ELSEVIER

INTEGRATION, the VLSI journal 28 (1999) 141–156

INTEGRATION
theVLSI journal

www.elsevier.com/locate/vlsi

Bounds, designs and layouts for multi-terminal FPIC architectures

Dinesh Bhatia[a,*], James Haralambides[b]

[a]*Design Automation Laboratory, ECE&CS Department, P.O. Box 210030, University of Cincinnati, Cincinnati, OH 45221-0030, USA*

[b]*Department of Mathematics and Computer Science, Barry University, 11300 NE 2nd Avenue, Miami Shores, FL 33161-6695, USA*

Received 24 April 1998

Abstract

Field-programmable interconnection chips provide the capability of realizing user programmable interconnections for any desired permutation. Such interconnections are very much desired for supporting rapid prototyping of hardware systems and for providing programmable communication networks for parallel and distributed computing. In this paper we address the problem of design of permutation network for routing any m to n, $m \leqslant n$ permutation. Specifically, we obtain lower bounds for the number of switches required to realize any one to many permutation of connections. In addition, we also show that m to n permutations whose underlying graph is of degree d has a routing delay of at least $\Omega(\log_d(m+n))$. Also, we propose efficient designs for implementing networks for any m to n permutations with asymptotically optimal utilization. Efficient two-layer layouts for such networks are also proposed. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Field programmable interconnection chips; Programmable devices; Permutation networks; Routing delay; VLSI layout; Area-performance trade-off

1. Introduction

Programmable devices are playing an important role in various areas of VLSI systems. The tremendous growth in density and performance of field programmable gate arrays (FPGAs) has opened new avenues for computing. In the mid-80s, the FPGAs were introduced with rapid

* Corresponding author. Fax: + 1-513-556-7326.

E-mail addresses: dinesh@ececs.uc.edu (D. Bhatia), haralam@euclid.barry.edu (J. Haralambides)

0167-9260/99/$ - see front matter © 1999 Elsevier Science B.V. All rights reserved.
PII: S0167-9260(99)00011-5

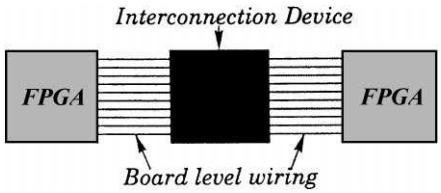

Fig. 1. A programmable board with the FPGAs and programmable interconnection device.

prototyping of glue logic as the main application in mind. Growing densities and improved CAD tools have made FPGAs very attractive for applications like hardware acceleration, logic emulation, custom (reconfigurable) computing, and more. Today's FPGAs can easily implement designs that are about 80,000 gates large. This density is soon expected to grow and one million gates will become very common in the next few years. While this is a favorable trend, applications that are being targeted towards FPGAs are much larger and thus designs have to be partitioned for a multi-device implementation. Partitioned and mapped segments of design have to be wired together to make the entire application executing on a programmable platform. Hardware systems used for logic emulation and reconfigurable computing should have the capability of mapping and wiring various types of applications. Thus, to provide a programmable capability for wiring the FPGAs together, programmable interconnections are introduced between FPGAs. Fig. 1 illustrates a simple two FPGA-based system where a dedicated device provides programmable interconnection between the two FPGAs. For commercial logic emulation systems like the *Realizer* system [1] and the *Enterprise Emulation* system from Quickturn Systems [2], the programmable interconnection between FPGAs is realized using multiple small crossbar interconnection chips. These chips called field programmable interconnection chips (FPICs) provide a programmable point-to-point routing capability. Commercially, such devices are manufactured by vendors like *Icube* [3] and *Aptix* [4]. There are some realizations of programmable boards or systems where dedicated FPGAs have been used as programmable interconnection chips. In such realizations, only the routing architecture of the FPGA is used and the logic block architecture is left untouched. The complexity of board level routing for routing two-terminal nets has been addressed in [5]. Mak and Wong also describe simple polynomial time algorithms for board level routing [5]. Mak and Wong also address the problem of routing multi-terminal nets in [6]. Khalid and Rose provide a framework for the evaluation of effectiveness of interconnection networks in multi-FPGA systems [7]. In addition, they also describe an experimental hybrid network for multi-FPGA systems [8].

