting the maximum sampling rate of the filter [9]. We choose to implement the direct form II second-order filter structure [10] since its sampling period can be as small as the time required for one evaluation of the multiply-add (MA) function $AX + Y$. It seems evident, therefore, that the sampling period of this direct form II structure is smaller than that of any other general purpose IIR filter structure. Comparisons with other IIR structures can be found in [9]. However, the performance of a filter implemented with this structure can be severely affected by nonlinear oscillations. Conventional methods for reducing or eliminating nonlinear oscillations [11], [12], are not easily converted, or even impossible to convert, to an on-line implementation. These conventional methods require additional hardware to be placed in the recursive loops of the filter, therefore, reducing the sampling rate. We propose to increase the internal wordlength of the filter such that the effects of the nonlinear oscillations are removed from the output of the filter. Since on-line arithmetic enables our design to have a sampling rate independent of the data wordlengths, our method for handling nonlinear oscillations will not affect the sampling rate. The penalty incurred by using our method is a substantial increase in hardware. However, the additional hardware is essentially a replication of the on-line MA module. Another advantage of this method is that the output of the filter is within a quantization error of the ideal output (using the same limited precision coefficients). Thus, appropriate scaling does not reduce the signal-to-noise ratio.

In this paper we present an integrated design approach based on on-line arithmetic which offers a solution to both the recurrence latency and the nonlinear oscillations problems, while maintaining a high signal-to-noise ratio. In Section 2 a brief discussion of the on-line approach and the choice of key parameters are given. The radix-4 on-line algorithm for the multiply-add operation is described in Section 3. In Section 4 we discuss its VLSI design based on standard cells and we comment on the performance of the implementation. In Section 5 we discuss the problem of nonlinear oscillations and the proposed solution. We conclude with a summary of features of the proposed design and review the results of our standard cell 1.5μ CMOS design. Using the on-line MA modules to implement the filter is fully described in [1] and [2].

2 The Basis for the Algorithm

The following derivation leads to an algorithm that performs the multiply-add function $AX+Y$ using on-line arithmetic with internal pipelining. The inputs X and Y are received by the on-line module digitwise (most significant digit first) and the coefficient A is completely specified in advance. The digits of X and Y are in signed digit format [13]. The derivation of the algorithm follows [3], [14], and [15].

* This work was supported in part by the National Science Foundation under Grants MIP8603639 and MIP8813340.

Let the full precision scaled result be

$$S^* = r^{-\delta}(AX+Y) \tag{1}$$

where r is the radix and δ is the on-line delay to be determined later. We denote

$$X = \sum_{j=0}^{d+\delta-1} x_j r^{-j} \qquad x_j \in D_\rho = \{-\rho, \ldots, \rho\}$$

$$Y = \sum_{j=0}^{d+\delta-1} y_j r^{-j} \qquad y_j \in D_\rho = \{-\rho, \ldots, \rho\}$$

$$S = \sum_{j=0}^{d+\delta-1} s_j r^{-j} \qquad s_j \in D_\rho = \{-\rho, \ldots, \rho\} \tag{2}$$

and

$$X[j] = X[j-1] + x_j r^{-j}\,, \quad X[-1] = 0$$

$$Y[j] = Y[j-1] + y_j r^{-j}\,, \quad Y[-1] = 0$$

$$S[j] = S[j-1] + s_j r^{-j}\,, \quad S[-1] = 0 \tag{3}$$

where $x_j = y_j = 0$ for $j \geq d$ and $\frac{1}{2}r \leq \rho < r$. Thus, S^*, the full precision scaled result, differs from S, the first $d+\delta-1$ digits of the scaled sum, by some error. This error is defined by the equation

$$S^* = S + E_S r^{-(d+\delta-1)}\,. \tag{4}$$

The precise value of E_S will be discussed later.

To define a recurrence relation we express s_j in terms of $S[j-1]$, $X[j]$, and $Y[j]$. At the j-th iteration, noting (1), the scaled true sum is $r^{-\delta}(AX[j]+Y[j])$ and the partial scaled sum found thus far is $S[j-1]$. We define the scaled residual

$$W[j] = \left[r^{-\delta}(AX[j]+Y[j]) - S[j-1]\right]r^{j} \tag{5}$$

where $W[j]$ differs from s_j by some remainder. We define a selection function

$$s_j = Sel(W[j]) \tag{6}$$

such that $Sel(\cdot)$ maps $W[j]$ to an integer in the set $D_\rho = \{-\rho, \ldots, \rho\}$, and such that

$$|W[j] - s_j| \leq \zeta \tag{7}$$

where $\frac{1}{2} \leq \zeta < 1$. Thus, ζ is a measure of the maximum remainder caused by the chosen selection function. For example, $\zeta = \frac{1}{2}$ corresponds to a rounding selection function.

