The transfer function of Fig. 8 can be shown to be

$$\frac{E_o}{E_i} = \frac{R_2}{R_1} \cdot \frac{A_1}{\left(1 + \frac{R_2}{R_1} + L\right)} \cdot \frac{(s^2 + 2\omega_o s + \omega_o^2)}{\left(s^2 + \frac{\left(1 + \frac{R_2}{R_1} - L\right)}{\left(1 + \frac{R_2}{R_1} + L\right)} 2\omega_o s + \omega_o^2\right)} \tag{20}$$

with the gains of the individual fixed gain amplifiers combined for simplicity as

$$L = A_1^2 \frac{A_2}{4}(A_4 - A_1^4 A_2). \tag{21}$$

The overall loop gains remain closer to unity over a broader range of gain error and, hence, the circuit quality factor remains closer to the desired value. Positive one and two percent change in amplifier gains result in a 6.2 and 21.4% reduction of Q, respectively. Similarly, negative one and two percent changes in amplifier gains cause a 5.8 and 19.5% reduction of Q. Thus, application of this filter approach to fixed gain amplifier integration requires gain tracking of around 1% for the various amplifiers. This appears reasonable for amplifiers fabricated on a common substrate.

VII. Conclusions

Several improvements have been made to a known active filtering method using all-pass networks. A new all-pass circuit, based on a fixed gain integrable amplifier extends the filter's resonant frequency to the 40–50 MHz range. A modified configuration improves the off-resonant performance of the filter. A second modification of the configuration improves the stability of the circuit and reduces the sensitivity of the Q with respect to amplifier gain.

References

[1] D. J. Comer and J. E. McDermid, "Inductorless bandpass characteristics using all-pass networks," *IEEE Trans. Circuit Theory*, vol. CT-15, pp. 501–503, Dec. 1968.

[2] R. Tarmy and M. S. Ghausi, "Very high Q, insensitive active RC networks," *IEEE Trans. Circuit Theory*, vol. CT-17, pp. 358–366, Aug. 1970.

[3] D. J. Comer, "High-frequency narrow-band active filters," *IEEE Trans. Circuit Theory*, vol. CAS-33, pp. 838–840, Aug. 1986.

[4] G. S. Moschytz, "A high Q, insensitive active RC network, similar to the Tarmy-Ghausi circuit, but using single-ended operational amplifiers," *Electron. Lett.*, vol. 8, pp. 458–459, Sept. 1972.

[5] M. E. Van Valkenburg, *Analog Filter Design.* New York: Holt, Rinehart and Winston, 1982, ch. 7.

[6] D. T. Comer, "A wideband, fixed gain BiMOS amplifier," in *Proc. Int. ASIC Conf.*, Rochester, NY, Aug. 1993, pp. 27–30.

[7] ——, "A wideband integrated circuit amplifier for fixed-gain applications," *J. Analog Integrat. Circuits Signal Processing*, vol. 11, no. 3, pp. 243–251, Nov. 1996.

[8] ——, *Introduction to Mixed Signal VLSI.* New York: Array, 1994, ch. 10.

[9] D. J. Comer, "A sensitivity study of bandpass filters using all-pass networks," in *Proc. Second Asilomar Conf. Circuits Syst.*, Nov. 1968, pp. 202–207.

A Method of Eliminating Oscillations in High-Speed Recursive Digital Filters

John S. Fernando and Miloš D. Ercegovac

***Abstract*—A dynamic scaling (DS) method is proposed as a cost-effective means of eliminating overflow and limit cycle oscillations in fixed-point direct-form recursive filters. It is implemented by adding a DS unit to a fixed-point on-line module without modifying latter. On-line modules consume inputs and produce outputs digit serially, most significant digit first. The DS method introduces a shared exponent into the fixed-point computation at reasonable cost. Implementation in a 1.5 μ gate array shows that the DS method is twice as cost effective as the previously known precision extension method. The need for scaling between filter sections is also eliminated.**

***Index Terms*—On-line arithmetic, oscillations, recursive digital filters.**

I. Introduction

Recursive digital filters are usually implemented as a cascade of second-order sections to minimize finite-precision effects. Nevertheless, due to the inherent feedback loop of the recurrence, such finite-precision effects can induce overflow and limit cycle oscillations even in stable filters. Overflow oscillations are of large amplitude, caused by arithmetic overflow due to fixed precision, and limit cycles are small amplitude oscillations caused by round-off error in multiplication. Both types of oscillations can be induced with zero or nonzero inputs. The direct-form filter is faster and cheaper than other structures but more susceptible to oscillation. Eliminating or reducing oscillations without compromising speed is desirable.

