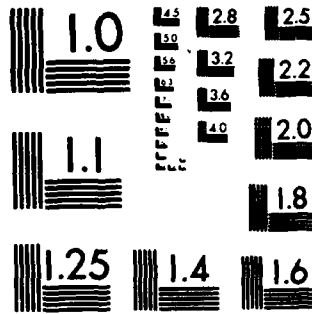


UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE
H T KUNG ET AL. APR 83 CMU-CS-84-100 N00014-76-C-0370

UNCLASSIFIED

F/G 12/1 NL

END
DATE
FILMED
8 84
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A142 796

An Algorithm for VLSI Algorithm Design

H. T. Kang and W. T. Liu

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

April, 1983

DEPARTMENT

OF COMPUTER SCIENCE

DTIC
SERIALS
SECTION

CMU-CS-84-100

An Algebra for VLSI Algorithm Design

H. T. Kung and W. T. Lin

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

April, 1983

An early version of this paper is to appear in the *Proceedings of
Conference on Elliptic Problem Solvers*, Monterey, California, January 1983.

This research was supported in part by the Office of Naval Research under Contracts N00014-76-C-0370, NR 044-422 and N00014-80-C-0236, NR 048-659. W. T. Lin is with the Electrical Engineering Department of Carnegie-Mellon University.

Abstract

Algorithms designed for VLSI implementation are usually parallel and two-dimensional in the sense that many processing elements laid out on a silicon surface can operate simultaneously. These algorithms have been typically described by graphs or networks where nodes represent processing elements or registers and edges represent wires. Although for many purposes these traditional representations are adequate for specifying VLSI algorithms, they are not suited for manipulating algorithm designs. In this paper an algebraic representation, together with a semantics, is proposed for VLSI algorithm designs. By algebraic transformations analogous to some typically used in linear algebra, alternative but equivalent designs satisfying desirable properties such as locality and regularity in data communication can be derived. This paper describes this powerful algebra for manipulating designs, and provides a mathematical foundation for the algebraic transformations. The algebraic framework is more suitable for supporting formal manipulation on designs than the network or graph-theoretic models, especially for complex designs. As an application of the proposed algebra, the paper demonstrates its use in the design and verification of systolic algorithms.

Accession For	
AAI	<input checked="" type="checkbox"/>
DAB	<input type="checkbox"/>
Unpublished	<input type="checkbox"/>
Classification	
Distribution	
Availability	
Availability	
Dist	Special
AI	

Table of Contents

1. Introduction	1
2. Basic Principles and Notation —illustrated by a FIR Filtering Example	2
2.1. FIR Filtering and z-Notation	2
2.2. Systolic FIR Filtering and z-Graph Representation	3
2.3. Algebraic Representation of Design	5
2.4. Deriving Systolic Designs by Algebraic Transformations	7
3. Foundation for Algebraic Transformations	10
3.1. Semantics of Design	10
3.2. Canonical Algebraic Representation	11
3.3. Well-Defined Design and Equivalent Designs	11
3.4. Fundamental Results	13
4. Determining Algebraic Transformations	17
5. IIR Filtering— A Further Example	18
6. Concluding Remarks	22
References	24

List of Figures

Figure 2-1. Design S (straightforward design).	2
Figure 2-2. Design S in the z-notation.	3
Figure 2-3. Design W2: systolic FIR filtering array (a) and cell (b).	3
Figure 2-4. Design W2 in the z-notation	4
Figure 2-5. Design W2 in the z-graph representation.	4
Figure 2-6. Design W2 in the graph representation.	4
Figure 2-7. Design C— a variant of design S of Figure 2-2.	5
Figure 2-8. Design C in the z-graph representation.	5
Figure 2-9. Design C in the algebraic representation.	6
Figure 2-10. Design W2 in the algebraic representation.	7
Figure 2-11. Design corresponding to (2.10) and (2.11) in the z-graph representation.	9
Figure 3-1. Semantics of basic design constructs.	10
Figure 5-1. Straightforward design for the IIR filter in the z-notation.	18
Figure 5-2. IIR filter in the z-graph representation.	18
Figure 5-3. IIR filter in the algebraic representation.	19
Figure 5-4. K -slowed IIR filter in the algebraic representation.	19
Figure 5-5. Systolic IIR filter in the z-graph representation.	21
Figure 6-1. Designs for band matrix multiplication in the z-graph representation: (a) a non-systolic design, and (b) a systolic design.	23

1. Introduction

Over the past several years, many systolic algorithms have been proposed as solutions to computation-bound problems (see, e.g., [6, 10, 12, 14]). By exploiting the regularity and parallelism inherent to given problems and by employing high degrees of parallelism and pipelining, systolic algorithms implemented in VLSI achieve high performance with regular communication structures and low I/O requirements (see [12] for detailed discussions of advantages of systolic structures). A number of prototype machines for implementing systolic algorithms, ranging from single-purpose chips [5, 9, 15], through application-oriented yet programmable systems [2, 23], to very general systems with reconfigurable interconnections [3, 19, 20], have been designed and built. More recently, building-block chips for systolic architectures have also been proposed or designed [8, 1, 18, 22], including the CMU programmable systolic chip (PSC) [7, 8]. The general question of automatically deriving systolic arrays and verifying their correctness, however, remains open, although several significant attempts have been made in this direction (see, e.g., [4, 16, 17, 21]). Instead of suggesting methods for deriving or verifying systolic designs, we provide in this paper an algebra for manipulating VLSI algorithm designs in general. With this algebra a designer is able to manipulate designs by "pushing symbols," in order to conveniently meet desirable design criteria such as locality and regularity of data communication.

Section 2 illustrates the notation and basic principles by considering the hardware implementation of a finite impulse response (FIR) filter. Two representations are proposed to specify a design with the property that from either representation we can derive the other. The *z-graph representation* is close to a hardware or VLSI specification of a design, and the *algebraic representation* is convenient for performing algebraic transformations on a design. Starting with a design that corresponds directly to the mathematical definition of the filtering problem (and thus its correctness is obvious), we perform a set of algebraic transformations on its algebraic representation and obtain the algebraic representation of a systolic design, from which a systolic filtering array can be derived automatically. Section 3, the heart of this paper, provides a mathematical foundation for the algebraic transformations used in Section 2. These transformations are formally justified with respect to a proposed semantics for design. Once justified, they become "legal" transformations that can be applied freely to any design without impairing correctness. Section 4 presents another application of the algebra, namely, the derivation of a systolic infinite impulse response (IIR) filtering array. The last section contains some concluding remarks.

2. Basic Principles and Notation —Illustrated by a FIR Filtering Example

To illustrate the basic idea and notation of this paper, this section considers a concrete example—the FIR filtering problem. We will use many diagrams to make the presentation as clear as possible, although algebraic transformations of this paper rely only on the algebraic representation. We will perform algebraic transformations formally here and postpone their justification to Section 3.

