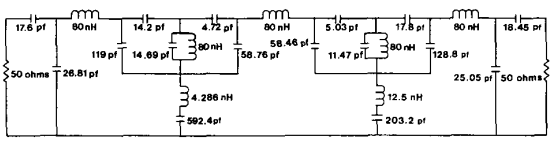

Fig. 11. Final filter schematic.

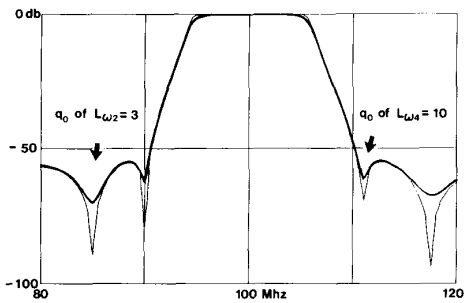

Fig. 12. Analysis of Fig. 11.

The multiplier for sub-branch W_4' is

$$K_{n4} = 2K_{L4}/(K_{L4}K_{r4}+1) = 0.1074$$

where $K_{L4} = K_1K_2K_3$ and $K_{r4} = K_6$.

The final filter schematic is shown in Fig. 11. Fig. 12 shows an analysis of the above example. Two plots are superimposed illustrating the affect of lower Q_L in the sub-branches. The lighter curve has infinite Q's. The darker curve has a $Q_L = 3(f_0/Bw)$ for the sub-branch associated with transmission zeros further removed from the passband, (LW_2) and $Q_L = 10(f_0/Bw)$ for the sub-branch associated with the closer in zeros (LW_4). Notice from Fig. 11 the inductor values further removed from the optimal value, i.e., 80 nH, have lower Q_L sensitivity.

Conclusion

This paper has shown a method for the realization of narrow bandwidth elliptic filters. Low-Q narrow-band transformations in concert with Norton transformers and Norton transformers referenced not to a ground plane, but to a virtual ground, (at band center) are combined to reduce critical inductor ratios to a minimum. The additional capacitors required by the Norton transformations are easily justified because the inductors whose unloaded Q is the determinant of performance can be assigned an optimal value by the filter designer. A ninth-order 13-percent bandwidth filter at 1.1 GHz and filters with 2-percent bandwidths have been successfully constructed using this technique.

Acknowledgment

The author would like to thank Philip Geffe for encouragement, editing, and especially for writing the Fortran program that effectively automates the above-described procedure.

References

[1] E. L. Norton, U.S. Patent 1 681 554 and U.S. Patent 1 708 950, 1928.
[2] *Reference Data for Radio Engineers*, fourth ed., IT&T, pp. 120 and 121, 1956.
[3] *Handbook of Filter Synthesis*, Zverev: Wiley, 1967, pp. 524, and 525.
[4] *Handbook of Filter Design*, Saal, Nachrichtentechnik Telecommunication, (Tables), Berlin, Germany, 1979.

2-D Systolic Arrays for Realization of 2-D Convolution

HON-KEUNG KWAN AND THOMAS SAMUEL OKULLO-OBALLA

***Abstract*—In this paper, an image is regarded as a 2-D array of pixels and is processed by a 2-D array architecture. The image can be acquired in the usual manner by a raster scan method which produces a 1-D array of pixels at real-time video rates. The two 2-D systolic arrays for a 2-D convolver presented have an architecture which accepts this 1-D array of pixels and processes them in a 2-D array of simple processors. This high degree of parallelism is achieved through matrix–vector formulations of 2-D convolution.**

One array has a serial input, a serial output, and uses a minimum number of multipliers; while the other array has parallel inputs, parallel outputs, and is suitable for high-speed processing using slow processing elements. Both arrays are modular with nearest neighbor communications and are suitable for VLSI implementation.

In addition, the algorithm for 2-D convolution presented here explicitly takes into account the boundary conditions. This feature allows a large image to be partitioned so that each partition may be processed by independent 2-D convolvers. It is then possible to process only a specified section of the image or carry out high-speed parallel processing using as many 2-D convolvers as are available.

I. Introduction

In image processing, convolution is often performed using a relatively short weighting sequence. Conceptually, an image is regarded as a 2-D array and so should be processed by a 2-D array architecture. However, in practice, it is acquired by a raster scan method which produces a 1-D array of pixels at real-time video rates. An earlier method [1] has made use of these facts in designing a 1-D array for 2-D convolution which performs convolution with a 3×3 weighting window. This idea was further advanced to general multidimensional convolution in a later paper [2].

These arrays however, are essentially stacks of 1-D systolic arrays whose outputs are suitably combined to give the required 2-D results. A more recent paper [3] presents semisystolic arrays which are based on matrix formulations of convolution and have a 2-D architecture. This formulation allows a high degree of parallelism in the computations but is limited in speed by broadcasting. Even in low-level vision applications, high-speed is necessary as image sizes are up to 1 M of pixels and it is often desirable to process at video rates. In another paper [4], a simple 2-D systolic array for 1-D convolution was presented. However, it seems more appropriate and economical to use a 2-D array to implement 2-D convolution.

This paper shows that the latter 2-D array [4] can be used to realize 2-D convolution. The two 2-D systolic arrays for a 2-D convolver presented have an architecture which accepts a 1-D

Manuscript received February 4, 1988; revised February 2, 1989. T. S. Okullo-Oballa was supported by the Hong Kong Government Education Department in the form of a Commonwealth Scholarship. This paper was recommended by Associate Editor H. Gharavi.