Filter rate is also affected by the arithmetic used for computation. Particularly with conventional arithmetic (bit-parallel, least-significant-bit-first computation), the dependence of the output $y(n)$ on $y(n-1)$ limits operating speed. Recent papers have shown that most-significant-digit-first (MSDF) or on-line arithmetic can reduce the dependency to the digit level [2], [4]. MSDF and on-line algorithms compute digit serially, with inputs and outputs in MSDF sequence, enabling the computation of $y(n)$ to begin as soon as the most significant digit of $y(n-1)$is produced. Consequently, the clock rate of MSDF and on-line designs is *independent* of word length, whereas the clock rate of conventional designs diminishes with increasing word length. Therefore, MSDF and on-line designs are faster than conventional ones for words longer than a particular precision which depends on technology [3]. For on-line designs, the precision extension (PE) method can be used to suppress all oscillations without affecting sampling rate [1]. The large precision extension makes the PE method costly.

II. MSDF and On-Line Arithmetic

MSDF and on-line algorithms produce outputs digit-serially, beginning with the most significant digit (MSD). In on-line algorithms, all operands are in on-line form (digit serial MSDF). If some of the operands are in parallel form, the algorithm is referred to as an MSDF

Manuscript received April 4, 1994; revised January 7, 1997. This paper was recommended by Associate Editor F. J. Kurdahi.

J. S. Fernando is with AT&T Bell Laboratories, Inc., Allentown, PA 18103 USA.

M. D. Ercegovac is with the Computer Science Department, University of California, Los Angeles, CA 90024 USA.

Publisher Item Identifier S 1057-7130(97)07478-8.

algorithm. An on-line radix-r operand, x, assumed to be a fraction, is expressed in terms of its digits X_j as $x = \Sigma_{j=0}^{d-1} X_j r^{-j}$, where X_j is in the redundant digit set $\{-\rho, \cdots, \rho\}$ such that $r > \rho \geq r/2$. Having input and output digits in the same digit set allows cascading of on-line modules and facilitates recursive computation. An important characteristic of on-line computation is the on-line delay, δ_{imp}, which is the number of clocks between input and output digits of identical weight. The on-line delay is typically two to five clocks. On-line arithmetic offers a systematic approach to deriving digit-serial algorithms. Well established on-line algorithms exist for common operations. On-line and MSDF algorithms are particularly well suited for VLSI implementations of high-speed recursive filters [1], [3], [4].

Precision Extension (PE) Method

The PE method of eliminating all self-sustained oscillations in a stable fixed-point second-order filter ($y(n) = u(n) + ay(n-1) + by(n-2)$, where coefficients a and b are real-valued) using on-line arithmetic is described in [1]. The relevant results described in [1] are summarized below. The symbols used are as follows: m is the number of *fraction bits* representing coefficients, b_D the number of *desired* output bits, b_L the number of *additional* least significant bits required to, eliminate *limit cycles* from the desired output, b_O the number of *additional* most significant bits required to eliminate *overflow* oscillations from the desired output, b_W the *working* precision in bits, and E the maximum normalized quantization *error* due to multiplication.

The limit cycle magnitude is less than $2E(4/\pi)2^{1.5m}$ for pole magnitudes >0.9. Since the limit cycle corrupts the least significant part of the fixed-point result, extending working precision sufficiently at the least significant bit (LSB) end eliminates the limit cycle from the actual output. The required extension is given by (1). Overflow oscillations are caused by internal overflows. Assuming the quantization error is negligible, and that $u(n) < 1$, the number of bits required at the most significant bit (MSB) end to prevent overflow is given by (2). Thus, the working precision required to eliminate both overflow and limit cycles from the actual output is given by (3).

$$b_L = 1 + \left\lceil 1.5m + \log_2\left(\frac{8E}{\pi}\right)\right\rceil \tag{1}$$

$$b_O = 1 + \lceil 1.5m + .5 \rceil \tag{2}$$

$$b_W = b_O + b_D + b_L. \tag{3}$$

In implementations using conventional arithmetic, such an extension of precision reduces the sampling rate. In contrast, if on-line or MSDF arithmetic is used sampling rate is unaffected. The increase in hardware for PE is significant regardless of the arithmetic used. For example, having coefficients with 10 fraction bits requires a working precision of 44 bits. The dynamic scaling (DS) method is a less costly method of eliminating oscillations in on-line filters.

