INFORMATICA, 1996, Vol. 7, No. 1, 15–26

PIPELINING OF IIR DIGITAL FILTERS

Kazys KAZLAUSKAS

Institute of Mathematics and Informatics
2600 Vilnius, Akademijos St. 4, Lithuania

Abstract. In this paper, we propose to present the direct form recursive digital filter as a state space filter. Then, we apply a look-ahead technique and derive a pipelined equation for block output computation. In addition, we study the stability and multiplication complexity of the proposed pipelined-block implementation and compare with complexities of other methods. An algorithm is derived for the iterative computation of pipelined-block matrices.

Key words: digital filters, recursive, state space, pipelining, stability, complexity.

1. Introduction. Digital filter transfer function theoretically can be realized in an infinite number of ways. From the standpoint of implementation, some structures may be of lower complexity, while others may be pipelinable, and yet some others may consist of regular modules that can save design time. Much research has been carried out in search of different realization structures with various desirable properties and enhanced performance. Achieving high speed in recursive direct-form filters is difficult because of the feedback loop (Chung and Parhi, 1994).

High performance in very large scale integration (VLSI) circuits can be achieved by using high speed technologies without modifying the algorithm. On the other hand, we can use a low cost technology and gain an impressive performance by exploiting concurrency. Concurrent structures can be derived by implementing the existing algorithms in new ways. We do not change the transfer function of the filter, but we do change the internal structure of the filter (Parhi and Messerschmitt, 1989). Pipelining and block processing are two of several algorithmic transformation techniques that can be used to exploit the concurrency within a digital signal processing algorithm to improve its operating speed or reduce the number of resources required in a parallel processing environment (Lucke and Parhi, 1994). Pipelining (Chung and Parhi, 1994; Parhi and Messerschmitt, 1989b; Lucke and Parhi, 1994; Parhi and Messerschmitt, 1989c; Jump and Ahuja, 1978; Cappello and Steiglitz, 1983; Lim and Bede Liu, 1992) increases the speed of a filter at the expense of latency. Block processing (Parhi and Messerschmitt, 1989a; Burrus, 1971; Barnes and Shinnaka, 1980; Azimi-Sadjadi and King, 1986; Azimi-Sadjadi and Rostampour, 1989; Nikias, 1984) is a form of parallel processing which transforms a single-input single-output filter into a multiple-input multiple-output filter.

Rapid advances in VLSI technology have made a great impact on modern signal processing. One of the desirable properties for VLSI realization is pipelinability. In (Parhi and Messerschmitt, 1989b) pipelining in direct form recursive digital filters with constant coefficients have been considered using the clustered look-ahead technique and scattered look-ahead pipelining without a decomposition and with a decomposition. The clustered look-ahead pipelined realizations require a linear complexity in the number of loop pipeline stages, and are not guaranteed to be stable. The scattered look-ahead realizations are stable and require a linear complexity with respect to the number of loop pipeline stages. The decomposition technique enables us to obtain an implementation with a logarithmic increase in hardware with respect to the number of loop pipeline stages.

In this paper (Section 2), for pipelining in direct form recursive digital filters with constant coefficients, we propose to present the direct form recursive digital filter as a state space filter. In Section 3 we apply the clustered look-ahead technique to derive a pipelined equation for block output computation. The stability and multiplication complexity are analyzed. In Section 4 the algorithm for computing pipelined-block matrices is given. Finally, in Section 5 conclusions are given.

2. Preliminaries: representation of direct form recursive filters in a state space domain. The transfer function of a direct form recursive digital filter is described by

$$H(z) = \frac{\sum_{i=0}^{M} b_i z^{-i}}{1 - \sum_{i=1}^{N} a_i z^{-i}}, \quad M \leqslant N. \tag{1}$$

Equivalently, the output sample $y(k)$ can be described in terms of the input sample $v(k)$, and the past input and output samples, and is given by

$$y(k) = \sum_{i=1}^{N} a_i y(k-i) + u(k), \quad k = 0, 1, 2, \ldots, \tag{2}$$

where

$$u(k) = \sum_{i=0}^{M} b_i v(k-i). \tag{3}$$

Then direct form recursive digital filter (1) can be represented as a sequential connection of two filters: FIR filter, described by Eq. 3, and all-pole filter, described by Eq. 2. Block implementation of the FIR filter is simple (Lucke and Parhi, 1994), so we consider only pipelined-block implementation of the all-pole filter (2).

