an arrangement can operate as a delta or delta–sigma modulator.

ACKNOWLEDGMENT

The author would like to thank Dr. J. C. C. Nelson, who first drew his attention to the relationship between the techniques described here and those employed in the DDA.

REFERENCES

[1] F. De Jager, "A new method of p.c.m. transmission using the 1 unit code," *Philips Res. Rep.*, vol. 7, pp. 442–466, Dec. 1952.
[2] E. N. Protonotarios, "Slope overload noise in differential pulse code modulation," *Bell Syst. Tech. J.*, vol. 46, pp. 2119–2162, Nov. 1967.
[3] T. R. H. Sizer, *The Digital Differential Analyser*. London, England: Chapman and Hall, 1968.
[4] G. B. Lockhart, "Digital encoding and filtering using delta modulation," *Radio Electron. Eng.*, vol. 42, Dec. 1972.
[5] J. E. Flood and M. J. Hawksford, "Exact model for delta modulation processes," *Proc. Inst. Elec. Eng. (London)*, vol. 118, Sept. 1971.

A New Hardware Realization of Digital Filters

ABRAHAM PELED, STUDENT MEMBER, IEEE, AND BEDE LIU, FELLOW, IEEE

Abstract—**A new approach to the implementation problem of digital filters is presented. This approach capitalizes on recent advances in semiconductor memory technology and is shown to offer significant reductions in cost and power consumption for the same speed of operation as that of existing realizations. Furthermore, this approach makes possible speeds of operation which cannot be achieved by existing realizations. The proposed approach yields a very flexible hardware configuration and a discussion of the various options is presented together with a comparison to existing realizations. The mean-squared error resulting from the use of finite word length is analyzed.**

I. INTRODUCTION

DIGITAL FILTERS have become an increasingly attractive replacement for analog filters due to recent advances in semiconductor technology. As the speed of operation increases, either to permit real-time processing of wide-band signals or to time share the arithmetic unit, there is a rapid increase in hardware complexity, as measured by the number of IC's used, and in power consumption. The major factor causing this increase lies with the high-speed multipliers.

Jackson *et al.* have proposed an approach to the implementation of digital filters that is well suited to large-scale integrated (LSI) construction [1]. They also propose a very efficient serial multiplier that produces a rounded binary number, and lends itself particularly well to multiplexed operation. Using current TTL technology this multiplier can accommodate a bit rate of approximately 25 MHz.

In this paper we propose a new approach for the hardware implementation of fixed point arithmetic digital filters. The new realization calls for the storing of the finite number of possible outcomes of an intermediate arithmetic operation, and using them to obtain the next output sample through repeated addition and shifting operations. No multiplications are needed. This hardware implementation is highly modular and uses only standard available IC's. We show that the proposed realization offers significant savings, for the same speed of operation as existing realizations, in terms of hardware complexity and power consumption. Furthermore, with this new realization, filters can be constructed to operate at speeds that are difficult or impossible to achieve with existing realizations. As in all realizations, the use of finite word lengths in the processor results in roundoff errors. These are analyzed and expressions are derived that allow the designer to choose the design parameters to achieve a specified performance.

II. A NEW APPROACH TO IMPLEMENTING DIGITAL FILTERS

A digital filter is characterized by an input–output relationship of the form

$$y_n = \sum_{k=0}^{N} a_k x_{n-k} - \sum_{k=1}^{N} b_k y_{n-k} \tag{1}$$

where $\{x_n\}$ is the input sequence, $\{y_n\}$ the output sequence, and $\{a_k\}_{k=0}^{N}$, $\{b_k\}_{k=1}^{N}$ are the filter coefficients. For hardware realizations, it is convenient to consider the z-domain transfer function of the digital filter $H(z)$, given by

$$H(z) = \sum_{j=0}^{N} a_j z^{-j} / (1 + \sum_{j=1}^{N} b_j z^{-j}) \tag{2}$$

Manuscript received February 8, 1974; revised July 1, 1974. This work was supported by the Air Force Office of Scientific Research, USAF, under Grant AF-AFOSR 71-2101, and the National Science Foundation under Grant GK-24187.

A. Peled was with the Department of Electrical Engineering, Princeton University, Princeton, N.J. 08540. He is now with the IBM Thomas J. Watson Research Laboratories, Yorktown Heights, N.Y. 10598.

B. Liu is with the Department of Electrical Engineering, Princeton University, Princeton, N. J. 08540.

where z^{-1} is the unit delay operator. It has been long recognized [2]–[5] that out of the numerous equivalent configurations that realize (2), the most advantageous are the cascade and parallel forms shown in Fig. 1. The cascade form corresponds to a factorization of the numerator and denominator polynomials of (2) to bring forth an $H(z)$ of the form

$$H(z) = a_0 \prod_{j=1}^{M} \left[\frac{(\alpha_{2j} z^{-2} + \alpha_{1j} z^{-1} + 1)}{(\beta_{2j} z^{-2} + \beta_{1j} z^{-1} + 1)} \right] \tag{3}$$

where M is the least integer greater than or equal to $N/2$. The parallel form results from a partial fraction expansion of (2),

$$H(z) = \gamma_0 + \sum_{j=1}^{M} \left[\frac{(\gamma_{1j} z^{-1} + \gamma_{0j})}{(\beta_{2j} z^{-2} + \beta_{1j} z^{-1} + 1)} \right]. \tag{4}$$