III. The DS Method

With a moderate extension of working precision, the MSD of the output $y(n)$ can be guaranteed to be zero during zero-input ($u(n) = 0$) limit cycle oscillations. This allows the output to be left-shifted by one digit for the computation of the next output $y(n+1)$, provided an exponent is introduced to keep track of the shifts. Shifting can be done again when the MSD becomes zero sometime later. By induction, the shifting can be done until the exponent is decremented to the point at which the desired output is zero for a given precision. Thus, limit cycles are eliminated from the desired output by increasing working precision just sufficient to guarantee a zero MSD when $u(n) = 0$. Thus, for radix-4 digits, the DS method requires a minimum working precision of $b_L + 2$ bits. Given $m, b_L + 2$ can be calculated from (1), where $E = 0.5$ for rounding. The number of bits required for the DS method and the simple PE method for various values of m is compared in Table I. The number of bits required for the PE method is calculated assuming $b_D = m$.

TABLE I
Precision (Bits) Required for DS and PE Schemes

Coeff. frac. bits (m)	8	10	12	14	16	18	20	22	24	26
Bits for DS (b_S)	16	19	22	25	28	31	34	37	40	43
Bits for PE (b_W)	36	44	52	60	68	76	84	92	100	108

TABLE II
Scaling Operations: Advance and Retard (for $n = 2$)

Operation	Input		Output	
	mantissa	exponent	mantissa	exponent
Advance	$0\,Y\,Y\,\cdots\,Y\,Y$	E_y	$Y\,Y\,\cdots\,Y\,Y\,0$	$E_y - 1$
Retard (n=2)	$Y\,Y\,\cdots\,Y$	E_y	$0\,0\,Y\,\cdots\,Y$	$E_y + 2$

Overflow oscillations are eliminated by incrementing the exponent (E_y) and right-shifting the output when $y(n)$ overflows. For radix-4 digits, the maximum value of E_y is $E_{\max} = \lceil b_O/2 \rceil$. To eliminate limit cycles from the desired output requires b_D zero bits to be absorbed into the exponent. For radix-4 digits, minimum value of E_y is $E_{\min} = -\lceil b_D/2 \rceil$.

The working precision for the PE method is about 2.5 times that required for the DS method as Table I shows. Cost savings result because, as shown later, the cost of introducing an exponent into the computation is less than that for precision extension according to (3).

The DS Algorithm

The DS algorithm is based on two scaling operations, *advance* and *retard*, performed on on-line operands. The advance operation performs a 1-digit left shift of the mantissa and also decrements the exponent. The retard operation performs an n-digit right shift and increments the exponent by n. Table II illustrates advance and retard (for $n = 2$) operations performed on operand y with digits Y.

Normalized $u(n)$ is denoted by u, and $y(n), y(n-1), y(n-2)$ are denoted by y, y_1, y_2. E_u is the exponent of u and E_y is the exponent for both y_1 and y_2. E'_y is the exponent of y and is also the new exponent of y_1. Advance operation is indicated by $ADV = 1$. R_u is the magnitude of shift, in digits, for retard on u and R_y the same for retard on y_1 and y_2. R_v is the magnitude of shift, in digits, caused by overflow in y_1 from previous computation ($R_v \epsilon \{0, 2\}$). Signals $Y1z = 1$ and $Y2z = 1$ indicate if leading fraction digit of y_1 or y_2 is zero.

