Parallel and Sequential Block Kalman Filtering and Their Implementations Using Systolic Arrays

Mahmood R. Azimi-Sadjadi, *Senior Member, IEEE*, Tongxin Lu, and Eduardo M. Nebot, *Member, IEEE*

Abstract—**Two sets of block Kalman filtering equations are derived that differ in the manner of generating the initial and updated estimates. Parallel and sequential schemes for generating these estimates are adopted. It is shown that the parallel implementation inherently leads to a block Kalman estimator which provides filtered estimates at the vector (block) level and fixed-lag smoothed estimates at the sample level. The sequential implementation scheme, on the other hand, generates the estimates of each sample recursively, leading naturally to a scalar (filter) estimator. These scalar estimates are arranged in a vector form, resulting in a block estimator which solely generates filtered estimates both at the vector and sample levels. Simulation results on a speech signal are also presented which indicate the advantages of the sequential block Kalman filter. An algorithm for iterative calculation of Kalman gain and error covariance matrices is given which does not require any matrix inversion operation. The implementation of this algorithm using available systolic array processors is presented. A ring systolic array is also suggested which can be used to implement the state update part of the block Kalman filter.**

I. Introduction

BLOCK processing has been applied to numerous areas in digital signal processing [1]–[9] and control [10]. This is to a great extent due to its advantages and utilities in performing parallel processing with increased throughput rate, increased computational efficiency, reduced roundoff error, and sensitivity performance [1]–[4]. Barnes and Shinnaka [4] used the concept of block processing to arrive at a block state-space formulation with states that propagate only at the edge of each block rather than at each sample. Their block state-space model has been used in a number of papers. In a paper in Anderson *et al.* [5], this block state-space is used as a dynamic model for modeling the signal and the observation in the Kalman filtering problem. Their block Kalman filter can be used to estimate a block of data at each iteration given the observation blocks up to the present block. Some of the advantages of this scheme which are inherent to the block processing method are reduced computational effort, and better sensitivity and stability performances. It is also shown that the block estimates are smoother as the block Kalman estimator provides fixed-lag smoothed estimates of the unblocked process. Jain and Jasiulek [6] proposed nonrecursive algorithms which utilize FFT to accomplish the operations required in linear smoothing, Riccati equations, boundary value problems, and block Kalman filtering problems. In [7] Lu *et al.* proposed several systolic architectures for block implemented 1-D FIR and IIR digital filters. These structures offer considerably higher sampling and throughput rates as compared with the single processing element. Parhi and Messerschmitt [8], introduced a look-ahead computation scheme for parallel implementation of the block state-space equation. This scheme utilizes pipelining at bit level and allows an even higher sampling rate. They have also proposed an incremental block state-space structure [9] which offers fewer computational complexities when compared to the standard and parallel block state-space structures.

In this paper two generalized sets of block Kalman filtering equations are derived. It is shown that the manner in which the data is processed determines the types of Kalman filtering equations. In particular, parallel and sequential implementation schemes are considered and the corresponding block Kalman filter equations are obtained. In the first case, the initial estimates are evaluated based upon the updated estimates in the previous block and the estimates are updated in parallel when the entire current data block is received. This scheme is found to be somewhat similar to that developed in [5]. Although the estimates are smoother because of the fixed-lag smoothing property of this structure, the oversmoothing in signal estimation applications may result in loss of some valid information. Moreover, the fixed-lag smoother is sluggish in responding to the fast transitions in the signal. In the second case, the initial estimate for each sample within the block is obtained sequentially from the updated estimates of the previous samples within the same block hence resulting in a more accurate estimation. In addition, this new formulation provides the true filtered estimate of each block without introducing any lag in the estimation. Simulation results on a speech signal are presented which indicate that the sequential block Kalman filter provides the best filtered estimates when compared with those of the original scalar Kalman filter and the parallel block Kalman filter. An iterative procedure for evaluating the Kalman gain and the error covariance matrices is presented which is ideally suited for implementation using current systolic array processors. A ring systolic array is also suggested to implement the state update process involved in any general Kalman filtering algorithm.

II. Block State-Space Formulations

Consider a discrete-time, stationary Markov process $\{x_k\}$ which is modeled by an autoregressive (AR) model of order M

$$x_k = \sum_{l=1}^{M} a_l x_{k-l} + u_k \tag{1}$$

where $\{u_k\}$ represents a white Gaussian (WG) process with zero mean and variance σ_u^2 which drives the autoregression process; and a_k's are the coefficients of the AR model. The measurement equation is given by

$$y_k = \sum_{l=0}^{M-1} h_l x_{k-l} + v_k \tag{2}$$

Manuscript received December 9, 1987; revised February 2, 1990.

M. R. Azimi-Sadjadi and T. Lu are with the Department of Electrical Engineering, Colorado State University, Fort Collins, CO 80523.

E. M. Nebot is with the Universidad Nacional del Sur, Bahia Blanca, Argentina.

IEEE Log Number 9040383.

1053-587X/91/0100-0137$01.00 © 1991 IEEE

where the sequence $\{y_k\}$ represents the observed signal; $\{h_k\}$ represents the impulse response sequence of an FIR system of order M; and $\{v_k\}$ represents the measurement or observation noise which is assumed to be WG process with zero mean and variance σ_v^2, and independent of $\{u_k\}$, i.e.,

$$E[u_k v_l] = 0 \qquad \forall k, l. \tag{3}$$

In state-space form the above equations can be written as

$$\begin{aligned} S_k &= AS_{k-1} + Bu_k, \qquad E[S_0 S_0^t] = Q_0 \\ y_k &= HS_k + v_k \end{aligned} \tag{4}$$

where

$$S_k = [x_{k-M+1} x_{k-M+2} \cdots x_k]^t$$

$$A = \begin{bmatrix} 0 & 1 & & & \bigcirc \\ 0 & 0 & 1 & & \\ \vdots & & & \ddots & \\ & & & & 1 \\ a_M & a_{M-1} & \cdots & \cdots & a_1 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$H = [h_{M-1} \cdots h_0]$$

and S_0 is a zero-mean Gaussian random vector independent of $\{u_k\}$ and $\{v_k\}$.

Now let us assume that the measurements are received in nonoverlapping blocks of N samples ($N = M$ for simplicity), and it is desirable to compute a block of filtered estimates at each stage of the algorithm. The block state-space equation can be obtained by partitioning the sequences $\{x_k\}$, $\{u_k\}$, $\{y_k\}$, and $\{v_k\}$ into nonoverlapping blocks of size M, letting $k = iM + j$ and writing the state-space equation (4) for all $j \in [0, M-1]$. This would yield

$$\begin{aligned} X_i &= \tilde{A}X_{i-1} + \tilde{B}U_i \\ Y_i &= \tilde{H}X_{i-1} + \tilde{D}U_i + V_i \end{aligned} \tag{5}$$

where

$$\tilde{A} = A^M$$

$$\tilde{B} = [A^{M-1}B \quad A^{M-2}B \cdots B]$$

$$\tilde{H} = \begin{bmatrix} HA \\ HA^2 \\ \vdots \\ HA^M \end{bmatrix}, \qquad \tilde{D} = H_c\tilde{B},$$

$$H_c = \begin{bmatrix} h_0 & & \bigcirc \\ h_1 & h_0 & \\ \vdots & & \ddots \\ h_{M-1} & \cdots & h_0 \end{bmatrix}$$

and X_i represents the ith block of signal x_k, i.e.,

$$X_i = S_{iM+M-1} = [x_{iM} \quad x_{iM+1} \cdots x_{iM+M-1}]^t$$

U_i, Y_i, and V_i are similarly defined.

In what follows, two sets of block Kalman filtering equations are derived which differ in the manner of generating the initial (*a priori*) and the updated (*a posteriori*) estimates.