Field-programmable interconnection chips (FPICs) are fully configurable devices used for customizing interconnections. A high-density FPIC can eliminate the need for designing PCBs for board level designs. Components on a board can be interconnected by configuring an FPIC. The FPIC can implement any interconnection permutation. Ideally, a hardware board made of FPGAs and FPICs should be able to support a variety of applications. In a majority of programmable hardware implementations. FPGAs are used to implement programmable interconnections. Also, interconnection networks that are either crossbar of Beneš are emulated on FPGAs. The obvious disadvantages of this approach are the complete waste of logic area that is available on FPGAs and the slowdown of the interconnection network due to the presence of (sometimes many) programmable switches.

An FPIC that allows programmable wiring for large systems like PCBs should allow any wiring permutation for a point to a point and from a point to many points (interconnections expand due to fanout and thus wiring should accommodate fanout). In our earlier work, we established lower bounds and presented layouts for networks that allow point-to-point interconnections. In general, the board routings are multi-terminal nets. In this paper we visit the problem of establishing bounds, resource requirements for programmable interconnection chips that allow multi-terminal wirings. We also present their fabricatable layouts.

2. Lower bounds for FPICs that allow one-to-many I/O connections

The FPIC model we use throughout this paper is characterized by input-to-output connections with fanouts higher than one. We are going to examine FPICs that implement networks of a total of $m + n$ terminal nodes (inputs and outputs), where any m nodes, designated as inputs can be connected to any n nodes designated as outputs. The set of inputs (and outputs) can be either predetermined or selected during the implementation of the interconnections.

In this section we obtain lower bounds for permutation networks that realize m-to-n interconnections. We examine variations of models with uniform or non-uniform fanouts and obtain results that are related to the performance of the architecture.

In what follows, we give useful definitions of basic terminology and of properties of the underlying networks for FPIC architectures. Assuming that we are using programmable switches and routing elements of uniform characteristics, we consider the underlying network of an FPIC to be an unweighted undirected graph $G = (V, E)$. Furthermore, an FPIC implements an m-to-n permutation if there exists an $(m + n)$-node dedicated subset $V_{m,n} \subseteq V$ such that any mapping of m nodes to n nodes in $V_{m,n}$ is possible. Specifically, a set of m node-disjoint connections of m nodes to n nodes in $V_{m,n}$ is possible simultaneously. Below, we give a more formal definition of an m-to-n permutation.

Definition 1. We call a path $P_{u,v}$ a permutation path, if it provides a connection between two nodes u, v in $V_{m,n}$. We call nodes in $V_{m,n}$ i-nodes. All other nodes are called *auxiliary* or *a-nodes*.

Definition 2. Given an i-node u and a set of i-nodes $U = \{u_1, \ldots, u_s\}$, $u \notin U$, we define a multipath P_U^u as a collection of s permutation paths P_{u,u_i} connecting node u to nodes u_i, $i \in \{1, \ldots, s\}$. Paths P_{u,u_i} need not be node-disjoint.

Definition 3. A graph $G = (V, E)$ is said to satisfy property $\delta(G, V_{m,n})$ if there exists an $(m + n)$-node subset of V, $V_{m,n} \subseteq V$, $m \leqslant n$, such that any m input nodes in $V_{m,n}$ can be connected to any n' output nodes in $V_{m,n}$, $n' \leqslant n$, via a set of m node-disjoint multipaths.

