Mapping Computation with No Memory

Serge Burckel[1], Emeric Gioan[2], and Emmanuel Thomé[1]

[1] INRIA-LORIA, France
[2] CNRS-LIRMM, France

Abstract. We investigate the computation of mappings from a set S^n to itself with *in situ programs*, that is using no extra variables than the input, and performing modifications of one component at a time. We consider several types of mappings and obtain effective computation and decomposition methods, together with upper bounds on the program length (number of assignments). Our technique is combinatorial and algebraic (graph coloration, partition ordering, modular arithmetics).

For general mappings, we build a program with maximal length $5n-4$, or $2n-1$ for bijective mappings. The length is reducible to $4n-3$ when $|S|$ is a power of 2. This is the main combinatorial result of the paper, which can be stated equivalently in terms of multistage interconnection networks as: any mapping of $\{0,1\}^n$ can be performed by a routing in a double n-dimensional Beneš network. Moreover, the maximal length is $2n-1$ for linear mappings when S is any field, or a quotient of an Euclidean domain (e.g. $\mathbb{Z}/s\mathbb{Z}$). In this case the assignments are also linear, thereby particularly efficient from the algorithmic viewpoint.

The *in situ* trait of the programs constructed here applies to optimization of program and chip design with respect to the number of variables, since no extra writing memory is used. In a non formal way, our approach is to perform an arbitrary transformation of objects by successive elementary local transformations inside these objects only with respect to their successive states.

Keywords: mapping computation, boolean mapping, linear mapping, memory optimization, processor optimization, program design, circuit design, multistage interconnection network, butterfly, rearrangeability.

1 Introduction

The mathematical definition of a mapping $E : S^n \to S^n$ can be thought of as the parallel computation of n assignment mappings $S^n \to S$ performing the mapping E, either by modifying at the same time the n component variables, or mapping the n input component variables onto n separate output component variables. If one wants to compute sequentially the mapping E by modifying the components one by one and using no other memory than the input variables whose final values owerwite the initial values, one necessarily needs to transform the n mappings $S^n \to S$ in a suitable way. We call *in situ computation* this way of computing a mapping, and we prove that it is always possible with a

C.S. Calude et al. (Eds.): UC 2009, LNCS 5715, pp. 85–97, 2009.

number of assignments linear with respect to n and a small factor depending on the mapping type. The impact of these results should be both practical and theoretical.

To be formal and to avoid confusions, let us already state a definition. For the ease of the exposition, we fix for the whole paper a finite set S of cardinal $s = |S|$, a strictly positive integer n and a mapping $E : S^n \to S^n$.

Definition 1. An *in situ program* Π of a mapping $E : S^n \to S^n$ is a finite sequence $(f^{(1)}, i^{(1)})$, $(f^{(2)}, i^{(2)})$, ..., $(f^{(m)}, i^{(m)})$ of *assignments* where $f^{(k)} : S^n \to S$ and $i^{(k)} \in \{1, ..., n\}$, such that every transformation $X = (x_1, ..., x_n) \mapsto E(X)$ is computed by the sequence of successive modifications

$$X := (x_1, ..., x_{i^{(k)}-1}, f^{(k)}(X), x_{i^{(k)}+1}, ..., x_n), \qquad k = 1, 2, ..., m$$

where $f^{(k)}$ modifies only the $i^{(k)}-$th component of X. In other words, every assignment $(f^{(k)}, i^{(k)})$ of an in situ program performs the elementary operation

$$x_{i^{(k)}} := f^{(k)}(x_1, ..., x_n).$$

The *length* of Π is the number m. The *signature* of Π is the sequence $i^{(1)}, i^{(2)}, ..., i^{(m)}$.

All in situ programs considered throughout this paper operate on consecutive components, traversing the list of all indices, possibly several times in forward or backward order. Thus program signatures will all be of type: $1, 2, ..., n-1, n, n-1, ..., 2, 1, 2, ...n-1, n, ...$ For ease of exposition, we shorten the above notations the following way: the mappings $S^n \to S$ corresponding to assignments in the several traversals will be simply distinguished by different letters, e.g. f_i denotes the mapping affecting the variable x_i on the first traversal, g_i the one affecting x_i on the second traversal, and so on, providing an in situ program denoted $f_1, f_2, ...,\ f_{n-1}, f_n, g_{n-1}, ..., g_2, g_1, ...$ For instance, a program f_1, f_2, g_1 on S^2 represents the sequence of operations: $x_1 := f_1(x_1, x_2), x_2 := f_2(x_1, x_2), x_1 := g_1(x_1, x_2)$.

As a preliminary example, consider the mapping $E : \{0, 1\}^2 \to \{0, 1\}^2$ defined by $E(x_1, x_2) = (x_2, x_1)$ consisting in the exchange of two boolean variables. A basic program computing E is: $x' := x_1$, $x_1 := x_2$, $x_2 := x'$. An in situ program f_1, f_2, g_1 of E avoids the use of the extra variable x', with $f_1(x_1, x_2) = f_2(x_1, x_2) = g_1(x_1, x_2) = x_1 \oplus x_2$.

Our results can be seen as a far reaching generalization of this classical computational trick. The motivations for this work will now be detailed. They are of three types : technological, combinatorial and algorithmic.

