An 85-MHz Fourth-Order Programmable IIR Digital Filter Chip

Mehdi Hatamian, *Senior Member, IEEE,* and Keshab K. Parhi, *Senior Member, IEEE*

Abstract—**This paper describes the design and VLSI implementation of a single-chip 85-MHz fourth-order infinite impulse response (IIR) digital filter chip fabricated in 0.9-μm CMOS technology. The coefficient and input data word lengths of the filter are 10 b each; output data word length is 15 b. The coefficients are fully programmable. The chip can be programmed to implement any IIR filter from first to fourth order or an FIR filter up to 16th order at sample rates up to 85 MHz. A total of seventeen 10 × 10 multiply-add modules are used in this chip. The chip contains 80 000 devices in an active area of 14 mm^2. It dissipates 2.2 W at 85-MHz clock rate and performs over 1.5 × 10^9 multiply-add operations per second. The underlying filtering algorithm, chip architecture, circuit and layout design, speed issues, and test results are described. The results of an E-beam probing experiment on packaged chips at 100-MHz clock rates are also presented and discussed.**

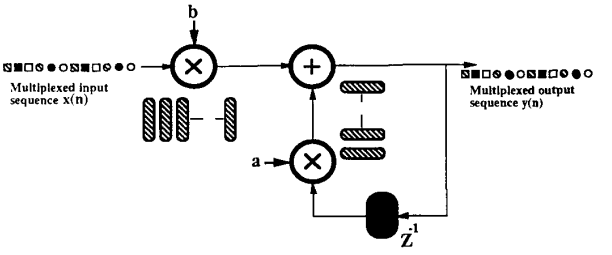


Fig. 1. A first-order IIR filter. A simple pipelining of the multiply and add operations (shown by cross-hatched registers) does not result in the same IIR filter with higher sample rate. It creates an architecture suitable for multiplexed multichannel filtering.

I. Introduction

INFINITE impulse response (IIR) and finite impulse response (FIR) digital filters are perhaps the two most fundamental tools in the world of digital signal processing. High-speed VLSI implementation of both of these filtering approaches is of interest to many in the signal processing community. While pipelining techniques can be rather easily used to implement high-sample-rate FIR filters, their use in IIR filters is not as straightforward due to the feedback loops involved. Fig. 1 shows a simple first-order IIR filter implementing the operation:

$$y(n) = ay(n - 1) + bx(n). \tag{1}$$

Simply pipelining the multiply and/or add operation in this structure results in a multichannel filter if the input samples are properly multiplexed as is graphically shown in Fig. 1. In other words, the same filtering operation will be performed on M different sequences (multiplexed at the input and fed to the filter at M times the sample rate of the original nonpipelined filter) where M is the degree of pipelining. Similarly, if a single input signal is fed at the original slow sample rate but the coefficients are changed at the fast clock rate, then it is possible to run M different filters (M sets of coefficients) on the same input signal. While these operations by themselves are useful in certain applications, the pipelining attempt does not improve the sample rate of the original IIR filtering operation. This is simply due to the fact that the output $y(n)$ at each sample point is a function of the sample point immediately preceding it and there are no inherent delay operators inside the feedback loop. How to insert such delay operators in the feedback loops of IIR filters without modifying the original filter transfer function is the topic of discussion in [1] and the basis of the algorithm used in the chip described in this paper. The IIR filtering algorithm is described in Section II. The chip's architecture, floor plan, and layout are discussed in Section III. Section IV presents a discussion of speed issues and performance evaluation. The chip has been fabricated in 0.9-μm CMOS technology. It has been tested at sample rates up to 85 MHz. The results of these tests and an E-beam probing experiment performed on packaged chips (at 100-MHz clock rate) are presented. To the best of our knowledge, the chip presented in this paper is the fastest and most complex single-chip IIR digital filter ever reported.

Manuscript received December 5, 1990; revised September 13, 1991.

M. Hatamian is with Silicon Design Experts, Inc., Lakewood, NJ 08701.
K. K. Parhi is with the Department of Electrical Engineering, University of Minnesota, Minneapolis, MN 55455.

IEEE Log Number 9104345.

II. The Algorithm

In this section we present details of the pipelining technique for feedback loops using a first-order recursive section as an example followed by an example of a second-order section [2]. The chip presented in this paper uses two cascaded second-order sections to implement a fourth-order IIR filter.

Consider the first-order section described by (1) having the z-domain transfer function

$$\frac{b}{1 - az^{-1}}. \tag{2}$$

As mentioned in Section I, in order to be able to apply fine-grain pipelining to the multiply–add operation without modifying the actual filtering operation itself, we need to come up with delay operators (latches) inside the feedback loop. The number of latches required in the loop would be equal to the desired degree of fine-grain pipelining. To accomplish this, we start by unfolding the loop in (1) as

$$\begin{aligned} y(n+1) &= ay(n) + bx(n+1) \\ &= a[ay(n-1) + bx(n)] + bx(n+1) \\ &= a^2y(n-1) + abx(n) + bx(n+1) \\ y(n+2) &= ay(n+1) + bx(n+2) \\ &= a^3y(n-1) + a^2bx(n) + abx(n+1) \\ &\quad + bx(n+2) \\ &\vdots \\ y(n+M-1) &= a^My(n-1) + a^{M-1}bx(n) \\ &\quad + a^{M-2}bx(n+1) + \cdots \\ &\quad abx(n+M-2) + bx(n+M-1) \end{aligned}$$

or equivalently

$$\begin{aligned} y(n) = a^My(n-M) + a^{M-1}bx(n-M+1) \\ + \cdots + abx(n-1) + bx(n). \end{aligned} \tag{3}$$