Definition 4. We call a graph $G = (V, E)$ an m-to-n permutation or (m, n)-permutation if it satisfies property $\delta(G, V_{m,n})$, $V_{m,n} \subseteq V$. The definition of property δ does not exclude any node in $V_{m,n}$ from becoming an input or an output node. We call graph G an (m, n)*-permutation if it satisfies property $\delta(G, V_{m,n})$ and $V_{m,n}$ is a set of m designated inputs and n designated outputs.

The definition of property δ gives rise to a variety of architectures implementing FPICs. Let m and n, $m \leqslant n$, denote the number of inputs and outputs of the architecture. We distinguish the following categories:

(a) 1-to-1 when $m = n$, where one input can be connected to one output at a time,
(b) 1-to-k with $m < n$, where each input is always connected to exactly k outputs, $k \leqslant n/m$,
(c) 1-to-k with $m < n$, where an input node u_i, $1 \leqslant i \leqslant n$, is connected to k_i outputs, $k_i \leqslant k$, and $\sum_{i=1}^{m} k_i \leqslant n$.
(d) 1-to-k_i with $m < n$, where an input node u_i, $1 \leqslant i \leqslant m$, can be connected to at most k_i outputs, $\sum_{i=1}^{m} k_i \leqslant n$.

Case (a) has been studied in [9,10]. Case (b) describes permutation architectures of uniform fan-out. Case (c) describes interconnection architectures of non-uniform, uniformly bounded fan-out. Architectures of this model can implement any m-to-n permutation. Finally, Case (d) deals with interconnection architectures of non-uniform, non-uniformly bounded fan-out. Architectures of this case are circuit-specific. For example, specific nodes may be dedicated to providing power and ground interconnections.

Definition 5. We use the notations (m, n)*-permutation and (m, n)-permutation to describe permutations with and without designated inputs/outputs, respectively, for Case (b). The corresponding notations for Case (c) are $(m, n)^*_{\leqslant k}$- and $(m, n)_{\leqslant k}$-permutation. For Case (d), they are $(m, n)^*_{k_i}$- and $(m, n)^*_{k_i}$-permutation.

We are going to present lower bounds on the number of programmable switches (edges) of permutations for Cases (c) and (d). Case (b) is a special case of Case (c) and all properties proven for Case (c) apply to it as well. To simplify the analysis that follows, we give some more definitions and notations.

Definition 6. We call two paths partially-node-disjoint or p-disjoint if they differ by at least one node, i.e., the intersection of their node sets is non-empty.

Definition 7. We call a multipath P^u_U a *permutation multipath*, if it is a collection of (not necessarily node-disjoint) permutation paths.

Theorem 1. *The number or edges of an* $(m, n)_{\leqslant k}$*-permutation is bounded below by* $\Omega([(m + n)/2] \log 2m)$.

Proof. By induction on m.

Basis. ($m = 1$) The smallest graph that implements all 1-to-n permutations is the star graph of $n + m + 1 = n + 2$ nodes (see Fig. 2). Clearly, a minimum of $n + 1 > [(m + n)/2] \log 2m$ edges are needed.

Inductive step. Assume that the theorem holds for all $(p, q)_{\leqslant k}$-permutations, $p < m$, $q < n$, we will show it holds for all $(2p, 2q)_{\leqslant k}$-permutations and thus, for all $(m, n)_{\leqslant k}$-permutations.

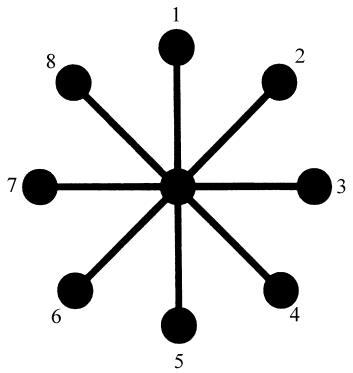

Fig. 2. A $n+2$ node star graph for $n=7$.

Let $G=(V, E)$ be a $(2p, 2q)_{\leqslant k}$-permutation satisfying property $\delta(G, V_{2p,2q})$. We call G an optimal permutation if it is a permutation with the minimum number of edges.