First, a permanent challenge in computer science consists in increasing the performances of computations and the speed of processors. A computer decomposes a computation in elementary operations on elementary objects. For instance, a 32 bits processor can only perform operations on 32 bits, and any

transformation of a data structure must be decomposed in successive operations on 32 bits. Then, as shown in the above example on the exchange of the contents of two registers, the usual solution to ensure the completeness of the computation is to make copies from the initial data. But this solution can generate some memory errors when the structures are too large, or at least decrease the performances of the computations. Indeed, such operations involving several registers in a micro-processor, through a compiler or an electronic circuit, will have to make copies of some registers in the cache memory or in RAM, with a loss of speed, or to duplicate signals in the chip design itself, with an extra power consumption.

On the contrary, the theoretical solution provided by in situ computation would avoid the technological problems alluded to, and hence increase the performance. We point out that theoretical and combinatorial approaches such as ours are found fruitful in the context of chip design in many electronic oriented publications, see for instance [10] for further references. A short note on our methods intended for an electronic specialist audience has already been published [4]. Further research in this direction (also related to algorithmic questions, see below) would be to develop applications in software (compilers) and hardware (chip design).

From the combinatorial viewpoint, the assignments, which are mappings $S^n \to S$, can be considered under various formalisms. For example, in *multistage interconnection networks*, an assignment is regarded as a set of edges in a bipartite graph between S^n and S^n where an edge corresponds to the modification of the concerned component.

Multistage interconnection networks have been an active research area over the past forty years. All the results of the paper can be translated in this context, since making successive modifications of consecutive components of $X \in S^n$ is equivalent to routing a butterfly network (i.e. a suitably ordered hypercube) when $S = \{0, 1\}$, or a generalized butterfly network with greater degree for an arbitrary finite set S (see Section 2). In the boolean case, the existence of an in situ program with $2n - 1$ assignments for a bijective mapping is equivalent to the well known [2] rearrangeability of the Beneš network (*i.e.* of two opposite butterflies), that is: routing a Beneš network can perform any permutation of the input vertices to the output vertices. Rearrangeability is a powerful tool in network theory. And we mention that butterfly-type structures also appear naturally in recursive computation, for example in the implementation of the well-known FFT algorithm [7], see [9].

First, we state such a rearrangeability result extended to an arbitrary finite set S (see Theorem 1, which is presumably not new). Next, we address the problem of routing a general arbitrary mapping instead of a permutation, which is a non-trivial and new extension. A general result is obtained involving $5n - 4$ mappings (see Corollary 2). Then, the main combinatorial result of the paper, on boolean mappings (see Theorem 3), proposes a more involved answer to this problem. An equivalent statement is the following: any mapping of $\{0, 1\}^n$ is performed by a routing in a double n-dimensional Beneš network.

From the algorithmic viewpoint, building assignments whose number is linear in n to perform a mapping of S^n to itself is satisfying in the following sense. If the input data is an arbitrary mapping $E : S^n \to S^n$ with $|S| = s$, given as a table of $n \times s^n$ values, then the output data is a linear number of mappings $S^n \to S$ whose total size is a constant times the size of the input data. This means that the in situ program of E has the same size as the definition of E by its components, up to a multiplicative constant. This complexity bound is essentially of theoretical interest, since in terms of effective technological applications, it may be difficult to deal with tables of $n \times s^n$ values for large n. Hence, it is interesting to deal with an input data given by algebraic expressions of restricted size, like polynomials of bounded degree for instance, and compare the complexity of the assignments in the output data with the input one. This general question (also related to the number of gates in a chip design) is motivating for further research (examples are given in [4]).

Here, we prove that, in the *linear* case, i.e. if the input is given by polynomials with degree at most 1, with respect to any suitable algebraic structure for S (e.g. any field, or $\mathbb{Z}/s\mathbb{Z}$), then the assignments are in number $2n-1$ and overall are also linear (see Theorem 4). Hence, we still obtain a program whose size is proportional to the restricted size of the input mapping. This result generalizes to a large extent the result in [6] obtained for linear mappings on the binary field.

We will also discuss the complexity of the algorithms that build the in situ programs, which is not the same complexity problem. In the linear case, the decomposition method takes $O(n^3)$ steps.

This paper is organized as follows. Section 2 investigates the link between in situ programs (as defined by definition 1) and multistage interconnection networks. Subsequent sections of the paper give reformulations of the presented results in both settings. In Section 3, we prove that every bijective, resp. general, mapping E on S^n is computed by a sequence of $2n-1$, resp. $5n-4$, assignments. In Section 4, we improve the bound $5n-4$ to $4n-3$ in the case $S = \{0, 1\}$. In Section 5, we consider a set S with an algebraic suitable ring structure. We prove that every linear mapping E on S^n is computed by a sequence of $2n-1$ linear assignments.

2 Multistage Interconnection Networks

A *multistage interconnection network*, or *MIN* for short, is a directed graph whose set of vertices is a finite number of copies $S_1^n, S_2^n, \ldots, S_k^n$ of S^n, called *columns*, and whose edges join elements of S_i^n towards some elements of S_{i+1}^n for $1 \leq i < k$. Then *routing* a MIN is specifying one outgoing edge from each vertex of S_i^n for $1 \leq i < k$. A mapping E of S^n is *performed* by a routing of a MIN if for each element $X \in S_1^n$ there is a directed path using specified edges from X to $E(X) \in S_k^n$. The *gluing* of two MINs M, M' is the MIN $M|M'$ obtained by identifying the last column of M and the first column of M'.