H. K. Kwan was with the University of Hong Kong, Pokfulam Road, Hong Kong. He is now with the Department of Electrical Engineering, University of Windsor, Windsor, Ont., Canada N9B 3P4.

T. S. Okullo-Oballa was with the University of Hong Kong, Pokfulam Road, Hong Kong. He is now with the Department of Electronics and Communication Engineering, Singapore Polytechnic, Singapore 0513.

IEEE Log Number 8932876.

0098-4094/90/0200-0267$01.00 ©1990 IEEE

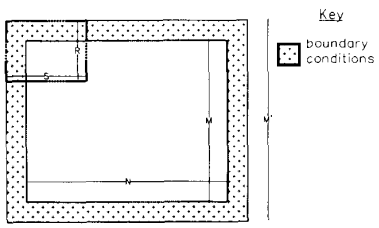

Fig. 1. The 2-D convolution field.

array of pixels and processes them in a 2-D array of simple processors. In addition, by explicitly taking into account the boundary conditions, a large image can be partitioned so that each partition may be processed by independent 2-D convolvers. It is then possible to process only a specified section of the image or carry out high-speed parallel processing using as many 2-D convolvers as are available.

In summary, from a matrix–vector representation of 2-D convolution, pure systolic arrays have been derived. Section II gives a definition of 2-D convolution which takes into account the boundary conditions and presents the decomposition of the 2-D convolution problem into a set of N simultaneous equations. In Section III, this set of simultaneous equations is realized by an array characterized by a serial input, a serial output, and uses a minimum number of multipliers. An alternative array in Section IV realizes the same set of equations using an array characterized by parallel inputs and parallel outputs which is suitable for high-speed processing. The same algebraic example has been used to illustrate both methods and it should be noted that they use the same basic cell.

II. A Definition of 2-D Convolution

A definition of 2-D convolution which includes the boundary conditions can be stated as: for an $R \times S$ array of weights, $w_{h,l}$ for $h=1,2,\cdots,R$ and $l=1,2,\cdots,S$ and an $M \times N$ input array, $x'_{i,j}$ for $i=1,2,\cdots,M$ and $j=1,2,\cdots,N$. The convolution of these two arrays can be defined as

$$y(i,j) = \sum_{h=1}^{R} \sum_{l=1}^{S} w_{h,l} x_{i',j'} \tag{1}$$

where $i' = i+h-1$ and $j' = j+l-1$.

The variable $x_{i',j'}$, is the augmented input array for which $i'=1,2,\cdots,M'$ and $j'=1,2,\cdots,N'$ where $M' = M+R-1$ and $N' = N+S-1$. Since the boundary conditions are determined by the weight array, it is necessary to consider the augmented input array, $x_{i',j'}$, instead of $x'_{i,j}$ in (1). We could assume all values of $x_{i',j'}$ to be zero outside the boundary defined by $M \times N$. However, this need not be so when x' defines a subset of an image as shown in the following subsection. These boundary conditions and the relation between x' and x are illustrated in Fig. 1.

Expansion of (1) in the variable l gives

$$y(i,j) = \sum_{h=1}^{R} y_h(i,j) \tag{2}$$

where

$$y_h(i,j) = w_{h,1} x_{i+h-1,j} + w_{h,2} x_{i+h-1,j+l} + \cdots + w_{h,S} x_{i+h-1,j+S-1}. \tag{3}$$

For any h, expansion of (3) in the variable j gives a system of N equations in the variable i. The expansion for any h can be expressed as

$$\begin{aligned} y_h(i,1) &= w_{h,1} x_{i+h-1,1} + w_{h,2} x_{i+h-1,2} + \cdots + w_{h,S} x_{i+h-1,S} \\ y_h(i,2) &= w_{h,1} x_{i+h-1,2} + w_{h,2} x_{i+h-1,3} + \cdots + w_{h,S} x_{i+h-1,S+1} \\ &\cdots \\ &\cdots \\ y_h(i,N) &= w_{h,1} x_{i+h-1,N} + w_{h,2} x_{i+h-1,N+1} \\ &\quad + \cdots + w_{h,S} x_{i+h-1,N'}. \end{aligned} \tag{4}$$

The problem of implementing (1) has thus been decomposed to that of implementing the R sub-systems ($h=1,2,\cdots,R$) each consisting of N equations given by (4), which can be done by the two methods described in Sections III and IV.

A. Explicit Relation Between x' and x

Let $I(A,B)$ denote a 2-D array, of ordered pairs of integers, of size $A \times B$. The following sets of 2-D images can be defined:

$$\mathscr{E} = \{ x'_{i,j} : (i,j) \in I(P,Q) \}$$

$$\mathscr{F} = \{ x'_{i,j} : (i,j) \in I(M,N) \} \text{ and } \mathscr{F} \subset \mathscr{E}$$

$$\mathscr{G} = \{ x_{i',j'} : (i',j') \in I(M',N') \} \text{ and } \mathscr{G} \subset \mathscr{E}$$

and

$$\mathscr{H} = \{ w_{h,l} : (h,l) \in I(R,S) \}.$$

Now let $x'_{p,q}$ be the first pixel in $\mathscr{F}$ and its associated weight in the $R \times S$ weight array by $w_{h',l'}$. A new set, $\mathscr{G}'$, whose corresponding first pixel is $x'_{p',q'}$ can be defined where

$$\begin{aligned} p' &= p - h' + 1 \\ q' &= q - l' + 1. \end{aligned} \tag{5}$$

The set $\mathscr{G}$ can be derived from $\mathscr{G}'$ by the following mapping:

$$x_{i',j'} = \begin{cases} x'_{p'+i'-1,q'+j'-1}, & \text{for } p', q' > 0 \\ 0, & \text{for } p' \leqslant 0 \text{ or } q' \leqslant 0 \end{cases} \tag{6}$$

for $i'=1,2,\cdots,M'$ and $j'=1,2,\cdots,N'$. Fig. 1 depicts the sets $\mathscr{F}$ and $\mathscr{G}$ for a pixel $x'_{p,q}$ and its associated weight, $w_{h',l'}$, both in the interior of $\mathscr{E}$ and $\mathscr{H}$, respectively.