III. Parallel Block Kalman Filter

In this scheme, the initial estimates of the samples within block "i" are obtained based upon the updated estimates within the previous blocks. The updating for the estimates in block "i" will take place when the entire observation block Y_i is received in parallel. Let us denote the updated (*a posteriori*) block estimate of X_i, conditioned on the observation blocks up to Y_i (or, equivalently, observation points up to y_{iM+M-1}) by $\hat{\hat{X}}_i$, i.e.,

$$\hat{\hat{X}}_i \triangleq E[X_i | Y_0 \cdots Y_i] = E[X_i | y_0 \cdots y_{iM+M-1}] \tag{6a}$$

and the initial (*a priori*) block estimate of X_i, conditioned on the past observation blocks up to Y_{i-1} (or equivalently observation points up to y_{iM-1}) by $\hat{X}_i$, i.e.,

$$\hat{X}_i \triangleq E[X_i | Y_0 \cdots Y_{i-1}] = E[X_i | y_0 \cdots y_{iM-1}]. \tag{6b}$$

Then, the "block innovation sequence" $\{Z_i\}$ can be defined as

$$Z_i = Y_i - \tilde{H}\hat{\hat{X}}_{i-1}. \tag{7}$$

Note that in spite of the fact that Y_i is dependent of $\{Y_0 \cdots Y_{i-1}\}$, since $\hat{\hat{X}}_{i-1}$ is obtained based upon the observation set $\{Y_0 \cdots Y_{i-1}\}$, the block innovation sequence Z_i will contain only the "new information" in Y_i and thus is orthogonal to $\{Y_0 \cdots Y_{i-1}\}$. Now, utilizing the linearity property of the minimum variance estimator (MVE) we get

$$\begin{aligned} \hat{\hat{X}}_i &= \hat{X}_i + E[X_i | Z_i] \\ &= \hat{X}_i + E[X_i | z_{iM} \cdots z_{iM+M-1}] \end{aligned} \tag{8a}$$

where the elements of Z_i, i.e., z_{iM+j} are given by

$$\begin{aligned} z_{iM+j} &= y_{iM+j} - H\hat{S}_{iM+j} \\ &= y_{iM+j} - HA^{j+1}\hat{\hat{X}}_{i-1} \qquad j \in [0, M-1]. \end{aligned} \tag{8b}$$

The initial estimate for each element of a block is obtained recursively based upon observations up to y_{iM-1}, i.e.,

$$\begin{aligned} \hat{S}_{iM+j} &= [\hat{\hat{x}}_{(i-1)M+j+1} \cdots \hat{\hat{x}}_{iM-1} \hat{x}_{iM} \cdots \hat{x}_{iM+j}]^t \\ &= A\hat{S}_{iM+j-1} = A^{j+1}\hat{\hat{X}}_{i-1}, \qquad j \in [0, M-2]. \end{aligned} \tag{9a}$$

For $j = M-1$

$$\begin{aligned} \hat{S}_{iM+M-1} &= [\hat{x}_{iM} \quad \hat{x}_{iM+1} \cdots \hat{x}_{iM+M-1}]^t \\ &= \hat{X}_i. \end{aligned} \tag{9b}$$

Thus we obtain

$$\hat{X}_i = \tilde{A}\hat{\hat{X}}_{i-1}. \tag{10}$$

Remark III-1: Note that since $\hat{S}_{iM+j}$, $j \in [0, M-1]$ is obtained based upon the observation points up to y_{iM-1}, z_{iM+j} does not solely represent the "new part" of y_{iM+j}. Thus

$$E[z_{iM+k} z_{iM+l}] \neq 0 \qquad k \neq l \qquad k, l \in [0, M-1]$$

whereas

$$E[z_{iM+j} y_{iM-l}] = 0 \qquad \forall j \in [0, M-1] \qquad \forall l \geq 1. \tag{11}$$

This fact indicates that although the block innovation sequence $\{Z_i\}$ represents a vectorially uncorrelated process, the elements of Z_i are mutually correlated. In other words, Z_i can be regarded as a multichannel process [11] with mutual coupling between the individual channels. This contradicts with the scalar nature of the original signal and observation models.

Remark III.2: The block estimate $\hat{\hat{X}}_i$ is a vector containing the smoothed estimates of the scalar signal x_k having different lags depending on the position of the particular element, i.e.,

$$\hat{\hat{X}}_i = \begin{bmatrix} E[x_{iM} | y_0 \cdots y_{iM+M-1}] \\ E[x_{iM+1} | y_0 \cdots y_{iM+M-1}] \\ \vdots \\ E[x_{iM+M-1} | y_0 \cdots y_{iM+M-1}] \end{bmatrix}$$

$$= \begin{bmatrix} E[x_{iM} | y_0 \cdots y_{iM-1}] \\ \vdots \\ E[x_{iM+M-1} | y_0 \cdots y_{iM-1}] \end{bmatrix}$$

$$+ \begin{bmatrix} E[x_{iM} | Z_i] \\ \vdots \\ E[x_{iM+M-1} | Z_i] \end{bmatrix}. \tag{12}$$

Using the orthogonal projection lemma [12] and considering that the updating takes place based upon the entire vector Z_i, we have

$$E[X_i | Z_i] = E[X_i Z_i^t] \left\{ E[Z_i Z_i^t] \right\}^{-1} Z_i. \tag{13}$$

Defining

$$K(i) \triangleq E[X_i Z_i^t] \left\{ E[Z_i Z_i^t] \right\}^{-1} \tag{14}$$

as the "Kalman gain matrix," the updating equation (8a) becomes

$$\hat{\hat{X}}_i = \hat{X}_i + K(i) Z_i. \tag{15}$$

New Kalman filter equations are derived which account for the presence of the direct feedthrough term with gain $\tilde{D}$ in (5). These formulations that differ in complexity from those obtained in [5] are given in order as

$$\hat{X}_i = \tilde{A} \hat{\hat{X}}_{i-1} \tag{16a}$$

$$R_z(i) \triangleq E[Z_i Z_i^t] = \tilde{H} P_a(i-1) \tilde{H}^t + \tilde{D} Q_U \tilde{D}^t + Q_V \tag{16b}$$

$$K(i) = (\tilde{A} P_a(i-1) \tilde{H}^t + \tilde{B} Q_U \tilde{D}^t) R_z^{-1}(i) \tag{16c}$$

$$\hat{\hat{X}}_i = \hat{X}_i + K(i) [Y_i - \tilde{H} \hat{\hat{X}}_{i-1}] \tag{16d}$$

$$P_b(i) \triangleq E[\tilde{X}_i \tilde{X}_i^t] = \tilde{A} P_a(i-1) \tilde{A}^t + \tilde{B} Q_U \tilde{B}^t \tag{16e}$$

$$P_a(i) \triangleq E[\tilde{\tilde{X}}_i \tilde{\tilde{X}}_i^t] = P_b(i) - K(i) [\tilde{H} P_a(i-1) \tilde{A}^t + \tilde{D} Q_U \tilde{B}^t] \tag{16f}$$

where $\tilde{X}_i \triangleq X_i - \hat{X}_i$ and $\tilde{\tilde{X}}_i \triangleq X_i - \hat{\hat{X}}_i$; $P_a(i)$ and $P_b(i)$ are, respectively, the *a posteriori* and the *a priori* error covariance matrices; Q_U and Q_V are the covariance matrices of $\{U_i\}$ and $\{V_i\}$ block sequences, given by

$$E[U_i U_j^t] = Q_U \delta(i-j) = \sigma_u^2 I \delta(i-j)$$

$$E[V_i V_j^t] = Q_V \delta(i-j) = \sigma_v^2 I \delta(i-j) \tag{17}$$

and $\delta(\cdot)$ represents the Kronecker delta function.

