Correspondence

A Systematic Design of Cellular Permutation Arrays

A. YAVUZ ORUÇ AND AJAI THIRUMALAI

Abstract—**This paper presents a parametrized design technique for cellular permutation arrays based on coset decompositions of symmetric groups. A new type of permutation cell, referred to as a *coset generator*, is introduced to customize the propagation delay, fan-in, fan-out, and number of edges in the target network. To aid in the design process, a cost function is derived expressing the number of edges in terms of the number of inputs and the sizes of cells. The results provide a spectrum of networks which vary with the size of the coset generator used, and range from a simple bipartite graph to several cellular permutation arrays reported in the literature.**

Index Terms—**Cellular permutation array, coset generator, hexagonal cells, rearrangeable network, symmetric group.**

I. Introduction

This paper extends upon the cellular permutation arrays which were described in [5] and [6]. Group theory was used in these efforts to generalize the cellular permutation arrays of Kautz *et al.* [4]. It was also shown in [7] and [8] that these networks can be programmed in $O(n)$ steps. A further generalization of these networks will be presented here by proving stronger results on coset decompositions of symmetric groups.

In geometrical terms, designing a cellular permutation array amounts to forming one- or two-dimensional arrays of polygonal permutation cells each of which has some or all of its sides abutted against another polygonal cell so that the combined structure can realize the set of all permutations from a set of sides on its boundary identified as *inputs* to another set of sides of equal cardinality also on its boundary and identified as *outputs*. Whether it is designed as a cellular array or not, a network which can realize the set of all permutations of its inputs onto its outputs is called *rearrangeable* and finds applications in circuit switching [10] and parallel processing [3], [9]. In addition to being rearrangeable, the networks considered in this paper will also be cellular.

In general, one- or two-dimensional cellular permutation arrays can be constructed by using any number and type of polygonal cells if not all the sides of polygons are required to be abutted against the sides of other *identical* cells. On the other hand, it is easily shown that two-dimensional cellular permutation arrays using polygonal cells any of which has each of its sides abutted against another identical cell can be constructed by using only three types of polygons, namely those with three, four, or five sides. Cellular permutation arrays with four-sided polygons, or square cells, have been extensively studied in the literature [4]–[6]. It should be easy to see that to each cellular permutation array with square cells there corresponds a cellular permutation array with three-sided polygons, or triangular cells. It can also be shown that a cellular permutation array which uses triangular cells can be converted to a cellular permutation array with square cells by a simple augmentation of its cells on its boundaries. Fig. 1 depicts two four-input triangular-shaped cellular permutation arrays, one constructed with triangular cells and the other with square cells. In general, we need n^2 triangular cells and $n(n-1)/2$ square cells to construct an n-input triangular permutation array which is easily shown to be rearrangeable.

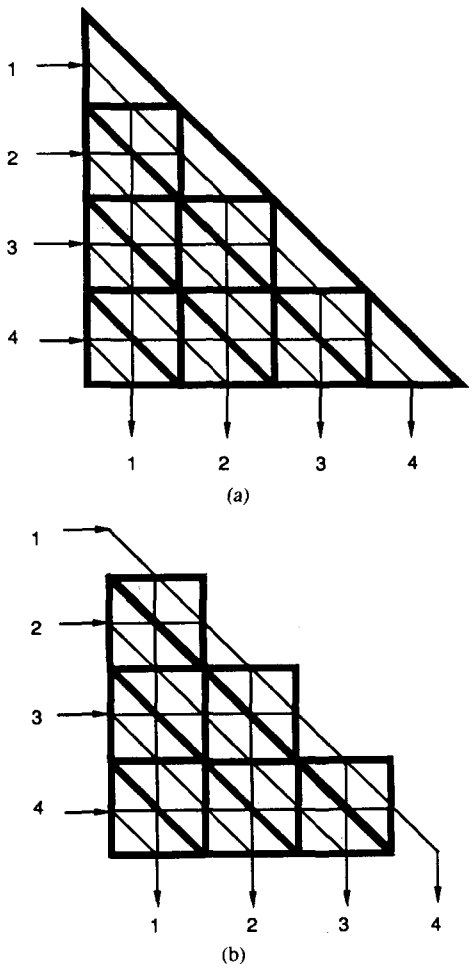


Fig. 1. 4-input triangular cellular arrays with triangular and square cells. (a) A 4-input triangular permutation array constructed with triangular cells. (b) A 4-input triangular permutation array constructed with square cells.

