Semi-Annual Report

to

NASA-Ames Research Center
Grant Award No. NAG 2-304

NASA Technical Officer: Karl Anderson

for

A Survey of the State of the Art and Focused Research in Range Systems - Task II

Principal Investigator
Kung Yao

Electrical Engineering Department
University of California
Los Angeles, California 90024

June 1986

(NASA-CR-176988) A SURVEY OF THE STATE OF
THE ART AND FOCUSED RESEARCH IN RANGE
SYSTEMS, TASK 2 Semiannual Report
(California Univ.) 22 p CSCL 09C
N86-30069 THRU N86-30072
G3/33 43497

UNCLASSIFIED
In the period from January to June 1986, the following research activities and publications performed under NASA-Ames Research Contract Grant Award Number NAG-2-304 are enclosed:


ABSTRACT

A systematic approach is presented for designing systolic arrays and their equivalent configurations for certain general classes of recursively formulated algorithms. A new method is also introduced to reduce the input bandwidth and storage requirements of the systolic arrays through the study of dependence among the input data. Many well known systolic arrays can be redetermined and also many new systolic arrays can be discovered by this approach.

I. INTRODUCTION

A systolic array is a network of processors that rhythmically process and pass data among themselves. It provides pipelining, parallelism, and simple adjacent neighbor cell interconnection structure so that it is suitable for VLSI implementation. While most of the earlier systolic array algorithms were discovered heuristically [1-3], there has been various work on systematic approaches to the design of systolic array algorithms [4-6]. In this paper, we shall present a systematic approach for designing systolic arrays and especially focus on their equivalent configurations for certain general classes of recursively formulated algorithms. In order to reduce the input bandwidth and storage requirements of the systolic arrays, the dependence among the input data is also investigated in details. It is shown that many well known systolic arrays can be redetermined and also many new systolic arrays can be discovered by this systematic approach. For simplicity of illustration, we mainly consider the linear systolic array in this paper. The same idea can also be generalized to the two dimensional mesh-connected systolic arrays.

II. IMPLEMENTATION OF RECURSIVELY FORMULATED ALGORITHMS

Consider two simple but important ways of data flow pattern in a linear systolic array as shown in Figure 1 and 2. In these two figures, P.., Q.., and b.. are three given input data sequences and R.. is to be the output data sequence, where O.. is the output sequence. For the systolic array shown in Figure 1, Q.. and R.. are stored in the j.. processor, where R.. will be updated while P.. is moving to the right and b.. is moving down. For the systolic array shown in Figure 2, P.. is stored in the i.. processor and R.. will be updated as it is moving to the right with Q.., while b.. is moving down. All of the data movements are synchronized. The R.. will successively have the required output data after m steps. For convenience, according to the R..'s behavior of these two systolic arrays, they are respectively named as R-stay and R-move linear systolic arrays. There is great similarity between these two systolic arrays. It can be shown that a large class of interesting problems in the real world can be implemented by these two types of linear systolic arrays. Besides, various different but equivalent configurations of linear systolic arrays can also be derived from them.

Procedure 1 : Given any problem which can be formulated so that it has P.., Q.., and b.., as three input data sequences and R.. as the output data sequence, where O.. and O.., if R.. can be generated through the following recurrence equation

\[ R_{j[i+1]} = f(P_{i}, Q_{j}, b_{i,j}; R_{j[i]},) \]  

where R.. contains some initial value, f is any function of four variables P.., Q.., b.., and R.., and R.. is the required output. This problem can be implemented by the R-stay linear systolic array of n processors and the R-move linear systolic array of m processors.

The complexity and the configuration of the systolic array depend on the complexity of the function f and the generation procedure of b.. Some regularity and dependence among b.. greatly simplify the whole system.

III. MAPPING INTO FAN-IN TYPE LINEAR SYSTOLIC ARRAY

Note that for the two linear systolic arrays shown in Figure 1 and 2, the input bandwidth and storage requirements are large in comparison to the number of processors in the array, which may be either infeasible or inefficient for many applications of interests. This is mainly because the dependence among the b..'s is not efficiently utilized so that each processor needs its own external input connection due to the existence of all the b..'s. It is expected that under certain circumstances not all of these external input connections are required. In this paper, we are also very interested in the issue of reducing the input bandwidth and storage requirements by showing under what conditions these external input connections can be removed so that only the very first processor is allowed to have such a connection, i.e., the input sequences can only be fanned-in through the systolic array. It is shown that the existence of certain patterns of dependence among the b..'s allows themselves to be fanned-in generated by slightly modifying the operations involved in each processor without losing the property of adjacent neighbor interconnection structure. These conditions are shown in the following two procedures.

Procedure 2 : For the R-stay linear systolic array, if b.. can be determined through the following dependence equation

\[ b_{ij} = T(b_{i-1,j}; b_{i,j-1}; u_{i}; v_{j}) \]  

where T.. is the transformation function of two variables b.., b.., and u.., and v...
where $u_i$ is a variable which depends only on $i$, $v_j$ is a variable which depends only on $j$, and $T$ is a function of four variables, then $b_i$ can be generated by the fan-in scheme systolic array as shown in Figure 3 rather than being broadcast as shown in Figure 1. Also note that $b_{i-1}$ as well as $v_j$, which depends only on $j$, can be preloaded in the $(j-1)$ processor, and $b_{i-1}$ as well as $u_i$, which depends only on $i$ can be used as a fanned-in input sequence.

Note that for the R-stay linear systolic array shown in Figure 1, if $b_{i-1}$ is the current input to the $i$ processor, then $b_{i-1}$ is the previous input to the $j$ processor and $b_{i-1}$ is the previous input to the $(j-1)$ processor. It is understandable that in order to avoid the violation of the adjacent neighbor interconnection structure, $b_{i-1}$ can only depend on $b_{i-1}$ and $b_{i-1}$ as well as the data that can be preloaded and the data that can be fanned in, which is what Procedure 2 is about. In general, the systolic array shown in Figure 3 has two sets of input data. One of them consists of three fanned-in data sequences, $P$, $q_j$, and $b_{i-1}$, which depend only on $j$, and the other set consists of three preloaded data sequences, $P$, $u_i$, and $b_{i-1}$, which depend only on the $i$ index, where $u_i$, $v_i$, $b_i$, and $b_{i-1}$ are used to generate all the $b_{i}$'s. For each processor, three registers are required, namely $Q$, $V$, and $B$, where registers $Q$ and $V$ are used to store the preloaded data $P$, and $V$, respectively. Initially register $Q$ is loaded as $b_{i-1}$ and register $B$ is set to be $b_{i-1}$, both of which will be updated as the systolic array start operation.

The previous three procedures provide a rather systematic approach to design the systolic array architecture for the implementation of a given problem. At first, by checking the existence of the recurrence relationship as shown in equation (1), we are able to know if there exist any systolic arrays as shown in Figure 1 and 2. Next, by checking the dependence among the $b_{i}$'s as shown in equations (2) and (3), we are able to know the existence of the fan-in type systolic arrays as shown in Figures 3 and 4 so that only small input bandwidth and storage are required. The key issue is how to search for the recurrence function $f$ and the dependence function $T$. It is expected that there may exist several different forms of functions due to different possible approaches to formulate a given problem. Various forms of these functions simply create many different but equivalent configurations of systolic arrays. Also note that in the previous discussion, $P$, $Q$, $b$, $u$, and $v$ are somewhat treated as single variables, however it is clear that they can be set of variables and the same results still hold. This approach can be applied to design systolic arrays for many interesting problems in the real world. Various new configurations of systolic arrays can be derived. In the next section, we shall illustrate this design approach by considering the DFT algorithm.