The various aspects of realizing higher order filters using basic second-order sections, including appropriate scaling to avoid overflow and ordering for minimum roundoff noise have been discusses in detail in [4]–[6]. Let us focus on the implementation of a second-order section specified by the input–output relationship

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} - b_1 y_{n-1} - b_2 y_{n-2} \tag{5}$$

with $a_2 = 0$ in the case of parallel realization. Let us assume that all signals are bounded by ± 1 and that B binary bits including sign bit in 2's complement code are used to represent the data, that is

$$x_k = -x_k^0 + \sum_{j=1}^{B-1} x_k^j 2^{-j} \qquad x_k^j = 0 \text{ or } 1. \tag{6}$$

Equation (5) can be rewritten as

$$\begin{aligned} y_n = {} & a_0 \Big(\sum_{j=1}^{B-1} x_n^j 2^{-j} - x_n^0 \Big) + a_1 \Big(\sum_{j=1}^{B-1} x_{n-1}^j 2^{-j} - x_{n-1}^0 \Big) \\ & + a_2 \Big(\sum_{j=1}^{B-1} x_{n-2}^j 2^{-j} - x_{n-2}^0 \Big) - b_1 \Big(\sum_{j=1}^{B-1} y_{n-1}^j 2^{-j} - y_{n-1}^0 \Big) \\ & - b_2 \Big(\sum_{j=1}^{B-1} y_{n-2}^j 2^{-j} - y_{n-2}^0 \Big). \end{aligned} \tag{7}$$

Define a function φ with five binary arguments as follows:

$$\varphi(x^1, x^2, x^3, x^4, x^5) = a_0 x^1 + a_1 x^2 + a_2 x^3 - b_1 x^4 - b_2 x^5. \tag{8}$$

Then we can rewrite (7) as

$$\begin{aligned} y_n = \sum_{j=1}^{B-1} 2^{-j} \varphi & (x_n^j, x_{n-1}^j, x_{n-2}^j, y_{n-1}^j, y_{n-2}^j) \\ & - \varphi(x_n^0, x_{n-1}^0, x_{n-2}^0, y_{n-1}^0, y_{n-2}^0). \end{aligned} \tag{9}$$

The function φ can take on only $2^5 = 32$ distinct values depending on the binary vector that forms its arguments. It can be realized by a combinatorial network or by a read only memory (ROM) addressed by the arguments. Thus (9) suggests a possible mechanization of (5) requiring only addition and shifting operations. This approach is illustrated in Fig. 2. Data enter serially into shift registers

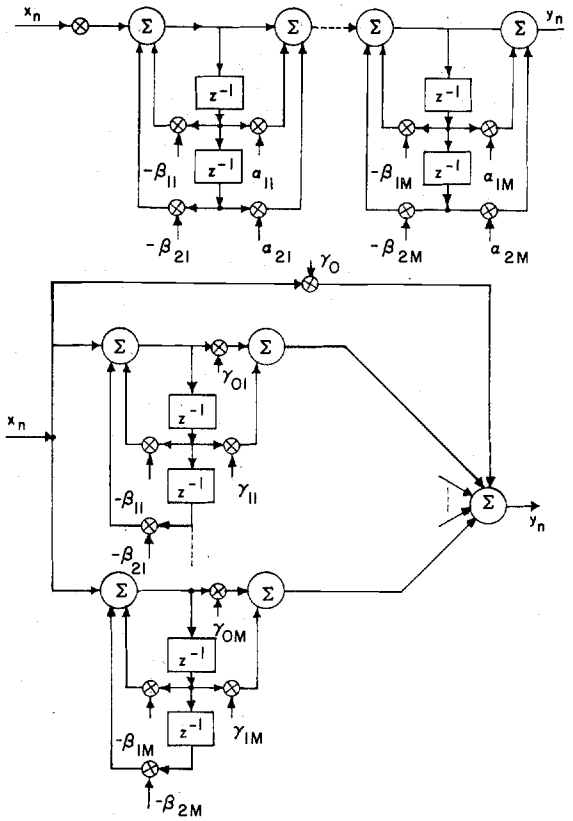

Fig. 1. (a) Cascade realization. (b) Parallel realization.

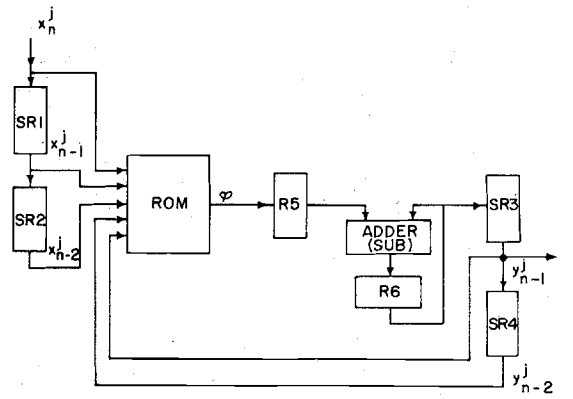

Fig. 2. New hardware realization of a second-order section of a digital filter.

