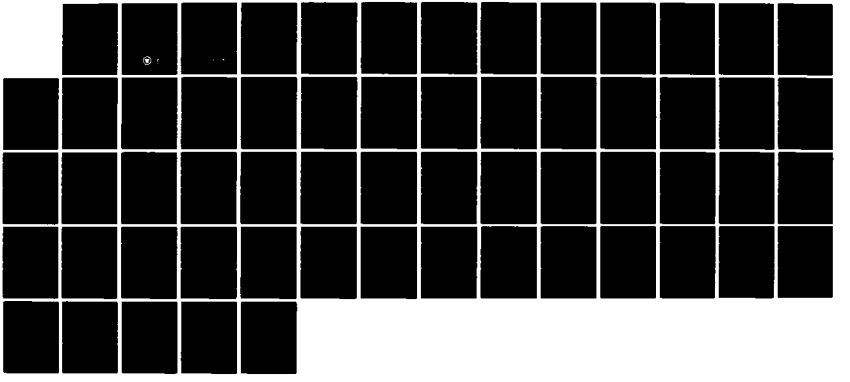
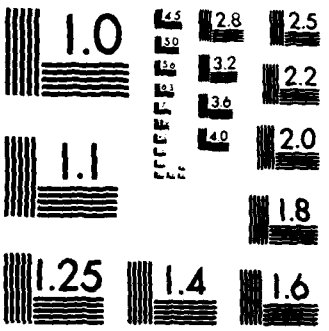
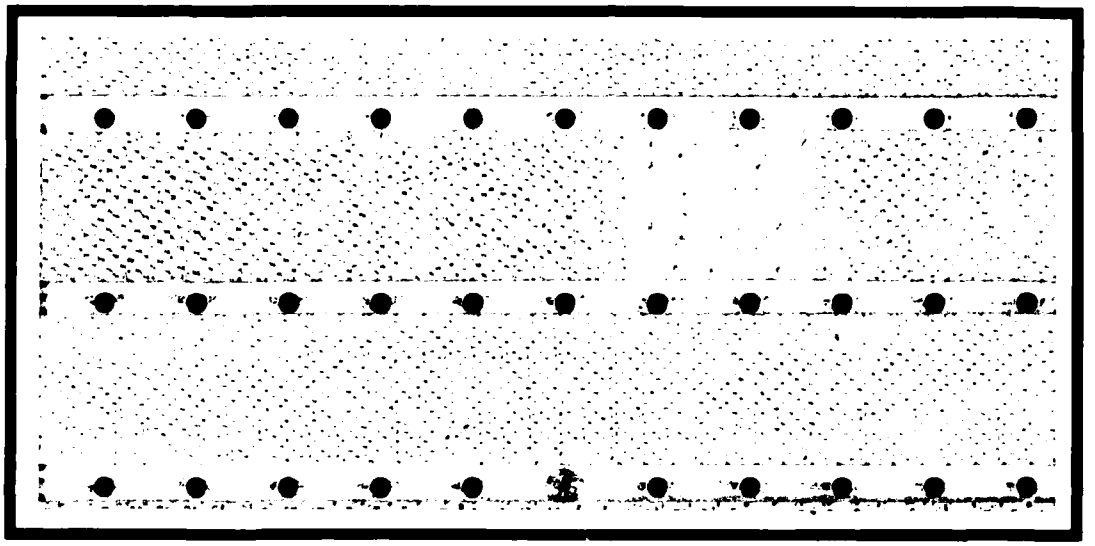
AD-A130 294 FORMAL VERIFICATION OF SYSTOLIC SYSTEM FOR FINITE ELEMENT STIFFNESS MATRICES(U) PITTSBURGH UNIV PA INST FOR COMPUTATIONAL MATHEMATICS AND APP.. R MELHEM 1982 1/1

UNCLASSIFIED ICMA-83-56 N00014-80-C-0455 F/G 12/1 NL





INSTITUTE FOR COMPUTATIONAL MATHEMATICS AND APPLICATIONS



Department of Mathematics and Statistics

University of Pittsburgh

Technical Report ICMA-83-56

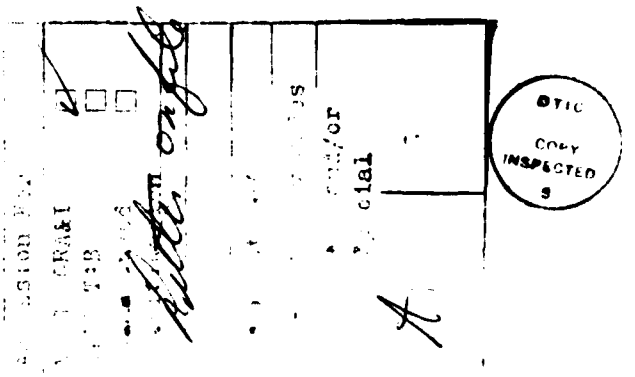
FORMAL VERIFICATION OF A SYSTOLIC SYSTEM FOR FINITE ELEMENT STIFFNESS MATRICES*)

by

Rami Melhem

Department of Computer Science
and
Department of Mathematics and Statistics
University of Pittsburgh
Pittsburgh, PA 15261

*)This work in part was supported under ONR Contract N00014-80-C-0455



1. Introduction

In [1] an abstract model was developed for the specification of systolic networks [2] and the verification of the correctness of their operation. The model was applied to the verification of the operation of four systolic networks that had been suggested in the literature. In this report, we extend this model to allow for networks with slightly more complicated types of computational cells, namely cells that have periodic memory or multiplexing capabilities.

The motivation for this extension is that we have to free ourselves from the simple inner product cell [3] if we want to use systolic networks in a wider range of applications. It should be noted, however, that the suggested extensions remain very simple in structure and should not result in a complicated design for the individual cells. Also it appears that the most desirable approach to the design of widely applicable systolic networks is to utilize a fairly general generic cell which is flexible enough to be used in more that one systolic network. If this generic cell were to be controlled by microcode, then it could be applied easily to the implementation of the suggested extended cells.

The model presented in [1] and extended in this report is similar to another model developed independently by M. C. Chen [4]. Both separate the network function from the specific details of a certain computation and allow for a precise specification and a formal verification of systolic networks. However, the model in [4] is oriented toward a procedural specification, while we followed a more algebraic approach. We should also mention previous approaches [5,6] for formalizing systolic networks by means of a delay operator [7] and a notation that envisions the flow of data as a wave front propagating over the network. This wave front

notation has been shown to be useful in mapping given algorithms to systolic implementations [8]. However, the notation does not seem to be powerful enough to describe the operation of any systolic network, especially if more elaborate computational cells are to be used.

The extended model is applied to the description and verification of a pipelined systolic system designed for the computation of finite element stiffness matrices. This represents an important step in the finite element analysis extensively used by engineers and scientists for the solution of boundary value problems.

Very briefly, the finite element analysis [9] is a technique for solving partial differential equations on a certain domain Q with given conditions on the boundary of Q. In the case of linear equations, it involves essentially the following four basis steps: 1) The generation of a finite element mesh that divides Q into m finite elements. 2) The generation of elemental stiffness matrices H^e and elemental load vectors f^e for each finite element e, e=1,....,m. 3) The assembly of the global stiffness matrix H and of the load vector f. 4) The solution of the linear system of equations Hx=f.

In the past two decades, many finite element software systems have been developed and widely used [10]. However, in practice, the time and storage required by these systems to complete an analysis may be extremely large. This usually imposes severe limitations on the size and type of the problem that can be handled and often leads engineers to use less accurate models or lower degrees of approximations. For this reason, many researchers have considered some form of parallel processing in the finite element analysis, as for instance, the use of array processors [11, 12, 13, 14], general purpose multiprocessors [15, 16], or adaptive, special purpose multiprocessor systems [17, 18]. A common result in

most of these experiments is that the time for data movement and interprocessor communication is very large and sometimes dominates the running time.

A significant achievement in this area is the design of a finite element machine at the NASA-Langley Research Center [19,20]. In this machine, a rectangular array of processors is formed by connecting each processor to its eight nearest neighbors with a global bus connecting all the processors of the system. Each processor is assigned to the computations associated with one or more node in the finite element mesh. Of course the nodes in the mesh should be mapped to the available processors in a way that reduces the communications over the global bus [21].

Along the line of systolic architectures, Law [22] suggested a systolic network to assemble the global stiffness matrix, and Kung and Leiserson [3] and Brent [23] designed systolic networks that can be used for solving the resulting system of equations. However, no attempts have been made to use systolic networks for generating the elemental stiffness matrices, which is the subject of this report.

The report is arranged as follows: In Section 2, we review and extend the basic features of the systolic model presented in [1], and in section 3, we give a general description of the system used to generate the elemental stiffness matrices. The different components of the system are formally described in section 4, where we also prove that the system indeed produces the stiffness matrix corresponding to any element. In section 5, we outline a general technique for the formal verification of the pipelined operation of any systolic network and then apply this technique to prove that the suggested finite element system can be pipelined to compute all the elemental matrices. A conclusion indicates some directions for further studies.

2. Review and Extension of the Formal Systolic Model.

In this section, we briefly review the main features of the abstract systolic model presented in [1]. Basically a systolic network is represented by a directed graph with two different types of nodes, namely interior nodes and I/O nodes corresponding to computational cells and I/O cells of the network, respectively. The edges of the graph model the communication links of the network. In order to identify the elements of the graph, every node is given a unique label and every edge is identified by a pair (c,i), where c is a color assigned to the edge from a finite set of colors, and i is the label of the node at which the edge terminates. The only restriction placed upon the edge colors is that edges directed to the same node should have different colors and that the same holds for all edges directed out of a node.

In addition to the graph that reflects the topology of the network, the model associates with each edge an infinite data sequence which is the sequence of data items that appear on the corresponding communication link at consecutive time units. More precisely, let N and R be the sets of positive integers and real numbers, respectively, and set $R_\delta = R \cup \{\delta\}$, where δ is a special element called the "don't care" element. Then the data sequence η_l associated with the edge (y,i) is a mapping $\eta_l : N \rightarrow R_\delta$ such that $\eta_l(t) \in R_\delta$ is the data item which appears on the link at time t. If $\eta_l(t)=\delta$ for some t, this indicates that we do not care (or do not know) about the data on (y,i) at the time t. We use the convention of denoting the pair (y,i) by y_l and the associated sequence by η_l, where η is the greek letter corresponding to y. At this point, we note that we have chosen R to be the set of real numbers because of the nature of our problem. More generally, R could be any set of items that can be transmitted on the communication links of the network.

Let $\bar{R}_\delta$ be the set of all sequences that contain at most a finite number of non-δ elements. Then it is natural to define the termination function $T:\bar{R}_\delta \to N_0 = N \cup \{0\}$ with the property that for any sequence η, $T(\eta)$ is the position of the last non-δ element in η. For the don't care sequence defined by $\delta^*(t)=\delta$ for all t, we then have $T(\delta^*)=0$. We also define the zero sequences ι with $\iota(t)=0$ for $1 \leq t \leq T(\iota)$ and any arbitrary large $T(\iota)$.

The computation performed by a computational cell with m input links and n output links is now modeled by n causal sequence operators $\Gamma_i:[\bar{R}_\delta]^m \to \bar{R}_\delta$, i=1,....,n, one for each output link. In essence, a causal operator is such that the t^{th} element of the image sequence can depend only on any element j of its operands with $j<t$. If the condition $j<t$ is replaced by $j \leq t$, the operator is called 'weakly causal'. For the exact definition of causal and weakly causal operators we refer to [1].

In order to model the computation of the entire network, we establish for each node of the network the sequence equations describing its operation, these are the equations relating the input sequences and the output sequences by means of causal operators. Then, if possible, we solve the resulting system of equations and obtain in this way an explicit relation between the network output sequences and the network input sequences. This relation is called the "Network I/O Description". Finally, for a verification of the operation of the network for a specific form of input sequences, we substitute these particular sequences into the I/O description, which, possibly after some manipulation, yields an explicit form of the network output sequences.

As the above review already indicates, operators on sequences play a key role in our model. One way of defining sequence operators is to extend known operators on R to $\bar{R}_\delta$ by applying the operator element-wise to the elements of

sequences. Examples are the sequence addition '+', multiplication '*' and scalar multiplication '.'. Element wise operators, in turn, can be classified in terms of the result of any operation involving the don't care element δ, namely: 1) δ-regular operators for which the result of any operation involving δ is δ. This class of operators treats δ as a "don't know" quantity, and consequently the result cannot be known if any of the operands is not known. 2) Non δ-regular operators, where δ is treated as a special symbol that affects the result of the operation. Example are the operators min_δ and max_δ defined in [1]. In practice, this class of operators can be used to model a network where the communication links are augmented by an additional wire to indicate whether the link carries valid data or not. The operation of each computational cell is then dependent on this additional piece of information.

A second class of operators consists of those defined directly on $\bar{R}_\delta$. In the remainder of this section we introduce several such operators that will be used in the specification and verification of our finite element system. For simplicity, given any operator $\Gamma:[\bar{R}_\delta]^n \rightarrow \bar{R}_\delta$, the notation $[\Gamma(\xi_1,\cdots,\xi_n)](t)$, will be employed to designate the t^{th} element $\eta(t)$ of the image sequence $\eta=\Gamma(\xi_1,\cdots,\xi_n)$. This is consistent with the convention of using square brackets for grouping. We will also use the symbol $\div$ for integer division and the Fortran function *mod*() that specifies the remainder of an integer division.

The Shift operator $\Omega^r : \bar{R}_\delta \rightarrow \bar{R}_\delta$ is defined by

$$[\Omega^r \ \xi](t) = \begin{cases} \delta & \text{if } r>0 \text{ and } t \leq r \\ \xi(t-r) & \textit{otherwise.} \end{cases}$$

Hence, for $r>0$, Ω^r inserts r δ-elements at the beginning of a sequence and therefore models the computation of a delay cell. On the other hand, for $r<0$, Ω^r

trims the first r elements of the sequence and thus is a non causal operator which cannot be used to model computational cells. The role of the negative shift operator is to provide in the proofs an inverse for the positive shift. More precisely, for any sequence ξ, we have $\Omega^{-r}\ \Omega^{r}\ \xi = \xi$. The converse is not always true, in the sense that $\Omega^{r}\ \Omega^{-r}\ \xi = \xi$ only if $\xi(t)=\delta$ for $t \leq r$.

The Zero Shift operator $\Omega_0^r : \bar{R}_\delta \rightarrow \bar{R}_\delta$ has the same definition as Ω^r except that Ω_0^r inserts r zeroes at the beginning of a sequence instead of r δ-elements. The zero shift operator is useful in modeling delay cells in networks that initially set the data on their communication links to zero. In such networks we must assume that the entries corresponding to the time t=1 in any non input sequence are equal to 0 rather than δ.

The Accumulator operator $A^{r,k,s} : \bar{R}_\delta \rightarrow \bar{R}_\delta$ is defined to model a cyclic accumulator that starts operation at time t=r, accumulates a new element every s time units and restart a new cycle every sk time units. The Accumulator operator can be defined in terms of the following algorithm that computes $[A^{r,k,s}\ \xi](t)$ for any $t>0$, given the sequence elements $\xi(l)$ for $l \leq t$.

IF $(t<r)$ THEN $[A^{r,k,s}\ \xi](t) = \delta$ /* accumulator is idle */

ELSE

BEGIN

$t_r = t - mod((t-r) + sk)$ /* time of last reset */

$na = ((t-t_r) \div s) + 1$ /* number of elements accumulated */

$$[A^{r,k,s}\ \xi](t) = \sum_{l=0}^{na-1} \xi(t_r + sl)$$ /* result of accumulating *na* elements */

END

Evidently, this algorithm is equivalent with

$$[A^{r,k,s}\ \xi](t) = \begin{cases} \delta & t<r \\ \sum_{j=0}^{na} \xi(t_r+sj) & t\geq r \end{cases}$$

where na and t_r are as specified before. As an example, let

$$\xi = a_1.b_1.a_2.b_2.\cdots.a_7.b_7.\delta.\delta.\cdots \tag{2.1}$$

then

$$A^{2,3,2}\ \xi = \delta.b_1.\bullet.b_1+b_2.\bullet.b_1+b_2+b_3.\bullet.b_4.\bullet.b_4+b_5.\bullet.b_4+b_5+b_6.\bullet.b_7.\bullet.\delta.\delta.\cdots$$

where $\bullet$ denotes an element that is equal to the preceding one.

The Multiplexer operator $M_r^{w1,\ldots,wn}(\xi_1,\cdots,\xi_n) : [\bar{R}_\delta]^n \rightarrow \bar{R}_\delta$ is defined to model a multiplexer that has n inputs $\xi_1,\cdots,\xi_n$. It starts its opration at time t=r and periodically multiplexes its inputs with a time ratio of $w1:w2:\cdots:wn$. If the length of the multiplexer cycle is denoted by $k=\sum_{e=1}^{n} w_e$, then the following algorithm defines the multiplexer operator

IF $(t<r)$ THEN $[M_r^{w1,\ldots,wn}(\xi_1,\cdots,\xi_n)](t) = \delta$ /* multiplexer idle */

ELSE

BEGIN

$t_c = t - mod((t-r) + k)$ /* start of current cycle */

Find the largest ineger $1\leq e\leq n$

such that $(t-t_c) < \sum_{j=1}^{e} w_j$ /* determine interval within cycle */

$[M_r^{w1,\ldots,wn}(\xi_1,\cdots,\xi_n)](t) = \xi_e(t)$ /* chose corresponding input */

END

As an example, let

$$\zeta = a_1.a_2.\cdots.a_7.a_8.a_9.\delta.\delta.\cdots$$

and

$$\eta = b_1.b_2.\cdots.b_7.\delta.\delta.\delta.\cdots$$

then

$$M_3^{1,2}(\zeta,\eta) = \delta.\delta.a_3.b_4.b_5.a_6.b_7.\delta.a_9.\delta.\cdots$$

It is also interesting to note that the multiplexer operator can be used to model a de-multiplexer cell. For example, if we want to sample the sequence ξ at times t=r,2r,3r,... , then we may express this operation as $M_r^{1,r-1}(\xi,\delta^*)$ where δ^* is the don't care sequence introduced earlier.