IV. SYSTOLIC ARRAY ARCHITECTURE FOR DISCRETE FOURIER TRANSFORM

Given $n$ discrete data $a_j$ in the time domain, where $0 \leq j \leq n-1$, and $n$ discrete frequencies $W_j = \left(n e^{j \pi \frac{2}{n}}\right)$ in the frequency domain, where $0 \leq j \leq n-1$, the discrete Fourier transform (DFT) is to compute

$$y_j = a_{n-1}W_j + a_{n-2}W_j^2 + \ldots + a_1W_j + a_0.$$ 

Let

$$f(P, Q, b; R) = (R \times b) + P.$$ 

By induction, it can be shown that by letting

$$y_{j+1} = (y_j \times W_j) + a_{n-1} \quad (4)$$

and $y_0 = a_0$, then $y_{j+1}$ is the required output. The existence of $a$ recurrence function $f$ and the satisfaction of the recurrence relationship guarantee that there exists systolic array for the implementation of discrete Fourier transform as shown in Figure 5 and 6.

It can be seen from Figure 5 and 6 that the $b_{i}$'s are not totally independent. Note that $P_0 = a_{n-1}$ and $b_0 = W_j$. In order to see if $b_{i}$ can be fanned-in generated, let us examine the data
dependence among the b_i's. Many different forms of dependence function \( f \) exist. For example,

\[
b_{ij} = T(b_{i-1,j}; b_{i,j-1}; u_i; v_j)
\]

(5)

where \( v_j = W_j \). The pair of systolic arrays based on equations (4) and (5) are shown in Figure 7 and 8. The systolic array shown in Figure 8 is the well-known systolic DFT [2], whose discovery appears to be heuristic rather than in a systematic manner as from our approach. For another example of \( T \) function, note that

\[
b_{ij} = \begin{cases} W_j & \text{if } j > 0 \\ b_{i-1,j} & \text{if } j = 0 \\ b_{i,j-1} & \text{if } j < 0 \end{cases}
\]

(6)

where \( u_i = W_i \) and \( b_{-1,j} = W_i^{-1} \), which can be either used as ‘fanned-in’ sequences of the R-move linear systolic array or preloaded in the \( i \) processor of the R-setup linear systolic array. The pair of systolic arrays based on equations (4) and (6) are shown in Figure 9 and 10.

Another interesting issue is that the type of function \( f \) used in this example does not belong to the class of general matrix vector multiplication. This confirm the fact that the class of problems covered in the Procedure 1 really contains not only the class of general matrix vector multiplication. As well known, there are two different ways to consider the discrete Fourier transform. One shows that the DFT is a special case of the evaluation of a polynomial and the other shows that the DFT is a special case of general matrix vector multiplication. The first way was just considered in this example. Let us see what can be obtained by following the second way. Let

\[ f(P, Q, b; R) = R \times (P \times b). \]

By induction, it can be shown that by letting

\[ y_j[i+1] = y_j[i] + (a_i \times W_j), \]

and \( y_j[0] = 0 \), then \( y_j[i] \) is the required output. The existence of a new recurrence function \( f \) and the satisfaction of the recurrence relationship guarantee that there exists systolic arrays for the implementation of DFT as shown in Figure 11 and 12.

From Figure 11 and 12 it can also be seen that the b_i's are not totally independent. Note that \( P = b_{i,j} \) and \( b_{-1,j} = W_i^{-1} \). Let us examine the data dependence among the b_i's. Note that

\[
b_{ij} = W_i' = W_j = W_{j-1}' = W_{j-1} \]

i.e.,

\[
b_{ij} = T(b_{i-1,j}; b_{i,j-1}; u_i; v_j)
\]

(8)

where \( u_i = W_i \) and \( b_{-1,j} = W_i^{-1} \), which can be either used as ‘fanned-in’ sequences of the R-setup linear systolic array or preloaded in the \( i \) processor of the R-move linear systolic array. The pair of systolic arrays based on equations (7) and (8) are shown in Figure 13 and 14. Also note that

\[
b_{ij} = \begin{cases} W_i & \text{if } j > 0 \\ b_{j-1,i} & \text{if } j = 0 \\ b_{j,i-1} & \text{if } j < 0 \end{cases}
\]

\[ i.e.,
\]

\[
b_{ij} = T(b_{i-1,j}; b_{i,j-1}; u_i; v_j)
\]

(9)

where \( v_j = W_j \) and \( b_{-1,i} = W_j^{-1} \), which can be either preloaded in the \( j \) processor of the R-setup linear systolic array or used as ‘fanned-in’ sequences of the R-move linear systolic array. The pair of systolic arrays based on equations (7) and (9) are shown in Figure 15 and 16.

This DFT example shows that under certain circumstances it is possible to formulate a given problem in several different ways to implement with various different but equivalent configurations of systolic arrays.

V. CONCLUDING REMARKS

A systematic approach is presented for designing systolic arrays and deriving their equivalent configurations for certain general classes of recursively formulated algorithms. This approach can be considered as a two-stage design procedure. In the first stage, the existence of recursiveness is investigated. If it exists, according to the same formulation the input data are classified into three parts, two of them, \( P \) and \( Q \), depend only on one index, and another one of them, namely \( b_i \), depends on both index \( i \) and \( j \), so that the systolic arrays shown in Figure 1 and 2 apply. However, for certain applications, it is either infeasible or inefficient to store all of the \( b_i \)'s. In the second stage, the dependence among the \( b_i \)'s is then investigated to see if it can be used to fan-in generate the \( b_i \)'s through the data sequence that can either be preloaded or fanned in. For a given problem, various formulations of the recursive property and the dependence among the \( b_i \)'s are possible, which simply lead to many different but equivalent configurations of systolic arrays.

So far we mainly deal with the linear systolic arrays. However, the same technique can be easily generalized to the two-dimensional mesh-connected systolic arrays, since the mesh-connected systolic arrays can be simply treated as the concatenation of many linear systolic arrays.

VI. ACKNOWLEDGEMENT

This work was partially supported by the NASA Ames research contract NAG-2-304.

VII. REFERENCES


Figure 1: The R-stay linear systolic array.

\[
\begin{align*}
\text{Pin} & \rightarrow Q, R \rightarrow \text{Pout} \\
\text{Pout} & \leftarrow \text{Pin} \\
R & \leftarrow f(\text{Pin}, Q, b_{i,n}; R)
\end{align*}
\]

Figure 2: The R-move linear systolic array.

\[
\begin{align*}
\text{Vin} & \rightarrow U, P \\
\text{Bin} & \rightarrow B, P \\
\text{Pout} & \leftarrow U, \text{vin} \\
R & \leftarrow f(\text{Pin}, Q, b_{i,n}; R)
\end{align*}
\]

Figure 3: The fan-in scheme of R-stay linear systolic array. Note that the register B in the jth processor is initially loaded with \(b_{i,j}\).

\[
\begin{align*}
\text{Win} & \rightarrow y \\
\text{ain} & \rightarrow a_{out} \\
\text{aout} & \leftarrow \text{ain} \\
y & \leftarrow (y \times \text{Win}) + \text{ain}
\end{align*}
\]

Figure 5: R-stay linear systolic array of discrete Fourier transform based on equation (4).

\[
\begin{align*}
\text{ain} & \rightarrow W, p \\
\text{ain} & \rightarrow a_{out} \\
y & \leftarrow (y \times W) + \text{ain}
\end{align*}
\]

Figure 7: R-stay linear systolic array of discrete Fourier transform based on equations (4) and (5).

\[
\begin{align*}
\text{Win} & \rightarrow U \\
\text{ain} & \rightarrow a_{out} \\
y & \leftarrow (y \times \text{Win}) + \text{ain}
\end{align*}
\]

Figure 6: R-move linear systolic array of discrete Fourier transform based on equation (4).

\[
\begin{align*}
\text{Win} & \rightarrow W, p \\
\text{ain} & \rightarrow a_{out} \\
y & \leftarrow (y \times \text{Win}) + \text{ain}
\end{align*}
\]

Figure 8: R-move linear systolic array of discrete Fourier transform based on equations (4) and (5).
Figure 9: R-stay linear systolic array of discrete Fourier transform based on equations (4) and (6).