One way to construct a cellular permutation array with hexagonal cells is to use a triangular geometry as shown in Fig. 2(a). The first two inputs are tied to the network through the hexagonal cell on the left, and afterwards a new input enters the network from left to right per each column of hexagonal cells. In general, an n-input triangular permutation array can be constructed by using $(n^2-4)/4$ cells if n is even and $n \geq 4$, and $(n^2-5)/4$ if n is odd and $n \geq 3$. If we assume that each cell has full connectivity, that is, each of its inputs is connected to each of its outputs as shown in Fig. 2(b), then it is obvious that the first cell can permute its inputs to its outputs in any one of $3! = 6$ ways. Furthermore, if we suppose that the first n columns of cells can permute the first $n+2$ inputs to the $n+2$ outputs in any one of $(n+2)!$ ways, it immediately follows from the

Manuscript received February 27, 1987; revised October 20, 1987.

A. Y. Oruç is with the Department of Electrical Engineering, University of Maryland, College Park, MD 20742.

A. Thirumalai is with Digital Equipment Corporation, Marlborough, MA 01752.

IEEE Log Number 8927544.

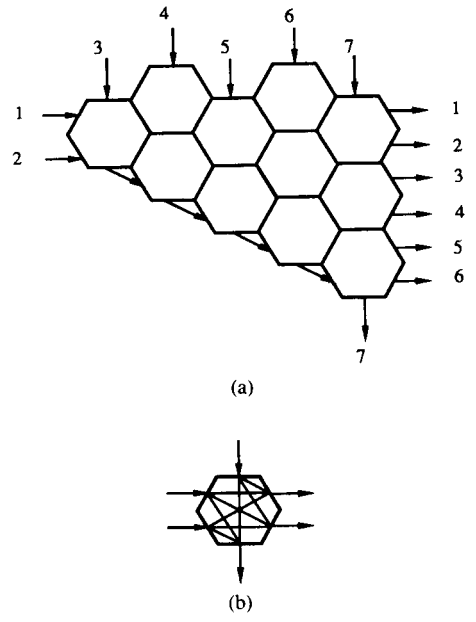


Fig. 2. A 7-input triangular array with hexagonal cells. (a) A 7-input hexagonal cellular array. (b) The basic cell structure.

construction that the first $n + 1$ columns can permute the first $n + 3$ inputs onto the $n + 3$ outputs in any one of $(n + 3)!$ ways.

Summarizing the preceding discussion we have the following.

Theorem 1: Every triangular cellular array constructed using triangular, square, or hexagonal cells as in Figs. 1 and 2 is rearrangeable. ||

The question that remains is to determine what other cellular geometries of triangular, square, and hexagonal cells also lead to rearrangeable networks. More generally, we need to resolve if cellular permutation arrays can be designed more systematically. These two questions are handled in the following sections.

II. Approach

In general, permutation networks are built from smaller permutation networks which are built from still smaller permutation networks, etc. In order to mechanize this recursive process in the case of cellular permutation arrays, we will use the coset decomposition technique introduced in [6]. Let N be a finite set of elements which designate the inputs and outputs of a permutation network. The set of all permutations on N forms the *symmetric group* of degree $|N|$ which will be denoted by Σ_n where $n = |N|$, and the network will be said to be *rearrangeable* if it realizes all of Σ_n. Without loss of generality we shall let $N = \{1, 2, \cdots, n\}$.