Following Isermann (1981), Eq. 2 can be written in state space recursive form:

$$x(k+1) = Ax(k) + bu(k), \tag{4}$$

where b and $x(k)$ are $N \times 1$ vectors defined by

$$b = [0, \ldots, 0, 1]^T, \quad x(k) = [x_1(k), x_2(k), \ldots, x_N(k)]^T,$$

in which

$$\begin{aligned} x_1(k) &= y(k-N), \\ x_2(k) &= y(k-N+1) = x_1(k+1), \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_N(k) &= y(k-1) = x_{N-1}(k+1), \\ y(k) &= x_N(k+1). \end{aligned} \tag{5}$$

The $N \times N$ state update matrix A is defined by

$$A = \begin{bmatrix} 0 & 1 & 0 & . & . & . & 0 \\ & & & & & \mathbf{0} & \\ & & & & . & & \\ & & & & & . & \\ & & & & & & . \\ & \mathbf{0} & & & & & 1 \\ a_N & & & . & . & . & a_1 \end{bmatrix}. \tag{6}$$

3. Pipelined-block implementation of IIR digital filter. Define $k = mL + n$, where L is the block size, $m = 0, 1, 2, \dots$ is a block variable, and $n = 0, 1, \dots, L-1$ is a variable inside the block. Then, varying n from 0 to $L-1$, and substituting $x(mL+1)$ into $x(mL+2)$, $x(mL+2)$ into $x(mL+3)$, and so on, we have from Eq. 4

$$x(mL+L) = \overline{A}x(mL) + \overline{B}U(mL), \quad m = 0, 1, 2, \dots \qquad (7)$$

where the state update matrix $\overline{A} = A^L$,

$$\begin{aligned}
&\overline{B} = \left[A^{L-1}b, \dots, Ab, b\right], \\
&x(mL) = \left[y(mL-N), y(mL-N+1), \dots, y(mL-1)\right]^T, \\
&U(mL) = \left[u(mL), u(mL+1), \dots, u(mL+L-1)\right]^T.
\end{aligned}$$

In the case $L \leqslant N$, using only L last values of $x(mL+L)$ we find from Eqs. 5 and 7

$$y_L(mL) = \overline{A}_L y_N(mL-N) + \overline{B}_L U(mL), \quad m = 0, 1, \dots, \qquad (8)$$

where $\overline{A}_L$ is an $L \times N$ matrix, whose L rows are equal to L last rows of the matrix A^L; $\overline{B}_L$ is an $L \times L$ matrix; $y_L(mL)$ is an $L \times 1$ column vector $y_L(mL) = [y(mL), \dots, y(mL+L-1)]^T$, and $y_N(mL-N)$ is an $N \times 1$ column vector $y_N(mL-N) = [y(mL-N), \dots, y(mL-1)]^T$.

It follows from Eq. 8 that the block of output values $y_L(mL)$, $m = 0, 1, \dots$ is computed in a parallel manner. Note that the output value $y(mL+1)$ is 2-stage pipelined, $y(mL+2)$ is 3-stage pipelined, and so on, finally, $y(mL+L-1)$ is L-stage pipelined. If $L = 1$, then Eq. 8 is equal to Eq. 4, and $\overline{A}_L = A$, $\overline{B}_L = b$.