In this form, the output $y(n)$ is a function of the Mth previous output sample, meaning that there are M delay operators inside the feedback loop. Fig. 2 shows the block diagram of the unfolded filter for $M = 4$ implementing the transfer function:

$$H(z) = \frac{b[1 + az^{-1} + a^2z^{-2} + a^3z^{-3}]}{1 - a^Mz^{-M}}. \tag{4}$$

Now, the four delay operators in the feedback loop can be redistributed inside the multiply–add operations itself providing a factor of 4 increase in throughput. The price paid is the increase in complexity introduced by the overhead section in (3) (i.e., the FIR filter section in Fig. 2). Figs. 3 and 4 show the z-domain poles and zeros of the original transfer function (1) and the pipelined transfer function (4) (for $M = 4$), respectively. As can be seen, converting the original IIR filter to a pipelined version is equivalent to adding new poles and zeros (canceling zeros) to the original transfer function. The added zeros form the FIR section in Fig. 2 and are canceled by the poles obtained from the roots of $1 - a^4z^{-4}$ (the denominator of the pipelined transfer function), which came about due to repeated unfolding operation (which made the fine grain pipelining possible). The pole located at $z = a$ is the original pole and is not canceled by any zeros.

A simple, yet very important, observation helps in reducing the complexity of the feedforward section of the pipelined IIR filter which implements the canceling zeros.

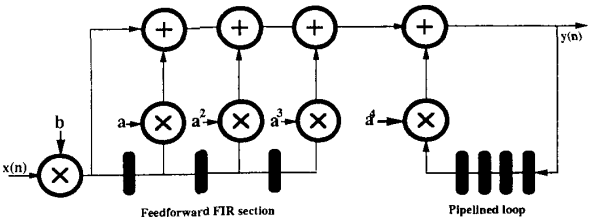


Fig. 2. Direct-form implementation of the pipelined transfer function of a first-order IIR section before decomposition.

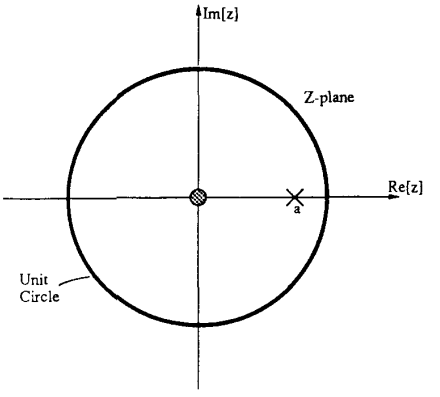


Fig. 3. Pole and zero location of a first-order IIR section.

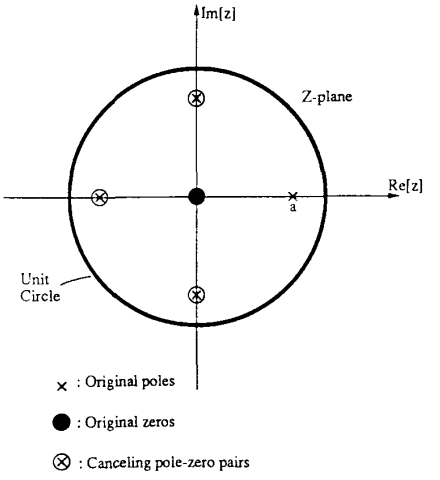


Fig. 4. Pole and zero locations of the transfer function described by (4), which is equivalent to the first-order IIR section but with four delay operators inside the feedback loop. The canceling pole–zero pairs provide the loop delay operators needed for fine-grain pipelining without modifying the original transfer characteristics of the filter.

The numerator of the transfer function in (4) can be decomposed as

$$1 + az^{-1} + a^2z^{-2} + a^3z^{-3} = (1 + az^{-1})(1 + a^2z^{-2}). \tag{5}$$

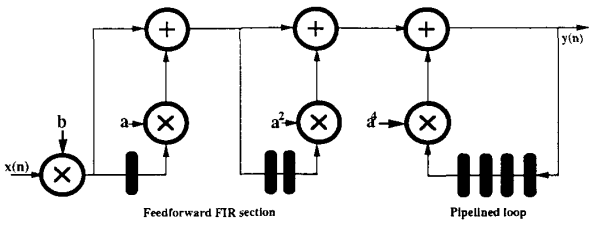


Fig. 5. Implementation of the pipelined first-order IIR section following the decomposition rule. The canceling zeros are implemented using two cascaded feedforward section having coefficients a and a^2.

A simple examination of the symmetric locations of the canceling zeros in Fig. 4 also reveals this fact. Fig. 5 shows the block diagram of the first-order pipelined IIR filter following this decomposition rule. The canceling zeros are implemented using a cascade of two feedforward sections having coefficients a and a^2.

In the remainder of this section we discuss the procedure for pipelining a second-order IIR section using the canceling pole–zero concept and the decomposition rule. We begin with the Z-domain transfer function of a complete second-order section described as

$$H(z) = \frac{a(1 + bz^{-1} + cz^{-2})}{(1 - 2r\cos\theta z^{-1} + r^2z^{-2})}. \tag{6}$$