Example I:

The following example illustrates the decomposition given by (2) and (3) for a 3×3 weight array and a 4×7 input array. The respective arrays are defined for

$$\begin{aligned} R = 3 \text{ and } S = 3\text{: } w_{h,l}\text{: } & h = 1,2,3 \\ & l = 1,2,3 \\ M = 4 \text{ and } N = 7\text{: } x'_{i,j}\text{: } & i = 1,2,3,4 \\ & j = 1,2,3,4,5,6,7 \\ M' = 6 \text{ and } N' = 9\text{; } x_{i',j'}\text{: } & i' = 1,2,3,4,5,6 \\ & j' = 1,2,3,4,5,6,7,8,9 \end{aligned}$$

and from (4), for $h=1$ and $i=1,2,3,4$ a set of seven simultaneous equations denoted by the vector $y_1(i,j)$ can be obtained. Similarly, the remaining partial output vectors $y_2(i,j)$ and $y_3(i,j)$ for $h=2$ and 3 respectively can be deduced. Hence by (2), the complete result is

$$y(i,j) = y_1(i,j) + y_2(i,j) + y_3(i,j),$$

$$\text{for } i=1,2,3,4;\ j=1,2,3,4,5,6,7.$$

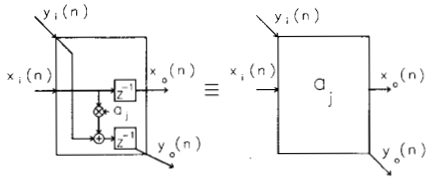

Fig. 2. Definition of the basic cell.

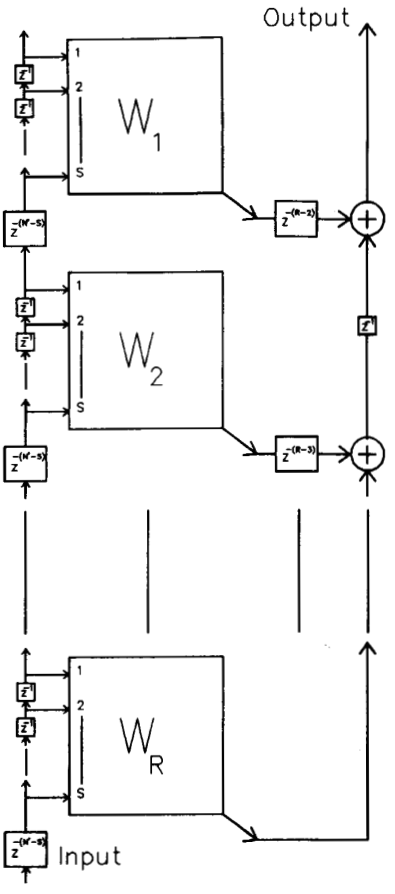

Fig. 3. General systolic array for method I.

III. Method I: Serial-Input Serial-Output Systolic Array

By considering (4) as a set of N simultaneous equations, it may be written in a matrix form as

$$\begin{bmatrix} y_h(i,1) \\ y_h(i,2) \\ \vdots \\ y_h(i,N) \end{bmatrix}^T = \begin{bmatrix} w_{h,1} \\ w_{h,2} \\ \vdots \\ w_{h,S} \end{bmatrix}^T \begin{bmatrix} x_{i+h-1,1} & x_{i+h-1,2} & \cdots & x_{i+h-1,N} \\ x_{i+h-1,2} & x_{i+h-1,3} & \cdots & x_{i+h-1,N+1} \\ \vdots & & & \vdots \\ x_{i+h-1,S} & x_{i+h-1,S+1} & \cdots & x_{i+h-1,N'} \end{bmatrix} \tag{7}$$

or

$$[y_h(i,j)]^T = [W_h]^T \cdot [X_{hi}]^T \tag{8}$$

where $[X_{hi}]^T$ is an $S \times N$ matrix.

Equation (7) represents the computation of (3) for any h by a serial-input serial-output system. The complete convolution is then obtained from (2) by summing the outputs of R arrays ($h=1$ to R) each defined by (7). Fig. 2 shows the definition of the basic cell for forming the systolic array of method I. The complete array for the general case is shown in Fig. 3 in which the W_h sub-array contains the entries in the $[W_h]^T$ matrix. The delay lines $z^{-(N'-S)}$ between the computing arrays are used to buffer the row pixels not involved in the immediate computation. For processing sequences longer than N', corresponding longer delay lines are required. For illustration, the general input and output sequences of the array are, respectively, given by

$$\begin{matrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,N'} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,N'} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ x_{M',1} & x_{M',2} & x_{M',3} & \cdots & x_{M',N'} \end{matrix}$$

and

$$\begin{matrix} y(1,1) & y(1,2) & y(1,3) & \cdots & y(1,N') \\ y(2,1) & y(2,2) & y(2,3) & \cdots & y(2,N') \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ y(M,1) & y(M,2) & y(M,3) & \cdots & y(M,N'). \end{matrix}$$

Note that the last $N'-N$ output values of $y(i,j)$ for $i=1,2,\cdots,M$ are not required and can be simply discarded. They arise from computations done when filling the input pipeline with the next row of pixels.