For the case when the block size L is greater than the filter order N, from Eq. 5 and 7, we get

$$y_N(mL+L-N) = \overline{A}_N y_N(mL-N) + \overline{B}_N U(mL), \quad m = 0, 1, \dots, \qquad (9)$$

where an $N \times N$ matrix $\overline{A}_N = A^L$; the matrix $\overline{B}_N$ is of dimension $N \times L$; $y_N(mL-N)$ is an $N \times 1$ column vector $y_N(mL-N) = [y(mL-N), \dots, y(mL-1)]^T$. The block of the output values $y_N(mL+L-N)$ is computed in a parallel manner. The output value $y(mL+L-N)$ is $(L-N+1)$-stage pipelined, $y(mL+L-N+1)$ is $(L-N+2)$-stage pipelined, and, finally, $y(mL+L-1)$ is L-stage pipelined.

If $L/N = M_1$ =const, then from Eq. 9, we have

$$\begin{aligned} &y_N(mL+L-N) = \overline{A}_N y_N(mL-N) + \overline{B}_N U(mL),\\ &y_N(mL+L) = \overline{A}_N y_N(mL) + \overline{B}_N U(mL+N),\\ &\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\ &y_N\left[mL+L+(M_1-2)N)\right] = \overline{A}_N y_N\left[mL+(M_1-2)N)\right]\\ &\qquad\qquad + \overline{B}_N U\left[mL+(M_1-1)N\right],\\ &\qquad\qquad m = 0,1,2,\dots, \end{aligned} \tag{10}$$

where $y_N(i)$ is an $N \times 1$ column vector, $\overline{A}_N$ is an $N \times N$ matrix, and $\overline{B}_N$ is an $N \times L$ matrix.

We can use Eqs. 10 to compute L output values in a parallel manner.

S t a b i l i t y. It is clear that since $\overline{A} = A^L$ the eigenvalues of $\overline{A}$ are the L-th power of the eigenvalues of A, i.e., $Ae_i = \lambda_i e_i$, and $\overline{A}e_i = A^L e_i = \lambda_i^L e_i$, where λ_i is an eigenvalue and e_i is an associated eigenvector of the matrix A. Since every pole is an eigenvalue, this implies that the poles of filter (7) are the L-th power of the poles of filter (4), i.e., $|\lambda_i^L| < |\lambda_i|$ if $|\lambda_i| < 1$. Thus, filter (7) is stable if filter (4) or filter (2) are stable.

M u l t i p l i c a t i o n   c o m p l e x i t y. For the case $L \leqslant N$, the number of multiplications required for computing one output value due to $\overline{A}_L$ is N and due to $\overline{B}_L$ is $(L-1)/2$. Thus, the average multiplication complexity required for computing one output value of filter (8) is $(L+2N-1)/2$. The computation of one output value of filter (3) requires $M+1$ multiplications. Then the average total multiplication complexity required for computing one output value of pipelined-block filter (1) is $(L+2N-1)/2+M+1$.

For the case $L > N$, the number of multiplications required for computing one output value due to $\overline{A}_N$ is N and due to $\overline{B}_N$ is $L-(N+1)/2$. Thus, the average multiplication complexity required for computing one output value of filter (9) is $(N+2L-1)/2$. The average total multiplication complexity required for computing one output value of pipelined-block filter (1) is $(N+2L-1)/2+M+1$.

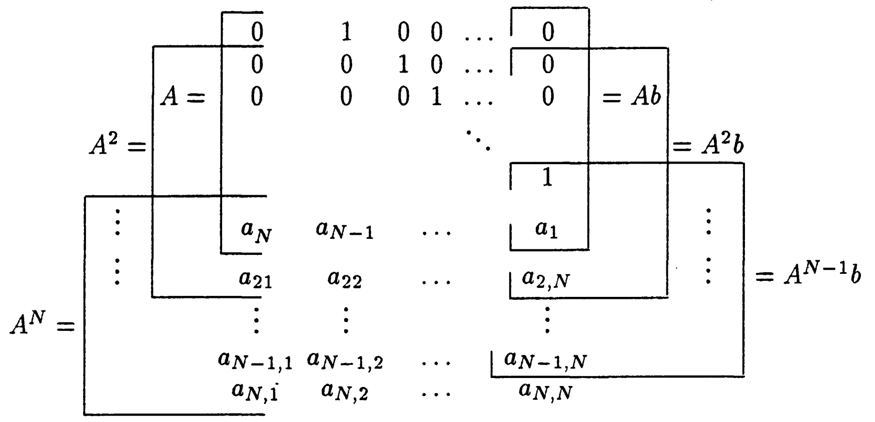
The computation complexity of one output value of direct form N-th order recursive digital filter (1) is $2N+L$ multiplications using scattered look-ahead pipelining, $NL + N + 1$ multiplications using clustered look-ahead pipelining, and $2N + N\log_2 L + 1$ multiplications using clustered look-ahead pipelining with decomposition (Parhi and Messerschmitt, 1989b).