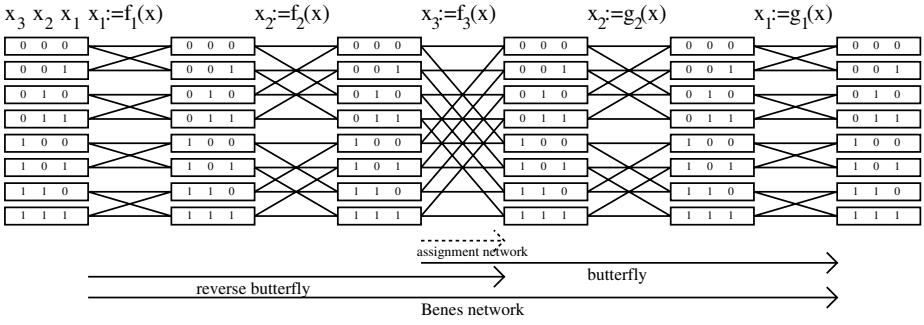

Fig. 1.

The *assignment network* A_i is the MIN with two columns whose edges join $(x_1, \ldots, x_n)$ to $(x_1, \ldots, x_{i-1}, e, x_{i+1}, \ldots, x_n)$ for an arbitrary $e \in S$. Hence each vertex has degree $s = \mid S \mid$. With notations of Definition 1, given an assignment $(f^{(k)}, i^{(k)})$ in an in situ program, we naturally define a routing of $A_{i^{(k)}}$ by specifying the edge between $X = (x_1, \ldots, x_n)$ and $(x_1, ..., x_{i^{(k)}-1}, f^{(k)}(X), x_{i^{(k)}+1}, ..., x_n)$. Hence, the modifications made by the assignments of the program are represented by this routing in the columns of this MIN. Denote $R_n^{(s)}$ the MIN $A_1|A_2|\ldots|A_n$ and $B_n^{(s)}$ the MIN $A_n|\ldots|A_2|A_1$. The usual *butterfly*, also called indirect binary cube, stands here as $B_n^{(2)}$. The *Beneš network* is the network obtained from $R_n^{(2)}|B_n^{(2)}$ by replacing the two consecutive assignment networks A_n by a single one. Note that this last reduction is not part of the usual definition, however it is more convenient here since two successive assignments on a same component can always be replaced with a single one. Note also that the historical definition of a Beneš network [2] is not in terms of butterflies, but that ours is topologically equivalent thanks to classical results (see [1] and [3] for instance), and hence they are equivalent in terms of mappings performed.

From the above definitions, an in situ program of signature $i^{(1)}, \ldots, i^{(m)}$ corresponds to a routing in $A_{i^{(1)}}|\ldots|A_{i^{(m)}}$. Figure 1 gives an example for the Beneš network, with corresponding in situ program f_1, f_2, f_3, g_2, g_1 (with shortened notation). Routing this network is exactly specifying these mappings.

3 Bijective and General Mappings on Finite Sets

The classical property of the Beneš network is that it is *rearrangeable*, that is for any permutation of $\{0,1\}^n$, there exists a routing performing the permutation (note that a routing performs a permutation when it defines disjoint directed paths). This result corresponds to the particular case $S = \{0,1\}$ of the next theorem, providing the rearrangeability property for a Beneš-type network with out-degree generalized from 2 to s. This generalization is presumably not new but the authors did not find such a statement in the literature.

Theorem 1. *Let E be a bijective mapping on S^n. There exists an in situ program for E of length $2n-1$ and signature $1 \dots n \dots 1$. Equivalently, $R_n^{(s)}|B_n^{(s)}$ has a routing performing E.*

Proof. Observe that one can permute the ordering of the variables in the above statement and obtain in situ programs with other signatures. We build such an in situ program $f_n, f_{n-1}, \dots, f_2, f_1, g_2, \dots, g_{n-1}, g_n$ for E by induction on n. For $n = 1$, it is obvious since $X := E(X)$ is computed by the program $x_1 := f_1(x_1)$ with $f_1 = E$. Assume $n > 1$. Let $G = (X, Y, A)$ be the bipartite multi-edges graph defined by: $X = Y = S^{n-1}$, and $(x, y) \in X \times Y$ is in A with label $(x_n, y_n) \in S^2$, if and only if $E(x, x_n) = (y, y_n)$.

Since E is bijective, any vertex of G has exactly degree $s = |S|$. Then the edges of G are colorable with the s elements of S (see [8]). Now, define s mappings $E^0, E^1, \dots, E^{s-1}$ on S^{n-1} and two mappings f_n, g_n from S^n to S as follow. For each color $i \in S$ and every edge (x, y) with color i and label (x_n, y_n), define: $E^i(x) = y$, $f_n(x, x_n) = i$, and $g_n(y, i) = y_n$.