2.1. FIR Filtering and z-Notation

Consider the following FIR filter with weights w_j :

$$y_i = w_1 x_i + w_2 x_{i+1} + w_3 x_{i+2} + w_4 x_{i+3}. \quad (2.1)$$

Figure 2-1 depicts a straightforward design, called design S, for the hardware implementation of the filter. In the diagram, each \otimes and \oplus represent a multiplier and adder, respectively and each \square or \square represents a register capable of latching incoming data for one cycle time. Note that the cycle time must be long enough to allow data flow from register to register, possibly performing some computations in between. One of the objectives of systolic designs is to minimize the cycle time by avoiding long communications and large numbers of computations done inside each cycle, and thus maximize the throughput of the resulting system.

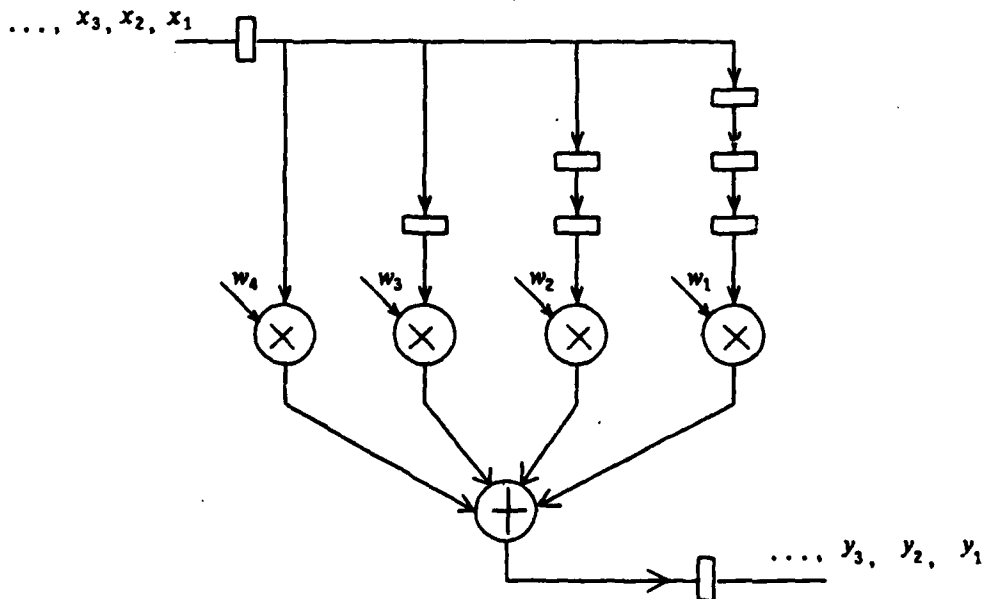


Figure 2-1. Design S (straightforward design).

Figure 2-2 describes design S (ignoring the input and output registers \square) with the usual z-notation, where a delay of k cycles is indicated by z^{-k} . We see that in the z-notation the minimum cycle time is the time to

perform all the operations connected by edges with label z^{-0} . Thus for design S the cycle time is at least the time to perform one multiplication (assuming that four hardware multipliers are available) and one 4-input addition. In the next section we show a systolic design for which only one multiplication and one 2-input addition will have to be done in each cycle.

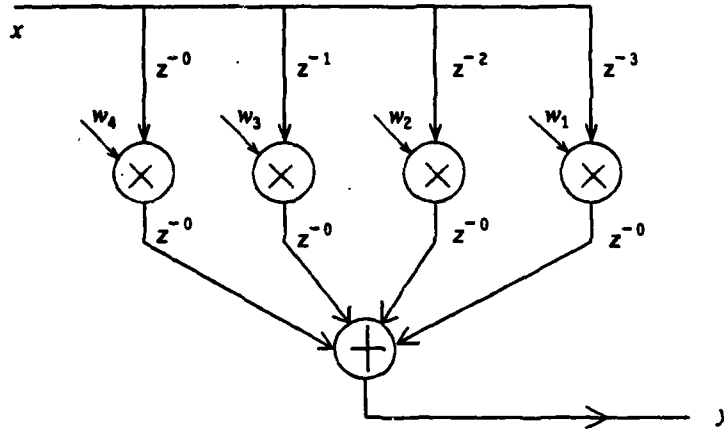


Figure 2-2. Design S in the z -notation.

2.2. Systolic FIR Filtering and z -Graph Representation

Figure 2-3 depicts a typical systolic design for FIR filtering, called design W2 in [12]. In this design the w_i stay and x_i and y_i both move systolically from left to right, but the x_i move twice as slowly as the y_i .

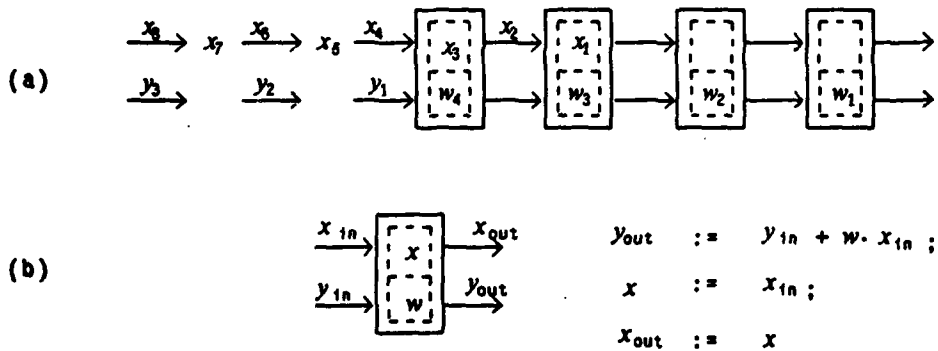
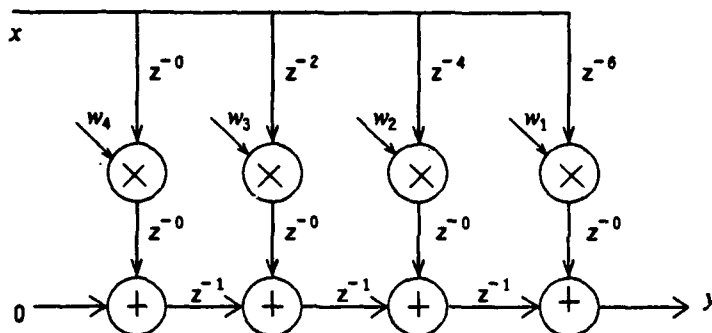
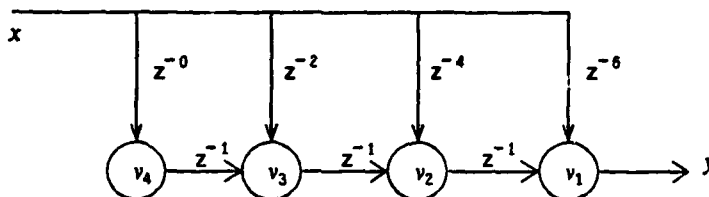


Figure 2-3. Design W2: systolic FIR filtering array (a) and cell (b).