IV. Sequential Block Kalman Filter

This case differs from the previous one in a number of ways. First, the initial estimate for each sample within a block is formed based upon the updated estimates of all the past samples including the ones in the same block. For example, in finding $\hat{x}_{iM+j}$ we use all the updated estimates $\hat{\hat{x}}_l$, $l = iM - M + j, \cdots, iM + j - 1$, that are better than $\hat{x}_l$ used in the previous case. Thus, the estimates generated using this structure are more reliable and accurate. Second, the updating in this case takes place sequentially on each sample within a block as opposed to the parallel method where the updating is done vectorwise. Each processor estimates based upon the relevant observation point available to it, e.g., first processor estimates based upon the first element of the observation block; second processor estimates based upon the first and second observation points, etc. In contrast to the parallel method this scheme generates purely filtered estimates both at the vector and the scalar levels. The updated block estimate in this case is given by

$$\hat{\hat{X}}_i \triangleq \begin{bmatrix} E[x_{iM} | y_0 \cdots y_{iM}] \\ E[x_{iM+1} | y_0 \cdots y_{iM+1}] \\ \vdots \\ E[x_{iM+M-1} | y_0 \cdots y_{iM+M-1}] \end{bmatrix} \tag{18}$$

which represents the scalar estimates arranged in a block form. The initial block estimate is then given by

$$\hat{X}_i \triangleq \begin{bmatrix} E[x_{iM} | y_0 \cdots y_{iM-1}] \\ E[x_{iM+1} | y_0 \cdots y_{iM}] \\ \vdots \\ E[x_{iM+M-1} | y_0 \cdots y_{iM+M-2}] \end{bmatrix}. \tag{19}$$

Applying the linearity property of MVE to each element yields

$$\hat{\hat{X}}_i = \hat{X}_i + \begin{bmatrix} E[x_{iM} | z_{iM}] \\ E[x_{iM+1} | z_{iM+1}] \\ \vdots \\ E[x_{iM+M-1} | z_{iM+M-1}] \end{bmatrix} \tag{20a}$$

where scalar innovation sequence is

$$z_{iM+j} = y_{iM+j} - H \hat{S}_{iM+j} \qquad j \in [0, M-1]. \tag{20b}$$

Note that in this case $\hat{S}_{iM+j}$ is given by

$$\hat{S}_{iM+j} = [\hat{\hat{x}}_{(i-1)M+j+1} \cdots \hat{\hat{x}}_{iM+j-1} \hat{x}_{iM+j}]^t$$

$$= A \hat{\hat{S}}_{iM+j-1} \qquad j \in [0, M-1]. \tag{21}$$

Also we have

$$\hat{\hat{S}}_{iM-1} = \hat{\hat{X}}_{i-1} \tag{22a}$$

and

$$\hat{\hat{S}}_{iM+M-1} = \hat{\hat{X}}_i. \tag{22b}$$

Remark IV.1: In this case $\hat{S}_{iM+j}$, $j \in [0, M-1]$, in the scalar innovation is obtained based upon the updated estimates $\hat{\hat{x}}_l$, $l = iM - M + j, \cdots, iM + j - 1$, which are in turn evaluated based upon the observation points up to y_{iM+j-1}. That is, z_{iM+j} does indeed represent the "new part" of y_{iM+j} and thus is independent of all the observation points in the set $\{y_0 \cdots y_{iM+j-1}\}$, i.e.,

$$E[z_{iM+j} y_l] = 0 \qquad \forall l \in [0, iM + j - 1]$$

and

$$E[z_k z_l] = 0 \qquad k \neq l, \qquad \forall k, l \tag{23}$$

This implies that the sequence $\{z_k\}$ represents an uncorrelated scalar process. This certainly matches more closely to the scalar nature of the original signal and observation models.

Applying the linearity property of the estimator to each scalar estimate yields

$$E[S_{iM+j} | y_0 \cdots y_{iM+j}] = E[S_{iM+j} | y_0 \cdots y_{iM+j-1}] + E[S_{iM+j} | z_{iM+j}]$$

or

$$\hat{S}_{iM+j} = \hat{S}_{iM+j} + E[S_{iM+j} | z_{iM+j}]. \tag{24}$$

Now using the orthogonal projection theorem [12] the second term on the right-hand side of (24) can be expressed as

$$E[S_{iM+j} | z_{iM+j}] = E[S_{iM+j} z_{iM+j}] \{E[z_{iM+j} z_{iM+j}]\}^{-1} z_{iM+j} = K_i(j) z_{iM+j} \tag{25}$$

where $K_i(j)$ is the Kalman gain matrix for the jth element in block "i." The Kalman filter equations for this structure are derived to be

$$\hat{S}_{iM+j} = A\hat{S}_{iM+j-1} \qquad j \in [0, M-1] \tag{26a}$$

$$P_b(iM+j) = AP_a(iM+j-1)A^t + BB^t\sigma_u^2 \tag{26b}$$

$$K_i(j) = P_b(iM+j)H^t[HP_b(iM+j)H^t + \sigma_v^2]^{-1} \tag{26c}$$

$$z_{iM+j} = y_{iM+j} - H\hat{S}_{iM+j}, \qquad j \in [0, M-1] \tag{26d}$$

$$\hat{S}_{iM+j} = \hat{S}_{iM+j} + K_i(j) z_{iM+j} \tag{26e}$$

$$P_a(iM+j) = [I - K_i(j)H] P_b(iM+j). \tag{26f}$$

Now writing (26e) for $j \in [0, M-1]$, i.e., for each element of block "i" and expressing all the initial estimate vectors in terms of $\hat{X}_{i-1}$ yields

$$\hat{S}_{iM+M-1} = \hat{X}_i = A^M \hat{X}_{i-1} + \tilde{K}_i Z_i \tag{27}$$

where

$$\tilde{K}_i \triangleq [A^{M-1}K_i(0) \cdots K_i(M-1)].$$

Similarly writing (26d) in block form and using (21) and (22) gives

$$Z_i = Y_i - \tilde{H}\hat{X}_{i-1} - \tilde{R}_i Z_i \tag{28}$$

where $\tilde{H}$ is defined in (5) and $\tilde{R}_i$ is given by its (k, l) elements as

$$\tilde{R}_i(k, l) = \begin{cases} 0 & k \leq l \\ HA^{k-l}K_i(l-1) & k > l \end{cases} \qquad k, l \in [1, M].$$

Equation (28) can be rewritten as

$$Z_i = \tilde{W}_i^{-1}(Y_i - \tilde{H}\hat{X}_{i-1}) \tag{29a}$$

where the nonsingular matrix $\tilde{W}_i$ is

$$\tilde{W}_i = I + \tilde{R}_i. \tag{29b}$$

Equations (27) and (29) together with (26b), (26c), and (26f) for calculating Kalman gain and error covariances represent a set of sequential block Kalman filter equations, obtained by blocking the scalar estimator equations. As can be seen, these formulations provide purely block filtered estimate. If stationarity is assumed, the steady-state Kalman gain $\tilde{K}_s$ can be obtained using (26b), (26c), and (26f). This gain matrix is then used throughout the entire updating process. Equations (27) and (29) would then represent a simplified sequential block Kalman filtering procedure.

Remark IV.2: In addition to the above benefits, this scheme provides a very efficient means of block Kalman filtering, since the Kalman gain and error covariance evaluations using (26b), (26c), and (26f) do not require any matrix inversion operation, as in the scalar case.