$SR1$–$SR4$, with the least significant bit leading. At each shift, a new vector $\{x_n^j, x_{n-1}^j, x_{n-2}^j, y_{n-1}, y_{n-2}^j\}$ appears at the input of the circuit realizing φ. The output φ is loaded into register $R5$ which is connected to one of the two inputs of the accumulator with a possible sign change for $j = 0$. The other input of the accumulator is hardwired to the output register ($R6$) with a 1 bit right shift. After B such shifts, the value in register $R6$ is rounded and the accumulator cleared. This rounded value is y_n, which is shifted serially into $SR3$, and the processor is ready to compute the next sample y_{n+1}.

Fig. 3 gives an example of a typical second-order section and its corresponding function φ defined by its truth table with $B = 8$. This table can be used directly to program a ROM of 256 bits organized as 32×8 bit

MEMORY MAP FOR SECOND-ORDER SECTION

MEMORY ADDRESS					CONTENTS							
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	1	1	0
0	0	0	1	0	0	1	1	1	0	1	0	0
0	0	0	1	1	0	0	1	1	1	0	0	1
0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	1	1	1	0	0	1	1	0	0
0	0	1	1	0	0	1	1	1	0	1	0	0
0	0	1	1	1	0	0	1	1	1	1	1	1
0	1	0	0	0	1	1	1	1	0	1	0	1
0	1	0	0	1	1	0	1	1	1	0	1	1
0	1	0	1	0	0	1	1	0	1	0	0	1
0	1	0	1	1	0	0	1	0	1	1	1	1
0	1	1	0	0	1	1	1	1	1	0	1	1
0	1	1	0	1	1	1	0	0	0	0	0	1
0	1	1	1	0	0	1	1	0	1	1	1	1
0	1	1	1	1	0	0	1	1	0	1	0	1
1	0	0	0	0	0	0	0	0	1	1	0	0
1	0	0	0	1	1	1	0	0	1	1	0	0
1	0	0	1	0	0	1	1	1	1	0	1	0
1	0	0	1	1	0	0	1	1	1	1	1	1
1	0	1	0	0	0	0	0	0	1	1	0	0
1	0	1	0	1	1	1	0	1	0	0	1	0
1	0	1	1	0	0	1	1	1	1	1	1	1
1	0	1	1	1	0	1	0	0	0	1	0	1
1	1	0	0	0	1	1	1	1	1	0	1	1
1	1	0	0	1	1	1	0	0	0	0	0	1
1	1	0	1	0	0	1	1	0	1	1	1	1
1	1	0	1	1	0	0	1	1	0	1	0	1
1	1	1	0	0	0	0	0	0	0	1	0	0
1	1	1	0	1	1	1	0	0	0	1	1	1
1	1	1	1	0	0	1	1	1	0	1	0	1
1	1	1	1	1	0	0	1	1	1	0	1	1

sign bit ——— binary point

Fig. 3. Typical second-order section and its φ.

words. In this example, $a_1 = 0.095$, $a_2 = -0.1665478$, $a_3 = 0.095$, $b_1 = -1.8080353$, and $b_2 = 0.9129197$. The five columns of the "memory address" correspond to the five binary arguments of the φ function, i.e., $(x_n{}^j, x_{n-1}{}^j, x_{n-2}{}^j, y_{n-1}{}^j, y_{n-2}{}^j)$. The first bit in the contents is the sign bit and the binary point is to the right of the sign bit. Here φ has been scaled down by 2.

Let us compare briefly this approach with existing realizations which use multipliers in the arithmetic unit as [1]. The realization illustrated in Fig. 2 substitutes a multiplier and storage for 5 filter coefficients with a combinatorial network or ROM and a shift register. Another difference is that our method requires no code conversions between 2's complement and sign magnitude [1], [7]. However, we hasten to mention that by fixing the function φ we lost the flexibility inherent in using multipliers. Each additional second-order section to be realized requires a distinct function φ and therefore an additional network. Nevertheless, we will show in Section IV that as a result of the recent advances in semiconductor memory technology our approach is significantly more economical in terms of power consumption and hardware complexity then existing realizations.

III. OPTIONS AND PERFORMANCE OF THE NEW APPROACH

In this section we discuss the various options of hardware implementation of second-order sections via (9), and analyze their performances. Higher order filters will be discussed in Section IV.

We have already mentioned that the function φ can be realized by a combinatorial network or by a ROM programmed appropriately. Although it may prove more economical in many instances to design a combinatorial network to realize φ, especially if high-density logic devices such as programmable logic arrays are used, the exact cost and performance would be heavily dependent on the specific coefficients and the intended volume of production. A most convenient way to realize φ, however, is to use ROM or programmable read only memory (PROM). In this paper we assume that ROM's are used to realize φ which makes our analysis independent of the filter coefficients and therefore applicable to every digital filter.

Now let us proceed to evaluate the performance of the second-order section implemented as in Fig. 2. Essentially two steps have to be completed before we can shift a new data bit into the data registers $SR1$ to $SR4$.