The multiplexer operator can be used to define two further operators, namely, the expansion and the piping operators.

The Expansion operator $E_r^k:\bar{R}_\delta \to \bar{R}_\delta$ models a cyclic memory that is loaded at time t=r and is overwritten every k time units. It is formally defined by

$$E_r^k\ \eta = M_r^{1,\ldots,1}(\eta\ ,\ \Omega\eta\ ,\ \Omega^2\eta\ ,\ \cdots\ ,\ \Omega^{k-1}\eta).$$

which on the basis of the definition of the multiplexer operator may be rewritten as

$$E_r^k\eta = \begin{cases} \delta & t<r \\ \eta(t-t_u) & t\geq r \end{cases}$$

where $t_u = mod((t-r) + k)$. For example, with ξ of (2.1) we have

$$E_2^4\xi = \delta.b_1.\bullet.\bullet.\bullet.b_3.\bullet.\bullet.\bullet.b_5.\bullet.\bullet.\bullet.b_7.\bullet.\bullet.\bullet.\delta.\delta.\cdots$$

It should be noted that the accumulator, multiplexer and expansion operators are weakly causal operators, and that their definitions allow us to model cells with memory capabilities, despite the fact that our abstract model does not expli-

citly allow the nodes to have memories or internal states.

Besides the causal and weakly causal operators used in modeling computational cells, some sequence operators are introduced here for the sole purpose of allowing us to simplify the description of data sequences. Following are two such operators:

The Piping operator $P_m^k : [\bar{R}_\delta]^m \to \bar{R}_\delta$ defined by

$$P_m^k(\eta^1,\cdots,\eta^m) = M_1^{k,\ldots,k}(\eta^1, \cdots, \Omega^{(i-1)k}\eta^i, \cdots, \Omega^{(m-1)k}\eta^m)$$

and $T(P_m^k(\eta^1,\cdots,\eta^m)) = mk$. In other words, P_m^k concatenates the first k elements of each of the m sequences η^e, $e=1,\cdots,m$, and forms one long sequence.

On the basis of the definition of the multiplexer operator it is easily shown that the following algorithm is equivalent with the above definition of the piping operator

IF$(t>mk)$ THEN $[P_m^k(\eta^1,\cdots,\eta^m)](t) = \delta$

ELSE

BEGIN

Find the largest integer $1\leq e\leq m$ such that $t\leq ek$

$[P_m^k(\eta^1,\cdots,\eta^m)](t) = \eta^e(t-(e-1)k)$

END

In the following sections, we will use the abbreviations $P_{e=1,m}^k(\eta^e)$ for $P_m^k(\eta^1,\cdots,\eta^m)$, and $P_m^k(\eta)$ for $P_m^k(\eta,\cdots,\eta)$. As will be seen later, the piping operator is very useful for the verification of pipelined operation of systolic networks.

The Spread Operator $\theta^s : \bar{R}_\delta \to \bar{R}_\delta$ defined by

$$[\theta^s \xi](t) = \begin{cases} \xi(\frac{t+s}{1+s}) & t = 1,(s+1)+1,2(s+1)+1,\cdots \\ \delta & \textit{otherwise} \end{cases}$$

Hence θ^s inserts s δ-elements between every two elements of ξ. With the sequence ξ of (2.1) we have, for example

$$\theta^2\xi = a_1.\delta.\delta.b_1.\delta.\delta.a_2.\delta.\delta.b_2.\cdots$$

Controlling the operation of systolic cells.

As mentioned earlier, the operators $A^{r,k,s}$, $M_r^{w1,\ldots,wn}$ and E_r^k can be used to model systolic cells, where the indices r,k and s control different timings as for instance, the reset times, the idle times and the active times of the cell. One way of monitoring these different timings in physical cells is by providing each cell with a separate circuit that generates reset and idle signals. On the other hand, timings may be monitored also by signals external to the cell. This external control method treats data and control signals in a uniform manner [24], and is especially preferred if the timing signals can be propagated in the network systolically.

The external control approach is equivalent with a redefinition of the operators where the control indices r,k and s are replaced by an additional control argument. For example, the expression "E_r^k ξ" used in modeling a periodic memory cell may be replaced by $E(\xi,\gamma)$, where the nonperiodic expansion operator E is defined by

$$[E(\xi,\gamma)](t) = \begin{cases} [E(\xi,\gamma)](t-1) & \text{if } \gamma(t)=0 \\ \xi(t) & \text{if } \gamma(t)=1 \end{cases}$$

and the control sequence γ controls the resetting of the memory element; that is

$$\gamma(t) = \begin{cases} 1 & t=r,\ r+k,\ r+2k,\cdots \\ 0 & \textit{otherwise} \end{cases}$$

It should be easy to verify that in all the networks presented in section 4, external control signals may be propagated in the network systolically.

3. Problem Definition and General Description of the System

The purpose of the systolic system presented in this report is to generate the finite element stiffness matrices H^e, $e=1,\cdots,m$, for a given finite element computation based on a given mesh on the domain Q of the problem. In order to simplify the design and the description of the system, we assume that all elements are of the same type, and hence that the number k of nodes per element is the same for all of them.

The class of problems to be considered is a fairly general class of 2-dimensional, stationary, elliptic boundary value problems [17]. The $(i,j)^{th}$ entry of the symmetric matrix H^e, corresponding to the element e, $1 \le e \le m$, is given by the general formula

$$H^e_{i,j} = \sum_{r,l=0}^{2} \int_{Q^e} a_{r,l} \, (D_r \varphi^e_i) \, (D_l \varphi^e_j) \, dx \, dy \qquad i,j=1,\cdots,k \qquad (3.1)$$

where $a_{r,l}$ are space dependent coefficients specified by the problem, and $D_1\varphi^e_i = \frac{\partial \varphi^e_i}{\partial x}$, $D_2\varphi^e_i = \frac{\partial \varphi^e_i}{\partial y}$, $D_0\varphi^e_i = \varphi^e_i$, and $\varphi^e_i(x,y)$ denote piece-wise smooth basis functions with the property that $\varphi^e_i(x,y)$ is equal to 1 at the i^{th} node of the finite element e and to 0 at any other node in Q. The integration in (3.1) is performed over the area Q^e of the finite element e.

Frequently in engineering applications the coefficients $a_{r,l}$, $r,l=0,1,2$ are approximated by piece-wise constant functions on each element, in which case we may rewrite (3.1) as

$$H^e_{i,j} = \sum_{r,l=0}^{2} a^e_{r,l} \int_{Q^e} (D_r \varphi^e_i) \, (D_l \varphi^e_j) \, dx \, dy \qquad (3.2)$$

where $a^e_{r,l}$ are constants on the element e. To evaluate these integrals, an iso-parametric transformation [9] is used to map the domain of each element Q^e into a standard element $\bar{Q}$ of the same type in another 2-dimensional space $(\bar{x},\bar{y})$.

namely

$$x = \sum_{i=1}^{k} \bar{\psi}_i(\bar{x},\bar{y})\ x_i \tag{3.3.a}$$

$$y = \sum_{i=1}^{k} \bar{\psi}_i(\bar{x},\bar{y})\ y_i \tag{3.3.b}$$

where $\bar{\psi}_i(\bar{x},\bar{y}) = \psi_i(x(\bar{x},\bar{y}),y(\bar{x},\bar{y}))$, $i=1,\cdots,k$, are the basis functions in the new space $(\bar{x},\bar{y})$.

The integrals in (3.2) are then evaluated numerically over $\bar{Q}$ instead of Q^e. Without entering into the mathematical details, we give only the final formula used to evaluate $H^e_{I,J}$:

$$H^e_{I,J} = \sum_{r,l=0}^{2} a^e_{r,I} \sum_{g=1}^{q} w_g\ \det{}^e(\bar{x}_g,\bar{y}_g)\ D_r\bar{\psi}_I(\bar{x}_g,\bar{y}_g)\ D_l\bar{\psi}_J(\bar{x}_g,\bar{y}_g) \tag{3.4}$$

where q is the order of the quadrature rule used in the numerical integration, $(\bar{x}_g,\bar{y}_g)$, $g=1,\cdots,q$ are the quadrature points with weights w_g and $\det^e(\bar{x},\bar{y})$ is the determinant of the Jacobian matrix J of the transformation $Q^e \to \bar{Q}$. From (3.3), this Jacobian is found to be

$$\begin{bmatrix} J_{1,1} & J_{1,2} \\ J_{2,1} & J_{2,2} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{k} \bar{D}_1\bar{\psi}_i(\bar{x},\bar{y})\ x_i & \sum_{i=1}^{k} \bar{D}_2\ \bar{\psi}_i(\bar{x},\bar{y})\ x_i \\ \sum_{i=1}^{k} \bar{D}_1\bar{\psi}_i(\bar{x},\bar{y})\ y_i & \sum_{i=1}^{k} \bar{D}_2\ \bar{\psi}_i(\bar{x},\bar{y})\ y_i \end{bmatrix}$$

Because of the regularity of the standard element $\bar{Q}$, we can easily write the formulas for $\bar{\psi}_i(\bar{x},\bar{y})$ and its derivatives $\bar{D}_1\bar{\psi}_i = \dfrac{\partial\bar{\psi}_i}{\partial\bar{x}}$ and $\bar{D}_2\bar{\psi}_i = \dfrac{\partial\bar{\psi}_i}{\partial\bar{y}}$. Then the derivatives $D_r\bar{\psi}_i$, $r=1,2$ and $i=1,\cdots,k$ used in (3.4) may be obtained from the transformation

$$\begin{bmatrix} D_1\bar{\psi}_i \\ D_2\bar{\psi}_i \end{bmatrix} = J^{-T} \begin{bmatrix} \bar{D}_1\bar{\psi}_i \\ \bar{D}_2\bar{\psi}_i \end{bmatrix} \tag{3.5}$$

where J^{-T} is the inverse of the transposed Jacobian matrix J^T.

It should be noted that the quadrature points and weights as well as the basis functions $\bar{\psi}_i$ and their derivatives $\bar{D}_1\bar{\psi}_i = \frac{\partial\bar{\psi}_i}{\partial\bar{x}}$ and $\bar{D}_2\bar{\psi}_i = \frac{\partial\bar{\psi}_i}{\partial\bar{y}}$ do not depend on the specific finite element that is to be processed. Hence, they may be computed at the quadrature points $(\bar{x}_g, \bar{y}_g)$, and pre-loaded into the system before it starts its operation which allows for their repeated use during the calculations of H^e for $e=1,\cdots,m$. On the other hand, the derivatives $D_1\bar{\psi}_i$ and $D_2\bar{\psi}_i$ in (3.3) have to be calculated for each element using (3.5).

We denote by $\nabla_i^0(g)$ the value of the basis function $\bar{\psi}_i(\bar{x}_g, \bar{y}_g)$ and by $\Delta_i^r(g)$ and $\nabla_i^r(g)$, $r=1,2$, its derivatives $\bar{D}_r\ \bar{\psi}_i(\bar{x}_g, \bar{y}_g)$ and $D_r\ \bar{\psi}_i(\bar{x}_g, \bar{y}_g)$, respectively. Suppose further that (x_i^e, y_i^e), $i=1,\cdots,k$, are the coordinates of the k nodes in the finite element e. Then the following algorithm computes the elemental stiffness matrices H^e for $e=1,\cdots,m$. (The steps N1 through N5 in the algorithm are partitioned in a manner needed for the description of our systolic system).

Algorithm ALG1

INPUTS

1) $(\nabla_i^0(g), \Delta_i^1(g), \Delta_i^2(g))$, $g=1,\cdots,q$ and $i=1,\cdots,k$

2) For each finite element $e=1,\cdots,m$

2.1) (x_i^e, y_i^e), $i=1,\cdots,k$

2.2) $a_{r,l}^e$, $r,l=0,1,2$

For each finite element $e=1,\cdots,m$ **DO**

N1) For each quadrature point $g=1,\cdots,q$ compute the Jacobian of the isoparametric transformation from

$$\begin{bmatrix} J_{1,1}(g) & J_{2,1}(g) \\ J_{1,2}(g) & J_{2,2}(g) \end{bmatrix} = \begin{bmatrix} \Delta_1^1(g) \cdots \Delta_k^1(g) \\ \Delta_1^2(g) \cdots \Delta_k^2(g) \end{bmatrix} \begin{bmatrix} x_1^e & y_1^e \\ . & . \\ x_k^e & y_k^e \end{bmatrix}$$

N2) For $g=1,\cdots,q$ compute the temporary quantities

$$\begin{bmatrix} T_1^1(g) \cdots T_k^1(g) \\ T_1^2(g) \cdots T_k^2(g) \end{bmatrix} = \begin{bmatrix} J_{2,2}(g) & -J_{2,1}(g) \\ -J_{1,2}(g) & J_{1,1}(g) \end{bmatrix} \begin{bmatrix} \Delta_1^1(g) \cdots \Delta_k^1(g) \\ \Delta_1^2(g) \cdots \Delta_k^2(g) \end{bmatrix}$$

N3) For $g=1,\cdots,q$ DO

N3.1) $\det(g) = J_{1,1}(g)\, J_{2,2}(g) - J_{1,2}(g)\, J_{2,1}(g)$

N3.2) $\nabla_l^r(g) = \frac{1}{det(g)}\, T_l^r(g), \qquad r=1,2,\ l=1,\cdots,k$

N3.3) $\overline{\nabla}_l^r(g) = w_g\ det(g)\ \nabla_l^r(g), \qquad r=0,1,2,\ l=1,\cdots,k$

N4) For $l=1,\cdots,k$ compute the approximate integrals

N4.1) For $j=1,\cdots,l-1$

$$Y_{l,j}^{r,l} = \sum_{g=1}^{q} \overline{\nabla}_l^r(g)\ \nabla_j^l(g) \qquad r,l=0,1,2$$

N4.2) $$Y_{l,l}^{r,l} = \sum_{g=1}^{q} \overline{\nabla}_l^r(g)\ \nabla_l^l(g) \qquad r=0,1,2,\ l=0,\cdots,r$$

N5) For $l=1,\cdots,k$ DO

N5.1) For $j=1,\cdots,l-1$

$$H_{l,j}^e = \sum_{r=0}^{2} \sum_{l=0}^{2} a_{r,l}^e\ Y_{l,j}^{r,l}$$

N5.2) $$H_{l,l}^e = 2 \sum_{r=0}^{2} \sum_{l=r}^{2} (c_{r,l}\ a_{r,l}^e)\ Y_{l,l}^{r,l}$$

where $c_{r,l}$ equals to 1 if $r \neq l$, and to 0.5 if $r=l$.

Figure 3.1 shows a block diagram of the systolic system that executes this algorithm. It consists of a local memory LM to store the pre-loaded values of $\nabla_l^0(g)$, $\Delta_l^1(g)$ and $\Delta_l^2(g)$, $g=1,\cdots,q$, and five systolic subnetworks N1$\cdots$N5 that are arranged in a cascade such that the output of a sub-network is an input for a following sub-network. Each sub-network is designed to perform the computa-

tion in the corresponding step in ALG1.

In order to compute the matrix H^e for a certain element e, the coordinates of the nodes (x_i^e, y_i^e), $i=1,\cdots,k$, and the coefficients $a_{r,l}^e$, $r,l=0,1,2$, for that element, are fed to the system via subnetworks N1 and N5, respectively. The entries $H_{i,j}^e$, $i=1,\cdots,k$, $j=1,\cdots,i$, of the symmetric matrix H^e are then obtained from the sub-network N5 after a delay period of (q+3k+16) time units, where a time unit is the maximum time needed by any computational cell in the system to perform its operation. This is basically the time required to perform a Multiply/Add operation, or a division whichever is larger.

Although this is a noticeble speedup of order k over the serial execution of algorithm ALG1, the real advantage of the system lies in the possibility of pipelining the computations of the stiffness matrices for $e=1,\cdots,m$, and of obtaining one matrix every 3k time units. Of course, we also obtain the advantage of a non-conflicting and smooth data flow in the system which greatly reduces the memory fetch times.