We get the residual recurrence equation by substituting (3) into (5)

$$W[j] = r(W[j-1]-s_{j-1}) + (Ax_j+y_j)r^{-\delta}\,. \tag{8}$$

Recall that S is the scaled result. Hence, the desired result $Q = AX+Y$ has digits q_j that are related to s_j by $q_{j-\delta} = s_j$. Fig. 1 shows a block diagram of the recurrence step.

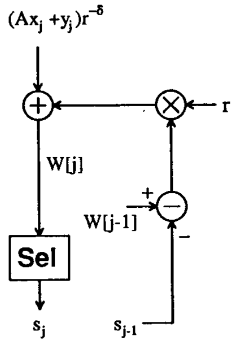

Fig. 1. A simple block diagram of the recurrence step of the on-line MA algorithm.

The On-Line Delay

The most significant digit (MSD) of the output of the on-line MA appears after $\delta+1$ inputs, where δ is the on-line delay. Since $|W[j]-s_j| \leq \zeta$ and $|W[j]| \leq \rho + \zeta$ (from (7)), and noting that $|x_j| \leq \rho$ and $|y_j| \leq \rho$, from (8)

$$\rho + \zeta \geq r\zeta + \rho(A_{max}+1)r^{-\delta} \tag{9}$$

where $A_{max} = \max(|A|)$. Since δ is an integer, solving (9) for δ yields

$$\delta \geq \left\lceil \frac{\log\left[\dfrac{\rho(A_{max}+1)}{\rho+\zeta-r\zeta}\right]}{\log(r)} \right\rceil. \tag{10}$$

Recalling that $\zeta = \frac{1}{2}$ corresponds to a rounding selection function we finally obtain

$$\delta \geq \left\lceil \frac{\log\left[\dfrac{2\rho(A_{max}+1)}{2\rho+1-r}\right]}{\log(r)} \right\rceil. \tag{11}$$

Fig. 2 illustrates δ_{min} (the minimum value of δ in (11)) versus A_{max} for several given r and ρ values, when a rounding selection function $\zeta = \frac{1}{2}$ is used. (In [1], we show that (11) yields the same result as (10) for any valid ζ.)

The Specifications for the On-Line MA

To design the on-line MA, we must choose the radix r, the digit set D_ρ, and A_{max} the maximum value of $|A|$. Choosing $A_{max}=2$ is necessary when designing a general purpose digital filter implemented using the direct form II second-order structure. The best values for r and D_ρ are not as obvious. Since a carry-save adder will be used inside the MA, the radix should be of the form 2^k, where k is an integer, thus simplifying multiplication of a two's complement number by r (which is required by the algorithm). Furthermore, since A must be multiplied by the digit x_j during every clock cycle, it would be an advantage to limit the digit set to be a subset of

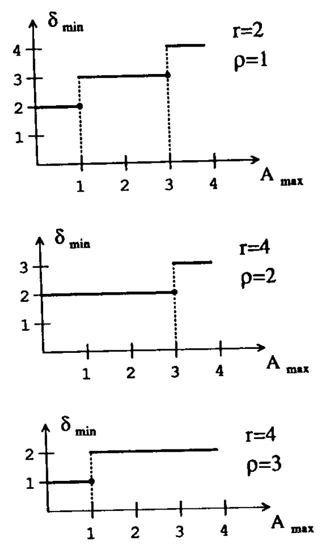

Fig. 2. Examples of δ_{min} versus A_{max} for various r and ρ values, using a rounding selection function.

$\{-2,-1,0,1,2\}$, thus making Ax_j trivial to compute. This restricts r to be either 2 or 4. Choosing r=4 is advantageous since a given number requires twice as many digits for its radix 2 representation versus radix 4. Thus, we have chosen r=4 and ρ=2 for the implementation. From (11), for r=4, ρ=2, and A_{max}=2

$$\delta \geq 2 . \tag{12}$$

Thus, we choose $\delta = 2$.