So, after the first step of the program and until the step before last, the component x_n equals a color i. Any mapping E^i being bijective on S^{n-1} is computed by induction in $2(n-1)-1$ steps: $f_{n-1}^i, \dots, f_2^i, f_1^i, g_2^i, \dots, g_{n-1}^i$. Now, define for every $i \in S$ and $x \in S^{n-1}$: $f_{n-1}(x, i) = f_{n-1}^i(x), ..., f_1(x, i) = f_1^i(x), g_2(x, i) = g_2^i(x), ..., g_{n-1}(x, i) = g_{n-1}^i(x)$. After the step before last, we have $x = y$. And after the last step, we have $x_n = y_n$. □

Observe that the computational complexity of our decomposition algorithm for *building* an in situ program for a bijective mapping E on $\{0,1\}^n$ given by a table of $t = n.2^n$ boolean entries is in $DTIME(t.log(t))$. Indeed, defining E_0, E_1 takes $n.2^n$ steps. Then, each E_i is decomposed in E_{i0}, E_{i1} in $(n-1).2^{n-1}$ steps, and so on... The total number of steps is bounded by $n.2^n + 2.(n-1).2^{n-1} + 4.(n-2).2^{n-2} + ... + 2^{n-1}.1.2^1 < 2^n.n^2$.

Corollary 1. *If Π is an in situ program of a bijection E on $\{0,1\}^n$, then the reversed sequence of assignments is an in situ program of the inverse bijection E^{-1}.*

Proof. First, we show that operations in the program Π are necessarily of the form $x_i := x_i + h(x_1, .., x_{i-1}, x_{i+1}, ..., x_n)$. One can assume without loss of generality that $i = 1$. Let $x_1 := f(x_1, ..., x_n)$ be an operation of Π. Denote $h(x_2, \dots, x_n) = f(0, x_2, \dots, x_n)$. We necessarily have $f(1, x_2, \dots, x_n) = 1 + h(x_2, \dots, x_n)$. Otherwise two different vectors would map to the same image. This yields $f(x_1, \dots, x_n) = x_1 + h(x_2, \dots, x_n)$. As a consequence, performing the operations in reverse order will compute the inverse bijection E^{-1}. □

Now, in order to build a program for a general mapping E on S^n, for which different vectors may have same images, we will use a special kind of mappings on S^n, that can be computed with n assignments.

Definition 2. Denote $[s^n]$ the interval of integers $[0, \dots, s^n - 1]$. The *index* of a vector $(x_1, x_2, \dots, x_n)$ is the integer $x_1 + s.x_2 + \dots + s^{n-1}.x_n$ of $[s^n]$. For

every $i \in [s^n]$, denote by X_i the vector of index i. The *distance* of two vectors X_a, X_b is the integer $\Delta(X_a, X_b) = |b - a|$. A mapping I on S^n is *distance-compatible* if for every $x, y \in S^n$, $\Delta(I(x), I(y)) \leq \Delta(x, y)$, which is equivalent to $\Delta(I(X_a), I(X_{a+1})) \leq 1$ for every a with $0 \leq a < s^n - 1$.

Proposition 1. *Every distance-compatible mapping I on S^n is computed by an in situ program $p_1, p_2, \ldots, p_n$. Hence, for $I(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ and for each $i = 1, 2, \ldots, n$: $p_i(y_1, \ldots, y_{i-1}, x_i, \ldots, x_n) = y_i$.*

Proof. Since each component is modified one time, necessarily each function p_i must give its correct final value to each component x_i. It remains to prove that this unique possible method is correct, that is the mappings p_i are well defined by the property above (note that this definition is partial, but sufficient for computing the image of any x). Assume that $p_1, ..., p_i$ are well defined. Assume that, after step i, two different vectors x, x' are given the same image by the process whereas their final expected images $I(x)$ and $I(x')$ were different. The components x_j, $j > i$, of x and x' have not been modified yet. Hence, they are equal and we deduce $\Delta(x, x') < s^i$. On the other hand, the components y_j, $j \leq i$, of $I(x)$ and $I(x')$ are equal but $I(x) \neq I(x')$. Hence $\Delta(I(x), I(x')) \geq s^i$: a contradiction. So p_{i+1} is also well defined by the property above. □

Definition 3. We call *partition-sequence of S^n* a sequence

$$P = (P_0, P_1, \ldots, P_k)$$

of subsets of S^n such that the non-empty ones form a partition of S^n. Then, we denote by I_P the mapping on S^n which maps $X_0, \ldots, X_{s^n-1}$ respectively to

$$\overbrace{X_0, \ldots, X_0}^{|P_0|}, \overbrace{X_1, \ldots, X_1}^{|P_1|}, \ldots, \overbrace{X_k, \ldots, X_k}^{|P_k|}.$$

Observe that I_P is well defined since the sum of sizes of the subsets equals s^n, and that I_P depends only on the sizes of the subsets and their ordering. Observe also that if no subset is empty, then I_P is distance-compatible since, by construction, $\Delta(I(X_a), I(X_{a+1})) \leq 1$ for every a.

Let E be a mapping on S^n, and $P = (P_0, P_1, \ldots, P_k)$ be a partition-sequence of S^n whose underlying partition of S^n is given by the inverse images of E: if $P_i \neq \emptyset$, then $P_i = E^{-1}(y_i)$ for some $y_i \in S^n$. Then, a *P-factorisation* of E is a triple of mappings (F, I, G) on S^n where: G is bijective and maps P_i to $I^{-1}(X_i)$; I is the mapping I_P; and F maps X_i to y_i and is arbitrarily completed to be bijective. By construction

$$E = F \circ I \circ G.$$

Using this construction with no empty subset in the sequence P, we obtain the following theorem, which significantly improves the result of [5] where boolean mappings on $\{0, 1\}^n$ are computed in n^2 steps.