Let $G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$ be the smallest subgraphs of G that satisfy properties $\delta(G_1, V^1_{p,q})$ and $\delta(G_2, V^2_{p,q})$, respectively. Sets $V^1_{p,q}$ and $V^2_{p,q}$ are $(p+q)$-node subsets of $V_{2p,2q}$, where $V^1_{p,q} \cup V^2_{p,q} = V_{2p,2q}$ and $V^1_{p,q} \cap V^2_{p,q} = \emptyset$. Then, according to the inductive hypothesis, the number of edges $|E_1|$ ($|E_2|$) is bounded below by $[(p+q)/2] \log 2p$.

It is not difficult to see that, if G satisfies property $\delta(G, V_{a,b})$ there exists a subgraph G' of G that satisfies property $\delta(G', V'_{a',b'})$, where $V'_{a',b'} \subseteq V_{a,b}$ is a set of $a' \leqslant a$ inputs and $b' \leqslant b$ outputs. It is, therefore, always possible to find subgraphs G_1 and G_2.

We are going to show that for G to be a $(2p, 2q)_{\leqslant k}$-permutation, G_1 and G_2 must be node-disjoint (and thus edge-disjoint) graphs connected by a minimum of $2q$ edges.

We claim that all edges of G_1 and G_2 are used by at least one permutation, otherwise unused edges can be removed, thus resulting in smaller graphs G'_1 and/or G'_2 that are $(p, q)_{\leqslant k}$-permutations, a contradiction.

Let $P^{u_1}_{U_1}$ and $P^{u_2}_{U_2}$ be multipaths in G_1 and G_2, respectively, connecting i-node u_1 to all nodes in $U_1 \subset V^1_{p,q}$ and i-node u_2 to all nodes in $U_2 \subset V^2_{p,q}$. Paths $P^{u_1}_{U_1}$ and $P^{u_2}_{U_2}$ may share no nodes or edges, otherwise the simultaneous connection of node u_1 to all nodes in U_1 and of node u_2 to all nodes in U_2 is not feasible. We remove all nodes and incident edges of nodes in $\{u_1, u_2\} \cup U_1 \cup U_2$. The resulting subgraphs are $(p_1, q_1)_{\leqslant k}$- and $(p_2, q_2)_{\leqslant k}$-permutations, respectively, $p_1 = p-1$, $q_1 = q - |U_1|$, $p_2 = p-1$, $q_2 = |U_2|$, and, therefore, the same arguments hold. An iterative application of the arguments leads to the conclusion that G_1 and G_2 may share no nodes or edges.

For G to be a $(2p, 2q)_{\leqslant k}$-permutation multipaths in G_1 and G_2 must communicate via a collection of paths and/or multipaths. In what follows, we will show that a minimum of $p+2q$ p-disjoint paths are needed to provide all these connections.

Let u_1, u_2 be nodes in V_1 and v_1, v_2 be nodes in V_2. Let $P_{u_1 v_1}$ and $P_{u_2 v_1}$ be the permutation paths connecting node u_1 and node u_2 to node v_1, respectively. Paths $P_{u_1 v_1}$ and $P_{u_2 v_1}$ share no edge in E_1, otherwise a simultaneous connection of node u_1 and node v_1 and of node u_2 to all nodes in $V_2 - \{v_1\}$ is not possible. For the same reason, they share no edges in $E - E_1 - E_2$. Therefore, for nodes u_1 and u_2, two distinct connections exist between G_1 and G_2. This is true for any pair of nodes in G_1 and so, we conclude that a minimum of $p+q$ edges are needed to connect nodes in G_1 to nodes in G_2.

The total number of edges E in G is bounded below by

$$|E| \geqslant |E_1| + |E_2| + (p+q) \geqslant [(p+q)/2] \log 2p + [(p+q)/2] \log 2p + (p+q)$$
$$= (p+q) \log 4p. \quad \square \tag{1}$$

We should point out that if $m = \Omega(n)$ then the lower bound resembles the one produced for 1-to-1 permutations. This is trivially true in the case where $m = n$. In that case, the lower bound could be rewritten as $\Omega[((m+n)/2] \log 2m) = \Omega(n \log 2n)$.