3 Radix-4 On-Line Multiply-Add Algorithm

The step time of the algorithm depends on the evaluation of the residual $W[j]$, the selection function, and their mutual dependency. A redundant addition coupled with a limited-precision selection function is used to make the recurrence step independent of the working precision. Further speed enhancement comes from internal pipelining of the recurrence evaluation and the selection function [16]. We now discuss these aspects in more detail.

The residual $W[j]$ is produced by a carry-save adder (CSA) because of the inherent simplicity of the CSA design. Therefore, $W[j] = WPS[j] + WSC[j]$, where $WPS[j]$ and $WSC[j]$ are the pseudosum and stored carry of the carry-save adder, respectively. The selection function is defined such that $Sel : W \rightarrow D_\rho$ where the domain

$$W = \{W[j] \mid W[j] \in I = [-\rho-\zeta , \rho+\zeta]\} \tag{13}$$

is the finite set of values of $W[j]$, defined by the recursion formula (8), and the range D_ρ is the redundant digit set $-\rho$ to ρ. Also, $\frac{1}{2} \leq \zeta < 1$. We let

$$I_k = [\, l_k \, , u_k \,] \, , \; k \in D_\rho \tag{14}$$

be a subinterval of I, with lower and upper endpoints l_k and u_k, respectively, such that if $W[j] \in I_k$ then $s_j = Sel(W[j]) = k$ is a valid digit choice. The overlap between adjacent subintervals is a constant and we define it as

$$\Delta = \Delta_i = u_i - l_{i+1} \quad \text{for all} \quad i \in D_\rho - \{\rho\} \tag{15}$$

and it must satisfy $\Delta \geq -r^{-m}$ for the domain W to be continuous, assuming m fractional bits for the representation of $W[j]$. In [1], we use (8) to show that

$$\Delta = \frac{2\rho + 1 - r - 2\rho(A_{max}+1)r^{-\delta}}{r - 1} \tag{16}$$

$$l_k = k - \tfrac{1}{2} - \tfrac{1}{2}\Delta \tag{17}$$

$$u_k = k + \tfrac{1}{2} + \tfrac{1}{2}\Delta \, . \tag{18}$$

Thus, for the chosen specifications, $\Delta = \frac{1}{12}$. To provide a selection function that is independent of the precision of $WPS[j]$ and $WSC[j]$, we let a limited number of the most significant bits of $WPS[j]$ and $WSC[j]$, denoted by $\hat{WPS}[j]$ and $\hat{WSC}[j]$, respectively, be used to determine s_j such that $0 \leq W[j] - \hat{W}[j] < \varepsilon$ (which is a one-sided error) where $\hat{W}[j] = \hat{WPS}[j] + \hat{WSC}[j]$, and the precision of $\hat{W}[j]$ is such that it has β fractional bits. We select ε and β such that if $Sel(\hat{W}[j]) = k$, then $W[j] \in I_k$ and, hence, the recursion equation will still be valid. Fig. 3 depicts a typical subinterval overlap for $\Delta = \frac{1}{12}$, and the potential values of $\hat{W}[j]$ when $\beta = 5$.

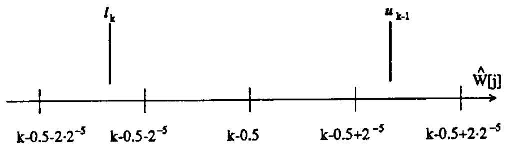

Fig. 3. An example of a subinterval overlap.

To achieve a selection function independent of the wordlength of $W[j]$, we let $\varepsilon = 3 \cdot 2^{-5}$ and the selection function

$$Sel(\hat{W}[j]) = \begin{cases} k & \text{if } k - \frac{1}{2} - 2^{-5} \leq \hat{W}[j] < k + \frac{1}{2} - 2^{-5}, \; k \in D_1 \\ 2 & \text{if } 2 - \frac{1}{2} - 2^{-5} \leq \hat{W}[j] \\ -2 & \text{if } \hat{W}[j] < -2 + \frac{1}{2} - 2^{-5} \, . \end{cases} \tag{19}$$

In Fig. 3, notice that when $\hat{W}[j] = k - \frac{1}{2} - 2 \cdot 2^{-5}$, the selection function (19) sets $s_j = k - 1$, and the maximum value for $W[j]$ is bounded by $\hat{W}[j] + \varepsilon = k - \frac{1}{2} + 2^{-5}$ which is within the interval I_{k-1}. Thus, $\beta = 5$, $\varepsilon = 3 \cdot 2^{-5}$, and the selection function (19) are valid for our chosen specifications. This value of ε requires $\hat{WPS}[j]$ and $\hat{WSC}[j]$ to have 5 and 4 fractional bits, respectively (or vice versa). Thus, $\hat{W}[j]$ is effectively the summation of two numbers with 4 fractional bits, and, hence, independent of the wordlength of $W[j]$.