DS Algorithm

Begin

Step 0: Initially $E_y = 0, y_1 = 0, y_2 = 0$

Step 1: Compute $R_y = \max(E_u - E_y, R_v); R_y \geq 0$

Step 2: Find (Boolean)

$$ADV = (R_v = 0)(Y1z)(Y2z)(E_u < E_y)(E_y > E_{\min})$$

Step 3: Compute

$$R_u = \begin{cases} E_y - E_u - 1 & \text{if } ADV = 1 \\ E_y - E_u + R_v & \text{if } ADV = 0 \\ & \text{and } R_v \geq E_u - E_y \\ 0 & \text{otherwise} \end{cases}$$

Step 4: $E'_y = E_y + R_y - ADV$

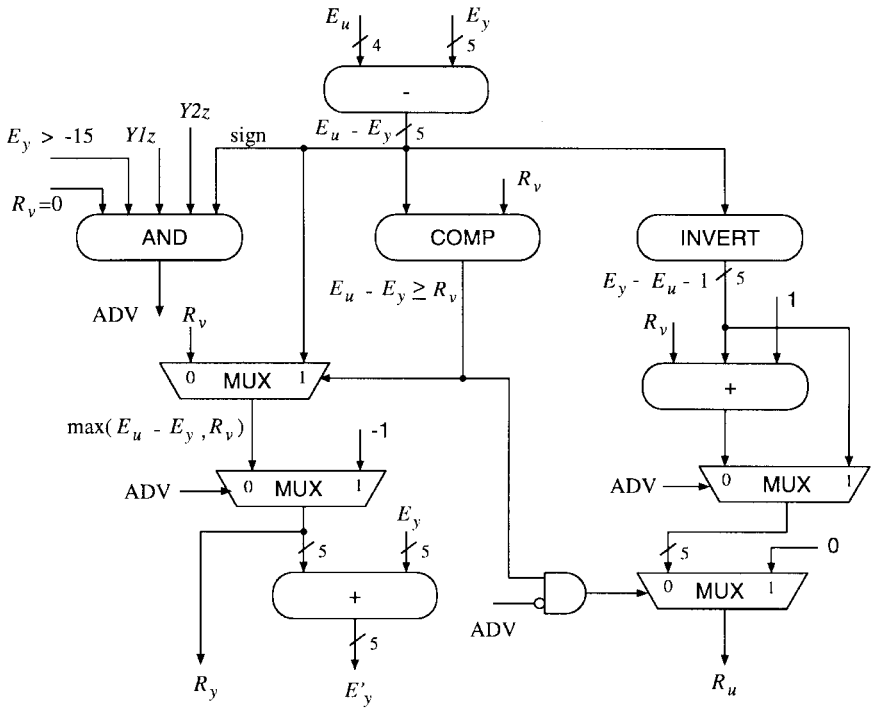

Fig. 2. Exponent computations.

Step 5: Retard/Advance u, y_1, y_2

Step 6: Execute on-line fixed point computation

$\cdot\ y = u + ay_1 + by_2$

Step 7: $E_y \leftarrow E'_y$

Step 8: Go to Step 1

End

Step 1 of the DS algorithm indicates the two conditions that require y_1 and y_2 to be retarded: when $E_u > E_y$, or when y_1 overflows from the previous computation. Since both conditions may occur simultaneously, the maximum retard value is chosen. Step 2 specifies the conditions for advance: no overflow, leading fraction digits of u, y_1, and y_2 must be zero, and the exponent must be greater than the minimum value. As Step 3 indicates, the retard value of u is different because u is normalized (i.e., MSD is nonzero). Normalization is convenient because u needs no advance subsequently. Also, advance by more than 1 increases complexity and delay of the DS unit. Step 4 calculates the new exponent based on the scaling operations performed. Step 5 scales the on-line operands that are input to the on-line fixed point computation in Step 6. To reduce complexity, y_1 and y_2 share the same exponent and are retarded or advanced identically. The radix and digit set chosen for the DS algorithm are the same as that for the fixed-point computation. Besides extending precision, no other changes are required in the fixed-point computation.

IV. Implementation

This section describes gate array (LCA10K) implementations of the DS scheme and the PE method and compares performance and cost. Both implementations are for $m = 8$ and $b_D = 8$. Fig. 1 shows the block diagram of the DS scheme. Three scalers and a single exponent unit are connected to a word module (two cascaded radix-4 on-line MA modules described in [3]) that computes the on-line fixed point recurrence $y = u + ay_1 + by_2$. The exponent unit computes the retard values and advance signal as shown in the graph in Fig. 2.