Figure 10: R-move linear systolic array of discrete Fourier transform based on equations (4) and (6). Note that register Up is preloaded with W1 and register B is initially loaded with W1⁻¹.

Figure 11: R-stay linear systolic array of discrete Fourier transform based on equation (7).

Figure 12: R-move linear systolic array of discrete Fourier transform based on equation (7).

Figure 13: R-stay linear systolic array of discrete Fourier transform based on equation (7) and (8).

Figure 14: R-move linear systolic array of discrete Fourier transform based on equations (7) and (8). Note that in the ith processor, register Up is preloaded with Wi and register B is initially loaded with Wi⁻¹.

Figure 15: R-stay linear systolic array of discrete Fourier transform based on equations (7) and (9). Note that in the jth processor, register Vp is preloaded with Wj and register B is initially loaded with Wj⁻¹.

Figure 16: R-move linear systolic array of discrete Fourier transform based on equations (7) and (9).
ABSTRACT

In this paper we consider the problem of modeling and equalization of a nonlinear satellite channel. The channel is assumed to be bandlimited and exhibits both amplitude and phase nonlinearities. A discrete time satellite link is modeled under both uplink and downlink white Gaussian noise. Under conditions of practical interest, a simple and computationally efficient design technique for the minimum mean square error linear equalizer is presented. The bit error probability and some numerical results for a BPSK system demonstrate that the proposed equalization technique outperforms standard linear receiver structures.

I. INTRODUCTION

The problem of nonlinear channel modeling and equalization is of analytical and practical interest. An important example of a nonlinear channel is a digital satellite communication link, which uses a Traveling Wave Tube (TWT) amplifier operating in a near saturation region. The TWT exhibits nonlinear distortion both in amplitude (AM-AM conversion) and phase (AM-PM conversion). In addition, at high transmission rates the channel’s finite bandwidth causes a form of distortion known as Intersymbol Interference (ISI).

In this paper, we will examine the problem of modeling and equalizing this type of nonlinear satellite communication link. The observed data are corrupted by additive white noise, uncorrelated with the input data.

A number of other researchers have studied this problem. Fredricsson [1] considered a QPSK system and specified an optimum linear receiver filter using a Mean Square Error (MSE) criterion. The channel nonlinearity in [1] was handled via successive number of linearizations. Mesva et al. [2-3] analyzed the BPSK system. In [2] a maximum likelihood receiver was considered, while in [3] a simpler linear receiver, based on the MSE criterion, was presented. In both [2] and [3], the nonlinearity of the TWT is expressed in terms of Bessel function integrals. The MSE criterion was also applied by Biglieri et al. [4] in their derivation of an optimum linear receiver. In [1], [3], and [4], the authors work in the frequency domain, and the solution is given in terms of integral equations that usually have to be solved using numerical techniques.

In [5], Ekanayake and Taylor presented a suboptimum maximum-likelihood type decision feedback receiver. However, because of the analytical complexity of their solution, they approximate the TWT with a hard limiter. A different modeling approach was taken by Benedetto et al. [6]. First, they identify the whole channel using a Volterra Series expansion [7]. Then they suggest a nonlinear equalizer, based again on the MSE criterion. Although at the output of a nonlinear equalizer the MSE is smaller than the MSE at the output of a linear equalizer, it is not completely clear if there is a significant improvement in the probability of error performance of the system to justify the complexity of the nonlinear receiver.

In this paper we consider the problem of modeling and equalization of a nonlinear satellite channel. While the methods considered here are applicable to various in-phase and quadri-phase modulation systems, for simplicity and lack of space we will illustrate this approach by using only BPSK examples. More generalized results will be presented elsewhere. There are two major differences between our design as compared to the above reviewed approaches. First, we use a very simple model for the input-output relationship of the TWT amplifier, proposed first by Saleh [8]. Second, by working in the discrete time domain we avoid the complex integral equations of the other approaches. In addition, a fast and simple iterative algorithm [9] permits the easy computation of the autocorrelation coefficients of the output of the nonlinear system. Thus, we are able to obtain a new simple and computationally efficient linear equalization technique under the MSE criterion. Based on the same modeling approach, a zero forcing type of linear equalizer was also presented in [10].

In Section 2, a simplified model for a typical satellite link is presented and the corresponding BPSK discrete model is derived. The optimum MSE equalizer is presented in Section 3. In Section 4, the probability of error performance of the receiver is derived. Finally, in Section 5 some numerical examples, and comparisons with standard linear receivers are presented.

II. CHANNEL MODELING

Consider the simplified model of a digital satellite communication channel as shown in Figure 1. We will examine each one of the different subsystems composing this model. This study will enable us to derive an equivalent discrete model. By working in discrete time we will avoid the analytical problems arising with continuous signals. Our analysis is similar to that of Ekanayake and Taylor [5].

The source output is a random sequence \( U(n) \) of equally probable uncorrelated symbols. Thus, in a BPSK system, \( U(n) = 1, -1 \) at \( n = 0, T, 2T, \ldots \) where \( P[U(n) = 1] = P[U(n) = -1] = 0.5 \), \( E[U(n)U(n-k)] = 0 \) for \( k \neq 0 \), and \( T^{-1} \) is the signaling rate.

Let \( p(t) \) denote a pulse shaping function. Often it can be a rectangular function of unit amplitude over a time period of length \( T \). In any case, the output of the modulator can be expressed in the form
where $w_n$ is the curia frequency.

We shall assume that the transmission filter is the one which determines the channel bandwidth. This filter is also responsible for the creation of ISI. Let $G(t) = 2g(t)\cos\omega_n t$ be the impulse response of this filter, where $g(t)$ is the impulse response of a corresponding low pass filter. Then the output of this filter can be expressed as

$$s_n(t) = \sum_{k=-\infty}^{\infty} U(k)h[(t-kT)\cos\omega_n t],$$

(2)

where $h(t) = g(t)\delta(t)$. The purpose of our analysis is the design of a receiver structure for the estimation of the transmitted source symbol during the $n$th signaling interval $nT \leq t \leq (n+1)T$. Thus during the $n$th signaling interval (2) can be rewritten as

$$s_n(t) = U(n)h[(t-nT)\cos\omega_n t] + \sum_{k \neq n} U(k)h[(t-kT)\cos\omega_n t],$$

(3)

$nT \leq t \leq (n+1)T$.

The first term in (3) represents the transmitted symbol we want to estimate, and the second term represents the ISI due to the filter.