Table 1. Comparison of multiplication complexity of direct form recursive filters

Speedup L	Clustered look-ahead pipelining	Scattered look-ahead pipelining	Scattered look-ahead pipelining with decomposition	Pipelined-block implementation
$M = N = 2$				
1	5	5	5	5.0
2	6	7	7	5.5
4	8	11	9	8.5
8	12	19	11	12.5
16	20	35	13	20.5
32	36	67	15	36.5
64	68	131	17	68.5
$M = N = 10$				
2	22	31	31	21.5
4	24	51	41	22.5
8	28	91	51	24.5
16	36	171	61	32.5
32	52	331	71	48.5
64	84	651	81	80.5
$M = N = L$				
2	6	7	7	5.5
4	12	21	17	10.5
8	24	73	41	20.5
16	48	273	97	40.5
32	96	1057	225	80.5
64	192	4061	513	160.5

Table 1 compares the number of multiplication operations for direct form recursive filters, for clustered look-ahead pipelining, scattered look-ahead pipelining with and without decomposition, and pipelined-block implementation, for typical factors of speedup L.

4. Computation of matrices $\overline{A}$ and $\overline{B}$. The structures of matrices A, $A^2, \ldots, A^N$ and vectors $Ab, A^2b, \ldots, A^{N-1}b$ are of the form:



Since $b = [0, \ldots, 0, 1]^T$, vectors $A^{L-1}b, \ldots, Ab$ are the last column vectors of the matrices $A^{L-1}, \ldots, A$, respectively.

C o m p u t a t i o n o f m a t r i x $\overline{A}$. For the case $L \leqslant N$ the elements of an $L \times N$ matrix $A^L = \{a_{ij}\}$ are computed iteratively using (12)

$$a_{ij} = a_{i-1,j-1} + a_{N+1-j} \cdot a_{i-1,N}, \quad i = \overline{2, L}, \ j = \overline{1, N}, \tag{12}$$

where $a_{11} = a_N, \ldots, a_{1,N} = a_1$, and $a_{i-1,0} = 0$, for all i.

For the case $L > N$, the elements of an $N \times N$ matrix $A^L = \{a_{ij}\}$ are computed iteratively using (13)

$$a_{ij}^{(L)} = a_{i,j-1}^{(L-1)} + a_{N+1-j} \cdot a_{i,N}^{(L-1)}, \quad i, j = \overline{1, N}, \tag{13}$$

where $a_{i,0} = 0$ for all i.

The impulse response in a state space is computed using Eq. 14

$$h(k) = c^T A^{k-1} b, \quad k > 0, \ h(0) = 1. \tag{14}$$

In our case, the $1 \times N$ vector $c^T = [0, \ldots, 0, 1]$, and $N \times 1$ vector $b = [0, \ldots, 0, 1]^T$. Then it follows from Eq. 14 that $h(1) = a_{1,N} = a_1$, $h(2) = a_{2,N}$, $h(3) = a_{3,N}, \ldots$. On the other hand, the impulse response of filter (2) can be computed from Eq. 15

$$h(k) = \sum_{j=1}^{N} a_j h(k-j), \quad k > 0; \quad h(0) = 1. \tag{15}$$

Thus, (12) can be rewritten in the form

$$a_{ij} = \sum_{l=1}^{j} a_{N-l+1} h(i-j+l-1), \quad i = \overline{1,L}, \; j = \overline{1,N}. \tag{16}$$