Step 1: Evaluation of the function φ. The time needed is the access time of the ROM—t_φ.

Step 2: Adding two B bit numbers. The time needed is the addition time—t_α.

The two steps just mentioned can be performed concurrently, provided that the accumulator has an additional shift register for storing the addend. Let

$$t_R = \max \{t_\alpha, t_\varphi\} \tag{10}$$

then $1/t_R$ will be the maximum bit rate at which the filter can operate.

As an example let us consider $B = 12$. Using standard available TTL integrated circuits and bipolar memory we can get $t_\alpha \approx 40$ ns, $t_\varphi \approx 50$ ns. Then the second-order section illustrated in Fig. 2 will operate at a bit rate of approximate 20 MHz, have a package count of 20 IC's, and consume 9.6 W. Assuming a 12-bit word, this bit rate implies that this section can operate in real time on a signal having a bandwidth in excess of 800 kHz or it can be multiplexed between some 200 signals having a bandwidth of 4 kHz each. We will also mention that should Fig. 2 be implemented with emitter-coupled logic (ECL) IC's the bit rate would be at least 50 MHz, the package count 33 IC's and power consumption 26.4 W.

Let us return for a moment to (9) which incorporates the basic approach we propose. Its mechanization as suggested in Fig. 2 is but one of many possible. Fig. 4 depicts another possible mechanization of (9) for the case of 8-bit data. Here data are loaded in parallel into $R1$ to $R5$, and there are 8 separate but identical ROM's storing the values of the function φ. The outputs of the ROM's, φ^0 to φ^7, are added in a tree like structure with a proper number of shifts hardwired, using 7 adders in this case. Again, by providing each adder with two storage registers, concurrent operation of all levels is possible. It is clear from the above that the number of bits used for the data will only determine the number of levels needed but will not affect the throughput rate.

As an example let us consider $B = 8$. Using standard TTL IC's and bipolar memory, a word rate of 20 MHz for the second-order section in Fig. 4 can be achieved. The package count is 60 IC's and the power consumption

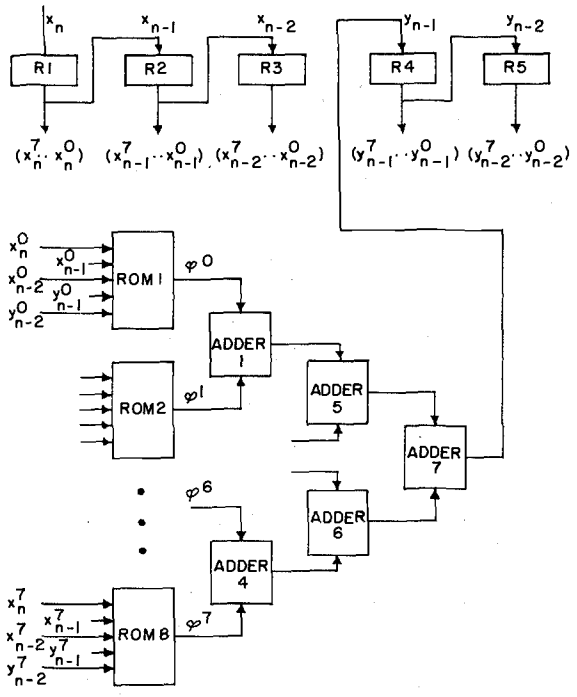

Fig. 4. High-speed realization of the second-order section.

24 W. This word rate implies that the section can operate in real time on a signal with a 10 MHz bandwidth. It should be noted that to achieve such a speed using multipliers would be very difficult or impossible unless several ECL multipliers are used. Such multipliers dissipate considerably more power and have a high-package count (e.g., a 9×9-bit multiplier described in [8] performs the multiplication in 35 ns and has 36 IC's dissipating 12.6 W). If ECL IC's are used to implement the section of Fig. 4, it is possible to realize a 50 MHz word rate, an operating speed unachievable using present multipliers.

Clearly, the two mechanizations of (9) illustrated in Figs. 2 and 4 represent two extreme cases. In the first one the data bits are processed serially, while in the second one all data bits are processed in parallel. Configurations that fall between these two extremes are possible and offer operating speeds between 2 MHz to 20 MHz word rate with a package count between 20 IC's to 60 IC's.

IV. HIGH-ORDER DIGITAL FILTERS— A COMPARISON WITH EXISTING REALIZATIONS

In this section we discuss the realization of high-order digital filters and multiplexed digital filters using second-order sections implemented as described in Section III, and attempt to compare their performance with the implementations proposed in [1], [7].

It is rather difficult to compare between the implementation we propose and the existing realization without referring to a specific speed to operation and filter order. Therefore we chose to present four illustrative examples: 1) an eight-order digital filter realized in parallel operating at a high speed (1 MHz word rate), 2) an eight-order digital filter realized in cascade operating at a medium speed (250 kHz word rate), 3) a multiplexed digital filter, where a second-order section is shared between 128 low-speed channels of 8 kHz word rate each and 4) a 96 channel, tenth-order filter each with a word rate of 8 kHz.