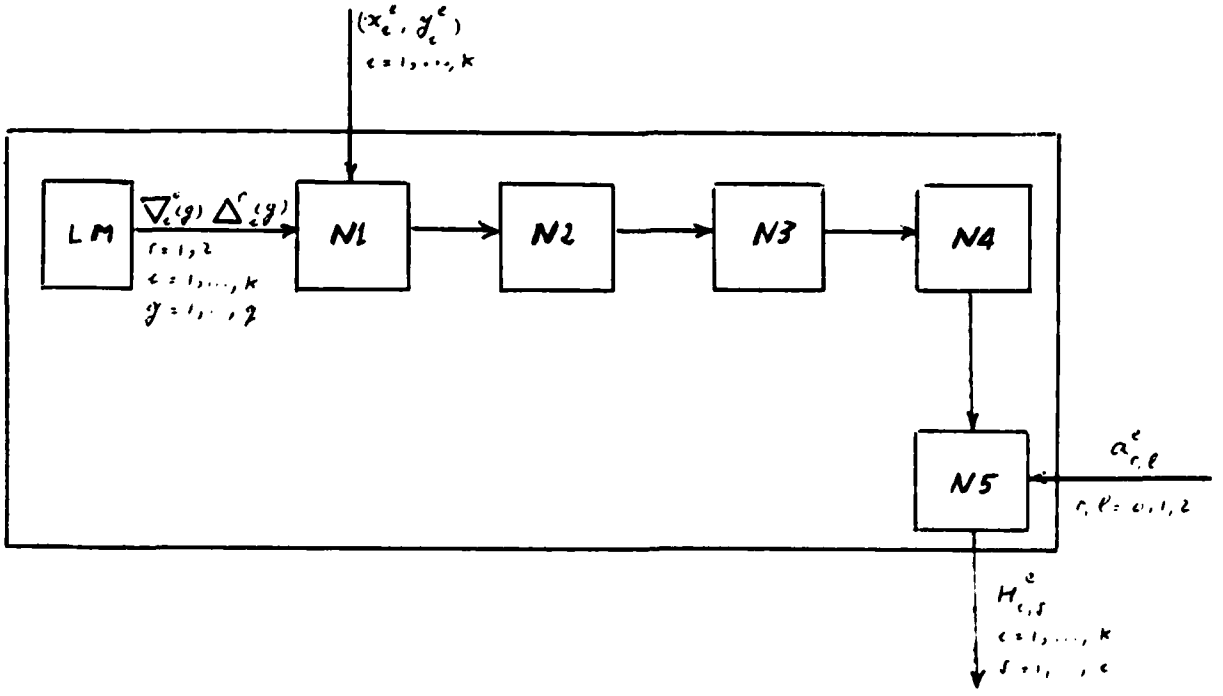


Figure 3.1 - A general block diagram of the system.

Finally, note that we have assumed that there is only one variable at each node, that is the degree of freedom per node is unity. In the general case of d degrees of freedom per node, the constants $a^{e}_{r,l}$, $r,l=0,1,2$ are $d \times d$ matrices, and consequently each entry $H^{e}_{i,j}$, $i,j=1,\cdots,k$, in the elemental stiffness matrix H^{e} is a $d \times d$ submatrix. To compute the d^2 elements of $H^{e}_{i,j}$ without slowing down the system, we replace the subnetwork N5 by d^2 identical subnetworks, each of which generates the corresponding entry in the submatrix $H^{e}_{i,j}$ when provided with the appropriate entry in the $d \times d$ matrix $a^{e}_{r,l}$.

4.Formal Description of the System's Components

In this section, we describe the architecture of the five subnetworks N1,···N5, that execute the corresponding steps in algorithm ALG1. Moreover, we will derive the I/O description of the individual subnetworks and prove that the system generates an elemental stiffness matrix if appropriate input data are provided.

It should be clear that alternate designs for the components of the system may be given. However, one advantage of the system described in this report is its flexibility in the sense that only minor modifications are needed to use the system for different values of k (element type) and q (quadrature formula). Moreover, our primary goal is to demonstrate the effectiveness of the formal model for a precise specification and verification of systolic networks with computational cells more complicated than those of the simple Multiply/Add type.

4.1. The Subnetwork N1

The graph of the systolic network N1 is composed of 2q internal nodes as shown in Figure 4.1; each node is labeled by two integers (i,g) i=1,2 and g=1,....q, where q is the number of points used in the numerical integration (3.3). The graph also shows the color assigned to each edge, namely r, p or z.

Each interior node (i,g) represents a computational cell whose operation is described by the causal relations

$$\zeta_{i,g+1} = \Omega\ \zeta_{i,g} \tag{4.1.a}$$

$$\rho_{i+1,g} = \Omega\ \rho_{i,g} \tag{4.1.b}$$

$$\pi_{i+1,g} = \Omega^{s}\ M_{g+i-1}^{3k-2,1,1}\ (\ \pi_{i,g}\ ,\ \lambda_{i,g}\ ,\ \bar{\lambda}_{i,g}\ ) \tag{4.1.c}$$

where s=1 for i=2 and s=3 for i=1, and

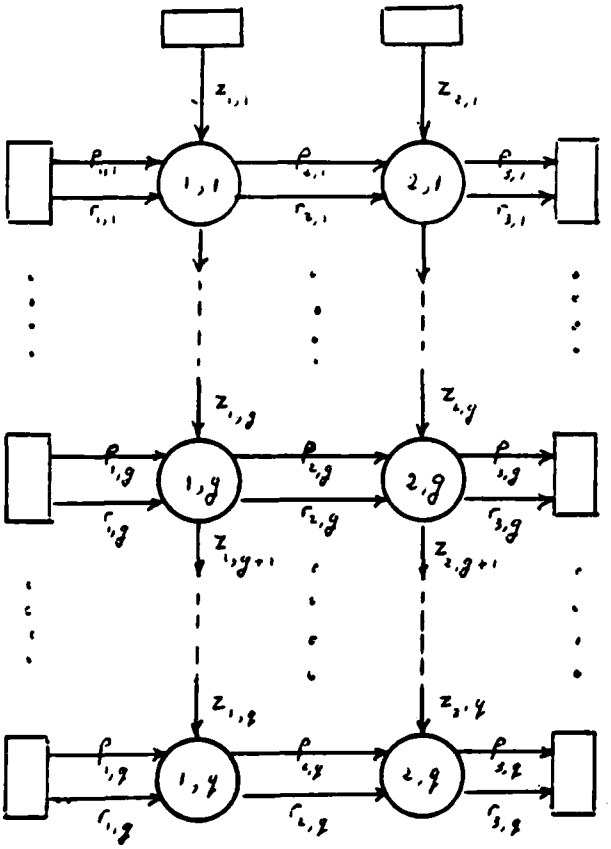


Figure 4.1 - The graph for N1.

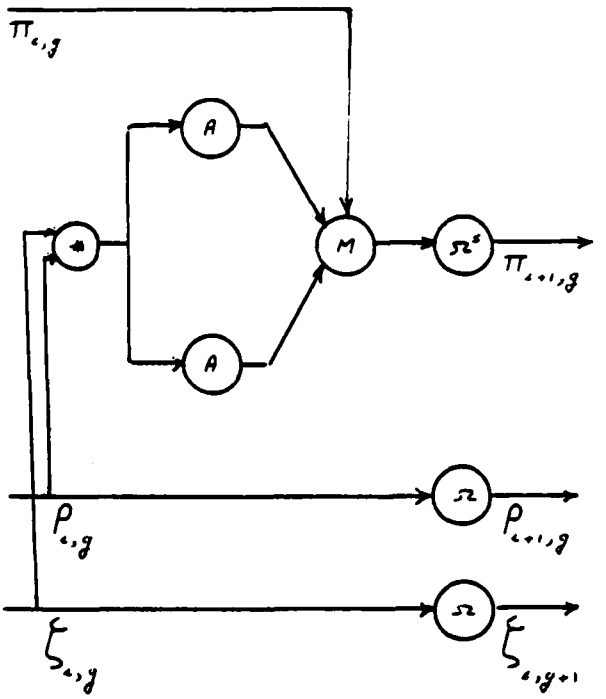


Figure 4.2 - The structure of a typical cell (i,g) in N1.

$$\lambda_{i,g} = A^{g+i,k,3}\,[\rho_{i,g} * \zeta_{i,g}] \qquad (4.2.a)$$

$$\bar{\lambda}_{i,g} = A^{g+i+1,k,3}\,[\rho_{i,g} * \zeta_{i,g}] \qquad (4.2.b)$$

The graph in Figure 4.1 and equations (4.1), (4.2) specify N1 completely. In order to analyze the internal structure of each cell (i,g) more closely, we first note that equations (4.2.a/b) indicate that a cell should contain a multiplier and two accumulators (see Figure 4.2). The accumulators start operating at times g+i and g+i+1, respectively, accumulate the output of the multiplier every third time unit and are reset to zero every 3k time units. The content of these accumulators at consecutive time units is expressed by the sequences $\lambda_{i,g}$ and $\bar{\lambda}_{i,g}$. As is clear from equation (4.1.c), each cell contains also a multiplexer that starts

operating at time g+i-1 and multiplexes the input $\pi_{i,g}$ and the contents of the accumulators with a time ratio of 3k-2:1:1. the delay element Ω^s is introduced in Figure 4.2 under the assumption that the elements '*', A and M do not consume any time. In practical implementations however, these elements do consume some time and consequently the element labeled Ω^s has the function of a synchronizer rather than a latch.

After having described the architecture of the network, we prove the following proposition that gives the I/O description for N1, which is an explicit relation between the network output sequences $\rho_{3,g}$, $\pi_{3,g}$, $g=1,\cdots q$, and the network input sequences $\zeta_{i,1}$, $\pi_{1,g}$, $\rho_{1,g}$, $i=1,2$, $g=1,\cdots q$.

Proposition N1.1 : I/O description of the network N1. For $g=1,\cdots,q$, the following relations hold:

$$\rho_{3,g} = \Omega^2 \rho_{1,g} \tag{4.3.a}$$

$$\pi_{3,g} = \Omega M_{g+3}^{3k-4,1,1,1,1} (\Omega^3\pi_{1,g} \cdot \lambda_{2,g} \cdot \bar{\lambda}_{2,g} \cdot \Omega^3\lambda_{1,g} \cdot \Omega^3\bar{\lambda}_{1,g}) \tag{4.3.b}$$

where

$$\lambda_{i,g} = A^{g+i,k,3} [\Omega^{i-1} \rho_{1,g} * \Omega^{g-1}\zeta_{i,1}] \tag{4.3.c}$$

$$\bar{\lambda}_{i,g} = A^{g+i+1,k,3} [\Omega^{i-1} \rho_{1,g} * \Omega^{g-1} \zeta_{i,1}] \tag{4.3.d}$$

Proof: To prove (4.3.b), we first note that (4.1.a/b) have the solutions

$$\zeta_{i,g} = \Omega^{g-1} \zeta_{i,1} \tag{4.4.a}$$

$$\rho_{i,g} = \Omega^{i-1} \rho_{i,g} \tag{4.4.b}$$

Then from (4.1.c) we obtain for $g=1,\cdots,q$ that

$$\pi_{3,g} = \Omega M_{g+1}^{3k-2,1,1} (\pi_{2,g} \cdot \lambda_{2,g} \cdot \bar{\lambda}_{2,g})$$

$$= \Omega M_{g+1}^{3k-2,1,1} (\Omega^3 M_g^{3k-2,1,1} (\pi_{1,g},\lambda_{1,g},\bar{\lambda}_{1,g}) \cdot \lambda_{2,g} \cdot \bar{\lambda}_{2,g})$$

where $\lambda_{l,g}$ and $\bar{\lambda}_{l,g}$ are as given in (4.3.c) and (4.3.d), respectively. Using property P5 from the Appendix, this may be rewritten as

$$\pi_{3,g} = \Omega\, M_{g+1}^{3k-2,1,1}\,(M_{g+3}^{3k-2,1,1}(\Omega^3\pi_{1,g}\,.\,\Omega^3\lambda_{1,g}\,.\,\Omega^3\bar{\lambda}_{1,g})\,.\,\lambda_{2,g}\,.\,\bar{\lambda}_{2,g})$$

Finally, we obtain (4.3.b) by applying property P13 from the Appendix. Equation (4.3.a) results directly from (4.4.b).■

In order to perform the calculations in step N1 of ALG1 for a certain finite element e, $1 \le e \le m$, the input sequences must be described by

$$\pi_{1,g} = \delta^* \qquad (4.5.a)$$

$$\zeta_{l,1} = \Omega^{l-1}\, E_1^3\,[\Theta^2\ \xi_l^e] \qquad l=1,2 \qquad (4.5.b)$$

$$\rho_{1,g} = \Omega^{g-1}\, P_2^{3k}(M_1^{1,1,1}(\Theta^2\psi_{g,0}\,.\,\Omega\Theta^2\varphi_{g,1}\,.\,\Omega^2\Theta^2\varphi_{g,2})) \qquad g=1,\cdots q \qquad (4.5.c)$$

where

$$T(\xi_l^e) = T(\psi_{g,0}) = T(\varphi_{g,1}) = T(\varphi_{g,2}) = k \qquad (4.5.d)$$

and

$$\psi_{g,0}(t) = \nabla_t^0(g)$$

$$\varphi_{g,1}(t) = \Delta_t^1(g)$$

$$\varphi_{g,2}(t) = \Delta_t^2(g)$$

$$\xi_l^e(t) = \begin{cases} y_t^e & \text{if } l=1 \\ x_t^e & \text{if } l=2 \end{cases}$$

In other words, ξ_1^e and ξ_2^e contain the coordinates of the nodes in the finite element e, and $\psi_{g,0}$, $\varphi_{g,1}$ and $\varphi_{g,2}$ contain the shape functions and their derivatives. A pictorial representation of these input sequences in the case k=3 and q=3 is provided in figure 4.9 using a time diagram in which the elements of the different sequences at consecutive time units are displayed.

Proposition N1.2 : With the inputs (4.5), the outputs of the network N1 are described by

$$\rho_{3,g} = \Omega^{g+1}\ P_2^{3k}(M_1^{1,1,1}(\theta^2\psi_{g,0} \cdot \Omega\theta^2\varphi_{g,1} \cdot \Omega^2\theta^2\varphi_{g,2})) \qquad g=1,\cdots,q \quad (4.6.a)$$

$$\pi_{3,g} = \Omega^{g+3k-1}\ \beta^e \qquad g=1,\cdots,q \quad (4.6.b)$$

where $T(\beta^e) = 4$ and $\beta^e(t) = J^e_{1,1}(g)$, $J^e_{1,2}(g)$, $J^e_{2,1}(g)$ and $J^e_{2,2}(g)$ for t=1, 2, 3 and 4, respectively.

Proof : The proof of (4.6.a) follows directly from (4.3.a). To prove (4.6.b), we first note that the operator P_2^{3k} in (4.5.c) indicates that the first 3k elements of the argument are repeated twice in $\rho_{1,g}$. This repetition is only necessary for the operation of the subnetwork N2, and will not be considered here. Hence, we will replace the last 3k elements of the repetition by don't care elements, which reduces (4.5.c) to

$$\rho_{1,g} = \Omega^{g-1}\ M_1^{1,1,1}(\theta^2\ \psi_{g,0} \cdot \Omega\ \theta^2\ \varphi_{g,1} \cdot \Omega^2\ \theta^2\ \varphi_{g,2}) \quad (4.5.e)$$

Now substitution of the input sequences (4.5.a/b/e) into the I/O description (4.3.b) results in

$$\pi_{3,g} = \Omega\ M_{g+3}^{3k-4,1,1,1,1}(\delta^* \cdot \lambda_{2,g} \cdot \bar{\lambda}_{2,g} \cdot \Omega^3\lambda_{1,g} \cdot \Omega^3\bar{\lambda}_{1,g}). \quad (4.7.a)$$

Here by (4.3.c) and the definition of the E operator and properties P1 and P7 we find that

$$\lambda_{i,g} = A^{g+i,k,3}\ [\Omega^{g+i-2}\ M_1^{1,1,1}(\theta^2\psi_{g,0} \cdot \Omega\theta^2\varphi_{g,1} \cdot \Omega^2\theta^2\varphi_{g,2}) * \Omega^{g+i-2}\ E_1^3\theta^2\xi_i^e]$$

$$= \Omega^{g+i-2}\ A^{2,k,3}\ M_1^{1,1,1}(\theta^2[\psi_{g,0} * \xi_i^e] \cdot \Omega\theta^2[\varphi_{g,1} * \xi_i^e] \cdot \Omega^2\theta^2[\varphi_{g,2} * \xi_i^e])$$

and by P14 that

$$\lambda_{i,g} = \Omega^{g+i-1}\ A^{1,k,3}\ \theta^2\ [\varphi_{g,1} * \xi_i^e] \quad (4.7.b)$$

Similarly, we can show that

$$\bar{\lambda}_{i,g} = \Omega^{g+i}\ A^{1,k,3}\ \theta^2\ [\varphi_{g,2} * \xi_i^e] \quad (4.7.c)$$

For a further simplification of the equations (4.7.a), we consider the definition of the multiplexer operator with the restrictions (4.5.d) on the involved sequences. This gives for $g=1,\cdots,q$

$$\pi_{3,g} = \Omega^{g+3k-1}\ \beta^{e}$$

where $T(\beta^{e}) = 4$ and

$$\beta^{e}(t) = \begin{cases} \lambda_{2,g}(g+3k-1) & \text{for } t=1 \\ \overline{\lambda}_{2,g}(g+3k) & \text{for } t=2 \\ \lambda_{1,g}(g+3k-2) & \text{for } t=3 \\ \overline{\lambda}_{1,g}(g+3k-1) & \text{for } t=4 \end{cases}$$

Moreover, from (4.7.b), P11 and the definitions of the shift and the spread operators, we obtain that

$$\begin{aligned} \lambda_{2,g}(g+3k-1) &= [\Omega^{g+1}\ \Theta^{2}\ A^{1,k,1}\ [\varphi_{g,1} * \xi_{2}^{e}\ ]](g+3k-1) \\ &= [A^{1,k,1}[\varphi_{g,1} * \xi_{2}^{e}\ ]](k) \\ &= \sum_{j=1}^{k} \varphi_{g,1}(j)\ \xi_{2}^{e}(j) = \sum_{j=1}^{k} \Delta_{j}^{1}(g)\ x_{j}^{e} = J_{1,1}^{e}(g) \end{aligned}$$

where $J_{1,1}^{e}(g)$ is specified in algorithm ALG1.