V. Simulation Results

In order to examine the effectiveness of the proposed block Kalman filters, a speech signal for the word "hello" is digitized at 16 KHz and used throughout the experimentation. This signal which contains 5000 points is shown in Fig. 1. This signal is then corrupted by adding white Gaussian noise with zero mean and variance 0.158 to obtain a SNR of 6.9 dB. The corrupted signal is shown in Fig. 2. The entire simulation is performed on a Mac II computer with a 68020 processor and a 68881 Math coprocessor. A fifth-order AR model is fitted to this data and the model parameters are obtained by solving the Yule–Walker equations. Using this method the variance of the driving noise sequence σ_u^2 is found to be 0.02. Standard Kalman filter is first applied to the scalar model in (4) with initial $P_a(0) = 10\,I$ and $H = [0 \;\; 0 \;\; 0 \cdots 0 \;\; 1]$. The filtered signal is shown in Fig. 3. The SNR is measured to be 9.7 dB which shows considerable noise reduction. The entire process is implemented on-line and it took 148 s. Fig. 4 shows the behavior of the parallel block Kalman filter. The effect of smoothing can clearly be seen in this result. This filter has slower response than the standard scalar Kalman filter. The degradation in performance can also be observed from the variance of the output noise that is found to be 0.286, giving a SNR of 4.3 dB which is even smaller than that of the degraded signal. Additionally, some loss of valid information is evident in the processed signal. The computational time for this process is approximately 94 s which is smaller than that of the standard Kalman filter. Again in this case the Kalman gain equations are evaluated on-line. The result of the sequential block Kalman filter is shown in Fig. 5 which indicates the effectiveness of this method when compared with the previous cases. This improvement is also reflected in the value of the variance of the noise in the output signal which is reduced to 0.058 giving a SNR of 11.25 dB. The estimates are obtained in approximately 31 s. In this case the steady state Kalman gain was obtained off-line, thus (27) and (29) are the only ones that are evaluated on-line. Overall, the tracking behavior of the sequential block Kalman filter is by far better than the parallel case. This, coupled with its accuracy and efficiency as discussed above makes this method very attractive for signal estimation applications.

VI. Implementation Using Systolic Arrays

The development of Kalman filtering algorithms suitable for implementation using systolic arrays has recently been considered in a number of papers, namely [13]–[15]. In this section, systolic array implementations of the block Kalman filter are presented. The results can be applied to any general MIMO Kalman filter structure. The proposed systolic architectures im-

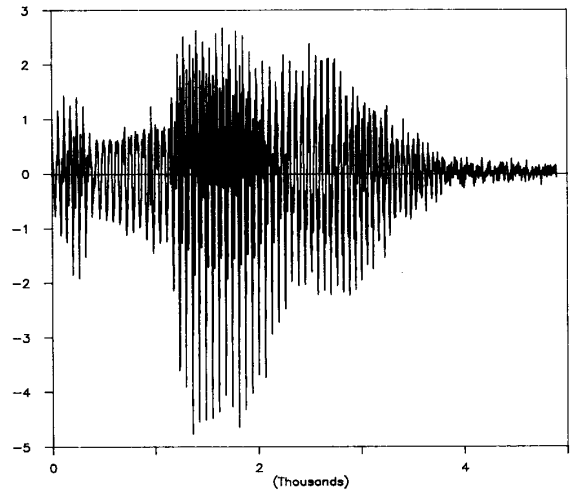


Fig. 1. Original speech signal.

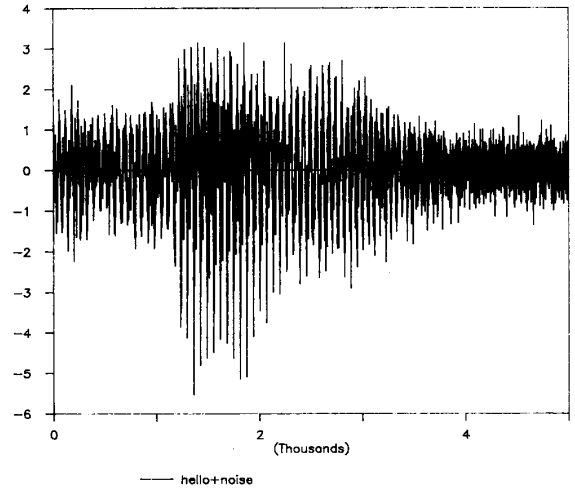


Fig. 2. Noisy speech signal (SNR = 6.9 dB).

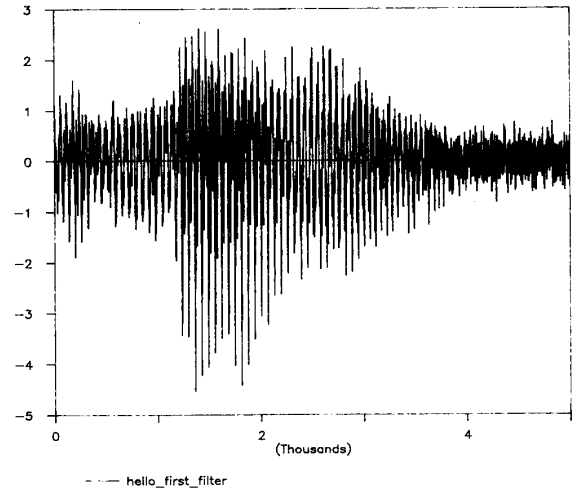


Fig. 3. Filtered speech signal using standard Kalman filter (SNR = 9.7 dB).

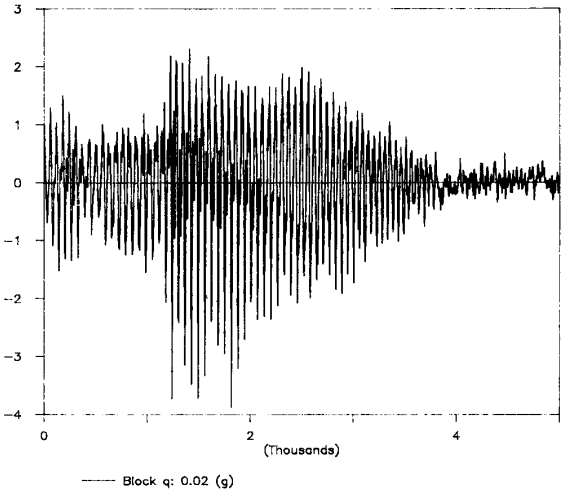


Fig. 4. Filtered speech signal using parallel block Kalman filter (SNR = 4.3 dB).

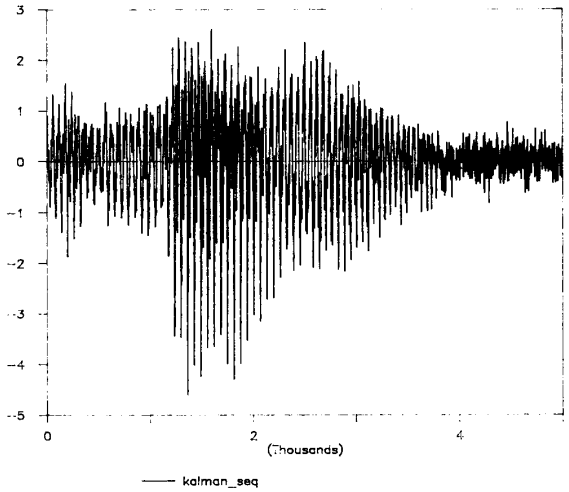


Fig. 5. Filtered speech signal using sequential block Kalman filter (SNR = 11.25 dB).

plement all the operations involved in the Kalman filtering process such as the Kalman gain evaluation, iterative update for the *a priori* and the *a posteriori* error covariance matrices and the state update procedure. An algorithm for iterative calculations of Kalman gain and the error covariance matrices is suggested which does not require any matrix inversion operation. The global convergence of this algorithm is also established. The consecutive matrix-matrix multiplications (MMM's) in this algorithm are implemented on an expanded skew pipeline structure which uses the available geometric array parallel processor (GAPP) [15], [16] type systolic arrays. The state update process is implemented using a ring systolic array structure [17]–[19]. Let us first briefly introduce the GAPP systolic array.