The design technique is based on decomposing Σ_n into right (or left) cosets of another symmetric group Σ_m in Σ_n where Σ_m is defined over $\{1, 2, \cdots, m\}$; $m \leq n$. A *right coset* of Σ_m in Σ_n is a set of permutations $\Sigma_m \cdot p = \{\sigma \cdot p : \sigma \in \Sigma_m\}$ where p is an arbitrary but fixed element in Σ_n, and called the *leader* of the coset. A *left coset* is defined by interchanging the order of Σ_m and p in the product $\Sigma_m \cdot p$. Since the statements about right cosets directly extend to those about left cosets, we shall limit our discussion to right cosets.

Our first remark is that two right cosets are either identical or disjoint. This fact provides the basis for decomposing Σ_n into the cosets of Σ_m. More specifically, we have the following fact [6].

Theorem 2: Any two permutations in Σ_n lie in different right cosets of Σ_m *in* Σ_n if and only if they map at least one symbol in $\{m + 1, m + 2, \cdots, n\}$ onto different symbols in N. ||

Example: Let $n = 6$ and $m = 5$. The symmetric groups Σ_6 and Σ_5 are then defined, respectively, over $\{1, 2, 3, 4, 5, 6\}$ and $\{1, 2, 3, 4, 5\}$. The permutation maps

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 2 & 1 \end{pmatrix}$$

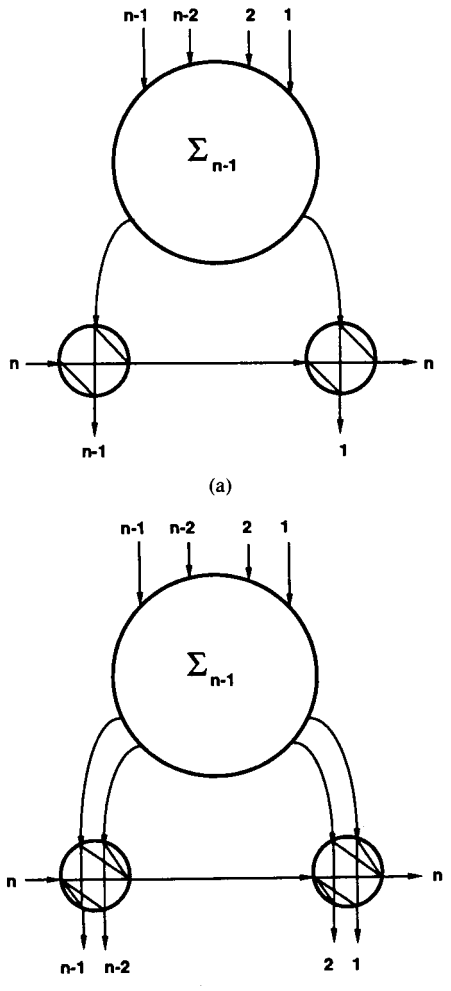


Fig. 3. Generating Σ_n from Σ_{n-1}. (a) By using 2-input cells. (b) By using 3-input cells.

and

$$q = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 1 & 3 & 4 & 5 & 2 \end{pmatrix}$$

belong to different cosets of Σ_5 since 6 is mapped to 1 by p and to 2 by q, i.e., to different outputs. Note that there are exactly six distinct right cosets of Σ_5 in Σ_6 as 6 can be mapped at most to six different places in $\{1, 2, \cdots, 6\}$. Also note that the cosets are distinguished only by where input 6 is mapped to, and they do not depend on how the images of 1, 2, 3, 4, 5 are fixed by the permutations which belong to a given coset. These six cosets of Σ_5 collectively sum to Σ_6. ||

The following theorem is easily proved by induction.

Theorem 3: The total number of distinct right cosets of Σ_{n-1} in Σ_n is n, and that for Σ_m in Σ_n is $n(n - 1) \cdots (m + 1)$.