By a similar argument, it can be shown that $\beta^{e}(2)$, $\beta^{e}(3)$ and $\beta^{e}(4)$ are equal to $J_{1,2}^{e}(g)$, $J_{2,1}^{e}(g)$ and $J_{2,2}^{e}(g)$, respectively, which proves the proposition and shows that the network performs successfully the calculations in step N1 of ALG1 for one finite element e.■

4.2. The Subnetwork N2

The graph of the subnetwork N2 is composed of q identical rows $g=1,\cdots,q$ (see Figure 4.3) where each row consists of three interior nodes (i,g), $i=3,4,5$. The edges are given the colors p,r,s and $\overline{s}$ as shown in the figure.

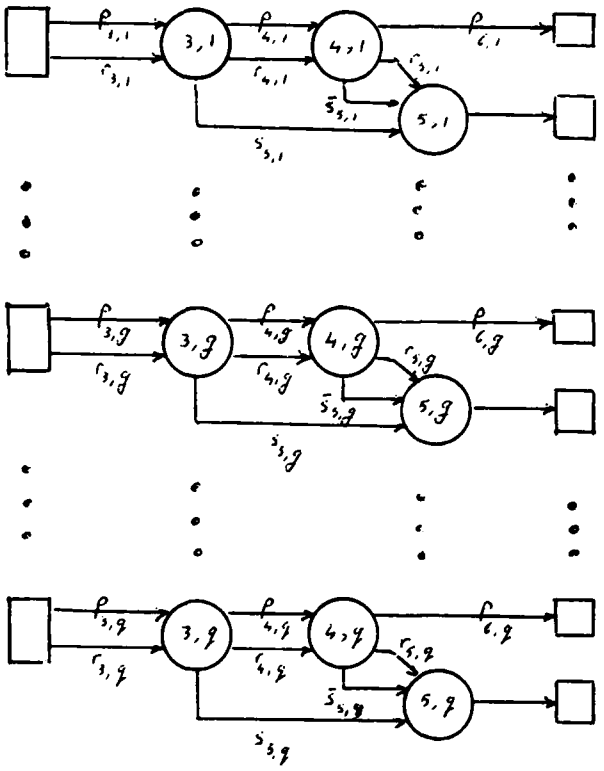


Figure 4.3 - The graph for N2.

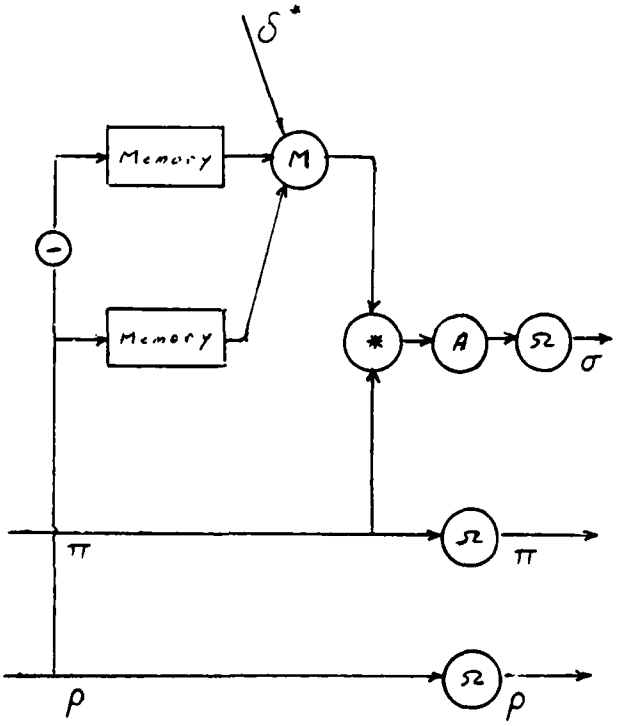


Figure 4.4

For a given row g, $1 \leq g \leq q$, the computation of a cell may be described as follows:

For cells (3,g)

$$\pi_{4,g} = \Omega^3 \; \pi_{3,g} \qquad \rho_{4,g} = \Omega \; \rho_{3,g}$$

$$\sigma_{5,g} = \Omega \; A^{g,3,1} \; [\, \rho_{3,g} \; * \; M_g^{1,1,1} \; (\; E_{g+3}^{3k} \; \pi_{3,g} \; . \; E_{g+2}^{3k} \; [-\pi_{3,g}] \; . \; \delta^* \;)] \qquad (4.8.a)$$

For cells (4,g)

$$\pi_{6,g} = \Omega \; \pi_{4,g} \qquad \rho_{5,g} = \Omega \; \rho_{4,g}$$

$$\bar{\sigma}_{5,g} = \Omega \; A^{g+1,3,1} \; [\rho_{4,g} \; * \; M_{g+1}^{1,1,1} \; (\; E_{g+4}^{3k} \; [-\pi_{4,g}] \; . \; E_{g+3}^{3k} \; \pi_{4,g} \; . \; \delta^*)] \qquad (4.8.b)$$

For cells (5,g)

$$\rho_{7,g} = \Omega \; M_{g+1}^{1,1,1} \; (\rho_{5,g} \; . \; \sigma_{5,g} \; . \; \bar{\sigma}_{5,g}) \qquad (4.8.c)$$

From the above specifications, it is clear that cells (3,g) and (4,g) have identical structure (see Figure 4.4) and differ only in the reset times of their accumulators, multiplexers and memories. To reset these elements at the proper time, external reset signal can be propagated in the network as explained in Section 2.

Proposition N2.1 : The network I/O description of N2 is given by

$$\pi_{6,g} = \Omega^4 \, \pi_{3,g} \qquad g=1,\cdots,q \qquad (4.9.a)$$

$$\rho_{7,g} = \Omega \, M_{g+1}^{1,1,1} \, (\Omega^2 \, \rho_{3,g} \, , \, \lambda_{3,g} \, , \, \lambda_{4,g}) \qquad g=1,\cdots,q \qquad (4.9.b)$$

where

$$\lambda_{3,g} = \Omega^2 \, [\rho_{3,g} \, * \, E_{g+3}^{3k} \, \pi_{3,g}] - \Omega \, [\rho_{3,g} \, * \, E_{g+2}^{3k} \, \pi_{3,g}] \qquad (4.9.c)$$

$$\lambda_{4,g} = \Omega \, [\Omega \, \rho_{3,g} \, * \, E_{g+3}^{3k} \, \Omega^3 \pi_{3,g}] - \Omega^2 \, [\Omega \, \rho_{3,g} \, * \, E_{g+4}^{3k} \, \Omega^3 \, \pi_{3,g}] \qquad (4.9.d)$$

Proof : Equation (4.9.a) is trivial. In order to prove (4.9.b), we begin by applying property P1.3 to equation (4.8.a):

$$\sigma_{5,g} = \Omega \, A^{g,3,1} \, M_g^{1,1,1} (\rho_{3,g} \, * \, E_{g+3}^{3k} \, \pi_{3,g} \, , \, \rho_{3,g} \, * \, E_{g+2}^{3k} \, [-\pi_{3,g}] \, , \, \delta^*)$$

Then, we apply property P14 and use δ^* to replace sequences whose values are irrelevant to our analysis. This gives

$$\sigma_{5,g} = \Omega \, M_g^{1,1,1} (\delta^* \, , \, \Omega \, [\rho_{3,g} \, * \, E_{g+3}^{3k} \, \pi_{3,g}] - [\rho_{3,g} \, * \, E_{g+2}^{3k} \, \pi_{3,g}] \, , \, \delta^*)$$

which from P5 may be written as

$$\sigma_{5,g} = M_{g+1}^{1,1,1} \, (\delta^* \, , \, \lambda_{3,g} \, , \, \delta^*) \qquad (4.10.a)$$

where $\lambda_{3,g}$ is as described by (4.9.c). Similarly from (4.8.b) we obtain

$$\overline{\sigma}_{5,g} = M_{g+1}^{1,1,1} \, (\delta^* \, , \, \delta^* \, , \, \lambda_{4,g}) \qquad (4.10.b)$$

where $\lambda_{4,g}$ is as described in (4.9.d). Finally, substituting (4.10) in (4.8.c), and using P13 we obtain (4.9.b), which completes the proof.■

The input links of N2 are directly connected to the outputs of N1, and hence the input sequences $\pi_{3,g}$ and $\rho_{3,g}$ $g=1,\cdots,q$ are described by the formulas (4.6).

Proposition N2.2 : If the inputs to N2 are given by (4.6), then its outputs may be described by

$$\pi_{6,g} = \Omega^{g+3k+3}\ \beta^{e} \qquad g=1,\cdots,q \quad (4.11.a)$$

$$\rho_{7,g} = \Omega^{g+3k+3}\ M_1^{1,1,1}(\Theta\psi_{g,0}\ .\ \Omega\Theta^2\psi_{g,1}\ .\ \Omega^2\Theta^2\psi_{g,2}) \qquad g=1,\cdots,q \quad (4.11.b)$$

where $T(\psi_{g,1})=T(\psi_{g,2})=k$ and $\psi_{g,1}(t)=T_t^1(g)$, $\psi_{g,2}(t)=T_t^2(g)$ with $T_t^1(g)$ and $T_t^2(g)$ as specified in algorithm ALG1.

Proof : The proof of (4.11.a) is trivial. In order to prove (4.11.b), we will ignore the value of the first 3k+g+1 elements in the input $\rho_{3,g}$, and hence rewrite (4.6.a) as

$$\rho_{3,g} = \Omega^{g+3k+1}\ M_1^{1,1,1}(\Theta^2\ \psi_{g,0}\ .\ \Omega\Theta^2\varphi_{g,1}\ .\ \Omega^2\Theta^2\varphi_{g,2}) \qquad (4.6.c)$$

In order to find the output sequences $\rho_{7,g}$, we obtain an explicit description for $\lambda_{3,g}$ and $\lambda_{4,g}$ by substituting the input sequences into (4.9.c/d). Indeed, from (4.6.b/c) it follows that

$$\rho_{3,g} * E_{g+3}^{3k}\ \pi_{3,g} = \Omega^{g+3k+1}\ M_1^{1,1,1}(\Theta^2\psi_{g,0}\ .\ \Omega\Theta^2\varphi_{g,1}\ .\ \Omega^2\Theta^2\varphi_{g,2}) * E_{g+3}^{3k}\ \Omega^{g+3k-1}\ \beta^{e}$$

We then interchange the shift and expand operators using P6 and apply P17 to get

$$\rho_{3,g} * E_{g+3}^{3k}\ \pi_{3,g} = \Omega^{g+3k-1}\ [\Omega^2\ M_1^{1,1,1}(\Theta^2\psi_{g,0}\ .\ \Omega\Theta^2\varphi_{g,1}\ .\ \Omega^2\Theta^2\varphi_{g,2}) * E_4^{3k}\ \beta^{e}]$$

$$= \Omega^{g+3k-1}\ [\Omega^3\Omega^{-1}\ M_1^{1,1,1}(\Theta^2\psi_{g,0}\ .\ \Omega\Theta^2\varphi_{g,1}\ .\ \Omega^2\Theta^2\varphi_{g,2})\ .\ \beta^{e}(4)]$$

$$= \Omega^{g+3k+1}\ M_1^{1,1,1}(\delta^*\ .\ \Omega\Theta^2\ [J_{2,2}^{e}(g)\ .\ \varphi_{g,1}]\ .\ \delta^*)$$

where, as usual, the sequences irrelevant in this context were replaced by δ^*. Similarly, we obtain

$$\rho_{3,g} * E_{g+2}^{3k} \pi_{3,g} = \Omega^{g+3k+1} M_1^{1,1,1}(\delta^* . \delta^* . \Omega^2\theta^2 [J_{2,1}^{\theta}(g) . \varphi_{g,2}])$$

and thus derive from (4.9.c) that

$$\lambda_{3,g} = \Omega^{g+3k+3} M_1^{1,1,1}(\delta^* . \Omega\theta^2 [J_{2,2}^{\theta}(g) . \varphi_{g,1}] . \delta^*) - \Omega^{g+3k+2} M_1^{1,1,1}(\delta^* . \delta^* . \Omega^2\theta^2 [J_{2,1}^{\theta}(g) . \varphi_{g,2}])$$

$$= \Omega^{g+3k+3} [M_1^{1,1,1}(\delta^* . \Omega\theta^2 [J_{2,2}^{\theta}(g) . \varphi_{g,1}] . \delta^*) - M_1^{1,1,1}(\delta^* . \Omega\theta^2 [J_{2,1}^{\theta}(g) . \varphi_{g,2}] . \delta^*)]$$

$$= \Omega^{g+3k+3} M_1^{1,1,1}(\delta^* . \Omega\theta^2 [J_{2,2}^{\theta}(g) . \varphi_{g,1} - J_{2,1}^{\theta}(g) . \varphi_{g,2}] . \delta^*)$$

By a similar analysis it follows that

$$\lambda_{4,g} = \Omega^{g+3k+3} M_1^{1,1,1}(\delta^* . \delta^* . \Omega^2\theta^2 [J_{1,1}^{\theta}(g) . \varphi_{g,2} - J_{1,2}^{\theta}(g) . \varphi_{g,1}])$$

Finally, we substitute into (4.9.b) the computed values for $\lambda_{3,g}$ and $\lambda_{4,g}$ together with the input sequence $\rho_{3,g}$ and apply properties P5 and P13 to obtain

$$\rho_{7,g} = \Omega^{g+3k+3} M_1^{1,1,1}(\theta^2\psi_{g,0} . \Omega\theta^2\psi_{g,1} . \Omega^2\theta^2\psi_{g,2}) \qquad g=1,\cdots,q$$

where

$$\psi_{g,1}(t) = J_{2,2}^{\theta}(g) . \varphi_{g,1}(t) - J_{2,1}^{\theta}(g) . \varphi_{g,2}(t)$$
$$= J_{2,2}^{\theta}(g)\ \Delta_t^1(g) - J_{2,1}^{\theta}(g)\ \Delta_t^2(g) = T_t^1(g)$$
$$\psi_{g,2}(t) = J_{1,1}^{\theta}(g)\ \varphi_{g,2}(t) - J_{1,2}^{\theta}(g)\ \varphi_{g,1}(t) = T_t^2(g)$$

This proves explicitly that the output sequences $\rho_{7,g}$ contain the results of step N2 in ALG1.■

4.3. The Subnetwork N3

As in the case of N2, the subnetwork N3 is composed of q independent, identical rows. Each row performs the calculation corresponding to step N3 in ALG1 for a certain value of g, $1 \le g \le q$. Due to the variety of possible designs and to the simplicity of the network, we will not describe N3 in any detail. Instead, we will assume that, with the inputs described by proposition N2.2, N3 takes five time units to complete its computation and to produce for any g, $1 \le g \le q$ the outputs

$$\pi_{9,g} = \Omega^{g+3k+8} M_1^{1,1,1}(\Theta^2 \bar{\nu}_{g,0} \cdot \Omega\Theta^2 \bar{\nu}_{g,1} \cdot \Omega^2\Theta^2 \bar{\nu}_{g,2}) \quad (4.12.a)$$

$$\rho_{9,g} = \Omega^{g+3k+8} M_1^{1,1,1}(\Theta^2 \nu_{g,0} \cdot \Omega\Theta^2 \nu_{g,1} \cdot \Omega^2\Theta^2 \nu_{g,2}) \quad (4.12.b)$$

where $\nu_{g,r}(t) = \nabla_t^r(g)$, $\bar{\nu}_{g,r}(t) = \bar{\nabla}_t^r(g)$, and the values of $\nabla_t^r(g)$ and $\bar{\nabla}_t^r(g)$ are as given in step N3 of ALG1.

4.4. The subnetwork N4

In this subsection, we describe a network that completes the numerical integration by computing the quantities $Y_{i,j}^{r,l} = \sum_{g=1}^{q} \bar{\nabla}_i^r \nabla_j^l$ for the ranges of the indices in the corresponding step of ALG1. The subnetwork is described by the graph in Figure 4.5 and the node I/O descriptions of a typical interior node (i,g) $i=9,\cdots,8+3k$, $g=1,\cdots,q$, are given by the causal relations

$$\pi_{i+1,g} = \Omega^2 \pi_{i,g} \quad (4.13.a)$$

$$\rho_{i+1,g} = \Omega \rho_{i,g} \quad (4.13.b)$$

$$\zeta_{i,g+1} = \Omega [\zeta_{i,g} + \pi_{i,g} * \rho_{i,g}] \quad (4.13.c)$$

As this description shows, each cell latches the p and r data streams by two and one time units, respectively. It also performs a Multiply/Add operation

and puts the result on the z output link.

