# SYSTEMATIC DESIGN OF TWO-LEVEL PIPELINED SYSTOLIC ARRAYS WITH DATA CONTRAFLOW

M. Valero-Garcia, J.J. Navarrro, J.M.LLaberia, M. Valero

Facultad de Informática de Barcelona (U.P.C)
Pau Gargallo,5, 08028 Barcelona (Spain)

## ABSTRACT

Many systolic algorithms and related design methodologies have been recently proposed. Frecuently, in these systolic algorithms practical considerations are not taken into account. Equitatively distributed load between processing elements, pipelined functional units etc, are desirable features when implementing systolic algorithms.In this paper we present a design methodology in which these features are considered. As an example, the methodology is applied to obtain a problem-size-independent, two-level pipelined 1D systolic algorithm with data contraflow to efficiently solve triangular systems of equations.

## INTRODUCTION

The design of simple and feasible, high performance, and algorithmically specialized concurrent systems is a very desirable goal when we consider many applications which require a great amount of computations performed at high speed. A present example of such systems, which are nowadays actively researched, are Systolic Array Processors [1].

A large set of Systolic Algorithms (SAs), as well as SA design methodologies [2] have been recently published. Nevertheless, a large part of these SAs shows several characteristics which are not suitable for its direct implementation. In these cases, adecuate transformations must be done on the SA to map it conveniently onto the proposed architecture, accordingly to the restrictions imposed. In this paper three types of transformations applied to SAs are considered:

1.-Partitioning. That is, problem-size-dependent SAs are transformed into problem-size-independent SAs. In a problem-size-dependent SA the number of processing elements (PEs) depends on some dimension of the data structures of the problem to be solved. The number of PEs is, on the contrary, previously fixed and is independent of the problem dimensions, when a problem-size-independent SA is appointed. Partitioning issues are important because the number of PEs in the SA becomes a fixed value on account of technological and/or economical considerations.

2.-Computational balance. This aspect means the need of transforming non-balanced SAs into balanced SAs. The non-balanced SAs are caracterized for the requirement imposed on some PEs to perform calculations which are of greater complexity that those to be accomplished, in a single systolic cycle, by other PEs. Nevertheless, in a balanced SA, the load is distributed between all the PEs as much equitatively as possible, in order to obtain that the PEs perform useful task the maximun amount of time attainable.

This work was supported by CAICYT under contract PA85-0314

3.-Two-level pipeline. It consist in the transformation of one-level pipelined SAs into two-level pipelined SAs, where the functional units inside the PEs are pipelined. To implement this kind of SAs is important if we want to get systems with high throughput.

Several authors have undertaken the design of problem-size-independent SAs. Reference [3] shows enough information on this subject.

We have not found, until the present, any report on research with a formal treatment of the design of balanced SAs to solve non-homogeneous problems.

The design of two-level pipelined SA ha been presented by Kung and Lam in [4], where SAs without data contraflow [3] are considered. It is well known the existence of non-homogeneous problems with dependences between results (i.e. triangular linear systems, LU decomposition,...) solved efficiently by means of SAs with data contraflow. Kung and Lam propose for these problems a new type of SAs without data contraflow, called "Systolic rings" [4]. It is to remark, nevertheless, that the implementation of SAs without data contraflow require a greater amount of hardware (for functional units and control) than SAs with data contraflow for these problems. For example, in a systolic ring to solve triangular system of equations, all the PEs must perform divisions in some cycles and multiplications and additions in other cycles.

In this paper, a systematic methodology to transform non-balanced and/or one-level pipelined SAs into balanced and/or two-level pipelined SAs, is to be presented. This methodology is based upon two transformation rules, to be applied on a model of the SA. The proposed model is an extension of one previously presented by Kung and Li [5]. The first rule is a combination of the two formal transformations proposed in [5]. The second rule we propose is a formal transformation to the attainment of a 1-slow algorithm from a k-slow one, through adjacent PEs coalescing. With these transformation rules we obtain balanced and/or two-level pipelined SAs with data contraflow. To reach problem-size-independent SAs, we use techniques of partitioning and DBT transformations recently published in [3].