Example II:

Example I can be written in the form of (7) and for the case of $R=S=3$, $M=4$, and $N=7$ is

$$\begin{bmatrix} y_h(i,1) \\ \vdots \\ y_h(i,7) \end{bmatrix}^T = \begin{bmatrix} w_{h,1} \\ w_{h,2} \\ w_{h,3} \end{bmatrix}^T \begin{bmatrix} x_{i+h-1,1} & x_{i+h-1,2} & \cdots & x_{i+h-1,7} \\ x_{i+h-1,2} & x_{i+h-1,3} & \cdots & x_{i+h-1,8} \\ x_{i+h-1,3} & x_{i+h-1,4} & \cdots & x_{i+h-1,9} \end{bmatrix}.$$

The complete computation for any i is the sum of the outputs of the W_1, W_2 and W_3 which is

$$[Y(i,j)]^T = [Y_1(i,j)]^T + [Y_2(i,j)]^T + [Y_3(i,j)]^T$$

for $i=1,2,3,4$ and $j=1,2,3,4,5,6,7$. The complete array is shown in Fig. 4. In this case $N'-N=2$ and depending on the application, it may be tolerable if the system's clock frequency is high.

IV. Method II: Parallel-Input Parallel-Output Systolic Array

In general, (4) can be partitioned into P sub-systems. In each of these P sub-systems, S parallel outputs can be obtained by passing I parallel inputs into an $S \times I$ weight array. The matrix

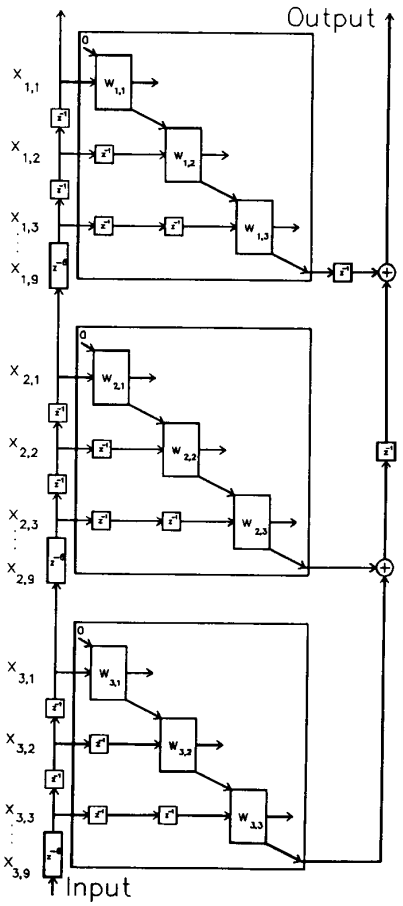

Fig. 4. Systolic array for method I for $R = S = 3$, $M = 4$, and $N = 7$.

representation for each of these sub-systems is

$$\begin{bmatrix} y_h(i,[p-1]S+1) \\ y_h(i,[p-1]S+2) \\ \vdots \\ y_h(i,[p-1]S+S) \end{bmatrix} = \begin{bmatrix} w_{h,1} & w_{h,2} & \cdots & w_{h,s} & 0 & 0 & 0 \\ 0 & 0 & \cdots & w_{h,s-1} & w_{h,s} & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & w_{h,1} & w_{h,2} & w_{h,s} & 0 \\ 0 & 0 & \cdots & 0 & w_{h,2} & w_{h,s-1} & w_{h,s} \end{bmatrix} \cdot \begin{bmatrix} x_{1+h-1,(p-1)S+1} \\ x_{i+h-1,(p-1)S+2} \\ \vdots \\ x_{i+h-1,(p-1)S+I} \end{bmatrix} \tag{9}$$

where for a given S: $h = 1,2,\cdots,R$, $i = 1,2,\cdots,M$ and $p = 1,2,\cdots,P$. Alternatively, (9) can be written as

$$\boldsymbol{Y}_h(i,j) = \boldsymbol{W}_{h,l} \cdot \boldsymbol{X}_p(i+h-1,k) \tag{10}$$

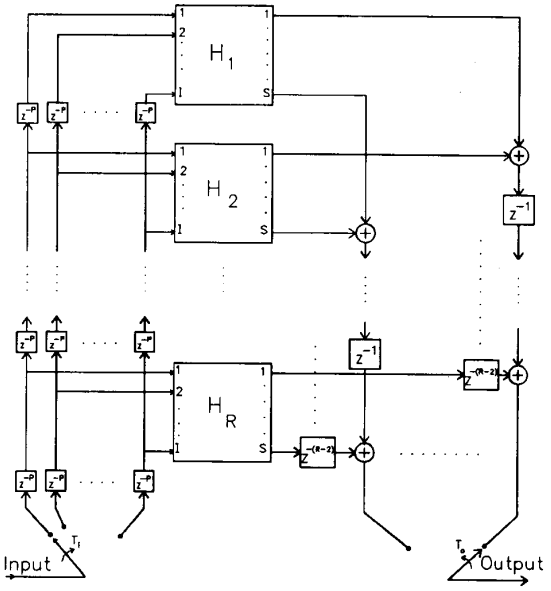

Fig. 5. The general systolic array for R sub-systems.