Note that each x value passes from cell to cell without changing. Figure 2-4 depicts the systolic array in the z -notation.

Figure 2-4. Design W2 in the z -notation.Figure 2-5. Design W2 in the z -graph representation.

By grouping every pair of multiplication and addition as one node to be executed by a separate processor, we derive the z -graph representation of the design (Figure 2-5). The z -graph representation of a systolic design has the "systolic property" that the input (the x in Figure 2-5) is distributed to all the nodes (v_1, v_2, v_3, v_4) at different time instants and edges between nodes have labels z^{-k} with $k \geq 1$. One of objectives of this paper is to introduce an algebra for deriving designs whose z -graph representations will have the systolic property (see Section 4 below for precise conditions for a systolic design). Given a design like Figure 2-5, whose z -graph representation enjoys the systolic property, a corresponding systolic array design is readily obtained by simply passing the input x through the nodes with appropriate delays as depicted in Figure 2-6. It is instructive to examine the correspondence between Figure 2-3 and 2-6.

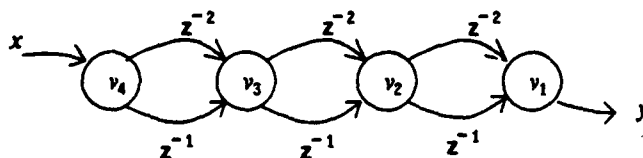


Figure 2-6. Design W2 in the graph representation.

2.3. Algebraic Representation of Design

In this and next sections we show that the systolic design W2 of the preceding section can be derived systematically by algebraic transformation analogous to some typically used in linear algebra. Our starting design is design C of Figure 2-7, which is a variant of the straightforward design, design S, of Figure 2-2. In design C the summation is distributed over a cascade of four 2-input adders as shown in Figure 2-7. Figure 2-8 describes design C in the z-graph representation.

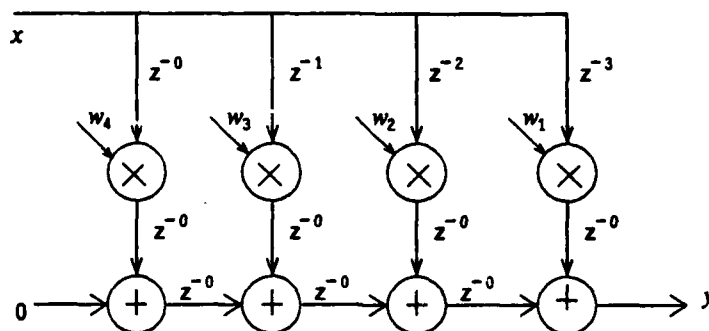


Figure 2-7. Design C—a variant of design S of Figure 2-2.

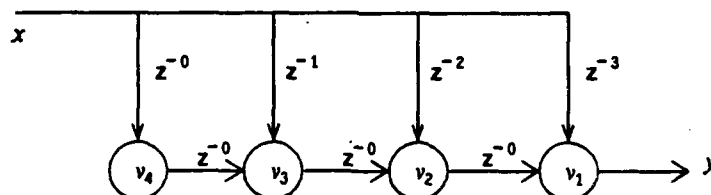


Figure 2-8. Design C in the z-graph representation.

Design C relies on the fact that in the filter computation (2.1) there are as many multiplications as additions. Similar designs apply to many other inner-product-like computations of this kind. Note that in design C of Figure 2-8 the edges linking nodes v_1, v_2, v_3 and v_4 all have labels z^{-0} and therefore the cycle time must be long enough to perform computations associated with all the nodes in sequence. Thus design C is not systolic. Assuming that design C in the z-graph representation (Figure 2-8) is given, our task is to transform it to the systolic design, design W2, of Figure 2-5 by linear algebra techniques. To this end, we formally associate the z-graph representation of design C of Figure 2-8 with an algebraic representation shown in Figure 2-9. To see the correspondence between the two representations, consider for example that

$$v_2 \leftarrow z^{-0}v_3 + z^{-2}x, \quad (2.2)$$

and

(2.3)

$$y = z^{-0} v_1.$$

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \leftarrow \begin{bmatrix} 0 & z^{-0} & 0 & 0 \\ 0 & 0 & z^{-0} & 0 \\ 0 & 0 & 0 & z^{-0} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} + \begin{bmatrix} z^{-3} \\ z^{-2} \\ z^{-1} \\ z^{-0} \end{bmatrix} x$$

$$y = \begin{bmatrix} z^{-0} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}$$

Figure 2-9. Design C in the algebraic representation.

Consistently with Figure 2-8, (2.2) states that at any time t , the value of node v_2 , $v_2(t)$, depends on the values of node v_3 at time t , $v_3(t)$, and the value of input x at time $t-2$, $x(t-2)$, and (2.3) states that the value of output y is the same as the value of node v_1 at any time. More precisely,

$$v_2(t) = f_2[v_3(t), x(t-2)], \quad (2.4)$$

where f_2 is a 2-variable function associated with v_2 such that

$$f_2[a, b] = a + w_2 b.$$

This defines one-to-one correspondence between the z -graph representation of a design and its algebraic representation, in the sense that from either representation one can derive the other. Note that the plus sign in (2.2) represents some combination of information by (2.4) rather than the usual arithmetic addition. In Section 3.2 below semantics for algebraic expressions involving the " \leftarrow " symbol such as (2.2) will be given.

It is readily seen from Figure 2-5 that the algebraic representation of design W2 is that shown in Figure 2-10.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \leftarrow \begin{bmatrix} 0 & z^{-1} & 0 & 0 \\ 0 & 0 & z^{-1} & 0 \\ 0 & 0 & 0 & z^{-1} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} + \begin{bmatrix} z^{-6} \\ z^{-4} \\ z^{-2} \\ z^{-0} \end{bmatrix} x$$

$$y = \begin{bmatrix} z^{-0} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}$$

Figure 2-10. Design W2 in the algebraic representation.

2.4. Deriving Systolic Designs by Algebraic Transformations

In this section we demonstrate that the algebraic representation of the systolic design, design W2, can be obtained from that of design C through formal algebraic transformations; in the next section we will provide a mathematical foundation for these transformations. To simplify notation, we denote the algebraic representation of design C by

$$v \leftarrow Av + bx, \quad (2.5)$$

$$y = c^T v, \quad (2.6)$$

where matrix A and vectors b, c are defined according to Figure 2-9. Consider the diagonal matrix

$$D = \begin{bmatrix} z^{-3} & 0 & 0 & 0 \\ 0 & z^{-2} & 0 & 0 \\ 0 & 0 & z^{-1} & 0 \\ 0 & 0 & 0 & z^{-0} \end{bmatrix}$$

and its "formal" inverse

$$D^{-1} = \begin{bmatrix} z^3 & 0 & 0 & 0 \\ 0 & z^2 & 0 & 0 \\ 0 & 0 & z^1 & 0 \\ 0 & 0 & 0 & z^0 \end{bmatrix}.$$

Let

$$u = Dv. \quad (2.7)$$

Then

$$v = D^{-1}u. \quad (2.8)$$

Multiplying (2.5) by D, we have

$$Dv \leftarrow DA v + D b x. \quad (2.9)$$

By (2.7) and (2.8), (2.9) and (2.6) become

$$u \leftarrow (D A D^{-1})u + (D b)x, \quad (2.10)$$

and

$$y = (c^T D^{-1})u, \quad (2.11)$$

respectively. Through formal calculation, one can check that

$$D A D^{-1} = \begin{bmatrix} 0 & z^{-1} & 0 & 0 \\ 0 & 0 & z^{-1} & 0 \\ 0 & 0 & 0 & z^{-1} \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$D b = \begin{bmatrix} z^{-6} \\ z^{-4} \\ z^{-2} \\ z^{-0} \end{bmatrix}, \quad \text{and} \quad c^T D^{-1} = [z^3 \ 0 \ 0 \ 0].$$

Thus (2.10) and (2.11) are the algebraic representation of the design whose z-graph representation is shown in Figure 2-11.