This paper is structured as follows: Section 2 presents the formal model for 1D SAs. The case of a problem-size-independent SA to solve triangular systems of equations is presented as an example. Section 3 enunciate the two proposed rules for transformations. In section 4, these rules are applied to obtain a problem-size-independent, balanced and two-level pipelined 1D SA with data contraflow from the non-balanced and one-level pipelined SA presented in section 2.

## CHARACTERIZATION OF 1D SA

A 1D SA consists in the grouping of w Processing Elements (PEs), which we shall name $PE_1,...,PE_w$. These PEs are interconnected through unidirectional links. Each PE is also possibly communicated with the outside world through I/O unidirectional links. The I/O links are used to input (or output) sequences of data to (or from) the SA; these sequences will be named here as input (or output) flows.

Every PE simultaneously receive all data involved in the computation to be performed in each cycle. Any computation may produce one or several results. These results may leave the PE in different cycles, depending upon the time interval needed for the PE to produce them. A time delay is associated to each link which is used to connect any two PEs. This delay is measured by the number of registers along the link.

Some PEs in the SA may perform certain operations during some cycles, and other type of operations during other cycles. If this is the case, we say that the PE present a non-homogeneous time behaviour. When each PE initiates one valid operation every k cycles, we say that the SA is k-slow.

In order to simplify the required notation, we assume that only one link, at most, in each direction exists between any pair of PEs in the SA. To extend this assumption to a general case is straighforward.

### Modelization of a 1D SA

A 1D SA with w PEs may be modelled by the tuple A=(w, I, O, L, R, E, S, P, k) and by the definition of operations performed in each cycle by every PE.

The value w is the number of PEs of SA. I is the set $I_1,...,I_p$, where p is the number of links entering into the SA from the outside. $I_j$, for $j \in [1..p]$ is the data sequence $\{I_j(1), I_j(2),...\}$ which inputs to the SA through the j-th input link. O is the set $O_1,...,O_q$, where q is the number of links leaving the SA. $O_j$, for $j \in [1...q]$ is the data sequence $\{O_j(1), O_j(2),...\}$ which outputs the SA through the j-th output link.

L, R, E and S are matrices in the form: $X(i,j) = z^{-x(i,j)}$ or $X(i,j) = 0$. L (latency matrix) has w-by-w elements. The value of $l(i,j)$ is the number of cycles needed by the $PE_j$ to calculate every data item to be sent to the $PE_i$. If no communication exist between $PE_j$ and $PE_i$, then we have L(i,j)=0. R (register matrix) has w-by-w elements. The value of r(i,j) is the delay associated with the link from $PE_j$ to $PE_i$. If such link does not exist, then R(i,j)=0.

When a non-homogeneous time behaviour is exhibited by $PE_j$, and $PE_j$ is linked with $PE_i$, then we have

$$L(i,j) = z^{-(l^1(i,j),\ l^2(i,j),...,l^{n_j}(i,j))}$$

$$R(i,j) = z^{-(r^1(i,j),\ r^2(i,j),...,r^{n_j}(i,j))}$$

where $n_j$ is the number of different operations performed by $PE_j$; we name these operations as $OP_1,..., OP_n$. The value of $l^s(i,j)$ is the number of cycles needed by $PE_j$ to calculate, by means of $OP_s$, the value to be sent to $PE_i$. Similary, $r^s(i,j)$ is the delay associated to the link between $PE_j$ and $PE_i$, when $PE_j$ performs $OP_s$.

E (entrance matrix) has w-by-p elements, and e(i,j) is the number of cycles from the begining of SA operation until the $PE_i$ receives the first data item in the $I_j$ sequence. If $PE_i$ receives no data of $I_j$, then E(i,j)=0. S (sally matrix) has w-by-q elements, and s(i,j) is the number of cycles from the begining of SA operation until $PE_i$ initiates the calculation of the first data item in the $O_j$ sequence. If $PE_i$ produces no data for $O_j$, then S(i,j)=0. P (periodicity vector) has w elements. The element P(i)= p(i) is the number of cycles from the begining of an operation in $PE_i$ until the moment in which $PE_i$ may initiate the next operation. If $PE_i$ exhibits a non-homogeneous time behaviour, we shall write $P(i)=(p^1(i), p^2(i),...,p^{n_k}(i))$, where $p^s(i)$ is the periodicity of operation $OP_s$. Finally, the value of integer k indicates the slow of the SA.