Fig. 5 illustrates a possible mechanization of an eight-order digital filter realized in parallel using the configuration of Fig. 2 for the second-order section. Since $a_2 = 0$ in parallel form, the ROM's needed are of an organization of $2^4 \times B$ or $16 \times B$. The contents of ROM J at address $\{x_n{}^j, x_{n-1}{}^j, v_{n-1}{}^j, v_{n-2}{}^j\}$ be

$$c[\text{ROM } J] = \gamma_{0J} x_n{}^j + \gamma_{1J} x_{n-1}{}^j - \beta_{1J} v_{n-1}{}^j - \beta_{2J} v_{n-2}{}^j \tag{11}$$

where $\{\gamma_{0J}, \gamma_{1J}, \beta_{1J}, \beta_{2J}\}$ are the coefficients or the Jth section of (4). The additional memory ROM 5 is used to combine the output of the sections and to allow for any necessary scaling to accommodate the dynamic range [4], [5].

Let us assume that the filter in Fig. 5 operates with $B = 12$. As mentioned in Section III, the second-order section in this case can operate at a word rate considerably faster than 1 MHz, if standard TTL IC's are used. The package count for the entire filter is approximately 72 IC's and the power consumption 28 W. The realization suggested in [1] uses a serial multiplier which can be constructed with approximately 20 standard TTL IC's operating at a word rate of 1 MHz (for 12 bit data) and consuming 8 W. Three such multipliers are needed for each second-order section if the filter is to operate at a 1 MHz word rate. At least 12 multipliers are needed for this eight-order filter. Thus the package count of the multipliers alone will be 240 IC's and their power dissipation in 96 W. To this, one has to add registers for the data, adders, and code converters.

The parallel TTL fast multiplier which uses 4 bit by 4 bit multipliers such as Texas Instruments SN74284, SN74285 can multiply 8 bit by 12 bit numbers in 90 ns. It has 24 IC's consuming 10 W. Although the speed is higher, additional hardware is required to parallel operation. Consequently, the final figures of package count and power consumption are similar to those obtained by the use of the serial multiplier.

Fig. 6 shows an eight-order digital filter realized in cascade form. Since each section operates successively, the speed of operation for the filter is reduced by a factor of 4 to a 250 kHz word rate. Again let us assume $B = 12$. Although the memory requirement is 32×12 bits for each section, a higher density memory such as $128 \times 8 = 1024$ bits can be used since each section is addressed at a different time. Therefore the package count of the arithmetic unit will be 33 IC's consuming approximately 14 W. The realization suggested in [1] operating at the same speed would need only 3 multipliers which will be shared between the sections. Thus the package count for the multipliers alone, in that case will be 60 IC's consumming 24 W.

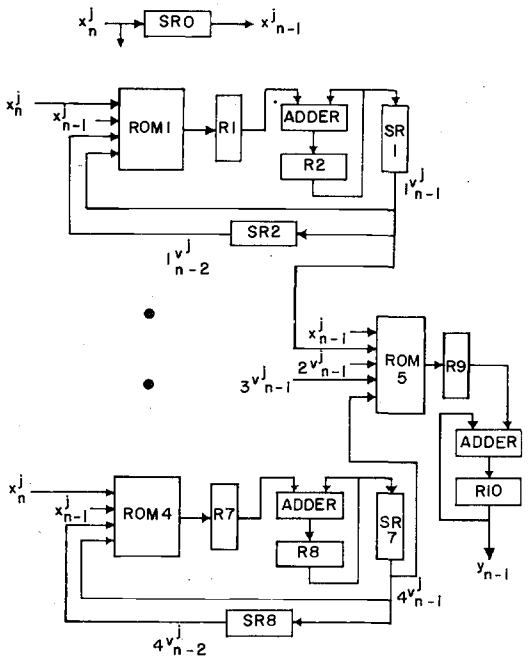

Fig. 5. Eighth-order filter realized in parallel form.

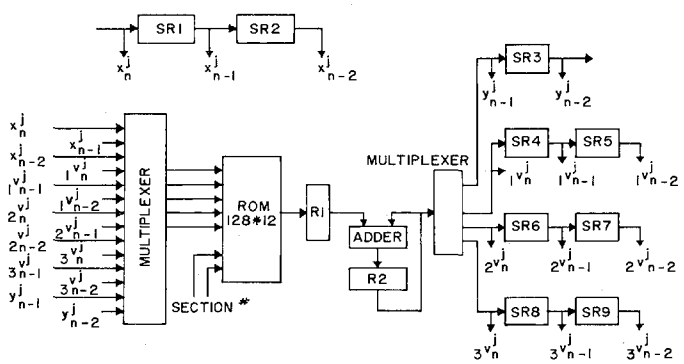

Fig. 6. Eighth-order digital filter realized in cascade form.