We have transformed design C of Figure 2-8 to the design of Figure 2-11. After renaming the value of

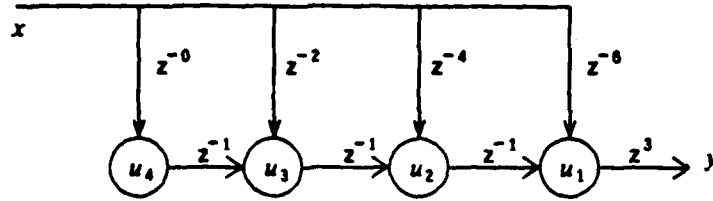


Figure 2-11. Design corresponding to (2.10) and (2.11) in the z -graph representation.

output y at time t to be that of output y at time $t+3$, the design becomes exactly the systolic design W2 of Figure 2-5. In conclusion, we have derived a systolic design by applying a transformation D to the algebraic representation of a non-systolic design.

3. Foundation for Algebraic Transformations

In Section 2.4 we illustrated that a systolic design could be derived by formal algebraic manipulations similar to those used in linear algebra. This section provides a mathematical foundation for these formal manipulations. To do so, we first need to give a semantics for VLSI algorithm design.

3.1. Semantics of Design

We define the semantics of a design to be a function of time that the design implements. More precisely, the semantics of some basic design constructs given in either the z-graph representation or the algebraic representation are summarized in the table of Figure 3-1 with the following comments:


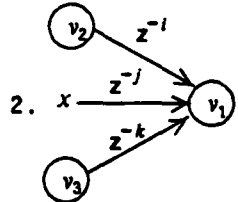
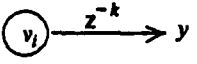
SYNTAX		SEMANTICS
z-graph representation	algebraic representation	
1. 	v_i	v_i is a function of time defined in terms of some associated function f_i .
2. 	$v_1 \leftarrow z^{-i} v_2 + z^{-j} x + z^{-k} v_3$	Function v_1 is defined by $v_1(t) = f_1[v_2(t-i), x(t-j), v_3(t-k)]$.
3. 	$y = z^{-k} v_i$	$y(t) = v_i(t-k)$.

Figure 3-1. Semantics of basic design constructs.

1. Each node v_i in the z-graph representation or each variable v_i in the algebraic representation is a function of time defined in terms of some *implicit function* f_i associated with v_i .
2. The value of node or variable v_i at time t , $v_i(t)$, is $f_i[v_2(t-i), x(t-j), v_3(t-k)]$, where $v_2(t-i)$ is the value of v_2 at time $t-i$, $x(t-j)$ is the value of input x at time $t-j$, and $v_3(t-k)$ is the value of v_3 at time $t-k$.
3. The value of output y at time t is the same as the value of v_i at time $t-k$. (If $k=0$, symbol z^{-k} can be omitted from the z-graph representation as Figures 2-5 and 2-8.)

Note that for designs of Figures 2-8 and 2-11, implicit function f_i , $i=1,2,3$, associated with node v_i or u_i with weight w_i , is defined by

$$f_i[a,b]=a+w_i b,$$

and implicit function f_4 associated with node v_4 or u_4 is defined by

$$f_4[b]=w_4 b,$$

where a and b are the left and top inputs to the node, respectively. Note that implicit functions f_i are functions independent of time. As far as the algebraic transformations of this paper are concerned, the semantics of implicit functions need not be specified, as they are invariant under these transformations. This is the reason why we call them implicit functions.

3.2. Canonical Algebraic Representation

As shown in Figures 2-5 and 2-8, a general design in the z -graph representation has input x , output y and nodes v_1, \dots, v_n . By grouping multiple expressions for defining individual functions v_1, \dots, v_n into a single matrix expression, the algebraic representation of a general VLSI algorithm design often has the form:

$$v \leftarrow Av + bx, \quad (3.1)$$

$$y = c^T v, \quad (3.2)$$

where $A=(z^{-a}ij)$ is an $n \times n$ matrix, $b=(z^{-b_1}, \dots, z^{-b_n})^T$, $v=(v_1, \dots, v_n)^T$, and $c^T=(z^{-c_1}, \dots, z^{-c_n})$ with only one nonzero entry. This *canonical form* of algebraic representation has been illustrated by Figures 2-9 and 2-10, and will be assumed in the rest of the paper except the concluding remarks section.

3.3. Well-Defined Design and Equivalent Designs

For $i=1, \dots, n$, the i -th component of (3.1) is

$$v_i \leftarrow z^{-a_{i1}} v_1 + z^{-a_{i2}} v_2 + \dots + z^{-a_{in}} v_n + z^{-b_i} x. \quad (3.3)$$

That is, (3.1) is a collection of expressions (3.3) for $i=1, \dots, n$. For defining the semantics of design (3.1) and (3.2), (3.3) means that function v_i satisfies

$$v_i(t) = f_i\{v_1(t-a_{i1}), v_2(t-a_{i2}), \dots, v_n(t-a_{in}), x(t-b_i)\} \quad (3.4)$$

for some implicit function f_i associated with node v_i , and (3.2) means that

$$y(t) = v_j(t-c_j),$$

where $-c_j$ is the exponent of the only nonzero entry in vector c^T . (Mechanically, we can think that in the transformation from (3.3) to (3.4) " \leftarrow " is replaced with " $= f_i$."") Here we use the convention that a zero entry of A , b or c^T is $z^{-\infty}$ and it is omitted from expressions (3.3) and (3.4).