The pole and zero locations of this transfer function are depicted in Fig. 6. In order to be able to pipeline this filter by a degree $M = 4$, we need four pairs of poles on a circle with radius r (which is the radius of the original poles) located at angles $\pm\theta$, $(\pi/2) \pm \theta$, $\pi \pm \theta$, and $(3\pi/2) \pm \theta$ [1]. The two poles at $\pm\theta$ are the original poles; the other six poles must be canceled by six canceling zeros in order to preserve the transfer characteristics of the original filter. The resulting pole and zero locations on the pipelined filter are shown in Fig. 7. It can be shown that the eight poles of the pipelined transfer function are the roots of $1 - 2r^4\cos 4\theta z^{-4} + r^8z^{-8}$. Out of the six canceling zeros, the four zeros located at angles $(\pi/2) \pm \theta$ and $(3\pi/2) \pm \theta$ are the roots of $1 + 2r^2\cos 2\theta z^{-2} + r^4z^{-4}$ and the remaining two zeros at angles $\pi \pm \theta$ are the roots of $1 + 2r\cos\theta z^{-1} + r^2z^{-2}$. As a result, the transfer function of the pipelined second-order IIR section has the form:

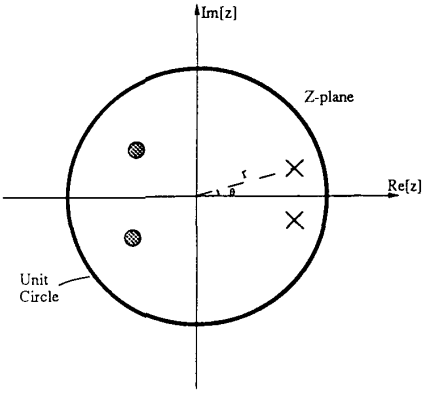


Fig. 6. Pole and zero locations of a second-order IIR section described by (6).

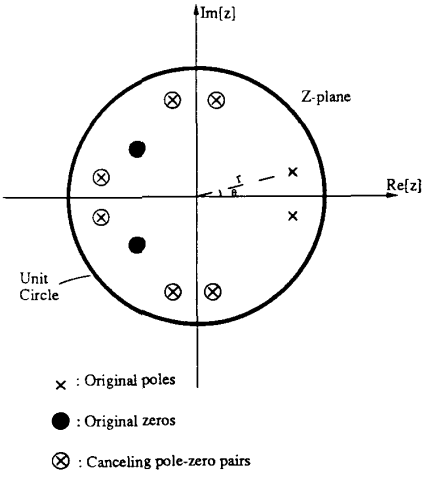


Fig. 7. Pole and zero locations of the pipelined transfer function described by (7), which is equivalent to a second-order IIR section but with delay operators inside the feedback loop which can be used for fine-grain pipelining.

$$H(z) = \frac{a(1 + bz^{-1} + cz^{-2})(1 + 2r\cos\theta z^{-1} + r^2z^{-2})(1 + 2r^2\cos 2\theta z^{-2} + r^4z^{-4})}{1 - 2r^4\cos 4\theta z^{-4} + r^8z^{-8}}. \tag{7}$$

Fig. 8 shows the block diagram implementing the above transfer function. The three feedforward stages implement the two original zeros and the six canceling zeros. The final feedback stage implements the eight poles of (7) and, as can be seen, has four delay operators inside the feedback loop which can be used for fine-grain pipelining by properly redistributing them inside the loop, as will be described in Section III. The filter in (7) is essentially an eighth-order filter that requires the output $y(n)$ to be computed as a function of $y(n - 4)$ and $y(n - 8)$. This process of computing $y(n)$ using two past samples that are scattered in time has been referred to as *scattered look-ahead*. The scattered look-ahead bypasses the dependency of $y(n)$ on its immediate past samples, resulting in more delay operators inside the feedback loop as was discussed above.

The details of the scattered look-ahead and the decomposition techniques for general recursive filters can be found in [3]. A discussion of the quantization and roundoff error issues in pipelined recursive filters can be found in [4]. In [4] it is shown that to maintain identical roundoff and quantization errors in the original and the scattered look-ahead IIR filter, the scattered look-ahead filter requires a 1- or 2-b increase in the word length (in the worst case with respect to all possible pole locations) for two to eight stages of pipelining.

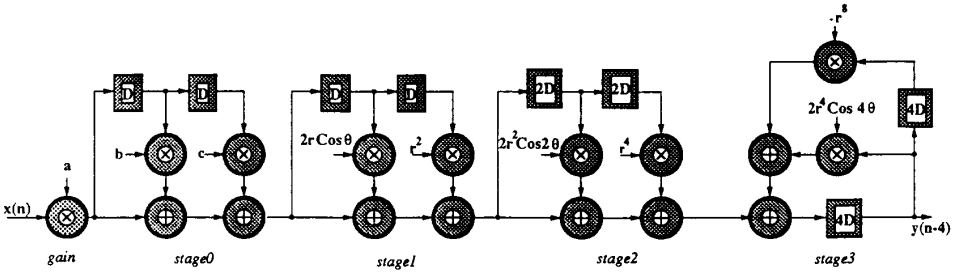


Fig. 8. Block diagram of the pipelined second-order IIR filter after adding the canceling pole-zero pairs and following the decomposition rules.

III. Architecture and Layout

The IIR chip described in this paper implements a cascade of two second-order modules similar to the block diagram of Fig. 8. The gain stages of the two modules (i.e., the first multiplier in Fig. 8) are combined into one gain block. As in numerous other signal-processing architectures, the core block of our chip is a multiply–add unit. Each multiply–add unit has four pipeline stages embedded in it. The FIR sections of the filter (stage0, stage1, and stage2 in Fig. 8) can directly use such a pipelined multiply–add unit. However, the feedback section, in the form shown in Fig. 8, is not directly suitable for fine-grain pipelining since all four delays in the loop are lumped together. To exploit pipelining, we need to redistribute the four delays to obtain four-stage pipelined implementation of the multiply–add circuitry. This process of redistributing the loop delays corresponds to retiming of the flow graph [5] (also often referred to as cutset transformation [6]).

