Transactions Briefs

A Hybrid Radix-4/Radix-8 Low Power Signed Multiplier Architecture

Brian S. Cherkauer and Eby G. Friedman

***Abstract*—A hybrid radix-4/radix-8 architecture targeted for high bit, general purpose, digital multipliers is presented as a compromise between the high speed of a radix-4 multiplier architecture and the low power dissipation of a radix-8 multiplier architecture. In this hybrid radix-4/radix-8 multiplier architecture, the performance bottleneck of a radix-8 multiplier, the generation of three times the multiplicand for use in generating the radix-8 partial product, is performed in parallel with the reduction of the radix-4 partial products rather than serially, as in a radix-8 multiplier. This hybrid radix-4/radix-8 multiplier architecture requires 13% less power for a 64 × 64-b multiplier, and results in only a 9% increase in delay, as compared with a radix-4 implementation. When the voltage supply is scaled to equalize delay, the 64 × 64-b hybrid multiplier dissipates less power than either the radix-4 or radix-8 multipliers. The hybrid radix-4/radix-8 architecture is therefore appropriate for those applications that must dissipate minimal power while operating at high speeds.**

***Index Terms*—Low power, multiplier, radix.**

I. Introduction

High speed digital multipliers are fundamental elements in signal processing and arithmetic based systems. The higher bit widths required of modern multipliers provide the opportunity to explore new architectures which would be impractical for smaller bit width multiplication. While much previous work has concentrated on reducing the delay of multipliers at the architectural level, very little effort has been spent on reducing the power dissipation of these multipliers. The power efficiency of multipliers has increased primarily due to improvements in technology, where power efficiency in a multiplier is defined here as the inverse of the multiplier power factor, the power dissipated per bit$^2 \cdot$ Hz [1].

The data in Fig. 1 describe the power factors for a number of recent implementations of digital multipliers. Sharma *et al.* utilized Booth radix-4 encoding along with a reduction array of carry save adders (CSA's) generated by a recursive algorithm to produce the 16 × 16-b multiplier in [2]. In [3], Yano *et al.* introduced the complementary pass transistor logic family (CPL) and implemented a 16 × 16-b multiplier in CPL which used no encoding but did use a Wallace tree for partial product reduction. Nagamatsu *et al.* presented a 32 × 32-b multiplier in which Booth radix-4 was used to generate the partial products and a tree of 4 : 2 counters was used to reduce these partial products [4]. Mori *et al.* designed a 54 × 54-b multiplier similar in structure to that of [4], also utilizing Booth radix-4 and 4 : 2 counters [5]. In [6], Goto *et al.*, presented a 54 × 54-b multiplier with Booth radix-4 partial product generation, but used a regularly structured tree

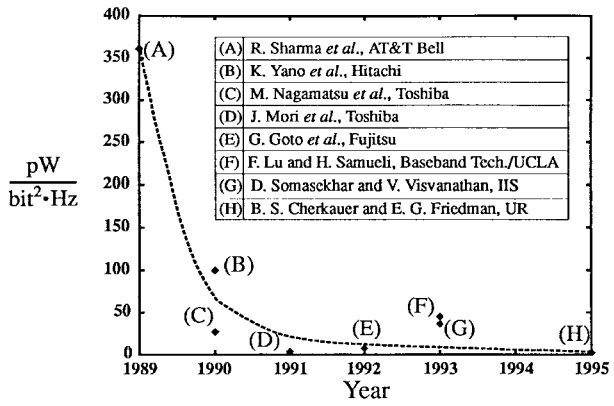


Fig. 1. Multiplier power factor.

for partial product reduction, thereby simplifying the physical layout. Lu and Samueli were most concerned with throughput in the design of the multiplier-accumulator described in [7], and thus they presented a 13-stage, deeply pipelined 12 × 12-b multiplier-accumulator which used no encoding and was implemented with a quasi-domino dynamic logic family. Somasekhar and Visvanathan were also concerned with the high throughput required by many DSP applications, and in [8] they presented an 8-b, unencoded multiplier pipelined at each half-bit stage. The data point representing the multiplier described in this paper is a 64 × 64-b hybrid radix-4/radix-8 multiplier with a Dadda reduction tree [9]. As described in this brief, this multiplier achieves high power efficiency by operating the radix-4 encoding and reduction in parallel with the high speed addition required by the radix-8 encoding.

A hybrid Booth radix-4/radix-8 multiplier architecture is presented in this paper as a method to tradeoff speed and power dissipation in two's complement signed multipliers. The improved speed and power dissipation characteristics of this new multiplier architecture are compared with that of standard radix-4 and radix-8 based multipliers. The hybrid radix-4/radix-8 architecture presented in this paper is described in Section II. The circuit components used to construct the multipliers are briefly summarized in Section III, while the speed and power dissipation characteristics of the three multiplier architectures are compared in Section IV. Finally, some conclusions are drawn in Section V.

II. Hybrid Radix-4/Radix-8 Multiplier Architecture

In order to perform high speed multiplication, an encoding scheme, such as that proposed by Booth [10], is often used. The objective of Booth encoding is to reduce the number of partial products which are summed to generate the complete product, and thereby decrease the time required to compute the final product. However, the encoding itself incurs a certain delay penalty which must be balanced against the delay saved by reducing the number of partial products. These delay tradeoffs in Booth encoding are well established [11]–[13]. In this paper, it is shown that in addition to reducing the number of partial products, higher radix Booth encoding also reduces the power dissipation of the multiplier architecture. It has been shown in [13] and [15] that by combining modified Booth radix-4 encoding

Manuscript received March 28, 1995; revised July 19, 1996. This work was supported in part by the National Science Foundation under Grant MIP-9208165, and Grant MIP-9423886, the Army Research Office under Grant DAAH04-93-G-0323, and by a Grant from the Xerox Corporation. This paper was recommended by Associate Editor W. Burleson.

B. S. Cherkauer is with Intel Corporation, Santa Clara, CA 95052 USA.

E. G. Friedman is with the Department of Electrical Engineering, University of Rochester, Rochester, NY 14627 USA.

Publisher Item Identifier S 1057-7130(97)03659-8.

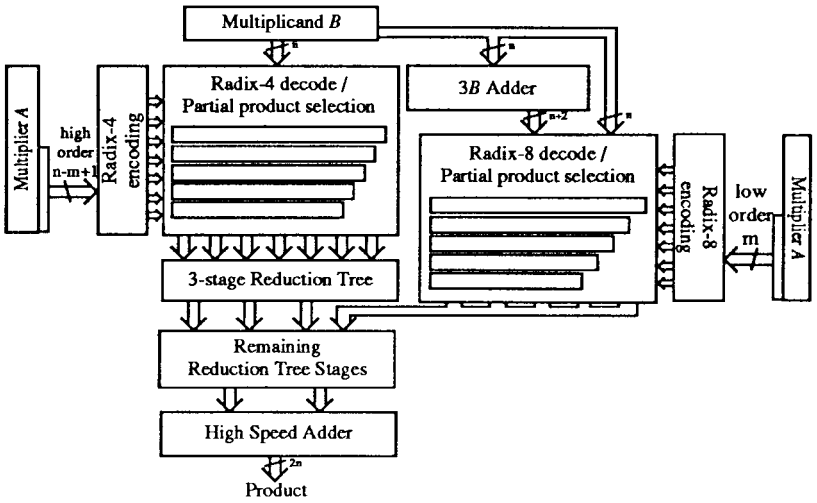


Fig. 2. Hybrid radix-4/radix-8 multiplier architecture.

[10] with Wallace/Dadda [16], [17] partial product reduction, a very high speed multiplier architecture is possible. This multiplier architecture has therefore been chosen as a baseline for comparing the performance characteristics of the hybrid radix-4/radix-8 multiplier architecture presented here.

The process of multiplying two numbers, $A \cdot B$, requires the generation of a set of partial products followed by the summation of these partial products. In radix-4 encoding, all partial products may be generated through simple shifts and negation. However, with radix-8 encoding all partial product terms may be generated with simple shifts and negation with the exception of $\pm 3B$. The $3B$ terms require an additional high speed adder. The delay overhead of this additional adder stage is a major disadvantage of radix-8 encoding. This increased delay overhead is addressed by the hybrid radix-4/radix-8 architecture.

The proposed hybrid radix-4/radix-8 multiplier architecture uses a combination of radix-4 and radix-8 encoding in order to mitigate the delay penalty associated with the generation of $3B$ for the radix-8 architecture. In this manner the hybrid radix-4/radix-8 multiplier combines the speed advantage of the radix-4 multiplier, by initiating the partial product reduction immediately after the radix-4 encoding, with the reduced power dissipation of the radix-8 multiplier, by reducing the overall number of partial products.

In a radix-8 architecture, the multiplication process is serially dependent upon the time required to generate $3B$: while $3B$ is being generated by a high speed adder, no partial product reduction can take place. This requirement to generate $3B$ leads to a significant delay penalty, on the order of 15–20%, as compared with a radix-4 architecture [13]. Alternatively, one could represent $3B$ in partially redundant form, reducing the time required to generate $3B$ but increasing the number of bits which need to be summed in the reduction tree [14].

In the hybrid radix-4/radix-8 architecture, a subset of the partial products are generated using radix-4 modified Booth encoding. Reduction begins on these radix-4 partial products while $3B$ is simultaneously being generated by a high speed adder. Upon generating $3B$, the remaining partial products are generated using radix-8 encoding, and these partial products are subsequently included within the reduction tree. In this manner, some reduction of the partial products takes place while the high speed adder is generating $3B$; therefore, less of a delay penalty is incurred. Utilizing radix-8 encoding for many of the partial products reduces the total number of partial products, thereby reducing the power dissipation required to sum the partial products. A diagram of the hybrid radix-4/radix-8 architecture is shown in Fig. 2.

A delay analysis of the multiplier components demonstrates that three reduction steps may take place during the generation of $3B$ for both the 32 × 32-b multiplier and the 64 × 64-b multiplier. The number of bits m of multiplier A which are used for radix-8 encoding (and hence the number of radix-8 and radix-4 partial products) may be determined by minimizing the number of reduction stages, given that the radix-8 partial products will not be available until after the third reduction stage, under the constraint that (m mod 3) = 0. The optimum value of m is $m = 18$ for a 32 × 32-b hybrid multiplier and is $m = 45$ for a 64 × 64-b hybrid multiplier. Note that, consistent with the 1-b overlap between adjacent bit fields in Booth encoding, there is a 1-b overlap between the bit fields of A utilized by the radix-4 and the radix-8 encoders, and the constraint (m mod 3) = 0 assures that all radix-8 encoders operate on full 4-b sets of data.

For this 64 × 64-b hybrid radix-4/radix-8 implementation, the nine required reduction steps are as follows: $11 \rightarrow 9 \rightarrow 6 \rightarrow (4+15) \rightarrow 13 \rightarrow 9 \rightarrow 6 \rightarrow 4 \rightarrow 3 \rightarrow 2$. For comparison, a 64 × 64-b radix-4 multiplier requires eight reduction steps, while a 64 × 64-b radix-8 multiplier requires only seven reduction stages [13]. Note that by using the one's complement plus the carry-in to form the two's complement, the number of bits at the start of the reduction process is 1 b greater than the number of partial products. This additional bit is the carry-in of the highest order partial product. Thus, the hybrid reduction begins at 11 b although there are only ten partial products.

With a 32 × 32-b multiplier, seven steps are required for partial product reduction in a hybrid radix-4/radix-8 implementation, as compared with six steps for a radix-4 implementation and five steps for a radix-8 implementation. The reduction steps for the 32 × 32-b hybrid radix-4/radix-8 implementation are$8 \rightarrow 6 \rightarrow 4 \rightarrow (3+6) \rightarrow 6 \rightarrow 4 \rightarrow 3 \rightarrow 2$.

It is important to note that the delay penalty associated with the generation of $3B$ can not be entirely mitigated using this hybrid approach. An additional delay penalty is incurred since all of the partial products are not immediately available when the reduction process is initiated. The more data available in parallel to the reduction tree, the more time efficient the reduction steps become. As the radix-8 partial products are not available until three reduction steps have been completed, fewer bits in parallel are available at the start of the reduction process. Thus, the reduction process is not as time efficient, requiring additional reduction steps as compared with an architecture in which all of the partial products are available

simultaneously when the reduction process begins. In essence, the parallelism of the reduction tree is reduced in exchange for operating the reduction tree in parallel with the $3B$ adder. It should also be noted that the relative delay between the reduction steps and the $3B$ adder is logic family dependent, and significant changes in this relative delay may reduce the advantages of this hybrid structure.

III. Multiplier Components

In this section the various functional components from which the multipliers are constructed are summarized. A detailed discussion of the circuits used to implement these multipliers is found in [9].

An encoding/decoding structure similar to that found in [12] is used to perform the Booth encoding. The encoding circuitry generates the control signals that select the correct partial product. Due to the high fanout at the output of these encoders, low power tapered buffers are included in the signal path [18]. These control signals are passed to the decoding/selection circuitry, which is a pass-gate style multiplexer with a conditional inversion to generate the one's complement.

The full adder cell in the Wallace/Dadda reduction tree utilizes a standard 28-transistor circuit implementation [19]. A multilevel carry lookahead adder [11], [20] is used for the final high speed addition and for the adder generating $3B$ in the architectures using radix-8 encoding. This adder architecture provides a satisfactory tradeoff between the propagation delay and the power dissipation [21], [22]. The carry lookahead adder provides the necessary speed to generate $3B$ in parallel with the three reduction steps, as required in the hybrid radix-4/radix-8 architecture.

Transistors are sized considering both speed and power dissipation characteristics. The transistors in the speed limiting paths, such as those in the high speed adders and Booth decoders, are sized to minimize delay. The transistors in the reduction tree are sized to reduce power dissipation.

IV. Performance

The propagation delay, transistor count, and power dissipation characteristics of the 32 × 32-b and the 64 × 64-b multipliers are presented in this section. In Section IV-A, the delay of the proposed hybrid radix-4/radix-8 multiplier architecture is compared with the delay of the radix-4 and radix-8 multiplier architectures. In Section IV-B, the number of transistors required to implement each of the multipliers is presented. The power dissipation characteristics of the three architectures are compared in Section IV-C. The power dissipation characteristics of the three architectures after the power supply voltages are scaled such that the same delay is achieved for each architecture are compared in Section IV-D. Note that the interconnect capacitance within the reduction tree is estimated rather than extracted from physical layout.

A. Delay Analysis

The 32 × 32-b and 64 × 64-b multipliers have been simulated in SPICE assuming a 5 V, 1.2 μm CMOS process technology. The output of each circuit was loaded with the next circuit stage and, when appropriate, with an estimated interconnect loading. The worst case delay values, derived from SPICE, are shown in Table I. The radix-4 multiplier exhibits the least delay, and the radix-8 multiplier exhibits the most delay. The delay of the hybrid radix-4/radix-8 multiplier falls between those of the radix-4 and radix-8 multipliers.

B. Transistor Count

The number of transistors required to implement a multiplier architecture can provide a metric by which to judge the relative area requirements and power dissipation of the different architectures, assuming that switching probabilities and sizing methodologies for the transistors are relatively constant across architectures, as is the case in these multipliers. The transistor count for the 32 × 32-b and 64 × 64-b implementations of each of the three architectures are compared in Table II. The radix-8 multipliers require the fewest transistors, while the radix-4 multipliers require the most transistors. The number of transistors required to implement the hybrid radix-4/radix-8 multipliers fall between those of the radix-4 and radix-8 multipliers.

TABLE I
Technology Dependent Delay of Multiplier Architectures (1.2 μm, 5 V CMOS)

		Radix-4	Hybrid Radix-4/8	Radix-8
64 x 64 bit	Partial Product Generation	3.3 ns	3.3 ns	9.2 ns
	Reduction	13.9 ns	16.3 ns	12.2 ns
	Final High Speed Addition	9.0 ns	9.0 ns	9.0 ns
	Total	26.2 ns	28.6 ns	30.4 ns
32 x 32 bit	Partial Product Generation	3.3 ns	3.3 ns	7.4 ns
	Reduction	10.4 ns	12.2 ns	8.7 ns
	Final High Speed Addition	7.9 ns	7.9 ns	7.9 ns
	Total	21.6 ns	23.4 ns	24.0 ns

TABLE II
Number of Transistors for Each Multiplier Implementation

Bit Width	Radix-4	Hybrid Radix-4/8	Radix-8
32 x 32	28,522	25,678	23,542
64 x 64	108,038	90,210	83,412

TABLE III
Total Multiplier Power Dissipation. 5 V, 1.2 μm CMOS, 10 MHz

	Power Dissipation (mW)		
Bits	Radix-4	Hybrid Radix-4/8	Radix-8
64 x 64	99.1	85.9	81.3
32 x 32	26.1	23.9	22.5

TABLE IV
Breakdown of Power Dissipation for a 64 × 64-b Multiplier. 5 V, 1.2 μm CMOS, 10 MHz

	Power Dissipation (mW)		
Function	Radix-4	Hybrid Radix-4/8	Radix-8
Booth Encoding	40.8	39.0	38.6
Wallace Tree	53.0	40.3	36.1
$3B$ Adder	-	2.2	2.2
Final Adder	4.4	4.4	4.4

C. Power Dissipation

The average power dissipation of each circuit operating at 10 MHz is determined from SPICE using the Kang power meter [23]. The power dissipation of each component is averaged over 100 input vectors. The input vectors were pseudorandomly generated with activity factors made to conform to those measured from one million vector Verilog-XL functional simulations of the multipliers.

The total power dissipated by each multiplier architecture is shown in Table III. The breakdown of power dissipation by functional block for the 64 × 64-b multiplier is shown in Table IV.

As described previously and shown in Tables III and IV, a radix-8 multiplier dissipates less power than a radix-4 multiplier. The hybrid

TABLE V
COMPARISON OF VOLTAGE SCALED PERFORMANCE OF DIFFERENT MULTIPLIER ARCHITECTURES

		Radix-4	Hybrid Radix-4/Radix-8	Radix-8
64 x 64-bit	V_{DD} for 30.4 ns delay	4.53 V	4.79 V	5.00 V
	Power dissipation	81.3 mW	78.8 mW	81.3 mW
32 x 32 bit	V_{DD} for 24.0 ns delay	4.65 V	4.91 V	5.00 V
	Power dissipation	22.5 mW	23.1 mW	24.0 mW

radix-4/radix-8 architecture dissipates power at a level between that of the radix-4 and radix-8 multipliers. Thus, the hybrid radix-4/radix-8 multiplier architecture is a useful architecture for those applications which require low power while operating at speeds greater than that of a full radix-8 multiplier. Radix-8 multiplication is appropriate for those ultra-low power systems in which added delay can be tolerated.

D. Voltage Scaled Performance

Voltage scaling [24], reducing the power supply voltage, may be applied to the radix-4 and hybrid multipliers to increase the delay to that of the radix-8 multipliers, while simultaneously reducing the power dissipation of these multipliers. The delay of the multipliers is proportional to the power supply, V_{DD}, as shown in (1), where V_T represents the average magnitude of the threshold voltages, and the power dissipation is proportional to the square of the power supply voltage, as shown in (2) [25].

$$\text{Delay} \propto \frac{V_{DD}}{(V_{DD} - V_T)^2} \tag{1}$$

$$\text{Power} \propto (V_{DD})^2. \tag{2}$$

The power dissipation of the radix-4, hybrid radix-4/radix-8, and radix-8 multipliers after voltage scaling is compared in Table V. Note that the scaled voltage levels are referenced to the radix-8 multiplier operating at 5 V. For shorter bit widths such as exemplified by a 32 × 32-b multiplier, the delay and power dissipation overhead due to the additional $3B$ adder and more complex encoding is not outweighed by the reduction in delay and power dissipation associated with the partial product summation. In this case, the simpler radix-4 encoded multiplier provides the lowest power dissipation at a given delay.

However at higher bit widths, as exemplified by the 64 × 64-b multipliers, the radix-4 and radix-8 multipliers dissipate approximately equivalent power at a given delay, whereas the hybrid radix-4/radix-8 multiplier dissipates less power than either the radix-4 or the radix-8 multiplier.

V. CONCLUSIONS

A hybrid radix-4/radix-8 multiplier architecture is presented in this paper that is both low power and high speed; this architecture provides a tradeoff between the high speed of a radix-4 multiplier architecture and the low power dissipation of a radix-8 multiplier architecture. In this hybrid radix-4/radix-8 multiplier architecture, the performance bottleneck of a radix-8 multiplier (the generation of $3B$ for the radix-8 partial product generation) is performed in parallel with the reduction of the radix-4 partial products rather than serially, as in a radix-8 multiplier. This strategy minimizes a portion of the delay penalty incurred by the radix-8 multiplier in generating $3B$. This hybrid radix-4/radix-8 multiplier architecture dissipates 13% less power in a 64 × 64-b multiplier with only a 9% increase in delay, as compared to a radix-4 implementation. When the supply voltage of the 64 × 64-b multipliers is scaled such that all three architectures exhibit the same delay, the hybrid radix-4/radix-8 multiplier dissipates the least power. The hybrid radix-4/radix-8 architecture therefore provides a tradeoff between high speed and low power for application to those systems which require both high speed and low power signed multiplication.

REFERENCES

[1] S. R. Powell and P. M. Chau, "Estimating power dissipation of VLSI signal processing chips: The PFA technique," *VLSI Signal Processing IV*. New York: IEEE Press, 1990, ch. 24.

[2] R. Sharma, A. D. Lopez, J. A. Michejda, S. J. Hillenius, J. M. Andrews, and A. J. Studwell, "A 6.75 ns 16 × 16-bit multiplier in single-level-metal," *IEEE J. Solid-State Circuits*, vol. 24, pp. 922–927, Aug. 1989.

[3] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu, "A 3.8 ns CMOS 16×16-b multiplier using complementary pass-transistor logic," *IEEE J. Solid-State Circuits*, vol. 25, pp. 388–395, Apr. 1990.