Theorem 2. *For every finite set S, every mapping E on S^n is computed by an in situ program of signature $1 \dots n \dots 1 \dots n \dots 1 \dots n$ and length $5n-4$. Equivalently, $R_n^{(s)}|B_n^{(s)}|R_n^{(s)}|B_n^{(s)}|R_n^{(s)}$ has a routing performing E.*

Proof. Consider any P-factorisation (F, I, G) of E with no empty subset in the sequence P. Then the mapping I is distance compatible. By Theorem 1, G (resp. F) can be computed by a program of signature $1 \dots n \dots 1$ (resp. $n \dots 1 \dots n$). By Proposition 1, I is computed by a program of signature $1 \dots n$. By composition and contracting two successive assignments of the same variable in one, E is computed by a sequence of $5n-4$ assignments of signature $1 \dots n \dots 1 \dots n \dots 1 \dots n$. □

Remark. To end this section, let us remark that, due to the fact that successive assignments operate on consecutive components, successive assignements of type $S^{mn} \to S$ can be grouped in assignments of fewer variables on a larger base set S^m defining successive mappings $S^{mn} \to S^m$:

$$\underbrace{f_{nm}, \dots, f_{n.(m-1)+1}}_{\tilde{f}_n}, \dots, \underbrace{f_m, \dots f_2, f_1, g_2, \dots, g_m}_{\tilde{f}_1}, \dots, \underbrace{g_{n.(m-1)+1}, \dots, g_{nm}}_{\tilde{g}_n}.$$

Hence, for instance, the case $S = \{0,1\}^m$ can be reduced to the case $S = \{0,1\}$. This is a particular case of the register integrability property described in [4].

4 General Mappings on the Boolean Set

In this section, we will fix $S = \{0,1\}$. The more involved method for general boolean mappings is a refinement of the method for general mappings on finite sets and provides a smaller number of assignments. It is valid when $S = \{0,1\}$, and, by extension, when $S = \{0,1\}^m$. We still use a P-factorisation (F, I, G) but the sequence P will possibly contain empty sets, and will be suitably ordered with respect to the sizes of its elements, in order to satisfy some boolean arithmetic properties. So doing, the intermediate mapping $I = I_P$ will have the property that its composition with the first n steps of the in situ program of the bijection F can also be computed with n assignments.

Lemma 1. *Every sequence of 2^n non negative integers whose sum is 2^n can be ordered in a* block-sequence $[v_0, v_1, \dots, v_{2^n-1}]$ *such that, for every $i = 0 \dots n$, the sum of values in consecutive* blocks *of size 2^i is a multiple of 2^i, that is, for all $0 \le j < 2^{n-i}$:*

$$\sum_{j2^i \le l < (j+1)2^i} v_l \ = 0[2^i].$$

Proof. The ordering is built inductively. Begin at level $i = 0$ with 2^n blocks of size 1 having each value in the sequence. At level $i+1$, form consecutive pairs of blocks $[B, B']$ that have values v, v' of same parity and define the value of this new block to be $(v+v')/2$. Each new level doubles the size of blocks and divides their number by 2. The construction is valid since the sum of values of blocks at level i is 2^{n-i}. □

Example. We illustrate below the process described in the proof of Lemma 1 ($n = 4$ and each block has its value as an exponent):

$$[4]^4, [1]^1, [1]^1, [1]^1, [1]^1, [1]^1, [1]^1, [3]^3, [3]^3, [0]^0, [0]^0, [0]^0, [0]^0, [0]^0, [0]^0, [0]^0$$
$$[4,0]^2, [1,1]^1, [1,1]^1, [1,1]^1, [3,3]^3, [0,0]^0, [0,0]^0, [0,0]^0$$
$$[4,0,0,0]^1, [1,1,3,3]^2, [1,1,1,1]^1, [0,0,0,0]^0$$
$$[4,0,0,0,1,1,1,1]^1, [1,1,3,3,0,0,0,0]^1$$
$$[4,0,0,0,1,1,1,1,1,1,3,3,0,0,0,0]^1$$

Definition 4. For a vector $(x_1, \ldots, x_n)$, we call *prefix of order* k, resp. *suffix of order* k, the vector $(x_1, \ldots, x_k)$, resp. $(x_k, \ldots, x_n)$. A mapping I of $\{0,1\}^n$ is called *suffix-compatible* if, for every $1 \leq k \leq n$, if two vectors X, X' have same suffixes of order k, then their images $I(X), I(X')$ also have same suffixes of order k.

Lemma 2. *Let $P = (P_0, P_1, \ldots, P_{2^n-1})$ be a partition-sequence of $\{0,1\}^n$ such that $[|P_0|, |P_1|, \ldots, |P_{2^n-1}|]$ is a block-sequence. Then the mapping I_P on $\{0,1\}^n$ is suffix-compatible.*

Proof. The sketch of the proof is the following. First, define the j-th block of level i of $\{0,1\}^n$ as the set of vectors with index $j2^i \leq l < (j+1)2^i$. Observe that the inverse image by I_P of a block is a union of consectutive blocks of same level. The result follows.