Fig. 9 illustrates one possible redistribution of the delays in the pipelined filter. The m_1, m_2, and m_3 units correspond to the multiply operation with $2r^4 \cos 4\theta$, and the m_4, m_5, and m_6 units correspond to the multiply operation with respect to $-r^8$. The unit m_7 represents a three-input adder. It should be emphasized that this redistribution could be uneven. Our chip, however, uses an even redistribution. The outer loop in Fig. 9 has eight delays, four of which are redistributed for pipelining, and the remaining four lumped delays are algorithm delays (not used for pipelining).

The floor plan of a complete second order pipelined retimed filter is shown in Fig. 10. The gain stage represents the multiplication by the gain coefficient. The numerator of the original second-order IIR section (equation (6)) is represented by stage0. Modules stage1 and stage2 implement two and four canceling zeros, respectively, and stage3 implements the denominator of the pipelined second-order filter (which includes two original poles and six redundant poles). The chip contains four additional stages stage5 through stage8 (exactly identical to stage0 through stage3) following stage3, corresponding to another second-order section. These stages have been omitted from the figure for clarity. Layouts of the multiply–add modules for each stage are interleaved as will be described shortly.

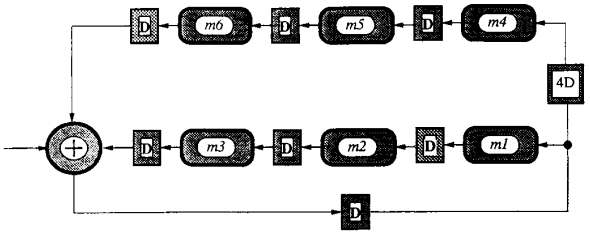


Fig. 9. A redistribution of the delays in the feedback loop (stage3) of the pipelined filter in Fig. 8 to obtain a four-stage pipelined implementation of the multiply–add circuits.

The entire fourth-order pipelined filter has 17 multiply–add units, and requires 17 10-b coefficients. These coefficients can be programmed serially in ten clock cycles (one pin per coefficient), and are held in static registers thereafter.

The multiply adders are realized using a combination of carry-save and carry-ripple approaches. The gain stage in Fig. 10 represents the multiplication by gain a. The total computational delay through the multiply unit equals the delay through 20 one-bit full adders. Since the multiply/add operations are pipelined by four levels, the computational delay in each pipeline stage is that of five full adders. The first two pipeline stages accommodate five rows of carry-save adders. The last two pipeline stages are based on ripple carry of five binary adders. Our implementation is multibit level pipelined, and the combination of carry save and carry ripple reduces the number of adders and latches. An example of a carry-save- and carry-ripple-based 8 × 8-b multiplier using four pipeline stages is illustrated in Fig. 11. In this example, the delay in each pipeline stage is that of four full adders. The chip uses a 10 × 10 version of this architecture. The static registers for holding the coefficients and the circuits to serially load the coefficients have been omitted from Fig. 11 for clarity.

Modules stage0 through stage3 implement operations of the form $ax + by + z$ as in the block diagram in Fig. 8. For example, stage1 is of the form $y(n) = x(n) + fx(n - 1) + gx(n - 2)$, where $f = 2r \cos \theta$ and $g = r^2$. One way to implement this operation is to first do the multiply operation with respect to f, and then with respect to g. The drawback of such an implementation is that it requires additional routing space to carry through the signal