[4] M. Nagamatsu, S. Tanaka, J. Mori, K. Hirano, T. Noguchi, and K. Hatanaka, "A 15 ns 32×32b CMOS multiplier with an improved parallel structure," *IEEE J. Solid-State Circuits*, vol. 25, pp. 494–497, Apr. 1990.

[5] J. Mori, M. Nagamatsu, M. Hirano, S. Tanaka, M. Noda, Y. Yoyoshima, K. Hashimoto, H. Hayashida, and K. Maeguchi, "A 10 ns 54 × 54 b parallel structured full array multiplier with 0.5 μm CMOS technology," *IEEE J. Solid-State Circuits*, vol. 26, pp. 600–606, Apr. 1991.

[6] G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A 54 × 54-b regularly structured tree multiplier," *IEEE J. Solid-State Circuits*, vol. 27, pp. 1229–1236, Sept. 1992.

[7] F. Lu and H. Samueli, "A 200-MHz CMOS pipelined multiplier-accumulator using quasi-domino full-adder cell design," *IEEE J. Solid-State Circuits*, vol. 28, pp. 123–132, Feb. 1993.

[8] D. Somasekhar and V. Visvanathan, "A 230-MHz half-bit level pipelined multiplier using true single-phase clocking," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 415–422, Dec. 1993.

[9] B. S. Cherkauer, "CMOS-based architectural and circuit design techniques for application to high speed, low power multiplication," Ph.D. disssertation, Univ. Rochester, Rochester, NY, Apr. 1995.

[10] A. D. Booth, "A signed binary multiplication technique," *Quart. J. Mech. Appl. Math.*, vol. 4, no. 2, pp. 236–240, June 1951.

[11] O. L. MacSorley, "High-speed arithmetic in binary computers," *Proc. IRE*, vol. 49, pp. 67–91, Jan. 1961.

[12] P. J. Song and G. De Micheli, "Circuit and architecture trade-offs for high-speed multiplication," *IEEE J. Solid-State Circuits*, vol. 26, pp. 1184–1198, Sept. 1991.

[13] B. Millar, P. E. Madrid, and E. E. Swartzlander, Jr., "A fast hybrid multiplier combining Booth and Wallace/Dadda algorithms," in *Proc. 35th IEEE Midwest Symp. Circuits Syst.*, Aug. 1992, pp. 158–165.

[14] G. Bewick and M. J. Flynn, "Binary multiplication using partially redundant multiples," Stanford Univ., CSL-TR-92-528, June 1992.

[15] R. K. Montoye, P. W. Cook, E. Hokenek, and R. P. Havreluk, "An 18 ns 56-bit multiple-adder circuit," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 1990, pp. 46–47.

[16] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Elect. Comput.*, vol. EC-13, pp. 14–17, Feb. 1964.

[17] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, no. 5, pp. 349–356, May 1965.

[18] B. S. Cherkauer and E. G. Friedman, "A unified design methodology for CMOS tapered buffers," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 99–111, Mar. 1995.

[19] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*, 2nd ed. Reading, MA: Addison-Wesley, 1993, pp. 515–517.

[20] K. Hwang, *Computer Arithmetic: Principles, Architecture, and Design*. New York: Wiley, 1979, pp. 84–91.

[21] T. K. Callaway and E. E. Swartzlander, Jr., "Estimating the power consumption of CMOS adders," in *Proc. 11th IEEE Symp. Comp. Arith.*, June/July 1993, pp. 210–216.

[22] C. Nagendra, R. M. Owens, and M. J. Irwin, "Power-delay characteristics of CMOS adders," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 377–381, Sept. 1994.

[23] S. M. Kang, "Accurate simulation of power dissipation in VLSI circuits," *IEEE J. Solid-State Circuits*, vol. SC-21, pp. 889–891, Oct. 1986.

[24] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Reading, MA: Addison-Wesley, 1990.

[25] A. P. Chandrakasan, S. Sheng, and R. W. Broderson, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, pp. 473–483, Apr. 1992.

On Fast Running Max-Min Filtering

Dinu Coltuc and Ioannis Pitas

***Abstract*—The problem of fast running max/min filters for arbitrary-size windows is addressed. The size of the filter window is increased to the least power of two greater than the given size and, the input sequence is expanded. The running max/min computation uses a fast algorithm for power of two window sizes. The computational complexity (comparisons per sample) of the proposed algorithm is very close to $\log_2 n$, where n is the size of the given window. A flexible hardware implementation for n ranging between two consecutive powers of two is discussed.**

***Index Terms*— Algorithm optimization, fast algorithms, Max-Mix filtering.**

I. Introduction

Max/min filters are widely used in signal/image processing [2]–[4]. Let $\{x_i, i = 0, \cdots, N\}$ be a sequence of samples. The problem of running max/min filtering within a window of size n is to determine a sequence $\{y_i\}$, where y_i is either the maximum or the minimum of n consecutive samples, $x_i, x_{i+1}, \cdots, x_{i+n-1}$. The computational complexity of max/min filters, i.e., the number of comparisons per sample depends on n, the size of the filter window. When n is a power of two, fast algorithms of $\log_2 n$ comparisons per sample complexity have been developed. They are based on a factorization of the computation by recursively dividing the sequence of n samples in subsequences of $n/2$ samples and so on until subsequences of size 2 are obtained. The requirement that n be power of two is crucial for assuring a perfect decomposition in $\log_2 n$ steps. The flowchart of such an algorithm is illustrated in Fig. 1 [1]. This brief deals with max/min filters for arbitrary n and it proposes a method that takes advantage of the fast power of two algorithms.

II. Arbitrary n-Size Window Filtering

Our approach consists of: 1) the expansion of the input sequence and the elimination of the corresponding extra output values; 2) the computation of the running filter.

A. Sequence Expansion

Let k be the least integer greater than or equal to $\log_2 n, 2^{k-1} < n \leq 2^k$. Let $p = 2^k - n, 0 \leq p < n$. In order to preserve the number n of original samples within the 2^k size sliding window, whenever a dummy sample z_j is inserted into the position i, then a dummy sample z_{j+p} has also to be inserted into the position $i + 2^k$. This insertion periodicity of the dummy samples assures the expansion of the entire input sequence, once the first group of n samples has been expanded. Let ϕ be any increasing mapping, ϕ: $I \rightarrow K$, where I and K are the set of integers $\{0, 1, \cdots, n-1\}$ and $\{0, 1, \cdots, 2^k - 1\}$, respectively. The expansion procedure for the first group of n samples defines for each position i the new position $\phi(i)$. The increasing property of ϕ preserves the order of the samples in the expanded sequence.

Manuscript received January 13, 1995; revised June 2, 1995. This paper was recommended by Associate Editor T. Hinamoto.

D. Coltuc is with the Research Institute for Electrical Engineering, Splaiul Unirii 313, Bucuresti 74204, Romania.

I. Pitas is with the Department of Informatics, University of Thessaloniki, Thessaloniki 540 06, Greece.

Publisher Item Identifier S 1057-7130(97)02741-9.

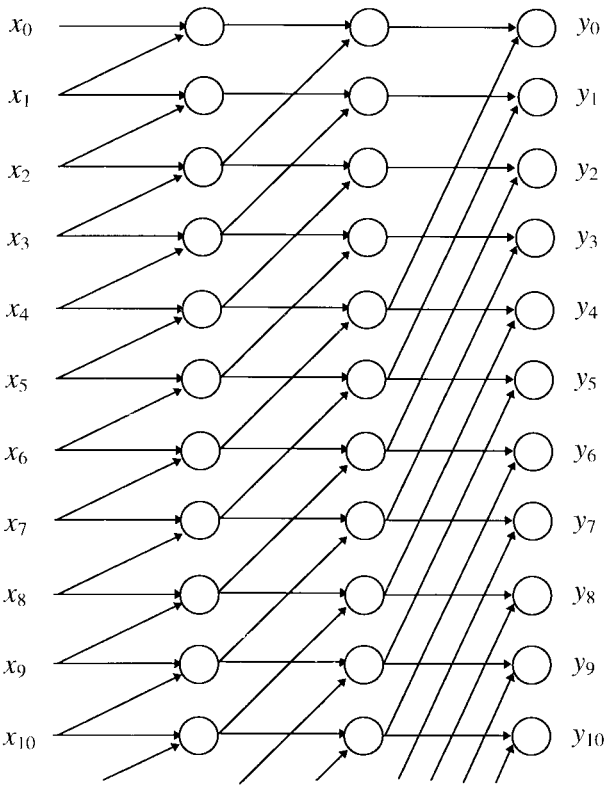


Fig. 1. Max/min filter flowchart for $n = 2^3$ window size.

The expansion of the entire input sequence can be further described by an integer mapping $\psi, \psi(i) = \lfloor i/n \rfloor 2^k + \phi(i \bmod n)$, where $\lfloor m \rfloor$ denotes the greatest integer less than or equal to m. ψ is an increasing mapping, too. Thus, the expanded sequence of the input $\{x_i\}$ is generated by the following procedure: each sample x_i is placed in position $\psi(i)$; between samples $x_{\psi(i)}$ and $x_{\psi(i+1)}$ a number of $\psi(i+1) - \psi(i) - 1$ dummy samples are inserted, and $\psi(0)$ dummy samples are inserted in front of $x_{\psi(0)}$.

A formal proof can be given for the validity of the expansion procedure. Let J be the set $\phi(I)$. The set J consists of n values and the set difference $K \backslash J$ consists of p values. By construction, the sample placed at position j belongs to the initial sequence provided that $j \bmod 2^k \in J$. Let us further consider a group of 2^k consecutive samples of the expanded sequence and let j be the index of its first sample. Their indexes are $j + q$, where $0 \leq q < 2^k$. $(j + q) \bmod 2^k$ takes distinct values for distinct values of q, i.e., $0 \cdots, 2^k - 1$. The set $\{(j+q) \bmod 2^k\}$ is identical with the set K. As required, exactly n samples belong to the initial sequence and p samples are dummy ones. Since j has been taken arbitrarily, the property holds for each group of 2^k consecutive samples.

The expanded sequence is determined by the mapping ϕ. The total number of possible mappings is $(n + p)!/n!$. For example, when ϕ is the identity application on $I, \phi(i) = i$, groups of p dummy samples are inserted between groups of n original samples: $x_0, x_1, \cdots, x_{n-1}, z_0, \cdots, z_{p-1}, x_n, x_{n+1}, \cdots, x_{2n-1}, z_p, \cdots, .$

The insertion of dummy samples should not alter the extreme values within any window. Thus, when maximum has to be computed, a straightforward possibility is to consider all dummy samples equal to a small value (smaller than any signal sample). Alternatively, a

large value is used for minimum computation. Another possibility that holds both for maximum and for minimum is to assign to each dummy sample the value of an original sample within the same window.

The total number of samples of the expanded sequence is $\psi(N)+1$. The output sequence will have $\psi(N)+1$ results, which means $\psi(N)-N$ extra output samples. Let us consider a window starting with a dummy sample z_i. When the window moves one position, z_i is discarded and a new sample is added. Due to the mapping periodicity, the new sample z_{i+p} is a dummy sample as well. Since dummy samples do not influence the computation, the result for both windows is identical. One out of the two samples has to be discarded, e.g., the result produced within the window starting with a dummy sample.

B. Computation

Flowcharts composed of branches and circles, as the one shown in Fig. 1, are a convenient mean to describe the fast max/min algorithms for 2^k size window filters. The flow of the operands is represented by branches. The circles represent places where the comparisons take place. Their output is either the maximum (max filters) or the minimum (min filters) between the two operands of the incoming branches. For a filter window of size 2^k, the flowchart is a cascade of k stages (in Fig. 1, $k=3$). Let stages be numbered from 1 (the input stage) to k (the output stage), then the comparisons of the ith stage are comparisons between operands placed at distance 2^{i-1}. The change of the order in which the stages are placed, yields to a flowchart of an algorithm that performs the same filtering. Thus, there are $k!$ possible algorithms for a filter with a window of size 2^k.

The computational complexity of the fast algorithm when $p=0$ is k comparisons per sample. When $p>0$, the computational complexity of the proposed algorithm increases to $k+kp/n$ comparisons per sample (only n results out of $n+p$ are preserved). However, when dummy samples are inserted, the overall performance of the algorithm can be improved since some comparisons can be eliminated (either the result is known in advance, or the result is not useful). These comparisons are: comparisons between two dummy samples, comparisons between a dummy sample and a signal sample and the final comparisons for the extra-output samples. Each input sample appears as operand in two comparisons whose results are further operands in four other comparisons and so on. When the sample is a dummy one, the two corresponding comparisons are eliminated. Furthermore, when two dummy samples are operands of the same comparison, they propagate a dummy result which normally should be an operand for two further comparisons, which, in this case, can be eliminated. Thus, five comparisons are eliminated instead of four that are taken out when the dummies do not compare with each other. From a computational complexity point of view, it occurs that the more dummy variables will compare to each other, the more computationally efficient the scheme will be.

The lower bound of the computational complexity of the max/min filters depends on ϕ, the insertion mapping, and on the selected 2^k algorithm. First we analyze the case when two dummy samples have to be inserted. Let 2^j be the distance of the samples that are compared together in the input stage. The two dummies will propagate one dummy to the next stage, if and only if they are placed at a distance equal to 2^j. By the periodicity of the insertion procedure we observe that, if the distance from which the operands are compared is $2^j=2^{k-1}$, both dummies will propagate to the next stage. Similarly, when 2^s dummy samples have to be inserted, all dummies will propagate to the second stage if they are inserted at equal distances of 2^{k-s}, provided that the operands placed at 2^{k-1} distance are compared in the first stage. Furthermore, the inserted dummies will propagate from one stage to another, until they will

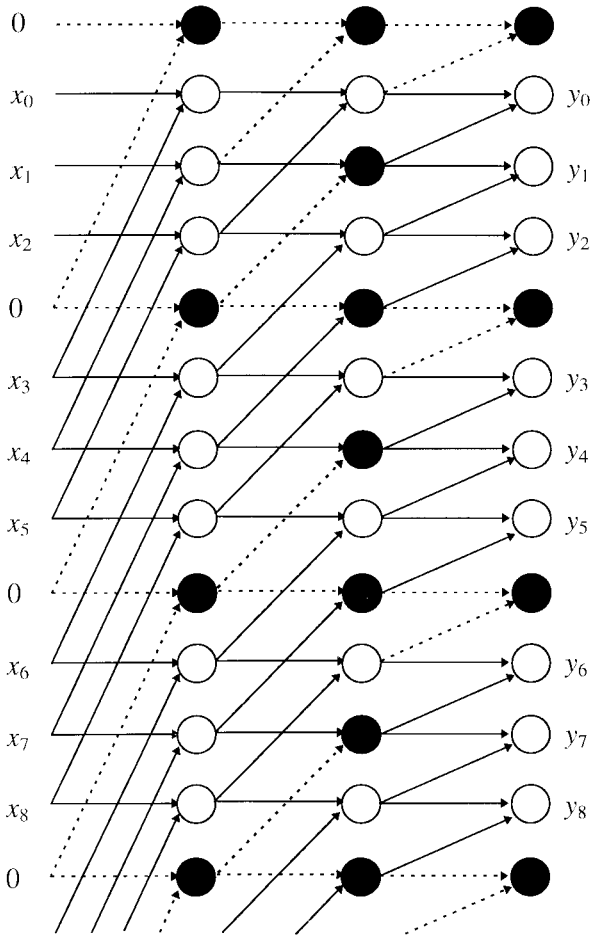


Fig. 2. Max/min filter flowchart derivation for $n=6$ window size. Unnecessary comparisons and operands are represented by filled circles and dashed branches, respectively.

reach the stage where the distance between the operands to be compared together is less than 2^{k-s}. The best performance will be achieved by using an algorithm where, in the stage i, the samples (operands) placed at 2^{k-i} distance are compared together ($i=1$ for the input stage and $i=k$ for the final one). Thus, the 2^s dummies will propagate through s stages. Besides the $s2^s$ comparisons that are eliminated due to dummy propagation through the flowchart, two more comparisons for each inserted dummy sample are also eliminated in the $s+1$ stage (comparisons between a dummy and a real operand). Since $s<k-1$, the propagated dummies can reach only the stage $k-2$. No dummy appears as operand in stage k and thus, the final comparison for each input dummy is eliminated (its result is discarded). The general case of arbitrary p inserted dummy samples immediately follows by considering the binary representation of p, i.e., $p=\Sigma_{j=0}^{k-1}\ p_j2^j, p_j=\{0,1\}$. When $p_j=1$, the group of 2^j dummies propagates through j stages, if they are placed at 2^{k-j} distance. The distance requirement for the p dummies holds if the dummies are inserted in the places that correspond to the bit reversed values of $\{0,\cdots,p-1\}$. If the binary representation of s by using k bits is $s_{k-1}s_{k-2}\cdots s_1s_0, (s=\Sigma_{j=0}^{k-1}\ s_j2^j)$, then the bit-reversed of s, denoted by $br(s)$, is the binary number $s_0s_1\cdots s_{k-2}s_{k-1}, (br(s)=\Sigma_{j=0}^{k-1}\ s_{k-1-j}2^j)$. The closed form of the mapping is

$$\phi(i)=i+\sum_{j=0}^{i} f(j), f(j)=\begin{cases}1 & br(j)\le p\\ 0 & \text{otherwise.}\end{cases} \tag{1}$$

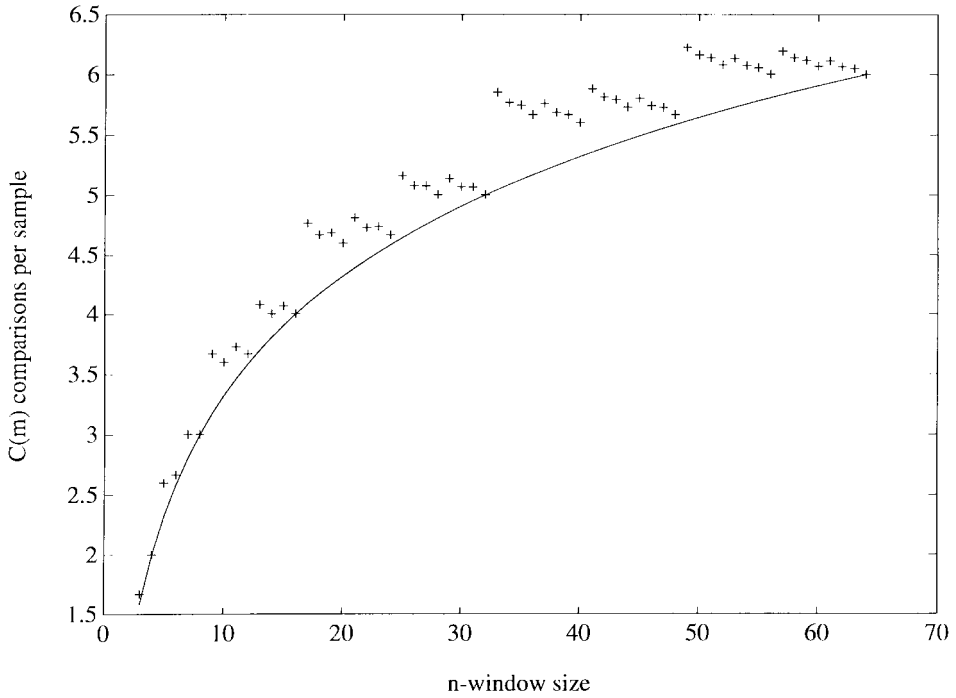


Fig. 3. Optimal algorithm performance: computational complexity $C(n)$ comparisons per sample (plus sign) with respect to the window size n compared with $\log_2 n$ (solid line) for n ranges [3–64].

The flowchart for a window of size 6 obtained by using the mapping ϕ, (1), is given in Fig. 2. The inserted dummies are denoted by 0. The comparisons that are not performed are represented by filled circles and dashed branches correspond to the unnecessary operands.

The algorithm derived by using the mapping (1) and the specified 2^k-size window scheme is optimal with respect to the computational complexity. With the above mentioned notations, its computational complexity $C(n)$ (comparisons per sample) is given by

$$C(n) = k - \frac{\sum_{j=0}^{k-1}(3+j-k)p_j 2^j}{n}. \tag{2}$$

When $p \geq 2^{k-2}$ (i.e., $2^{k-1} < n \leq 2^{k-1} + 2^{k-2}$), the fraction of (2) is positive and $C(n) \leq k$ $(C(n) \leq \lceil log_2 n \rceil)$. $C(n)$ is very close to $\log_2 n$ as can be seen in Fig. 3, where $C(n)$ (plus sign) and $\log_2 n$ are plotted for $n = [3, \cdots, 64]$. When no comparisons are eliminated, the proposed algorithms perform in $k+kp/n$ comparisons per sample. Regardless the insertion mapping and the fast 2^k scheme, this upper bound can be slightly improved to $k + (k-1)p/n$, or $k + (k-3)p/n$, by elimination (for each inserted dummy sample) of the final comparison or of the final plus the two comparisons of the first stage, respectively.

III. Flexible Implementation

The computational structures that yield to the lower limit $C(n)$ are of interest for software implementations. When hardware implementations have to be considered, the irregularity of these structures becomes a major drawback since, the position-dependent processing of the samples results in hardware complexity. Besides, even for slightly different window sizes, very different structures are needed. In the sequel, we investigate a particular mapping more suitable for hardware implementation. Let ϕ: $I \rightarrow K$ be the mapping defined as

$$\phi(i) = \begin{cases} 2i & \text{if } i < p \\ i + p & \text{otherwise.} \end{cases} \tag{3}$$

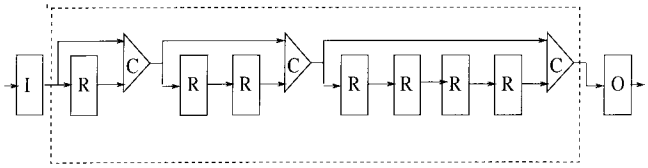


Fig. 4. Pipeline architecture for max/min filters ($5 \leq n \leq 8$).