Fig. 3 depicts the realization of Σ_n by the right cosets of Σ_{n-1} with 2-input and 3-input switching cells. Since input n can be mapped onto any one of the n outputs through the small circular cells, all the right cosets of Σ_{n-1} in Σ_n can be realized by each of the networks. By applying this decomposition recursively upon Σ_{n-1}, Σ_{n-2}, etc., we can realize Σ_n as a cellular permutation array consisting of 2×2 or 3×3 cells [5], [6]. For example, the networks shown in Fig. 1 can be obtained by recursively decomposing the network in Fig. 3(a) and that in Fig. 2 can be obtained by recursively decomposing the network in Fig. 3(b). The only remark that needs to be made is that the hexagonal cells, except the one to which the first two inputs are tied in Fig. 2, need not be fully connected as stated in Section I. Following the edge specification in Fig. 3(b), Fig. 4 shows the edges needed. The leftmost cell requires full connectivity and hence needs

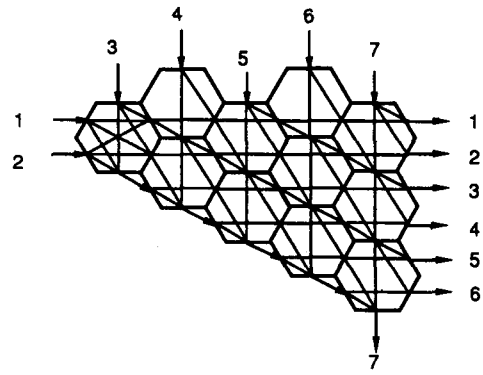


Fig. 4. A 7-input cellular array with specified cell connections.

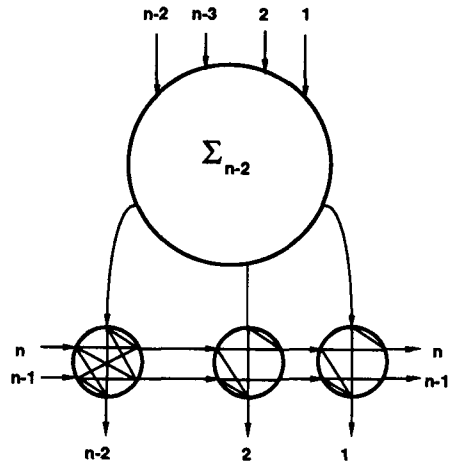


Fig. 5. Generating Σ_n from Σ_{n-2}.

all nine edges. The cells on the top row have only two inputs and hence require four edges each. All other cells have the functionality of the small circular cells in Fig. 2(b) and each needs seven edges.

Using these figures, the total number of edges in an n-input array can be shown to be

$$7\left(\frac{n^2-4}{4}\right)-3\left(\frac{n-2}{2}\right)+2 \text{ when } n \text{ is even}$$

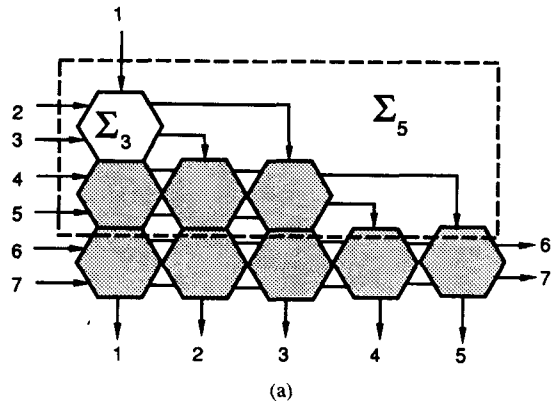
and

$$7\left(\frac{n^2-5}{4}\right)-3\left(\frac{n-3}{2}\right)+2 \text{ when } n \text{ is odd.} \quad (1)$$