where

$$j = \begin{cases} 1,2,\cdots,S, & \text{for } p=1 \\ S+1,\cdots,2S, & \text{for } p=2 \\ \cdots & \\ (p-1)S+1,\cdots,pS, & \text{for any } p \\ \cdots & \\ (P-1)S+1,\cdots,PS, & \text{for } p=P \end{cases}$$

$$k = \begin{cases} 1,2,\cdots,I, & \text{for } p=1 \\ S+1,\cdots,S+I, & \text{for } p=2 \\ \cdots & \\ (p-1)S+1,\cdots,(p-1)S+I, & \text{for any } p \\ \cdots & \\ (P-1)S+1,\cdots,(P-1)S+I, & \text{for } p=P. \end{cases} \tag{11}$$

$\boldsymbol{W}_{h,l}$ is an $S \times I$ matrix, $\boldsymbol{X}_p(i+h-1,k)$ is an $I \times 1$ vector hereafter referred to as a component and $\boldsymbol{Y}_h(i,j)$ is an $S \times 1$ partial convolution vector. Hence (9) represents the computation of (3) for any h by an I-input S-output parallel system. The complete convolution is obtained by evaluation of (9) for each h and then summing the corresponding outputs of the R sub-arrays ($h = 1 \cdots R$) as written in (2).

From (4), each partial convolution sum, $y_h(i,j)$, requires S elements of the input sequence. In particular, $y_h(i,k)$ requires $x_{i+h-1,j}$ with j ranging from k to $S+(k-1)$ hence $y_h(i,S)$ requires $x_{i+h-1,j}$ with j ranging from S to $S+(S-1) = 2S-1$.

Since (9) computes S outputs in parallel, the above argument shows that the range of inputs required for the Sth output to be completely computed is from 1 to $2S-1$ for each h. The number of inputs, I, to a sub-array must thus be

$$I = 2S-1. \tag{12}$$

Furthermore, the above argument also indicates that in computing the partial convolution sums, successive partial sums require some common input elements. To take this into account, the component $\boldsymbol{X}_p(i+h-1,k)$, in (10), has its last $I-S$ ele-

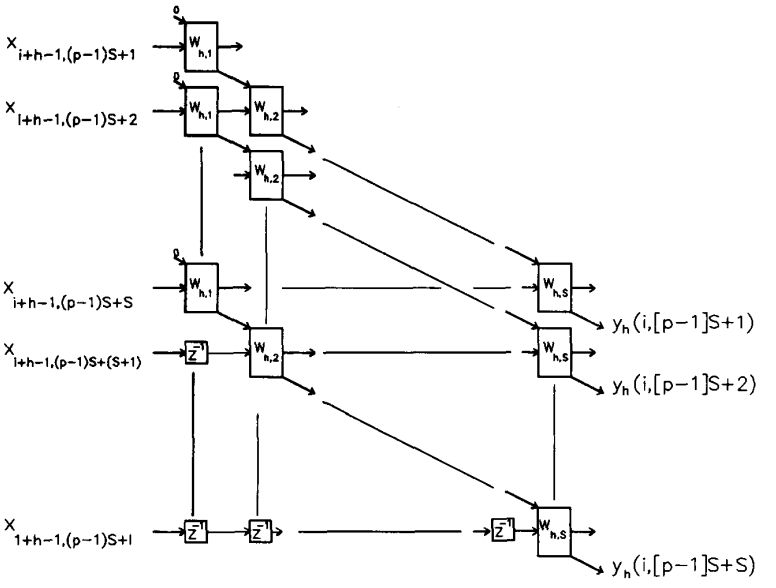

Fig. 6. The general sub-array H_h.

ments equal to the first $I-S$ elements of the next component, $X_{p+1}(i+h-1,k)$. Thus in general, from (12), the last $S-1$ elements of $X_p(i+h-1,k)$ have to be overlapped in successive computations and is achieved by the following partitioning of the input sequence. Let a subset of the input sequence including the boundary conditions be denoted by the sub-vector X_i' defined as

$$X_i' = [\, X_{i,1} \quad x_{i,2} \quad \cdots \quad x_{i,N'}]^T \tag{13}$$

which is of length N' and is greater than I. A new partitioned sub-vector X_{iP} of P components, can be defined as

$$X_{iP} = \left[X_1(i,k) \,|\, X_2(i,k) \,|\, \cdots \,|\, X_p(i,k) \,|\, \cdots \,|\, X_P(i,k) \right]^T \tag{14}$$

where each of the components, $X_p(i,k)$ containing I elements, can be obtained by processing successive elements of X_i' such that the last $I-S$ elements of $X_p(i,k)$ form the first $I-S$ elements of the next component $X_{p+1}(i,k)$. Instead of storing these overlapping elements in the array, they are retransmitted, which results in simpler interconnections. The relationship between the component $X_p(i+h-1,k)$, which is the vector input to a computing array, H_h, and the component, $X_p(i,k)$, of the input sub-vector X_{iP} into the input switch (see Fig. 5) is

$$X_p(i,k) = X_p(i+h-1,k)\big|_{h=1,2,\cdots,R}, \qquad \text{for any } i \text{ and } p. \tag{15}$$

Equation (15) gives the particular component of the input sub-vector, X_{iP}, which is applied to the sub-array H_h for a given row, i, and a partition p. A distinction between, $X_p(i,k)$ and $X_p(i+h-1,k)$ is necessary because the same component, $X_p(i,k)$, is processed by each computing sub-array, H_h, but at different time instants. Since each of the P components of X_{iP} has the last $I-S$ elements equal to the first $I-S$ elements of the next component, an expression for the total number of nonzero elements in X_{iP} is

$$IP = N' + (I-S)(P-1) \tag{16}$$

using (12) and (16) can be simplified to

$$P = (N'+1)/S - 1.$$

Hence,

$$P = \left[\frac{N'+1}{S} - 1 \right] + 1 \tag{17}$$

where $[z]$ denotes the integer part of z. The elements of the Pth component of X_{iP} with no corresponding entries in the X_i' matrix are set to zero. It should be noted that no overlapping of elements is needed between the last component of X_{iP} and the first component of X_{i+1P} of the input sequence.