On the uplink channel, $s_u(t)$ is corrupted by additive white Gaussian noise. Thus, using the narrow band model for the noise, the input to the TWT can be expressed as

$$s_u(t) = s_u(t) + n_{aw}(t)\cos\omega_n t - n_{aw}(t)\sin\omega_n t,$$

(4)

$n_{aw}(t)$ and $n_{aw}(t)$ represent the in-phase and quadrature components of the uplink noise, each with zero mean and variance $\sigma_n^2$. From (3) and (4)

$$s_u(t) = r_u(t)\cos(\omega_n t + \lambda(t)),$$

(5)

where

$$r_u(t) = \left[\sqrt{(r(t) - n_{aw}(t))^2 + n_{aw}^2(t)}\right]^*$$

(6)

and

$$\lambda(t) = \tan^{-1}\frac{n_{aw}(t)}{r(t)} - n_{aw}(t).$$

(7)

The TWT is a nonlinear memoryless amplifier. It exhibits nonlinear distortion in both the amplitude and phase. Using a quadrature model, the output $s_x(t)$ of the TWT can be expressed in the form

$$s_x(t) = P(r_u(t))\cos(\omega_n t + \lambda(t)) - Q[r_u(t)]\sin(\omega_n t + \lambda(t)).$$

(8)

From Saleh [8] an expression of $P(r)$ and $Q(r)$ is given by

$$P(r) = \alpha_0 \frac{r}{1 + \beta r^2},$$

(10a)

and

$$Q(r) = \alpha_1 \frac{r^3}{1 + \beta r^2}.$$ 

(10b)

The coefficients of (10) are obtained by a least-square error curve fitting procedure of the specific TWT characteristics, which are originally specified by the manufacturer. In Figure 2 the $P(r)$ and $Q(r)$ functions of (10) are plotted for $\alpha_0 = 2.0922$, $\beta_0 = 1.2466$, $\alpha_1 = 5.529$ and $\beta_1 = 2.7088$ [8]. All input and output voltages were normalized.

Because of the downlink additive white noise $n_d(t)$, the received waveform $s'_d(t)$ can be expressed as

$$s'_d(t) = s_u(t) + n_{aw}(t)\cos(\omega_n t - n_{aw}(t)\sin(\omega_n t).$$

(9)

$nT \leq t \leq (n+1)T$.

The signal $s'_d(t)$ of (11) is now coherently demodulated by a carrier $2\cos\omega_n t$. We assume that the bandwidth of the receiving filters is wide enough so that no additional ISI distorts the signal. The output $y(t)$ of the demodulator is sampled every $T$ seconds to produce in the $n$th signaling interval the in-phase sample

$$y(n) = y(t_0) = y(t_0) + P[r_u(t_0)]\cos(\omega_n t_0) + Q[r_u(t_0)]\sin(\omega_n t_0) + n_{aw}(t_0).$$

(12)

$t_0$ is an appropriately chosen sampling instant within the interval $nT \leq t \leq (n+1)T$.

Under the assumption of high available power at the earth stations, the effects of the uplink noise can be considered negligible. Thus we can assume that $\lambda(t)=0$. Then $y(n)$ of (12) becomes

$$y(n) = P[r(t_0)] + n_{aw}(t_0).$$

(13)

From (7) and (13) an equivalent discrete-time model for the communication channel of Figure 1 can be represented as in Figure 3. Now, with $U(n) = \{1,-1\}$, the basic relationships are

$$r(n) = A^{-\frac{2g}{\sigma_n^2}} h_t U(n-k),$$

(14)

$$P(n) = P[r(n)] = \frac{\alpha r(n)}{1 + \beta r^2(n)},$$

(15)

$$y(n) = P(n) + w(n).$$

(16)

where $\alpha$ and $\beta$ are specified constants that depend on the specific type of the TWT, $w(n)$ is white Gaussian noise of zero mean and variance $\sigma_n^2$, and uncorrelated with the input data.
The values of N1 and N2 represent the memory of the transmitting filter. The gain A depends on the specific operating point of the TWT.

### III. THE MEAN-SQUARE ERROR EQUALIZER

Let the receiver output z(n) be expressed as the output of a Tapped-Delay Line (TDL) filter in the form of

\[ z(n) = \sum_{k=-\infty}^{\infty} c_k y(n-k) \, . \]  

(17)

where \( y(n) = P(n) + w(n) \).

In the theory of the Mean-Squares criterion, the tap weight coefficients \( \{c_k\} \) of the equalizer are adjusted to minimize the mean square error

\[ e^2 = E[|y(n) - \sum_{k=1}^{\infty} c_k y(n-k)|^2] \, . \]  

(18)

Minimization of (18) with respect to the \( \{c_k\} \) coefficients, yields the linear system of \( M = M1 + M2 + 1 \) equations

\[ \sum_{k=1}^{\infty} c_k R_y(j-k) = R_m(j), \quad j = M1, \ldots, M2 \, . \]  

(19)

where \( R_y(j-k) = E[y(n)y(n-k)] \) and \( R_m(j) = E[U(n)y(n-k)] \) for all values of k.

From the uncorrelatedness of the input data and the noise, \( R_m(j) = R_m(k) \), for all values of k. Also, since the output P(n) of the nonlinearity, and the noise w(n) are independent

\[ R_y(j-k) = a^2 \delta_{jk} \, , \]  

(20)

where \( \sigma^2 \) is the variance of the noise. Thus in order to solve (19) it is necessary to evaluate first the necessary \( R_y(.) \) and \( R_m(.) \) coefficients. While in the case of a linear channel the calculation for the \( R_y(.) \) coefficients is straightforward, in the nonlinear case the evaluation may present some numerical difficulties.

### Computation of the Autocorrelation coefficients

The sequence \( P(n) \) at the output of the nonlinearity can be considered as the output of a finite state sequential machine. Since the nonlinearity has no memory, from (14) the state sequence can be given by

\[ s(n) = [U(n-N1), \ldots, U(n), U(n-1), \ldots, U(n-N2)] \, . \]  

(21)

\([U(n)] \) is an i.i.d sequence, thus \( s(n) \) is itself a stationary Markov chain [9].

Let us denote by \( \Pi \) the transition probability matrix of the Markov chain \( s(n) \). A brute force evaluation of \( R_y(.) \) involves multiplications of square matrices of dimension \( 2^{M1+M2+1} \) [9-10], which would be computational impractical unless special consideration is given to the special properties of \( \Pi \). In [9] a particularly effective and simple algorithm for the evaluation of autocorrelation coefficients of a nonlinear system was presented. The algorithm, as applied to our specific problem is given below.

Algorithm for the computation of \( R_y(.) \)

1. Let \( N = N1 + N2 + 1 \), and store in vector \( \beta_0 \) (of dimension \( 2^N \)) the values at the output of the nonlinearity for each state \( s(j) \) of (21), for \( j = 1,2,\ldots,2^N \).

2. Compute the vector \( \alpha_0 \) (of dimension \( 2^N \)), where the \( j \) th component is given by

\[ \alpha_0(j) = \beta_0(j)2^N \, , \quad j = 1,2,\ldots,2^N \, . \]

3. For \( k = 0,1,\ldots,N-1 \), do the following computations

a) \[ R_y(k) = \sum_{j=0}^{2^N-1} \alpha_0(j) \beta_0(j) \, . \]

b) Store in the first \( 2^{N-1} \) positions of \( \alpha_0 \), the vector \( \alpha_{k+1} \), computed by

\[ \alpha_{k+1}(j) = \alpha_0(j) + \alpha_0(j + 2^N - 1) \, , \quad j = 1,2,\ldots,2^{N-1} \, . \]

c) Store in the first \( 2^{N-1} \) positions of the \( \beta_0 \) vector, the vector \( \beta_{k+1} \), where

\[ \beta_{k+1}(j) = \frac{\beta_0(2j-1) + \beta_0(2j)}{2} \, , \quad j = 1,2,\ldots,2^{N-1} \, . \]

4. \( R_y(k) = 0 \), for \( k \geq N \).