Proposition N4.1 : The I/O description for N4 is given by

$$\zeta_{i,q+1} = \Omega^{q}\,\zeta_{i,1} + \sum_{g=1}^{q} \Omega^{i-8+q-g}\,[\Omega^{i-9}\,\pi_{9,g} * \rho_{9,g}] \qquad i=9,\cdots,8+3k \qquad (4.14)$$

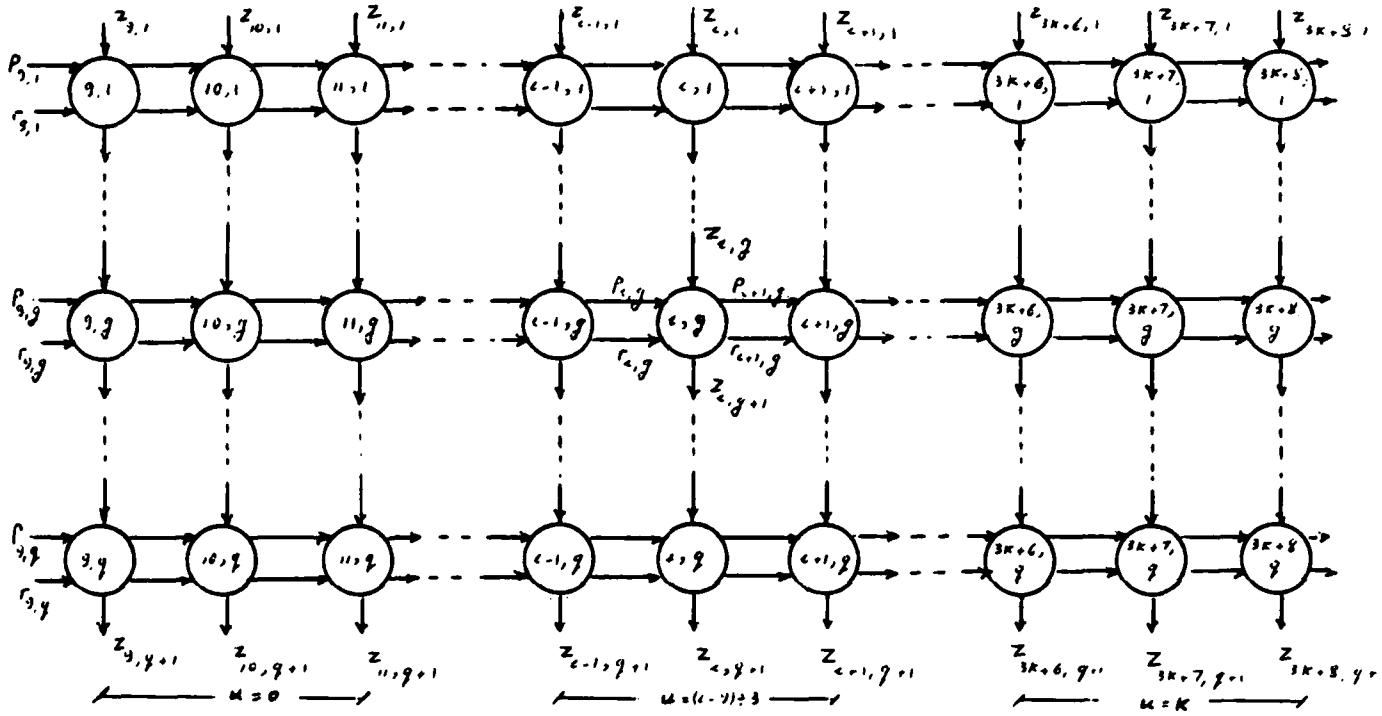


Figure 4.5 - The graph for N4.

Proof : To prove this proposition, we first write the solutions of (4.13.a) and (4.13.b) in the form

$$\pi_{i,g} = \Omega^{2(i-9)}\,\pi_{9,g} \qquad i=9,\cdots,8+3k, \quad g=1,\cdots,q$$

$$\rho_{i,g} = \Omega^{(i-9)}\,\rho_{9,g} \qquad i=9,\cdots,8+3k, \quad g=1,\cdots,q$$

and then substitute them into (4.13.c). This gives

$$\zeta_{i,g+1} = \Omega\,[\zeta_{i,g} + \Omega^{2(i-9)}\,\pi_{9,g} * \Omega^{i-9}\,\rho_{9,g}] \qquad (4.15)$$

By Lemma 1 in the Appendix, the solution of (4.15) for a fixed i, $9 \le i \le 8+3k$ is then found to be identical to equation (4.14), which completes the proof.■

In order to perform the computation in step N5 of ALG1, the input links $z_{i,1}$, $i=9,\cdots,8+3k$ should be permanently set to zero. That is to say, in the I/O description (4.14) we must set $\zeta_{i,1} = \iota$, where ι denotes the zero sequence of section 2. With this, we rewrite (4.14) as

$$\zeta_{i,q+1} = \sum_{g=1}^{q} \Omega^{i-8+q-g} [\Omega^{i-9} \pi_{9,g} * \rho_{9,g}] \qquad i=9,\cdots,8+3k \qquad (4.16)$$

The next step in the verification of N4 is the calculation of the output sequences for a specific form of the input sequences $\pi_{9,g}$ and $\rho_{9,g}$. As Figure 3.1 shows, the outputs of N3 are inputs to N4, and hence $\pi_{9,g}$ and $\rho_{9,g}$ are described by the formulas (4.12). Unfortunately, it is not at all simple to find an explicit description of the output sequences for this specific input. In order to simplify the equations, we will replace the index i, $9\leq i\leq 8+3k$ by $i=9+3u+v$, where the indices u and v vary in the ranges $0\leq u\leq k-1$ and $0\leq v\leq 2$. More descriptively, we divide the 3k columns of N4 into k groups of 3 columns each. Thus, we rewrite the network description (4.16) as

$$\zeta_{u,v,q+1} = \sum_{g=1}^{q} \Omega^{3u+v+1+q-g} [\Omega^{3u+v} \pi_{9,g} * \rho_{9,g}] \qquad (4.17)$$

Proposition N4.2 : With the inputs described by (4.12), the network N4 has the following output. For $0\leq u\leq k-1$ and $0\leq v\leq 2$

$$\zeta_{u,v,q+1} = \Omega^{2(3u+v)+w} M_1^{1,1,1}(\Theta^2 \eta_u^{0,v} , \Omega\Theta^2 \eta_u^{1,v\boxplus 1} , \Omega^2\Theta^2 \eta_u^{2,v\boxplus 2}) \qquad (4.18)$$

where $\boxplus$ is a modulo 3 addition, $w=q+3k+9$ and for $0\leq l\leq 2$, we have

$$T(\eta_u^{r,l}) = \begin{cases} k-u & \text{if } r\leq l \\ k-u-1 & \text{if } r>l \end{cases} \qquad \text{and} \qquad \eta_u^{r,l}(t) = \begin{cases} Y_{t,t+u}^{r,l} & \text{if } r\leq l \\ Y_{t,t+u+1}^{r,l} & \text{if } r>l \end{cases}$$

Proof : Using the input sequences (4.12) in (4.17) we obtain

$$\zeta_{u,v,q+1} = \sum_{g=1}^{q} \Omega^{3u+v+w} \left[\Omega^{3u+v} M_1^{1,1,1}(\Theta^2 \bar{\nu}_{g,0},\ \Omega\Theta^2 \bar{\nu}_{g,1},\ \Omega^2\Theta^2 \bar{\nu}_{g,2}) \right.$$
$$\left. * M_1^{1,1,1}(\Theta^2 \nu_{g,0},\ \Omega\Theta^2 \nu_{g,1},\ \Omega^2\Theta^2 \nu_{g,2}) \right]$$

$$= \Omega^{3u+v+w} \sum_{g=1}^{q} [\lambda_{u,v,g} * M_1^{1,1,1}(\Theta^2 \nu_{g,0},\ \Omega\Theta^2 \nu_{g,1},\ \Omega^2\Theta^2 \nu_{g,2})] \qquad (4.19)$$

where $w = q+3k+9$ and $\lambda_{u,v,g}$ is found by properties P4 and P5 to be equal to

$$\lambda_{u,v,g} = \begin{cases} M_1^{1,1,1}(\Theta^2\Omega^u\ \bar{\nu}_{g,0},\ \Omega\Theta^2\Omega^u\ \bar{\nu}_{g,1},\ \Omega^2\Theta^2\Omega^u\ \bar{\nu}_{g,2}) & \text{if } v=0 \\ M_1^{1,1,1}(\Theta^2\Omega^{u+1}\ \bar{\nu}_{g,2},\ \Omega\Theta^2\Omega^u\ \bar{\nu}_{g,0},\ \Omega^2\Theta^2\Omega^u\ \bar{\nu}_{g,1}) & \text{if } v=1 \\ M_1^{1,1,1}(\Theta^2\Omega^{u+1}\ \bar{\nu}_{g,1},\ \Omega\Theta^2\Omega^{u+1}\ \bar{\nu}_{g,2},\ \Omega^2\Theta^2\Omega^u\ \bar{\nu}_{g,0}) & \text{if } v=2 \end{cases}$$

The result (4.18) is then obtained by first applying P1 to perform the multiplication in (4.19), then by pulling Ω^{3u+v} out of the M operator with the help of P5 and by applying the summation to the arguments of M (property P1.2). As an illustration of the derivation procedure, we consider the case $v=1$ for which we have

$$\zeta_{u,1,q+1} = \Omega^{3u+1+w} \sum_{g=1}^{q} M_1^{1,1,1}(\Theta^2[\Omega^{u+1}\ \bar{\nu}_{g,2} * \nu_{g,0}],$$
$$\Omega\Theta^2[\Omega^u\ \bar{\nu}_{g,0} * \nu_{g,1}],\ \Omega^2\Theta^2[\Omega^u\ \bar{\nu}_{g,1} * \nu_{g,2}])$$

$$= \Omega^{3u+1+w} M_1^{1,1,1}(\Theta^2\Omega^{u+1}\ \eta_u^{2,0},\ \Omega\Theta^2\Omega^u\ \eta_u^{0,1},\ \Omega^2\Theta^2\Omega^u\ \eta_u^{1,2})$$

where, from P1, $T(\eta_u^{2,0}) = k-u-1$, $T(\eta_u^{0,1}) = T(\eta_u^{1,2}) = k-u$ and

$$\eta_u^{2,0}(t) = \sum_{g=1}^{q} [\bar{\nu}_{g,2}(t) * \nu_{g,0}(t-u-1)] = \sum_{g=1}^{q} [\bar{\nabla}_t^2(g)\ \nabla_{t-u-1}^0(g)] = Y_{t,t-u-1}^{2,0}$$

$$\eta_u^{0,1}(t) = \sum_{g=1}^{q} [\bar{\nu}_{g,0}(t) * \nu_{g,1}(t-u)] = \sum_{g=1}^{q} [\bar{\nabla}_t^0(g)\ \nabla_{t-u}^1(g)] = Y_{t,t-u}^{0,1}$$

$$\eta_u^{1,2}(t) = \sum_{g=1}^{q} [\bar{\nu}_{g,1}(t) * \nu_{g,2}(t-u)] = \sum_{g=1}^{q} [\bar{\nabla}_t^1(g)\ \nabla_{t-u}^2(g)] = Y_{t,t-u}^{1,2}$$

Finally, we apply P5 to get

$$\zeta_{u,1,q+1} = \Omega^{2(3u+1)+w} \; M_1^{1,1,1}(\theta^2 \; \eta_u^{0,1} \; , \; \Omega\theta^2 \; \eta_u^{1,2} \; , \; \Omega^2\theta^2 \; \eta_u^{2,0})$$

which is a special case of (4.18) for v=1. The cases v=0 and v=2 are proved in an exactly similar way.■

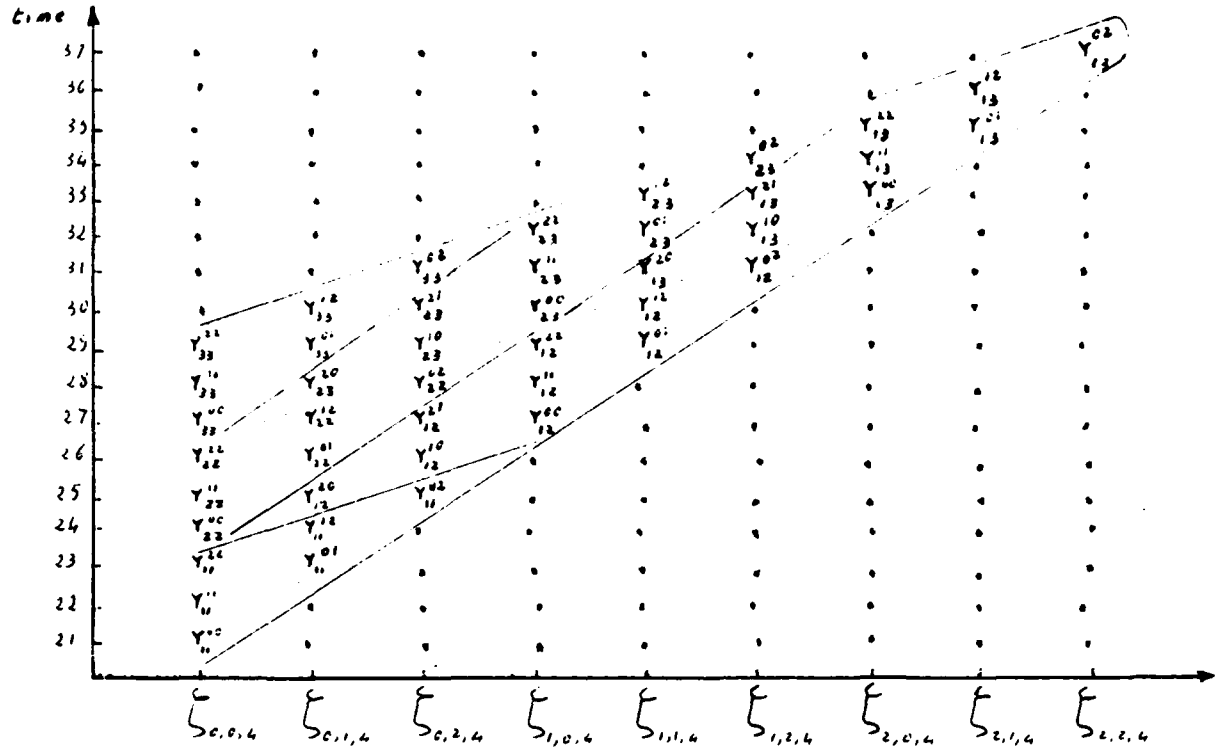


Figure 4.6

Equation (4.18) shows that the output sequences $\zeta_{u,v,q+1}$ contain the results of the numerical integration needed for the calculation of the stiffness matrices in the next subnetwork N5. It also specifies precisely the time of each output data item. In figure 4.6, this specification is translated into a time diagram, where we plot the elements of $\zeta_{u,v,q+1}$ versus time for the special case of k=3 and q=3.

4.5. The Subnetwork N5

The network N5 is composed of three different rows (see Figure 4.7). Row q+1 contains 3k identical nodes. It receives the constants $a^{e}_{r,l}$ on the links $p_{9,q+1}$, $r_{9,q+1}$ and $s_{9,q+1}$ and distributes them appropriately on the b colored links such that each integral $Y^{r,l}_{l,l}$ appearing on a z-colored link meets the corresponding constant $a^{e}_{r,l}$ at the right time. Row q+2 also contains 3k identical nodes and computes the partial sums $U^{r}_{l,l} = \sum_{l=0}^{2} a^{e}_{r,l}\, Y^{r,l}_{l,l}$ and $\sum_{l=r}^{2} (c_{r,l}\, a^{e}_{r,l})\, Y^{r,l}_{l,l}$ for $l \neq l$ and $l = l$, respectively, where $c_{r,l}$ is as given in ALG1. Finally, row q+3 contains only k nodes that complete the sum $H^{e}_{l,l} = \sum_{r=0}^{2} U^{r}_{l,l}$. The edges of the graph are given the colors p, r, s, b, z, z^0, z^1 or z^2 as shown in figure 4.7. Note that we used three different colors z^0, z^1 and z^2 to satisfy the restriction that no two edges ending at a node have the same color. To simplify the analysis, we consider each of the three rows separately.