Consider a multiplexed digital filter where the arithmetic unit of the second-order section is shared between 128 second-order filters operating at a 8 kHz word rate each with $B = 12$. Intuitively, it would seem that the new approach proposed in this paper would be inferior to existing realizations due to the large amount of memory needed. However a closer examination shows that this is not the case. Fig. 7 shows the arithmetic unit of such a multiplexed filter. The data registers are not shown since the same number would be required in the existing realization. Assuming a parallel configuration for the second-order section a total of $128 \times (16 \times 12)$ bits of storage will be needed if $B = 12$. If we use a high density storage of 4096 bits organized as 512×8 bits (such as Signetics 8205) then only 6 IC's will be needed for the memory. This brings the total package count for the arithmetic unit to 18 IC's consuming about 10 W. Again 3 multipliers are needed by a realization such as [1] to achieve the word rate. Thus a total of 60 IC's consuming 24 W are needed for the multipliers alone. To this, one has to add also storage for 512 filter coefficients adders, etc.

Finally, let us consider a tenth-order digital filter to

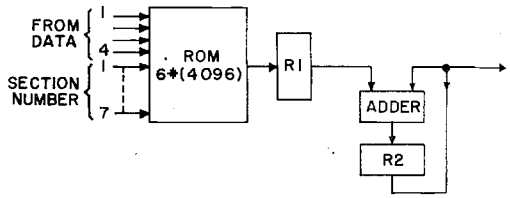

Fig. 7. Arithmetic unit of a multiplexed filter.

be multiplexed between 96 low-speed channels as the one mentioned in [7]. There, the same serial multiplier as in [1] is used and it is suggested that by using off-the-shelf medium-scale integrated (MSI) T^2L circuits, the tenth-order filter would require about 190 IC's and consume approximately 100 W. If we assume a cascade realization for the tenth-order filter as the one in Fig. 6 which can operate at 400 kHz word rate based on a 2 MHz second-order section of Fig. 2, two such sections are needed to accommodate the 96 channels. By using 4096-bit ROM's, the total package count using standard IC's will be 54 IC's consuming 22 W.

All the above comparisons hold only for the case that the filters to be implemented are a part of a fixed system and not programmable. In the case of programmable filters, i.e., filters where characteristics are changed by inserting different coefficients, our approach would call for the use of random access memory (RAM) for implementing the function φ. Consequently, the programming would be considerably more time consuming, since up to $2^5/5 = 6.4$ times more data must be inserted then in the existing realization.

The examples we have presented clearly show that the proposed approach will offer significant savings in package count and power consumption over existing realizations. These savings further translate in reduced cost of parts, reduced assembly costs and increased reliability.

V. ERROR ANALYSIS

In the previous sections we have not discussed the errors due to the finite word length used to represent the data and the values of the function φ. The effect of finite word length on digital filters using a multiplier has been investigated, and methods are well known to compute the mean-squared error [9], [10]. These results can be applied to the hardware configurations we proposed in this paper. In this section we derive expressions that will enable the system designer to evaluate the expected performance of the various hardware configurations proposed, in terms of the mean-squared error committed.

To begin with, a scaling of φ is often necessary to prevent overflow. For the second order described by (5), the scaling of the coefficients a_0, a_1, and a_2 is straightforward. For stability reason, we must have $|b_1| < 2$ and $|b_2| < 1$. According to (8), it is then possible for φ to be larger than 1. Therefore, it may be necessary to store in the ROM the value of φ scaled by 2^{-K}. In the majority of cases, $K = 1$ would be adequate. Effectively, this scaling corresponds to modifying (8) and (9) to

$$\varphi(x^1,x^2,x^3,x^4,x^5) = 2^{-K}(a_0x^1 + a_1x^2 + a_2x^3 - b_1x^4 - b_2x^5) \tag{12}$$

and

$$y_n = \sum_{j=1}^{B-1} 2^{-j+K}\varphi(x_n{}^j,x_{n-1}{}^j,x_{n-2}{}^j,y_{n-1}{}^j,y_{n-2}{}^j) - 2^K\varphi(x_n{}^0,x_{n-1}{}^0,x_{n-2}{}^0,y_{n-1}{}^0,y_{n-2}{}^0). \tag{13}$$

Although the desired output is given by (5), due to finite word length used, the actual computed output is different. Let $\bar{x}_n$ be the value of x_n rounded to B bits, and let $\bar{x}_n{}^j$ be its jth bit. The roundoff error $e_n{}' = \bar{x}_n - x_n$ is an uncorrelated sequence with zero mean and the variance of $e_n{}'$ is $2^{-2B}/3$ [9]. Let $\bar{\varphi}(\cdot,\cdot,\cdot,\cdot,\cdot)$ be $\varphi(\cdot,\cdot,\cdot,\cdot,\cdot)$ defined by (14) but rounded to B bits. Let v_n be the actual computed output and let $v_n{}^j$ be its jth bit, $j = 0,1,\cdots, B-1$. v_n is the round value of

$$\sum_{j=1}^{B-1} 2^{-j+K}\bar{\varphi}(\bar{x}_n{}^j,\bar{x}_{n-1}{}^j,\bar{x}_{n-2}{}^j,v_{n-1}{}^j,v_{n-2}{}^j) - 2^K\bar{\varphi}(\bar{x}_n{}^0,\bar{x}_{n-1}{}^0,\bar{x}_{n-2}{}^0,v_{n-1}{}^0,v_{n-2}{}^0). \tag{14}$$