Let us now detail the proof. For $0 \leq i \leq n$ and $j \in [2^{n-i}]$, define the j-th block at level i of $\{0,1\}^n$ as

$$V_{i,j} = \{X_l : l \in [j2^i, (j+1)2^i - 1]\}.$$

(i) First, we prove that, for every i, j as above, there exists $k, k' \in [2^{n-i}]$, such that

$$I_P^{-1}(V_{i,j}) = \bigcup_{k \leq l \leq k'} V_{i,l}.$$

Let us call interval of $\{0,1\}^n$ the set of vectors X_l for l belonging to an interval of $[2^n]$. First, notice that the inverse image by I_P of an interval of $\{0,1\}^n$ is an interval of $\{0,1\}^n$. By definition of I_P, we have $\left|I_P^{-1}(V_{i,j})\right| = \sum_{j2^i \leq l < (j+1)2^i} v_l$. Remark that $I_P^{-1}(V_{i,j})$ may be empty, when $v_l = 0$ for all $l \in [j2^i, (j+1)2^i]$. Since $[v_0, \ldots, v_{2^n-1}]$ is a block sequence, we have $\sum_{j2^i \leq l < (j+1)2^i} v_l = 0 \mod 2^i$. Hence, $\left|I_P^{-1}(V_{i,j})\right| = 0 \mod 2^i$.

For a fixed i, we prove the result by induction on j. If $j = 0$ then $\left|I_P^{-1}(V_{i,0})\right| = k.2^i$ for some $k \in [2^{n-i}]$. If $I_P^{-1}(V_{i,0})$ is not empty, then it is an interval of $\{0,1\}^n$ containing $(0, ..., 0)$ by definition of I_P. Since this interval has a size $k.2^i$ multiple of 2^i, it is of the form $\bigcup_{0 \leq l \leq k} V_{i,l}$.

If the property is true for all l with $0 \leq l < j$, then $I_P^{-1}\left(\bigcup_{0 \leq l < j} V_{i,l}\right) = \bigcup_{0 \leq l \leq j'} V_{i,l}$. Since $\left|I_P^{-1}(V_{i,j})\right| = k.2^i$ for some $k \in [2^{n-i}]$, we must have $I_P^{-1}\left(\bigcup_{0 \leq l \leq j} V_{i,l}\right) = \bigcup_{0 \leq l \leq j'+k} V_{i,l}$, hence $I_P^{-1}(V_{i,j}) = \bigcup_{j' < l \leq j'+k'} V_{i,l}$.

(ii) Now, we prove the Lemma itself.

Assume $a = (a_1, ..., a_n)$ and $b = (b_1, ..., b_n)$ have same suffix of order i. For all $l \geq i$ we have $a_l = b_l$. Let $c \in S^n$ be defined by $c_n = a_n = b_n, \ldots, c_i = a_i = b_i, c_{k-1} = 0, \ldots, c_1 = 0$. Let $\phi(x)$ denote the index of vector x. We have $\phi(c) = 0 \mod 2^{i-1}$, that is $\phi(c) = j.2^{i-1}$ for some $j \in [2^{n-i+1}]$. And $\phi(a)$ and $\phi(b)$ belong to the same interval $[j.2^{i-1}, (j+1).2^{i-1} - 1]$ whose elements have same components for $l \geq i$. That is a and b belong to $V_{i-1,j}$. By (i), the inverse images of intervals of type $V_{i-1,k}$ by I_P are unions of such consecutive intervals. Hence the image of an interval $V_{i-1,j}$ by I_p is an interval contained in an interval $V_{i-1,k}$ for some $k \in [2^{n-i+1}]$. Hence $I_P(a)$ and $I_P(b)$ have same components $l \geq i$. □

Proposition 2. *Let I be a suffix-compatible mapping. Let B be a bijective mapping on $\{0,1\}^n$ computed by a program $b_1, \ldots, b_n$. The mapping $B \circ I$ is computed by a program $p_1, p_2, \ldots, p_n$. Hence, for $B \circ I(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$: $p_i(y_1, \ldots, y_{i-1}, x_i, \ldots, x_n) = y_i$.*

Proof. Just as for Proposition 1, assume that $p_1, ..., p_i$ are well defined by the necessary property above, and that, after step i, two different vectors x, x' are given the same image by the process whereas their final expected images $y = B \circ I(x)$ and $y' = B \circ I(x')$ were different (hence $I(x) \neq I(x')$). By construction, y, y' have a same prefix P of order i and x, x' have a same suffix Q of order $i+1$. Moreover, since I is suffix-compatible, the vectors $I(x), I(x')$ also have a same suffix R of order $i+1$. Let B_i be the mapping computed by $b_1, \ldots, b_i$. We obtain that $B_i(I(x)) = B_i(I(x')) = (P, R)$. Since B_i is necessarily bijective, then $I(x) = I(x')$: a contradiction. □

Now, given a mapping E of S^n, using a P-factorisation of E for a sequence P whose sequence of cardinalities is a block-sequence, we can improve the result of Section 3.