A. GAPP [16]

The geometric array parallel processor or GAPP is a commercial two-dimensional systolic array processor chip manufac-

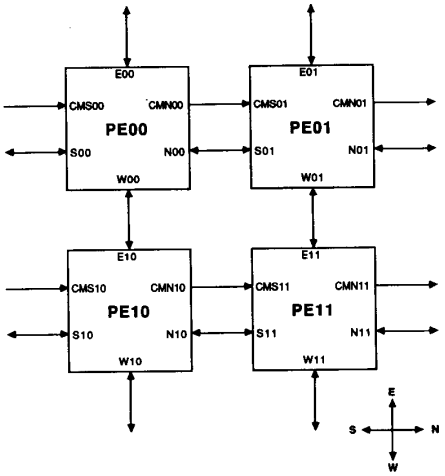


Fig. 6. Interconnections among four cells of the GAPP.

tured by NCR. It is a mesh-connected 6-by-12 arrangement of 1-b processor elements, or PE's. Each PE can communicate with four neighbors: North, East, South, and West. Each PE is composed of a 1-b serial ALU, 128 word by 1-b RAM and 4 single bit latches: three latches hold inputs to the ALU and the fourth latch allows I/O through the cell without interrupting the ALU, i.e., I/O operations are overlapped with the computations. Each instruction is broadcast to all the PE's making the array to operate like a single instruction mulitple data (SIMD) machine.

The cascadeability of the GAPP allows system designers to implement arrays of processors of arbitrary size in multiples of 6-by-12 elements. Fig. 6 shows the interconnections among four cells. The following development is based upon a generalized GAPP type systolic array and not limited by the specific size of memory and/or the dimension.

B. Consecutive Matrix-Matrix Multiplication Using GAPP

A GAPP type array is shown in Fig. 7(a). External connections are added to this array in order to perform the operations needed in this section. The resultant array is shown in Fig. 7(b). Note that the required multiplexers for the external communication links are not shown in this figure for the sake of simplicity.

Let us assume that a matrix, say A, is cyclically loaded from the bottom of the array in a row parallel fashion as shown in Fig. 8(a). The matrix stored in the array shall be denoted by A^*. Note that the diagonal shift operations in this array can be accomplished in two steps. Fig. 8(b) shows the loading of a 3 $\times$ 3 matrix. Once the loading operation is completed, all the MMM combinations of matrix A (or A') by matrix B (or B') which result in C (or C') can be computed in a skew pipelined fashion as shown in Fig. 9. Without loss of generality and to simplify the discussion here we have assumed that matrices A, B, and C are of the same size as $M \times M$. At every PE, an entry of matrix B is multiplied by the corresponding entry of matrix A and the result is added to the entry of matrix C. then all the entries of matrices B and C move to other PE's following their corresponding paths. All the possible combinations of MMM's and their data flows are shown in Table I. A 3 $\times$ 3 example for performing matrix operation BA and the types of inputting and

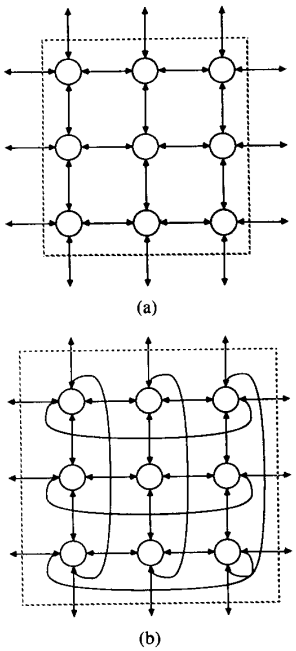


Fig. 7. (a) A 3 $\times$ 3 GAPP-type systolic array. (b) Ring connection of a GAPP-type systolic array.

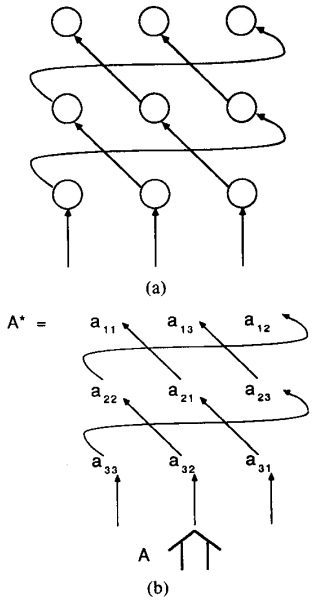


Fig. 8. (a) The path of circulant loading. (b) The loading path of matrix A.

outputting are shown in Fig. 10. As can be seen, the partial summations for matrix C is skewed for this particular operation.

For a GAPP type SIMD machine, one cannot change the stored matrix A^* before the completion of the first MMM (before the last row or column of C is generated). Thus, the efficiency of this structure is limited only to 50% if the array is to be used for consecutive MMM operations. However, C or C^* can be loaded into the same array, but with different memory

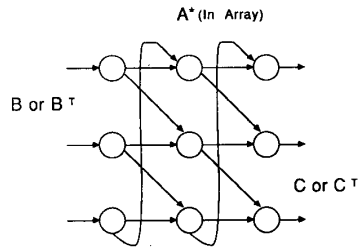


Fig. 9. The MMM operation path.

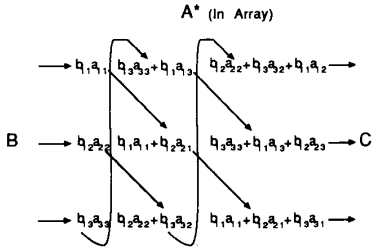


Fig. 10. A 3 × 3 example showing the data flow in performing matrix operation BA.

TABLE I
ALL EIGHT POSSIBLE COMBINATIONS OF MMM'S OF MATRICES A, B, AND C IMPLEMENTED ON THE STRUCTURE IN FIG. 9

$C =$	Type of Inputting B	Path of B	Summing Path of C	Type of Outputting C
AB	Column Parallel	Skewed	Left-to-right	Column Parallel
$A'B$	Column Parallel	Left-to-right	Skewed	Column Parallel
$B'A'$	Column Parallel	Skewed	Left-to-right	Row Parallel
$B'A$	Column Parallel	Left-to-right	Skewed	Row Parallel
BA	Row Parallel	Left-to-right	Skewed	Row Parallel
BA'	Row Parallel	Skewed	Left-to-right	Row Parallel
$A'B'$	Row Parallel	Left-to-right	Skewed	Column Parallel
AB'	Row Parallel	Skewed	Left-to-right	Column Parallel

address, during the MMM time for the next consecutive MMM operation as shown in Fig. 11. If the type of inputting of the next MMM matches the type of outputting of the present MMM (see Table I), e.g., both are row parallel, then the two MMM's can be performed using a pair of identical arrays without interruption. The relevant architecture which improves the speed by a factor of two is shown in Fig. 12. These architectures will be used to implement the consecutive MMM operations involved in the iterative update of Kalman gain and also the error covariance matrices.

In what follows, an algorithm for iterative computation of Kalman gain and error covariance matrices is proposed which avoids any matrix inversion operation.

C. An Iterative Procedure for Updating Kalman Gain and Error Covariance Matrices Using Only MMM Operations

Let us consider the implementation of the Kalman filtering algorithm in Section III. A similar algorithm can be derived for any general (MIMO) Kalman filtering process.

In [20] an algorithm is developed based upon the matrix inversion lemma which can be used to generate a first-order approximation of $R_z^{-1}(i)$ in (16c). However, the first-order approximation may not generally guarantee the global convergence of the algorithm. Additionally, the first-order approximation may result in poor filtering performance. In this section, inspired by the method in [20], an iterative procedure is suggested which can generate $R_z^{-1}(i)$ using only MMM's with any arbitrary accuracy.

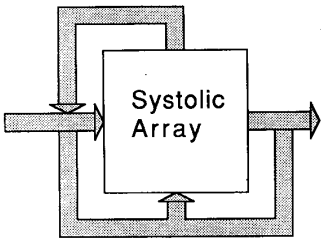


Fig. 11. The external data flow of a systolic array.

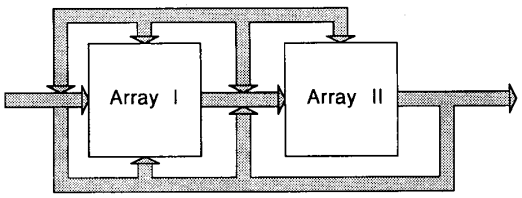


Fig. 12. A pair of systolic arrays.