### Modelling of a 1D SA to solve triangular systems of linear equations.

In figure 1 we show the structure of an special type of 1D SAs, which we name1D spiral SAs with data contraflow. This kind of SAs may serve to solve a broad range of matrix problems, such as matrix multiplication, triangular systems of linear equations, LU decomposition, QR decomposition by Givens rotations, etc. All these problems can be solved, for any value of the involved matrix dimensions, on a fixed-size SA (with a given number of PEs), by means of the partitioning technique known as DBT [3].

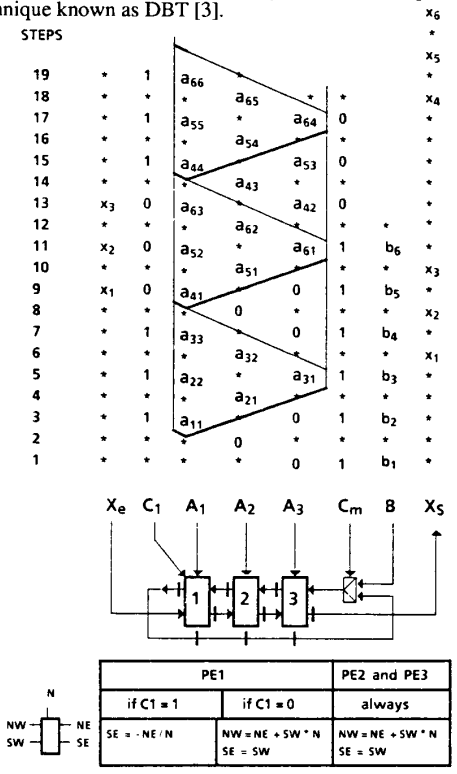


Figure 1. Problem-size-independent Systolic Algorithm for triangular system of equations.

We shall now present the modelization of a 1D spiral SA with data contraflow to solve a triangular system of linear equations, with any size. In [3] a detailed description of this SA operation can be found. Figure 1 shows the SA in the particular case of 3 PEs, as well as the I/O data sequences to solve a system with 6 unknowns. There is non-homogeneous time behaviour in $PE_1$, because it performs one division and one sign change during some cycles, but one multiplication followed by one addition during other cycles. This $PE_1$ receives a control signal ($C_1$) which can be treated as another input flow. $C_1$ determines, in each cycle, which one of the two possible operations are to be performed in the PE. All other PEs, except $PE_1$, always perform one multiplication followed by one addition. The model is as follows:

$l(i,i+1) = 0, \quad i \in [1...w-1]; \qquad l(i+1,i) = 0, \quad i \in [2...w-1]$
$l(w,1) = l^1(2,1) = l^2(2,1) = 0; \quad L(i,j) = 0$ in the other cases;

because, in the original design, the assumption of zero cycles to perform any operation is made. On the other hand,

$r(i,i+1) = 1, \quad i \in [1...w-1]; \qquad r(i+1,i) = 1, \quad i \in [2...w-1];$
$r(w,1) = w+1; \; r^1(2,1)=r^2(2,1) = 1; \; R(i,j) = 0$ in the other cases;

because all the links between PEs have an associated delay equal to 1, except the link between $PE_1$ and $PE_w$, which has a delay of value w+1.

We denote the input flows to the SA as follows:

$I_i = A_i \; i \in [1...w]; \; I_{w+1}=B; \; I_{w+2}=X_e; \; I_{w+3}=C_1; \; I_{w+4} = C_m$

We denote the only output flow as: $O_1=X_s$.

The $A_i$, B, $X_e$, $C_1$, $C_m$ and $X_s$ sequences are specified according to the DBT transformation rules, when applied to this kind of problem (see figure 1).