We say that a design is *well-defined* starting from some t_0 , if for $i=1, \dots, n$ and $t \geq t_0$, $v_i(t)$ is completely determined by values in the sets $\{x(t'): t' \leq t\}$ and $\{v_i(t'): t' < t\}$, $i=1, \dots, n$, and this property holds for *any* implicit functions. In view of (3.4) a sufficient condition for design (3.1) to be well-defined is that a_{ij} 's are all positive. This is, however, not a necessary condition. It is instructive to see that design C of Figure 2-9 is well-defined in spite of the fact that for this design $a_{12} = a_{23} = a_{34} = 0$. From Figure 2-9, we have

$$v_1(t) = f_1[v_2(t), x(t-3)],$$

$$v_2(t) = f_2[v_3(t), x(t-2)],$$

$$v_3(t) = f_3[v_4(t), x(t-1)],$$

$$v_4(t) = f_4[x(t)].$$

Therefore

$$v_1(t) = f_1[f_2[f_3[f_4[x(t)], x(t-1)], x(t-2)], x(t-3)],$$

$$v_2(t) = f_2[f_3[f_4[x(t)], x(t-1)], x(t-2)],$$

$$v_3(t) = f_3[f_4[x(t)], x(t-1)],$$

$$v_4(t) = f_4[x(t)].$$

We see that for $i=1, \dots, 4$, $v_i(t)$ is completely determined by values in the set $\{x(t'): t' \leq t\}$ for any implicit functions f_i , and thus design C is well-defined. It is easy to prove that a sufficient and necessary condition for a design to be well-defined is that in its z-graph representation there does not exist any cycle whose edges all have label z^{-0} . Verifying this condition for a design can be done in linear time. Hereafter we are only interested in designs that are well-defined.

Consider a well-defined design (3.1), with some implicit function associated with each node. Given an input function (of time) x and initial values $v_i(t)$ for $t < t_0$, by (3.4) design (3.1) defines a unique vector function (of time) $v = (v_1, \dots, v_n)^T$,* and together with (3.2), defines a unique output function (of time) y . We say two output functions A and B are essentially the same if $A(t) = B(t + \alpha)$, where α is some constant, for all t greater than certain time instant.

Definition 3.1: Two given designs are *equivalent*, if for any initial values given for one design, there exist initial values for the other design such that with the same input function the two designs produce essentially the same output function.

In the following section we will show that design defined by (2.5) and (2.6) and one defined by (2.10) and (2.11) are equivalent.

* In the semantics literature, function v such defined is called the "fixpoint solution" of "fixpoint equation" (3.1).

3.4. Fundamental Results

To express our results on algebraic transformations, we need the following definitions. Let $D=(z^{-d})$ be an $n \times n$ diagonal matrix.

1. For $v=(v_1, \dots, v_n)^T$, define Dv to be $u=(u_1, \dots, u_n)^T$ such that for $i=1, \dots, n$,

$$u_i(t) = v_i(t-d)$$

for all t for which $v_i(t-d)$ is defined. Thus, D can be viewed as an operator that maps a vector function v to another vector function Dv .

2. For $b=(z^{-b_1}, \dots, z^{-b_n})^T$, define Db to be $e=(z^{-e_1}, \dots, z^{-e_n})^T$ where

$$e_i = d_i + b_i$$

for $i=1, \dots, n$.

3. Let $A=(z^{-a_{ij}})$ be an $n \times n$ matrix. Define DA to be an $n \times n$ matrix $B=(z^{-b_{ij}})$ where

$$b_{ij} = d_i + a_{ij}$$

for $i, j=1, \dots, n$. Product AD is defined similarly. We can easily check that

$$(DA)D^{-1} = D(AD^{-1}),$$

and thus we can simply denote them by DAD^{-1} .

Here we use the convention that

$$\infty = d_i + \infty$$

for any d_i . Thus zero entries of b or A remain to be zero entries in Db or DA , respectively.

Lemma 3.1: Suppose that v and u are defined by well-defined designs

$$v \leftarrow Av + bx \tag{3.5}$$

and

$$u \leftarrow (DAD^{-1})u + (Db)x, \tag{3.6}$$

with their initial values satisfying

$$u_i(t+d_i) = v_i(t) \tag{3.7}$$

for $t < t_0$. Then

$$u = Dv.$$

Proof: Let v_i and u_i be the i -th components of v and u , respectively. Note that $DAD^{-1}=(z^{-d_i+d_j-a_{ij}})$ and $Db=(z^{-d_1-b_1}, z^{-d_2-b_2}, \dots, z^{-d_n-b_n})^T$. Thus, u_i defined by (3.6) satisfies

$$u_i(t) = f_i[u_1(t-d_i+d_1-a_{1n}), u_2(t-d_i+d_2-a_{2n}), \dots, \\ u_n(t-d_i+d_n-a_{in}), x(t-b_i-d_i)].$$

Replacing t with $t+d_i$ in the above equation, we have

$$u_i(t+d_i) = f_i[u_1(t+d_1-a_{1n}), u_2(t+d_2-a_{2n}), \dots, \\ u_n(t+d_n-a_{in}), x(t-b_i)]. \quad (3.8)$$

By (3.4),

$$v_i(t) = f_i[v_1(t-a_{1n}), v_2(t-a_{2n}), \dots, v_n(t-a_{in}), x(t-b_i)]. \quad (3.9)$$

We prove by induction on i that for $i=1, \dots, n$,

$$u_i(t+d_i) = v_i(t) \quad (3.10)$$

for $t = t_0, t_0+1, t_0+2, \dots$. By (3.7), (3.10) holds for $t < t_0$. Thus,

$$u_j(t_0+d_j-a_{jn}) = v_j(t_0-a_{jn})$$

for any j for which $a_{jn} > 0$. Since designs (3.5) and (3.6) are well-defined, (3.8) and (3.9) imply that

$$u_j(t_0+d_j) = v_j(t_0)$$

that is, (3.10) holds for $t = t_0$. By induction (3.10) holds for $t = t_0+1, t_0+2, \dots$, and so on. \square

The following lemma can be proven by a similar method:

Lemma 3.2: If

$$y = c^T v \quad \text{and} \quad u = Dv,$$

then

$$y = (c^T D^{-1})v.$$

Immediately following from Lemmas 3.1 and 3.2, we have the following result:

Theorem 3.1: Design

$$v \leftarrow Av + bx,$$

$$y = c^T v$$

is equivalent to design

$$u \leftarrow (DAD^{-1})u + (Db)x,$$

$$y = (c^T D^{-1})v,$$

assuming that both designs are well-defined.

The above theorem is essentially the "retiming lemma" of Leiserson and Saxe [17]. Not using the algebraic notation and approach taken here, they had to rely on a very long (4 pages) and rather unclear proof.

In the following, we introduce another transformation whose function is to scale down the throughput of an existing design. Consider a well-defined design M with input function x and output function y , and another design M' with input function x' and output function y' . We say that design M' is a k -slowed design of M for some positive integer k , if the following holds for some integer p :

for any initial values for M , there exist initial values for M' such that if

$$x'(kt+p) = x(t)$$

for all t , then

$$y'(kt+p) = y(t)$$

for all t where $y(t)$ is defined.

Therefore as far as the outside world is concerned, the function of a k -slowed design is the same as that of the original design, except that input and output are taken in and out, respectively, once every k time units. The usefulness of k -slowed designs in the derivation of systolic designs was first pointed out in [17], and it will become clear in the next two sections. The following lemma shows a simple way to implement a well-defined, k -slowed design.