Theorem 2. *The number of edges of an* $(m, n)^*_{\leqslant k}$*-permutation is bounded below by* $\Omega(n \log 2m)$.

Proof. $V_{m,n}$ can be considered as the union of node disjoint sets V_m and V_n having m inputs and n outputs, respectively. We divide V_n into $[n/m]$ groups of m nodes each. Each group of m outputs must be connected to the group of m inputs via an 1-to-1 permutation network. Assuming that the optimal permutation network is used for each one of the groups, all edges and nodes in a group are used by some permutation that involves nodes in that group. Therefore, no edge or node of a group can be used by another group, otherwise a simultaneous permutation won't be feasible. If $|E_i|$ is the lower bound on the edges required by group i, we can express $|E|$, the total number of edges, as follows:

$$|E| = \sum_{i=1}^{\lceil n/m \rceil} E_i \geqslant \lceil n/m \rceil (m/2) \log 2m \geqslant (n/2) \log 2m = \Omega(n \log 2m). \quad \square$$

Corollary 1. *The number of edges of* $(m, n)_{\leqslant k_i}$*- and* $(m, n)^*_{\leqslant k_i}$*-permutations is bounded below by* $\Omega((m+K) \log 2m)$ *and* $\Omega(K \log 2m)$, *respectively, where* $K = \sum_{i=1}^{m} k_i$.

Proof. It is evident from the previous two theorems if we substitute $\sum_{i=1}^{m} k_i$ for n. $\square$

An important issue of designers of FPICs that implement m-to-n permutations is the performance of the system measured in terms of routing delays for permutation paths. In what follows we will deal with path lengths of permutation paths for m-to-n permutations for which the underlying graph is of maximum node degree d.

Lemma 1. *The maximum path length connecting inputs to outputs in* $(m, n)_{\leqslant k}$*-permutations is bounded below by* $\Omega(\log_{d-1}(m+n))$.

Proof. Let P denote the maximum path length connecting an input to an output in $V_{m,n}$ for some permutation. An input node u can reach at most d nodes in one step, $d(d-1)$ nodes in two steps, and so on. For u to reach all $m+n-1$ nodes in $V_{m,n} - \{u\}$, P steps are needed for $d(d-1)^{P-1} \geqslant m+n-1$. Solving for P produces

$$P \geqslant \log_{d-1}((m+n-1)/d) + 1 = \log_{d-1}(m+n-1) - \log_{d-1} d + 1$$
$$\geqslant \log_{d-1}(m+n-1) - 1 = \Omega(\log_{d-1}(m+n)). \quad \square \tag{2}$$

Corollary 2. *The maximum path length connecting inputs to outputs in an* $(m, n)^*_{\leqslant k}$*-permutations is bounded below by* $\Omega(\log_{d-1} n)$.

Proof. It follows from the fact the maximum number of nodes that must be reached from an input is $n - 1$. □

Corollary 3. *The maximum path length connecting inputs to outputs in* $(m, n)_{\leqslant k_i}$*- and* $(m, n)^*_{\leqslant k_i}$*-permutations is bounded below by* $\Omega(\log_{d-1}(m + K))$ *and* $\Omega(\log_{d-1} K)$, *respectively, where* $K = \sum_{i=1}^{m} k_i$.

Proof. It follows from the previous lemma and corollary by substituting $\sum_{i=1}^{m} k_i$ for n. □

Another important characteristic of m-to-n permutations is the utilization of resources (use of programmable switches). More importantly, we are interested in the average utilization of resources defined as the average total path length for permutation multipaths divided over the total number of edges of the permutation. In what follows, we deal with the average total path length for FPICs that implement m-to-n permutations whose underlying graph is of maximum degree d.