Each dummy sample z_i takes the value of its previous sample x_i. The expanded sequence is given by

$$x_0, z_0, x_1, z_1, \cdots, x_{p-1}, z_{p-1}, x_p, x_{p+1}$$
$$\cdots, x_{n-1}, x_n, z_n, \cdots.$$

A hardware implementation for the flowchart of Fig. 1 was proposed in [1]. The architecture shown in Fig. 4, is a pipeline. At each clock cycle, a new sample is loaded into the pipeline from the input register I and a result is loaded into the output register O. Blocks denoted by "C" compute the extreme value of the entries. A "C" block is a multiplexer driven by a comparator. If the expanded sequence were already available, the pipeline would work without any modification. However, extra hardware must expand the input sequence and discard certain output samples. Besides, a severe degradation in performance appears since only n results are produced in $n + p$ clock cycles. We shall overcome these drawbacks by using an appropriate timing command, such that no degradation in performance occurs. Our idea is to insert hidden clock cycles in the pipeline, keeping the input/output synchronization. The problems appear when a dummy sample has to be inserted. Each dummy sample is a copy of the previous sample. If the comparators are two times faster, two cycles (instead of one) are computed, by processing the window starting with a true sample and the next one. This means that the sample x_i is loaded twice, is forwarded into the pipeline and only one output sample is preserved. When no dummy sample is inserted, the pipeline and the I/O registers have the same timing. Different clock signals are necessary for the command of the pipeline registers

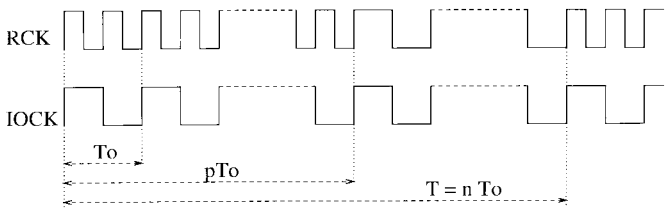


Fig. 5. Clock signals: RCK pipeline clock, IOCK input/output clock.

R and for the I/O registers. The clock signals are shown in Fig. 5. The pipeline clock (RCK) is a periodic signal $T = nT_0$ where T_0 is the period of the I/O clock. At each period there are $n + p$ clock ticks, the first $2p$ ticks being twice shorter than the next $n - p$ ones. The processing is performed for different sizes of the window only by changing the pipeline command clock.

IV. Concluding Remarks

Max/min filters within any arbitrary n window size, $2^{k-1} < n \leq 2^k$, are computed by using fast 2^k window size structures operating on an expanded input sequence. Appropriate mappings assures that each 2^k window contains exactly n original samples. The derived algorithms depend both on the expansion mapping and on the selected fast power of two window size algorithm. For each n, the existence of a fast algorithm of very close to $\log_2 n$ comparisons per sample performance is proven. When $2^{k-1} < n \leq 2^{k-1} + 2^{k-2}$ the algorithm performs in less than $\lceil \log_2 n \rceil$ comparisons per sample. Another algorithm suitable for a flexible hardware implementation for all $n, 2^{k-1} < n \leq 2^k$, is presented.

References

[1] I. Pitas, "Fast algorithms for running ordering and max/min calculation," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 795–804, 1989.
[2] Y. Nakagawa and A. Rosenfeld, "A note on the use of local min and max operations in digital picture processing," *IEEE Trans. Syst. Man, Cybern*, vol. SMC-8, pp. 632–635, Aug. 1978.
[3] M. Werman and S. Peleg, "Min-max filters in texture analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, pp. 730–733, 1986.
[4] P. W. Verbeek, H. A. Vrooman, and L. J. Vliet, "Low-level image processing by max-min filters," *Signal Processing*, vol. 15, no. 3, pp. 249–258, Oct. 1988.

Multiple Transform Algorithms for Time-Varying Signal Representation

Victor DeBrunner, Wei Lou, and Jonathan Thripuraneni

***Abstract*— We develop two new classes of multiple transform algorithms for representing time-varying signals. The algorithms use either a gradient search or a recursive greedy search over partial sets of several different basis functions to capture different signal characteristics. We see that our proposed algorithms require fewer than one-half the computations required by the previous methods and represent the signal with less error.**

***Index Terms*—Nonorthogonal signal representations, transform coding.**

I. Introduction

Transform-based analysis/synthesis models have been widely used for nonstationary signal representation, including speech. A sinusoidal model for speech signal representation was proposed in [1]. Other transform-based models for speech signal representation can be seen in [2] and [3]. It has been shown that the use of a partial set of basis functions from one orthogonal set is insufficient to efficiently represent nonstationary signals in low rate coding applications [4]. Therefore, multitransform algorithms have been introduced [5]–[8]. Different transforms have different properties which can effectively match various aspects of the nonstationary signals. The algorithms described in [5]–[8] are based on a cascade structure, where the dominant projections (DP) are selected from one transform before examining the second transform. The selection of the DP is critical. Two methods exist: HPF (harmonics of pitch frequencies) [6] and SP (spectral peak) [8].

In this brief, we propose a new implementation structure, the parallel structure, and a new DP strategy, the look-back recursive residual projection (LBRRP) algorithm based on ideas in [9] and [10], to develop new multitransform algorithms. We show that the proposed parallel structure yields algorithms that require fewer computations than those based on the cascade structure while improving representation performance. We also show that the LBRRP algorithms have superior representation performance to the parallel structure algorithms except when very low numbers of DP are selected.

II. Existing Multitransform Algorithms

The Gauss-Seidel algorithm proposed in [6] and [7] assumes that the time-varying signal is the superposition of a narrow-band signal and a broad-band signal. A partial set of basis functions is first used to model the narrow-band portion, and then a partial set of basis functions from a different transform are used to represent the residual created by removing the narrow-band model. The block algorithm works in a cascade fashion, and the selections in each transform domain are mutually dependent. This means that both the projection values and the selected DP in each domain depend on each other. An iterative technique to select the DP and their projections was developed in [6] and [7]. The technique assumes independence of the two transform domains. The total represenation error is minimized

Manuscript received August 23, 1995; revised November 6, 1996. This work was supported in part by the National Science Foundation under Grant BCS-9308925. This paper was recommended by Associate Editor K. K. Parhi.

The authors are with the School of Electrical and Computer Engineering, University of Oklahoma, Norman, OK 73019 USA.

Publisher Item Identifier S 1057-7130(97)03643-4.

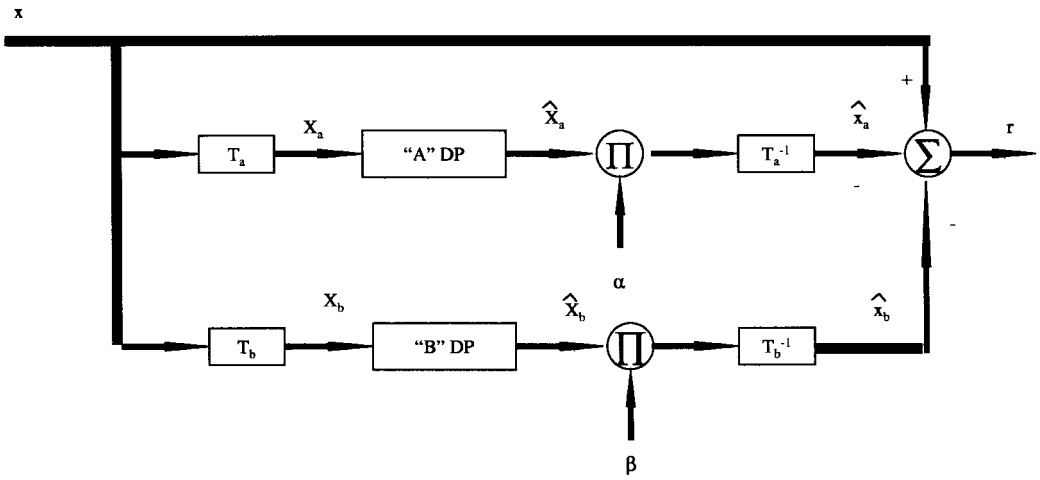


Fig. 1. Parallel implementation structure.

using a Gauss-Seidel search technique. The DP selection process used is the HPF method.

The gradient method [8] uses the same processing scheme as the Gauss-Seidel method. However, the SP DP selection process is used, and the convergence determination is different. After numerical convergence is achieved, the smallest weights in both domains are discarded. Then, the remaining large weights are retained and used as the initial values in a new numerical solution of the minimization problem. In [11], a slight modification of this procedure is used: not only are the smallest projections monitored for convergence, but also the smallest gradients are monitored. This method is more computationally intense than the Gauss-Seidel method.

III. New MBR Algorithms

In the cascade structure, each transform operates on the residual resulting from subtracting the selected DP in a previous transform stage from the input residual to that stage. Each of the selected DP are weighted appropriately by a nonzero gain which must be optimally determined. In contrast, our parallel implementation, shown in Fig. 1, loosens the tight interconnection between the various transforms. We also find that the parallel structure is computationally less burdensome. We use the two previously discussed DP selection processes in our parallel structure multitransform algorithms.

Also, we develop an alternative nongradient DP selection strategy, the LBRRP [10], [16]. As it turns out, the LBRRP is closely related to the "matching pursuits" algorithm developed in [17]. However, the LBRRP may be slightly more efficient computationally, and the residual error is always reduced at each iteration.

A. Parallel Implementation Structure

Refer to Fig. 1. The weight estimates are found by minimizing the block mean square of the residual error. As in the cascade structure, the performance index is the square residual error (SRE)

$$J(\alpha,\beta) = r^t r \tag{1}$$

where $r = x - \hat{x}_a - \hat{x}_b$. Each transform branch of the parallel structure uses the original signal, while successive transforms in the cascade structure use the residuals from the previous transform. Working with the performance index (and assuming a block length of N)

$$J(\alpha,\beta) = \sum_{i=0}^{N-1} (x[i] - \hat{x}_a[i] - \hat{x}_b[i])^2. \tag{2}$$

Now, using

$$\sum_{i=0}^{N-1} (\hat{x}_a[i])^2 = \sum_{j=0}^{N-1} (\alpha[j]\hat{X}_a[j])^2 \tag{3}$$

$$\sum_{i=0}^{N-1} x[i]\hat{x}_a[i] = \sum_{j=0}^{N-1} \alpha[j](\hat{X}_a[j])^2 \tag{4}$$

$$\sum_{i=0}^{N-1} (\hat{x}_b[j])^2 = \sum_{j=0}^{N-1} (\beta[j]\hat{X}_b[j])^2 \tag{5}$$

$$\sum_{i=0}^{N-1} x[i]\hat{x}_b[i] = \sum_{j=0}^{N-1} \beta[j](\hat{X}_b[j])^2 \tag{6}$$

$$\sum_{i=0}^{N-1} \hat{x}_a[i]\hat{x}_b[i] = \sum_{j=0}^{N-1} \alpha[j]\hat{X}_a[j]\left\{\sum_{l=0}^{N-1} \cdot \left(\sum_{m=0}^{N-1} T_a[m,j]T_b^{-1}[m,l]\right)\beta[l]\hat{X}_b[l]\right\} \tag{7}$$

we put the performance index in the convenient matrix form

$$J(\alpha,\beta) = x^t x + \alpha^t \operatorname{diag}(\hat{X}_a^2)\alpha + \beta^t \operatorname{diag}(\hat{X}_b^2)\beta - 2(\hat{X}_a^2)^t\alpha - 2(\hat{X}_b^2)^t\beta + 2\alpha^t R_{ab}\beta \tag{8}$$

where $\hat{X}_{a,b}^2$ is the column vector of transform coefficients squared and

$$R_{ab} = \operatorname{diag}(\hat{X}_a)T_a T_b^{-1} \operatorname{diag}(\hat{X}_b). \tag{9}$$

We have assumed the transform to be real-valued. We use a gradient iterative search over α and β to minimize the energy of the residual error. In this case, the gradients are calculated using

$$\nabla_\alpha J = 2\operatorname{diag}(\hat{X}_a^2)\alpha - 2\hat{X}_a^2 + 2R_{ab}\beta \tag{10}$$

$$\nabla_\beta J = 2\operatorname{diag}(\hat{X}_b^2)\beta - 2\hat{X}_b^2 + 2R_{ab}^t\alpha. \tag{11}$$

The weight update equations are

$$\alpha(n+1) = \alpha(n) - \tfrac{1}{2}\mu\nabla_\alpha J \tag{12}$$

$$\beta(n+1) = \beta(n) - \tfrac{1}{2}\mu\nabla_\beta J. \tag{13}$$

Each step in the minimization procedure involves moving the weight vectors α and β along the gradient directions a distance controlled by step sizes μ_{α_i} and μ_{β_i}, respectively. The optimal step sizes are

$$\mu_{\alpha_i} = \frac{1}{2}\frac{(\nabla_{\alpha_i}J)^t(\nabla_{\alpha_i}J)}{(\nabla_{\alpha_i}J)R_{ab}(\nabla_{\alpha_i}J)} \tag{14}$$

and the step sizes for the β are found by substituting $(\nabla_{\beta_i}J)$.

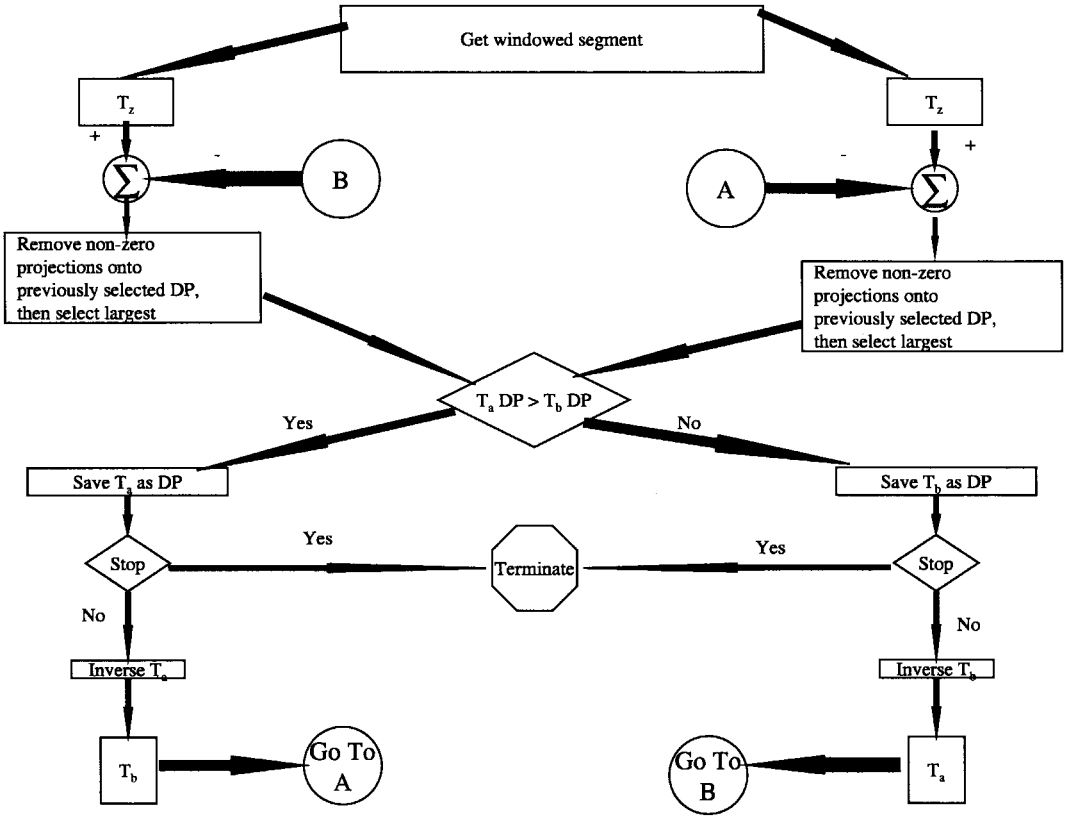


Fig. 2. LBRRP DP selection.

B. LBRRP

The processing of the LBRRP strategy is shown in Fig. 2. The LBRRP compares components from the various transform domains at each iteration step. Initially, the two transformations are applied to the input signal segment. The component with the largest projection in each transform domain is selected and compared to the largest projections in the other transform domains. The overall largest projection is removed, creating a residual signal. This residual is then projected onto each transform domain, and again the largest projection from among all transforms is removed, creating a new residual. Because only the largest projection from among all transform domains is removed at each iteration without regard to future iterations, the LBRRP is a "greedy" search and is not globally optimal. Several stopping criteria can be used: mean square error, maximum deviation, or the total number of selected projections (i.e., number of iterations of LBRRP). What distinguishes the LBRRP from the RRP developed in [9] and [10]? Because the selected DP come from nonorthogonal sets of basis functions, a basis function projection removed in a previous iteration can "come back to life." For instance, suppose that our search is the plane, and we have four basis functions. If the vertical component is removed followed by the selection and removal of a nonhorizontal basis function, then a vertical portion of the new residual will exist. This phenomena is addressed in the "matching pursuits" algorithm in [17]. The look-back portion of the LBRRP consists of monitoring projections of residuals onto any previously removed basis function. Any nonzero (or resuming) projection is removed and added directly to the previously removed value. The LBRRP requires one N-point transform to find one DP (after the first DP). The transform of the new residual in one domain must be calculated while the other transform is known. The cascade and parallel structure algorithms require 2 N-point transformations for each iteration, and the number of DP found is not related to the number of iterations.

IV. Simulation Results: Summary

The multitransform algorithms based on the above described structures and DP selection strategies are developed and tested. Several consonant-vowel-consonant (CVC) speech signals are examined. The 8 kHz-sampled speech signal was processed using a 32 ms trapezoidal window (256 points wide with a flat center region of length 200 points). Each segment of the windowed speech signal has total 20% overlap. For the various speech segments we use the DCT and WHT transforms. By examining the error surface of the criterion function at the converged-to-point, we make the following observations.

1) For unvoiced sound, most optimal weights are around 1 in value. Therefore, the assumption of domain independence is reasonable.
2) For voiced and transition sounds, the optimal weights do not necessarily converge to 1. Therefore the independence assumption is violated for these signals.
3) The LBRRP DP selection strategy has superior performance when compared to both the HFP and MSP. Even for the transition sound, the optimal weights are still close to 1. The independence assumption is reasonable.
4) For a large number (128 or more) of DP, the optimal weights are no longer approximately 1 for any of the DP selection strategies. However, the weight variation of the LBRRP strategy is the smallest among all observed strategies.

Using different implementation structures with the same total number of DP and different DP selection strategies, the performances of the multitransform algorithms are evaluated using signal-to-noise ratio and the average FLOPS (counted using MATLAB®) per iteration. The comparisons are shown in Table I. The total number of DP is 16 and the final weights are chosen as the weights at the 1000th iteration (except for LBRRP). The performances of the

TABLE I
COMPUTATIONAL COMPARISON OF IMPLEMENTATION STRUCTURE

Structure		SNR (dB) MSP	HPF	RRP	LBRRP	Average FLOPS per iteration MSP	HPF	RRP & LBRRP
Parallel	Voiced	8.2	4.8	10.6	10.6	$4.3 * 10^9$		$1.5 * 10^5$
	Unvoiced	3.5	2.5	3.9	3.9	$4.3 * 10^9$		$1.5 * 10^5$
	Transition	5.7	4.5	9.4	9.4	$4.3 * 10^9$		$1.5 * 10^5$
Cascade	Voiced	7.6	5.0	10.6	10.6	$10 * 10^9$		$1.5 * 10^5$
	Unvoiced	3.4	2.4	3.9	3.9	$10 * 10^9$		$1.5 * 10^5$
	Transition	5.6	4.4	9.4	9.4	$10 * 10^9$		$1.5 * 10^5$

TABLE II
COMPARISON OF DP SELECTION STRATEGIES (VOICED SOUND)

Number of DP Selected	SNR (dB) Multitransform Algorithms: Parallel Structure MSP	HPF	Cascade Structure MSP	HPF	Residual Projection RRP	LBRRP	SBR Algorithms DCT	DWT
4	-	-	-	-	-	-	4.9	2.4
8	-	-	-	-	-	-	7.2	3.9
16	8.2	4.8	7.6	5.0	-	-	10.6	5.9
32	10.2	9.6	9.6	7.1	14.7	14.9	14.4	8.9
64	10.5	9.9	9.9	9.3	20.2	21.2	20.4	13.6
128	10.4	9.9	9.9	9.3	29.1	31.6	30.7	22.1

TABLE III
COMPARISON OF DP SELECTION STRATEGIES (UNVOICED SOUND)

Number of DP Selected	SNR (dB) Multitransform Algorithms: Parallel Structure MSP	HPF	Cascade Structure MSP	HPF	Residual Projection RRP	LBRRP	SBR Algorithms DCT	DWT
4	1.6	1.1	1.6	1.1	1.4	1.4	1.3	0.8
8	2.2	1.6	2.1	1.5	2.4	2.4	2.1	1.5
16	3.5	2.5	3.4	2.4	3.9	3.9	3.6	2.6
32	5.2	4.0	4.8	3.9	6.3	6.5	6.0	4.2
64	7.8	7.8	7.4	7.3	9.9	10.4	9.7	7.1
128	9.6	8.3	8.7	7.7	16.6	19.2	16.6	12.7

multitransform algorithms with varying numbers of selected DP is given in Tables II–IV. For cases where the number of selected DP is less than 16, some strategies only choose projections from a single transform. In this case, the multitransform algorithm has defaulted to a DCT-only algorithm. Also, for cases where the number of selected DP is greater than 64, some DP selection strategies cannot find enough peaks in the spectrum. Analyzing those results which are presented in the tables, we summarize as follows.