Entrance and sally matrices are specified as:

$e(i,i) = w+i-2, \; i \in [1...w]; \; e(1,w+2)= 3w-1; \; e(1,w+3) = w-1$
$e(w,w+1) = 0; \; e(w,w+4) = 0; \; E(i,j) = 0$ in the other cases.

$s(w,1) = -(2w-1); \; S(i,j) = 0$ in the other cases.

The periodicity vector can be specified as:

$P(i) = 1, \quad i \in [2...w]; \qquad P(1) = (1,1).$

The slow of the SA is k=2.

## TRANSFORMATION RULES IN 1D SAs

Now we shall present two transformation rules for 1D SAs, which are to be used in the proposed design methodology.

Rule 1. H.T. Kung presents in [5] two transformations applicable to time homogeneous SAs. The first one is equivalent to the retiming lemma proposed by Leiserson [6]; it allows a register redistribution inside the SA. The second transformation allows to attain a c-slow version of the SA; this is accomplished through a multiplication by c of the number of registers associated to each link between PEs, and through an adequate modification of the input and output data sequences.

A rule, extending these two mentioned transformations, follows in this paper. Main features of this rule are: a) computation time intervals (L matrix) and communication time intervals (R matrix) are distinguished; b) specification of SAs with non-homogeneous time behaviour is alloved. Rule is:

The A' = (w, I', O', L', R', E', S', P', k') SA performs the same calculation as A = (w, I, O, L, R, E, S, P, k) if:

$L' \circ R' = D\,(L \circ R)^c\,D^{-1}$
$E' = D\,E^c$
$S' = (S^c)\,D^{-1}$
$k' = ck$
$I'_i(k't+1) = I_i(kt+1) \qquad i \in [1...p]$ and $t \geq 0$
$O'_i(k't+1) = O_i(kt+1) \qquad i \in [1...q]$ and $t \geq 0$

where symbol "o" denotes a matrix operation defined as follows: if $PE_j$ exhibits non-homogenous time behaviour

$(L \circ R)(i,j) = z^{-(l^1(i,j)+r^1(i,j),\ldots,l^{n_j}(i,j)+r^{n_j}(i,j))}$

and if $PE_j$ always performs same operation (homogeneous time behaviour)

$(L \circ R)(i,j) = z^{-(l(i,j)+r(i,j))}$

Any matrix of the form $X^c$ is defined as: $X^c(i,j) = z^{-cx(i,j)}$ if $X(i,j) = z^{-x(i,j)}$ or $X^c(i,j) = 0$ if $X(i,j) = 0$, D is a diagonal matrix, with w*w elements of the type $D(i,i) = z^{-d_i}$, where $d_i$ and c belong to the set of rationale numbers.

Rule 2 (coalescing rule).The goal of this second rule is to obtain a transformation of a k'-slow SA, in which any PE performs only one valid operation every k' cycles, into a 1-slow SA with lower number of PEs, in which any PE performs useful task in all cycles. In this manner we perform the same computations, in the same time , but using less hardware more efficiently. This rule is expressed as follows:

Given the A'= (w, I', O', L', R', E', S', P', k') SA; where $p'^s(i)=1$, with $s \in [1...n_i]$ and $i \in [1..w]$, we shall obtain another SA, A*= (w*, L*, R*, E*, S*, P*, k*), able to perform the same calculation as A' if

$$t(i) \bmod k' <> t(j) \bmod k'; \quad i,j \in [(q-1)k'+1,\ldots,qk']; \quad i <> j \qquad q \in [1...k^*] \qquad (1)$$

where t(i) is the cycle in which $PE_i$ of A' initiates the first valid operation.

For different periodicities ($\neq 1$) this rule may be enunciated in a similar way, by modifying condition (1).

The described condition guarantees that k' adjacent PEs of SA A' initiate valid operations in k' different cycles. Accordingly, the $PE_q$ of A* may perform, via coalescing, the calculations performed in A' by processing elements $PE_{(q-1)k'+1},\ldots,PE_{qk'}$, without any conflict in the utilization of functional units.