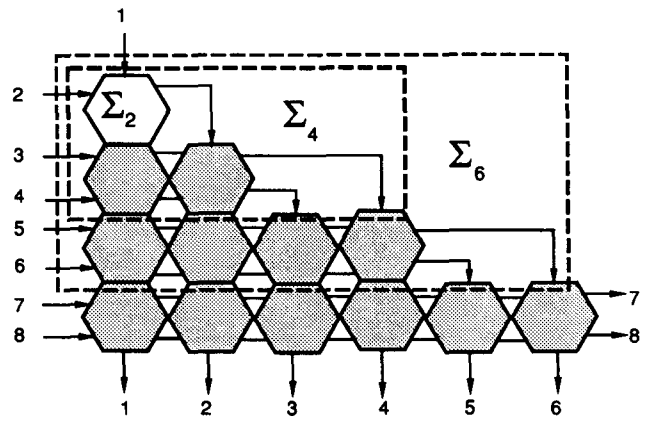
We can use the same technique to design other cellular permutation arrays with hexagonal cells. As another example, suppose we use the right cosets of Σ_{n-2} to decompose Σ_n. The resultant network then consists of two stages where the first stage is assumed to realize all of Σ_{n-2} and the second stage generates the $n(n-1)$ right cosets of Σ_{n-2} in Σ_n. The second stage can be realized by a set of 3-input cells as shown in Fig. 5. The connections for the small circular cells are designed to provide inputs n and $n-1$ with paths to any two of the outputs with the desired order. At the same time, all the remaining inputs can be permuted among themselves by the Σ_{n-2} network in the first stage, and any of them can be mapped to either of outputs n and $n-1$ via the small circular cells. It follows that the entire structure realizes Σ_n.

Cellular arrays which result from this kind of decomposition are shown in Fig. 6. The number of edges in an n-input network constructed this way can be shown to be

$$6\frac{n(n-2)}{4}+3\frac{n}{2}+1 \text{ when } n \text{ is even} \quad (2)$$



(a)



(b)

Fig. 6. Recursive realization of Σ_n using hexagonal cells. (a) n is odd. (b) n is even.

and

$$6\frac{(n^2-2n+2)}{4}+3\left(\frac{n-1}{2}\right) \text{ when } n \text{ is odd.} \quad (3)$$

III. Permutation Networks Based Coset Generators

In the preceding section, we formed two cellular permutation arrays of hexagonal cells using the decompositions of Σ_n into the right cosets of Σ_{n-1} and Σ_{n-2}. By direct comparison of the numbers of edges in the two networks, it is clear that the decomposition with Σ_{n-2} has led to fewer directed edges. This fact raises the question whether we can reduce the number of edges even further by working with symmetric groups such as Σ_{n-3}, Σ_{n-4}, $\cdots$, etc. Another question that should also be raised is about the geometry of the cells used in the construction of cellular permutation arrays. There is no reason why one should limit the geometry to hexagonal cells, and other forms of cells should also be investigated in connection with reducing the total number of edges in the network.

These two degrees of freedom in decomposing Σ_n bring us to the concept of what we shall call a *generalized mixer* or *coset generator*. Consider the right cosets of Σ_m in Σ_n, and let $k = n - m$. We can view the generation of these cosets as mixing some k *new* inputs with some m *old* inputs in a way to realize Σ_{m+k} if a network is given to permute the m inputs in any one of $m!$ ways. The schematic in Fig. 7 illustrates this concept where the top network realizes Σ_m and the bottom one generates all of its right cosets.

In order to describe the realization of Σ_n by a two-stage network, we must specify the edges inside the coset generator. First note that an edge is needed between every horizontal input and every output of the coset generator, since lacking any such edge will prevent the realization of some $(n-1)!$ permutations. Since there are k horizontal inputs and $m+k$ outputs, this requires a total of $(m+k)k$ edges as depicted in Fig. 8.