1) Among the multitransform algorithms, the LBRRP consistently yields superior results except when only a few DP are chosen. In that case, the parallel implementation structure with the MSP DP selection strategy is superior.
2) For a moderate number of selected DP, the LBRRP performance is superior to the single transform methods as well as the GHPF and MSP strategies using either cascade or parallel structures.
3) For a large number of selected DP, the LBRRP performance is only slightly better when compared to the single transform methods.
4) For some specific signal segments, such as unvoiced segments, the MBR algorithms using the GHPF or MSP strategies and either the cascade or parallel structures are superior to a single transform method.

TABLE IV
COMPARISON OF DP SELECTION STRATEGIES (TRANSITION SOUND)

Number of DP Selected	SNR (dB) Multitransform Algorithms: Parallel Structure MSP	HPF	Cascade Structure MSP	HPF	Residual Projection RRP	LBRRP	SBR Algorithms DCT	DWT
4	-	-	-	-	-	-	4.9	1.8
8	4.7	2.4	4.6	2.3	7.2	7.2	7.1	2.5
16	5.7	4.5	5.6	4.4	9.4	9.4	9.2	3.6
32	7.1	5.1	7.0	5.2	11.6	11.8	11.1	5.5
64	10.0	7.9	9.7	7.3	13.3	16.2	14.1	9.2
128	10.8	10.3	10.5	10.0	22.0	25.1	20.8	16.1

V. CONCLUSIONS

The development of constrained LMS algorithms using a parallel multitransform structure for time-varying signal representation is given in this brief. This structure is compared to cascade structure multitransform algorithms developed elsewhere. The LBRRP strategy, similar to Mallat's "matching pursuits" algorithm, is also developed. The LBRRP performance is shown to be superior (in terms of performance and computational complexity) to either the cascade or parallel structure algorithms in most cases.

REFERENCES

[1] R. J. McAuley and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 744–754, Aug. 1986.
[2] F. Y. Y. Shum, A. R. Elliott, and W. O. Brown, "Speech processing with Walsh-Hadamard transforms," *IEEE Trans. Audio Electron.*, vol. 21, pp. 174–178, June 1973.
[3] M. R. Portnoff, "Short-time Fourier analysis of sampled speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 364–373, June 1981.
[4] M. Berouti, R. Schwartz, and J. Makhoul, "Enhancement of speech corrupted by additive noise," in *Proc. ICASSP'79*, Apr. 1979, pp. 208–211.
[5] A. S. Spanias and P. C. Loizou, "Mixed Fourier/Walsh transform scheme for speech coding at 4.0 kbits/s," *Proc. Inst. Elect. Eng. I*, vol. 139, pp. 473–478, Oct. 1992.
[6] W. B. Mikhael and A. S. Spanias, "Accurate representation of time-varying signals using mixed transforms with applications to speech," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 329–331, Feb. 1989.
[7] ——, "Efficient modeling of dominant transform components representing time-varying signals," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 331–334, Feb. 1989.
[8] W. B. Mikhael and A. Ramaswamy, "Residual error formulation and adaptive minimization for representing nonstationary signals using mixed transforms," *IEEE Trans. Circuits Syst. II*, vol. 39, pp. 489–492, July 1992.
[9] F. G. Safar, "Signal compression and representation using multiple bases representation," M.S. thesis, Virginia Polytech. Inst. State Univ., Blacksburg, June 1988.

[10] R. Khanna, "Image compression using multiple bases representation," M.S. thesis, Virginia Polytech. Inst. State Univ., Blacksburg, Aug. 1990.
[11] W. B. Mikhael and A. Ramaswamy, "Application of multitransforms for lossy image representation," *IEEE Trans. Circuits Syst. II*, vol. 41, pp. 431–434, June 1994.
[12] W. B. Mikhael, F. H. Wu, L. G. Kazovsky, G. Kang, and L. J. Fransen, "Adaptive filters with individual adaptation of parameters," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 677–685, Jul. 1986.
[13] J. S. Marques, L. B. Almeida, and J. M. Tribolet, "Harmonic coding at 4.0 kB/s," in *Proc. ICASSP'90*, Apr. 1990, pp. 17–20.
[14] A. S. Spanias, "A hybrid model for speech synthesis," in *Proc. ICASSP'90*, Apr. 1990, pp. 1521–1524.
[15] S. S. Narayan, A. M. Peterson, and M. J. Narasimha, "Transform domain LMS algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 609–615, June 1983.
[16] H. Li, "An image compression system," M.S. degree thesis, Univ. Oklahoma, June 1994.
[17] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Processing*, vol. 39, pp. 489–492, July 1992.

2-D Adaptive State-Space Filters Based on the Fornasini–Marchesini Second Model

Takao Hinamoto, Akimitsu Doi, and Mitsuji Muneyasu

***Abstract*— Based on the Fornasini–Marchesini second model, a technique is developed for implementing two-dimensional (2-D) adaptive state-space filters. First, the relationship between the coefficient sensitivities and the intermediate transfer functions is investigated for the Fornasini–Marchesini second model. A least mean square (LMS) adaptive algorithm is then presented by using new systems that generate the gradient signals. Finally, a 2-D adaptive line enhancer is constructed by using the 2-D adaptive state-space filter to illustrate the utility of the proposed technique.**

***Index Terms*— Adaptive filter, Fornasini–Marchesini's second model, LMS algorithm, 2-D system.**

I. Introduction

In order to achieve desired filtering performance, adaptive recursive filters are preferred because of lower order filter structure compared to that of adaptive transversal filters [1]–[4]. As an alternative to this technique, adaptive state-variable filters by using a gradient-based algorithm have been proposed recently [5]. Due to the capability of adapting arbitrary state-space filters, the designer can enjoy freedom to explore the performance advantages of different structures [6]. More recently, 2-D adaptive filters using the structure of 2-D adaptive FIR filters [7]–[14], 2-D adaptive IIR filters [15], [16], and 2-D adaptive state-space filters [17] have been studied with applications to image enhancement and noise reduction in an image. In [17], 2-D adaptive state-space filters which rely on the LMS algorithm have been developed by using the Roesser local state-space (LSS) model.

In this brief, based on the Fornasini–Marchesini second LSS model [18], a technique is developed for implementing 2-D adaptive state-space filters. The LMS algorithm is used to update the coefficients in the 2-D adaptive state-space filters. To compute all the required gradients of the filter coefficients, new systems are derived from the relation between coefficient sensitivities and intermediate functions. Finally, an illustrative example demonstrates the validity of the proposed technique.

Throughout this brief, the n-dimensional identity matrix is denoted by $\boldsymbol{I}_n$. The transpose and (i, j)th element of any matrix $\boldsymbol{A}$ are indicated by $\boldsymbol{A}^T$ and $(\boldsymbol{A})_{ij}$, respectively. Moreover, $E[\cdot]$ is used to denote the expected value.

Manuscript received August 23, 1995; revised June 10, 1996. This paper was recommended by Associate Editor R. W. Newcomb.

The authors are with the Faculty of Engineering, Hiroshima University, Higashi-Hiroshima 739, Japan (e-mail: hinamoto@ecl.sys.hiroshima-u.ac.jp).

Publisher Item Identifier S 1057-7130(97)03655-0.

II. 2-D Adaptive State-Space Filters

A. Generation of Gradient Signals

Let the LSS model for a 2-D digital filter be specified by

$$\begin{aligned}\boldsymbol{x}(i+1, j+1) &= \boldsymbol{A}_1\boldsymbol{x}(i, j+1) + \boldsymbol{A}_2\boldsymbol{x}(i+1, j) \\ &\quad + \boldsymbol{b}_1 u(i, j+1) + \boldsymbol{b}_2 u(i+1, j) \\ y(i, j) &= \boldsymbol{c}^T\boldsymbol{x}(i, j) + du(i, j)\end{aligned} \tag{1}$$

where $\boldsymbol{x}(i, j)$ is an $n \times 1$ local state vector, $u(i, j)$ is a scalar input, $y(i, j)$ is a scalar output, and $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, $\boldsymbol{c}$, and d are real matrices of appropriate dimensions. The transfer function of (1) is given by

$$\begin{aligned}H(z_1, z_2) &= \frac{Y(z_1, z_2)}{U(z_1, z_2)} \\ &= \boldsymbol{c}^T(\boldsymbol{I}_n - z_1^{-1}\boldsymbol{A}_1 - z_2^{-1}\boldsymbol{A}_2)^{-1} \\ &\quad \cdot (z_1^{-1}\boldsymbol{b}_1 + z_2^{-1}\boldsymbol{b}_2) + d\end{aligned} \tag{2}$$

where $U(z_1, z_2)$ and $Y(z_1, z_2)$ denote the z-transforms of the input and the output, respectively. Some sensitivity formulas for the LSS model (1) will be given to adapt the state-space parameters. These formulas require the definition of three sets of intermediate functions:

$$\begin{aligned}\boldsymbol{F}(z_1, z_2) &= \frac{\boldsymbol{X}(z_1, z_2)}{U(z_1, z_2)} \\ &= (\boldsymbol{I}_n - z_1^{-1}\boldsymbol{A}_1 - z_2^{-1}\boldsymbol{A}_2)^{-1} \\ &\quad \cdot (z_1^{-1}\boldsymbol{b}_1 + z_2^{-1}\boldsymbol{b}_2)\end{aligned} \tag{3}$$

$$\boldsymbol{G}_k^T(z_1, z_2) = z_k^{-1}\boldsymbol{c}^T(\boldsymbol{I}_n - z_1^{-1}\boldsymbol{A}_1 - z_2^{-1}\boldsymbol{A}_2)^{-1}, \quad k = 1, 2 \tag{4}$$

where $\boldsymbol{X}(z_1, z_2)$ stands for the z-transform of the local state vector, $Y(z_1, z_2) = \boldsymbol{G}_k^T(z_1, z_2)\boldsymbol{\varepsilon}_k(z_1, z_2)$, and $\boldsymbol{\varepsilon}_k(z_1, z_2)$, $k = 1, 2$ denote signal injection vectors at the inputs of the delay operators $z_k^{-1}\boldsymbol{I}_n$.

Definition 1: Let $\boldsymbol{Q}$ be an $m \times n$ real matrix and let $f(\boldsymbol{Q})$ be a scalar complex function of $\boldsymbol{Q}$, differentiable w.r.t. all the entries of $\boldsymbol{Q}$. The sensitivity function of f w.r.t. $\boldsymbol{Q}$ is then defined as

$$\boldsymbol{S}_Q = \frac{\partial f}{\partial \boldsymbol{Q}}$$

with

$$(\boldsymbol{S}_Q)_{kl} = \frac{\partial f}{\partial q_{kl}} \tag{5}$$

where q_{kl} denotes the (k, l)th entry of the matrix $\boldsymbol{Q}$.

To obtain gradient signals, the derivatives of the output signal w.r.t. each of the filter coefficient matrices are related to the intermediate functions.

$$\frac{\partial Y(z_1, z_2)}{\partial \boldsymbol{A}_k} = \boldsymbol{G}_k(z_1, z_2)\boldsymbol{X}^T(z_1, z_2) \tag{6}$$

$$\frac{\partial Y(z_1, z_2)}{\partial \boldsymbol{b}_k} = \boldsymbol{G}_k(z_1, z_2)U(z_1, z_2) \tag{7}$$

$$\frac{\partial Y(z_1, z_2)}{\partial \boldsymbol{c}} = \boldsymbol{X}(z_1, z_2) \tag{8}$$

$$\frac{\partial Y(z_1, z_2)}{\partial d} = U(z_1, z_2), \qquad k = 1, 2. \tag{9}$$

From (6) to (9) it is clear that the gradient signals needed to adapt the $\boldsymbol{c}$ vector are available as the output states, $\boldsymbol{x}(i, j)$, whereas the gradient signal for the d scalar is the input signal $u(i, j)$. However, the gradient signals required to adapt the $\boldsymbol{A}_k$ matrix and $\boldsymbol{b}_k$ vector, $k = 1, 2$ must be created by new systems with the intermediate functions from the input to the states, equal to $\boldsymbol{G}_k(z_1, z_2)$, $k = 1, 2$ of the original filter. The new systems are described by

$$\begin{aligned}\boldsymbol{W}_1(i+1, j+1) = {} & \boldsymbol{A}_1^T(i, j+1)\boldsymbol{W}_1(i, j+1) \\ & + \boldsymbol{A}_2^T(i+1, j)\boldsymbol{W}_1(i+1, j) \\ & + \boldsymbol{c}(i, j+1)\boldsymbol{x}^T(i, j+1)\end{aligned} \tag{10a}$$

$$\begin{aligned}\boldsymbol{W}_2(i+1, j+1) = {} & \boldsymbol{A}_1^T(i, j+1)\boldsymbol{W}_2(i, j+1) \\ & + \boldsymbol{A}_2^T(i+1, j)\boldsymbol{W}_2(i+1, j) \\ & + \boldsymbol{c}(i+1, j)\boldsymbol{x}^T(i+1, j)\end{aligned} \tag{10b}$$

$$\begin{aligned}\boldsymbol{v}_1(i+1, j+1) = {} & \boldsymbol{A}_1^T(i, j+1)\boldsymbol{v}_1(i, j+1) \\ & + \boldsymbol{A}_2^T(i+1, j)\boldsymbol{v}_1(i+1, j) \\ & + \boldsymbol{c}(i, j+1)u(i, j+1)\end{aligned} \tag{11a}$$

$$\begin{aligned}\boldsymbol{v}_2(i+1, j+1) = {} & \boldsymbol{A}_1^T(i, j+1)\boldsymbol{v}_2(i, j+1) \\ & + \boldsymbol{A}_2^T(i+1, j)\boldsymbol{v}_2(i+1, j) \\ & + \boldsymbol{c}(i+1, j)u(i+1, j)\end{aligned} \tag{11b}$$

where

$$\begin{aligned}\boldsymbol{x}(i+1, j+1) = {} & \boldsymbol{A}_1(i, j+1)\boldsymbol{x}(i, j+1) \\ & + \boldsymbol{A}_2(i+1, j)\boldsymbol{x}(i+1, j) \\ & \cdot \boldsymbol{b}_1(i, j+1)u(i, j+1) \\ & + \boldsymbol{b}_2(i+1, j)u(i+1, j)\end{aligned}$$

and the initial conditions of all the above systems are assumed to be null. Here, $\boldsymbol{A}_k(i, j)$ and $\boldsymbol{b}_k(i, j)$, $k = 1, 2$ are the estimates of coefficient matrices $\boldsymbol{A}_k$ and $\boldsymbol{b}_k$ at location (i, j), respectively, and are updated in the following manner.

B. Adaptive Algorithm

A block diagram of a 2-D adaptive state-space filter is depicted in Fig. 1 where the state-space parameters now change with each location and, hence, are functions of location (i, j). Suppose $u(i, j)$ and $r(i, j)$ are stationary discrete stochastic processes. Let an error signal $e(i, j)$ be defined by the difference between a reference signal $r(i, j)$ and the filter output $y(i, j)$. During adaption, the coefficients of an adaptive filter are changed to minimize the mean-squared error signal $E[e^2(i, j)]$. To find a minimum of the mean-squared error performance surface, the steepest descent algorithm can be employed with the use of gradient signals. It is assumed that 2-D data are of size $M \times N$, i.e., $\{(i, j) | 0 \le i \le M-1, 0 \le j \le N-1\}$.

Taking the 2-D spatial correlations of pixels in a neighborhood into account, the idea of 2-D diagonal processing is useful. Combining the steepest descent method with the diagonal scanning scheme, an updating equation for any coefficient p of the adaptive filter is written as

$$p(i', j') = p(i, j) - \mu \frac{\partial \mathrm{E}[e^2(i, j)]}{\partial p(i, j)} \tag{12}$$

where $p(i', j')$ is the updated coefficient at location (i, j) and μ is a step-size parameter which controls convergence of the algorithm.

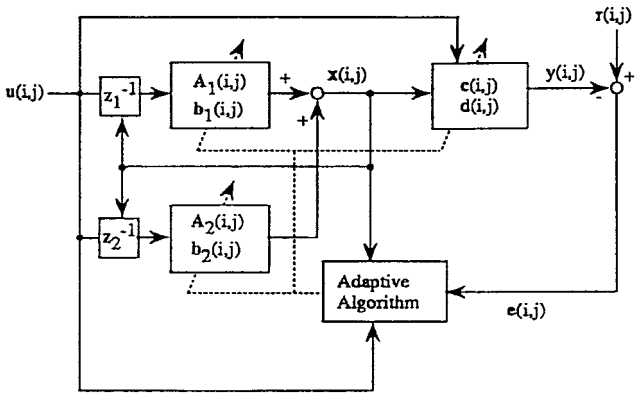


Fig. 1. Block diagram of 2-D adaptive state-space filters.

During adaption, the same data are used repeatedly. The indexes (i, j) and (i', j') satisfy the condition that the sum of i and j (that of i' and j') along each diagonal line is constant. There are six different cases for changing the indexes as given below.

For $i + j$ an even number

$$\begin{aligned} &i' = i+1, \quad j' = j-1, \quad \text{if } i < M-1 \text{ and } j \neq 0 \\ &i' = i+1, \quad j' = j, \quad \text{if } i < M-1 \text{ and } j = 0 \\ &i' = i, \quad j' = j+1, \quad \text{if } i = M-1. \end{aligned}$$

For $i + j$ an odd number

$$\begin{aligned} &i' = i-1, \quad j' = j+1, \quad \text{if } j < N-1 \text{ and } i \neq 0 \\ &i' = i, \quad j' = j+1, \quad \text{if } j < N-1 \text{ and } i = 0 \\ &i' = i+1, \quad j' = j, \quad \text{if } j = N-1. \end{aligned}$$

The instantaneous value of the squared error signal is usually utilized to approximate the expected value in the LMS algorithm. With such an approximation, we obtain

$$p(i', j') = p(i, j) + 2\mu e(i, j)\frac{\partial y(i, j)}{\partial p(i, j)}. \tag{13}$$

From (13), adaption equations for the filter coefficients are obtained as

$$\boldsymbol{A}_k(i', j') = \boldsymbol{A}_k(i, j) + 2\mu e(i, j)\boldsymbol{W}_k(i, j) \qquad k = 1, 2 \tag{14}$$

$$\boldsymbol{b}_k(i', j') = \boldsymbol{b}_k(i, j) + 2\mu e(i, j)\boldsymbol{v}_k(i, j), \qquad k = 1, 2 \tag{15}$$

$$\boldsymbol{c}(i', j') = \boldsymbol{c}(i, j) + 2\mu e(i, j)\boldsymbol{x}(i, j) \tag{16}$$

$$d(i', j') = d(i, j) + 2\mu e(i, j)u(i, j) \tag{17}$$

where the gradient signals, $\boldsymbol{x}(i, j)$, $\boldsymbol{W}_k(i, j)$, and $\boldsymbol{v}_k(i, j)$, $k = 1, 2$ are obtained from the new systems shown in (10) and (11). From the foregoing arguments, it is possible to implement 2-D adaptive state-space filters.

III. Illustrative Example

A block diagram of the 2-D adaptive line enhancer is drawn in Fig. 2 where $d(i, j)$ is an original image, $v(i, j)$ is an additive noise, and $r(i, j)$ is a reference signal specified by $r(i, j) = d(i, j) + v(i, j)$. The input signal $u(i, j)$ to the 2-D adaptive state-space filter is formed by delaying the reference signal by $(z_1^{-1} + z_2^{-1})/2$ and is given by $u(i, j) = [r(i-1, j) + r(i, j-1)]/2$. The delay is used as a decorrelation operator to obtain the input signal from the reference

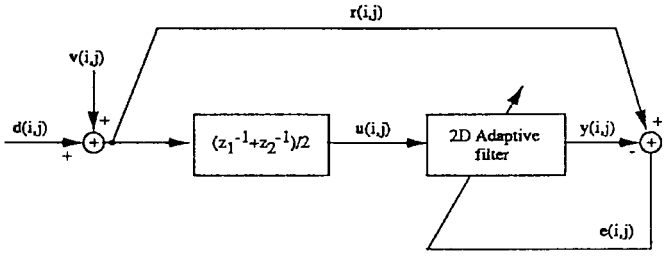


Fig. 2. 2-D adaptive line enhancer structure.

signal. This enables us to remove the effects of additive noise from an image by making use of autocorrelations.

The original image of 100 × 100 pixels with 256 gray levels is shown in Fig. 3. The Gaussian noise of zero mean and variance 500 is added to the original image to produce the degraded image in Fig. 4.

Let the order of the 2-D adaptive line enhancer be $n = 4$ and let the initial coefficient matrices be chosen to be a certain 2-D IIR low-pass filter [19, p. 108] given as follows:

$$\boldsymbol{A}_1(0,\,0) = \begin{bmatrix} 0.057\,002 & 0.083\,792 & 0.404\,317 & -0.150\,120 \\ -0.205\,230 & 0.334\,741 & -0.138\,844 & 0.055\,005 \\ 0.224\,854 & 0.249\,008 & 0.112\,312 & 0.032\,287 \\ 0.059\,384 & 0.333\,645 & 0.090\,921 & 0.149\,825 \end{bmatrix}$$

$$\boldsymbol{A}_2(0,\,0) = \begin{bmatrix} 0.501\,861 & 0.029\,451 & -0.048\,577 & 0.125\,627 \\ 0.009\,376 & 0.459\,600 & -0.133\,237 & 0.283\,091 \\ -0.018\,777 & 0.294\,124 & 0.380\,790 & -0.365\,214 \\ -0.132\,510 & -0.326\,956 & 0.282\,682 & 0.369\,630 \end{bmatrix}$$

$$\boldsymbol{b}_1(0,\,0) = \begin{bmatrix} -0.270\,682 & 0.184\,186 & 0.195\,692 & 0.500\,127 \end{bmatrix}^T$$

$$\boldsymbol{b}_2(0,\,0) = \begin{bmatrix} 0.076\,862 & 0.186\,164 & 0.018\,185 & -0.045\,435 \end{bmatrix}^T$$

$$\boldsymbol{c}(0,\,0) = \begin{bmatrix} 0.409\,775 & -0.105\,288 & 0.249\,880 & 0.181\,697 \end{bmatrix}^T$$

$$d(0,\,0) = 0.001\,924.$$

The step-size parameter μ was chosen to 1.1×10^{-7} by experiments (trial and error). To evaluate the characteristics of the 2-D adaptive line enhancer, we used a signal to noise ratio (SNR) defined by

$$\text{SNR} = 10\,\log_{10}\frac{\sum_{i=n}^{99}\sum_{j=n}^{99} d(i,j)^2}{\sum_{i=n}^{99}\sum_{j=n}^{99}[y(i,j)-d(i,j)]^2}. \tag{18}$$

The 2-D adaptive line enhancer was realized by the proposed technique. Consequently, the initial SNR equal to 11.893 756 was changed to 15.602 459 at $l = 30$. Here, l stands for the number of normalized iterations, i.e., $l = m/MN = m \times 10^{-4}$ where m is the number of iterations. The output image produced by the 2-D adaptive line enhancer at l =30 is shown in Fig. 5.