Lemma 3.3: If

$$v \leftarrow Av + bx, \quad (3.11)$$

$$y = c^T v$$

is a well-defined design, then the design

$$v' \leftarrow A'v' + b'x', \quad (3.12)$$

$$y' = c'^T v', \quad (3.13)$$

with $A' = (z^{-ka_1}, \dots, z^{-ka_n})$, $b' = (z^{-kb_1}, \dots, z^{-kb_n})^T$, and $c'^T = (z^{-kc_1}, \dots, z^{-kc_n})$ is a well-defined, k -slowed design.

Proof: Since in their z -graph representations the two designs have the same set of edges with label z^{-k} , well-definedness of one design implies that of the other. Let v_i and v'_i be the i -th components of v and v' , respectively. Without loss of generality, assume that the output functions y and y' of the two designs satisfy

$$y(t) = v_i(t - c_i), \quad \text{and} \quad y'(t') = v'_i(t' - kc_i),$$

respectively. Suppose that the original design is well-defined starting from t_0 . It suffices to prove that if

$$x'(kt) = x(t)$$

for all t , and

$$v_i'(kt) = v_i(t) \tag{3.14}$$

for $i=1, \dots, n$ and $t < t_0$, then

$$y'(kt) = y(t)$$

for all t for which $y(t)$ is defined. The proof is similar to that of Lemma 3.1 and is omitted.

4. Determining Algebraic Transformations

Given a well-defined design, we want to determine a k -slowed design and $D=(z^{-d_i})$ such that design

$$u \leftarrow (DA'D^{-1})u + (Db')x',$$

$$y' = (c'^T D^{-1})u$$

will be well-defined and systolic. This imposes the following conditions on the entries of $DA'D^{-1}$ and Db' :

C1. For $i=1, \dots, n$,

$$d_i + ka_{ij} - d_j \geq 1.$$

{This assures not only that the design is well-defined, but also that the cycle time only has to be long enough to perform the computation of at most one node.}

C2. All nonzero entries of any column of $DA'D^{-1}$ and Db' must be distinct.

{This assures that the value of a node at any time never has to be sent to more than one node simultaneously, and thus no broadcasting or fanout of data is needed.}

It is an easy exercise to show that if the original design is well-defined, that is, in its z -graph representation there does not exist cycles whose edges all have label z^{-0} , then there exist k and D for which conditions C1 and C2 are satisfied. To maximize throughput we are interested in a solution which has the smallest-possible k . It turns out that for some designs to satisfy C1 and C2, k must be greater than one, as to be illustrated by IIR filtering example in the next section. This is the reason why we perform transformations on a k -slowed design, with $k \geq 1$, rather than the original design.

5. IIR Filtering— A Further Example

Consider the implementation of the following infinite impulse response (IIR) filter with weights w_i :

$$y_i = w_1 y_{i-1} + w_2 y_{i-2} + w_3 x_i + w_4 x_{i-1}. \quad (5.1)$$

The above equation states that at any given time t , the value of output y depends on the values of y at times $t-1$ and $t-2$, and input x at times t and $t-1$. Figure 5-1 depicts a straightforward design for the IIR filter in the z -notation.

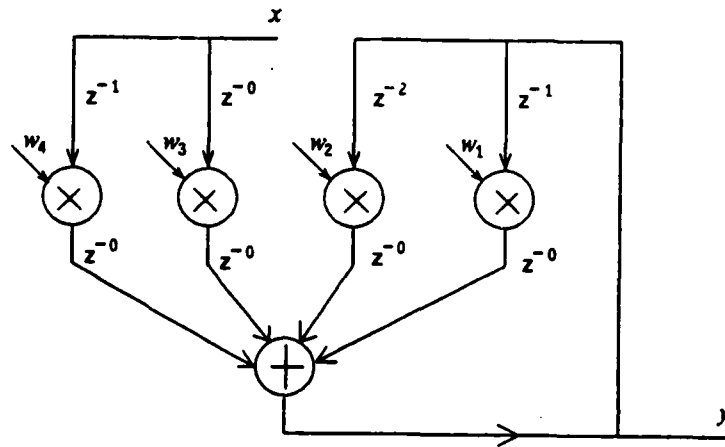


Figure 5-1. Straightforward design for the IIR filter in the z -notation.

Similar to the FIR design of Figure 2-7, the 4-input adder of Figure 5-1 can be distributed over a cascade of four 2-input adders. This forms a design with four identical nodes, whose z -graph representation is depicted in Figure 5-2. Figure 5-3 describes the algebraic representation of the design.

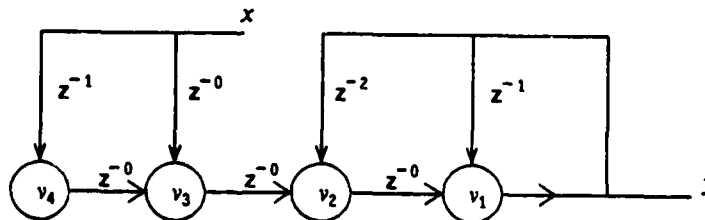


Figure 5-2. IIR filter in the z -graph representation.

According to Lemma 3.3, a k -slowed design can be obtained by changing labels z^{-h} to z^{-kh} for any h . The algebraic representation of the k -slowed IIR filter is described in Figure 5-4, and is denoted by

$$v \leftarrow A'v + b'x,$$

$$y = c'Tv.$$

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \leftarrow \begin{bmatrix} z^{-1} & z^{-0} & 0 & 0 \\ z^{-2} & 0 & z^{-0} & 0 \\ 0 & 0 & 0 & z^{-0} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ z^{-0} \\ z^{-1} \end{bmatrix} x$$

$$y = \begin{bmatrix} z^{-0} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}$$

Figure 5-3. IIR filter in the algebraic representation.

$$\begin{bmatrix} v_1' \\ v_2' \\ v_3' \\ v_4' \end{bmatrix} \leftarrow \begin{bmatrix} z^{-k} & z^{-0} & 0 & 0 \\ z^{-2k} & 0 & z^{-0} & 0 \\ 0 & 0 & 0 & z^{-0} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1' \\ v_2' \\ v_3' \\ v_4' \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ z^{-0} \\ z^{-k} \end{bmatrix} x$$

$$y = \begin{bmatrix} z^{-0} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1' \\ v_2' \\ v_3' \\ v_4' \end{bmatrix}$$

Figure 5-4. K -slowed IIR filter in the algebraic representation.