We now modify the selection function of (19) to provide: (a) integer valued selection comparison constants, and (b) a recurrence equation that does not need the value of s_{j-1}. The result of (a) is that the comparisons of the selection function are low precision. Providing (b) allows the evaluation of the recurrence equation and the selection function to be pipelined.

We let $W[j] \rightarrow W[j] - \alpha$, where $\alpha = \frac{1}{2} + 2^{-\beta}$. Thus, we provide integer valued selection comparison constants by changing the boundaries of the selection function (19). To provide pipelining, we essentially remove the recursive effects of s_{j-1} from the residual recurrence equation (8) and implement it as part of the selection function. This results in a selection function depending on s_{j-1} (or some part of it), and a redefinition of the output of the carry-save adder ($VPS[j]$ and $VSC[j]$).

The On-Line MA Algorithm

In [1], we show that the radix-4 on-line MA algorithm for the specifications we have chosen (i.e., ρ=2, A_{max}=2) becomes (see Fig. 4)

Step 1 (Initialization)

$$VPS[-1] = \alpha = \tfrac{17}{32}, \quad VSC[-1] = 0, \quad IC_{-1} = 0 \tag{20}$$

Step 2 (Recurrence)

For $j = 0, \ldots, d+1$

2.1 $$P[j] = (Ax_j+y_j)4^{-2} - \tfrac{51}{32} = (G[j]+y_j)4^{-2} - \tfrac{51}{32} \tag{21}$$

2.2 $$(VPS[j], VSC[j]) = CSA(\, 4\cdot frac(VPS[j-1]),\ 4\cdot frac(VSC[j-1]),\ P[j]\,) \tag{22}$$

2.3 $$q_{j-2} = s_j = int(VPS[j]) + int(VSC[j]) + IC_j - 4IC_{j-1} \tag{23}$$

where

$$IC_j = int(frac(VPS[j],4) + frac(VSC[j],4)) \tag{24}$$

and

$frac(X)$ are the fractional bits of the operand X
$int(X)$ are the non-fractional bits of the operand X
$frac(X,Y)$ are the Y most significant fractional bits of the operand X.

Error Analysis

We now determine $\max(|E_S|)$. From (4) and (1)

$$E_S = (S^* - S)r^{d+\delta-1} = (r^{-\delta}(AX+Y) - S)r^{d+\delta-1} . \tag{25}$$

Using (8) and (25), with $j = d+\delta$, it can be to shown that

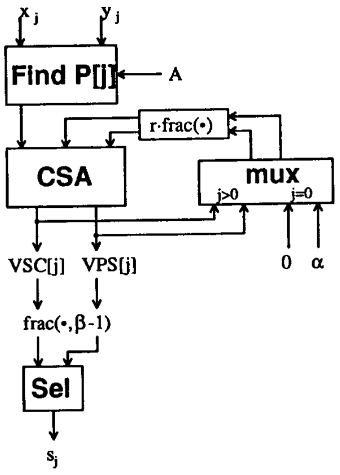

Fig. 4. A block diagram of the on-line MA.

$$|E_S| = |W[d+\delta-1] - s_{d+\delta-1}| \le \zeta . \tag{26}$$

We define $E_Q = \max(|E_S|) = \zeta$. Thus,

$$E_Q = \max(|W[j] - s_j|) = \max(u_k - k\ ,\ k - l_k) = \tfrac{1}{2} + \tfrac{1}{2}\Delta . \tag{27}$$

Therefore, the radix-4 on-line MA that we have just derived produces a result that is within $E_Q r^{-d} = \frac{13}{24}4^{-d}$ of the actual result, where d is the number of input/output digits.

4 VLSI Implementation

The algorithm that we have just defined is in a form that is easily translated to a hardware realization. To accomplish this translation, we first designed hardware that is a complete, logically correct implementation of the pipelinable on-line MA. This initial design may not be the fastest or smallest possible solution, but it serves as a well defined starting point for any custom design. Then we use a 1.5μ CMOS standard cell library to obtain worst-case design estimates of the on-line MA.