The 2-D adaptive state-space filter presented here was compared to the 2-D adaptive FIR filter reported in [7] with the order $n = 6$ and the step size parameter $\mu = 1.2 \times 10^{-8}$. Applying the 2-D adaptive FIR filter, the initial SNR equal to 11.943 223 became 15.453 219 at $l = 30$. In other words, the proposed 2-D adaptive state-space filter yields higher SNR than the 2-D adaptive FIR filter presented in [7]. Notice that the number of the coefficients of the transfer function of the adaptive state-space filter is equal to 29. Alternatively, the adaptive FIR filter has 36 coefficients.

Fig. 3. Original image.

Fig. 4. Image degraded by additive noise.

Fig. 5. Result processed by the 2-D adaptive line enhancer.

IV. Conclusion

Based on the Fornasini–Marchesini second model, a technique has been developed for implementing 2-D adaptive state-space filters. This has been done by using the LMS algorithm. To obtain all the gradients required for adapting a 2-D state-space filter, new systems related to the intermediate functions have been presented. Finally, an illustrative example has been given to demonstrate the effectiveness of the proposed technique.

References

[1] P. L. Feintuch, "An adaptive recursive LMS filter," *Proc. IEEE,* vol. 64, pp. 1622–1624, Nov. 1976.

[2] M. G. Larimore, J. R. Treichler, and C. R. Johnson, "SHARF: An algorithm for adapting IIR digital filters," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-28, pp. 428–440, Aug. 1980.

[3] T. C. Hsia, "A simplified adaptive recursive filter design," *Proc. IEEE,* vol. 69, pp. 1153–1155, Sept. 1981.

[4] H. Fan and W. K. Jenkins, "A new adaptive IIR filter," *IEEE Trans. Circuits Syst.,* vol. CAS-33, pp. 939–947, Oct. 1986.

[5] D. A. Johns, W. M. Snelgrove, and A. S. Sedra, "Adaptive recursive state-space filters using a gradient-based algorithm," *IEEE Trans. Circuits Syst.,* vol. 37, pp. 673–684, June 1990.

[6] M. Nayeri and W. K. Jenkins, "Analysis of alternate realizations of adaptive IIR filters," in *Proc. IEEE Int. Symp. Circuits Syst.,* June 1988, pp. 2157–2160.

[7] M. M. Hadhoud and D. W. Thomas, "The two-dimensional adaptive LMS (TDLMS) algorithm," *IEEE Trans. Circuits Syst.,* vol. 35, pp. 485–494, May 1988.

[8] M. Ohki and S. Hashiguchi, "Two-dimensional LMS adaptive filters," *IEEE Trans. Consum. Electron.,* vol. 37, pp. 66–73, Feb. 1991.

[9] W. B. Mikhael and S. M. Ghosh, "Two-dimensional variable step-size sequential adaptive gradient algorithms with applications," *IEEE Trans. Circuits Syst.,* vol. 38, pp. 1577–1580, Dec. 1991.

[10] ____, "Two-dimensional block adaptive filtering algorithms," in *Proc. IEEE Int. Symp. Circuits Syst.,* May 1992, pp. 1219–1222.

[11] S.-C. Pei, C.-Y. Lin, and C.-C. Tsung, "Two-dimensional LMS adaptive linear phase filters," in *Proc. IEEE Int. Symp. Circuits Syst.,* May 1993, pp. 311–314.

[12] W. B. Mikhael and S. M. Ghosh, "Two-dimensional optimum block adaptive filtering algorithms for image restoration and enhancement," in *Proc. IEEE Int. Symp. Circuits Syst.,* May 1993, pp. 419–422.

[13] J. C. Strait and W. K. Jenkins, "A fast two-dimensional quasinewton adaptive filter," in *Proc. IEEE Int. Symp. Circuits Sys.,* May/June 1994, vol. 2, pp. 149–152.

[14] S.-G. Chen and Y.-A. Kao, "A new efficient 2-D LMS adaptive filtering algorithm," in *Proc. IEEE Int. Symp. Circuits Syst.,* May/June 1994, vol. 2, pp. 233–236.

[15] M. Kawamata, E. Kawakami, and T. Higuchi, "Realization of lattice-form separable-denominator 2-D adaptive filters," in *Proc. IEEE Int. Symp. Circuits Syst.,* May 1993, pp. 295–298.

[16] A. C. Tan and S.-T. Chen, "Two-dimensional adaptive LMS IIR filter," in *Proc. IEEE Int. Symp. Circuits Syst.,* May 1993, pp. 299–302.

[17] M. Muneyasu and T. Hinamoto, "2-D adaptive state-space digital filters using LMS algorithm," in *Proc. IEEE Int. Symp. Circuits Syst.,* May/June 1994, vol. 2, pp. 65–68.

[18] E. Fornasini and G. Marchesini, "Doubly-indexed dynamical systems: State-space models and structural properties," *Math. Sys. Theory,* vol. 12, pp. 59–72, 1978.

[19] T. Hinamoto, "2-D Lyapunov equation and filter design based on the Fornasini–Marchesini second model," *IEEE Trans. Circuits Syst. I,* vol. 40, pp. 102–110, Feb. 1993.

Efficient Polyphase DFT Filter Banks with Fading Memory

Annamária R. Várkonyi-Kóczy

***Abstract*— In this brief, new composite polyphase filter banks are presented for the implementation of the recursive Fourier transformation and some of its generalizations. These structures can be operated both in sliding window and block recursive modes with a computational complexity in the order of the fast algorithms. The parallelization applied enables very high speed and also a considerably higher sampling rate. Based on this structure almost all issues of the so-called transform-domain digital signal processing can be reconsidered.**

***Index Terms*— Discrete transforms, fading memory effect, fast algorithms, IIR digital filtering, polyphase filterbanks, recursive filters, transform domain analysis.**

I. Introduction

The discrete Fourier transformation (DFT) of a sequence of N samples can be defined as

$$X_m = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)mn} \tag{1}$$

where $x(n)$ denotes the nth input sample, X_m is the mth DFT component (n the discrete "time" index $n = 0, 1, \cdots, N-1$ or running; m is the discrete "frequency" index, $m = 0, 1, \cdots, N-1$). The factor of $1/N$ is a simple normalizing factor. The recursive version of the DFT is a sliding-window technique for calculating the Fourier coefficients and/or components [1]. For its implementation an observer structure [2], [3] proved to be efficient [4], [5].

Recently a new technique has been developed for the implementation of the recursive DFT based upon the concepts of polyphase filtering and the fast Fourier transformation (FFT) [14]. The novelty of this approach is a decomposition of a larger size single-input multiple-output (SIMO) DFT filter-bank into a proper parallel combination of smaller ones similarly as it is made in the nondestructive zoom technique applied in the case of larger scale data blocks (see e.g., [6]).

A DFT of size N can be evaluated as L separate DFT's of size M ($N = ML$) as it is illustrated in Fig. 1. (for $M = N/2$ and $L = 2$) together with a "phase compensation." The 2-point DFT blocks operate on complex values and are preceded by a "phase compensation" which is unavoidable due to the delayed reference positions of the $N/2$-point DFT blocks. The overall decomposition scheme on the input side follows the "decimation-in-time" (see e.g., [7]) while on the output side the "decimation-in-frequency" principles.

Obviously for every smaller size recursive DFT block a similar decomposition is possible if their size M and/or L are not prime numbers. A "total" decomposition corresponds to the structure of the FFT (see e.g., [5]), but operates recursively and has remarkable features with respect to other techniques of similar complexity [8], [9].

Manuscript received March 2, 1995; revised August 16, 1996. This work was supported by the Hungarian Fund for Scientific Research (OTKA) under Contract 4-100. This paper was recommended by Associate Editor S. Kiaei.

The author is with the Department of Measurement and Instrument Engineering, Technical University of Budapest, H-1521 Budapest, Hungary.

Publisher Item Identifier S 1057-7130(97)03660-4.

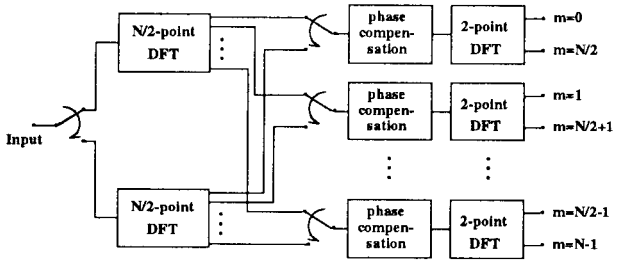
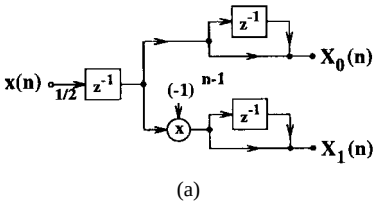
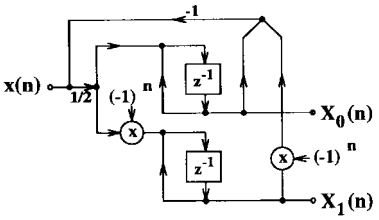


Fig. 1. Polyphase decomposition for $M = N/2$ and $L = 2$.



(a)



(b)

Fig. 2. 2-point DFT structure versions. (a) Tree structure version. (b) Resonator-based version.

This recursive FFT technique is well suited to system identification problems where periodic, multifrequency (e.g., multisine) perturbation signals [10] are applied. If the actual frequency components of the perturbation signal directly correspond to the discrete frequency locations of the FFT, then the systematic error of frequency-component measurements can be avoided.

Section II presents the implementation of the radix-2 case for the recursive DFT while Section III is a generalization toward IIR applications like recursive DFT with fading memory [11].

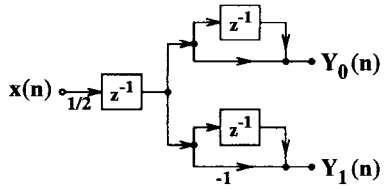
II. The Basic Building Blocks

In the radix-2 case a total decomposition results in 2-point DFT's. In Fig. 2 two possible versions are presented. In case of complex inputs such blocks are required both for the real and for the imaginary parts. Fig. 2(a) shows the simplest tree-structure version which due to its "pipelined" nature meets the requirements of real-time execution. Fig. 2(b) presents the so-called resonator-based version [4], [5] which can play an interesting role in the generalizations toward IIR applications.

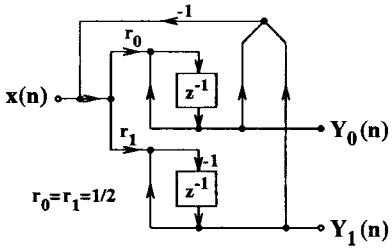
The decomposition described above can be directly utilized even in the case of DFT filter-banks. The DFT filter-banks produce the Fourier components instead of the Fourier coefficients. For many applications this fact has real practical advantages. With a slightly different "phase compensation," i.e., with the proper combination of the complex demodulation-modulation and the phase compensation, the DFT filter-bank version can also be generated. For this case the corresponding 2-point DFT filter-bank forms are given in Fig. 3.

III. Recursive DFT Filter Bank with Fading Memory

As it is well known from the literature (see e.g., [1]) the averaging effect in (1) can have an interesting frequency-domain interpretation.



(a)



(b)

Fig. 3. 2-point DFT filter-bank versions. (a) Tree structure version. (b) Resonator-based version.

One of the typical characterizations is the z-domain transfer function of the averager

$$\frac{1}{N}\,\frac{1-z^{-N}}{1-z^{-1}}. \tag{2}$$

This transfer function is present in every sliding DFT channel that produces the linear average of the last N (properly demodulated) input samples. A very similar characterization is valid for the DFT filter-bank

$$\frac{1}{N}\,\frac{1-z^{-N}}{1-z_m z^{-1}}, \qquad z_m = e^{j(2\pi/N)m} \quad m = 0,\, 1,\, \cdots,\, N-1, \tag{3}$$

which operates as an "averager" at the mth frequency position. The extension of the recursive DFT to a fading memory version, as it is motivated in [11] and [12], can be easily solved with the application of recursive building blocks producing also certain poles for the overall transfer function. Due to implementational reasons the resonator-based approach is applied which proved to be advantageous also for higher-order blocks. If in Fig. 3(b) the coefficients r_0 and r_1 are not fixed as 1/2 we can implement the transfer function of the channels:

$$H_0(z) = \frac{r_0 z^{-1}(1+z^{-1})}{1+(r_0-r_1)z^{-1}+(r_0+r_1-1)z^{-2}}, \tag{4}$$

$$H_1(z) = \frac{-r_1 z^{-1}(1-z^{-1})}{1+(r_0-r_1)z^{-1}+(r_0+r_1-1)z^{-2}}. \tag{5}$$

The fading memory effect described in [11] and in [12] with an other interpretation, requires uniformly distributed poles within the unit circle. The characterization of this modified DFT system can be given if (2) is replaced by

$$\frac{1-a}{N}\,\frac{1-z^{-N}}{1-az^{-N}}\,\frac{1}{1-z^{-1}} \tag{6}$$

where the poles, which are the Nth roots of a are responsible for the fading memory effect. If we consider the behavior of the polyphase structures [13], it turns out that these poles can be generated for the radix-2 case in such a way that the first stage of the overall structure consisting of $N/2$ 2-point DFT blocks is realized using blocks of Fig. 3(b) with

$$r_0 = r_1 = \frac{1-a}{2}. \tag{7}$$

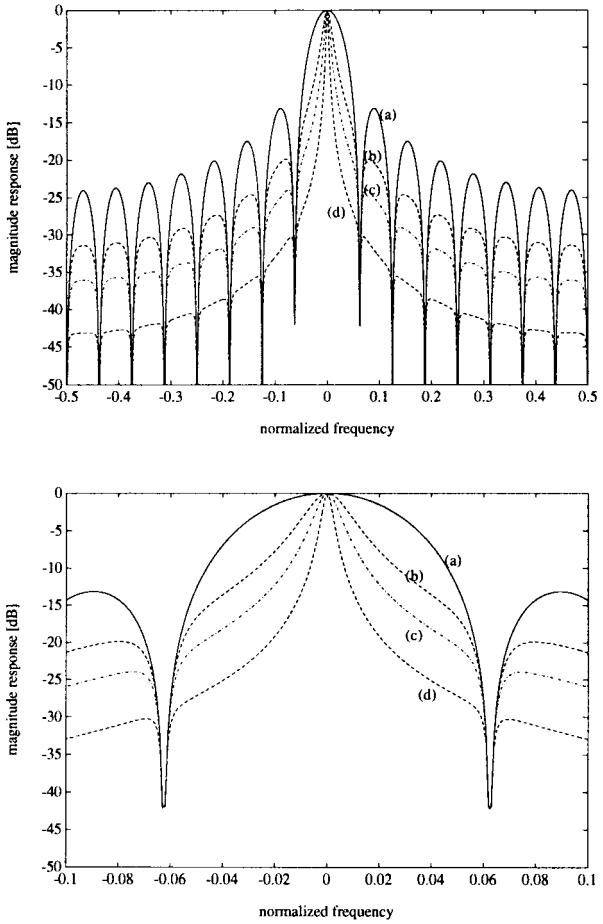


Fig. 4. Magnitude characteristics of the DFT filter-banks with different $a > 0$ parameters. (a) $a = 0$. (b) $a = 0.4$. (c) $a = 0.6$. (d) $a = 0.8$.

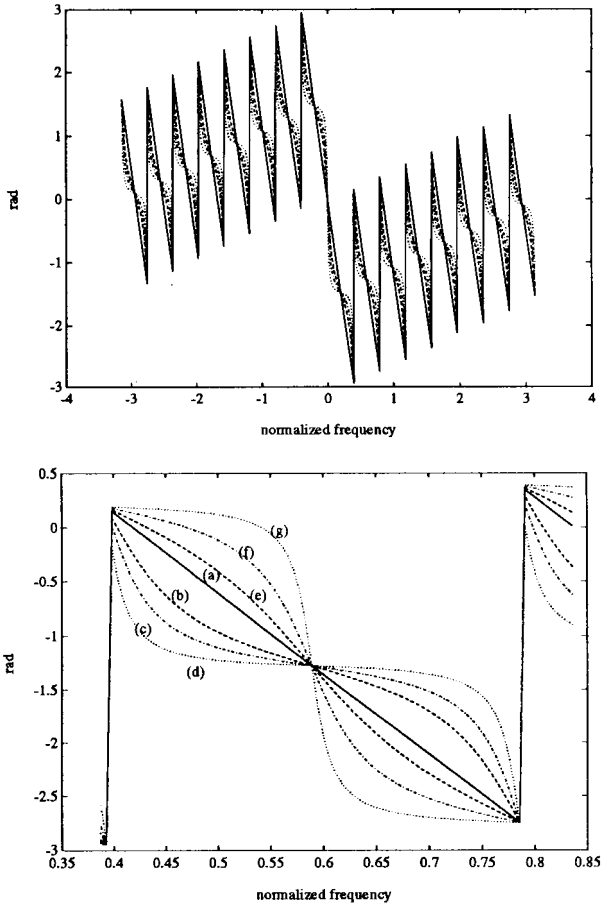


Fig. 5. Phase characteristics of the DFT filter-banks with different a parameters. (a) $a = 0$. (b) $a = 0.4$. (c) $a = 0.6$. (d) $a = 0.8$. (e) $a = -0.2$. (f) $a = -0.5$. (g) $a = -0.8$.

The effects of these poles can be illustrated with the corresponding magnitude and phase characteristics. Fig. 4 presents the magnitude behavior for different $a > 0$ values. From this figure it turns out that in the case of synchronized sine-waves higher selectivity can be achieved, i.e., this simple modification can really serve the multisine perturbation approaches. Fig. 5. shows the phase characteristics for different a values.

If not only the first stage of the radix-2 structure is replaced with the block of Fig. 3(b), but also some others, further poles can be introduced.

The proposed DFT structure together with this possibility to introduce poles can serve also as an alternative solution for signal processing problems to be solved in the transform-domain. This means the generalization of the frequency sampling concept (see, e.g., [1]) related to the sliding DFT and the introduction of arbitrary pole-sets by setting the r_0, r_1 parameters properly within the last stage of the overall system. Obviously this latter can be combined with the fading memory effect of (6), as well.

IV. Conclusions

In this brief new polyphase DFT filter bank versions has been reported. The parallelization applied enables decimation and the use of more (in the radix-2 case up to $N/2$) parallel A/D converters and thus a considerably higher sampling rate can be achieved with the very same resolution and accuracy. By applying recursive building blocks within the structure the recursive DFT with fading memory can also be implemented. This latter has real importance in such system identification problems where periodic, multifrequency perturbation signals [10] are applied or the Kalman filtering approach [12] is utilized. In this brief the details are given only for the radix-2 case, however, the ideas are quite general, the extension to other cases is straightforward.

References

[1] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1975, p. 47.
[2] G. H. Hostetter, "Recursive discrete Fourier transformation," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 183–190, Apr. 1980.
[3] G. Péceli, "A common structure for recursive discrete transforms," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 1035–1036, Oct. 1986.
[4] ——, "Resonator-based digital filters," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 156–159, Jan. 1989.
[5] M. Padmanabhan and K. Martin, "Resonator-based filter-banks for frequency-domain applications," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 1145–1159, Oct. 1991.
[6] R. B. Randall, *Frequency Analysis.* Bruel & Kjaer, 1987.
[7] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1975.
[8] G. Bruun, "z-transform DFT filters and FFT's," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 56–63, Feb. 1978.
[9] M. Vetterli, "Tree structures for orthogonal transforms and application to the Hadamard transform," *Signal Processing*, vol. 5, pp. 473–484, 1983.
[10] K. Godfrey, *Perturbation Signals for System Identification.* Englewood Cliffs, NJ: Prentice Hall, 1993.
[11] G. H. Hostetter, "Recursive discrete Fourier transform with fading memory," in *Proc. ISCAS'83*, pp. 80–85.

[12] R. R. Bitmead, A. C. Tsoi, and P. J. Parker, "A Kalman filtering approach to short-time Fourier analysis," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1493–1501, Dec. 1986.
[13] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1983.
[14] A. R. Várkonyi-Kóczy, "A recursive fast Fourier transformation algorithm," *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 614–616, Sept. 1995.

Implementations of Adaptive IIR Filters with Lowest Complexity

Geoffrey A. Williamson

***Abstract*—The problem of implementing adaptive IIR filters of minimum complexity is considered. The complexity used here is the number of multiplications in the implementation of the structures generating both the adaptive filter output and the sensitivities to be used in any gradient-based algorithm. This complexity is independent of the specific adaptive algorithm used. It is established that the sensitivity generation requires a minimum of N additional states, where N is the order of the filter. This result is used to show a minimum complexity of $3N + 1$ multiplications for an order N filter. Principles to use in the construction of such lowest complexity implementations are provided, and examples of minimum complexity direct-form, cascade-form, and parallel-form adaptive IIR filters are given.**

***Index Terms*—Adaptive filtering, adaptive IIR filters, sensitivity functions.**

I. Introduction

For several realizations of adaptive IIR filters, most notably the cascade-and lattice-forms, computational complexity has been prohibitively large. To implement gradient descent based algorithms such as the least mean square (LMS) and the Gauss-Newton (GN) algorithms, one must generate output sensitivity functions with respect to the adapted parameters, and these computations must be included in the implementation complexity. For lattice-form adaptive IIR filters [1], [2], the computational burden of sensitivity generation is formidable, though complexity reduction from the original algorithm is possible [3]. Cascade-form adaptive IIR filters [4] also engender complicated sensitivity generation, but reconfigurations of the cascaded filter structure can reduce the complexity of the sensitivity generation [5]. The same holds true for adaptive FIR filters implemented in cascade-form [6].