Recall from (16b) that

$$R_z(i) = \tilde{H}P_a(i-1)\tilde{H}^t + \tilde{D}Q_U\tilde{D}^t + Q_V. \tag{30}$$

In this expression all the terms except $P_a(i-1)$ are constants. If the Kalman filter is assumed to be convergent (the convergence issue will be discussed later), then the elements of P_a matrix decrease from one iteration to the next. Thus we can write

$$R_z(i) = R_z(i-1) - \Delta R_z(i-1) \tag{31a}$$

where the error matrix $\Delta R_z(i-1)$ is

$$\begin{aligned}\Delta R_z(i-1) &= \tilde{H}\big(P_a(i-2) - P_a(i-1)\big)\tilde{H}^t \\ &= \tilde{H}\Delta P_a(i-1)\tilde{H}^t. \end{aligned}\tag{31b}$$

The inverse of $R_z(i)$ can also be written as

$$R_z^{-1}(i) = R_z^{-1}(i-1) + \Delta R_z^{-1}(i-1). \tag{32}$$

Note that $\Delta R_z^{-1} \neq (\Delta R_z)^{-1}$. Premultiplying (32) by (31a) gives

$$\begin{aligned}\Delta R_z^{-1}(i-1) = {} & R_z^{-1}(i-1)\,\Delta R_z(i-1)R_z^{-1}(i-1) \\ & + \Delta R_z^{-1}(i-1)\,\Delta R_z(i-1)R_z^{-1}(i-1).\end{aligned}\tag{33}$$

If we ignore the effects of the second term on the right-hand side of (33) the algorithm in [20] will be obtained. Now let us denote R_z^{-1} by Φ and ΔR_z^{-1} by $\Delta\Phi$, then the following iterative

procedure for solving (33) can be used:

$$\Delta\Phi_k(i-1) = \Phi(i-1)\,\Delta R_z(i-1)\Phi(i-1) + \Delta\Phi_{k-1}(i-1)\,\Delta R_z(i-1)\Phi(i-1) \quad (34)$$

where $\Delta\Phi_k(i-1)$ is the $(k+1)$th order solution of (33) at ith iteration of the filtering algorithm. For $k = 0$, the first-order approximation is

$$\Delta\Phi_0(i-1) = \Phi(i-1)\,\Delta R_z(i-1)\Phi(i-1) \quad (35)$$

which is the solution used in [20].

Expression (34) can be rewritten as

$$\Delta\Phi_k(i-1) = \Phi_{k-1}(i)\,\Delta R_z(i-1)\Phi(i-1), \quad \forall k \geq 0 \quad (36a)$$

where

$$\Phi_{k-1}(i) \triangleq \Phi(i-1) + \Delta\Phi_{k-1}(i-1) \quad (36b)$$

with $\Phi_{-1}(i) = \Phi(i-1)$.

Note that $\Phi_{k-1}(i)$ is the kth order estimate of $\Phi(i)$. Using (36) we have

$$\Phi_k(i) = \Phi(i-1) + \Delta\Phi_k(i-1) = \left[I + \Phi_{k-1}(i)\,\Delta R_z(i-1)\right]\Phi(i-1), \quad \forall k \geq 0. \quad (37)$$

This offers a recursive equation for generating the estimate of $R_z^{-1}(i)$ (or $\Phi(i)$) with any arbitrary accuracy given the matrices $\Delta R_z(i-1)$ and $\Phi(i-1)$. The error matrix $\Delta R_z(i-1)$ is computed from (31b) and $\Phi(i-1)$ is the final solution of Φ (or R_z^{-1}) at filtering iteration $(i-1)$.

Lemma VI.1: The recursive algorithm in (37) is globally convergent in the sense that

$$\lim_{k\to\infty} \left\|\Phi_k(i) - \Phi_{k-1}(i)\right\| = 0. \quad (38)$$

Proof: From (37) we can write

$$\Phi_k(i) - \Phi_{k-1}(i) = \left[\Phi_{k-1}(i) - \Phi_{k-2}(i)\right]\Delta R_z(i-1)\Phi(i-1) = \left[\Phi_1(i) - \Phi_0(i)\right]\left[\Delta R_z(i-1)\Phi(i-1)\right]^{k-1}. \quad (39)$$

Now, using (34)–(36) yields

$$\Phi_k(i) - \Phi_{k-1}(i) = \Phi(i-1)\left[\Delta R_z(i-1)\Phi(i-1)\right]^{k+1}. \quad (40)$$

Taking the norm of both sides of (40) gives the following inequality:

$$\left\|\Phi_k(i) - \Phi_{k-1}(i)\right\| \leq \left\|\Phi(i-1)\right\|\left\|\Delta R_z(i-1)\Phi(i-1)\right\|^{k+1} \leq \lambda_{\min}^{-1}\left[R_z(i-1)\right]\left\|\Delta R_z(i-1)\Phi(i-1)\right\|^{k+1} \quad (41)$$

where $\lambda_{\min}[R_z(i-1)]$ represents the smallest eigenvalue of $R_z(i-1)$. The second term on the right-hand side is

$$\Delta R_z(i-1)\Phi(i-1) = (\tilde{H}\Delta P_a(i-1)\tilde{H}^t)\left[\tilde{H}P_a(i-1)\tilde{H}^t + \tilde{D}Q_U\tilde{D}^t + Q_V\right]^{-1}. \quad (42)$$

Thus it is clear that

$$\left\|\Delta R_z(i-1)\Phi(i-1)\right\| \leq 1.$$

Additionally, due to the presence of Q_V and $\tilde{D}Q_U\tilde{D}^t$ terms in $R_z(i-1)$

$$\lambda_{\min}^{-1}\left[R_z(i-1)\right] \leq 1.$$

Thus

$$\lim_{k\to\infty} \left\|\Phi_k(i) - \Phi_{k-1}(i)\right\| = 0.$$

Q.E.D.

As a result, the iterative procedure for on-line computation of the Kalman gain and error covariance matrices using the above algorithm becomes:

Step 1: Compute the error matrix

$$\Delta R_z(i-1) = \tilde{H}\left[P_a(i-2) - P_a(i-1)\right]\tilde{H}^t. \quad (43a)$$

Step 2: Iterate

$$\Phi_k(i) = \left[I + \Phi_{k-1}(i)\,\Delta R_z(i-1)\right]\Phi(i-1), \quad \forall k \geq 0 \quad (43b)$$

with

$$\Phi_{-1}(i) = \Phi(i-1) \quad (43c)$$

for $k = 0, 1, \cdots$, until for $k = k^*$ (say) we have

$$\Delta\phi_{k^*}^2(i)_{m,n} \triangleq \left[\phi_{k^*}(i)_{m,n} - \phi_{k^*-1}(i)_{m,n}\right]^2 < \epsilon, \quad \forall m, n \in [1, M]$$

where $\phi_{k^*}(i)_{m,n}$ represents (m, n)th entry of $\Phi_{k^*}(i)$ matrix and ϵ is a small positive threshold. Then set $\Phi(i) = \Phi_{k^*}(i)$ and proceed.

Step 3: Evaluate Kalman gain and error covariances using

$$K(i) = (\tilde{A}P_a(i-1)\tilde{H}^t + \tilde{B}Q_U\tilde{D}^t)\Phi(i) \quad (43d)$$

$$P_a(i) = \tilde{A}P_a(i-1)\tilde{A}^t + \tilde{B}Q_U\tilde{B}^t - K(i)\left[\tilde{H}P_a(i-1)\tilde{A}^t + \tilde{D}Q_U\tilde{B}^t\right]. \quad (43e)$$

Repeat these steps for all i.

This algorithm is very well suited for implementations using the systolic array architectures designed in Section VI-B. This is considered in the next section.