As described in Section II, v_n is calculated by adding the shifted $\bar{\varphi}$'s. Only B bits needed to be kept in the accumulation, and rounding occurs at the last addition. Let the roundoff error introduced be denoted by $e_n{}'''$. Then $e_n{}'''$ are uncorrelated each with zero mean and variance $2^{-2B}/3$. Therefore,

$$v_n = \sum_{j=1}^{B-1} 2^{-1+K}\bar{\varphi}(\bar{x}_n{}^j,\bar{x}_{n-1}{}^j,\bar{x}_{n-2}{}^j,v_{n-1}{}^j,v_{n-2}{}^j) - 2^K\bar{\varphi}(\bar{x}_n{}^0,\bar{x}_{n-1}{}^0,\bar{x}_{n-2}{}^0,v_{n-1}{}^0,v_{n-2}{}^0) + e_n{}'''. \tag{15}$$

Each $\bar{\varphi}$ is related to the unrounded value φ by

$$2^K\bar{\varphi}(\bar{x}_n{}^j,\bar{x}_{n-1}{}^j,\bar{x}_{n-2}{}^j,v_{n-1}{}^j,v_{n-2}{}^j) = \varphi(\bar{x}_n{}^j,\bar{x}_{n-1}{}^j,\bar{x}_{n-2}{}^j,v_{n-1}{}^j,v_{n-2}{}^j) + e_{n,j}{}'' \tag{16}$$

where $e_{n,j}{}''$ is due to roundoff. The $e_{n,j}{}''$ are zero mean, uncorrelated, and each have variance $2^{-2B}/3$.

On using (13), (15), and (16), we have

$$v_n = e_n{}''' + \sum_{j=1}^{B-1} 2^{-j}e_{n,j}{}'' - e_{n,0}{}'' + a_0e_n{}' + a_1e_{n-1}{}' + a_2e_{n-2}{}'. \tag{17}$$

Define the roundoff error at the output of the second-order section by

$$e_n = v_n - y_n. \tag{18}$$

By using (5), (17), and (18), we have

$$e_n + b_1e_{n-1} + b_2e_{n-2} = a_1e_n{}''' + a_2e_{n-1}{}' + a_3e_{n-2}{}' + \delta_n \tag{19}$$

where

$$\delta_n = e_n{}''' + \sum_{j=1}^{B-1} 2^{-j}e_{n,j}{}'' - e_{n,0}{}'' \tag{20}$$

is an uncorrelated zero mean sequence and the variance of δ_n can be easily shown to be $(2^{-2B}/3)[1 + (1 - 2^{-2B})/(1 - 2^{-2})]$ or approximately $(7/9)2^{-2B}$.

The error $\{e_n\}$ can be regarded as the output of a linear system with two inputs $\{e_n{}'\}$ and $\{\delta_n\}$ as illustrated in Fig. 8 where $B(z) = 1/(1 + b_1z^{-1} + b_2z^{-2})$. It is straightforward to show that the e_n has zero mean and mean-square value given by

$$E\{\epsilon_n{}^2\} = \frac{2^{-2B}}{3}\frac{1}{2\pi i}\oint_{|z|=1}[H(z)H(1/z) + (7/3)B(z)B(1/z)]\,(dz/z). \tag{21}$$

Given the filter coefficients, such integrals can be evaluated, either numerically or algebraically, by a computer program or table [11]. To evaluate the mean-squared error of higher order filters, realized in either parallel or cascade form, we assume that each second-order section contributes an uncorrelated error component as the one described above and the total output error is obtained by summing these various errors weighted by the transfer function from their point of injection to the output. For the parallel form, the output error is simply the sum of the various errors from the second-order sections, Fig. 1(b). That is

$$E\{(e_{P,n})^2\} = \sum_{j=1}^{M} E\{(e_n{}^j)^2\} \tag{22}$$

where $e_{P,n}$ is the error in parallel form realization and M is the number of second-order sections. Equation (22) does not take into account the additional error committed in the final scaling of the outputs of the second-order sections before summing them up. However, if the scaling coefficients are known this can be easily accounted for.

For the cascade form realization, the output mean-squared error can be easily computed using the error model given in Fig. 8.

$$E\{(e_{C,n})^2\} = \sum_{j=1}^{M-1} \frac{E\{(e_n{}^j)^2\}}{2\pi i}\oint H^j(z)H^j(1/z)z^{-1}\,dz \tag{23}$$

where $e_{C,n}$ is the error in the cascade form realization, M is the number of second-order sections, and $H^j(z)$ is the transfer function between the output of the jth second-order section and the final output. We see that the common factor in the error expression (21) is the exponential decrease as a function of the word length B, as is the case in existing realizations. Equations (21)–(23) allow the system designer to choose the word length B according to the errors that can be tolerated.

VI. CONCLUSION

We have proposed a new approach to the implementation problem of digital filters, which offers significant savings in cost and power consumption over existing realizations and permits operation at higher speeds than those achievable by existing realizations. The savings made possible by the proposed approach are due to the fact that the flexibility inherent in using a multiplier and thus permitting an infinite number of transfer functions

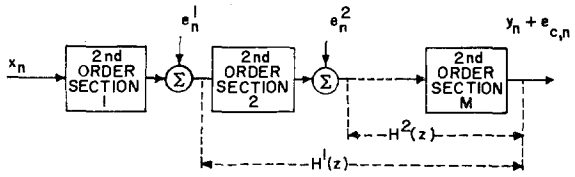

Fig. 8. Model for the mean squared error of a digital filter realized in cascade form.

to be realized is not utilized in practice when implementing filters for fixed systems, and our approach eliminates that over capacity. An error analysis was carried out and expressions derived which will allow the designer to choose the word length which will insure a specified performance.