The above algorithm is easy to implement and requires only two vectors of size \( 2^N \) as basic computation storage.

### Computation of Cross-correlations

Since for each state \( s(j) \) of (21) the value of \( P(k) \) has already been computed for the evaluation of the \( R_y(.) \) coefficients, a brute force technique can be easily applied for the evaluation of the cross-correlation terms. Thus from [10],

\[ R_{y_m}(k) = (1/2^{M1}) \sum_{j=0}^{2^N-1} P[s(j)] \, . \, \quad N1 \leq k \leq N2 \, . \]  

(22)

where the summation in (22) is done over all those states where \( U(n-k) = 1 \).

In summary, the design procedure for the optimum linear MSE equalizer is given as follows. First, compute the \( 2^N \) possible values of \( P(n) \) at the output of the nonlinearity. Then use the algorithm to compute the \( R_y(.) \) coefficients and (22) to
compute the $R_{mk}(\cdot)$ coefficients. Finally, solve the linear system in (19). The solution of (19) yields the tap-weight coefficients of the MSE receiver.

**IV. EVALUATION OF BIT-PROBABILITY OF ERROR**

Unfortunately, there is no simple relationship between the residual mean square error of the MSE receiver and the bit error probability [11]. For moderate channel and equalizer memories, a brute force method that yields an exact result could be applied. Denote by $D_k$ one of the $2^{N_k-2}$ possible realizations at the input of the receiver, with $U(n)=1$, and by $c$ the $M \times 1$ vector of the filter coefficients. Then from (16) and (17), the receiver output due to a specific input $U(n)$ sequence is given by

$$z_k = D_k c + W_k,$$

where $W$ is a $M=M1+M2+1$ row vector of noise samples.

Let $\{\eta(\cdot)\}$ be a white noise sequence of zero mean and variance $\sigma^2$. Let $\eta = W_k$. Then $E[\eta]=0$ and $\sigma^2 = \sigma^2 \sum_{i=1}^{M} c_i^2$. Then with $U(n)=1$ and for a threshold of zero, the conditional error probability

$$P_r(i) = P_{r}[D_k c + \eta < 0 | U(i)]$$

is fixed, and

$$P_r(i) = Q(D_k c / \sigma),$$

where

$$Q(x) = \sqrt{2 \pi} e^{-x^2 / 2}.$$

Then the average error probability $P_r$ is given by

$$P_r = (U/L) \sum_{i=1}^{L} P_r(i), \quad L = 2^{N-2}.$$

In our numerical example the SNR is defined as

$$SNR = 10 \log_{10} \left( P_0 / 2 \sigma^2 \right).$$

If the exact error probability of (27) proves too cumbersome and too time consuming to evaluate because of the large number of terms, one can resort to a number of different approximate methods that yield tight upper and lower bounds of $P_r$ [11].

**V. NUMERICAL EXAMPLE**

The purpose of this section is to illustrate the application of our results in the design of a linear optimal receiver, and to compare its performance with other receivers for a digital satellite link.

In our model of the linear part of the satellite link, we assume that the ISI is introduced by a 3-pole Butterworth filter. The two sided bandwidth $B$ of the filter is the same as the minimum Nyquist rate (i.e., $BT=1$). The number of samples considered for the ISI is determined by those ISI samples whose magnitude are at least greater than 0.01 times the main sample. In our example, a channel memory $(N1+N2)$ of 3 ISI terms was considered adequate.

The characteristics of the TWT for this study are the same as those in Figure 2. Thus in the evaluation of $P(r(n))$ in (15), the parameters of the TWT are $\alpha = 2.0922$ and $\beta = 1.2466$. As mentioned before, those values were taken from Saleh [18]. Figure 5 and represent a specific satellite TWT. The gain factor $A$, of (14) was determined so that with no ISI the TWT would operate at the 2 dB input backoff point. Because of the low ISI introduced by the transmission filter, a 4-tap $(M1+M2+1=4)$ TDL linear receiver was considered to be adequate. Thus, $L = 2^{N-2.5} = 64$.

Now we compare our optimum linearly equalized MSE receiver with the conventional linear receivers. Using the brute force technique described in Section 4, the bit error probability for the various receivers was evaluated and plotted in Figures 4 and 5, for values of SNR as defined in (28).

Figure 4 exhibits the $P_r$ performance of the designed MSE filter, and the $P_r$ performance of two 3-pole Butterworth receiving filters. One receiving filter (with $BT=1$) is identical to the transmitting filter, while the other one has $BT=0.75$. In Figure 5, the performance of the M.S receiver is compared with that of two 4-pole Butterworth receiving filters. One has $BT=0.75$ and the other one has $BT=1$. A numerical search procedure for Butterworth filters with different $BT$ products, showed that an increase in $BT$ does not necessarily correspond to an improved $P_r$ performance. In fact, filters with $BT=2$ are only marginally better than filters with $BT=1$.
VI. SUMMARY AND CONCLUSIONS

In this paper we considered the problem of modeling and equalization of a digital satellite nonlinear and bandlimited channel. Starting from a typical satellite link, we developed the corresponding BPSK discrete-time model, and solved for the optimum linear MSE receiver. A simple and computationally efficient algorithm was derived for the evaluation of the equalizer coefficients, based on the memoryless nonlinearity of the system. Numerical examples for a typical satellite link demonstrated that the optimum linear MSE receiver outperforms the conventional linear type receiving filters. In general, our modeling and equalization techniques provide a simple and computationally efficient alternative to existing approaches.

REFERENCES


1. INTRODUCTION

Least-squares (LS) estimation is a basic operation in many signal processing problems. Given \( y = Ax + v \), where \( A \) is a \( mxn \) coefficient matrix, \( y \) is a \( mx1 \) observation vector, and \( v \) is a \( mx1 \) zero mean white noise vector, a simple least-squares solution is finding \( x \) which minimizes \( \| Ax - y \| \). It is well known that for an ill-conditioned matrix \( A \), solving least-squares problems by orthogonal triangular (QR) decomposition and back substitution has robust numerical properties under finite word length effect since 2-norm is preserved. Many fast algorithms have been proposed and applied to systolic arrays. Gentleman-Kung (1981) first presented the triangular systolic array for a basic Givens reduction. McWhirter (1983) used this array structure to find the least-squares estimation errors. Then by geometric approach, several different systolic array realizations of the recursive least-squares estimation algorithms of Lee et al (1981) were derived by Kalson-Yao (1985). We consider basic QR decomposition algorithms and find that under one-row time updating situation, the Householder transformation degenerates to a simple Givens reduction. Next, we derive an improved least-squares estimation algorithm by considering a modified version of fast Givens reduction. From this approach, the basic relationship between Givens reduction and Modified-Gram-Schmidt transformation can easily be understood. We also can see this improved algorithm has simpler computational and inter-cell connection complexities while compared with other known least-squares algorithms and is more realistic for systolic array implementation.