D. Systolic Array Implementation of Kalman Filter Equations

Using the structure in Fig. 12 an implementation scheme can be devised which provides minimum computation time when the inputting and outputting patterns are matched in accordance with Table II. The critical (longest) path of successive matrix operations in algorithm of (43) when implemented on this structure can be as small as 5.5 MMM time. This particular case occurs when the first-order approximation of $\Phi(i)$ would suffice. A MMM time is defined as the time elapsed between the first vector of the input matrix entering the array and the last vector of the output matrix leaving the array. The decision rule in step 2 of the algorithm can be implemented using the global output of the GAPP type architecture [16]. Assuming that k^* is the value of k at which the appropriate solution for $\Phi(i)$ is obtained, then from Table II one iteration period is

$$T = (4k^* + 11)T_i \quad (44a)$$

TABLE II
SCHEDULE FOR COMPUTING KALMAN GAIN AND ERROR COVARIANCE MATRICES USING THE ARCHITECTURE IN FIG. 12

	Array I			Array II		
MMM Period	Input Type	Operation	Output Type	Input Type	Operation	Output Type
T_1	C	$L_1 = \tilde{H}^* P_a(i-1)$	C	C	$M_1 = \tilde{A}^* P_a(i-1)$	C
T_2						
T_3	R	$L_3(i) = L_1 \tilde{H}^{*t}$	R	R	$M_3 = M_1 \tilde{H}^{*t}$	R
T_4		$L_4(i) = L_3(i) - L_3(i-1)$			$M_4 = M_3 + (\tilde{B} Q_U \tilde{D}^t)$	
T_5	R	$L_5 = \Phi(i-1)\Delta R_z^*(i-1)$	R	R	$M_5 = M_1 \tilde{A}^{*t}$	R
T_6		$L_6 = I + L_5$			$M_6 = M_5 + \tilde{B} Q_U \tilde{B}^t$	
T_7	C	$L_7 = L_6^* \Phi(i-1) = \Phi_0(i)$	C			
T_8		$\Delta\phi_0^2(m,n) > \epsilon$		C	$M_8 = M_4^* L_7$	C
T_9	R	$L_9 = \Phi_0(i)\Delta R_z^*(i-1)$	R		M_8 Invalid	
T_{10}		$L_{10} = I + L_9$				
T_{11}	C	$L_{11} = L_{10}^* \Phi(i-1) = \Phi_1(i)$	C			
T_{12}		$\Delta\phi_1^2(m,n) > \epsilon$		C	$M_{12} = M_4^* L_{11}$	C
T_{13}					M_{12} Invalid	
$\vdots$		$\vdots$			$\vdots$	
T_{4k^*+7}	C	$L_{4k^*+7} = L_{4k^*+6}^* \Phi(i-1) = \Phi_{k^*(i)}$	C			
T_{4k^*+8}		$\Delta\phi_{k^*}^2(m,n) < \epsilon, \quad \forall m,n$		C	$M_{4k^*+8} = M_4^* L_{4k^*+7} = K(i)$	C
T_{4k^*+9}						
T_{4k^*+10}	R	$L_{4k^*+10} = K(i) M_4^{*t}$	R			
T_{4k^*+11}		$L_{4k^*+11} = P_a(i) = M_6 - L_{4k^*+10}$				

R = Row parallel, C = Column parallel.

where T_i is one half of a MMM computation time, i.e.,

$$T_i = M(T_{\text{MUL}} + T_{\text{ADD}}) \tag{44b}$$

and T_{MUL} is the scalar multiplication time and T_{ADD} is the scalar addition time. Note that it is assumed that the results of $\tilde{B} Q_U \tilde{B}^t$ and $\tilde{B}^t Q_U \tilde{D}^t$ are precomputed and stored in the array. The calculation of the state vector $\hat{\hat{X}}_i$ can be inserted during any idle period of the array processor in Fig. 12.

If the Kalman gain matrix is preevaluated off-line on a host computer, the state update part can be implemented using a simpler structure described in the next section.

E. Systolic Array Implementation of State Update Part

In our earlier work [17] several parallel and pipeline algorithms and structures for 2-D recursive filters were proposed. The skew pipeline structure presented in [17] uses a GAPP type array for 2-D scalar and block filtering operations. In this section, a skew pipeline architecture is presented for both parallel and sequential block Kalman filters which uses a 1-D systolic array in which each PE is assigned to the computation of an element within the block estimate. The 1-D skew pipeline structure is quite similar to the ring systolic array in [18] and [19].

In the development of this architecture, it is assumed that the steady-state Kalman gain matrices are evaluated off-line and used throughout the updating process. In this case (16a) and (16d) can be rewritten as

$$\hat{\hat{X}}_i = \tilde{A}\hat{\hat{X}}_{i-1} + K_s[Y_i - \tilde{H}\hat{\hat{X}}_{i-1}] \tag{45a}$$

or

$$\hat{\hat{X}}_i = D\hat{\hat{X}}_{i-1} + K_s Y_i \tag{45b}$$

where

$$D \triangleq \tilde{A} - K_s \tilde{H} \tag{46}$$

and K_s is the steady-state Kalman gain matrix for the parallel block Kalman filter. Similarly, (27) and (29) can be rewritten as

$$\hat{\hat{X}}_i = \tilde{A}\hat{\hat{X}}_{i-1} + \tilde{K}_s \tilde{W}_i^{-1}(Y_i - \tilde{H}\hat{\hat{X}}_{i-1}) \tag{47a}$$

or

$$\hat{\hat{X}}_i = F\hat{\hat{X}}_{i-1} + GY_i \tag{47b}$$

where

$$F \triangleq \tilde{A} - \tilde{K}_s \tilde{W}_i^{-1} \tilde{H}$$
$$G \triangleq \tilde{K}_s \tilde{W}_i^{-1} \tag{48}$$

and $\tilde{K}_s$ is the steady-state Kalman gain matrix for the sequential block Kalman filter. The processes in (45b) and (47b) can now be implemented using only two matrix-vector multiplications.

A single PE is assigned to each element of the block estimate $\hat{\hat{X}}_i$. Fig. 13(a) shows the configuration of the 1-D ring systolic array processor, for $M = 4$, where M is the block or the array size. The structures of all PE's are identical. Fig. 13(b) shows the internal structure of PE_m, i.e., the mth PE where $\alpha_{m,n}$ and $\beta_{m,n}$, $m, n \in [0, M-1]$, denote the (m, n)th elements of the matrices D (or F) and K_s (or G) in (45b) (or (47b)), respectively. The elements of these matrices in PE_m are stored in the following order: $\{(m, m), (m, (m+M-1) \bmod M), (m, (m+M-2) \bmod M), \cdots, (m, (m+1) \bmod M)\}$ $m \in [0, M-1]$.

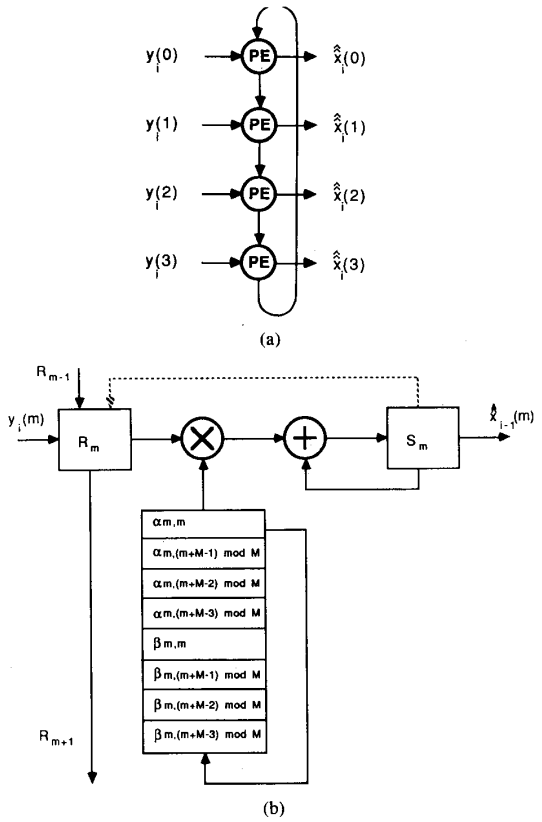


Fig. 13. (a) One-dimensional systolic array processor. (b) Structure of a single PE of the 1-D systolic array ($M = 4$).