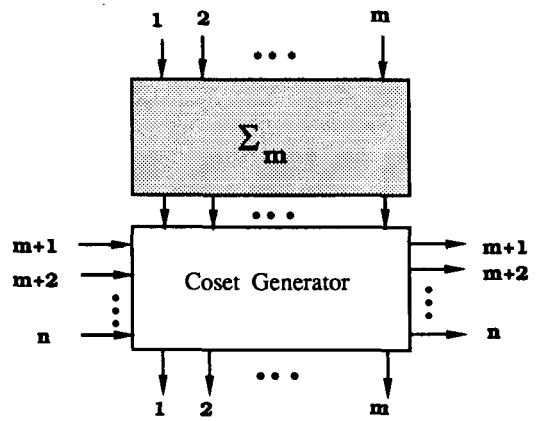


Fig. 7. Realization of Σ_n by a two-stage permutation network.

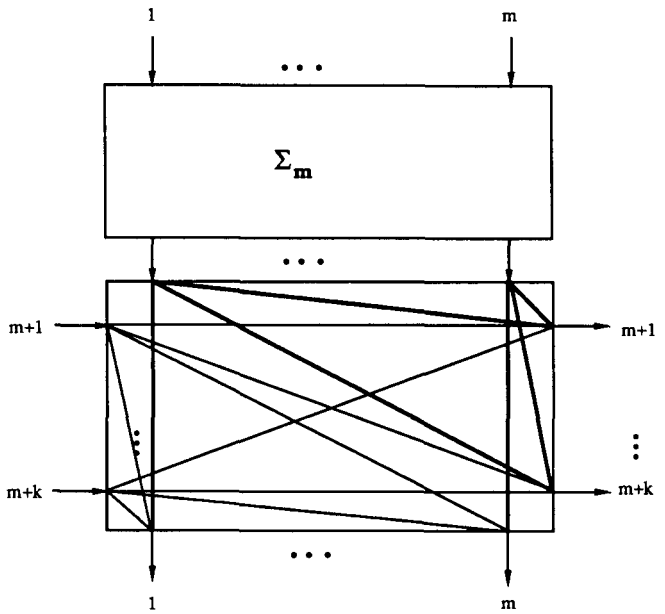


Fig. 8. The edges in the coset generator of a two-stage permutation network.

As for the vertical inputs it is easy to see that each vertical input of the coset generator must be connected to at least $k + 1$ outputs since otherwise we can define a permutation that can block that input from being connected to any output by mapping a subset of k horizontal inputs to wherever that input is connected. This then implies that the combined structure is not rearrangeable. Furthermore, it can also be shown that $k + 1$ edges per each vertical input suffice to make the overall network rearrangeable [8]. These $k + 1$ edges can be connected as shown in Fig. 8 among other ways.

It follows that we need to have $(m + k)k + (k + 1)m$ edges in all for the coset generator. Substituting $n - k$ for m, the total number of edges can be expressed as

$$2kn + n - k - k^2. \tag{4}$$

We can use this expression to minimize the total number of edges in a cellular permutation array obtained by recursively decomposing a two-stage permutation network. Let $C_{n,k}$ denote the number of edges in such a cellular permutation array. Using (4), we can write the recurrent equation

$$C_{n,k} = n(2k+1) - k^2 - k + C_{n-k,k} \tag{5}$$

where $C_{n-k,k}$ is the number of edges in the Σ_m or Σ_{n-k} network. Repeating the same decomposition recursively, i.e., decomposing Σ_{n-k} into Σ_{n-2k} and that into Σ_{n-3k}, etc., it can be shown that

$$C_n = (2k+1)\left\{n\alpha - \frac{k\alpha(\alpha-1)}{2}\right\} - \alpha(k^2+k) + C_{n-\alpha k,k} \tag{6}$$

where α is the depth of recursion, and $C_{n-\alpha k,k}$ is the number of edges in the $\Sigma_{n-\alpha k}$ subnetwork which is left undecomposed. It can further be shown that this recursion has the following closed form solution for the boundary condition $C_{k,k} = k^2$, i.e., $\alpha = n/k - 1$:

$$C_{n,k} = n^2 - \frac{n}{2} + \frac{n^2}{2k}. \tag{7}$$

If we define the propagation delay $P_{n,k}$ of a network as the length of the longest path between its inputs and outputs then

$$P_{n,k} = \alpha + 1 = \frac{n}{k}. \tag{8}$$

It also follows from the construction of the coset generator that the fan-out $F_{n,k}$ is given by

$$F_{n,k} = k + 1 \tag{9}$$

for all vertical inputs and

$$F_{n,k} = n - (i-1)k \tag{10}$$

for all horizontal inputs of the coset generator in decomposition level i; $1 \le i \le n/k$.