And (13) can be rewritten as

$$a_{ij} = \sum_{l=1}^{j} a_{N-l+1} h(i-j+l-1+L-N), \quad i = \overline{1,N}, \; j = \overline{1,N}. \tag{17}$$

C o m p u t a t i o n o f m a t r i x $\overline{B}$. $\overline{B} = [A^{L-1}b, \ldots, Ab, b]$. Since $b = [0, \ldots, 0, 1]^T$, vectors $A^{L-1}b, \ldots, Ab$ are the last column vectors of the matrices $A^{L-1}, \ldots, A$, respectively. As $h(k) = a_{k,N}$, it follows from (11) that

$$\begin{aligned} \left[\overline{B}\right]_{ij} &= h(i-j), \quad i, j = \overline{1,L}, \text{ if } L \leqslant N \text{ and} \\ \left[\overline{B}\right]_{ij} &= h(i-j+L-N), \quad i = \overline{1,N}, \; j = \overline{1,L}, \text{ if } L > N. \end{aligned} \tag{18}$$

The matrix $\overline{B}$ is formed from the values of the impulse response of filter (2)

$$\overline{B} = \begin{bmatrix} h(L-N) & \ldots & h(0) & & & \\ & & & & \mathbf{0} & \\ \vdots & & \vdots & \ddots & & \\ & & & & h(0) & \\ h(L-1) & \ldots & h(N-1) & \ldots & h(1) & h(0) \end{bmatrix}. \tag{19}$$

5. Example. Consider the example of an all-pole second order stable IIR filter with poles at the points $\frac{3}{4}$ and $\frac{1}{2}$ (Parhi and Messerschmitt, 1989b). This filter is described by the transfer function

$$H(z) = \frac{Y(z)}{U(z)} = \frac{1}{1 - \frac{5}{4}z^{-1} + \frac{3}{8}z^{-2}} = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}}. \tag{20}$$

The values of the impulse response are computed from Eq. 15

$$\begin{aligned} &h(0)=1,\ \ h(1)=\frac{5}{4},\ \ h(2)=\frac{19}{16},\\ &h(3)=\frac{65}{64},\ \ h(4)=\frac{211}{256},\ \ \ldots . \end{aligned} \tag{21}$$

State space recursive form of (20) is

$$x(k+1)=Ax(k)+bu(k)$$

or

$$\begin{bmatrix} y(k-1)\\ y(k) \end{bmatrix} = \begin{bmatrix} 0 & 1\\ -\frac{3}{8} & \frac{5}{4} \end{bmatrix} \begin{bmatrix} y(k-2)\\ y(k-1) \end{bmatrix} + \begin{bmatrix} 0\\ 1 \end{bmatrix} u(k). \tag{22}$$

A 2-stage pipelined equivalent recursive digital filter in state space is

$$x(k+2)=A^2x(k)+\overline{B}U(k),$$

or

$$\begin{aligned} \begin{bmatrix} y(k)\\ y(k+1) \end{bmatrix} &= \begin{bmatrix} a_2 & a_1\\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} y(k-2)\\ y(k-1) \end{bmatrix}\\ &+ \begin{bmatrix} h(0) & 0\\ h(1) & h(0) \end{bmatrix} \begin{bmatrix} u(k)\\ u(k+1) \end{bmatrix}. \end{aligned} \tag{23}$$

Using Eq. 16, we get

$$a_{21}=\sum_{l=1}^{1}a_{3-l}h(l)=a_2h(1)=-\frac{3}{8}\cdot\frac{5}{4}=-\frac{15}{32},$$

$$a_{22}=\sum_{l=1}^{2}a_{3-l}h(l-1)=a_2h(0)+a_1h(1)=-\frac{3}{8}+\frac{5}{4}\cdot\frac{5}{4}=\frac{19}{16}.$$