The structures of each PE in A*, the form of L*, R*, E* and S* matrices, and the form of vector P*(i) are obtained through a simple algorithm; details are omitted here for reasons of space, but may be found in [7]. The number of PEs in the new SA is w*=w/k'; the new slow is k*=1; periodicity is equal to 1 for all operations. Input sequences to each $PE_q$ of A* are obtained by interleaving the input sequences to PEs of A' whose task is performed now by $PE_q$ (figure 2). Output sequences are obtained in a similar way. Detailed specification of these sequences is also presented in [7].

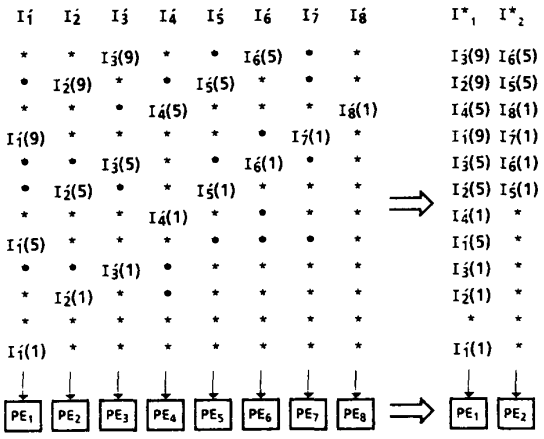


Figure 2. Example of sequences interleaving when applying rule 2 (coalescing) for k´ = 4, and w = 8.

## TWO-LEVEL PIPELINED 1D SA WITH DATA CONTRAFLOW FOR TRIANGULAR SYSTEMS OF EQUATIONS

In this section a two-level pipelined 1D SA with data contraflow, which attains maximum hardware utilization, is to be obtained. This SA is designed by using transformation rules 1 and 2 of previous section. These transformations are applied here to SA A of section 2.2. SA A only has one specified level of pipelining, and all the operations show an equal time cost. Nevertheless, in the attained design, some implementation features which are taken into account include: pipelining of functional units of each PE, periodicity and different calculation time intervals for every operation performed by each PE. The first applied transformation allows to pass from A to A´; A´ includes the above mentioned implementation features. For that reason, in the new SA A´, matrix L´ and vector P´ are predetermined. Afterwards, the second transformation is applied and we pass from A´ to A* by means of rule 2. So, A´ must satisfy the required condition to be guaranteed if rule 2 is used. Now, we have obtained a SA in which every PE is maximally utilized.

That is, we have now the problem of how to obtain a D matrix and a value c, in order to achive the SA A´ through application of rule 1.

In the considered example, conditions to be satisfied by matrix D and by value c can be expresed as:

$$c+d_i-d_{i+1} \geq \max(1, l'(i,i+1)), \quad i \in [1...w-1] \qquad (a.1)$$

$$c+d_{i+1}-d_i \geq \max(1, l'(i+1,i)), \quad i \in [2...w-1] \qquad (a.2)$$

$$(w+1)c+d_w-d_1 \geq \max(1, l'(w,1)) \qquad (a.3)$$

$$c+d_2-d_1 \geq \max(1, l'^1(2,1), l'^2(2,1)) \qquad (a.4)$$

$$(w+i-2)c+d_i \geq 0, \quad i \in [1...w] \qquad (b.1)$$

$$d_w \geq 0 \qquad (b.2)$$

Because t(i) = (w+i-2)c+di;

$$((w+i-2)c+d_i) \bmod k' <> ((w+j-2)c+d_j) \bmod k' \qquad (c)$$

$i,j \in [(q-1)k'+1,...,qk']$; $i \neq j$, $q \in [1,...,w/k']$; and k' = 2c

Conditions (a) and (b) determine that the obtained SA A´ satisfies the implementation constrictions imposed by L´ and P´. More in detail, conditions (a) establish that the number of cycles elapsed from the begining of a calculation in one PE, and the arrival of its result to the destination PE must be greater than or equal to the number of cycles needed for the production of the data item in the PE. This value must be equal to 1, at least, to avoid global communication requeriments. Conditions (b), in the other hand, establish that the number of cycles elapsed from the begining of the operation in A´ and the arrival to any PE of the first data item coming from any one of their input flows, is a positive number. (Obviously, the contrary has no physical sense). Condition (c) serves to that rule 2 may be applied to A´.