These issues raise the question of what is the minimal level of computation, including that of sensitivity function generation, that is needed to implement an Nth-order adaptive IIR filter. We use as a measure of complexity the total number of multiplications required to compute, at iteration k, the adaptive filter output together with the sensitivities with respect to all adapted parameters. We demonstrate in this brief that the minimal complexity in this sense is $3N + 1$ multiplications. Of these, $2N + 1$ correspond to multiplications in the generation of the filter output: one multiplication for each of $N + 1$ degrees of freedom in the numerator of the filter's transfer function, and one multiplication for each of N degrees of freedom for the denominator. Only N additional multiplications are required to obtain sensitivities that are not already generated in the process of obtaining the filter output.

Manuscript received December 1, 1994; revised December 18, 1996. This paper was recommended by Associate Editor P. A. Regalia.

The author is with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: gaw@ece.iit.edu).

Publisher Item Identifier S 1057-7130(97)06031-X.

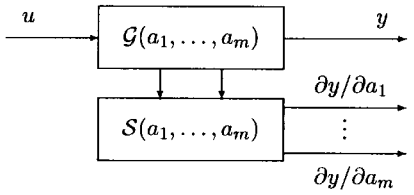


Fig. 1. Composite output and sensitivity generation.

Note that this result gives only a lower bound on the implementation complexity of Nth-order adaptive IIR filters. For a particular realization, there is the possibility that the lower bound cannot be achieved. Furthermore, we exclude algorithm complexity from the measure. To implement GN algorithms requires a significant number of additional multiplications, so that the complexity from signal generation may be small in comparison. However, all gradient descent based algorithms require the sensitivities, so our work establishes minimum complexity levels for this aspect of adaptive filter implementation. Furthermore, when employing the LMS algorithm, the potential for complexity reduction is significant.

In addition to establishing the main result, we also motivate general techniques for reducing the implementation complexity. We then indicate lowest complexity realizations for direct-form, cascade-form, and parallel-form adaptive IIR filters.

II. Lowest Complexity Adaptive IIR Filters

Fig. 1 shows the general form for the composite system $(\mathcal{G}, \mathcal{S})$ generating both the filter output (via subsystem $\mathcal{G}$) and the sensitivities (via subsystem $\mathcal{S}$) when the filter is parametrized via parameters $a_1, \ldots, a_m$. The structure of $\mathcal{G}$ and $\mathcal{S}$ can be related: each sensitivity

$$\frac{\partial y}{\partial a_i}$$

may be generated by replicating in $\mathcal{S}$ the filter structure $\mathcal{G}$, and exciting this replication with a signal taken from $\mathcal{G}$ [2].

We rely in the exposition on a feedback gain model (FGM) representation for $\mathcal{S}$. A system representable by an FGM is one in which the adjustable parameters all appear as internal feedback gains. Most of the usual direct-, parallel-, cascade-, and lattice-form parametrizations of adaptive filters possess such a representation.[1] In an FGM with parameters $a_1, \ldots, a_m$, the dependence on parameter a_i is as shown in Fig. 2. The dimension of I appearing in the feedback block of the figure indicates the number of times that a_i appears as a gain in the filter.[2] The transfer functions $G^i_{jk}(z)$ have no dependence on a_i, but may depend on a_ℓ, $\ell \neq i$. This framework encompasses treatment of multi-input, multi-output systems, but in this brief we view u and y as scalar signals.

Bingulac *et al.* studied the generation of sensitivity functions with respect to parameters in a finite dimensional, linear, time-invariant system [9]. They showed conditions under which sensitivity

[1] For an exception, see the lattice-form realization of [7].

[2] Typically, each parameter will appear only once. However, in IIR lattice models [8], the reflection coefficients appear twice.

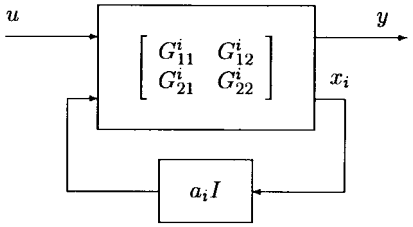


Fig. 2. Feedback gain model for parameter a_i.

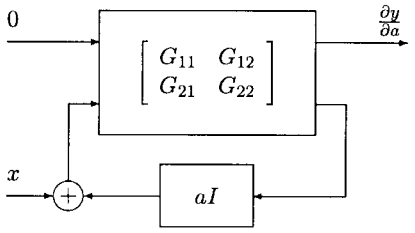


Fig. 3. Sensitivity generation for parameter a.

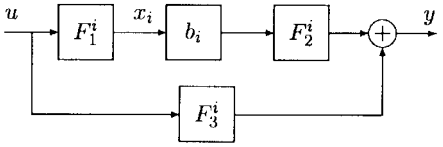


Fig. 4. Filter with feedforward parameter b_i.

functions for all parameters in an Nth-order, single input system $\mathcal{G}$ may be simultaneously generated by augmenting the system with N additional states in $\mathcal{S}$. Hence the composite system $(\mathcal{G},\mathcal{S})$ has dimension $2N$. Below we show that when all N poles of the system depend upon the adjustable parameters, N is a lower bound on the number of additional states necessary to generate sensitivities for all the parameters.

Theorem 1: Let $\mathcal{G}$ be representable as an FGM all of whose N poles depend upon the values of the parameters. Then to generate the sensitivity functions for all parameters, one must augment $\mathcal{G}$ with a system $\mathcal{S}$ having at least N states.

Proof: Let p be a pole of

$$\frac{Y(z)}{U(z)}$$

the u to y transfer function of $\mathcal{G}$, that is influenced by the value of a. By assumption, $\mathcal{G}$ is representable in the form of Fig. 2. For convenience of notation, we drop the i-dependence in G_{jk}^i, x_i and a_i as given in that figure. One may show that

$$x = [I - aG_{22}]^{-1}G_{21}u \tag{1}$$

$$y = G_{11}u + aG_{12}[I - aG_{22}]^{-1}G_{21}u. \tag{2}$$

Using results from [2], one may establish that

$$\frac{\partial y}{\partial a}$$

is generated as shown in Fig. 3. We see that

$$\frac{\partial y}{\partial a} = G_{12}[I - aG_{22}]^{-1}x \tag{3}$$

$$= G_{12}[I - aG_{22}]^{-1}[I - aG_{22}]^{-1}G_{21}u. \tag{4}$$

As p is a pole of

$$\frac{Y(z)}{U(z)}$$

depending on a, and each G_{jk} does not depend on a, we see from (2) that p must be a pole of $[I - aG_{22}]^{-1}$. Then from (4), we may conclude that the transfer function generating

$$\frac{\partial y}{\partial a}$$

from x must of necessity have p appear with twice the multiplicity it has in

$$\frac{Y(z)}{U(z)}.$$

Therefore, $\mathcal{S}$ must contain p with at least multiplicity one, to complement the occurrence of p in $\mathcal{G}$. Since the above fact holds true for all poles $p_1, \cdots, p_N$ of

$$\frac{Y(z)}{U(z)}$$

the system S must have at least N poles, and hence N states. □

Theorem 1 gives a lower bound on the additional states needed for sensitivity generation. Note that it does not state that this lower bound is achievable, and there may be situations where the minimum number of additional states that are required can exceed N. Furthermore, we are here interested in the *complexity* of the sensitivity generation, and not simply the dimensionality of the system.

Theorem 2: Let $\mathcal{G}$ be given by an FGM that can model an arbitrary Nth-order transfer function

$$\frac{Y(z)}{U(z)}$$

by choice of parameters $a_1, \cdots, a_m$. Then $(\mathcal{G},\mathcal{S})$ requires a minimum of $3N + 1$ multiplications.

Proof: In order to set the $2N+1$ degrees of freedom in an Nth-order transfer function, we require that the number of parameters m satisfies $m \geq 2N + 1$. Each a_i necessitates a multiplication in the implementation of $\mathcal{G}$. By Theorem 1, $\mathcal{S}$ must have at least N poles. The minimal number of additional multiplications required to implement these is N, yielding the total of $3N + 1$ multiplications as a minimum for $(\mathcal{G},\mathcal{S})$. □

We say that $\mathcal{G}$ has lowest complexity if there is an $\mathcal{S}$ such that $(\mathcal{G},\mathcal{S})$ contains only $3N + 1$ multiplications, where N is the order of $\mathcal{G}$ and assuming that the poles of $\mathcal{G}$ all have a dependence on the parameters. In such a case, $(\mathcal{G},\mathcal{S})$ is termed a lowest complexity implementation. The following development establishes structural requirements on $\mathcal{G}$ for it to have lowest complexity.

First, we examine necessary conditions on the way feedforward parameters enter $\mathcal{G}$ for it to have lowest complexity.

Lemma 1: Let $\mathcal{G}$ be an order N filter having feedfoward parameters $b_1, \ldots, b_{N+1}$, where b_i is a feedforward parameter if the dependence of $\mathcal{G}$ on b_i is representable as shown in Fig. 4. Then $\mathcal{G}$ has lowest complexity only if for each $i = 1, \ldots, N+1$, F_2^i contains no multiplications and hence has no parametric dependence.

Proof: Suppose that $(\mathcal{G},\mathcal{S})$ is a lowest complexity implementation. Let $(\bar{\mathcal{G}},\bar{\mathcal{S}})$ be the structure obtained by setting $b_i = 1$ for each i in $(\mathcal{G},\mathcal{S})$. Since $\bar{\mathcal{G}}$ retains N parameter dependent poles, $\bar{\mathcal{S}}$ has N multiplications. Thus, $\mathcal{S}$ has the same number of multiplications as $\bar{\mathcal{S}}$, and therefore cannot depend on $\{b_1, \ldots, b_{N+1}\}$.

With reference to Fig. 4, we observe that the sensitivity function for b_i is $F_2^i x_i$, so that $F_2^i x_i$ must be available from $(\mathcal{G},\mathcal{S})$ by virtue of $(\mathcal{G},\mathcal{S})$ being lowest complexity. Clearly $F_2^i x_i$ is not available in $\mathcal{G}$ unless $F_2^i = 1$, in which case F_2^i has no parametric dependence. Suppose instead that $F_2^i x_i$ is in $\mathcal{S}$, implying that F_2^i has been

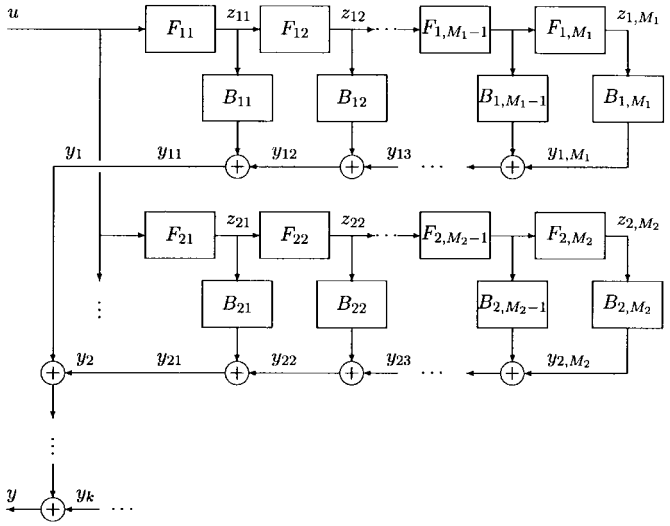


Fig. 5. Filter structure for lowest complexity implementations.

implemented within $\mathcal{S}$. We reason that in this case F_2^i cannot depend on any parameters. First, if F_2^i depends on a feedforward parameter, then $\mathcal{S}$ has such a dependence, which is not possible. Second, suppose F_2^i depends on a parameter that determines a pole location. The sensitivity with respect to that parameter will require replication of the pole that it influences, and will also be proportional to b_i. These two conditions together are incompatible with F_2^i being in $\mathcal{S}$: to avoid duplication of multiplications (which would result in excess of N multiplications in $\mathcal{S}$), the replication of the pole must occur within the realization of F_2^i producing $F_2^i x_i$, which forces a multiplication by b_i to occur in $\mathcal{S}$. □

Remark 1: In most filter configurations, the only interesting F_2^i satisfying the requirements of Theorem 1 is $F_2^i = 1$. In that case,

$$y(k) = \sum_{i=1}^{N+1} b_i x_i(k) + y_{\bar{b}}(k), \tag{5}$$

where $y_{\bar{b}}(k)$ does not depend on any of the b_i parameters. In situations were $F_2^i \neq 1$, it is always possible to modify the implementation to incorporate F_2^i into F_1^i, leaving the new F_2^i in the modification equal to unity. Also, in most cases, $y_{\bar{b}}(k) = 0$.

Theorem 3: If $\mathcal{G}$ is representable as shown in Fig. 5, then $\mathcal{G}$ is lowest complexity if and only if each B_{ij} and F_{ij} satisfy the following conditions. Each B_{ij} depends only on feedforward parameters, is linear in those parameters, and contains no further multiplications. Each F_{ij} depends only on feedback parameters, assembled in the vector $\mathbf{a}_{ij}$, and has minimum complexity. Furthermore, there must exist a minimum complexity implemention for F_{ij} such that the sensitivity of z_{ij} with respect to $\mathbf{a}_{ij}$ is generated as $S_{ij} z_{ij}$.

Proof: First we address sufficiency. As per Lemma 1, the conditions on each B_{ij} make available the sensitivities with respect to the feedforward parameters directly within B_{ij}, as for instance the x_i values in (5), or from signals within B_{ij} but without additional multiplications. The sensitivities with respect to the feedback parameters in F_{ij} may be constructed as $S_{ij} y_{ij}$, with the number of multiplications in S_{ij} equal to the dimension of $\mathbf{a}_{ij}$. This yields a lowest complexity implementation.

For necessity, we begin by noting that Lemma 1 establishes that all feedforward parameters in $\mathcal{G}$ must appear within the B_{ij} transfer functions in Fig. 5 and that the required conditions on B_{ij} must be satisfied. So, each F_{ij} must depend only on feedback parameters. If some F_{ij} is not lowest complexity, then neither is $\mathcal{G}$. We further

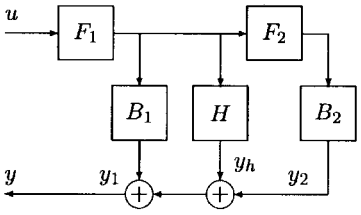


Fig. 6. Recursively nested lowest complexity structure.

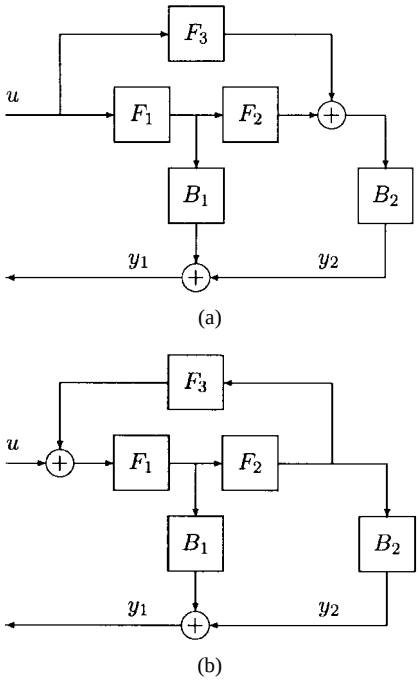


Fig. 7. Structures without the lowest complexity property. (a) Feedforward across pick-off point. (b) Feedback around pick-off point.

argue that the sensitivities with respect to $\mathbf{a}_{ij}$ must be available as $S_{ij} z_{ij}$ as claimed. Letting T_{ij} denote the transfer function between z_{ij} and y_{ij} in Fig. 5, we see that the sensitivity of y with respect to $\mathbf{a}_{ij}$ must include T_{ij} as a factor. To avoid replication of T_{ij} in $\mathcal{S}$, one must exploit the availability of T_{ij} in $\mathcal{G}$, and this is possible only if the sensitivity of z_{ij} with respect to $\mathbf{a}_{ij}$ is available as $S_{ij} z_{ij}$. This allows $S_{ij} y_{ij}$ to implement the sensitivity with respect to y, with T_{ij} appearing in the generation of y_{ij}. □

The question then arises as to whether structures other than that of Fig. 5 have lowest complexity. Technically, the answer is yes. For instance, one possibility is the arrangement of Fig. 6, where H is a system of the form of Fig. 5 with the properties demanded by Theorem 3. The key property of Fig. 5 that is preserved in this variation is that the effects of a given set of feedback parameters are isolated in one signal available in $\mathcal{G}$, and that the sensitivities with respect to those parameters can be obtained from that signal in a lowest complexity fashion. For instance, in Fig. 6, the effects of parameters in F_i are isolated in y_i, $i = 1, 2$, and the effects of parameters in H remain isolated from those in F_2 due to the parallel construction.

Some structures that are not lowest complexity are shown in Fig. 7. Here, the feedforward connection F_3 in Fig. 7(a) mixes the effects of both F_2 and F_3 in y_2, and the feedback connection F_3 in Fig. 7(b) mixes F_1, F_2, and F_3 in y_1. Connections such as these in the lattice filter of [1] and [2] prevent their being lowest complexity implementations.

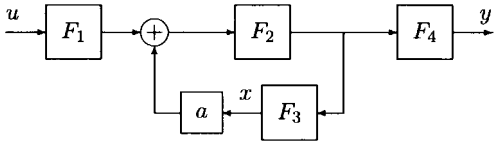


Fig. 8. Filter with cascaded feedback section.

III. Constructing Lowest Complexity Implementations

In order to construct lowest complexity implementations, we must first isolate the feedforward parameters in the B_{ij} transfer functions of Fig. 5. This is in general simplest to do by letting each B_{ij} be a tapped delay line of the form

$$\sum_{\ell=0}^{N} b_\ell q^{-\ell}. \tag{6}$$

We then need to develop lowest complexity building blocks to implement the F_{ij} in Fig. 5. For this purpose, we identify two key principles.

Cascaded Feedback Parameters Suppose that parameter a enters $\mathcal{G}$ in the fashion shown in Fig. 8. Such a parameter appears within feedback that is in cascade with the remainder of the u_a to y_a transfer function. To relate Figs. 2 and 8, we have $G_{11} = F_4F_2F_1$, $G_{12} = F_4F_2$, $G_{21} = F_3F_2F_1$, and $G_{22} = F_3F_2$. It is straightforward to show that

$$y_a = \frac{G_{11}}{1 + aG_{22}} u_a. \tag{7}$$

Using (1) and (4), we have

$$\frac{\partial y_a}{\partial a} = \frac{G_{12}G_{21}}{(1 + aG_{22})^2} u_a. \tag{8}$$

Noting that in this situation $G_{11}G_{22} = G_{12}G_{21}$, and taking into account (7), (8) becomes

$$\frac{\partial y_a}{\partial a} = \frac{G_{22}}{1 + aG_{22}} y_a. \tag{9}$$

Compare (3) to (9). In both we generate

$$\frac{\partial y}{\partial a}$$

from a signal obtained from $\mathcal{G}$ passed through a transfer function, but in (9) this transfer function depends only upon the local dynamics $G_{22} = F_3F_2$, while (3) depends as well on $G_{21} = F_4F_2$, which includes a potentially complex term F_4. Furthermore, the sensitivity generation of (9) is accomplished by filtering the output. If the system of Fig. 8 represents one of the F_{ij} blocks in Fig. 5, then this manner of sensitivity generation satisfies one of the requirements of Theorem 3.

Delay: If for two parameters a_i and a_j we have $x_j(k) = x_i(k - \Delta)$, then

$$\frac{\partial y}{\partial a_j}(k) = \frac{\partial y}{\partial a_i}(k - \Delta).$$

In $\mathcal{S}$, we need implement only

$$\frac{\partial y}{\partial a_i}$$

and construct

$$\frac{\partial y}{\partial a_j}$$

as a delayed version of

$$\frac{\partial y}{\partial a_i}$$

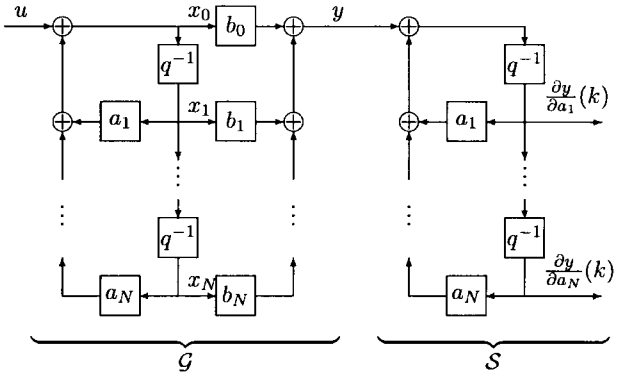


Fig. 9. Direct-form II filter, $\mathcal{G}$ and $\mathcal{S}$ portions.

.

One may construct a prototypical building block F_{ij} exploiting these two features as follows. Let the u_a to y_a transfer function of Fig. 8 be

$$\frac{1}{1 - \sum_{\ell=1}^{N} a_\ell q^{-\ell}}. \tag{10}$$

With respect to Fig. 8, let $a = a_1$, $F_1 = 1$, $F_2 = 1/\sum_{i=2}^{N} a_i q^{-i}$, $F_3 = q^{-1}$, and $F_4 = 1$. We then have

$$\frac{\partial y_a}{\partial a_1}(k) = \frac{F_3F_2}{1 + a_1F_3F_2} y_a(k) = \frac{q^{-1}}{1 - \sum_{i=1}^{N} a_i q^{-i}} y_a(k) \tag{11}$$

requiring N multiplications to implement. For $i = 2, \cdots, N$, we exploit the delay relationships and set

$$\frac{\partial y_a}{\partial a_i}(k) = \frac{\partial y_a}{\partial a_1}(k - i + 1)$$

.