Minimum variance estimation (popularized by Kalman (1960)) is the generalized form of a least-squares problem, where the state vector \( x \) is characterized by the state equation \( x_{k+1} = F x_k + w \), the system noise \( w \) and the observation noise \( v \) are colored. The original algorithm presented by Kalman can have poor numerical property. Some algorithms for improving numerical properties, such as square-root covariance and square-root information methods have been studied. Now, we find that after the whitening processing, this minimum variance estimation can be formulated as the modified square-root information filter and be solved by the simple least-squares processing. This new approach contains advantages in both numerical accuracy as well as computational efficiency as compared to the original Kalman algorithm. Since all these processings can be implemented by systolic arrays, high throughput rate computation for Kalman filtering problems become feasible.
2. SIMPLE LEAST-SQUARES ESTIMATION

Given the equation $b = Ax + v$, it is well known that we can solve the least-squares solution $\hat{x}$ by normal equation. However, this approach not only requires the computation of a matrix inverse but also doubles the condition number when we form $A'A$. Although using singular value decomposition for least-squares solution can improve numerical properties, the computational complexity involved in SVD is not low. Besides, fast algorithm for SVD is still underdevelopment. Lattice structure for least-squares solution was proposed and studied by Lee et al (1981). This approach was shown to have stable numerical property and regular hardware structure. However, this method required shifting property of the coefficient matrix and can not apply to all general cases. QR decomposition is another solution to obtain $\hat{x}$, since 2-norm is preserved by multiplying an orthogonal matrix $Q$, then by letting $QA = R$ be a upper triangular matrix, the $\hat{x}$ can be obtained by using back substitution for the equation $Rx = b$. This approach has robust numerical properties since the 2-norm is fixed, the rounding error caused by finite word length effect will not grow. Basically, there are three ways for performing QR decomposition, namely, Householder transformation, Givens reduction, and Modified-Gram-Schmidt orthogonalization. It can be shown that under one row time updating situation (as in the systolic array implementation), the Householder transformation matrix will degenerate to a simple Givens reduction case.

Systolic array implementation for QR decompositions in least-squares estimation was first explored by Gentleman-Kung and followed by McWhirter and Kalson-Yao. By using a triangular systolic array, it was shown that the estimation error for the last observation can be solved at every clock period. The systolic array structure for least-squares estimation is shown in Figure 2.1. To achieve fully pipelined operation, the input rows are skewed and propagated like wavefronts in the diagonal direction. There are only two basic processing units, boundary cell and internal cell, are required by this systolic array. Communication between different processing units are all local. The properties of regularity and local communication are consistent with the philosophy of VLSI implementation. Summary of input/output formats and operation functions for two kinds of processing units are shown in Table 1 and Table 2 respectively.

| Table 1. Input/Output format of systolic array algorithms |
|-------------------|---|---|---|---|---|---|---|---|
|                  | $B_1$ | $B_2$ | $B_0$ | $B_2$ | $I_1$ | $I_2$ | $I_0_1$ | $I_0_2$ |
| Givens           | $\sigma$ | $x$ | $\sigma'$ | $d/d'$, $x/d'$ | $d/d'$, $x/d'$ | $b$ | $d/d'$, $x/d'$ | $b'$ |
| F-Givens         | $\sigma$ | $x$ | $\sigma'$ | $d/d'$, $x/\sigma/d'$ | $d/d'$, $x/\sigma/d'$ | $x$ | $x$ | $b'$ |
| M-G-S(I)         | $\sigma$ | $x$, $e$ | $\sigma'$ | $x/d$, $x/(1-\sigma)$ | $x/d$, $x/(1-\sigma)$ | $b$, $e$ | $x/d$, $x/(1-\sigma)$ | $b'$, $e'$ |
| M-G-S(II)        | $\sigma$ | $x$ | $\sigma'$ | $x/d'$, $x/(1-\sigma)$ | $x/d'$, $x/(1-\sigma)$ | $b$ | $x/d'$, $x/(1-\sigma)$ | $b'$ |
The above symbols are for notations only, their physical meaning may change for different algorithms.

Table 2. Operational functions of processing units

<table>
<thead>
<tr>
<th></th>
<th>Boundary cells</th>
<th>Internal cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>Givens</td>
<td>(d'=(d^2+x^2)^{1/2})</td>
<td>(b'=(d/d')*b - (x/d')*k)</td>
</tr>
<tr>
<td></td>
<td>(B_0 \leftarrow d/d', x/d')</td>
<td>(k'=(x/d')*b + (d/d')*k)</td>
</tr>
<tr>
<td></td>
<td>(\sigma \leftarrow (d/d')*\sigma)</td>
<td>(\sigma \leftarrow (d/d')*\sigma)</td>
</tr>
<tr>
<td>F-Givens</td>
<td>(d'=d + (\sigma*x)*x)</td>
<td>(b'=b - x*k)</td>
</tr>
<tr>
<td></td>
<td>(B_0 \leftarrow (\sigma*x)/d', d/d')</td>
<td>(k'=(d/d')<em>k + (\sigma</em>x/d')*b)</td>
</tr>
<tr>
<td></td>
<td>(\sigma \leftarrow \sigma*(d/d'))</td>
<td>(\sigma \leftarrow \sigma*(d/d'))</td>
</tr>
<tr>
<td>M-G-S(I)</td>
<td>(d=e)</td>
<td>(k'=\sigma + b*(x/(1-c)))</td>
</tr>
<tr>
<td></td>
<td>(B_0 \leftarrow x/d, x/(1-c))</td>
<td>(b'=b - k'<em>b</em>(x/d))</td>
</tr>
<tr>
<td></td>
<td>(\sigma \leftarrow (x/d)*x)</td>
<td>(e'=e - k'^2/d)</td>
</tr>
<tr>
<td>M-G-S(II)</td>
<td>(d'=d + (x/(1-c))*x)</td>
<td>(k'=\sigma + b*(x/(1-c)))</td>
</tr>
<tr>
<td></td>
<td>(B_0 \leftarrow x/d', x/(1-c))</td>
<td>(b'=b - k'<em>b</em>(x/d'))</td>
</tr>
<tr>
<td></td>
<td>(\sigma \leftarrow \sigma + (x/d')*x)</td>
<td>(\sigma \leftarrow \sigma + (x/d')*x)</td>
</tr>
</tbody>
</table>

From systolic array point of view, the difference between algorithms proposed by McWhirter and Kalson-Yao lies in the basic computations in two kinds of processing units. Since these algorithms were derived from two different approaches, specifically Givens reduction and Modified-Gram-Schmidt orthogonalization, the basic relationship for these two QR decomposition methods under one row time updating can be compared as follows. First, we derived the modified expression for the fast-Givens reduction as given by

\[
Q = \begin{bmatrix}
(1/\sqrt{d})d, (1/\sqrt{d})d_k, \ldots, (1/\sqrt{d})d_k

\sqrt{\sigma}x, \quad \sqrt{\sigma}b_2', \quad \ldots \quad \sqrt{\sigma}b_k
\end{bmatrix}
\begin{bmatrix}
(1/\sqrt{d'})d', (1/\sqrt{d'})d'k_2', \ldots, (1/\sqrt{d'})d'k_k'

0, \quad \sqrt{\sigma'}b_2', \quad \ldots \quad \sqrt{\sigma'}b_k'
\end{bmatrix}
\]

the updating equation for this modified-fast-Givens algorithm becomes,

Boundary cell: \(d'=d + x^2/(1-c)\) \( (1/\sigma')=(1/\sigma) + x^2/d\)
Internal cell: \(b'=b - (x/d)*dk\) \( d'k'=dk + b*x/(1-c)\) \[1\]

By comparing the computational complexity between the fast Givens algorithm by Gentleman (1973) and that in [1], we can see [1] has one multi
plication less than the original algorithm. And since we do not have interest on the real rotated elements like \((1/\sqrt{d})dk\), we do not have the risk of dividing by a very small \(d\). The numerical properties of the modified algorithm is then expected to comparable to the numerical properties of the original one. By equation (11), the basic duality associations between Givens reduction and Modified-Gram-Schmidt orthogonalization is summarized in Table 3, which allows us to derive different algorithms for least-squares estimation from different approaches with efficiency.