### SA implementation example.

Let us assume that we have the following characteristics of a certain implementation of a 1D spiral SA with data contraflow, used to solve triangular systems of linear equations:

- Multipliers and adders, used in the design of functional units are pipelined with respectively m and a stages. Consequently, one multiplication requires m cycles and one addition requires a cycles.

- To perform divisions, the divisor inversion algorithm is used [8]: Q = A/B ≈ -AR(2+RB); where R is an approximation to -1/B, obtained by indexing a table with some bits of B. This calculation requires one access to the table, besides 3 multiplications and 1 addition. Two of these multiplications can be performed in parallel. Consecuently, if we use 2 multipliers and 1 adder, the number of required cycles to perform one division is 2m+a, if we negleet the time to access the table.

About the SA A´ which we are looking for, the following relations are known, in this case:

$l'(i,i+1) = m+a$, $i \in [1...w-1]$; $l'(i+1,i) = 0$, $i \in [2...w-1]$
$l'(w,1) = m+a$; $l'^1(2,1) = 2m+a$; $l'^2(2,1) = 0$
L'(i,j) = 0 in the other cases.

P'(i) = 1, $i \in [2...w]$; P'(1) = (1,1).

One posible set of values $d_i$ and c, satisfying conditions (a), (b) and (c) is :

$d_1 = c(w-1)-w+1-2m-a$; $d_j = c(w-i)-w+i-1$, $i \in [2...m+a+1]$;

$d_i = c(w-i)-w+i$, $i \in [m+a+2...w]$; $c = (3m+2a)/2$ ;

As the found solution satisfies condition (c), we are able to transform the k´-slow SA A´ into a 1-slow SA A* (figure 3) with w*=w/k´ PEs, and where the PEq of A* performs those operations performed by $PE_{(q-1)k'+1},...,PE_{qk'}$ of A´. Figure 4 shows, for instance, the internal structure of $PE_1$ of A* in the considered example. This $PE_1$ performs the same operations which were performed by $PE_1,...,PE_{3m+2a}$ of SA A´. The selection signals of multiplexors are generated from a module 3m+2a counter and from the external signal $C_1$*. In figure 5 the internal structure of any one of the remaining PEs of A*, can be seen.

The input sequence to each PE of A* is obtained by interleaving the input sequences of k´ consecutive PEs (k´= 3m+2a) of A´. More precisely:

$$m(q) = \min_{h=(q-1)k'+1}^{qk'} t(i)$$

$A^*_q(t+1) = A'_r(hk'+1)$ if $t-t(r)+m(q) = hk'$, $q \in [1...w^*]$

$B^* = B'$; $X_e^* = X_e'$; $C_1^* = C_1'$; $C_m^* = C_m'$; $X_s^* = X_s'$

The computational time required to solve in this SA a triangular system of equations with N unknows, becomes:

$$T=\left[\frac{3m+2}{2}\left[\frac{N^2}{(3m+2a)w^*}+N+2(3m+2a)w^*-4\right]-(3m+2a)+1\right]tc$$

where tc is one cycle time.

As an example, if our implementation has w*=10, m=a=3, and $t_c$ = 100ns, a number of 500 triangular systems of equations with 600 unknowns each, can be solved in approx. 1.22 seconds.

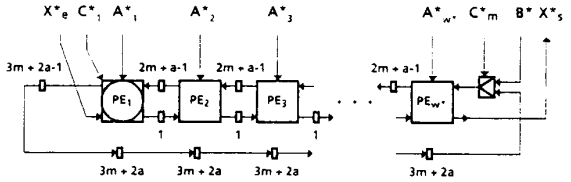


Figure 3. Array structure of A*.