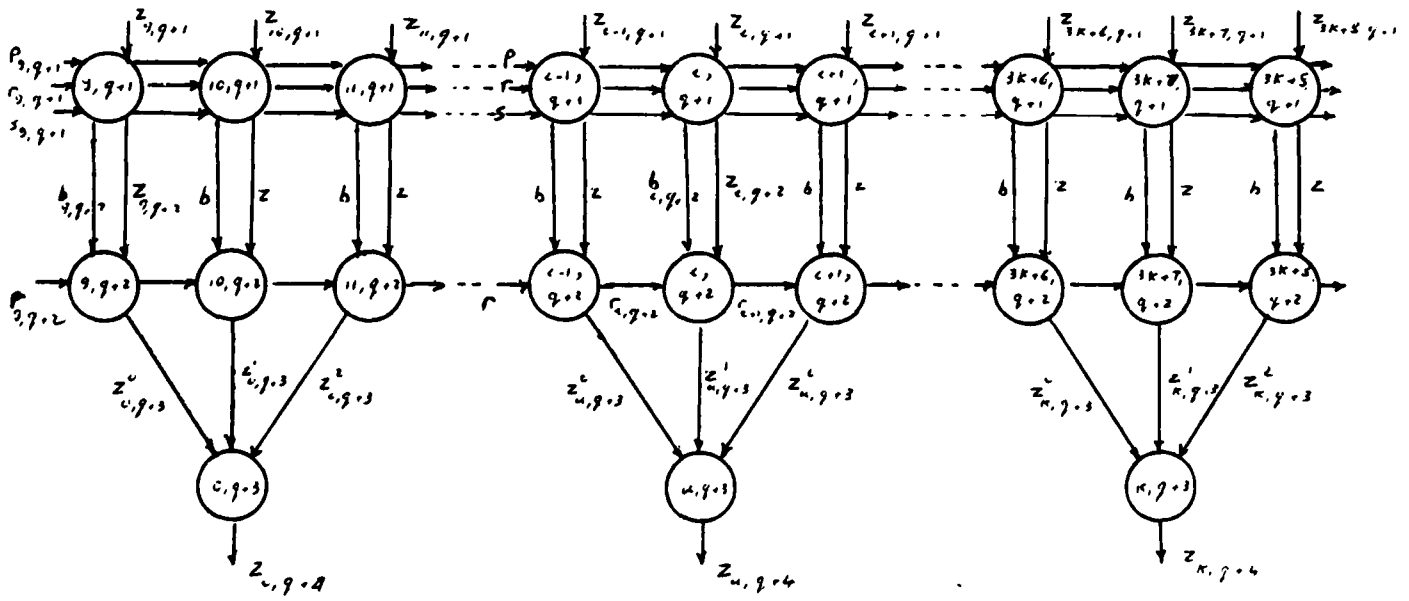


Figure 4.7

We consider first the row q+1 in which each cell simply latches the four data streams z, p, r and s by one time unit, and selects the output on the b link to be

$$\beta_{l,q+1} = \begin{cases} \Omega\ [h_l \cdot \pi_{l,q+1}] & \text{if } l=9+3u,\ u=0,\cdots,k-1 \\ \Omega\ \rho_{l,q+1} & \text{if } l=9+3u+1,\ u=0,\cdots,k-1 \\ \Omega\ \sigma_{l,q+1} & \text{if } l=9+3u+2,\ u=0,\cdots,k-1 \end{cases}$$

where h_l =0.5 for l=9 and h_l =1.0 for l>9. The factor 0.5 is needed to implement step N5.2 in ALG1, where only the $Y^{r,l}_{l,l}$, $l \leq r$ are explicitly available for the computation of $H^{e}_{l,l}$, while we have $Y^{r,l}_{l,l} = Y^{l,r}_{l,l}$ for $l>r$

For the proper operation of the system the input sequences should be described by

$$\begin{aligned} \pi_{9,q+1} &= \Omega^{w}\ P^{3}_{k}(\alpha_0) \\ \rho_{9,q+1} &= \Omega^{w}\ P^{3}_{k}(\alpha_1) \\ \sigma_{9,q+1} &= \Omega^{w}\ P^{3}_{k}(\alpha_2) \end{aligned} \qquad (4.20)$$

where for j=0,1,2, $T(\alpha_j)$ = 3 and $\alpha_j(t) = a^{e}_{t-1\square 2j,t-1}$, with $\square$ denoting the modulo 3 addition operation. More descriptively, we input on each line three of the constants $a^{e}_{r,l}$, r,l=0,1,2, repeated k times as indicated by the piping operator P^{3}_{k} (for more details see Figure 4.9).

Using the two indices $0 \leq u \leq k-1$ and $0 \leq v \leq 2$ as in the previous subsection, and noting that the input $\zeta_{l,q+1}$ is given by (4.18), we can easily show that

$$\zeta_{u,v,q+2} = \Omega^{2(3u+v)+w+1}\ M^{1,1,1}_{1}(\Theta^{2}\ \eta^{0,v}_{u}\ ,\ \Omega\Theta^{2}\ \eta^{1,v\square 1}_{u}\ ,\ \Omega^{2}\Theta^{2}\ \eta^{2,v\square 2}_{u}) \qquad (4.21.a)$$

$$\beta_{u,v,q+2} = \Omega^{3u+v+w+1}\ P^{3}_{k}(\ h_{u,v}\ \cdot\ \alpha_{v}\ ) \qquad (4.21.b)$$

where $h_{u,v}$ = 0.5 if u=v=0 and $h_{u,v}$ = 1.0 otherwise.

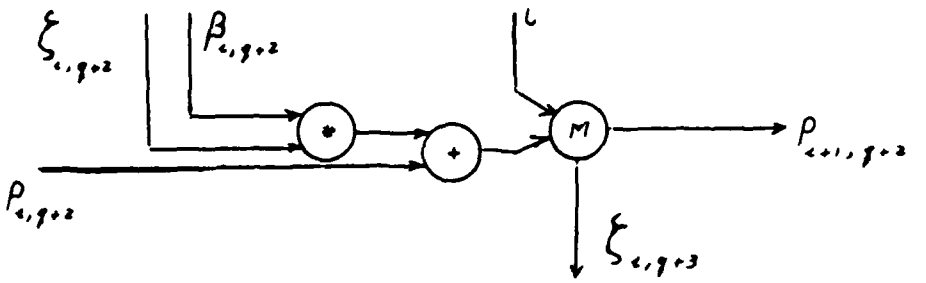


Figure 4.8 - A typical cell in row q+2 of N5.

The 3k cells, (i,q+2), $9 \leq i \leq 8+3k$, in row q+2 have basically the same structure, each is a multiplier/adder equipped with a demultiplexer that distributes the results to the output links $\rho_{i+1,q+2}$ and $\zeta^{v}_{u,q+3}$ (see figure 4.8 where u and v equal the quotient and the remainder of $\frac{i-9}{3}$, respectively). Formally, the operation of each cell (i,q+2) is described by

$$\rho_{i+1,q+2} = \Omega^2 \, M^{1,2}_{i-9+w+1}(\iota \, , \, \rho_i + \lambda_i) \tag{4.22.a}$$

$$\zeta^{v}_{u,q+3} = \Omega \, M^{1,2}_{i-9+w+1}(\rho_i + \lambda_i \, , \, \iota) \tag{4.22.b}$$

where $\lambda_i = \beta_{i,q+2} * \zeta_{i,q+2}$ and the input $\rho_{9,q+2}$ is permanently set to the zero sequence ι. For a description of the outputs $\zeta^{v}_{u,q+3}$, we solve (4.22) using Lemma 2 in the Appendix. This yields

$$\zeta^{v}_{u,q+3} = \begin{cases} \Omega \, M^{1,2}_{i-9+w+1}(\lambda_i \, , \, \iota) & i=9 \\ \Omega \, M^{1,2}_{i-9+w+1}([\Omega^2 \lambda_{i-1} + \lambda_i] \, , \, \iota) & i=10 \\ \Omega \, M^{1,2}_{i-9+w+1}([\Omega^4 \lambda_{i-2} + \Omega^2 \lambda_{i-1} + \lambda_i] \, , \, \iota) & i=11,\cdots,8+3k \end{cases} \tag{4.23}$$

where by (4.22) and (4.21), $\lambda_i = \lambda_{3u+v+9}$ is given by

$$\lambda_i = \Omega^{3u+v+w+1}\,[P^3_k(h_{u,v} \, , \alpha_v) * \Omega^{3u+v} \, M^{1,1,1}_1(\Theta^2 \eta^{0,v}_u \, , \, \Omega\Theta^2 \eta^{1,v\square 1}_u \, , \, \Omega \, \Theta^2 \eta^{2,v\square 2}_u)]$$

$$= \Omega^{6u+v+w+1}\,[P^3_{k-u}(h_{u,v} \, , \alpha_v) * \Omega^{v} \, M^{1,1,1}_1(\Theta^2 \eta^{0,v}_u \, , \, \Omega\Theta^2 \eta^{1,v\square 1}_u \, , \, \Omega^2\Theta^2 \eta^{2,v\square 2}_u)]$$

Moreover, with the help of property P17.2 we rewrite this as

$$\lambda_i = \lambda_{3u+v+9} = \Omega^{6u+v+w+1} \, \Omega^{v} \, M^{1,1,1}_1(\Theta^2 \mu^{0,v}_u \, , \, \Omega\Theta^2 \mu^{1,v\square 1}_u \, , \, \Omega^2\Theta^2 \mu^{2,v\square 2}_u) \tag{4.24}$$

where

$$\mu_u^{r,v\square r} = h_{u,v}\ \alpha_v((v\square r)+1)\ .\ \eta_u^{r,v\square r} = h_{u,v}\ a_{r,v\square r}^{e}\ \eta_u^{r,v\square r}, \qquad r=0,1,2$$

that is $T(\mu_u^{r,l}) = T(\eta_u^{r,l})$ and

$$\mu_u^{r,l}(t) = h_{u,v}\ a_{r,l}^{e}\ \eta_u^{r,l} \tag{4.25}$$

Proposition N5.1 : With the input described by (4.18) and (4.20), the intermediate sequences $\zeta_{u,q+3}^{v}$, $u=0,\cdots,k-1$, $v=0,1,2$, are given by

$$\zeta_{u,q+3}^{v} = \begin{cases} \Omega^{6u+v+w+1}\ M_1^{1,2}(\Omega^3\theta^2\ [\mu_u^{2,2} + \mu_{u-1}^{2,1} + \mu_{u-1}^{2,0}]\ .\ \delta^*) & \text{for } v=0 \\ \Omega^{6u+v+w+1}\ M_1^{1,2}(\Omega^3\theta^2\ [\mu_u^{1,2} + \mu_u^{1,1} + \mu_{u-1}^{1,0}]\ .\ \delta^*) & \text{for } v=1 \\ \Omega^{6u+v+w+1}\ M_1^{1,2}(\Omega^3\theta^2\ [\mu_u^{0,2} + \mu_u^{0,1} + \mu_u^{0,0}]\ .\ \delta^*) & \text{for } v=2 \end{cases} \tag{4.26}$$

where we extended the definition of $\mu_u^{r,l}$ such that $\mu_{-1}^{r,l}$ equals the zero sequence.

Proof : For the case $l \geq 11$, we first use P5 to rewrite (4.24) in the detailed form

$$\lambda_l = \lambda_{3u+v+9} = \begin{cases} \Omega^{6u+w}\ M_1^{1,1,1}(\Omega^3\theta^2\mu_u^{2,2}\ .\ \Omega\theta^2\mu_u^{0,0}\ .\ \Omega^2\theta^2\mu_u^{1,1}) & \text{for } v=0 \\ \Omega^{6u+w+1}\ M_1^{1,1,1}(\Omega^3\theta^2\mu_u^{1,2}\ .\ \Omega^4\theta^2\mu_u^{2,0}\ .\ \Omega^2\theta^2\mu_u^{0,1}) & \text{for } v=1 \\ \Omega^{6u+w+2}\ M_1^{1,1,1}(\Omega^3\theta^2\mu_u^{0,2}\ .\ \Omega^4\theta^2\mu_u^{1,0}\ .\ \Omega^5\theta^2\mu_u^{2,1}) & \text{for } v=2 \end{cases}$$

Then, for the evaluation of $\lambda_{l-1} = \lambda_{3u+v'+9}$, we note that $0 \leq v' \leq 2$ and hence $l-1 = 3u+v+8$ should be written in the form $3(u-1)+2+9$, $3u+0+9$ and $3u+1+9$ for $v=0,1$ and 2, respectively. With these forms for $l-1$ in (4.24) and the help of P5 we get

$$\Omega^2\lambda_{l-1} = \begin{cases} \Omega^{6u+w}\ M_1^{1,1,1}(\Omega^3\theta^2\mu_{u-1}^{2,1}\ .\ \Omega\theta^2\mu_{u-1}^{0,2}\ .\ \Omega^2\theta^2\mu_{u-1}^{1,0}) & \text{for } v=0 \\ \Omega^{6u+w+1}\ M_1^{1,1,1}(\Omega^3\theta^2\mu_u^{1,1}\ .\ \Omega^4\theta^2\mu_u^{2,2}\ .\ \Omega^2\theta^2\mu_u^{0,0}) & \text{for } v=1 \\ \Omega^{6u+w+2}\ M_1^{1,1,1}(\Omega^3\theta^2\mu_u^{0,1}\ .\ \Omega^4\theta^2\mu_u^{1,2}\ .\ \Omega^5\theta^2\mu_u^{2,0}) & \text{for } v=2 \end{cases}$$

Similarly, we write $l-2$ as $3(u-1)+1+9$, $3(u-1)+2+9$ and $3u+0+9$ for v=0,1 and 2, respectively and get

$$\Omega^4\lambda_{i-2} = \begin{cases} \Omega^{6u+w} \; M_1^{1,1,1}(\Omega^3\Theta^2\mu_{u-1}^{2,0} \cdot \Omega\Theta^2\mu_{u-1}^{0,1} \cdot \Omega^2\Theta^2\mu_{u-1}^{1,2}) & \text{for } v=0 \\ \Omega^{6u+w+1} \; M_1^{1,1,1}(\Omega^3\Theta^2\mu_{u-1}^{1,0} \cdot \Omega^4\Theta^2\mu_{u-1}^{2,1} \cdot \Omega^2\Theta^2\mu_{u-1}^{0,2}) & \text{for } v=1 \\ \Omega^{6u+w+2} \; M_1^{1,1,1}(\Omega^3\Theta^2\mu_{u}^{0,0} \cdot \Omega^4\Theta^2\mu_{u}^{1,1} \cdot \Omega^5\Theta^2\mu_{u}^{2,2}) & \text{for } v=2 \end{cases}$$

Then by adding these three formulas to get $\Omega^4\lambda_{i-2} + \Omega^2\lambda_{i-1} + \lambda_i$, by and substituting the result in (4.23), we directly obtain the equation (4.26) for $1 \leq u \leq k-1$.

The case $u=0$, that is $i=9,10$ and 11, can be analyzed in an exactly similar manner yielding the result

$$\zeta_{0,q+3}^{v} = \begin{cases} \Omega^{6u+w+v+1} \; M_1^{1,2}(\Omega^3\Theta^2 \; [\mu_u^{2,2}] \cdot \delta^*) & \text{for } v=0 \\ \Omega^{6u+w+v+1} \; M_1^{1,2}(\Omega^3\Theta^2[\mu_u^{1,2} + \mu_u^{1,1}] \cdot \delta^*) & \text{for } v=1 \\ \Omega^{6u+w+v+1} \; M_1^{1,2}(\Omega^3\Theta^2 \; [\mu_u^{0,2} + \mu_u^{0,1} + \mu_u^{0,0}] \cdot \delta^*) & \text{for } v=2 \end{cases}$$

which by defining $\mu_{-1}^{r,l} = \iota$ may also be put in the form (4.26). ■

Finally, each group of three sequences $\zeta_{u,q+3}^{0}$, $\zeta_{u,q+3}^{1}$ and $\zeta_{u,q+3}^{2}$ is considered as input to a cell $(u,q+3)$, $0 \leq u \leq k-1$, in row q+3 of N5. The operation of a typical cell in row q+3 is formally expressed by

$$\begin{aligned} \zeta_{u,q+4} &= \Omega\,[c_u \cdot A^{6u+w+2,3,1} \; M_{6u+w+2}^{1,1,1}(\zeta_{u,q+3}^{0} \cdot \zeta_{u,q+3}^{1} \cdot \zeta_{u,q+3}^{2})] \qquad (4.27) \\ &= \Omega\,[c_u \cdot M_{6u+w+2}^{1,1,1}(\delta^* \cdot \delta^* \cdot [\Omega^2\zeta_{u,q+3}^{0} + \Omega\zeta_{u,q+3}^{1} + \zeta_{u,q+3}^{2}])\,] \end{aligned}$$

where c_u equals to 2 for u=0 and to 1 otherwise.

By substituting the sequences (4.26) into the network description (4.27) we easily find the description of the output sequences as

$$\zeta_{u,q+4} = \Omega^{6u+w+7} \; \Theta^2 \; \bar{\mu}_u \qquad u=0,\cdots,k-1 \qquad (4.28)$$

where

$$\bar{\mu}_u = \begin{cases} \sum_{r=0}^{2} \sum_{l=r}^{2} \mu_u^{r,l} + \sum_{r=1}^{2} \sum_{l=0}^{r-1} \mu_{u-1}^{r,l} & \text{if } u>0 \\ \sum_{r=0}^{2} \sum_{l=r}^{2} \mu_u^{r,l} & \text{if } u=0 \end{cases}$$

Using the definition of $\mu_u^{r,l}$ from (4.25) in (4.28) and comparing the result with step N5 in algorithm ALG1, we readily prove the following proposition:

Proposition N5.2 : If the inputs to the network N5 are given by (4.18) and (4.20), then the network's output sequences are given by

$$\zeta_{u,q+4} = \Omega^{6u+w+7}\ \Theta^2\ \bar{\mu}_u \qquad u=0,\cdots,k-1 \qquad (4.28)$$

where $T(\bar{\mu}_u) = k-u$ and $\bar{\mu}_u(t) = H^{\theta}_{t,t+u}$.

Proposition N5.2 states that after an initial time period of 6u+3k+q+16 units, each output link $\zeta_{u,q+4}$ will carry the elements of the u^{th} off-diagonal of the stiffness matrix H^{θ}, separated from each other by 2 time units.

To summarize the behavior of the entire system, we show in Figure 4.9 a time diagram of the data on all the input and output links of the global system. It represents a translation of the sequence equations (4.5) (4.20) and (4.28) for the special case k=3 and q=3. The data items in the input sequences ξ_1, ξ_2, $\pi_{9,q+1}$, $\rho_{9,q+1}$ and $\sigma_{9,q+1}$ depend on the finite element that is being processed and hence they must be provided from outside the system. On the other hand, the data in $\rho_{1,g}$, $g=1,\cdots,q$ do not depend on a particular finite element and thus, as mentioned in Section 3, they are provided from a memory local to the system.