We seek a diagonal matrix D such that the design described by

$$u \leftarrow (DA'D^{-1})u + (Db')x',$$

$$y = (c'^T D^{-1})u,$$

will be well-defined and systolic. By condition C1 of Section 4,

$$k \geq 1,$$

$$2k + d_2 - d_1 \geq 1,$$

$$d_2 - d_3 \geq 1,$$

$$d_3 - d_4 \geq 1,$$

and by condition C2,

$$k \neq 2k + d_2 - d_1,$$

$$d_3 \neq k + d_4.$$

One can check that a solution with the minimum-possible value for k is that $k=2$ and

$$D = \begin{bmatrix} z^{-2} & 0 & 0 & 0 \\ 0 & z^{-1} & 0 & 0 \\ 0 & 0 & z^{-0} & 0 \\ 0 & 0 & 0 & z^1 \end{bmatrix}.$$

Note that

$$DA'D^{-1} = \begin{bmatrix} z^{-2} & z^{-1} & 0 & 0 \\ z^{-3} & 0 & z^{-1} & 0 \\ 0 & 0 & 0 & z^{-1} \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$Db' = \begin{bmatrix} 0 \\ 0 \\ z^{-0} \\ z^{-1} \end{bmatrix}, \quad \text{and} \quad c'^T D^{-1} = [z^2 \ 0 \ 0 \ 0].$$

Thus the resulting systolic IIR filtering array in the z -graph representation is shown in Figure 5-5. This systolic array was previously described in [11].

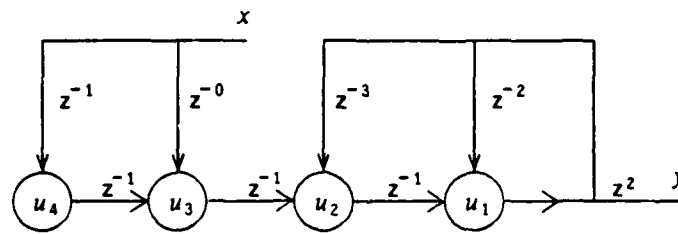


Figure 5-5. Systolic IIR filter in the z-graph representation.

6. Concluding Remarks

We proposed two representations for specifying a design—the z-graph representation and the algebraic representation. From either representation we can derive the other. The z-graph representation is readily mappable to a hardware or VLSI implementation, whereas the algebraic representation is suitable for algebraic transformations. For algebraic transformations, only algebraic representations of designs are needed. By working within an algebraic framework, rather than a network or graph-theoretic framework, one can use powerful algebraic operators to manipulate designs and can deal with abstraction conveniently. For example, using matrix notation, a simple algebraic expression such as (3.1) can represent design of arbitrary size.

A more general algebraic representation than the one described in (3.1) and (3.2) is:

$$v \leftarrow Av + Bx, \quad (6.1)$$

$$y = C^T v, \quad (6.2)$$

where input x and output y are vectors rather than scalars, and B and C are matrices rather than vectors b and c . This general form of representation seems to cover all the interesting VLSI algorithm designs that we know of and can anticipate. For example, for the design of Figure 6-1(a) for multiplying a bidiagonal upper triangular matrix with a bidiagonal lower triangular matrix, we have

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ z^{-0} & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & z^{-0} & z^{-0} & 0 \\ z^{-0} & 0 & z^{-1} & 0 \\ 0 & z^{-0} & 0 & z^{-0} \\ z^{-0} & 0 & 0 & z^{-1} \end{bmatrix}.$$

and

$$C^T = \begin{bmatrix} 0 & z^{-0} & 0 & 0 \\ 0 & 0 & 0 & z^{-0} \\ 0 & 0 & z^{-0} & 0 \end{bmatrix}.$$

Without loss of generality we can always assume that there is only one nonzero entry in each row of C^T , that is, at any time the value of each output y_i is equal to that of some node at that time or earlier. Results and definitions of this paper can all be extended in a straightforward way to this general form of the algebraic representation (6.1) and (6.2). For example we can show that starting with the non-systolic design of Figure 6-1(a), a systolic solution with the minimum-possible value for k is that $k=1$ and

$$D = \begin{bmatrix} z^{-0} & 0 & 0 & 0 \\ 0 & z^{-0} & 0 & 0 \\ 0 & 0 & z^{-1} & 0 \\ 0 & 0 & 0 & z^{-1} \end{bmatrix}$$

The resulting systolic array is illustrated in Figure 6-1(b), which is precisely the systolic design for band matrix multiplication proposed in [21]. Detailed discussions of this and other results including the use of the proposed algebra in the derivation of two-level pipelined systolic arrays [13] and systolic arrays for priority queues and LU-decomposition of matrices will appear in forthcoming papers.

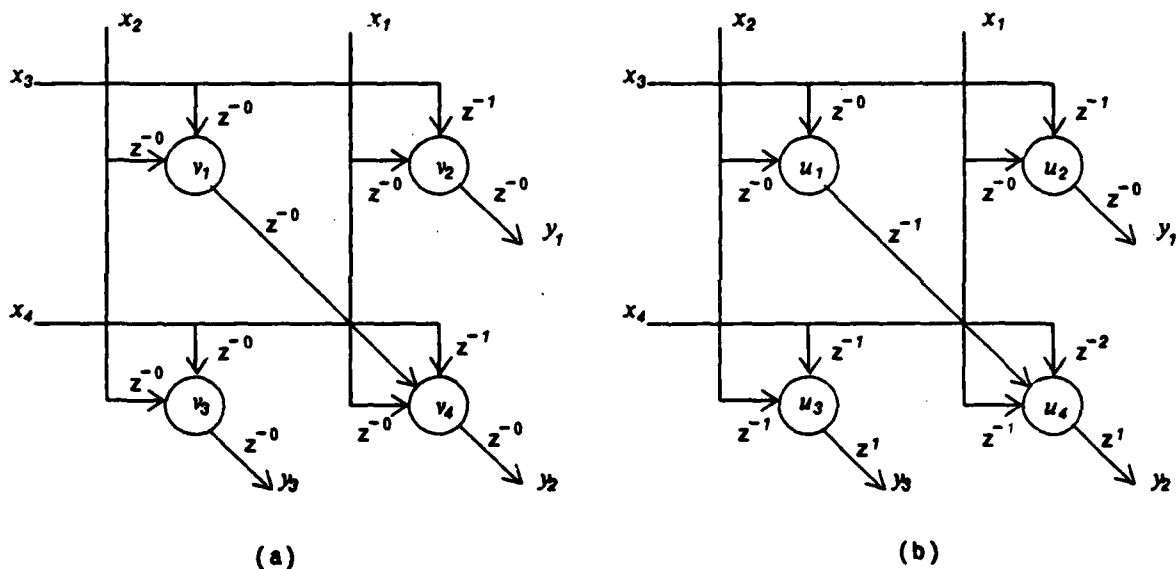


Figure 6-1. Designs for band matrix multiplication in the z-graph representation: (a) a non-systolic design, and (b) a systolic design.

We view that major contributions of this paper are at the proposed semantics for VLSI algorithm design, algebraic representation and transformations, and the mathematical foundation for these transformations. With these algebraic tools, we are able to manipulate designs by "pushing symbols" as we do in algebra, and to prove theorems about design transformations (e.g., Theorem 3.1) without relying on any drawings. Deriving systolic design is just one of many potential applications of the proposed algebra.

Acknowledgments

We want to thank Allan Fisher and Monica Lam for their comments on an early draft of the paper.