Notes on Figures

These notes apply to all of the remaining figures, unless otherwise stated. All iteration numbers (j, j+1, etc.) are left off. Any line (bus) without an indicated width is one bit wide. Any bit vector has individual bits subscripted such that a bit indexed by i has twice the weight of a bit indexed by i+1, unless otherwise specified. For example, the coefficient A is defined

$$A = -2a_{-2} + \sum_{i=-1}^{m} a_i 2^{-i} .$$

We define the mapping of y_j (and hence x_j) to its digit vector by using two's complement (TC) representation. (Recall that $y_j \in D_2 = \{-2, \cdots, 2\}$.) We denote the three bits used for the on-line digits

$$x_j = (\, x_j^0\, x_j^1\, x_j^2\,)_2 = -\,4x_j^0 + 2x_j^1 + x_j^2$$
$$y_j = (\, y_j^0\, y_j^1\, y_j^2\,)_2 = -\,4y_j^0 + 2y_j^1 + y_j^2\,.$$

Notice that the superscripts here denote individual bits, not powers.

Hardware Implementation

We pipeline the structure into three stages, where each stage is defined by one of the equations (21), (22), and (23). Thus, the MSD of the output of the on-line MA appears after δ+3 = 5 clock cycles. The implementation of $P[j]$ (21) is rather trivial and need not be discussed in detail. The constant $\frac{51}{31}$ of (21) is introduced when we let $W[j] \rightarrow W[j] - \alpha$. This constant can be easily incorporated into the hardware that implements (21) and does not require an actual subtraction. Fig. 5 depicts a block diagram of the recurrence equation that includes the multiplexing of the initial values of *VPS* and *VSC*.

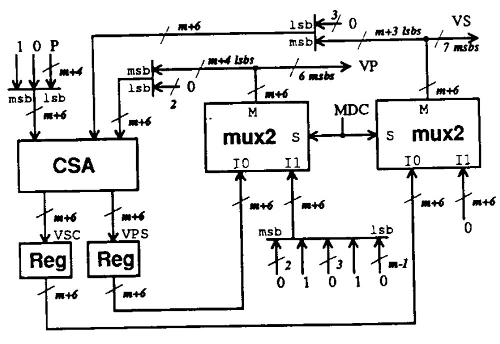

Fig. 5. The carry-save adder stage.

Notice that the outputs of this stage (VP and VS) occur after the multiplexers, and, hence, do not fully agree with the general block diagram of Fig. 4. This position allows a register that has clear/preset capabilities to emulate the combination of the depicted register and multiplexer, which was used in our standard cell VLSI design. The result of doing this is that the MA will be busy for d+δ+1 cycles rather than d+δ as defined by the algorithm. Fig. 6 depicts an architecture for the selection function (23). In the figure, the **add** block is a 2-bit adder with carry-in CI and carry-out CO as labeled. The **carry** block produces the carry-out C of the sum of two 4-bit operands A+B. To be able to detect when the MSDs arrive at the input of the on-line MA, we require a single line *MD* such that *MD* = 1 when the input is the MSD, and *MD* = 0, otherwise. In the figures, *MDC* and *MDS* are the pipelined MD signals for the CSA stage and selection function stages, respectively.

A bit-slice realization of the on-line MA provides simple expansion of the MA, and, hence, the precision of the multiplicand A. The processing of the least significant $m-1$ bits of the multiplicand is characterized by the i-th bit-slice ($2 \le i \le m$) of Fig. 7. Therefore, to provide a modular design, we first design the hardware to process the most-significant-bits of the multiplicand (the integer bits and one fractional bit), then, for a

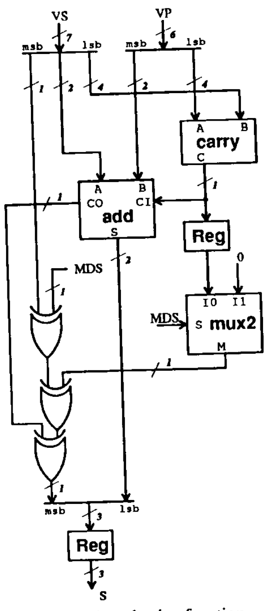

Fig. 6. The selection function.