Theorem 3. *Every mapping E on $\{0,1\}^n$ is computed by an in situ program of length $4n-3$ and signature $1 \ldots n \ldots 1 \ldots n \ldots 1$. Equivalently, the gluing of two Beneš networks has a routing performing E.*

Proof. Let (F, I, G) be a P-factorisation of E for a sequence $P = (P_0, P_1, \ldots, P_{2^n-1})$ such that $[|P_0|, |P_1|, \ldots, |P_{2^n-1}|]$ is a block-sequence (it exists thanks to Lemma 1). By Theorem 1, G (resp. F) can be computed by a program of signature $1 \ldots n \ldots 1$ (resp. $1 \ldots n \ldots 1$). By Lemma 2, the mapping $I = I_P$ on $\{0,1\}^n$ is suffix-compatible. Call B the mapping computed by the n first assignments $b_1, ..., b_n$ of the program of F. By Proposition 2, $B \circ I$ is also computed by a program of signature $1 \ldots n$. Then, by composition and contracting two successive assignments of the same variable in one, E is computed by a sequence of $4n-3$ assignments of signature $1 \ldots n \ldots 1 \ldots n \ldots 1$. □

Example. Figure 2 gives an example for the construction of this section. The elements of $\{0,1\}^3$ are grouped by the bijection G at column 6, accordingly with the block sequence $[1,3,2,2]$ induced by E^{-1}. Then, at column 9 all elements

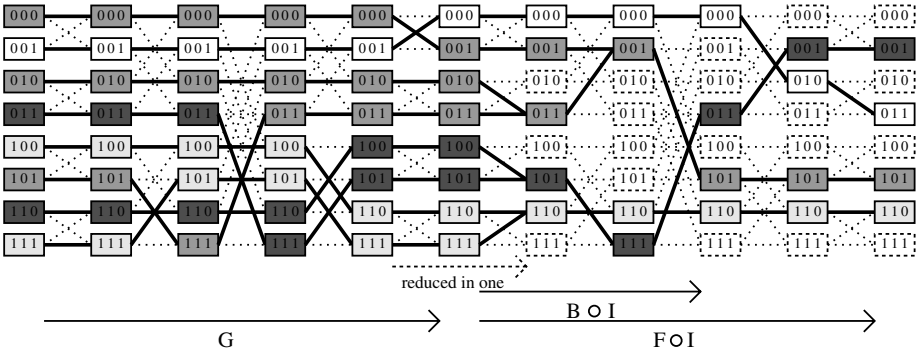

Fig. 2.

with same final image have been given a same image by $B \circ I$. At last, the second part of the bijection F allows to finalize the mapping E.

Remark. Finally, to illustrate the importance of block-sequences and of $S = \{0, 1\}$ for this section, let us give two small significant examples.

First, consider the partition-sequence $P = (\{00\}, \{10, 01\}, \{11\})$ of $\{0, 1\}^2$ whose sequence of cardinalities $(1, 2, 1, 0)$ is not a block-sequence. The mapping I_P is not suffix-compatible, since $I_P(01) = 10$ and $I_P(11) = 01$ have distinct x_2 coordinate whereas 01 and 11 have same x_2 coordinate.

Second, the result of Lemma 1 building a block-sequence does not generalize to arbitrary S. For instance, with $s = 3$, and the integer values 1,2,2,4 whose sum equal 3^2, one cannot find $[a, b, c]$ within these values such that $a + b + c = 0[3]$.

5 Linear Mappings on Suitable Ring Powers

This last section takes advantage of the structure provided by *linear* mappings $S^n \to S^n$. The results of Section 3 show that $O(n)$ assignments are sufficient to compute such a mapping. Here, we achieve a stronger result: the number of required mappings is bounded by $2n - 1$, *and* all intermediary assignments are linear. In [6], a similar result is obtained in the particular case of linear boolean mappings.

Here, S only needs to be a (non-necessarily finite) quotient of an Euclidean domain R by an ideal I. Classical examples for S are: any field (the result of this paragraph for S being a field is easier, since most technicalities can be skipped), the rings $\mathbb{Z}/s\mathbb{Z}$, or $K[x]/(P)$ for some polynomial P with coefficients in a field K. In the sequel, S is assumed to satisfy this property; R and I are defined accordingly, and S^* denotes the invertible elements of S.

Lemma 3. *Let $x_1, \ldots, x_n$ be coprime elements of R. Let $i_0 \in [1 \ldots n]$. There exists multipliers $\lambda_1, \ldots, \lambda_n$ such that $\lambda_{i_0} = 1$, and $\sum_i \lambda_i x_i \in S^*$.*

Proof. This a consequence of the Chinese Remainder Theorem. We treat the case where the ideal I is generated by a prime power p^v. If x_{i_0} is itself coprime to p, the result holds with $\lambda_i = \delta_i^{i_0}$. Otherwise, the integers x_i being coprime, there exists an integer $i_1(p)$ such that $x_{i_1(p)}$ is coprime to p. Therefore $x_{i_0} + x_{i_1(p)}$ is coprime to p, hence we may set $\lambda_i = \delta_i^{i_0} + \delta_i^{i_1(p)}$. □

Theorem 4. *Every linear mapping E on S^n is computed by an in situ program of length $2n-1$ and signature $1, 2, ..., n, n-1, ..., 1$ made of linear assignments. Furthermore, if E is bijective, then the inverse mapping E^{-1} is computed by the in situ program defined by the sequence of assignments in reversed order together with the following transformation: $[x_i := A(x_i) + F] \mapsto [x_i := A^{-1}(x_i - F)]$.*

Proof. The proof proceeds by induction. Let M be a matrix representing a mapping which leaves the first $k-1$ variables unchanged (we initially have $k = 1$ for the starting matrix). Hence the first $k-1$ rows of M equal those of the identity matrix. We explore the possibility of rewriting M as a product $L_k M' R_k$, where the first k rows of M' match those of the identity matrix.