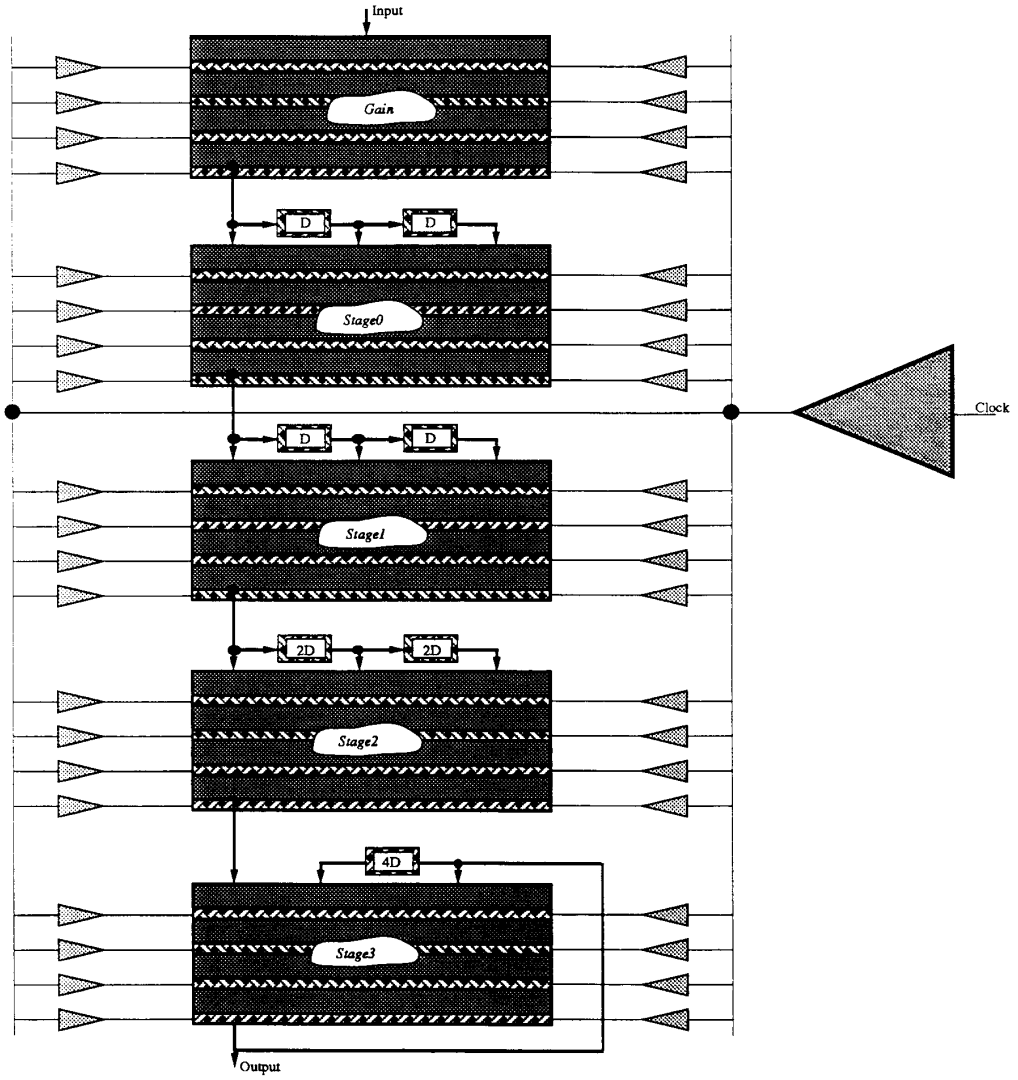


Fig. 10. Floor plan of the pipelined second-order filter. Clock signals running at the two sides of the chip feed the pipeline latches. Each computation stage has a delay of five 1-b full adders. Outputs of the recursive portion are fed back to the input of stage3 without being delayed.

$x(n)$ and the intermediate results (15-b buses). Alternatively, both multiplications can be performed simultaneously in parallel by interleaving the two multiplier arrays as shown in Fig. 12 (for a 4 × 4-b multiplier, 6-b input and output signals, and for coefficient word length of 4). This floor-plan interleaving of the multipliers reduces the amount of routing space, and results in a dense layout.

Layout designs of the main modules of the IIR chip (gain, stage0, and stage3) were carried out in symbolic domain [7], and subsequently converted to mask domain using a graph-based layout compactor [8]. The basic low-level building blocks of the entire layout are 1-b full adders, 1-b half adders, registers, and clock buffer cells. The interconnections between the main modules, as well as I/O frame routing, were carried out in the mask domain using a routing program. Fig. 13 shows the final layout of the fourth-order IIR chip in a microphotograph of the fabricated silicon. Various modules are identified in this figure. In the following section, we discuss high-speed clocking, operation speed, and simulation issues.

IV. Performance and Test Results

The chip described in this paper was fabricated in AT&T's 0.9-μm two-level-metal CMOS process. It has been packaged in a 101-pin ceramic PGA package. High-speed tests were performed using a pattern generator to deliver test vectors at speed and verify the filter output. A printed circuit board was designed for this purpose. The packaged chip was placed in a PGA socket on the printed circuit board. Clock lines on this board were 50-Ω mi-

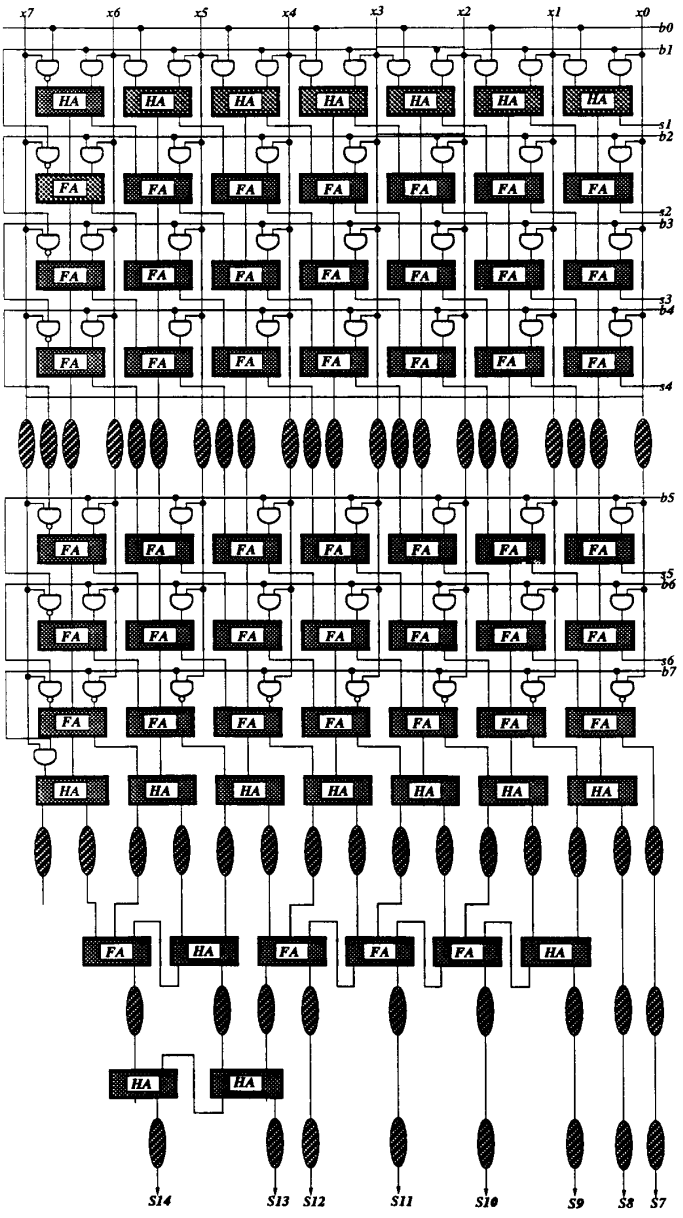


Fig. 11. Example of a carry-save- and carry-ripple-based 8 × 8 multiplier using four pipeline stages. Skewing registers for I/O synchronization are not shown.

crostrips terminated at the chip. The chip's output data lines were source terminated using a 39-Ω series resistor. Under these conditions, all the packaged chips tested were functional up to a 85-MHz sample rate. A further clean-up of the system-level signal environment resulted in achieving a 92-MHz sample rate for over 50% of the packaged chips tested. Table I summarizes the chip characteristics.