Table 3. Duality association for M-G-S and Fast-Givens reduction.

<table>
<thead>
<tr>
<th>M-G-S(II)</th>
<th>(k_{mgs})</th>
<th>(c_{mgs})</th>
<th>(x_{mgs})</th>
<th>(b_{mgs})</th>
<th>(d_{mgs})</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-Givens</td>
<td>(d_{fg})</td>
<td>(k_{fg})</td>
<td>(1-o_{fg})</td>
<td>(c_{fg}*x_{fg})</td>
<td>(c_{fg}*b_{fg})</td>
</tr>
</tbody>
</table>

With systolic array implementation, comparison of computational complexity for algorithms discussed above can be made by comparing the number of operations required in each processing unit. When the dimension of the coefficient matrix becomes large, wavefront array processing of Kung (1983) becomes more appropriate for the control scheme. In this case, the speed of this "wavefront" will be decided by the slowest processing unit along each wavefront. In modified fast Givens algorithm, equations for boundary cell are non-recursive and can be done in parallel if we can double the computational capability of each boundary cell. In this case, the wavefront speed and then the throughput rate can be doubled. The systolic array we discussed above will generate estimation error at each clock period. While the estimated vector \(\hat{x}\) is not shown explicitly, \(\hat{x}\) can be solved by back substitution which can be done by just appending a \(n\times n\) identity matrix after the coefficient matrix \(A\).

3. MINIMUM VARIANCE ESTIMATIONS AND KALMAN FILTERING

Often the signal vector \(x\) is a random process and can be modeled as a first order recursive equation. In this case, a first order recursive estimation (or Kalman filtering) problem can be stated as follows,

\[
\begin{align*}
x_{k+1} &= Fx_k + w_k, \\
y_k &= Cx_k + v_k,
\end{align*}
\]

where \(F\) and \(C\) are time-varying coefficient matrices with dimension \(n\times n\) and \(m\times n\) respectively. \(w_k\) is a \(n\times 1\) and \(v_k\) is a \(m\times 1\) zero mean noise vectors with known covariance matrices \(W_k\) and \(V_k\) respectively. It is assumed that noises \(w\) and \(v\) are uncorrelated and \(E[w_kw_k'] = E[v_kv_k'] = 0\) for all \(i \neq j\). Under the minimum variance criterion, we want to find \(x_k^\dagger\) for all \(k\), such that \(E[\|x_k - x_k^\dagger\|^2]\) is minimized. Kalman showed that \(x_k^\dagger\) can be obtained by the recursive algorithm given as
The information matrix is defined as the inverse of the error covariance matrix $P$. Besides [3], it is shown that instead of propagating the error covariance matrix, the Kalman filtering problem can be solved by propagating the information matrix during the iterations. Both covariance and information filters are recursive since the current updating depends only on results from previous stage. The choice between covariance filter and information filter depends on the values of $n$ and $m$. When $n>m$, which is usually the case, the original Kalman filtering is chosen to avoid the inverse of the $n \times n$ matrix. However, Kalman algorithm is known for its poor numerical properties, especially for non-observable coefficient matrices. The original Kalman filter needs an approximate $O(n^3)$ multiplication time for each iteration. If $m>1$, computation of a matrix inversion is inevitable. Since all equations are sequential in manner, if real time computation is required for a Kalman filtering problem, some modifications must be done to ensure the capability for parallel computation. Among many possible modified algorithms, square-root filtering have been proved to have computational efficiency and robust numerical properties under finite word length effect (Kaminski 1971). The main advantage of the square root filter is that we can handle the covariance matrix by its square root form which has condition number smaller than the original one. Therefore, for ill-conditioned problems, when we used the square root filter with a single precision machine, we can expect the same numerical result as if we have used the original algorithm on a double precision machine. Updating processings for both square root covariance filter and square root information filter can be expressed in matrix forms and handled by the QR decomposition method which is capable of systolic array implementation. However, only square-root information filter allows us to update the estimated state vector as well as the information matrix by using the same transformation matrix $Q$. When both updated covariance matrix and state vector are important to us, we find square-root information filter is a better solution for the systolic array implementation. The square-root information filter requires computation of the inverse of the coefficient matrix $F$, which will cause bad numerical properties for $F$ being near singular. One version of the square root information matrix method for Kalman filtering was considered by Paige and Saunders (1977). It is shown that by using whitening processing through Cholesky decomposition, the Kalman filtering can be represented as a simple least-squares problem. This approach does not require the computation of the inverse of the matrix $F$ and is more suitable for systolic array implementation.

The whitening processing can be briefly described as below. Assume $W=L_W W'$ and $V=L_V V'$ are the Cholesky decomposition of covariance matrices $W$ and $V$. With $W^{-1}=L_W W'$ and $V^{-1}=L_V V'$, it can be proved that $L_W = W_W$ and $L_V = V_V$. $\tilde{w} = L_W w'$ and $\tilde{\nu} = L_V \nu'$ are whitened noises with identity covariance matrices.

Denote $F=L_F F'$, $C=L_C C'$, and $y_k = L_y y_k$. We can express the whitened system equations in the matrix-vector form as
\[
\begin{bmatrix}
\hat{y}_1 \\
\hat{y}_0 \\
\vdots \\
\hat{y}_k
\end{bmatrix} = \begin{bmatrix}
-L_w^c & 0 & \cdots & 0 \\
F & -L_w^c & \cdots & 0 \\
0 & \ddots & \ddots & \ddots \\
0 & \cdots & \cdots & \cdots & 0
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_k
\end{bmatrix} + \begin{bmatrix}
\hat{w}_1 \\
\hat{w}_2 \\
\vdots \\
\hat{w}_k
\end{bmatrix}
\]

Since the noise vector in [4] has zero mean and identity covariance matrix, we can get \( x_k = [\hat{x}_1, \ldots, \hat{x}_k] \) by solving [4] as a LS problem. After applying QR decomposition to [4] at time \( k \), we have

\[
\begin{bmatrix}
R_1 & R_{12} & \cdots & 0 \\
R_2 & R_{22} & \cdots & \cdots \\
\vdots & \vdots & \ddots & \ddots \\
0 & \cdots & \cdots & R_{k-1,k}
\end{bmatrix}
\begin{bmatrix}
x_{1,k} \\
x_{2,k} \\
\vdots \\
x_{k-1,k}
\end{bmatrix}
= \begin{bmatrix}
\hat{y}_1 \\
\hat{y}_2 \\
\vdots \\
\hat{y}_{k-1} \\
\hat{y}_k
\end{bmatrix}
\]

We can see that \( R_i, i=1,2,\ldots,k \), in [5] are all upper triangular matrices, and \( \hat{x}_k \), the optimum estimated vector at time \( k \), depends only on the last line, i.e., \( R_k \hat{x}_k = \hat{y}_k \). Furthermore, at \( T=k+1 \), the updating equation depends only on the last row of [5] only. That is, the QR decomposition at \( T=k+1 \) only depends on a \((2n+m) \times (2n+1)\) matrix as in [6]. When the QR decomposition of [6] is completed, we have \( R_{k+1} \) (upper triangular) and \( \hat{y}_{k+1} \) ready for iteration of next stage.

\[
Q \begin{bmatrix}
\bar{R}_k & 0 & \bar{y}_k \\
F & -L_w^c & 0 \\
0 & C_w^c & \bar{y}_k
\end{bmatrix} = \begin{bmatrix}
R_k & R_{k,k+1} & \bar{y}_k \\
0 & R_{k+1} & \bar{y}_{k+1}
\end{bmatrix}
\]

where * is the term used to compute the residue.