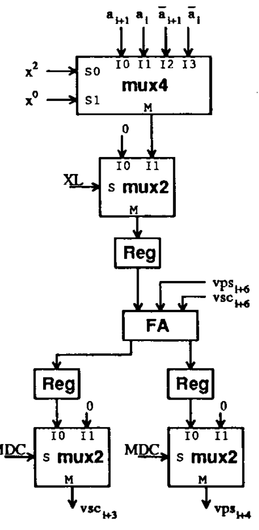

Fig. 7. The i-th bit-slice.

given m, add $m-1$ bit-slices to the structure to complete the on-line MA. This design methodology was used in our standard cell VLSI design. (In Fig. 7, the signal XL is the logical OR of the bits x^0, x^1, and x^2.)

VLSI Simulation Results

The worst-case estimates for the cycle time of each level of the pipe are

First stage (find P)	: 28.0ns
Second stage (CSA)	: 17.3ns
Third stage (selection)	: 18.3ns

The cycle times for the second and third stages do not include the additional time delay resulting when a new computation begins. However, simulations indicate that the cycle times of these stages are still less than the 28.0ns of the first stage. We do not include the delays associated with the new computation cycle since it is anticipated that a clever design should be able to approach the cycle times given above. Therefore, the clock cycle τ_P of the hardware is determined by the first stage, yielding $\tau_P = 28.0\text{ns}$. Thus, the MSD of the output appears $5\tau = 140\text{ns}$ after the MSD of the input. Therefore, the worst-case sampling rate of a filter implemented with this standard cell module is 7.14MHz. Recall that m is the number of fractional bits for the coefficient. The number of transistors in the standard cell on-line MA is

$$1778 + 200(m-1)\ . \tag{28}$$

Thus, there are 200 transistors in a bit-slice. The size of a bit-slice is 874λ by 104λ, and the size of the standard cell MA is

$$874\lambda \text{ by } (1146 + 104(m-1))\lambda\ . \tag{29}$$

The VTI 1.5μ CMOS standard cell library states that $\lambda = 0.8\mu$. (VTI's 1.5μ process yields a minimum channel length of 1.5μ.)

We compare our standard cell on-line MA with a standard cell parallel MA generated with the VTI design tools. We choose to implement an 18-bit parallel MA since this is very close to our proposed chip design [1]. The cycle time of the 18-bit standard cell parallel MA is 118.5ns, and each MA consumes an area of 4324λ by 4652λ (3.4592mm by 3.7216mm). The cycle time of 118.5ns will be significantly increased when additional hardware is included to reduce the effects of nonlinear oscillations in the filter's response. The Booth multiplier generated by the VTI design tools is fairly compact and fast. Thus, we expect a custom design of the on-line MA to yield higher improvements (in percent) of speed and area when compared with a custom parallel implementation.

5 Nonlinear Oscillations

A filter implemented with the direct form II filter structure can be severely affected by all four kinds of nonlinear oscillations [9], [11], [12], [17], [18], (zero-input overflow oscillations, zero-input limit cycles, forced overflow oscillations, and forced limit cycles). However, where speed is of concern, the direct form II structure is unsurpassed as a general purpose filter structure because of its ability to accept input samples at a rate of one new sample every multiply-add computation time. Conventional methods for handling nonlinear oscillations require the inclusion of saturation arithmetic (SA) [11] and controlled-rounding arithmetic (CRA) [12] in the recursive loops of the direct form II second-order filter structure to eliminate zero-input overflow oscillations and zero-input limit cycles. However, since CRA and SA are implemented in the recursive loop, the sampling period of the filter is increased by the amount of time necessary to evaluate the implemented CRA or SA. Thus, the filter's maximum sampling rate is reduced. (In [1], we define an algorithm that provides SA for the minimally redundant radix-4 on-line MA without increasing the sampling period.) Neither CRA nor SA provide the elimination of forced overflow oscillations or forced limit cycles [12], [19]. Error-feedback circuits have been developed [20] to eliminate some forced overflow oscillations. However, these circuits require significantly more hardware and still retain some oscillations.