The main design objective here has been that of pipelining an IIR filtering algorithm to achieve higher sample rates. Due to the high-sample-rate requirement and a large number of pipeline registers to be synchronously clocked, the clock distribution network and the clock drivers are important design issues. The clock network in this chip uses two levels of clock distribution similar to those used in [9]. Each pipeline stage (forming a horizontal row in the chip) is driven by two clock buffers at the two ends. These stage drivers are in turn driven by a master clock buffer at the input clock pad. The microphotograph of Fig. 13 shows the placement of these drivers. The master driver is a two-stage cascaded CMOS buffer. The capacitive load on the output of this buffer is approximately 52 pF. This

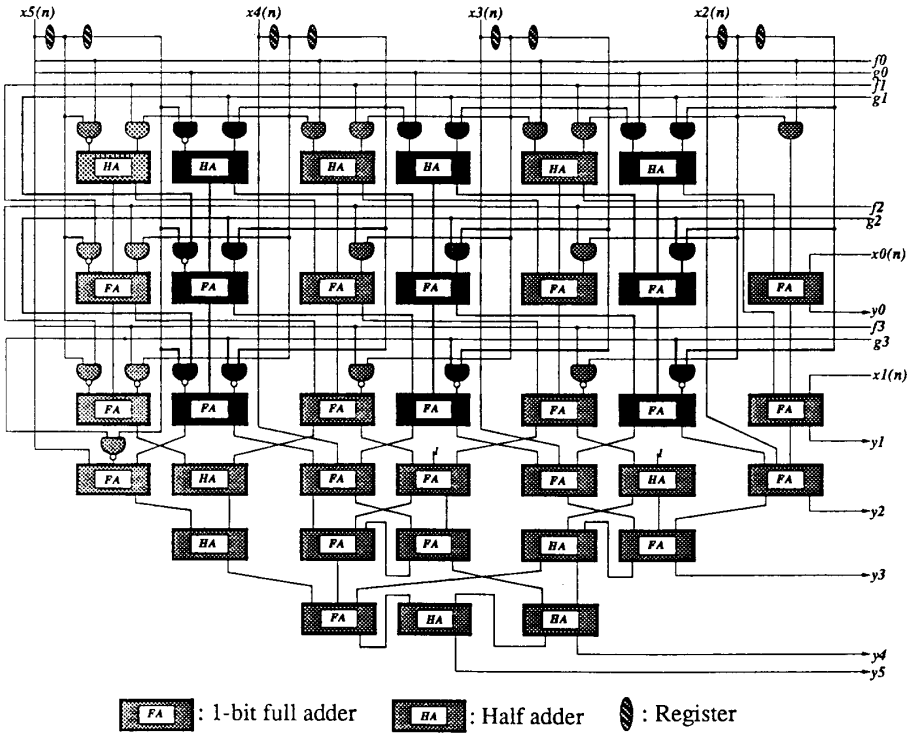


Fig. 12. Example of the interleaved architecture for performing the operation $y(n) = x(n) + f \cdot x(n-1) + g \cdot x(n-2)$. In this example, coefficients f and g are 4 b each; x and y are 6 b each. The actual implementation of the IIR chip uses one 10-b version of this architecture. The pipeline registers are not shown in this figure for clarity in representing the interleaved approach. $xm(n)$ represents bit m of x at sample point n.

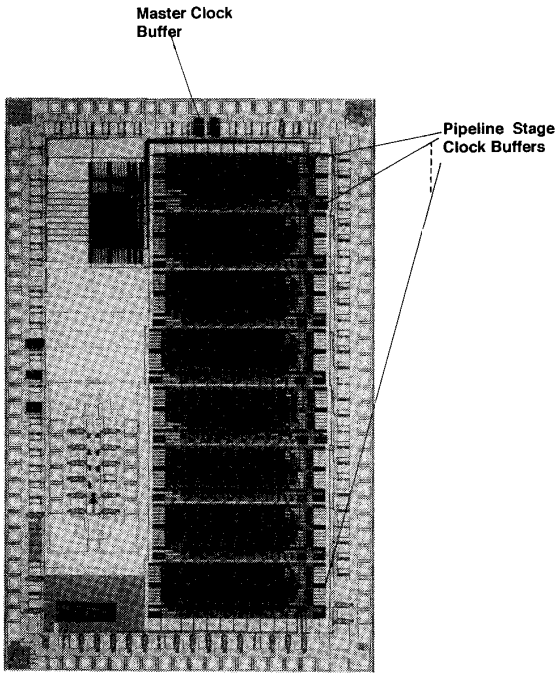


Fig. 13. Microphotograph of the IIR filter chip. Die size is 4 mm × 7 mm. The two isolated regions on the lower left-hand corner of the die are experimental circuits not related to the IIR layout.

TABLE I
CHARACTERISTICS OF THE FOURTH-ORDER IIR FILTER CHIP

Technology	0.9-μm CMOS (Two-Layer Metal)
No. of multiply-add units	17
Level of pipelining in a multiply/add	4
Sample rate (measured on packaged chips)	85 MHz
Power supply voltage	5 V
Power consumption	2.2 W at 85 MHz
No. of transistors	80 000
Active area	13.8 mm^2
Multiplication precision	10 × 10 b
Accumulation precision	15 b

buffer was designed using a buffer design program which takes into account parasitic capacitances that become significant when driving large loads with fast rise and fall times [9].