During each iteration, these coefficients are shifted in a round-robin fashion. R_m and S_m are two registers used to store the variables. It is assumed that before stage i, $\hat{x}_{i-1}(m)$, i.e., the mth element of vector $\hat{X}_{i-1}$ has been generated and held in register S_m. The operations at each iteration start by shifting $\hat{x}_{i-1}(m)$ from register S_m to R_m and clearing S_m. Then in the next $2M$ steps the first and second terms in (45b) or (47b) are generated consecutively in registers S_m's. At the completion of the last step, the elements of the updated estimate vector $\hat{X}_i$ can be obtained from registers S_0 to S_{M-1}. The process continues until all the block estimates are computed. The average sampling period for each element of a block is

$$T_s = 2(T_{\text{MUL}} + T_{\text{ADD}}) \tag{49}$$

where T_{MUL} and T_{ADD} are defined before. The delay time (latency) from Y_i arrival to $\hat{X}_i$ generation is

$$T_D = M(T_{\text{MUL}} + T_{\text{ADD}}). \tag{50}$$

VII. Conclusion

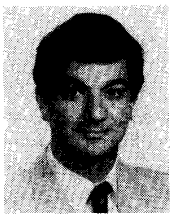
Two different sets of block Kalman filter equations are derived which correspond to the parallel and sequential modes of implementation. It is shown that the parallel implementation provides fixed-lag smoothed estimates. However, in this scheme the estimation cannot start until the entire observation block is available. The sequential block implementation does not have this particular problem. The initial estimates are more accurate in this latter scheme since they are produced based upon the updated estimates of the samples in the previous block and also those in the current block. This scheme provides purely filtered estimates both at the block and scalar levels and also does not introduce any delay (lag) in the response. A comprehensive study among these two structures and the standard Kalman filter is made which reveals the effectiveness of the sequential block Kalman filter for signal estimation applications. The implementation of the block Kalman filter algorithm using systolic array is also considered. An iterative procedure for efficient computation of Kalman gain and error covariance matrices is proposed which is free of any matrix inversion operation. A general systolic array architecture which utilizes a GAPP-type processor is designed to implement these operations. The state update part when the steady-state Kalman gain is evaluated off-line can be implemented using a simple ring systolic array structure.

References

[1] C. S. Burrus, "Block realization of digital filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 230–235, Oct. 1972.
[2] R. A. Meyer and C. S. Burrus, "Design and implementation of multirate digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 53–58, Feb. 1976.
[3] G. A. Clark, S. K. Mitra, and S. R. Parker, "Block implementation of adaptive digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 584–592, June 1981.
[4] C. W. Barnes and S. Shinnaka, "Block-shift invariance and block implementation of discrete-time filters," *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 667–672, Aug. 1980.
[5] B. D. O. Anderson, S. K. Mitra, and N. S. Ramesh, "Block Kalman filtering," in *Proc. Euro. Conf. Circuit Theory Design* (The Hague, The Netherlands), 1981, pp. 723–731.
[6] A. K. Jain and J. Jasiulek, "Fast Fourier transform algorithms for linear estimation, smoothing, and Riccati equations," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1435–1446, Dec. 1983.
[7] H. H. Lu, E. A. Lee, and D. G. Messerschmitt, "Fast recursive filtering with multiple slow processing elements," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 1119–1129, Nov. 1985.
[8] K. K. Parhi and D. G. Messerschmitt, "Look-ahead computation: Improving iteration bound in linear recursions," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Dallas, TX), 1987.
[9] K. K. Parhi and D. G. Messerschmitt, "Block digital filtering via incremental block-state structure," in *Proc. IEEE Int. Symp. Circuits Syst.* (Philadelphia, PA), May 1987.
[10] K. Khorasani and M. R. Azimi-Sadjadi, "Feedback control of two-time scale block implemented discrete time systems," *IEEE Trans. Automat. Contr.*, vol. AC-32, pp. 69–73, Jan. 1987.
[11] S. L. Marple, *Digital Spectral Analysis with Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1987.
[12] A. P. Sage and J. L. Melsa, *Estimation Theory with Applications to Communications and Control.* New York: McGraw-Hill, 1971.
[13] H. G. Yeh, "Systolic implementation on Kalman filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp. 1514–1517, Sept. 1988.
[14] J. H. Graham and T. F. Kadela, "Parallel algorithms and architectures for optimal state estimation," *IEEE Trans. Comput.*, vol. C-34, pp. 1061–1068, Nov. 1985.
[15] S. Y. Kung, *VLSI Array Processors.* Englewood Cliffs, NJ: Prentice-Hall, 1988.
[16] *Electron. Design*, Oct. 31, 1984.
[17] T. Lu, "Implementations, algorithms, and architectures for 2-D recursive filtering," M.S. thesis, Colorado State Univ., July 1988.
[18] S. Y. Kung and J. N. Hwang, "Systolic designs for state-space models: Kalman filtering and artificial neural networks," in *Proc. 26th IEEE Conf. Decision Contr.* (Los Angeles, CA), Dec. 1987, pp. 1461–1467.
[19] W. G. Bliss and L. L. Scharf, "Algorithms and architectures for

dynamic programming on Markov chains," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, pp. 900-912, June 1989.

[20] R. K. Mehra, "On-line identification of linear dynamic systems with applications to Kalman filtering," *IEEE Trans. Automat. Contr.*, vol. AC-16, pp. 12-21, Feb. 1971.



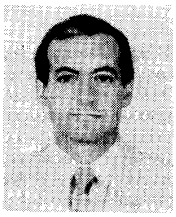
Mahmood R. Azimi-Sadjadi (S'81-M'81-SM'89) was born in Tehran, Iran, in 1952. He received the B.Sc. degree from the University of Tehran, Iran, in 1977, the M.Sc. and Ph.D. degrees from Imperial College, University of London, England, in 1978 and 1982, respectively, all in electrical engineering.

He served as an Assistant Professor in the Department of Electrical and Computer Engineering, University of Michigan-Dearborn. Since July 1986 he has been with the Department of Electrical Engineering, Colorado State University, where he is now an Associate Professor. His areas of interest are digital signal/image processing, multidimensional system theory and analysis, adaptive filtering, system identification, and neural networks. He is a coauthor of the book, *Digital Filtering in One and Two Dimensions* (Plenum Press, 1989).

Dr. Azimi-Sadjadi is the recipient of the DOW Chemical Outstanding Young Faculty Award of the American Society for Engineering Education.

Tongxin Lu received the B.S. degree from Beijing University of Aeronautics and Astronautics, in 1978, and the M.S. degree from Colorado State University, in 1988. Since 1989, he has been working toward the Ph.D. degree in electrical engineering.

From 1978 to 1986, he served as an Electronics Engineer at Beijing Institute of Astronautic Control Engineering. His current research interests include VLSI signal processing and optoelectronic implementation of numerical computing and image processing.



Eduardo Mario Nebot (S'79-M'81) received the E.E. degree from the Universidad Nacional del Sur, Bahia Blanca, Argentina, in 1980, and the master's and Ph.D. degrees in electrical engineering from Colorado State University in 1986 and 1988, respectively.

From 1982 to 1985 he worked with the PLAPIQUI Control Group, Argentina, in the area of process control. During the period 1985-1988 he was engaged in graduates studies at Colorado State University where he was a Teaching Assistant in the Department of Electrical Engineering. From 1988 to 1989 he worked as a Research Associate at CSU in the area of robotics. He is now an Assistant Professor at Universidad Nacional del Sur, Argentina, and is also working at PLAPIQUI. His research interests include statistical signal and image processing and automatic control.