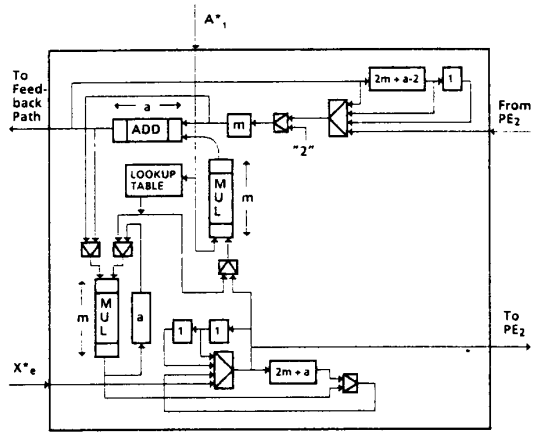


Figure 4. $PE_1$ internal structure of A*.

## CONCLUSIONS

In this work we show how the use of a mathematical model and the application of adequate transformations rules, allows the design of new efficient SAs.

Starting from simple SAs which are not well suited for a real implementation , we obtain SAs where practical features are taken into account, namely: two levels of pipelining, fixed number of processing elements, unequal computational times for different operations, etc.

The proposed design technique has been applied in this paper to get the design of a problem-size-independent, computational balanced and two-level pipelined 1D SA with data contraflow to solve triangular systems of equations.

This methodology can be easily generalized for any type of 1D as well as 2D SAs and is suitable for any interconnection topology.

## REFERENCES


[1] H.T. Kung and C.E. Leiserson, "Systolic Arrays (for VLSI)", Sparse Matrix Proc. 1978, 1979, Society for Industrial and Applied Mathematics (SIAM), pp. 256-282.

[2] J.A.B. Fortes, S.Y. Fu, and B.W. Wah, "Systematic Approaches to the Design of Algorithmically Specified Systolic Array Arrays",Proc. Int'l Conf. Acoustics, Speech, and Signal Processing, 1985, pp. 300-303.

[3] J.J. Navarro, J.M. Llaberia, and M. Valero, "Partitioning:An Essential Step in Mapping Algorithms into Systolic Array Processors", Computer, Vol.20, No. 7, July 1987, pp. 77-89.

[4] H.T. Kung, and M. Lam, "Wafer-Scale Integration and Two-Level Pipelined Implementations of Systolic Arrays",J. Parallel and Distributed Processing. Vol. 1, No. 1, August 1984, pp. 32-63.

[5] H.T. Kung, and W.T. Lin, "An Algebra for Systolic Computation", Elliptic Problem Solvers II, Academic Press 1984, pp. 141-160.

[6] C.E. Leiserson, and J.B. Saxe, "Optimizing Synchronous Systems", Proc. of the 22nd Annual Symp. on Foundations of Computer Science, October 1981, pp. 23-36.

[7] M. Valero-Garcia, J.M. LLaberia, J.J. Navarro, "Considering implementations features in the design of systolic array processors", Internal report, RR 88/01, Facultad Informática de Barcelona.

[8] Floating Point Division/ Square Root/ IEEE Arithmetic WTL 1032/1033 , Application Note, Weitek, 1983.


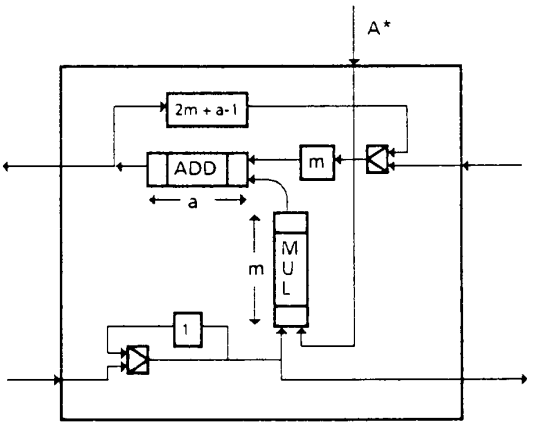


Figure 5. Internal structure of $PE_i$ for i = 2 to w* of A*.