The method we employ is to increase the working wordlength of the filter such that all zero-input and forced nonlinear oscillations are eliminated. (We define the working wordlength to be the wordlength used for the operands and results in all operations of the filter.) The basic idea is to increase the working wordlength such that all internal overflows are eliminated and such that limit cycles do not approach the least significant bit of the data. We define b_L to be the minimum number of bits required to fully contain the maximum limit cycle, and b_O to be the minimum number of additional bits required to fully contain the maximum signal value when the magnitude of the input is bounded by 1 (i.e., eliminate zero-input and forced overflow oscillations). We have determined that [1]

$$b_L = 1 + \left\lceil \frac{3m}{2} + \log_2 \left\lceil \frac{32E}{\pi} \right\rceil \right\rceil \tag{30}$$

$$b_O = 3 + \left\lceil \frac{3m+1}{2} \right\rceil \tag{31}$$

where m is the number of fractional bits used to represent the coefficients and E is the maximum quantization error after each multiplication. Thus, to eliminate all zero-input and forced nonlinear oscillations by increasing wordlengths, the working wordlength of the filter b_W must be defined by

$$b_W = b_O + b_D + b_L \tag{32}$$

where b_D is the desired number of bits of the output data.

As shown in Fig. 8, a b_I-bit input is transformed to b_W bits by concatenating b_O sign extension bits to the MSB of the input and then concatenating $b_D - b_I + b_L$ zeros to the LSB of the input. (Notice, if the input data is b_D bits long, then b_L zeros are concatenated to the LSB of the input.) In [1], we show that adequate scaling of the filter results in a scale factor C at the output of the section bounded by $2^{-b_O} \le C \le 1$. Since the input is extended to b_W bits in the manner just described, then scaling the b_W-bit output of the filter section by C yields a b_D-bit scaled output that is fully contained in the $b_O + b_D$ MSBs of the b_W-bit output. Thus, the scaled output will not be affected by limit cycles.

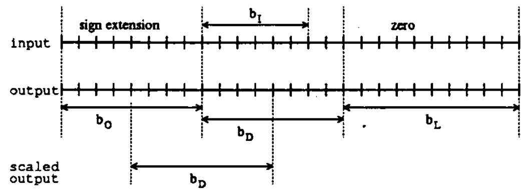

Fig. 8. Example of increasing wordlengths to eliminate nonlinear oscillations.

In our implementation, we approximate C by a shift of b_S bits (as is frequently done)

$$b_S = \left\lceil -\log_2 C \right\rceil . \tag{33}$$

Thus, $0 \le b_S \le b_O$. Since we increase the working wordlength such that no internal overflows can occur, and we guarantee that the output is free of limit cycles, then the output of the filter is within a quantization error of the ideal output (using the same limited precision coefficients). The conventional methods for eliminating nonlinear oscillations do not provide this benefit.

We define d_W to be the minimum number of digits that can represent all numbers in the range $[-2^{b_W-1}, 2^{b_W-1})$. Thus, for $r = 4$ and $\rho = 2$,

$$d_W = \left\lceil \frac{b_W}{2} \right\rceil . \tag{34}$$

However, this does not eliminate redundant representations of the working wordlength that require more than d_W digits. Therefore, we define d_R to be the minimum number of digits that can represent all redundant representations of all numbers in the range $[-2^{b_W-1}, 2^{b_W-1})$. When $\rho = r - 1$, the value of d_R becomes dependent on the algorithm, and it may not even exist. However, when $r = 4$ and $\rho = 2$,

$$d_R = \left\lceil \frac{b_W + 1}{2} \right\rceil . \tag{35}$$

Notice that (34) and (35) differ by 0 and 1 when b_W is odd and even, respectively. Thus, for the minimally redundant case where $r = 4$ and $\rho = 2$, the number of digits required to represent all redundant representations of all numbers in the range $[-2^{b_W-1}, 2^{b_W-1})$ (35) is at most one digit more than the minimum number of digits required to represent all numbers in the range $[-2^{b_W-1}, 2^{b_W-1})$. Therefore, given b_W bits, we use d_R digits of representation to eliminate all nonlinear oscillations.

6 Summary

We have presented an on-line multiply-add module that enables high filter sampling rates when used to implement the direct form II second-order filter structure. The on-line module's regular design provides a simple method for expanding the data wordlengths without affecting the sampling rate of the filter. Thus, we increase the internal wordlength of the filter such that the potentially devastating nonlinear oscillations are eliminated without lowering the sampling rate of the filter. This is not possible with conventional implementations. Another benefit of increasing the wordlength of the filter is that the output of the filter is within a quantization error of the ideal output (using the same limited precision coefficients). Thus, the signal-to-noise ratio is not reduced since no scaling of the input is required.