The upper triangular matrix \( \bar{R}_k \) can be shown to be the square-root of the inverse of the error covariance matrix \( P_k = \mathbb{E}[(x_k-x_k^*)(x_k-x_k^*)'] \). That is, this algorithm, which propagates the square-root information matrix for next iteration, is actually a modified square-root information filtering.

4. SYSTOLIC ARRAY IMPLEMENTATIONS FOR KALMAN FILTERING

From last section, we can see that the basic operations for square root Kalman filtering can be described in two parts. The first one, whitening processing includes operations such as Cholesky decomposition, inverse of triangular matrix, and matrix multiplication. Secondly, the QR decomposition is applied. Obviously, these two parts can be operated in parallel. That is, we can start the whitening processing for the \((k+1)\)st iteration as well as the QR decomposition for the \( k \)-th iteration at the same time in a pipelined manner.

The original square-root information filter involves the computation of the inverse of the coefficient matrix \( F \) which not only increases the computational complexity but also causes bad numerical properties when coefficient matrix \( F \) is singular or near singular. This shortcoming can be
recovered by choosing the modified square root information filtering in [4]. As shown from [4]-[6], formulation of the modified square-root information filter involves only multiplication between coefficient matrices and the inverse of the square root noise covariance matrices. For noise with positive definite covariance, square root covariance matrix always exists.

4.1 Whitening Processing

The whitening processing is done by multiplying the coefficient matrix with a whitening operator \( L' \) where \((LL')^{-1}\) is the given covariance matrix of the additive noise. Since a covariance matrix is a positive definite symmetric matrix, the square root matrix can be obtained by the Cholesky decomposition. A triangular systolic array for Cholesky decomposition is designed for this purpose with outputs skewed to match the input format of the QR systolic array.

The inversion of a upper triangular matrix is simple after we built the basic systolic array for QR decomposition. The idea for the inversion of a upper triangular matrix is the same as solving the back substitution.

With \( UU^{-1} = I \), let \( U^{-1} = [u_1, u_2, \ldots, u_n] \), with \( u_i \), being a nx1 column vector. A matrix inversion can be divided into \( n \) sets of linear equations, each having the form of \( u_i = e_i \), \( i=1,2,\ldots,n \), where \( e_i \) is a nx1 column vector with \( i \)th element equals to 1, and all others being 0, and can be solved by a systolic array.

4.2 QR Decomposition for Kalman Filtering

Equation [6] suggests that \( x_k \) can be solved as a least-squares solution by a 2nx2n QR systolic array. However, serious delay will be caused by the fact that \( R_k \) and \( R_{k+1} \) are not in-place computations. That is, we have trouble to move the newly formed \( R \) from the upper-right corner to the lower-left corner in our triangular array for the next iteration. That is, the computation at stage \( k+1 \) can not start until the last element of \( R_k \) is completed. In this "waiting" period, most of processing units are idle and the pipeline is empty. It will cause delay for at least 2n clock periods.

This disadvantage can be overcome by in-place computations for \( R_k \) and \( R_{k+1} \). This can be done by partitioning the original matrix into two strips, and perform the partitioned QR decomposition by the systolic array structure proposed in Figure 2. In this approach, a n*2n QR systolic array as well as a rotation array which consists of n*(n+1) internal cells are used. Once elements of \( R_k \) are formed, it is ready to be used for computations at stage \( k+1 \). Here we need only to pass transformed elements generated by the first strip to the rectangular rotation array for the pre-processing of the second strip. This input format is shown in Figure 3. Since all these can be done in fully pipelined manner and in-place computations are obtained, complicated inter-cell connection and control scheme can both be avoided. To obtain the estimated value \( x_k \), we can just append an identity matrix \( I \) after the second strip, and we get result every 3n+m clock periods.
5. CONCLUSION

In this paper, we first survey existing algorithms for least-squares estimations by systolic arrays. Basic comparisons are made based on computation and inter-cell connection complexities of elementary units. Finally, by choosing the square-root information filtering algorithm, we showed a simple way to solve the Kalman filtering as a least-squares problem that can be processed by systolic arrays. Systolic array for Cholesky decomposition is also proposed for whitening processing. By manipulating the data properly, the Kalman filtering can be processed under fully pipelined manner. There is no special constraint on our system equations and standard time-varying coefficient matrices and non-stationary colored noises are assumed in our model. Most of the processing units we need for this square root information filter do not involve square-root computations. The only exception is the computations for the Cholesky decomposition. However, for pipelined operation between whitening processing and QR decomposition, the later certainly involved more computational work than the former. Since there is only n square-root computations required in each iteration as compared with the operations required for QR decomposition, Cholesky decomposition will not become the bottleneck for this algorithm. For many real life problems where we can assume noises are stationary, then covariance matrices W and V are fixed during our operation. In this case, inversed square-root covariance matrices can be obtained by pre-processing and our Kalman filtering can be solved as a simple least-square problem. Since all operations can be performed by the designed systolic array processing, which have the input/output formats matched to each other, the entire hardware design can be viewed as a pipelined structure. The estimated vector can be obtained with the $O(n)$ in time while compared with the $O(n^2)$ for the original Kalman filter. Finally, since this is a square root matrix operation, good numerical property can also be obtained.

This work is partially supported by NASA/Ames Research Contract NAG-2-304

REFERENCES

Bierman G 1982 Square-Root Information Filtering and Smoothing for Precision Orbit Determination, Math. Programming Study 18
Bjorck A 1967 Solving Linear Least Squares Problems by Gram-Schmidt Orthogonalization, BIT 7 1-21
Duncan D B and Horn S D 1972 Linear Dynamic Recursive Estimation from the Viewpoint of Regression Analysis, J. ASA 67 815-821
Gentleman W M 1975 Error Analysis of QR decompositions by Givens Transformations, Linear Algebra and Its Applications, 10 189-197
Kalman R E 1960 A New Approach to Linear Filtering and Prediction Problems, J. Basic Engineering, 82 35-45
Kung S Y 1983 VLSI Design for Massively Parallel Signal Processors, Microprocessors and Microsystems
Lawson C and Hanson R 1974 Solving Least Squares Problems, Prentice-Hall, NJ
Lee D T, Morf M and Friedlander B 1981 Recursive least square ladder estimation algorithms, IEEE Tran. ASSP, 3 627-641
Ling F and Proakis J 1984 A Recursive Modified Gram-Schmidt Algorithm with Application to Least Squares Estimation and Adaptive Filtering, Int. Sym. on Circuit and System
Mead C and Conway L 1980 Introduction to VLSI systems, Addison-Wesley, Mass.

Figure 1: Systolic array for least-squares estimation.
Figure 2: QR systolic array for Kalman filtering

Figure 3: Input format for systolic array Kalman filtering