In general, for d-digit working precision, $d + \delta_{\text{imp}}$ clocks are required. For 8-digit working precision, the computation of $y(n)$ takes 12 clocks, $C0, C1, \cdots, C11$. Synchronization of the digits is done by introducing appropriate delays (1D, 4D, and 6D in Fig. 1) based on the known latencies of the fixed-point computation and the scalers. The latency of the fixed-point computation is nine clocks (two cascaded MA modules, each with a four-clock latency, plus a latch) and that of each scaler is two. Since the delay for the y_1 loop must be 12 (same as the computation cycle) a delay of 1 is inserted (shown as 1D in Fig. 1). To synchronize inputs for the fixed-point computation, y_1 is delayed another 12 clocks to produce y_2. Since the scaler takes two clocks, a delay of 10 clocks must be inserted in the y_2 path. The 10-clock delay is split into delays 6D and 4D, to achieve synchronization at the scaler input and at the fixed-point computation input.

The DS unit operates as follows. In clock C0 the scalers receive R_u, R_y, and ADV from the exponent unit and also the MSD's of u, y_1, and y_2. The on-line inputs are scaled and the fixed-point computation begins in C2. The exponent unit begins computation in C10, when inputs $E_y, E_u, R_v, Y1z$, and $Y2z$ are gated by GATE. ADV, R_y and R_u have to be stable in C1. Computation of E'_y is not critical because the value is not needed at the beginning of the iteration. In C0, the most significant fraction digits are available at the inputs of the scalers shown in Fig. 1. Since the scalers have a delay of two clocks, the scaled outputs are available in C2. The signal CLD clears the digit registers of the scalers before each computation cycle begins. CLZ clears the flip-flop that outputs Z to the scalers.

For a working precision of 8 radix-4 digits (16-bits) with $E_{\min} = -15$ and $E_{\max} = 15$ the entire circuit takes 4437 gates (includes digit delays) and runs at a minimum clock period of 11.2 ns. The component sizes and minimum clock periods are: DS Unit—1179 gates and 9.4 ns, Normalizer—426 gates and <9.4 ns, Cascaded

TABLE III
COMPARISON OF DS SCHEME AND PE SCHEME FOR WORD MODULES

Scheme	Gates	Rate (Msamples/s)	t_{clk} (ns)	Rate/Gate (Msamples/s/gate)
DS	4437	7.44	11.2ns	0.00167
PE	5130	4.13	11.0ns	0.00081

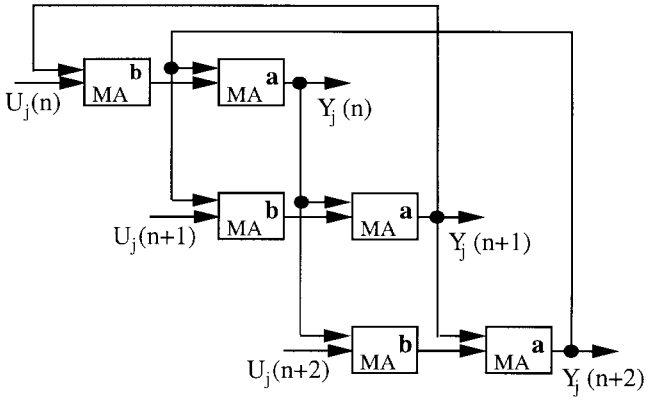

Fig. 3. Maximum-rate array for 16-bit I/O.

MAs—2424 gates and 10 ns. A large range for the exponent was chosen to show that the clock rate of the DS Unit is not critical even for large values of m and b_D.

Rate/Cost Comparison of Schemes: To compare the DS scheme requires implementing the *equivalent* PE scheme, with identical values of m and b_D and different working precisions. For $m = b_D = 8$, the PE method requires $b_W = 14 + 8 + 14 = 36$ [(1)–(3)] and the DS scheme requires $b_W = 16$ (Table I). The exponent range required for the DS scheme is $-4 \leq E_y \leq 7$. The two implementations are compared in Table III. The sampling rate of the DS implementation is 80% higher than that of the equivalent PE implementation and is 13% smaller. Thus the rate/gate ratio for the DS scheme is more than twice that for the PE implementation. As m or b_D is increased, the required working precision for the PE scheme increases relative to that of the DS scheme (Table I) and the increase in cost of the scalers and the exponent unit of the DS scheme is relatively small. Thus for higher working precision the DS scheme is even more cost effective.