Let g be the greatest common divisor of (arbitrary representatives in R of) the coefficients of column k in M. A favorable situation is when $m_{k,k}$ is in gS^*. Should this not be the case, let us see how we can transform the matrix to reach this situation unconditionally. Assume then for a moment that $m_{k,k} \notin gS^*$. Lemma 3 gives multipliers $\lambda_1, \ldots, \lambda_n$ such that $\sum_\ell \lambda_\ell m_{\ell,k} \in gS^*$, with the additional constraint that $\lambda_k = 1$. Let us now denote by T the $n \times n$ matrix defined by $t_{i,j} = \delta_i^j$ for $i \neq k$, and $t_{k,j} = \lambda_j$. Clearly T is an invertible assignment matrix, and the product $T * M$ has a coefficient at position (k, k) which is in gS^*.

Now assume $m_{k,k} \in gS^*$. Let G be the diagonal matrix having $G_{k,k} = g$ as the only diagonal entry not equal to 1. Let $M'' = MG^{-1}$ (M'' has coefficients in R because g is the g.c.d. of column k). We have $m''_{k,k} \in S^*$. We form an assignment matrix U defined by $u_{i,j} = \delta_i^j$ for $i \neq k$, and $u_{k,j} = m''_{k,j}$. The matrix U is an invertible assignment matrix (its determinant is $m''_{k,k}$). The k first rows of the matrix $M' = M'' * U^{-1}$ match the k first rows of I_n, and we have $M = T^{-1} \times M' \times (UG)$. Our goal is therefore reached with $L_k = T^{-1}$ and $R_k = UG$.

Repeating the procedure, our input matrix is rewritten as a product $L_1 L_2 \ldots L_{n-1} R_n \ldots R_1$, where all matrices are assignment matrices. No left multiplier L_n is needed for the last step. Finally, the determinant of M is invertible if and only if all the matrices R_k are invertible, hence the reversibility for bijective mappings. □

We digress briefly on the computational complexity of *building* the in situ programs for the linear mappings considered here. The matrix operations performed here all have complexity $O(n^2)$ because of the special shape of the assignment matrices. Therefore, the overall computational complexity of the decomposition is $O(n^3)$.

The procedure above can be illustrated by a small example. Assume we want to decompose the mapping in $\mathbb{Z}/12\mathbb{Z}$ given by the matrix

$$M = \begin{pmatrix} 4 & 5 \\ 6 & 4 \end{pmatrix}.$$

Our first step is the left multiplication, which gives

$$\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} M = \begin{pmatrix} -2 & 1 \\ 6 & 4 \end{pmatrix}.$$

Then, the common divisor 2 can be set aside, and the matrix U appears:

$$\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \mathrm{M} = \begin{pmatrix} -1 & 1 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -3 & 7 \end{pmatrix} \begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\mathrm{M} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -3 & 7 \end{pmatrix} \begin{pmatrix} -2 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 9 & 7 \end{pmatrix} \begin{pmatrix} 10 & 1 \\ 0 & 1 \end{pmatrix}$$

This corresponds to the following sequence of assignments:

$$x_1 := 10x_1 + x_2; \qquad x_2 := 9x_1 + 7x_2; \qquad x_1 := x_1 + x_2.$$

References

1. Agrawal, D.P.: Graph theoretical analysis and design of multistage interconnection networks. IEEE Trans. Computers C32, 637–648 (1983)
2. Beneš, V.E.: Optimal Rearrangeable Multistage Connecting Networks. Bell System Technical J. 43, 1641–1656 (1964)
3. Bermond, J.C., Fourneau, J.M., Jean-Marie, A.: A graph theoretical approach to equivalence of multi-stage interconnection networks. Discrete Applied Maths. 22, 201–214 (1988)
4. Burckel, S., Gioan, E.: In situ design of register operations. In: IEEE Proceedings of ISVLSI 2008 (2008)
5. Burckel, S., Morillon, M.: Quadratic Sequential Computations. Theory of Computing Systems 37(4), 519–525 (2004)
6. Burckel, S., Morillon, M.: Sequential Computation of Linear Boolean Mappings. Theoretical Computer Science serie A 314, 287–292 (2004)
7. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Math. Comput. 19, 297–301 (1965)
8. König, D.: Uber Graphen und ihre Anwendung auf Determinantentheorie und Mengenlhere. Math. Ann. 77, 453–465 (1916)
9. Nussbaumer, H.J.: Fast Fourier transform and convolution algorithms, 2nd edn. Springer Series in Information Sciences, vol. 2. Springer, Heidelberg (1982)
10. Ravikumar, C.P., Aggarwal, R., Sharma, C.: A graph theoretic approach for register file based synthesis. In: Proceedings of VLSID 1997, Tenth International Conference on VLSI Design (1997)