Fig. 5 shows the general systolic array for method II. In this figure, the input sequence is fed at a higher sample rate $f_i = 1/T$ through a switch while each of the sub-arrays ($H_h, h=1,2,3$) operates at a lower processing rate $f_p = f_i/I$. The output is collected by another switch at the rate $f_0 = 1/T_0 = Sf_p$. With an alternative choice of I and S, a different design for high-speed processing can be obtained. The computing sub-array for any h, that is the H_h array, is shown in Fig. 6.

Associated with each sub-array, H_h, is a group of I delay lines each having a delay of z^{-P}. They are used to store all the P components of X_{iP}, i.e., $X_p(i,k)$, for $p=1,2,\cdots,P$ by the hth (for $h=1,\cdots,R$) group of delay lines, in the sequence of their p variable, beginning with the delay line feeding the first input of the associated sub-array. For illustration, the general input sequence of the complete array can be deduced from (11) and (14) as

$$[X_{iP}]^T = \left\{ [X_1(i,k)]^T \,|\, \cdots \,|\, [X_p(i,k)]^T \,|\, \cdots \,|\, [X_P(i,k)]^T \right\}$$

for $i=1,2,\cdots,M'$ and which can be written as

$$[X_{1P}]^T = [x_{1,1} \cdots x_{1,I}] \,|\, \cdots \,|\, [x_{1,(p-1)S+1} \cdots x_{1,(p-1)S+I}] \,|\, \cdots \,|\, [x_{1,(P-1)S+1} \cdots x_{1,(P-1)S+I}]$$

$$[X_{2P}]^T = [x_{2,1} \cdots x_{2,I}] \,|\, \cdots \,|\, [x_{2,(p-1)S+1} \cdots x_{2,(p-1)S+I}] \,|\, \cdots \,|\, [x_{2,(P-1)S+1} \cdots x_{2,(P-1)S+I}]$$

$$\cdots$$

$$[X_{M'P}]^T = [x_{M',1} \cdots x_{M',I}] \,|\, \cdots \,|\, [x_{M',(p-1)S+1} \cdots x_{M',(p-1)S+I}] \,|\, \cdots \,|\, [x_{M',(P-1)S+1} \cdots x_{M',(P-1)S+I}].$$

Similarly, the output sequence is

$$[y(1,1)\,y(1,2)\cdots y(1,S)]\,|\,[y(1,S+1)\cdots y(1,2S)]$$
$$|\cdots|\,[y(1,[P-1]S+1)\cdots y(1,PS)]$$
$$|\,[y(2,1)\,y(2,2)\cdots y(2,S)]$$
$$|\cdots|\,[y(2,S+1)\cdots y(2,2S)]\,|\cdots$$
$$|\,[y(2,[P-1]S+1)\cdots y(2,PS)]$$
$$|\cdots[y(M,1)\,y(M,2)\cdots y(M,S)]$$
$$|\,[y(M,S+1)\cdots y(M,2S)]$$
$$|\cdots|\,[y(M,[P-1]S+1)\cdots y(M,PS)].$$

Example III:

Example I can also be implemented by method II. Consider $S=R=3$, $M=4$, $M'=6$, $N=7$, $N'=9$, $I=5$, and $P=3$. The input sequence is given by

$$\boldsymbol{X}_i' = [x_{i,1}\ \ x_{i,2}\ \ x_{i,3}\ \ x_{i,4}\ \ x_{i,5}\ \ x_{i,6}\ \ x_{i,7}\ \ x_{i,8}\ \ x_{i,9}]^T$$
$$\boldsymbol{X}_{i3} = [\boldsymbol{X}_1(i,k)\,|\,\boldsymbol{X}_2(i,k)\,|\,\boldsymbol{X}_3(i,k)]^T$$
$$\boldsymbol{X}_1(i,k) = [x_{i,1}\ \ x_{i,2}\ \ x_{i,3}\ \ x_{i,4}\ \ x_{i,5}]^T$$
$$\boldsymbol{X}_2(i,k) = [x_{i,4}\ \ x_{i,5}\ \ x_{i,6}\ \ x_{i,7}\ \ x_{i,8}]^T$$
$$\boldsymbol{X}_3(i,k) = [x_{i,7}\ \ x_{i,8}\ \ x_{i,9}\ \ 0\ \ 0]^T$$

for $i=1,2,\cdots,M'$. From (9), the parallel input–output matrix representation is

$$\begin{bmatrix} y_h(i,[p-1]3+1) \\ y_h(i,[p-1]3+2) \\ y_h(i,[p-1]3+3) \end{bmatrix} = \begin{bmatrix} w_{h,1} & w_{h,2} & w_{h,3} & 0 & 0 \\ 0 & w_{h,1} & w_{h,2} & w_{h,3} & 0 \\ 0 & 0 & w_{h,1} & w_{h,2} & w_{h,3} \end{bmatrix} \cdot \begin{bmatrix} x_{i+h-1,(p-1)3+1} \\ x_{i+h-1,(p-1)3+2} \\ x_{i+h-1,(p-1)3+3} \\ x_{i+h-1,(p-1)3+4} \\ x_{i+h-1,(p-1)3+5} \end{bmatrix}$$

for $h=1,2,3$, $i=1,2,3,4$, and $p=1,2,3$.