These results lead to the following statement.

Theorem 4: In a cellular permutation array which is recursively obtained from a two-stage network with k horizontal inputs, $C_{n,k}$, $P_{n,k}$, and fan-out for horizontal inputs decrease with increasing values of k while the fan-out for vertical inputs increases with the increasing values of k. ‖

IV. Analysis of Results

To provide a concrete comparison of cellular arrays, we list in Table I the total number of edges, propagation delay, and fan-out for various cellular permutation arrays described in the paper and the networks of Kautz *et al.* [4] and Bandyopadhyay *et al.* [1], abbreviated, respectively, as KLW and BBC networks. While the first six rows refer to specific networks, the last three rows depict only a few of many cellular arrays which can be formed by using coset generators. In particular, the entries in the last row correspond to a complete bipartite graph realization of Σ_n.

The table reveals that all of the listed networks have $O(n^2)$ edges although the number of edges decreases as k increases as stated in Theorem 4 and reaches its minimum value at $k = n$. This may appear to be counterintuitive at first since the decomposition in many other

TABLE I
NUMBER OF EDGES FOR VARIOUS CELLULAR NETWORKS

Network	Edge-Count	Delay	Fan-out
BBC	$3\frac{n^2}{2} - \frac{n}{2} + 1$	$n-1$	2, but n for one
KLW	$2n^2 - 2n$	$n-1$	2
Hexagonal ($k=1$), n is even	$7(\frac{n^2-4}{4}) - 3(\frac{n-2}{2}) + 2$	$n-2$	2 or 3
Hexagonal ($k=1$), n is odd	$7(\frac{n^2-5}{4}) - 3(\frac{n-3}{2}) + 2$	$n-2$	2 or 3
Hexagonal ($k=2$), n is even	$6(\frac{n(n-2)}{4}) + 3(\frac{n}{2}) + 1$	$n-2$	2 or 3
Hexagonal ($k=2$), n is odd	$6(\frac{n^2-2n+2}{4}) + 3(\frac{n-1}{2})$	$n-1$	2 or 3
Coset generator ($k=2$)	$5\frac{n^2}{4} - \frac{n}{2}$	$\frac{n}{2}$	varies between 3 and n
Coset generator ($k=\frac{n}{2}$)	$n^2 + \frac{n}{2}$	2	varies between $\frac{n}{2}$ and n
Coset generator ($k=n$)	n^2	1	n

network designs including the Benes network [2] is a common method of reducing the network's cost. The reason for this anomally is that the cosets are not optimally coded into the coset generators in that all the required connections between the inputs and outputs are specified in terms of direct edges. In order to reduce the edge count, one needs to code coset leaders so as to maximize the number of edges shared between different cosets. The treatment of this coding problem will be deferred to another place.

In contrast, the tradeoff between $P_{n,k}$ and $F_{n,k}$ is more noticeable: cellular permutation arrays constructed with small cells have $O(1)$ fan-out and $O(n)$ propagation delay while those constructed with large coset generators have $O(n)$ fan-out but $O(1)$ propagation delay. This tradeoff is also more intuitive even though the $O(n)$ propagation delay for constant fan-out is not necessarily the best possible. In principle, one can construct networks with $O(\log_2 n)$ propagation delay by using switching cells with $O(1)$ fan-out. Nonetheless, this construction shares the same goal with our approach in that one uses decomposition which causes the propagation delay to increase from $O(1)$ to $O(\log_2 n)$ (rather than $O(n)$ in our case) to reduce the fan-out from $O(n)$ to $O(1)$. It seems that one can also achieve $O(\log_2 n)$ propagation delay with our coset decomposition technique by using a more compact coding for the coset leaders. As stated earlier, the possibility of such a coding will be explored elsewhere.