In the context of Fig. 5, with F_{ij} given by (10), then the sensitivity of y with respect to the parameters in F_{ij} is generated in the same way, but with the operator in (11) acting on y_{ij} in Fig. 5, as discussed in the proof of Theorem 3.

Thus, if each F_{ij} in Fig. 5 is of the form of (10), and each B_{ij} is of the form (6), we then have a lowest complexity implementation.

IV. Examples

Lowest complexity implementation of direct-form, cascade-form, and parallel-form are demonstrated below. As noted previously, the lattice-form is not known to admit a lowest complexity implementation. Other implementations can be checked for the possibility of lowest complexity sensitivity generation by comparing them with the form of Fig. 5.

A. Direct-Form

A direct-form II implementation of an Nth-order IIR filter is shown in Fig. 9. This filter is essentially the configuration of Fig. 5 with only F_{11} and B_{11} as nonzero transfer functions, and with these implemented as (10) and (6), respectively. The prototypical sensitivity generation for the denominator (feedback) parameters is shown in Fig. 9. Note the total of $3N + 1$ multiplications in $(\mathcal{G}, \mathcal{S})$.

The direct-form I implementation, which is that typically given in adaptive filtering texts, requires filtering of the inputs by the same operator appearing in (11) in order to construct sensitivities for the numerator parameters. The total number of multiplications becomes $4N + 2$ (again exploiting delay relationships). By collapsing the states of direct-form I into direct-form II, we obtain linearity in the numerator parameters, as required, and the consequent lowest complexity property follows.

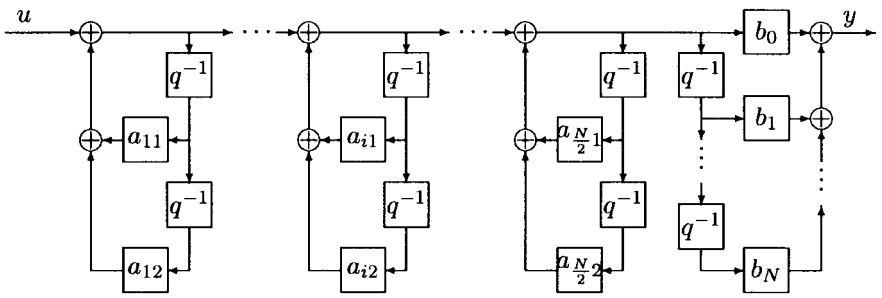


Fig. 10. Cascade-form filter, $\mathcal{G}$-portion only.

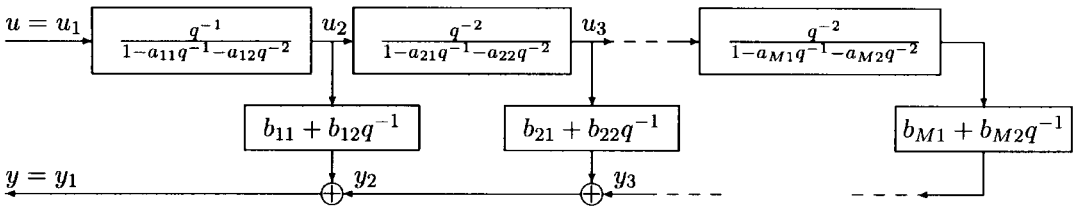


Fig. 11. Tapped cascade-form filter.

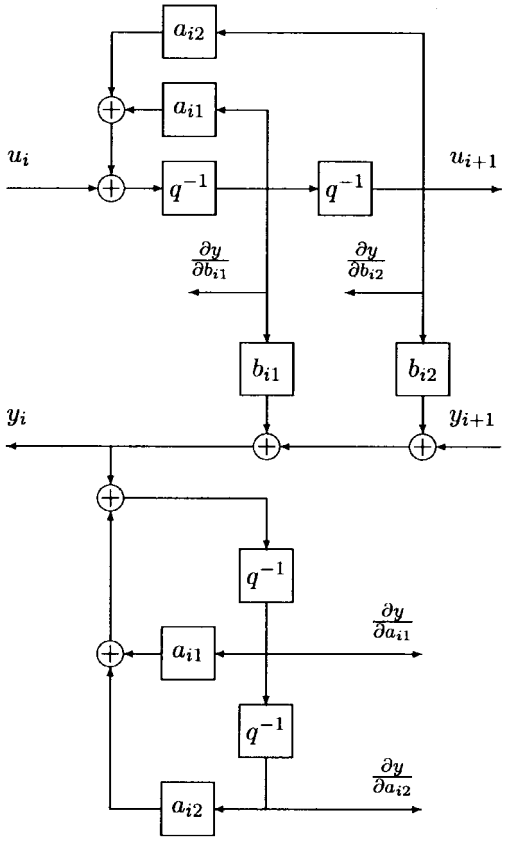


Fig. 12. Tapped cascade, ith section with sensitivitygeneration.

B. Cascade-Form

In [5], Rao proposed the cascade-form implementation of Fig. 10. We can interpret this via Fig. 5 by noting that the ith second-order section implementing two poles corresponds to F_{1i}, with $B_{1,\frac{N}{2}}$ the $(N+1)$-order tapped delay and all other $B_{1i} = 0$. The sensitivities with respect to the a_{ij} parameters are implemented in a fashion similar to $\mathcal{S}$ in Fig. 9, with

$$\frac{\partial y}{\partial a_{i1}}(k) = \frac{q^{-1}}{1 - a_{i1}q^{-1} - a_{i2}q^{-2}} y(k).$$

Noting that

$$\frac{\partial y}{\partial a_{i2}}$$

is obtained from

$$\frac{\partial y}{\partial a_{i1}}$$

via the delay relationship, we see that sensitivity generation for this section requires two multiplications. The same is true for the other $(N/2) - 1$ sections, for a total of N. With the tapped delay line implemented at the end of the cascade, its parameters enter linearly, so no additional multiplications are need to yield those sensitivities. The total of the multiplications comes to $3N + 1$, indicating that this is a lowest complexity adaptive IIR filter structure.

C. Parallel-Form

A lowest complexity parallel-form realization may be readily constructed from a parallel combination of lowest complexity direct-form II implementations of second-order sections. With respect to Fig. 5, we would implement F_{i1} as (10) with $N = 2$, and $B_{11} = b_{10} + b_{11}q^{-1} + b_{12}q^{-2}$ and $B_{i1} = b_{i1}q^{-1} + b_{i2}q^{-2}$ for $i = 2, \ldots, N/2$. All F_{ij} and B_{ij} with $j \geq 2$ are set to zero, so we have a basic parallel connection in Fig. 5. The output is of course linear in the numerator parameters of all stages, so their sensitivities are available in $\mathcal{G}$. The sensitivities of the denominator parameters for each parallel section are computed as for the direct-form II. Each section thus contributes four multiplies to implement, plus two multiplies for sensitivity generation, for a total of six. Multiply by $N/2$ to sum the multiplies for all sections, and add one multiply for the one direct feedthrough parameter, for a total of $3N + 1$ multiplications.

D. Tapped Cascade-Form

A novel implementation structure having lowest complexity can be developed from Fig. 5 as follows. Let

$$F_{i1}(q^{-1}) = \begin{cases} \frac{q^{-1}}{1-a_{i1}q^{-1}-a_{i2}q^{-2}}, & i = 1 \\ \frac{q^{-2}}{1-a_{i1}q^{-1}-a_{i2}q^{-2}}, & i = 2, \ldots, M \end{cases} \tag{12}$$

and

$$B_{i1}(q^{-1}) = b_{i0} + b_{i1}q^{-1}$$

and assume that $B_{ij}(q^{-1}) = 0$ for $j > 1$. The principles of Section III indicate that such a structure has lowest complexity. These choices interpret Fig. 5 as a tapped cascade of second-order sections, as shown in Fig. 11. It is shown in [10] that such a structure is able to represent an arbitrary strictly proper transfer function of order $2M$. A proper transfer function may be realized by including a tap directly between u and y.

To implement the ith section, consisting of F_{i1} and the subsequent "tap" transfer function B_{i1}, in lowest complexity form, we apply the concepts given in Section 3. In particular, the sensitivities for the parameters in B_{i1} will be available in its tapped delay line, while the sensitivities for the parameters in F_{i1} can be constructed as discussed below (10). One must be careful, however, to apply the filtering operation of $1/(1 - a_{i1}q^{-1} - a_{i2}q^{-2})$ that is used in the sensitivity generation only to the part of the output y that is influenced by the parameters in the i section. This is the reason why the signals from the taps are summed from right to left in Fig. 11 (as is done for the outputs of B_{ij} in Fig. 5).

The resulting structure showing both the ith section itself and also its associated sensitivity generations is given in Fig. 12. Note that the additional delays present in F_{i1} in (12) do not modify this construction. Notice also that only two additional multiplies occur in the sensitivity generation, indicating the lowest complexity characteristic.

V. Conclusion

We have examined in this brief the problem of implementing adaptive IIR filters with lowest complexity, as measured by the number of multiplications used to generate the filter output and additionally the sensitivities with respect to all adapted parameters. We have shown that for an order N filter, the minimum number of such multiplications is $3N + 1$. We outlined some strategies for obtaining a lowest complexity implementation, and applied these to direct-, cascade-, and parallel-form implementations.

References

[1] D. Parikh, N. Ahmed, and S. Stearns, "An adaptive lattice algorithm for recursive filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 110–111, Feb. 1980.

[2] G. A. Williamson, C. R. Johnson Jr., and B. D. O. Anderson, "Locally robust identification of linear systems containing unknown gain elements with application to adapted IIR lattice models," *Automatica*, vol. 27, pp. 783–798, May 1991.

[3] J. A. Rodriguez-Fonollosa and E. Masgrau, "Simplified gradient calculation in adaptive IIR lattice filters," *IEEE Trans. Signal Processing*, vol. 39, pp. 1702–1705, July 1991.

[4] N. Nayeri and W. K. Jenkins, "Alternate realizations of adaptive IIR filters and properties of their performance surfaces," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 485–496, Apr. 1989.

[5] B. D. Rao, "Adaptive IIR filtering using cascade structures," in *Proc. 27th Asilomar Conf. Signals, Syst., and Comput.*, Nov. 1993, Pacific Grove, CA, pp. 194–198.

[6] L. B. Jackson and S. L. Wood, "Linear prediction in cascade form," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 26, pp. 518–528, Dec. 1978.

[7] P. A. Regalia, "Stable and efficient lattice algorithms for adaptive IIR filtering," *IEEE Trans. Signal Processing*, vol. 40, pp. 375–388, Feb. 1992.

[8] A. H. Gray Jr. and J. D. Markel, "Digital lattice and ladder filter synthesis," *IEEE Trans. Audio Electroacoust.*, vol. 21, pp. 491–500, Dec. 1973.

[9] S. Bingulac, J. H. Chow, and J. R. Winkelman, "Simultaneous generation of sensitivity functions—Transfer function matrix approach," *Automatica*, vol. 24, pp. 239–242, Feb. 1988.

[10] G. A. Williamson and S. Zimmermann, "Globally convergent adaptive IIR filters based on fixed pole locations," *IEEE Trans. Signal Processing*, vol. 44, pp. 1418–1427, June 1996.

On the Common Mode Rejection Ratio in Low Voltage Operational Amplifiers with Complementary N–P Input Pairs

Fan You, Sherif H. K. Embabi, and Edgar Sánchez-Sinencio

***Abstract*—Low voltage op amps with complementary N–P input differential pairs are known to suffer from low common mode rejection ratio due to mismatch errors and the tail current switching between the N and P input stage. To understand the contribution of the systematic and the random common mode gains to the overall common mode rejection ratio (CMRR) we studied three op amp topologies, which use N–P complementary input differential pairs. A detailed small signal analysis for each of them has been performed to compare their systematic and random CMRR. The analysis shows that random CMRR caused by mismatch does not depend on the topology, while the systematic CMRR is topology dependent. It is also concluded that the CMRR of low voltage op amps with N–P complementary input pairs will be ultimately limited by the process mismatch and that the random CMRR will determine the overall CMRR.**

***Index Terms*— Common mode rejection ratio (CMRR), low voltage, operational amplifier.**

I. Introduction

There is a strong demand for lowering the supply voltage of analog circuits including op amps. To increase the signal to noise ratio of low voltage op amps, it is highly desirable to have a rail-to-rail input voltage swing. N–P complementary pairs have been widely used in the input stage of low voltage op amps to achieve a rail-to-rail input voltage swing [1]–[8]. An advantage of using N–P complementary differential pairs is that the op amps can be implemented in a standard digital process. Fig. 1 shows a typical structure of a low voltage op amp with N–P differential pairs. Using N–P complementary input pairs will, however, degrade the common mode rejection ratio (CMRR). This occurs while the tail current switches between the P and N pairs. A CMRR as low as 40–55 dB has been reported in [4], [6], and [7]. This brief presents a rigorous analysis of the CMRR of low voltage op amps with N–P differential pairs. Three illustrative topologies have been considered here. In Section II, a derivation of the CMRR of the three op amp topologies with complementary N–P pairs is presented. In Section III, we compare the systematic and random CMRR of the different topologies. The random CMRR is compared with the systematic CMRR in Section IV, to find which

Manuscript received October 31, 1995; revised April 5, 1996. This paper was recommended by Associate Editor F. Larsen.

F. You was with the Department of Electrical Engineering, Texas A&M University, College Station, TX 77843 USA. He is now with Bell Laboratories, Lucent Technologies, Allentown, PA 18103 USA.

S. H. K. Embabi and E. Sánchez-Sinencio are with the Department of Electrical Engineering, Texas A&M University, College Station, TX 77843 USA.

Publisher Item Identifier S 1057-7130(97)03654-9.

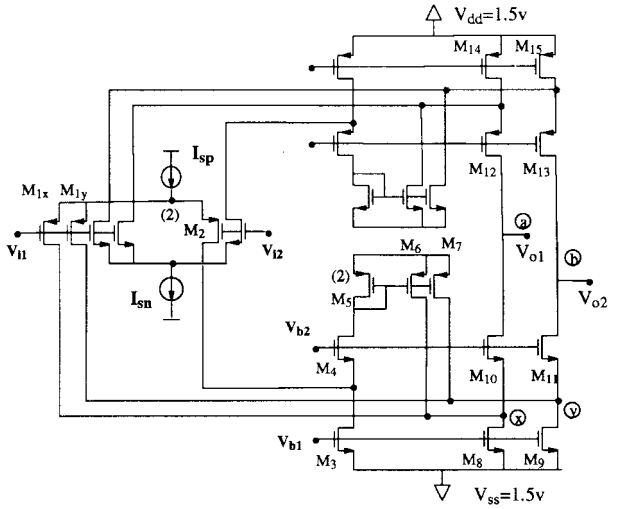


Fig. 1. An N–P complementary input stage with common mode cancellation proposed in [1]—Topology I.

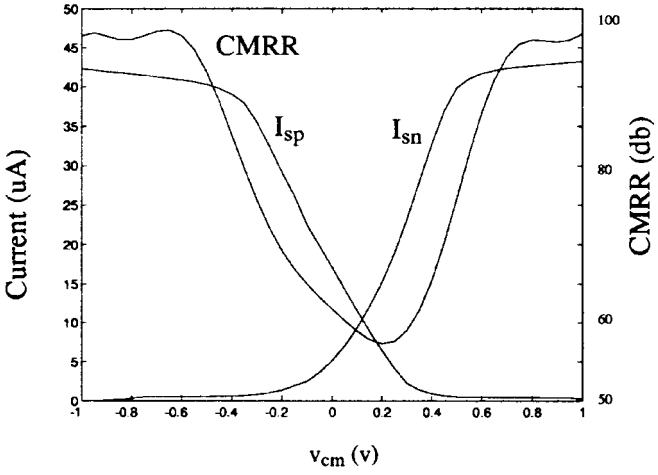


Fig. 2. CMRR and tail current I_{sn} and I_{sp} versus $V_{\rm cm}$.

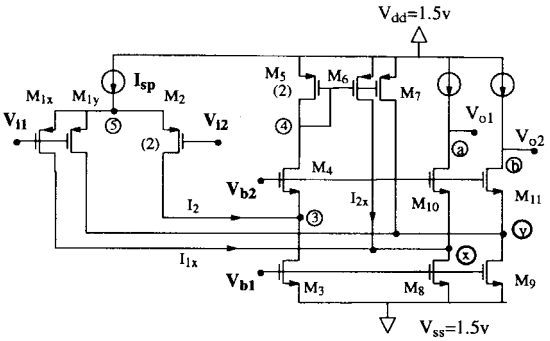


Fig. 3. The half of the amplifier of Fig. 1 used for small signal analysis.

of them determines the overall CMRR for each of the topologies under consideration. Finally, we verify the results of the analysis with simulation and study the CMRR as a function of frequency.

II. Derivation of Common Mode Gain of the Low Voltage Op Amp

The use of N–P complementary input pairs to achieve rail to rail input swing may result in a variable transconductance of the input stage—a property which severely affects the optimal compensation of the op amp. In order to make the overall g_m constant, the tail currents I_{sn} and I_{sp} (in Fig. 1) are generated using a square-root current source which maintains the sum of the square root of both currents constant ($\sqrt{I_{sn}}+\sqrt{I_{sp}}=$ constant) [1], [4]. If the input devices of the differential pair operate in the weak inversion region, a current source which maintains the sum of the two tail currents constant ($I_{sn}+I_{sp}=$ constant) [5], [6] is used to achieve a constant g_m. The tail currents I_{sp} and I_{sn} are, however, dependent on the common mode input voltage ($V_{\rm cm}$) as illustrated in Fig. 2. Both currents exhibit sharp changes in magnitude as the tail current switches between the N and the P pair. Although the g_m may be constant, the CMRR is not. Fig. 2 shows the simulation result of using a constant-g_m input stage with a square-root current source. A drop of at least 35 dB in the CMRR can be observed. For the N-channel input stage, as $V_{\rm cm}$ is lowered toward V_{ss}, the NMOS which is acting as the current source is pushed into triode region. This means that the resistance of the current source decreases and that the common mode gain increases. If $V_{\rm cm}$ is further lowered, the N-pair is completely turned off and it will not contribute to the overall common mode gain. A similar explanation applies for the P stage, and we will have an increase in common mode gain when either current source operates in the triode region.

In the following subsections, we present a detailed analysis of the small signal differential and common mode gains of three op amp topologies. All three have N–P complementary differential pairs. The second stage is different for each topology. The first and the second topology (Figs. 1 and 5) have been reported in [1]–[3], respectively. We are proposing a third topology (Fig. 6) which is basically an improved version of the second topology shown in Fig. 5. For each circuit, we will derive the systematic CMRR, which is topology dependent, and the random CMRR, which is a function of the process mismatching.

A. Topology I

The circuit topology shown in Fig. 1 has a special circuit (M_4–M_7) whose function is to cancel the common mode current resulting from the change of the tail current [1]. For the common mode gain analysis, we will consider only one input pair as shown in Fig. 3. For the N–P complementary input amplifiers, the overall small signal gain is simply the summation of the gains of the two input pairs. The tail current I_{sp} in the figure is assumed to be generated by a constant-g_m current biasing circuit. To maintain generality, we use a generic model for the tail current generator I_{sp} in the small signal analysis. Since the value of the tail current is dependent on the common mode input voltage $v_{\rm cm}$, we may use a voltage controlled current source $G_{ms}v_{\rm cm}$ as its ac model. Note that G_{ms} is a function of the dc common mode input voltage. The finite output conductance of the I_{sp} current source is also accounted for through the use of g_{os} as shown in Fig. 4. The conductance seen through the source of M_{10} and M_{11} in Fig. 3 has been modeled as g_{ex} and g_{ey} as shown in Fig. 4. The conductance g_{ex} and g_{ey} are fairly low because the resistance of the loads connected to nodes a and b are very large. It can be shown that g_{ex} and g_{ey} are in the order of g_o ($1/r_{ds}$) and not g_m as expected for small load resistance [9].

Note that the following analyses are carried out for the range of $V_{\rm cm}$ where the tail currents (I_{sp} and I_{sn}) are switching. It is in this current transition range where the CMRR becomes minimum.

In the CMRR analysis the input voltages v_{i1} and v_{i2} are usually expressed as functions of the differential and common mode inputs:

$$v_{i1} = v_{\rm cm} - \tfrac{1}{2}\, v_{\rm dm} \tag{1a}$$

$$v_{i2} = v_{\rm cm} + \tfrac{1}{2}\, v_{\rm dm}. \tag{1b}$$

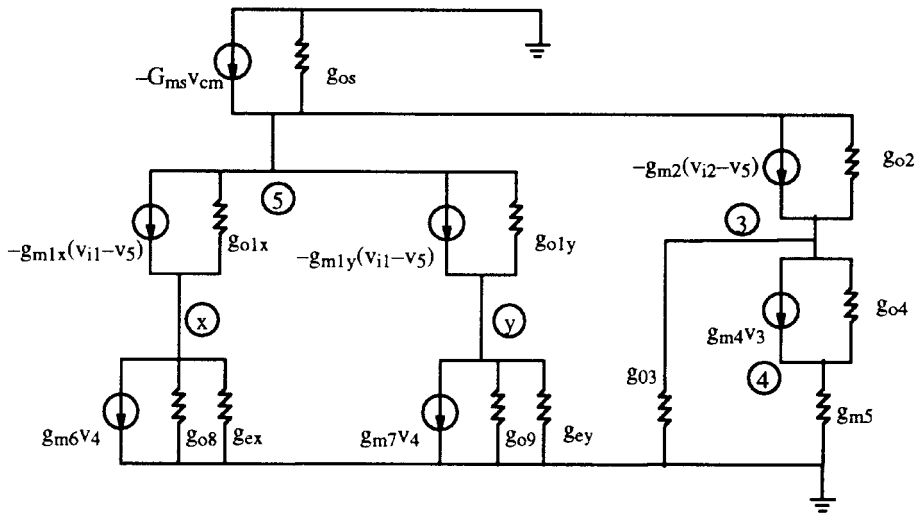


Fig. 4. Low frequency small signal model of the circuit shown in Fig. 3.