References

- [1] Avila, J. and Kuekes, P.
One-Gigaflop VLSI Systolic Processor.
In *Proceedings of SPIE Symposium, Vol. 431, Real-Time Signal Processing VI*, pages 159-165. Society of Photo-Optical Instrumentation Engineers, August, 1983.
- [2] Blackmer, J., Frank, G. and Kuekes, P.
A 200 Million Operations per Second (MOPS) Systolic Processor.
In *Proceedings of SPIE Symposium, Vol. 298, Real-Time Signal Processing IV*, pages 10-18. Society of Photo-Optical Instrumentation Engineers, August, 1981.
- [3] Bromley, K., Symanski, J.J., Speiser, J.M., and Whitehouse, H.J.
Systolic Array Processor Developments.
In Kung, H.T., Sproull, R.F., and Steele, G.L., Jr. (editors), *VLSI Systems and Computations*, pages 273-284. Computer Science Department, Carnegie-Mellon University, Computer Science Press, Inc., October, 1981.
- [4] Chen, M.C. and Mead, C.A.
A Hierarchical Simulator Based on Formal Semantics.
In Bryant, R. (editor), *Proceedings of the Third Caltech Conference on Very Large Scale Integration*, pages 207-223. California Institute of Technology, Computer Science Press, Inc., March, 1983.
- [5] Corry, A. and Patel, K.
A CMOS/SOS VLSI Correlator.
In *Proceedings of 1983 International Symposium on VLSI Technology, Systems and Applications*, pages 134-137. 1983.
- [6] Fisher, A.L. and Kung, H.T.
Special-Purpose VLSI Architectures: General Discussions and a Case Study.
In *VLSI and Modern Signal Processing*. Prentice-Hall, November, 1982.
- [7] Fisher, A.L., Kung, H.T., Monier, L.M. and Dohi, Y.
Architecture of the PSC: A Programmable Systolic Chip.
In *Proceedings of the 10th Annual International Symposium on Computer Architecture*, pages 48-53. June, 1983.
- [8] Fisher, A.L., Kung, H.T., Monier, L.M., Walker, H. and Dohi, Y.
Design of the PSC: A Programmable Systolic Chip.
In Bryant, R. (editor), *Proceedings of the Third Caltech Conference on Very Large Scale Integration*, pages 287-302. California Institute of Technology, Computer Science Press, Inc., March, 1983.
- [9] Foster, M.J. and Kung, H.T.
The Design of Special-Purpose VLSI Chips.
Computer Magazine 13(1):26-40, January, 1980.
Reprint of the paper appears in *Digital MOS Integrated Circuits*, edited by Elmasry, M.I., IEEE Press Selected Reprint Series, 1981, pp. 204-217.
- [10] Kung, H.T.
Let's Design Algorithms for VLSI Systems.
In *Proceedings of Conference on Very Large Scale Integration: Architecture, Design, Fabrication*, pages 65-90. California Institute of Technology, January, 1979.
Also available as a CMU Computer Science Department technical report, September 1979.

- [11] Kung, H.T.
Special-Purpose Devices for Signal and Image Processing: An Opportunity in VLSI.
In *Proceedings of the SPIE, Vol. 241, Real-Time Signal Processing III*, pages 76-84. Society of Photo-Optical Instrumentation Engineers, July, 1980.
- [12] Kung, H.T.
Why Systolic Architectures?
Computer Magazine 15(1):37-46, January, 1982.
- [13] Kung, H.T. Ruane, L.M., and Yen, D.W.L.
Two-Level Pipelined Systolic Array for Multidimensional Convolution.
Image and Vision Computing 1(1):30-36, February, 1983.
An improved version appears as a CMU Computer Science Department technical report, November 1982.
- [14] Kung, H.T. and Leiserson, C.E.
Systolic Arrays (for VLSI).
In Duff, I. S. and Stewart, G. W. (editors), *Sparse Matrix Proceedings 1978*, pages 256-282. Society for Industrial and Applied Mathematics, 1979.
A slightly different version appears in *Introduction to VLSI Systems* by C. A. Mead and L. A. Conway, Addison-Wesley, 1980, Section 8.3, pp. 37-46.
- [15] Kung, H.T. and Song, S.W.
A Systolic 2-D Convolution Chip.
In Preston, K., Jr. and Uhr, L. (editors), *Multicomputers and Image Processing: Algorithms and Programs*, pages 373-384. Academic Press, 1982.
- [16] Lam, M. and Mostow, J.
A Transformational Model of VLSI Systolic Design.
In Uehara, T. and Barbacci, M. (editors), *Proceedings of the 6th International Symposium on Computer Hardware Description Languages and their Applications*, pages 65-77. IFIP, May, 1983.
- [17] Leiserson, C.E. and Saxe, J.B.
Optimizing Synchronous Systems.
Journal of VLSI and Computer Systems 1(1):41-68, 1983.
- [18] Sorasen, O., Solberg, B., and Alker, H.-J.
VLSI Implemented Systolic Array Processor for Vector Processing.
In Anceau, F. and Aas, E.J. (editors), *VLSI '83*, pages 307-316. North-Holland, August, 1983.
- [19] Symanski, J.J.
Systolic Array Processor Implementation.
In *Proceedings of SPIE Symposium, Vol. 298, Real-Time Signal Processing IV*, pages 27-32. Society of Photo-Optical Instrumentation, August, 1981.
- [20] Symanski, J.J.
Progress on a Systolic Processor Implementation.
In *Proceedings of SPIE Symposium, Vol. 341, Real-Time Signal Processing V*, pages 2-7. Society of Photo-Optical Instrumentation, May, 1982.

- [21] Weiser, U. and Davis, A.
A Wavefront Notation Tool for VLSI Array Design.
In Kung, H.T., Sproull, R.F., and Steele, G.L., Jr. (editors), *VLSI Systems and Computations*, pages 226-234. Computer Science Department, Carnegie-Mellon University, Computer Science Press, Inc., October, 1981.
- [22] Weste, N.H.E., Burr, D.J. and Ackland, B.D.
A Systolic Processing Element for Speech Recognition.
In *Proceedings of 1982 IEEE International Solid-State Circuits Conference*, pages 274-275. February, 1982.
- [23] Yen, D.W.L. and Kulkarni, A.V.
Systolic Processing and an Implementation for Signal and Image Processing.
IEEE Transactions on Computers C-31(10):1000-1009, October, 1982.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CMU-CS-84-100	2. GOVT ACCESSION NO. ADA142796	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN ALGEBRA FOR VLSI ALGORITHM DESIGN		5. TYPE OF REPORT & PERIOD COVERED Interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) H.T. Kung , W.T. Lin		8. CONTRACT OR PROGRAM NUMBER(s) N00014-78-C-0370 NR044-422, NR 048-659 N00014-80-C-0236
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Department Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217		12. REPORT DATE April 1983
		13. NUMBER OF PAGES 31
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Appr for public release; distribution unlimited		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-010-6001

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DATE
ILME