V. Concluding Remarks

This paper presented a technique for customizing the design of cellular permutation networks. It has been shown that there exists a rich spectrum of cellular permutation arrays which results from recursive coset decompositions of symmetric groups. Cellular permutation arrays constructed from 2×2 cells and the bipartite graphs have been shown to populate the two extreme ends of this spectrum. In particular, the bipartite graph realization of the set of all permutations is shown to outperform any cellular permutation array in number of edges, as well as in propagation delay. Nonetheless its $O(n)$ fan-out may be intolerable for large n. In this case, the coset decomposition technique can be used to systematically trade the fan-out with the propagation delay.

References

[1] S. Bandyopadhyay, S. Basu, and K. Choudhury, "A cellular permuter array," *IEEE Trans. Comput.*, vol. C-21, pp. 1116-1119, Oct. 1972.

[2] V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic.* New York: Academic, 1965.

[3] T. Feng, "A survey of interconnection networks," *IEEE Computer Mag.*, vol. 14, pp. 12-27, Dec. 1981.

[4] W. H. Kautz *et al.*, "Cellular interconnection arrays," *IEEE Trans. Comput.*, vol. C-17, pp. 443-451, May 1968.

[5] S. A. Nadkarni, "The design and performance evaluation of hybrid cellular interconnection arrays," M.Sc. thesis, Rensselaer Polytechnic Institute, Troy, NY, Aug. 1985.

[6] A. Y. Oruç, "Designing cellular permutation networks through coset decompositions of symmetric groups," *J. Parallel Distrib. Comput.*, pp. 30-45, Aug. 1987.

[7] A. Y. Oruç and M. Y. Oruç, "Linear-time algorithms for programming cellular permutation arrays," in *Proc. ACM Nat. Comput. Sci. Conf.*, Cincinnati, OH, 1986, pp. 129-136.

[8] A. Y. Oruç and S. Schneider, "Coset networks for parallel processors," *J. Supercomput.*, pp. 23-39, 1989.

[9] H. J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies.* Lexington, MA: Lexington Books.

[10] K. J. Thurber, "Circuit switching technology: A state of the art survey," in *Conf. Proc.—Compcon 1978 Fall Conf.*, Sept. 1978, pp. 116-124.

On Systolic Contractions of Program Graphs

WEICHENG SHEN AND A. YAVUZ ORUÇ

***Abstract*—One of the active areas in supercomputer research is concerned with mapping programs onto networks of processors. In this paper, a variant of the mapping problem, namely, systolic contractions of program graphs are considered. The notion of time links is introduced to mechanize the contraction process; the timing of information flow between processors is modeled in terms of fundamental loop and path equations of delays, and optimized using linear programming.**

***Index Terms*—Fundamental loops of delays, graph contraction, processor graph, program graph, systolic array, time links.**

I. Introduction

An active area in supercomputing research is concerned with designing systematic procedures for mapping algebraic computations onto networks of processors. Significant results have been reported in the literature, especially about mapping matrix computations onto a family of processor networks collectively referred to as *systolic arrays* [1]-[4]. These arrays prompted a considerable interest due to their regular structures, adjacency of interconnections between their processors, and simplicity of their control.

Previous efforts on mapping algorithms onto systolic arrays deal

Manuscript received February 18, 1987; revised December 23, 1987. This work was supported in part by the Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute.

W. Shen is with the Department of Electrical Engineering, University of New Hampshire, Durham, NH 03824.

A. Y. Oruç, is with the Department of Electrical Engineering, University of Maryland, College Park, MD 20742.

IEEE Log Number 8927545.