Then

$$\begin{bmatrix} y(k)\\ y(k+1) \end{bmatrix} = \begin{bmatrix} -\frac{3}{8} & \frac{5}{4}\\ -\frac{15}{32} & \frac{19}{16} \end{bmatrix} \begin{bmatrix} y(k-2)\\ y(k-1) \end{bmatrix} + \begin{bmatrix} 1 & 0\\ \frac{5}{4} & 1 \end{bmatrix} \begin{bmatrix} u(k)\\ u(k+1) \end{bmatrix}. \tag{24}$$

We compute the eigenvalues of matrix A^2

$$\begin{vmatrix} -\frac{3}{8}-\lambda & \frac{5}{4} \\ -\frac{15}{32} & \frac{19}{16}-\lambda \end{vmatrix} = \lambda^2 - \frac{13}{16}\lambda + \frac{18}{128} = 0. \tag{25}$$

From Eq. 25 we have $\lambda_1 = \frac{9}{16}$ and $\lambda_2 = \frac{1}{4}$. 2-stage pipelined equivalent state space recursive digital filter (24) is stable, because $|\lambda_1| < 1$ and $|\lambda_2| < 1$.

A 3-stage pipelined equivalent recursive digital filter in state space is

$$x(k+3) = A^3x(k) + \left[A^2b, Ab, b\right]\left[u(k), u(k+1), u(k+2)\right]^T \tag{26}$$

or

$$\begin{bmatrix} y(k+2) \\ y(k+3) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} y(k-2) \\ y(k-1) \end{bmatrix} + \begin{bmatrix} h(2) & h(1) & 0 \\ h(3) & h(2) & h(1) \end{bmatrix} \left[u(k), u(k+1), u(k+2)\right]^T . \tag{27}$$

We find from Eq. 17 that

$$a_{11} = \sum_{l=1}^{1} a_{3-l}h(l) = a_2h(1) = -\frac{3}{8}\cdot\frac{5}{4} = -\frac{15}{32},$$

$$a_{21} = \sum_{l=1}^{1} a_{3-l}h(l+1) = a_2h(2) = -\frac{3}{8}\cdot\frac{19}{16} = -\frac{57}{128},$$

$$a_{12} = \sum_{l=1}^{2} a_{3-l}h(l-1) = a_2h(0) + a_1h(1) = -\frac{3}{8} + \frac{5}{4}\cdot\frac{5}{4} = \frac{19}{16} = h(2),$$

$$a_{22} = \sum_{l=1}^{2} a_{3-l}h(l) = a_2h(1) + a_1h(2) = -\frac{3}{8}\cdot\frac{5}{4} + \frac{5}{4}\cdot\frac{19}{16} = \frac{65}{64} = h(3).$$

Then

$$\begin{bmatrix} y(k+2) \\ y(k+3) \end{bmatrix} = \begin{bmatrix} -\frac{15}{32} & \frac{19}{16} \\ -\frac{57}{128} & \frac{65}{64} \end{bmatrix} \begin{bmatrix} y(k-2) \\ y(k-1) \end{bmatrix} + \begin{bmatrix} \frac{5}{4} & 1 & 0 \\ \frac{19}{16} & \frac{5}{4} & 1 \end{bmatrix} \left[u(k), u(k+1), u(k+2)\right]^T . \tag{28}$$

We compute the eigenvalues of matrix A^3

$$\begin{vmatrix} -\frac{15}{32}-\lambda & \frac{19}{16} \\ -\frac{57}{128} & \frac{65}{64}-\lambda \end{vmatrix} = \lambda^2 - \frac{35}{64}\lambda + \frac{108}{2048} = 0. \qquad (29)$$

From Eq. 29 we get $\lambda_1 = \frac{27}{64}$ and $\lambda_2 = \frac{1}{8}$. 3-stage pipelined equivalent state space recursive digital filter (28) is stable, because $|\lambda_1| < 1$ and $|\lambda_2| < 1$.

6. Concluding remarks. Pipelined equations for output computation are derived. The stability of a filter, described by a pipelined-block equation, is studied. The expressions of computing multiplication complexity for one output value are presented. The formulas for computing matrices $\overline{A}$ and $\overline{B}$ in a pipelined-block equation are given.

