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


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. Frequently, in these systolic algorithms practical considerations are not taken into account. Equitably 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, adequate 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 characterized 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 equitably as possible, in order to obtain that the PEs perform useful task the maximum amount of time attainable.

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 straightforward.

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: \( X(i,j) = x^{X(i,j)} \) 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

\[
L(i,j) = z^{-l^2(i,j)} \cdot r^2(i,j) ...
\]

\[
R(i,j) = z^{-r(i,j)} \cdot r^2(i,j) ...
\]

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 \( l^2(i,j) \) is the number of cycles needed by PE \( j \) to calculate, by means of \( OP_n \), the value to be sent to PE \( i \). Similary, \( r^2(i,j) \) is the delay associated to the link between PE \( j \) and PE \( i \), when PE \( j \) performs \( OP_n \).

\( E \) (entrance matrix) has \( w \)-by-\( p \) elements, and \( e(i,j) \) is the number of cycles from the beginning of SA operation until the PE \( j \) receives the first data item in the \( I_j \) sequence. If PE \( j \) 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 beginning of SA operation until PE \( j \) initiates the calculation of the first data item in the \( O_j \) sequence. If PE \( j \) 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 beginning of an operation in PE \( i \) until the moment in which PE \( i \) may initiate the next operation. If PE \( j \) exhibits a non-homogeneous time behaviour, we shall write \( P(i)=(p_1(i), p_2(i), ..., p_q(i)) \), where \( p_q(i) \) is the periodicity of operation \( OP_q \). Finally, the value of integer \( k \) indicates the slow of the SA.

Modelling of a 1D SA to solve linear equations.

In figure 1 we show the structure of a special type of 1D SAs, which we name 1D 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.](image_url)
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 PE1, because it performs one division and one sign change during some cycles, but a multiplication followed by one addition at other cycles. This PE1 receives a control signal (C1) which can be treated as another input flow. C1 determines, in each cycle, which one of the two possible operations to be performed in the PE. All other PEs, except PE1, always perform one multiplication followed by one addition. The model is as follows:

\[
I(i, i+1) = O(i), \quad i \in [1, w-1]; \\
I(w, i) = I(2, i), \quad i \in [2, w-1]; \\
I(w, 1) = I(2, 1), \quad i = 0;
\]

because, in the original design, the assumption of zero cycles to perform any operation is made. On the other hand, because all the links between PEs have an associated delay equal to 1, except the link between PE1 and PEw, which has a delay of value w + 1.

We denote the input flows to the SA as follows:

\[
I_1 = A_1, \quad i \in [1, w]; \\
I_{w+1} = B_1, \quad i \in [1, w]; \\
I_{2w} = C_1, \quad i \in [1, w]; \\
I_{2w+4} = C_m.
\]

We denote the only output flow as: \( O_1 = X_i \).

The \( A_1, B, X, C_1, C_m \) and \( X_1 \) 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:

\[
e(1, i) = w + 2, \quad i \in [1, w]; \\
e(1, w + 1) = 3w - 1; \\
e(1, w + 2) = w - 1; \\
e(1, w + 3) = w - 1; \\
e(1, w + 4) = 4; \\
e(1, w + 5) = 0 \quad \text{in the other cases}.
\]

\[
s(w, 1) = -2w + 1; \\
s(2w, 1) = 0 \quad \text{in the other cases}.
\]

The periodicity vector can be specified as:

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

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 presented 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 allowed. Rule is:
Nevertheless, in the attained design, some implementation features are predetermined. Afterwards, the second transformation is applied to the first transformation allows to pass from time intervals for every operation performed by each PE. The functional units of each PE, periodicity and different calculation features which are taken into account include: pipelining 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 achieve 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 expressed as:

\[ c+d_1-d_{i+1} \geq \max\{1,l'(i,i+1)\}, \quad i \in \{1\ldots w-1\} \]  
(a.1)  
\[ c+d_{i+1}-d_{i} \geq \max\{1,l'(i+1,i)\}, \quad i \in \{2\ldots w-1\} \]  
(a.2)  
\[ (w+1)c+d_w-d_0 \geq \max\{1,l'(w,1)\} \]  
(a.3)  
\[ c+d_2-d_1 \geq \max\{1,l'(2,1), l'(2,1)\} \]  
(a.4)  
\[ (w+i-2)c+d_i \geq 0, \quad i \in \{1\ldots w\} \]  
(b.1)  
\[ d_w \geq 0 \]  
(b.2)  

Because \( t(i) = (w+i-2)c+d_i \)  
\( (w+i-2)c+d_i \mod k' \ll (w+i-2)c+d_i \mod k' \)  
(c)  
\[ i,j \in \{q-1\ldots k',qk'\}; \quad i+j, \quad q \in \{1\ldots wk'\}; \]  
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 beginning 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 requirements. Conditions (b), in the other hand, establish that the number of cycles elapsed from the beginning 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. Consequently, if we use 2 multipliers and 1 adder, the number of required cycles to perform one division is 2m+a, if we neglect the time to access the table.

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

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

\[ p(i) = 1, \quad i \in \{2\ldots w\}; \quad P(1) = (1,1). \]

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

\[ d_1 = c(w-1)-w+2m-a; \quad d_j = c(w-j)-w+i, \quad i \in \{2\ldots m+a+1\}; \quad d_i = c(w-j)+w+i, \quad i \in \{m+a+2\ldots w\}; \quad c = (3m+2a)/2 \]

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 PE of A* performs those operations performed by \( PE_{q-1<k',1\ldots k'} \) of A'. Figure 4 shows, for instance, the internal structure of \( PE_1 \) of A* in the considered example. This PE performs the same operations which were performed by \( PE_1 \ldots PE_{3m+2a} \) 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:

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 beginning 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 requirements. Conditions (b), in the other hand, establish that the number of cycles elapsed from the beginning 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. Consequently, if we use 2 multipliers and 1 adder, the number of required cycles to perform one division is 2m+a, if we neglect the time to access the table.

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

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

\[ p(i) = 1, \quad i \in \{2\ldots w\}; \quad P(1) = (1,1). \]

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

\[ d_1 = c(w-1)-w+2m-a; \quad d_j = c(w-j)-w+i, \quad i \in \{2\ldots m+a+1\}; \quad d_i = c(w-j)+w+i, \quad i \in \{m+a+2\ldots w\}; \quad c = (3m+2a)/2 \]

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 PE of A* performs those operations performed by \( PE_{q-1<k',1\ldots k'} \) of A'. Figure 4 shows, for instance, the internal structure of \( PE_1 \) of A* in the considered example. This PE performs the same operations which were performed by \( PE_1 \ldots PE_{3m+2a} \) 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:
The computational time required to solve in this SA a triangular system of equations with $N$ unknowns, becomes:

$$T = \left(\frac{3m+2}{2}\right) \frac{N^2}{(3m+2a)w_*} + N + 2(3m+2a)w_* - 4 - (3m+2a) + 1 \cdot t_c$$

where $t_c$ is one cycle time.

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

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

REFERENCES