We used VTI's 1.5μ CMOS standard cell library and design tools to design and simulate the minimally redundant radix-4 implementation of the on-line MA. We pipelined the design into three stages, and, therefore, the delay from the MSD of the input to the output is 5 clock cycles (recall $\delta = 2$). Simulations yield 28.0ns for the worst-case estimate of the clock cycle. Therefore, the worst-case sampling period of the filter using this implementation is 140.0ns (i.e., a sampling rate of 7.14MHz). We compare this with an 18-bit standard cell parallel MA that has a clock rate of 118.5ns. This sampling period will be increased when additional hardware is introduced to handle the nonlinear oscillations. Thus, we demonstrate that the on-line MA is a useful design approach for high-speed digital filters.

References

[1] R. H. Brackert Jr., "Design and Implementation of a High-Speed Recursive Digital Filter Using On-Line Arithmetic," Ph. D. dissertation, UCLA, 1989.

[2] R. H. Brackert Jr., et al., "A High-Speed Recursive Digital Filter Using On-Line Arithmetic," *IEEE ISCAS*, 1989.

[3] M. D. Ercegovac, "On-line Arithmetic: An Overview," *Proc. SPIE Conf. Real-Time Signal Process.*, San Diego, 1984, pp. 667-680.

[4] M. Renfors and Y. Neuvo, "The Maximum Sampling Rate of Digital Filters Under Hardware Speed Constraints," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp.196-202, Mar. 1981.

[5] R. F. Woods, et al., "Systolic IIR Filters with Bit Level Pipelining," *Proc. ICASSP*, New York, 1988, pp. 2072-2075.

[6] K. S. Arun, "Ultra-High-Speed Parallel Implementation of Low-Order Digital Filters," *IEEE ISCAS*, 1986, vol. 3, pp. 944-946.

[7] K. Hayashi, et al., "Design of High-Speed Digital Filters Suitable for Multi-DSP Implementation," *IEEE Trans. Circuits Syst.*, vol. CT-33, pp. 202-217, Feb. 1986.

[8] K. K. Parhi and M. Hatamian, "A High Sample Rate Recursive Digital Filter Chip," *VLSI Signal Processing III*, pp. 3-14, New York: IEEE Press, 1988.

[9] H. Samueli, et al., "A Comparison of Recursive Digital Filter Structures Suitable for High-Speed Custom VLSI Implementation," *1987 Asilomar Conf. Circuits, Syst., Comput.*, 1987.

[10] A. Antoniou, *Digital Filters: Analysis and Design*. New York: McGraw-Hill, 1979.

[11] P. M. Ebert, et al., "Overflow Oscillations in Digital Filters," *Bell Syst. Tech. J.*, vol. 48, pp.2999-3020, Nov. 1969.

[12] D. Mitra and V. B. Lawrence, "Controlled Rounding Arithmetics, for Second-Order Direct-Form Digital Filters, that Eliminate All Self-Sustained Oscillations," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp.894-905, Sept. 1981.

[13] A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Trans. Electron. Comput.*, vol. EC-10, pp.389-400, Sept. 1961.

[14] M. D. Ercegovac and T. Lang, *CS252 Lecture Notes*, UCLA Comp. Sci. Dept., 1985.

[15] M. D. Ercegovac, "A General Method for Evaluation of Functions and Computations in a Digital Computer," Ph. D. dissertation, Univ. of Illinois, Urbana-Champaign, July 1975.

[16] D. M. Tullsen and M. D. Ercegovac, "Design and VLSI Implementation of an On-Line Algorithm," *Proc. SPIE Conf. Real-Time Signal Process.*, 1986.

[17] J. L. Long and T. N. Trick, "Roundoff-Noise Analysis for Fixed-Point Digital Filters Realized in Cascade or Parallel Form," *IEEE Trans. Audio Electroacoust.*, vol. AU-18, pp.107-122, June 1970.

[18] Z. Unver and K. Abdullah, "A Tighter Practical Bound on Quantization Errors in Second-Order Digital Filters with Complex Conjugate Poles," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp.632-633, July 1975.

[19] A. N. Willson, "Some Effects of Quantization and Adder Overflow on the Forced Response of Digital Filters," *Bell Syst. Tech. J.*, vol. 51, pp.863-887, Apr. 1972.

[20] A. N. Willson, "Error-Feedback Circuits for Digital Filters," *Electron. Let.*, vol. 12, pp.450-452, Sep. 1976.