Several E-beam probing experiments were carried out to study the performance of the clock network. The experiments were performed on the packaged chips (101-pin ceramic PGA package) placed in a zero-insertion-force socket attached to the top plate of the E-beam probing machine. This presents a realistic worst-case situation (such high-speed chips should normally be used in a printed circuit board environment without sockets). Fig. 14 shows the result of one of the E-beam probing experiments at 100-MHz clock rate. The waveforms recorded

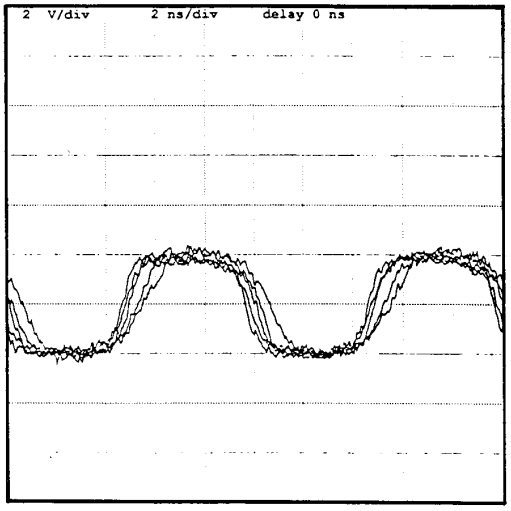


Fig. 14. Results of E-beam probing performed on the packaged chips at 100-MHz clock rate. Waveforms recorded from a number of critical points throughout the clock distribution network are superimposed on each other to give a picture of the clock skew inside the chip. A maximum skew of about 1 ns is observed across the network.

at a number of critical points throughout the clock distribution network are superimposed on each other to give a picture of clock skew across the chip. A maximum skew of about 1 ns is measured for the entire clock network. Notice that the important skews in a synchronous system are those that exist between modules directly communicating with each other. For the chip desired here, the maximum of such skew is about 250 ps. This skew in no way hampers the high-speed operation of the chip. This is true because the skew is always less than the propagation delay of the computation modules (see [9] and [10] for a detailed discussion of clock skew in high-speed chips). As mentioned above, the chip has been tested at sample rates up to 85 MHz. At 85 MHz and below, the clock-skew picture is the same as what is observed at 100 MHz, indicating that the clock skew and the clock distribution network are not the speed-limiting factor in this chip. The major speed limitation factor is the rather high power consumption of the chip. At 5-V supply voltage and 85-MHz clock rate, the power consumption is about 2.2 W, which causes a high junction temperature and a slowdown of the computational modules. The operation speed of the chip can be momentarily pushed to 100 MHz (limit of the test setup) by cooling the package with cooling spray. Timing simulations of the entire chip (including the I/O pads), using nominal process parameters, indicated an operation speed of 80 MHz at 125°C junction temperature (in close agreement with the measured results) and 126 MHz at 25°C.

We should also point out that the chip is capable of implementing 85-MHz FIR filtering operations up to eighth order with arbitrary coefficients or up to 16th order with constrained coefficients. As can be seen in the block diagram of Fig. 8, other than the gain coefficient, each second-order section has eight coefficients. Even though these coefficients form a dependent set and can be determined using only four parameters (c, b, r, and θ) in IIR mode, they can be independently loaded into the chip as separate coefficients. This feature allows the feedback section (stage3 in Fig. 8) to be disabled by setting its coefficients to zero. Given this mode of operation and the fact that the chip contains two cascaded second-order sections, an eighth-order FIR with arbitrary coefficients (when both stage2 and stage3 are disabled) or a 16th-order FIR with constrained coefficients (when only stage3 is disabled) can be implemented.

V. Conclusion

The chip presented in this paper performs over 1.5×10^9 multiply–add computations per second in less than 14 mm^2 of silicon area to implement a fourth-order IIR filtering operation at 85-MHz sample rate. This is a significant improvement over any existing IIR filtering hardware, and a demonstration of the practical feasibility of the underlying algorithm as well as the potential of the existing mature CMOS technologies in implementing very high-performance VLSI circuits for signal-processing applications. While the increase in the hardware complexity of the high-sample-rate pipelined IIR implementation (compared to a nonpipelined slow-rate IIR) may seem rather large at the first glance, the final single-chip design is extremely compact and very regular in stucture. In the world of practical VLSI designs, what counts at the end is the area–speed–power performance of a manufacturable device.

Other promising approaches to high-speed pipelined IIR filtering based on most-significant-bit-first arithmetic are described in [11] and [12]. Reference [12] also proposes a combination of MSB-first and scattered carry-look-ahead approaches. While these methods are very promising in achieving efficient implementations, a comparison with what has been presented in this paper would prove useful when a chip with a comparable complexity is designed and fabricated in a comparable CMOS technology. It is also our experience that the design environment and the available CAD tools make an appreciable difference in the final implementation efficiency of a given VLSI solution for a signal processing problem. It is important to point out that the chip described in this paper was fabricated and tested in early 1989. Since then, a number of improvements in our design environment and circuit and layout techniques have made it possible to achieve even higher performance levels with the same 0.9-μm CMOS technology as described in [13] and [14]. Based on the existing actual circuits, leaf cell layouts, and methods and tools presented in [13] and [14], simulations and floorplan studies show that it is now possible to implement the fourth-order IIR filter chip described here in less than 6.5 mm^2 of silicon area using 16-b coefficients (in the same

CMOS technology) running at over 90-MHz sample rate at 125°C junction temperature.