Expanding in h, the sub-systems H_1, H_2, and H_3 for $h=1$, 2, and 3, respectively, can be implemented as arrays. The H_1 array does the following computations:

$$\begin{bmatrix} y_1(i,1) \\ y_1(i,2) \\ y_1(i,3) \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & 0 & 0 \\ 0 & w_{1,1} & w_{1,2} & w_{1,3} & 0 \\ 0 & 0 & w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix} \begin{bmatrix} x_{i,1} \\ x_{i,2} \\ x_{i,3} \\ x_{i,4} \\ x_{i,5} \end{bmatrix}$$

$$\begin{bmatrix} y_1(i,4) \\ y_1(i,5) \\ y_1(i,6) \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & 0 & 0 \\ 0 & w_{1,1} & w_{1,2} & w_{1,3} & 0 \\ 0 & 0 & w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix} \begin{bmatrix} x_{i,4} \\ x_{i,5} \\ x_{i,6} \\ x_{i,7} \\ x_{i,8} \end{bmatrix}$$

$$\begin{bmatrix} y_1(i,7) \\ y_1(i,8) \\ y_1(i,9) \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & 0 & 0 \\ 0 & w_{1,1} & w_{1,2} & w_{1,3} & 0 \\ 0 & 0 & w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix} \begin{bmatrix} x_{i,7} \\ x_{i,8} \\ x_{i,9} \\ 0 \\ 0 \end{bmatrix}$$

or

$$\boldsymbol{Y}_1(i,j) = \boldsymbol{W}_{1,l} \cdot \boldsymbol{X}_p(i,k) \quad \text{for } p=1,2,3;\ l=1,2,3.$$

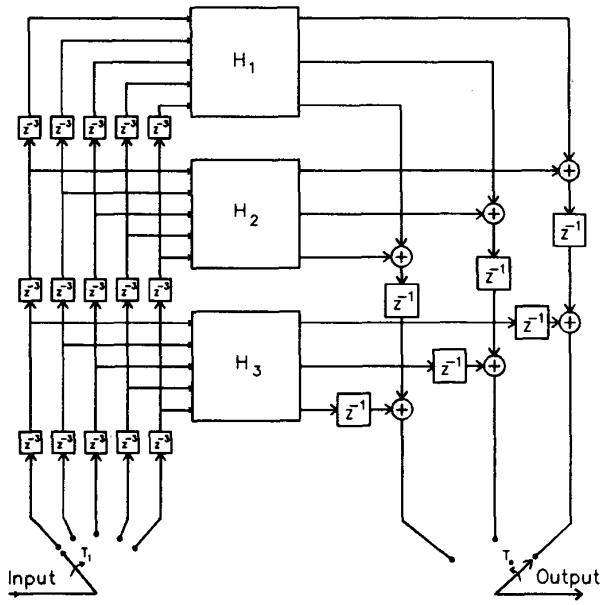

Fig. 7. Overall systolic array for $R=S=3$, $M=4$, and $N=7$.

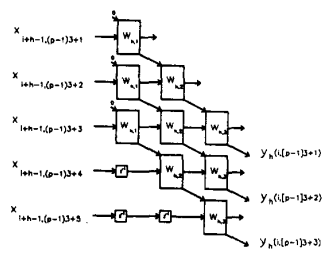

Fig. 8. Sub-array H_h for $R=S=3$, $M=4$, and $N=7$.

TABLE I
COMPARISON OF THE THROUGHPUT RATE AND HARDWARE REQUIREMENTS FOR AN $R \times S$ 2-D CONVOLVER

Method	Number of Cells Required	Output Throughput Rate	Latency	I/O Bandwidth Input	I/O Bandwidth Output
I.	RS	$1/T_p$	$(S+R-2)T_p$	$1/T_p$	$1/T_p$
II.	RS2	S/T_p	$(S+R-2)T_p$	I/T_p	S/T_p
III.	QRS	Q/T_p	$(S+R-2)T_p^*$	K/T_p	Q/T_p

* A delay of QST_p is needed for the full set of Q simultaneous output.

Similarly, the partial output vectors $\boldsymbol{Y}_2(i,j)$ and $\boldsymbol{Y}_3(i,j)$ for which $h=2$ and 3, respectively, are

$$\boldsymbol{Y}_2(i,j) = \boldsymbol{W}_{2,l} \cdot \boldsymbol{X}_p(i+1,k), \quad \text{for } p=1,2,3;\ l=1,2,3$$
$$\boldsymbol{Y}_3(i,j) = \boldsymbol{W}_{3,l} \cdot \boldsymbol{X}_p(i+2,k), \quad \text{for } p=1,2,3;\ l=1,2,3.$$

From (2) the complete output is then given by

$$\boldsymbol{Y}(i,j) = \boldsymbol{Y}_1(i,j) + \boldsymbol{Y}_2(i,j) + \boldsymbol{Y}_3(i,j),$$
$$\text{for } i=1,2,3,4;\ j=1,2,3,4,5,6,7.$$

The overall input and output sequences for this example can be derived from (14).

Note that the output values for $j = 8$ and 9 are zero and can be discarded. They arise from the zero input values appended to X_{iP} to make each of its components, $X_p(i,k)$, have I elements. Similarly, the computation of these $N' - N$ output values cannot be avoided; however, they do not increase the overall computation time. The complete array is shown in Fig. 7 while Fig. 8 shows the computing array for H_h for $h = 1,2,3(= R)$.