The continuing emergence of technologies which further reduce the power requirements and increase the devsity of semiconductor memory such as silicone on saphire (SOS) appears to suggest that in the future the exchange between multiplier logic and memory bits for digital filters will only become more economic and offer even more savings than what has been pointed out by us in this paper.

ACKNOWLEDGMENT

The authors wish to thank the reviewers for their valuable comments and suggestions. One of the reviewers also pointed out that the basic configuration of Fig. 2 has been disclosed recently in [12].

REFERENCES

[1] L. B. Jackson, J. F. Kaiser, and H. S. McDonald, "An approach to the implementation of digital filters," *IEEE Trans. Audio Electroacoust. (Special Issue on Digital Filters: The Promise of LSI Applied to Signal Processing)*, vol. AU-16, pp. 413–421, Sept. 1968.

[2] J. F. Kaiser, "Digital filters," in *Systems Analysis by Digital Computer*, F. F. Kuo and J. F. Kaiser, eds. New York: Wiley, 1966, ch. 7.

[3] J. B. Knowles and R. Edwards, "Effects of a finite-word-length computer in a sampled-data feedback system," *Proc. Inst. Elec. Eng. (London)*, vol. 112, pp. 1197–1207, June 1965.

[4] L. B. Jackson, "On the interaction of roundoff noise and dynamic range in digital filters," *Bell Syst. Tech. J.*, vol. 49, pp. 159–184, Feb. 1970.

[5] ——, "Roundoff-noise analysis for fixed-point digital filters realized in cascade or parallel form," *IEEE Trans. Audio Electroacoust. (Special Issue on Digital Filtering)*, vol. AU-18, pp. 107–122, June 1970.

[6] W. S. Lee, "Optimazation of digital filters for low roundoff noise," in *Proc. 1973 IEEE Int. Symp. Circuit Theory*, Toronto, Ont., Canada, Apr. 1973, pp. 418–421.

[7] S. K. Tewksbury, "Special purpose hardware implementation of digital filters," in *Proc. 1973 IEEE Int. Symp. Circuit Theory*, Toronto, Ont., Canada, Apr. 1973.

[8] S. D. Peraris, "A 40-ns 17-bit by 17-bit array multiplier," *IEEE Trans. Comput.* (Short Notes), vol. C-20, pp. 442–447, Apr. 1971.

[9] B. Liu, "Effect of finite word length on the accuracy of digital filters—a review," *IEEE Trans. Ciruit Theory (Special Issue on Active and Digital Networks)*, vol. CT-18, pp. 670–677, Nov. 1971.

[10] A. V. Oppenheim and C. J. Weinstein, "Effects of finite register length in digital filtering and the fast Fourier transform," *Proc. IEEE*, vol. 60, pp. 957–976, Aug. 1972.

[11] K. J. Åström, E. I. Jury, and R. G. Agniel, "A numerical method for the evaluation of complex integrals," *IEEE Trans. Automat. Contr.* (Short Papers), vol. AC-13, pp. 468–471, Aug. 1970.

[12] A. Croisier, D. J. Esteban, M. E. Levilion, and V. Rizo, "Digital filter for PCM encoded signals," U. S. Patent 3 777 130, Dec. 3, 1973.

Some Considerations in the Design of Multiband Finite-Impulse-Response Digital Filters

LAWRENCE R. RABINER, MEMBER, IEEE, JAMES F. KAISER, FELLOW, IEEE, AND RONALD W. SCHAFER, MEMBER, IEEE

Abstract—Although much has been learned about the relationships between design parameters for finite impulse-response (FIR) low-pass digital filters, very little is known about the relationships between the parameters of multiband filters. Thus given a set of design specifications for a multiband FIR filter (e.g., filter band edge frequencies and desired ripples in each of the bands) it is difficult to choose a set of modified parameters which will yield an acceptable filter using a standard FIR design algorithm. By an acceptable filter we mean one with monotonic behavior of the frequency response in the DON'T-CARE or transition regions between bands and one providing at least the desired attenuation (or ripple) in each of the bands. In this paper, we examine the theoretical and practical issues of designing multiband filters and present several strategies for choosing the input parameters for the McClellan *et al.* filter-design algorithm to yield reasonable filters which meet arbitrary specifications.

Manuscript received May 29, 1974.
The authors are with Bell Laboratories, Murray Hill, N.J. 07974.

I. INTRODUCTION

THE PROBLEM of designing optimal (in a Chebyshev or minimax sense) linear phase, finite impulse-response (FIR) digital filters has two aspects. The first part involves choosing a set of parameters (e.g., band edge frequencies, impulse response duration, desired response in the various bands, and weights on the approximation error in each of the bands) to specify the desired filter. The second part