The pipelined-block implementation of a direct form recursive digital filter is more stable than that of the direct form recursive digital filter. This implementation requires a linear complexity with respect to the number of loop pipeline stages and filter order. For small filter orders and for large speedups, it is preferable to use scattered look-ahead pipelining with a decomposition. For large filter orders and for small speedups, it is preferable to use the pipelined-block implementation. In the case when the filter order is equal to the number of pipelined stages, it is preferable to use the pipelined-block implementation.

REFERENCES

Azimi-Sadjadi, M.R., and R.A. King (1986). Two-dimensional block processors – structures and implementations. *IEEE Trans. Circuits Syst.,* **33**, 42–50.

Azimi-Sadjadi, M.R., and A.R. Rostampour (1989). Parallel and pipeline architectures for 2-D block processing. *IEEE Trans. Circuits Syst.,* **36**, 443–448.

Barnes, C.W., and S. Shinnaka (1980). Block shift invariance and block implementation of discrete-time filters. *IEEE Trans. Curcuits Syst.,* **27**, 667–672.

Burrus, C.S. (1971). Block implementation of digital filters. *IEEE Trans. Circuit Theory,* **18**, 697–701. .

Cappello, P.R., and K. Steiglitz (1983). Completely-pipelined architectures for digital signal processing. *IEEE Trans. Acoust., Speech, Signal Processing,* **31**, 1016–1023.

Chung, J.G., and K.K. Parhi (1994). Pipelining of lattice IIR digital filters. *IEEE Trans. on Signal Processing,* **42**, 751–761.

Isermann, R. (1981). *Digital Control Systems.* Springer-Verlag, Berlin.

Jump, J.R., and S.R. Ahuja (1978). Effective pipelining of digital systems. *IEEE Trans. Comput.,* **27**, 855–865.

Lim, Y.C., and Bede Liu (1992). Pipelined recursive filter with minimum order augmentation. *IEEE Trans. on Signal Processing,* **40**, 1643–1651.

Lucke, L.E., and K.K. Parhi (1994). Parallel processing architectures for rank order and stack filters. *IEEE Trans. on Signal Processing,* **42**, 1178–1189.

Nikias, C.L. (1984). Fast block data processing via a new IIR digital filter structure. *IEEE Trans. Acoust. Speech, Signal Processing,* **32**, 770–779.

Parhi, K.K., and D.G. Messerschmitt (1989a). Concurrent architectures for two-dimensional recursive digital filtering. *IEEE Trans. Circuits Syst.,* **36**, 813–829.

Parhi, K.K., and D.G. Messerschmitt (1989b). Pipeline interleaving and parallelism in recursive digital filters – Part I: Pipelining using scattered look-ahead and decomposition. *IEEE Trans. Acoust., Speech, Signal Processing,* **37**, 1099–1117.

Parhi, K.K., and D.G. Messerschmitt (1989c). Pipeline interleaving and parallelism in recursive digital filters – Part II: Pipelined incremental block filtering. *IEEE Trans. Acoust., Speech, Signal Processing,* **37**, 1118–1134.

Received February 1996

K. Kazlauskas received Ph. D. degree from Kaunas Polytechnic Institute (Kaunas, Lithuania) in 1975. He is a senior researcher of the Technological Process Control Department at the Institute of Mathematics and Informatics. His research interests include design of concurrent algorithm and architectures for signal processing, and computer aided design of signal processing systems.

REKURSINIŲ SKAITMENINIŲ FILTRŲ KONVEJERIZAVIMAS

Kazys KAZLAUSKAS

Straipsnyje nagrinėjamas rekursinių skaitmeninių filtrų konvejerizavimo metodas. Rekursinis filtras aprašomas būsenų lygtimi. Iš jos gaunama konvejerinio rekursinio filtro lygtis. Analizuojamas konvejerinio filtro stabilumas bei sudėtingumas ir palyginama su kitų metodu sudėtingumais. Pasiūlytas konvejerinių matricų iteratyvus apskaičiavimo algoritmas.