In general, the time for completing the computation of one element stiffness matrix is 9k+q+10 time units. In the next section, we will prove that the computa-

tion for different elemental stiffness matrices can be pelined through the system and that the elemental stiffness matrices can be generated at a rate of one matrix for every 3k time units.

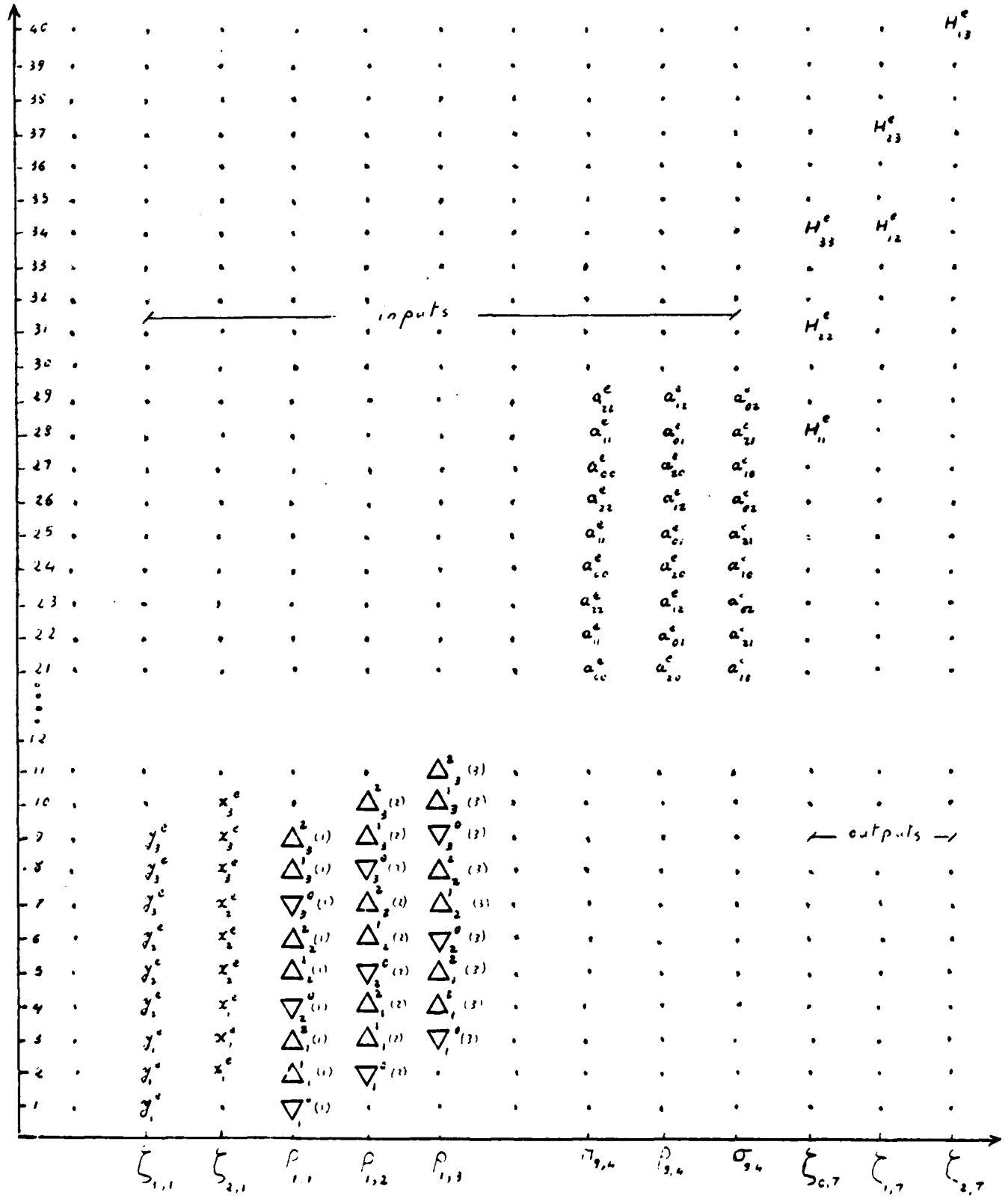


Figure 4.9

5. Verification of Pipelined Operation

For a given systolic network that has been shown to perform successfully a certain computation, we want to study the issue of repeating the same computation on different data in a pipelined fashion. Assume that a certain systolic network N has the I/O description

$$\eta_i = \Gamma_i(\xi_1, \cdots, \xi_n) \qquad i=1, \cdots, p \qquad (5.1)$$

where ξ_j, $j=1, \cdots, n$ and η_i, $i=1, \cdots, p$ are the input and output sequences of the network, respectively, and Γ_i, $i=1, \cdots, p$ denote certain causal operators that model the behavior of the network. Suppose also that for a certain input description

$$\xi_j = \Omega^{rj}\ \alpha_j \qquad j=1, \cdots, n \qquad (5.2)$$

with given integers rj and sequences α_j, we were able to show that the outputs are described by

$$\eta_i = \Omega^{si}\ \beta_i \qquad i=1, \cdots, p \qquad (5.3)$$

with certain integers si and sequences β_i. That is, in other words, suppose that when (5.2) is used in the equations (5.1), then we were able to prove that

$$\Omega^{si}\ \beta_i = \Gamma_i(\Omega^{r1}\ \alpha_1, \cdots, \Omega^{rn}\ \alpha_n) \qquad i=1, \cdots, p \qquad (5.4)$$

The calculation of the elements of β_i, $i=1, \cdots, p$, from those of α_j, $j=1, \cdots, n$ using the network N shall be called the computation "C". The time of this computation is defined as the time required by N to complete C from the moment when the first non-δ input entered N to the moment when the last non-δ output was produced. More precisely,

$$Time(C) = \max(\ T(\Omega^{si}\ \beta_i);\ 1 \le i \le p) - \min(\ rj:\ 1 \le j \le n) \qquad (5.5)$$

where T is the termination function defined in Section 2.

Often, it is desirable to repeat the computation C, say m times, with different data sets $A^e=(\alpha_j^e; j=1,\cdots n)$, $e=1,\cdots,m$. Let us denote these m instances of C by C^e, $e=1,\cdots,m$. In many networks, this may be accomplished by pipelining $C^1, \cdots, C^m$. The time difference between the initiations of two successive instances C^e and C^{e+1} will be defined as the pipe separation τ of the computation C. In this case, the inputs for the different instances of C should be pipelined on the input links. That is equation (5.2) for the input sequences should be replaced by

$$\xi_j^* = \Omega^{rj}\ P^{\tau}_{e=1,m}(\alpha_j^e) \qquad j=1,\cdots,n \tag{5.6}$$

where we used the asterix in ξ_j^* to indicate that the sequences represent the input data during the pipeline operation. We will also use ξ_j^e to represent the inputs (5.2) for a specific instance C^e of the computation. This * and e superscript notations will be used in the remainder of this section for sequences on any communication link.

If the computation can be successfully pipelined on N with a separation τ, then by using the inputs (5.6) in the network I/O description (5.1), we should be able to prove that the output sequences during pipelined operation are described by

$$\eta_i^* = \Omega^{si}\ P^{\tau}_{e=1,m}(\beta_i^e) \qquad i=1,\cdots,p \tag{5.7}$$

In order to ensure a successful pipelined operation, the pipe separation τ must be large enough so that the inputs of the different instances C^e do not overlap and the corresponding outputs do not overwrite each other. The first condition implies that $\tau \geq T(\alpha_j^e)$, $j=1,\cdots,n$, and the second that $\tau \geq T(\beta_i^e)$, $i=1,\cdots,p$. In other words, the minimum pipe separation $\tau_m(C)$ for the computation C is equal to the maximum span of all the input and output sequences in C, where the span of a sequence is defined as the time difference between the first

and the last non δ-elements in the sequence plus 1, that is the time during which the sequence carries imformation relevant to the computation. Hence, from the viewpoint of pipeline operation, a network that can be used to pipeline a computation C with a pipe separation $\tau_m(C)$ achieves maximum efficiency.

In order to prove (5.7) from (5.6) and (5.1) without repeating the effort spent in deriving (5.4), we use the negative shift operator and the equation (5.2) to rewrite the pipelined input (5.6) as

$$\xi_j^* = \Omega^{rj} \; P^{\tau}_{e=1,m} \; (\Omega^{-rj} \; \xi_j^e) \qquad j=1,\cdots,n \qquad (5.8)$$

where ξ_j^e are the inputs that would be used if the instance C^e of C had been performed on N without any pipelining. Next, we substitute (5.8) into the network I/O description (5.1) and obtain for $j=1,\cdots,p$

$$\eta_j^* = \Gamma_j([\Omega^{r1} \; P^{\tau}_{e=1,m} \; (\Omega^{-r1}\xi_1^e)],\cdots,[\Omega^{rn} \; P^{\tau}_{e=1,m} (\Omega^{-rn}\xi_n^e)]) \qquad (5.9)$$

The remainder of the proof is based on the use of the different properties in the Appendix for factoring the shift and the piping operators out of the causal operator Γ_j. If the computation can be successfully pipelined through N, then we should be able to transform (5.9) into the form

$$\eta_j^* = \Omega^{sj} \; P^{\tau}_{e=1,m} (\; \Omega^{-sj}\Gamma_j(\xi_1^e,\cdots,\xi_n^e) \;) \qquad j=1,\cdots,p \qquad (5.10)$$

which by (5.1) and (5.3) directly reduces to (5.7).

It should be noted however that there exist computations for which there is no value for τ for which (5.10) is derivable from (5.9) which means that the computation can not be pipelined. On the other hand, we can identify a class of computations for which pipelining is always possible. We use the term "inert" to identify computations in this class. In other words, a computation C on a systolic network N is called inert if it has the following two properties

1) At its initiation, C does not care about the data on the non input

communication links of N, that is we may assume that at time t=1, the data in any non input sequence are δ's. This implies that any delay in N should be modeled using the shift operator and not the zero shift operator.

2) Only δ-regular operators are used for modeling the cells in N. This implies that the network does not treat δ as a special symbol.

It is always possible to pipeline an inert computation C through the corresponding network N. In fact we may simply chose the pipe separation τ to be the time of the computation as defined by (5.5). With this value of τ, C^{e+1} does not start before C^{e} is terminated. Of course, we are not interested in such large values of τ, and hence, the problem arises of finding the least value of τ for which (5.10) is derivable from (5.9).

As should be clear from the above discussion, the ability to derive (5.10) from (5.9) is the major issue in verifying the pipeline operation of any systolic network, and this ability depends principally on the value of τ. However, for any inert computation C, we know that there exist a value for which (5.10) is derivable from (5.9). In order to find the least possible τ, we start with $\tau = \tau_m(C)$ and proceed to factor out the shift and piping operators from (5.9) until we either reach (5.10), which is our goal, or we cannot continue the factorization due to our small value of τ. In the latter case, we increase τ appropriately and repeat the derivation procedure.

For all the networks presented in Section 4, where all the computations are inert and the maximum span of all input and output sequences is 3k, it can be proved by the above technique that the m instances of the computation of the stiffness matrices can be pipelined through the system with a separation $\tau = \tau_m = 3k$. hence, the entire system can be used to generate the m stiffness matrices in a time equal to $t_c+3(m-1)k$, where t_c=9k+q+10 is the time consumed

by the first instance of the computation. In order to illustrate the derivation procedure, we will apply it to the verification of the pipeline operation of the subnetwork N4.

5.1. Pipeline verification of N4

Before starting our verification procedure, we recall that in section 4 the network I/O description of N4 was found to be given by equation (4.16), which is

$$\zeta_{i,q+1} = \sum_{g=1}^{q} \Omega^{i-8+q-g} [\Omega^{i-9} \pi_{9,g} * \rho_{9,g}] \qquad i=9,\cdots,3k+8 \qquad (5.11)$$

Moreover, when the inputs for a certain instance C^e of the computation are

$$\pi^e_{9,g} = \Omega^{g+3k+8} \overline{\nu}^e_g \qquad g=1,\cdots,q \qquad (5.12.a)$$

$$\rho^e_{9,g} = \Omega^{g+3k+8} \nu^e_g \qquad g=1,\cdots,q \qquad (5.12.b)$$

then the outputs are given by

$$\zeta^e_{i,q+1} = \Omega^{2i+3k+q-9} \eta^e_i \qquad i=9,\cdots,3k+8 \qquad (5.12.c)$$

where the detailed forms of the sequences $\overline{\nu}^e_g$ and ν^e_g containing the input data for C^e, and the sequences η^e_i containing the results of C^e are specified by (4.12) and (4.18), respectively. For the following discussion, we do not need these detailed forms. It suffices to know that $T(\overline{\nu}^e_g) = T(\nu^e_g) = 3k$ and $T(\eta^e_i) = 3k-(i-9)$, and hence that the minimum pipe separation is $\tau_m = 3k$.

If the computation C is pipelined through N4 with a separation of 3k, the inputs should have the form

$$\pi^*_{9,g} = \Omega^{g+3k+8} P^{3k}_{e=1,m}(\overline{\nu}^e_g) = \Omega^{g+3k+8} P^{3k}_{e=1,m}(\Omega^{-(g+3k+8)} \pi^e_{9,g}) \qquad (5.13.a)$$

and

$$\rho^*_{9,g} = \Omega^{g+3k+8} P^{3k}_{e=1,m}(\nu^e_g) = \Omega^{g+3k+8} P^{3k}_{e=1,m}(\Omega^{-(g+3k+8)} \rho^e_{9,g}) \qquad (5.13.b)$$

Using this in the network I/O description (5.11), we get the pipeline outputs in the form

$$\zeta^{*}_{l,q+1} = \sum_{g=1}^{q} \Omega^{l-8+q-g} [\Omega^{l-9} \Omega^{g+3k+8} P^{3k}_{e=1,m}(\Omega^{-(g+3k+8)} \pi^{e}_{9,g}) * \Omega^{g+3k+8} P^{3k}_{e=1,m}(\Omega^{-(g+3k+8)} \rho^{e}_{9,g})] \quad (5.14)$$

Now, by properties P1 and P8 in the Appendix we obtain

$$\zeta^{*}_{l,q+1} = \Omega^{l+3k+q} P^{3k}_{e=1,m}(\Omega^{-(l+3k+q)} \sum_{g=1}^{q} \Omega^{l-8+q-g}[\Omega^{l-9} \pi^{e}_{9,g} * \rho^{e}_{9,g}]) \quad (5.15)$$

which by (5.11) and (5.12.c) reduces to

$$\zeta^{*}_{l,q+1} = \Omega^{l+3k+q} P^{3k}_{e=1,m}(\Omega^{l-9} \eta^{e}_{l}) \quad (5.16)$$

Finally, because of $T(\Omega^{l-9}\eta^{e}_{l}) = l-9+T(\eta^{e}_{l}) = k$, we use P8 to write (5.16) as

$$\zeta^{*}_{l,q+1} = \Omega^{2l+3k+q-9} P^{3k}_{e=1,m}(\eta^{e}_{l})$$

which proves that the sets of results $(\eta^{e}_{l};\ l=9,\cdots,3k+8)$ of the different instances $e=1,\cdots,m$ will be correctly produced at the rate of $\frac{1}{3k}$ if the set of inputs $(\overline{\nu}^{e}_{g}\ ,\ \nu^{e}_{g};\ g=1,\cdots,q)$ are pumped through N4 at the same rate.

6. Concluding Remarks

This paper demonstrates the power of the extended systolic model by applying it to the specification and formal verification of a systolic system that can pipeline the computation of the elemental stiffness matrices.

There were no difficulties in establishing analytical proofs for the operation of the different components of our system. The reason for that may be the absence of feed back loops and the fact that our system does produce what we called an inert computation. However, an analytical verification of a systolic network is not always possible, and any conditions under which a network is analytically verifiable using our model are as yet still unknown, in part due to our incomplete sequence algebra. As a means for alleviating this problem, a computer program was developed that solves iteratively any system of consistent causal equations. This solver may be used in the verification of particular instances of computations whenever analytical verifications are not possible. The details of this solver/simulator will appear elsewhere.

Although the abstract model has been used here to specify the architecture at the level of the computational cells, the same model can also be used for lower or higher levels of architectures provided we define appropriately the domain R_δ of the data items that are transmitted on the communication links of the network, and the corresponding operators.

Besides its value in demonstrating the power of the systolic model, the system that generates the elemental stiffness matrices appears to have merit of its own. In fact, it is a contribution to the design of an integrated systolic finite element machine. For the implementation of any such machine, two alternatives may be considered: 1) We may use a systolic network similar to the one proposed in [22] to assemble the global stiffness matrix and then

apply one of many systolic networks suggested in the literature [3, 23, 25] for solving the resulting linear systems of equations. 2) Or we may use the system described in this report in a larger system that employs an iterative scheme for completing the finite element analysis. Further research is needed to assess the merits of the two approaches and to determine the global configuration of the system.

In addition to being adequate for VLSI implementation, the design presented in this report has the important advantage of being modular in the sense that if the system is designed for a specific value of k (element type) and q (quadrature formula), it can be easily modified to perform the analysis for different values of k and q. Of course the design is independent of the finite element mesh or the number of elements m in this mesh.

We have shown that for the general class of problems described in section 3, the computation of the elemental matrices is completed in approximately 3km time units. However, a careful examination of the design shows that this time may be reduced to (2k+1)m for some special problems in which the coefficients $a^{e}_{r,l}$ are equal to zero for r=0 or l=0. Examples of this important class of problems are the heat flow, the plain strain and plain stress problems [9]. To obtain this reduction in time, some control parameters have to be changed as well as the forms of the input sequences. At this point we note that with the technique described in section 5, it can be proved that a successful pipelining of the operation on the modified network requires a pipe separation τ equal to 2k+1. This is larger than the minimum pipe separation τ_m=2k for the computation, which means that the modified network cannot operate at maximum pipeline efficiency.