V. Hardware Requirements

The arrays of both methods have a latency of $S + R - 2$ at the processing period T_p, which for method II, is equivalent to $I(S + R - 2)$ at the input sample period T_i. Table I gives a comparison of the hardware requirements of methods I and II with that of the method described in [1], which shall be referred to as method III. The given latency is obtained assuming that the input pipeline of the array has been completely filled with the first values of the input sequence by the host such that the first set of input values have been positioned at the inputs to each sub-array, H_h for $h = 1,2,\cdots,R$. The output throughput rates of methods I and II are constant for a given R, S and I. On the other hand, method III has a varying output throughput rate ranging from $1/T_p$, after S initial clock periods, and increases every S clock periods to a maximum of Q/T_p when all kernels are in operation.

The output bandwidth also varies in a similar manner. In Table I, the number of rows that the array of method III can access simultaneously from the host has been denoted K while Q denotes the number of $R \times S$ kernels used.

VI. Concluding Remarks

In this paper, two 2-D systolic arrays for 2-D convolution have been presented. The arrays are regular with only nearest neighbor interconnections. The array of method I has the smallest I/O bandwidth and it performs parallel computations with the fewest number of cells thus it is suitable for applications limited by chip area. Method II is most suitable for high-speed implementation with slow processing elements. In comparison with method III, method II requires slightly more hardware for the implementation of the switches involved. However, for the same input and output bandwidths, method II has the lower input and output pin count per chip.

Furthermore, slower processing elements can be used in method II which results in a lower I/O bandwidth as compared with method III. On the whole, the inherent parallelism of the matrix–vector algorithms in both methods I and II have resulted in regular structures and are most suited for VLSI implementation.

References

[1] H. T. Kung, "Special-purpose devices for signal and image processing: An opportunity in very large scale integration (VLSI)," in *Proc. Soc. of Photo-optical Instrumentation Engineers*, vol. 241, pp. 76–84, July 1980.

[2] H. T. Kung and R. L. Picard, "One-dimensional systolic arrays for multidimensional convolution and resampling," in *VLSI for Pattern Recognition and Image Processing*, New York: Springer-Verlag, 1984, pp. 9–24.

[3] Okan Ersoy, "Semisystolic array implementation of circular, skew circular, and linear convolutions," *IEEE Trans. Computers*, vol. C-34, pp. 190–196, Feb. 1985.

[4] T. Gross, H. T. Kung, M. Lam, and J. Webb, "Wrap as a machine for low-level vision," in *IEEE Int. Conf. on Robotics and Automation*, St. Louis, MO, pp. 790–800, Mar. 1985.

Computer-Aided Mode Analysis of Coupled Nonlinear Oscillators

D. AOUN AND D. A. LINKENS

***Abstract*—The theoretical advances in the analysis of coupled nonlinear oscillators have been strongly motivated by the mathematical modeling of biomedical systems, particularly the electrical activity of the gastro-intestinal digestive tract. In recent years a matrix extension of the Krylov–Bogolioubov linearization technique has been developed and applied to a wide range of structures comprising chains, arrays, rings, and tubes. This technique produces complicated stability criteria for the existence of stable limit cycles. The CAD package described here solves these criteria for the structures mentioned above with an arbitrary number of either third or fifth power conductance van der Pol oscillators coupled mutually.**

I. Introduction

Nonlinear systems have been investigated over the past half century by many authors both in the East and the West, van der Pol [17], [18], Voronin [19], and so on. Although it is clear that oscillations and nonlinear oscillators are of great importance in many different applications and for various technical systems, the focus here will be on their application in bioelectronics, in particular, towards the electrical activity of the mammalian gastro-intestinal tract. This activity known as "slow-waves" has been extensively modeled using nonlinear oscillatory dynamics. Such a mathematical model was first proposed by Nelsen and Becker in 1968 as an alternative to a linear cable model. Since that time the mathematical model has been extended to comprise a one-dimensional chain of coupled van der Pol oscillators for small-intestinal studies [15] and a two-dimensional array for gastric modeling [16].

Mode analysis of intercoupled van der Pol oscillators has been keeping pace with simulation studies. Harmonic balance techniques have established the multimode behavior for general *RLC* coupling of two oscillators [10], while one-dimensional chains with intrinsic frequency gradients have yielded analytical entrainment conditions [9]. The same methods have also been applied successfully to the condition of "almost-entrainment" or modulation, where multiple spectral components occur [12]. In parallel with this a matrix Krylov–Bogolioubov method has been developed for one-dimensional ladders [6], a ring of oscillators [7] and tubular structures [2]. The fifth power conductance van der Pol oscillator with its stable zero state has relevance to gastro-intestinal modeling, and has also been analyzed with the matrix Krylov–Bogolioubov approach [4].

Mode analysis of such structures produces equations for mode frequencies, criteria for determining stable modes and multimodes together with their types, as well as information on the structural solutions of these modes. However, these stability criteria contain complicated mathematical expressions and rigorous matrix search procedures (e.g., in the determination of double mode stability for an $m \times n$ array it is necessary to search an $mn \times mn$ matrix for every pair of modes). Hence it was decided to engineer a CAD package for the complete analysis of any system of mutually coupled oscillators. Four different structures

Manuscript received March 9, 1988; revised April 29, 1989. This paper was recommended by Associate Editor T. Matsumoto.

The authors are with the Department of Control Engineering, University of Sheffield, Sheffield S1 3JD, U.K.

IEEE Log Number 8932877.

0098-4094/90/0200-0273$01.00 ©1990 IEEE