Applying DS to Arrays: The DS method may be applied to maximum-rate arrays such as that shown in Fig. 3 for 16-bit I/O. Since the output of a word module is input to the next word module without delay, the array delivers the maximum rate achievable with the given MA modules (MaxRate $= 1000/t_{\rm clk}\delta_{\rm imp}$).

Rather than advancing on-line inputs before each computation, the advancing can be done once for several computations. Thus leading zeros are allowed to accumulate and are removed simultaneously. Limit cycles would still be effectively eliminated since leading zeros in the I/O would be gradually removed. Consider an array similar to Fig. 3 with one DS unit placed before the first word module (top left in array). To allow word modules without a DS unit to handle overflow in the output of the preceeding module, the working precision of each word module is extended by $d_{\rm ovf} = N_{\rm DS} - 1$ digits. $N_{\rm DS}$, the number of word modules in the maximum-rate array with DS, must satisfy $N_{\rm DS} = \lceil(\delta_{\rm imp} + d + d_{\rm ovf})/\delta_{\rm imp}\rceil$. For the example considered, $d = 8, \delta_{\rm imp} = 4, d_{\rm ovf} = 3$, and $N_{\rm DS} = 4$. Thus the array requires 8 MA modules with a working precision of 22 bits.

With one DS unit for the array, the normalizer feeding the independent inputs $u(n)$ through $u(n+3)$ to the word modules must perform block normalization, producing four on-line inputs with a single exponent E_u. The cost of a maximum rate word module array with DS is estimated at 14 000 gates (Block Normalizer $\simeq$1000 gates, eight MA modules with 22-bit I/O 11 856 gates, DS Unit 1179 gates). In contrast using PE in the array takes 28 260 gates (six word modules, with 36-bit precision). Thus, the cost of a PE scheme is twice the cost of a DS scheme for maximum rate arrays. With DS, the maximum rate is given by $\mathrm{MaxRate}_{\rm DS} = 1000/t_{\rm clk}(\delta_{\rm imp} + (2/N_{\rm DS})$. For the example considered, $d = 8, \delta_{\rm imp} = 4$ and $N_{\rm DS} = 4$, $\mathrm{MaxRate}_{\rm DS} = 24.4$ MSamples/s, which is 88% of the maximum rate without DS (27.8 MSamples/s). Although the array using PE is more regular and 12% faster, the maximum-rate array with DS is more cost-effective with a rate/cost ratio of 1.8 times that of an array with PE.

V. CONCLUSION

We have proposed the DS scheme and shown that it is more cost-effective than the PE method in eliminating limit cycles oscillations and overflow oscillations in on-line implementations of direct form recursive filters. The scheme is implemented by adding a DS unit to a fixed-point on-line word module. Except for precision adjustment, easily achieved by adding bit-slices, no changes to the word module are required. Implementation in a 1.5-μm gate array technology shows that, for word modules with an output precision of 8 bits, the DS scheme is 13% smaller than the PE scheme and has a sampling rate 80% higher. Maximum-rate arrays using the DS scheme require only half as many gates as an array using PE and operate at 88% of the maximum rate. For higher output precision the DS scheme is even more cost effective. Having automatic scaling, the DS scheme eliminates the need for scaling between cascaded sections.

REFERENCES

[1] R. H. Brackert, Jr., "Design and implementation of a high-speed recursive digital filter using on-line arithmetic," Ph.D. dissertation, Univ. California, Los Angeles, 1989.
[2] M. D. Ercegovac and T. Lang, "Most-significant-digit-first and on-line arithmetic approaches for the design of recursive filters," in *Proc. 23rd Annual Asilomar Conf. Signals, Syst. Comput.*, 1989, pp. 7–11.
[3] J. S. Fernando and M. D. Ercegovac, "Conventional and on-line arithmetic designs for high-speed recursive digital filters," in *VLSI Signal Processing V*, K. Yao, R. Jain, W. Przytula, J. Rabaey, Eds. Piscataway, NJ: IEEE Press, 1992, pp. 81–90.
[4] O. C. McNally, J. V. McCanny, and R. F. Woods, "A 40 Megasample IIR filter chip," in *Proc. Int. Conf. Application Specific Array Processors*, IEEE Press, 1991, pp. 416–430.