To simplify the analytical computation, the small signal model of the amplifier shown in Fig. 4 is used to derive the nodal equations at nodes x and y. If the node voltages v_x and v_y could be expressed in terms of the common mode gain ($A_{\rm cm}$) and the differential mode gain ($A_{\rm dm}$) as follows:

$$v_x = A_{\rm dm} v_{\rm dm} + A_{cm} v_{\rm cm} \tag{2}$$

the CMRR can then be obtained by calculating ($A_{\rm dm}/A_{\rm cm}$). Note that such a simplification in the small signal model will not affect the accuracy of the CMRR analysis since the CMRR at output nodes a or b is the same as the CMRR at nodes x or y. This is due to the fact that both differential and common mode voltage signals at nodes x (or y) will be amplified by the gain of the common gate configuration of M_{10} (or M_{11}).

For the circuit topology in Fig. 1, the matching of the differential pairs and that of the current mirrors are crucial for the performance of the amplifier. As an example to demonstrate how mismatching affects the common mode gain, we only consider the mismatching between M_{1x} (or M_{1y}) and M_2. Hence, we can make the following assumptions

$$\begin{aligned} g_{m2} &= 2(1+\epsilon) g_{m1x} \\ g_{o2} &= 2 g_{o1x} \\ g_{m1x} &= g_{m1y} \\ g_{o1x} &= g_{o1y} \\ g_{m6} &= g_{m7} \\ &= \tfrac{1}{2}\, g_{m5} \\ g_{ex} &= g_{ey} \\ g_{o8} &= g_{o9}. \end{aligned}$$

The mismatching in the output conductance of M_{1x} (or M_{1y}) and M_2 is ignored because of its little significance on the analysis result. The factor ϵ may account for mismatching in the sizing, V_T, K_p, etc. Based on (1), (2) and the assumptions above, the differential gain ($A_{\rm dm}$) and common mode gain ($A_{\rm cm}$) can be solved for by using MAPLE [10]. By ignoring the second order terms in the numerator and denominator the following expressions for $A_{\rm dm}$ and $A_{\rm cm}$ can be obtained:

$$A_{\rm dm} \approx \frac{g_{m2}}{2(g_{ex} + g_{o1x} + g_{o8})} \tag{3}$$

$$A_{\rm cm} = A_{\rm cms} + A_{\rm cmr} \tag{4}$$

where

$$A_{\rm cms} \approx \frac{(g_{o2} + g_{o3})\, G_{ms}}{4 g_{m4}\, (g_{ex} + g_{o1x} + g_{o8})} \tag{5a}$$

and

$$A_{\rm cmr} \approx \frac{\epsilon G_{ms}}{4(g_{ex} + g_{o1x} + g_{o8})}. \tag{5b}$$

The first term of (4) will be referred to as Systematic Common Mode Gain ($A_{\rm cms}$), since it is independent of the mismatching. The second term is a function of the mismatching and, hence, will be called the Random Common Mode Gain ($A_{\rm cmr}$). Now we can express the CMRR in terms of the systematic and random common mode rejection ratios which are given by

$$\text{CMRR}_s \approx \frac{2 g_{m2}\, g_{m4}}{G_{ms}\,(g_{o2} + g_{o3})} \tag{6}$$

$$\text{CMRR}_r \approx \frac{2 g_{m2}}{\epsilon G_{ms}}. \tag{7}$$

The overall CMRR is given by

$$\text{CMRR} = \frac{1}{\dfrac{1}{\text{CMRR}_r} + \dfrac{1}{\text{CMRR}_s}}. \tag{8}$$

B. Topology II

To reduce the systematic common mode gain of Topology I, one can use the circuit topology in [2] and [3] which is illustrated in Fig. 5. This will be discussed in Section III. Following the same procedure used to analyze the circuit in Fig. 1, we obtained the following systematic and random CMRR's:

$$\text{CMRR}_s \approx \frac{2 g_{m2}\, g_{m4}\, g_{m6}}{G_{ms}\, g_{o6}\, (g_{o2} + g_{o4})} \tag{9}$$

$$\text{CMRR}_r \approx \frac{2 g_{m2}}{\epsilon G_{ms}}. \tag{10}$$

C. Topology III

The systematic CMRR of the second topology can be further improved by introducing an extra gain stage A_b, which for example can be implemented using a simple noninverting amplifier, with Miller compensation, as shown in Fig. 6. Care should be taken to insure that this added stage will not degrade the high frequency performance of the amplifier. It can be proven that the CMRR

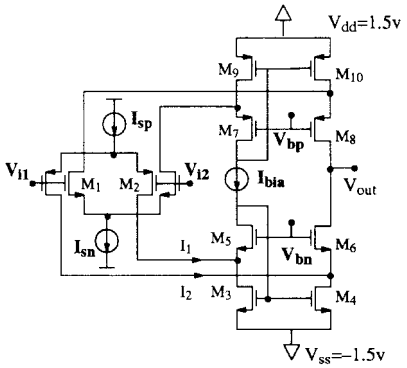


Fig. 5. An op amp with N–P complementary differential pairs [2], [3]—Topology II.

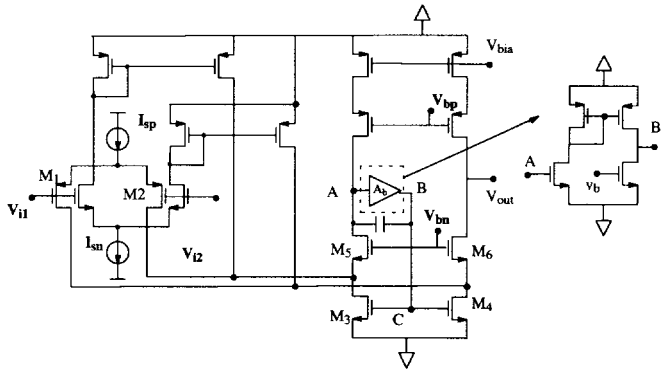


Fig. 6. A low voltage amplifier with systematic CMRR enhancement—Topology III.

improves by A_b as shown by the following equation:

$$\text{CMRR}_s \approx A_b \frac{2g_{m2}\, g_{m4}\, g_{m6}}{G_{ms}\, g_{o6}\, (g_{o2}+g_{o4})}. \tag{11}$$

The random CMRR, however, remains unchanged and is given by

$$\text{CMRR}_r \approx \frac{2g_{m2}}{\epsilon G_{ms}}. \tag{12}$$

III. Comparison of the CMRR of the Three Topologies

Although each of the topologies has a scheme for systematic common mode current cancellation, yet, the accuracy of the cancellation varies. In the first topology (Fig. 3), the common mode current I_{1x} is supposed to be cancelled out through I_{2x} which is half of the common mode current I_2. This is only true if all of I_2 is injected into M_4. Due to the finite conductance (g_{o3}) of M_3, I_{2x} will be slightly less than $I_2/2$. Thus the cancellation is not exact even if the mirror transistors M_5 and M_6 are perfectly matched. In the case of the second topology (Fig. 5), if we assume perfect matching between M_3 and M_4, it can be easily seen that the common mode current I_2 will be exactly cancelled by the mirror of I_1. A similar explanation can be given for Topology III. It is, hence, expected that the common mode cancellation of Topologies II and III is more accurate than that of Topology I. This implies that the systematic CMRR of II and III will be superior to that of I which is confirmed by the analytical expressions derived in Section II and summarized in Table I.

The similarity of CMRR_r of the three different topologies can be explained by using a more general amplifier model which is illustrated in Fig. 7. In the figure, the block "$I_2 - I_1$" is an abstract model for the cancellation of the common mode current due to G_{ms}. A common mode input (v_{cm}) will generate a tail current $I_{sp} = G_{ms}v_{\text{cm}}$.

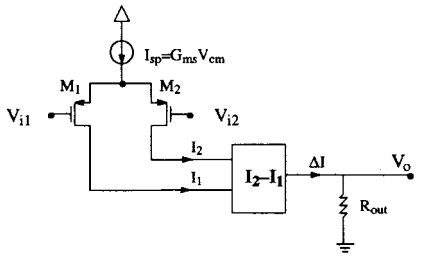


Fig. 7. A general amplifier model with tail current variation and common mode cancellation.

Assuming that there is mismatch (ϵ) in the input differential pair [i.e., $g_{m2} = g_{m1}(1+\epsilon)$], the current in the two input transistors could be expressed as $I_1 = G_{ms}v_{\text{cm}}/2$ and $I_2 = G_{ms}v_{\text{cm}}(1+\epsilon)/2$. The difference between the currents of the two input transistors due to $G_{ms}v_{\text{cm}}$ can be written as: $\Delta I_{\text{cm}} = \epsilon G_{ms}v_{\text{cm}}/2$. A differential input v_{dm} will otherwise generate the following current difference: $\Delta I_{\text{dm}} = g_m v_{\text{dm}}$, where g_m is the transconductance of the input pair. Since the differential and common mode output voltages are given by $\Delta I_{\text{dm}} R_{\text{out}}$ and $\Delta I_{\text{cm}} R_{\text{out}}$, respectively, we can derive the following generic expression for the random CMRR:

$$\text{CMRR}_r = \frac{A_{\text{dm}}}{A_{\text{cm}}} = \frac{2g_m}{\epsilon G_{ms}}$$

which is the same as (7), (10), and (12). This simple analysis confirms that the CMRR_r is topologies independent.

IV. Comparison Between Systematic CMRR and Random CMRR

It is interesting to note that both the systematic and the random common mode rejection ratios are reciprocally proportional to the common mode transconductance (G_{ms}). To compare between CMRR_s and CMRR_r, we first need to compare the magnitude of G_{ms} with that of g_o's and g_m's. G_{ms} is the rate of change of I_{sp} (or I_{sn}) when I_{sp} and I_{sn} are switching. The expression of G_{ms} is $I_{\max}/(V_{dd}-V_{ss})\alpha$, where $I_{\max}$ is the maximum value of I_{sp} (or I_{sn}) and α is typically 0.5 or less (see Fig. 2). The typical value of g_o is in the order of $\lambda I_{\max}$. Hence

$$\frac{G_{ms}}{g_o} \approx \frac{1}{\lambda\alpha(V_{dd}-V_{ss})}.$$

For $\lambda = 0.01$ and $V_{dd} = -V_{ss} = 1.5$ V and $\alpha = 0.5$, $G_{ms}/g_o \approx 67$. G_{ms} and g_m are of comparable magnitudes. So, we may assume that

$$g_o < G_{ms} \le g_m. \tag{13}$$

Let us first ignore the mismatching. The minimum common mode rejection is determined by the systematic common mode gain. Using the inequality (13) we can determine the order of the CMRR_s for all three topologies as shown in Table I. For Topology I, the CMRR_s is in the order of $g_m^2/G_{ms}g_o$ (25–35 dB), the CMRR_s of the second topology is in the order of $g_m^3/G_{ms}g_o^2$ (50–70 dB). The third topology may have a CMRR_s of the order of 70–95 dB. For a typical mismatching factor (ϵ) less than 1% [11], the CMRR_r is close to 40–60 dB.

For all three topologies, we now compare the CMRR_s with CMRR_r to evaluate which of the two components limit the improvement of CMRR. The ratios of $\text{CMRR}_s/\text{CMRR}_r$ for all topologies is summarized in Table I. For the first topology (Fig. 1), the ratio is less than unity, which implies that the overall CMRR will be determined by the low systematic CMRR. In the case of the second topology, the CMRR_s approaches the CMRR_r. As for the third topology, the

TABLE I
COMPARISON OF RANDOM AND SYSTEMATIC CMRR

CMRR / Topology	I	II	III
CMRR_s	$\frac{2g_{m2}g_{m4}}{G_{ms}(g_{o2}+g_{o3})}$ $O\left(\frac{g_m^2}{G_{ms}g_o}\right) \approx 25$–$35$ dB	$\frac{2g_{m2}g_{m4}g_{m6}}{G_{ms}g_{o6}(g_{o2}+g_{o4})}$ $O\left(\frac{g_m^3}{G_{ms}g_o^2}\right) \approx 50$–$75$ dB	$\frac{2A_bg_{m2}g_{m4}g_{m6}}{G_{ms}g_{o6}(g_{o2}+g_{o4})}$ $O\left(\frac{A_bg_m^3}{G_{ms}g_o^2}\right) \approx 70$–$95$ dB
CMRR_r	$\frac{2g_{m2}}{\epsilon G_{ms}}$ $O\left(\frac{2}{\epsilon}\right) \approx 40$–$60$ dB	$\frac{2g_{m2}}{\epsilon G_{ms}}$ $O\left(\frac{2}{\epsilon}\right) \approx 40$–$60$ dB	$\frac{2g_{m2}}{\epsilon G_{ms}}$ $O\left(\frac{2}{\epsilon}\right) \approx 40$–$60$ dB
$\frac{\text{CMRR}_s}{\text{CMRR}_r}$	$\frac{\epsilon g_{m2}}{(g_{o2}+g_{o3})} < 1$	$\frac{\epsilon g_{m4}g_{m6}}{g_{o6}(g_{o2}+g_{o4})} \geq 1$	$\frac{A_b\epsilon g_{m4}g_{m6}}{g_{o6}(g_{o2}+g_{o4})} > 1$

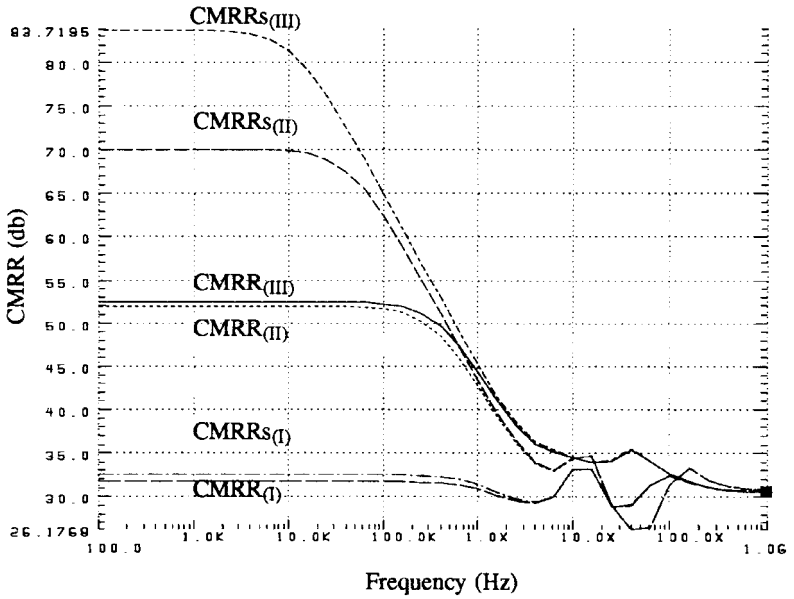


Fig. 8. CMRR versus frequency with and without mismatching.

systematic CMRR exceeds the random CMRR, hence, the overall CMRR will be determined by the CMRR_r.

V. SIMULATION RESULTS

To verify the results of the above analysis, the CMRR of all the three topologies has been simulated using HSPICE. The three amplifiers were designed to have the same gain bandwidth product of 3 MHz with 10 pF of capacitive load and the same low frequency differential gain. First the systematic CMRR was simulated assuming no mismatches. The result of the simulations are depicted in Fig. 8. Note that the curves denoted as CMRR_s (I), CMRR_s (II), and CMRR_s (III) represent the systematic CMRR for Topologies I, II, and III, respectively. From these three curves we can make the following observations. First, the low frequency CMRR_s of Topology I is the lowest with 32 dB, the CMRR_s of Topology II is 70 dB, and that of Topology III is the largest with 84 dB. These numbers agree with the theoretical analysis (see Table I). The second observation is that the systematic CMRR of Topologies II and III drops beyond 10 kHz, but is still greater than that of Topology I even at 3 MHz. The advantage of II and III over I in terms of CMRR_s, however, gradually diminishes as the frequency increase. To study the effect of mismatching, the simulation was performed with 2% mismatch in the input pair. The simulated total CMRR, which includes systematic and random CMRR, is also shown in Fig. 8 as CMRR(I), CMRR (II) and CMRR (III). It is interesting to note that Topologies II and III have similar CMRR which is smaller than their systematic CMRR. This confirms that the CMRR of these two topologies will be limited by the random CMRR which is equal for all three topologies. As for the first amplifier, the CMRR_s is smaller than CMRR_r and therefore the total CMRR is slightly smaller than CMRR_s.

The above theoretical analysis and simulation all confirm that the systematic CMRR can be improved through topology modification. By doing that, the random common mode gain becomes the ultimate factor to determine the overall CMRR. The effect of mismatching on the simulated CMRR for the circuits in Figs. 5 and 6 is illustrated in Fig. 9. It is observed that the CMRR of the circuit in Fig. 6 is much greater than that in Fig. 5, when the mismatching is small (below 0.1%). However, this is hardly realizable in practical amplifiers. The topology with the systematic CMRR enhancement is useful only if the transistor matching is very good. It is also observed from the figure that both circuits have similar CMRR when the matching is poor since the typical mismatching factor (ϵ) is in the order of 0.1% or more. It is expected that mismatching will be the dominant factor in determining the CMRR.

VI. CONCLUSION

In this brief, the CMRR degradation problem in low voltage op amps with N–P complementary pairs is discussed. A small signal

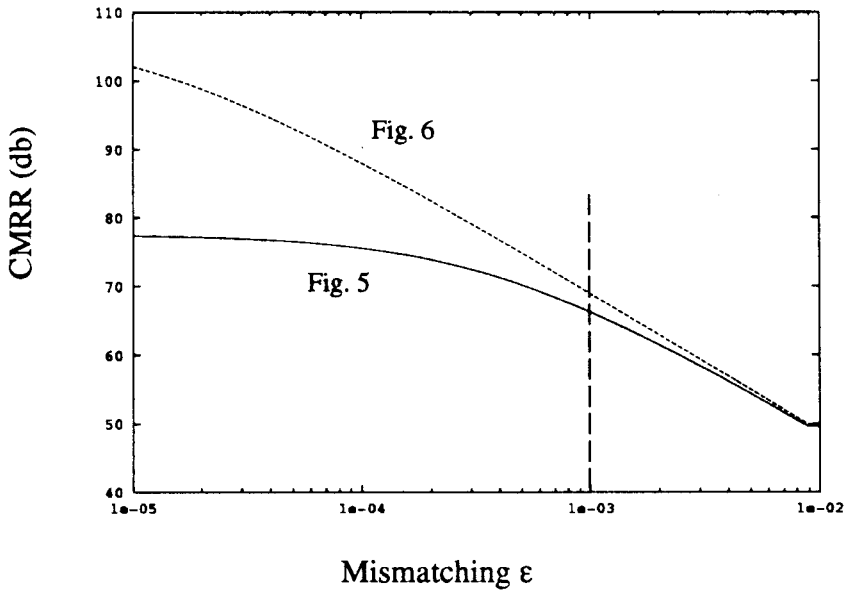


Fig. 9. Effect of mismatching on the CMRR.

analysis revealed that the increase of both systematic and mismatching common mode gain in the low voltage op amp is due to the change of the tail current of the N–P complementary pairs. The systematic CMRR degradation can be improved by using suitable topologies. However the common mode gain due to mismatching remains to be a dominant factor which limits the CMRR improvement.

REFERENCES

[1] J. H. Botma, R. F. Wassenaar, and R. J. Wiegerink, "A low-voltage CMOS op amp with a rail-to-rail constant-g_m input stage and a class AB rail-to-rail output stage," in *IEEE 1993 ISCAS*, Chicago, IL, pp. 1314–1317.

[2] R. Hogervorst, J. P. Tero, R. G. H. Eschauzier, and J. H. Huijsing, "A compact power-efficient 3 V CMOS rail-to-rail input/output operational amplifier for VLSI cell libraries," in *ISSCC*, Feb. 1994, pp. 244–245.

[3] W. S. Wu, W. J. Helms, J. A. Kuhn, and B. E. Byrkett, "Digital-compatible high-performance operational amplifier with rail-to-rail input and output ranges," *IEEE J. Solid-State Circuits*, vol. 29, pp. 63–66, Jan. 1994.

[4] J. F. Duque-Carrillo, R. Perez-Aloe, and J. M. Valverde, "Biasing circuit for high input swing operational amplifiers," *IEEE J. Solid-State Circuits*, vol. 30, pp. 156–159, Feb. 1995.

[5] J. F. Duque-Carrillo, J. M. Valverde, and R. Perez-Aloe, "Constant-g_m rail-to-rail common-mode range input stage with minimum CMRR degradation," *IEEE J. Solid-State Circuits*, vol. 28, pp. 661–666, June 1993.

[6] M. D. Pardoen and M. G. Degrauwe, "A rail-to-rail input/output CMOS power amplifier," *IEEE J. Solid-State Circuits*, vol. 25, pp. 501–504, Apr. 1990.

[7] R. Hogervorst *et al.*, "CMOS low-voltage operational amplifiers with constant-g_m rail-to-rail input stage," in *Proc. ISCAS*, 1992, pp. 2876–2879.

[8] M. Ismail and T. Fiez, *Analog VLSI Signal and Information Processing.* New York: McGraw-Hill, 1994.

[9] K. Laker and W. Sansen, *Design of Analog Integrated Circuits and Systems.* New York, 1994.

[10] B. Char *et al.*, *Maple V Library Reference Manual.* New York: Springer-Verlag, 1991.

[11] J. Franca and Y. Tsividis, *Design of Analog-Digital VLSI Circuits for Telecommunication and Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1994.