REFERENCES

[1] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters—Part I: Pipelining using scattered look-ahead and decomposition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 7, pp. 1099–1117, July 1989.

[2] K. K. Parhi and M. Hatamian, "A high sample rate digital filter chip," *VLSI Signal Processing II*. New York: IEEE Press, 1988, pp. 3–14.

[3] K. K. Parhi and D. G. Messerschmitt, "Pipelined VLSI recursive filter architectures using scattered look-ahead and decomposition," in *Proc. 1988 IEEE Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1988, pp. 2120–2123.

[4] K. K. Parhi, "Finite word effects in pipelined recursive filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 39, no. 6, pp. 1451–1454, June 1991.

[5] C. E. Leiserson, F. Rose, and J. Saxe, "Optimizing synchronous circuitry by retiming," presented at the 3rd Caltech Conf. VLSI, Pasadena, CA, Mar. 1983.

[6] S. Y. Kung, "On supercomputing with systolic/wavefront array processors," *Proc. IEEE*, vol. 72, no. 7, July 1984.

[7] N. Weste, "Virtual grid symbolic layout," in *Proc. 18th Design Automation Conf.* (Nashville, TN), June 1981, pp. 225–233.

[8] C.-Y. Lo, R. Varadarajan, and W. H. Crocker, "Compaction with performance optimization," in *Proc. IEEE Int. Symp. Circuits Syst.* (Philadelphia, PA), May 1987, pp. 514–517.

[9] M. Hatamian and G. L. Cash, "Parallel bit-level pipelined VLSI designs for high speed signal processing," *Proc. IEEE*, vol. 75, no. 9, pp. 1192–1202, Sept. 1987.

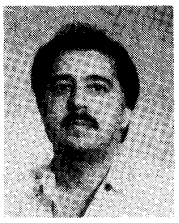
[10] M. Hatamian, "Understanding clock skew in synchronous systems," in *Concurrent Computations: Algorithms, Architectures, and Technology*, S. K. Tewksbury, Ed. New York: Plenum, 1988, pp. 87–96.

[11] S. C. Knowles, J. G. McWhirter, R. F. Woods, and J. V. McCanny, "Bit-level systolic architectures for high performance IIR filtering," *J. VLSI Signal Processing*, pp. 9–24, Feb. 1989.

[12] O. C. McNally, J. V. McCanny, and R. F. Woods, "Optimized bit-level architectures for IIR filtering," presented at the IEEE Int. Conf. Computer Design, Boston, MA, Sept. 1990.

[13] M. Hatamian and S. K. Rao, "A 100 MHz 40-tap programmable FIR filter chip," in *Proc. IEEE Int. Symp. Circuits Syst.* (New Orleans, LA), May 1990, pp. 3053–3056.

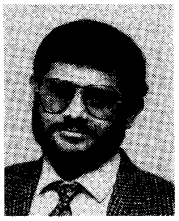
[14] S. K. Rao and M. Hatamian, "A 65 MHz 16-tap FIR filter chip with on-chip video delay lines," in *Proc. IEEE Int. Symp. Circuits Syst.* (New Orleans, LA), May 1990, pp. 3050–3052.



Mehdi Hatamian (S'75–M'82–SM'88) was born in Tehran, Iran, in 1954. He received the B.S. degree from the University of Technology, Tehran, in 1977, and the M.S. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1978 and 1982, respectively, all in electrical engineering.

From 1979 to 1982 he was a Research Assistant on a NASA project working on the design and development of real-time hardware and control software for a Space Shuttle experiment. He joined AT&T Bell Laboratories in 1982 where he became a Distinguished Member of the Technical Staff in the VLSI Systems Research Department in 1988. He left AT&T Bell Laboratories in 1991 and became a cofounder of Silicon Design Experts, Inc., Lakewood, NJ. His research interests are in high-speed and real-time signal processing, algorithms and VLSI architectures for signal processing, high-speed circuit design and simulation, CAD tools, and superconducting interconnects. He holds three patents, and has published over 34 papers.

Dr. Hatamian is a member of Sigma X. He has served as the organizer, session chair, and panelist in a number of national and international conferences and workshops in his active research areas.



Keshab K. Parhi (S'85–M'88–SM'91) received the B.Tech. (Honors) degree from the Indian Institute of Technology, Kharagpur, in 1982, the M.S.E.E. degree from the University of Pennsylvania, Philadelphia, in 1984, and the Ph.D. degree from the University of California, Berkeley, in 1988.

He is currently an Assistant Professor of Electrical Engineering at the University of Minnesota, Minneapolis. He held short-term positions at AT&T Bell Laboratories, Holmdel, NJ, IBM T. J. Watson Research Center, Yorktown Heights, NY, and the Tata Engineering and Locomotive Company, Jamshedpur, India. In 1989, he was a consultant to the U.S. West Advanced Technologies in the area of neural network signal processing. His research interests include concurrent algorithm and architecture designs for communications, signal and image processing systems, digital integrated circuits, VLSI digital filters, computer arithmetic, high-level DSP synthesis, and multiprocessor prototyping and task scheduling for programmable software systems. He has published over 60 papers in these areas.

Dr. Parhi received the 1991 Browder Thompson prize of the IEEE, the 1989 research initiation award of the National Science Foundation, and the 1987 Eliahu Jury award and the 1987 Demetri Angelakos award of the University of California (U.C.), Berkeley. At U.C. Berkeley, he was a U.S. Regents Fellow and an IBM Graduate Fellow. He is a past Associate Editor for Image Processing and VLSI Applications of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, and is a member of Eta Kappa Nu and the Association for Computing Machinery.