Finally, we note that it is not simple to define a measure that estimates the efficiency of systolic networks. An intuitive measure would be $\frac{TP}{C}$, where T is the time needed by a systolic network to complete the computation, P is the number of computational cells in the network, and C is the number of operations to be computed by the network. This measure, however, does not take into consideration the type of operation performed by a cell, which ranges in our case from simple memory cells to floating point dividers. It also ignores the benefit obtained by the regular movement of the data in the network. In [26] the authors suggested a more elaborate measure that takes into account the band width of the input and output links in the network in comparing the efficiency of the different systolic networks. Both measures estimate the utilization of the computational cells in a network without differentiating between the different types of cells. This is acceptable if all the cells in the network are of the same type. However, if the network contains more than one type of cells, as is the case with our system, we believe that the utilization of each cell should be multiplied by a weight that reflects the hardware complexity of the different cells. More work is needed to develop an efficiency measure of this type.

Acknowledgment

I would like to take the opportunity to thank professor Werner Rheinboldt for his continued guidance throughout the course of this research and for the time he spent in proofreading and commenting on earlier versions of this report.

Appendix

In this appendix we list some properties about combinations of the different operators defined in this report. All the properties are directly verifiable from the definition of the operators and are very useful in simplifying any manipulation of the sequence expressions. It should be noted that the zero shift operator is not included in any property. This is due to the fact that it was not used at all in modeling the networks presented in the report.

Most of the properties take the form "sequence expression = sequence expression". However, some have the form " sequence expression → sequence expression", where we formally define the implication operator → as follows:

$$IF \text{ for } any\ t\ either\ \eta(t)=\xi(t)\ or\ \eta(t)=\delta\ THEN\ \xi\rightarrow\eta$$

that is η is equal to ξ after replacing some of its elements by δ. Consequently, if $\xi\rightarrow\eta$, then we may replace ξ by η in any sequence expression as long as δ is treated as a don't care and not as a special symbol, that is in the contest of inert computations. Of course, if $\xi\rightarrow\eta$ and $\eta\rightarrow\xi$ then $\xi=\eta$.

P1) For any element-wise operator 'op' with δ 'op' $\delta=\delta$ we have

1.1) For Γ = Ω, Θ, E or P

$$\Gamma(\xi)\ 'op'\ \Gamma(\eta) = \Gamma(\xi\ 'op'\ \eta)$$

1.2) $$M_r^{w1,\ldots,wn}(\xi_1,\cdots,\xi_n)\ 'op'\ M_r^{w1,\ldots,wn}(\eta_1,\cdots,\eta_n) = M_r^{w1,\ldots,wn}([\xi_1\ 'op'\ \eta_1],\cdots,[\xi_n\ 'op'\ \eta_n])$$

1.3) As a direct result of P1.2 we have

$$\xi\ 'op'\ M_r^{w1,\ldots,wn}(\eta_1,\cdots,\eta_n) = M_r^{w1,\ldots,wn}([\xi\ 'op'\ \eta_1],\cdots,[\xi\ 'op'\ \eta_n])$$

1.4) if, in addition, 'op' is a δ-regular operator then

$$\Omega^r\ \xi\ 'op'\ \eta = \Omega^r\ \zeta$$

where $T(\zeta) = \min(T(\eta)-r, T(\xi))$ and $\zeta(t) = \xi(t)\ 'op'\ \eta(t+r)$

P2) For the scalar multiplication operator '.', it follows that

2.1) For $\Gamma = \Omega, \Theta, E$ or P

$$w \cdot \Gamma(\xi) = \Gamma(w \cdot \xi)$$

2.2) $$w \cdot M_r^{w1,\ldots,wn}(\eta_1,\cdots,\eta_n) = M_r^{w1,\ldots,wn}([w \cdot \eta_1] \cdot \cdots \cdot [w \cdot \eta_n])$$

P3) Composition of Ω with itself

3.1) $$\Omega\ \Omega\ \xi = \Omega^2\ \xi$$

3.2) $$\Omega^{-1}\ \Omega\ \xi = \xi$$

3.3) $$\Omega\ \Omega^{-1}\ \xi = \xi$$ if and only if $\xi(1)=\delta$

3.4) $$\xi \rightarrow \Omega\ \Omega^{-1}\ \xi$$

P4) Composition of Ω with Θ

$$\Theta^r\ \Omega^k\ \xi = \Omega^{(r+1)k}\ \Theta^r\ \xi$$ for $r \geq 0$ and *any* k

P5) Composition of Ω with M

5.1) $$\Omega^s\ M_r^{w1,\ldots,wn}(\xi_1,\cdots,\xi_n) = M_{r+s}^{w1,\ldots,wn}(\Omega^s\xi_1,\cdots,\Omega^s\xi_n)$$

for *any* r and $s > -r$

5.2) $$\Omega\ M_r^{w1,\ldots,wn}(\xi_1,\cdots,\xi_n) = M_r^{w1,\ldots,wn}(\Omega\xi_n \cdot \Omega\xi_1 \cdot \cdots \cdot \Omega\xi_{n-1})$$

5.3) $$\Omega^k\ M_r^{w1,\ldots,wn}(\xi_1,\cdots,\xi_n) = M_r^{w1,\ldots,wn}(\Omega^k\xi_1,\cdots,\Omega^k\xi_n)$$

where $k = w_1+\cdots+w_n$

5.4) $$M_{r+1}^{w1,\ldots,wn}(\xi_1,\cdots,\xi_n) = \Omega^r\ \Omega^{-r}\ M_{r+1}^{w1,\ldots,wn}(\xi_1,\cdots,\xi_n)$$

5.5) $$M_1^{w1,\ldots,wn}(\xi_1,\cdots,\xi_i,\cdots,\xi_n) = M_i^{w1,\ldots,wn}(\xi_1,\cdots,\Omega^q\ \Omega^{-q}\xi_i,\cdots,\xi_n)$$

where $q = w_1+\cdots+w_{i-1}$

P6) Composition of Ω with E

6.1) $$E_{r+s}^k\ \Omega^s\ \xi = \Omega^s\ E_r^k\ \xi$$ for *any* r and $s > r$

6.2) $$E_{r+1}^k\ \xi = \Omega^r\ \Omega^{-r}\ E_{r+1}^k\ \xi$$

6.3) $$E_r^k\ \Omega^k\ \xi \rightarrow \Omega^k\ E_r^k\ \xi$$

P7) Composition of Ω with A

7.1) $A^{r,k,s}\ \Omega^{u}\ \xi = \Omega^{u}\ A^{r-u,k,s}\ \xi$ for $u < r$

7.2) $A^{r+1,k,s}\ \xi = \Omega^{r}\ \Omega^{-r}\ A^{r+1,k,s}\ \xi$

P8) Composition of Ω with P

8.1) $\Omega^{r}\ P^{k}_{\theta=1,m}(\xi^{\theta}) \rightarrow P^{k}_{\theta=1,m}(\Omega^{r}\ \xi^{\theta})$

8.2) $P^{k}_{\theta=1,m}(\Omega^{r}\ \xi^{\theta}) \rightarrow \Omega^{r}\ P^{k}_{\theta=1,m}(\xi^{\theta})$ if $T(\xi^{\theta}) \leqslant k-r$

8.3) $\Omega^{-r}\ P^{k}_{\theta=1,m}(\xi^{\theta}) \rightarrow P^{k}_{\theta=1,m}(\Omega^{-r}\ \xi^{\theta})$ if $T(\xi^{\theta}) \leqslant k$

8.4) $P^{k}_{\theta=1,m}(\Omega^{-r}\ \xi^{\theta}) \rightarrow \Omega^{-r}\ P^{k}_{\theta=1,m}(\xi^{\theta})$ if $\Omega^{-r}\ \Omega^{r}\ \xi^{\theta} \rightarrow \xi^{\theta}$

8.5) $P^{k}_{m}(\xi) \rightarrow \Omega^{k}\ P^{k}_{m-1}(\xi)$

P9) Composition of Θ with itself

$$\Theta^{r}\ \Theta^{k}\ \xi = \Theta^{k}\ \Theta^{r}\ \xi = \Theta^{kr+k+r}\ \xi$$

P10) Composition of Θ with E

$$E^{s}_{1}\ \Theta^{s-1}\ \xi \rightarrow \Theta^{s-1}\ \xi$$

P11) Composition of Θ with A

$$A^{1,k,s}\ \Theta^{s-1}\ \xi = E^{s}_{1}\ \Theta^{s-1}\ A^{1,k,1}\ \xi$$

P12) Composition of Θ with P

$$P^{sk}_{\theta=1,m}(\Theta^{s-1}\ \xi^{\theta}) = \Theta^{s-1}\ P^{k}_{\theta=1,m}(\xi^{\theta})$$

P13) Composition of M with itself

13.1) if $\xi_{j} = M^{1,\ldots,1}_{r}(\eta_{1},\cdots,\eta_{n})$ then

$$M^{1,\ldots,1}_{r}(\xi_{1},\cdots,\xi_{j},\cdots,\xi_{n}) = M^{1,\ldots,1}_{r}(\xi_{1},\cdots,\eta_{j},\cdots,\xi_{n})$$

13.2) $M^{k-m,1,\ldots,1}_{r}(M^{k-n,1,\ldots,1}_{r+n}(\zeta,\xi_{1},\cdots,\xi_{n}),\ \eta_{1},\cdots,\eta_{m}) =$

$$M^{k-n-m,1,\ldots,1}_{r+n}(\zeta,\eta_{1},\cdots,\eta_{m},\xi_{1},\cdots,\xi_{n}) \qquad \text{for } m+n < k$$

P14) Composition of M with A

14.1) $A^{r,k,s}\ M_1^{1,\cdots,1}(\xi_1,\cdots,\xi_s) = A^{r,k,s}\ \xi_r$ for $1 \leq r \leq s$

14.2) $A^{r,k,1}\ M_r^{1,\cdots,1}(\xi_1,\cdots,\xi_k) = M_r^{1,\cdots,1}(\eta_1,\cdots,\eta_k)$

where $\eta_i = \sum_{u=1}^{i} \Omega^{i-u}\ \xi_u$

P15) Composition of E with P

$$E_1^k\ P_{\theta=1,m}^k(\xi^\theta) = P_{\theta=1,m}^k(E_1^k\ \xi^\theta)$$

P16) Composition of A with P

$$A^{1,k,1}\ P_{\theta=1,m}^{nk}(\xi^\theta) = P_{\theta=1,m}^{nk}(A^{1,k,1}\ \xi^\theta)$$

P17) Other properties involving the multiplication operator '*'

17.1) $E_{r+1}^k\ \xi * \eta \rightarrow [\xi](r+1)\ .\ \Omega^r\Omega^{-r}\ \eta$ if $T(\eta) \leq k+r$

17.2) $\Omega^i\ M_1^{1,\cdots,1}(\xi_0,\cdots,\xi_{n-1}) * P_k^n(\eta) = \Omega^i\ M_1^{1,\cdots,1}(\zeta_0,\cdots,\zeta_{n-1})$

where $\zeta_r = \eta((i \square_n r)+1)\ .\ \xi_r\ ,\ r=0,1,\cdots,n-1$ and $\square_n$ is the modulo addition operation on integers.

Next, we state two lemmas that can be proved using the above properties :

Lemma 1 : The system of difference equations

$$\xi_{g+1} = \Omega\ \xi_g + \Delta_g \qquad g=1,\cdots,k$$

has the solution

$$\xi_{r+1} = \Omega^r\ \xi_1 + \sum_{l=1}^{r} \Omega^{r-l}\ \Delta_l \qquad r=1,\cdots,k$$

Lemma 2 : The system of difference equations

$$\rho_{i+1} = \Omega^2\ M_{s+i}^{1,2}(\iota\ .\ [\lambda_i + \rho_i]) \qquad i=r_0,\cdots,r_1$$

$$\zeta_i = \Omega\ M_{s+i}^{1,2}([\lambda_i + \rho_i]\ .\ \iota) \qquad i=r_0,\cdots,r_1$$

with the condition $\rho_{r0} = \iota$ has the solution

$$\zeta_i = \begin{cases} \Omega\ M_{s+i}^{1,2}(\lambda_i\ ,\ \iota) & i=r_0 \\ \Omega\ M_{s+i}^{1,2}([\Omega^2\lambda_{i-1} + \lambda_i]\ ,\ \iota) & i=r_0+1 \\ \Omega\ M_{s+i}^{1,2}([\Omega^4\lambda_{i-2} + \Omega^2\lambda_{i-1} + \lambda_i]\ ,\ \iota) & i=r_0+2,\cdots,r_1 \end{cases}$$

REFERENCES

[1] Melhem R. G. and Rheinboldt W. C., "A Mathematical Model for the Verification of Systolic Networks," Technical Report ICMA-82-47 (Oct. 1982). The University of Pittsburgh.

[2] Kung H. T., "Why Systolic Architecture," *Computer Magazine*, pp.37-46 (Jan. 1982).

[3] Kung H. T. and Leiserson C. E., *Systolic Arrays for VLSI*, Addison_Wesley, Reading Mass., 1980.

[4] Chen M. C. and Mead C. A., "Formal Specification of Concurrent Systems," *USC Workshop on VLSI and Modern Signal Processing* (Nov. 1982).

[5] Johnsson L., Weiser U., Cohen D., and Davis A., "Toward a Formal Treatement of VLSI Arrays," Tecnical Report 4191 (1981). Dept. of Computer Science, California Institute of Technology

[6] Weiser U. and Davis A., "Mathematical Representation for VLSI Arrays," Technical Report UUCS-80-111 (Sept. 1980). Department of Computer Science, University of Utah

[7] Cohen D., "Mathematical Approach to Iterative Computational networks," *Proccedings of the Fourth Symposium on Computer Arithmetics*, pp.226-238 (Oct. 1978).

[8] Weiser U. and Davis A., "A Wavefront Notation Tool for VLSI Array Design," *VLSI Systems and Computations*, ed. by H. T. Kung, B. Sproull and G. Steele (1981).

[9] Zienkiewicz O. C., *The Finite Element Method*, McGraw-Hill (1979). Third edition.

[10] Pilkey W., Saczalski K., and Schaeffer H., *Eds. Structural Mechanics Computer Programs, Survey, Assesements and Availability*, University of Virginia Press. Charlottesville, VA. (1974).

[11] Sarigul N., Mai-tan J., and Kamel H., "Solution of Nonlinear Structure Problems Using Array Processors." *The FENOMECH 81 conference, Stuttgart* (Aug. 1981).

[12] Mai-tan J., Sarigul N., Palusinski O., and Kamel H., "Balanced Array Processor Configuration for Finite Element Analysis," N00014-75-C-0837 (Feb. 1982). University of Arizona

[13] Noor A. K. and Hartley S. J., "Evaluation of Element Stiffness Matrices on a CDC Star-100 Computer." *Computer and Structures* Vol. **9**, pp.151-161 (1978).

[14] Noor A. K. and Lambiotte J. J., "Finite Element Dynamic Analysis on CDC Star-100 Computer," *Computer and Structures* Vol. **10**, pp.7-19 (1979).

[15] Deminet J. , "Experiments with Multiprocessor Algorithms," *IEEE Trans. on Computers* Vol. **C.31**(4), pp.278-287 (April 1982).

[16] Gehringer E. F., Jones A. K., and Segall Z. Z., "The Cm* Testbed," *Computer* Vol. **15**(10), pp.40-53 (Oct. 1982).

[17] Zave P. and Rheinboldt W. C., "Design of an Adaptive, Parallel, Finite Element System," *ACM Trans. on Mathematical Software*, pp.1-17 (March 1979).

[18] Zave P. and Cole G., "A Quantitative Evaluation of the Feasibility of a suitable Hardware Architecture for an Adaptive Finite Element System." Report number BN-971. (Sept. 1981). Institute for Physical Science and Technology. University of Maryland.

[19] Jordan H. F., "A Multiprocessor System for Finite Element Structural Analysis." *Computer and Structures* Vol. **10**, pp.21-29 (1979).

[20] Jordan H. F., "A Special Purpose Architecture for Finite Element Analysis." *Proc. of the 1978 International Conference on Parallel Processing*, pp.263-266.

[21] Bokhary S. H., "On the Mapping Problem." *Proc. of the 1979 International Conference on Parallel Processing* (1979).

[22] Law K. H., "Systolic Schemas for Finite Element Methods." R-82-139 (July 1982). Carnegie Institute of Technology, Carnegie-Mellon University.

[23] Brent R. R. and Luk F. T., "Computing the Cholesky Factorization using a Systolic Architecture." Technical Report 82-521 (Sept. 1982). Dept. of Computer Science, Cornell University

[24] Johnsson L. and Cohen D., "A Mathematical Approach to Modeling the Flow of Data and Control in Computational Networks." *VLSI Systems and Computations* (1981). ed. by H. T. Kung, B. Sproull and G. Steele.

[25] Anmed H., Delosme J., and Morf M., "Highly Concurent Computing Structures for Matrix Arithmetic and Signal Processing." *Computer* (Jan. 1982).

[26] Huang K. and Abraham J., "Efficient Parallel Algorithms For Processor Arrays." *Proc. of the 1982 International Conference on Parallel Processing*, pp.271-279.