Lemma 2. *The average total path length* T_{avg} *for* $(m, n)_{\leqslant k}$*-permutations is bounded below by* $\Omega(n \log(m + n))$.

Proof. As we have seen earlier, the number of nodes that can be reached in k steps for permutations of bounded node degree d is $d(d-1)^{k-1}$. Let P denote the lower bound on the maximum path length connecting a node u to the rest of the nodes in $V_{m,n}$. Then, the lower average value for a permutation path originating in u is obtaining if we have the maximum number of paths of length $k < P$ and a single path of length P. Let L_{avg} denote the lower bound on this average path length. L_{avg} can be expressed as

$$\begin{aligned} L_{\text{avg}} &\geqslant [d + d(d-1) + \cdots + (P-1)d(d-1)^{P-2} + P]/(m + n - 1) \\ &= (L_a + P)/(m + n - 1), \end{aligned} \tag{3}$$

where

$$L_a = \sum_{i=1}^{P-1} id(d-1)^{i-1} = (P-1)d(d-1)^{P-1}/(d-2) - \sum_{i=0}^{P-2} d(d-1)^i/(d-2). \tag{4}$$

The minimum average value for the total path length is obtained if the number of permutation paths originating in u and having path length less than P is the maximum possible, i.e., $m + n - 2$. We express that as

$$\sum_{i=0}^{P-2} d(d-1)^i = [d(d-1)^{P-1} - d]/(d-2) = m + n - 2. \tag{5}$$

From (5) we deduce

$$d(d-1)^{P-1}/(d-2) \geqslant m+n-1. \tag{6}$$

Using (5) and (6) in (3) produces

$$L_{\text{avg}} \geqslant [(P-1)(m+n-1)-(m+n-2)/(d-2)+P]/(m+n-1) \geqslant P-1-1/(d-2)$$

$$= \Omega(P) = \Omega(\log_{d-1}(m+n)). \tag{7}$$

We assume that each one of the m inputs is going to be connected to the average number of outputs which is $k = \lceil n/m \rceil$. We also assume that k permutation paths each of average path length participate in the permutation multipath forming the connections from input node u to these k output nodes. Since these permutation paths may be partially overlapping, we have to examine the degree of overlapping to determine the average length for the permutation multipath. Assuming that L_{avg} denotes the average length for permutation paths, permutation paths may share a maximum of $L_{\text{avg}}-1$ edges and a minimum of zero edges for an average of $(L_{\text{avg}}-1)/2$ edges. We conclude that the average multipath length M_{avg} is bounded below by

$$M_{\text{avg}} \geqslant k(L_{\text{avg}}-1)/2 = \lceil n/m \rceil (L_{\text{avg}}-1)/2 = \Omega((n/m)\log_{d-1}(m+n)). \tag{8}$$

We select a multipath M of average length originating in u, connecting u to nodes in set $V_{n/m} \subset V_{m,n}$, $|V_{n/m}| = \lceil n/m \rceil$. Since M must be node disjoint to all multipaths connecting nodes in $V_{m,n} - V_{m/n} - \{u\}$, the bounds for the average multipath length apply to the $(m-1)$-to-$(n - \lceil n/m \rceil)$ permutation graph induced by removing M. A repetitive application of this process yields an average total path length T_{avg} quantified as

$$T_{\text{avg}} = \Omega[k(\log_{d_0-1}(m+n) + \log_{d_1-1}(m+n-1-k) + \cdots + \log_{d_{m-1}-1}(k+1))]. \tag{9}$$

In Eq. (9), $k = \lceil n/m \rceil$, $d_0 = d$, and d_j denotes the degree of the induced permutation graph of $(m+n-j(k+1))$ i-nodes. The index $m-1$ in d_{m-1} is also indicative of the fact that we have m logarithmic terms in (9), as many as groups of k outputs. Since $d_j \leqslant d_0 = d$, we can unify all logarithmic terms to base $d-1$. This result in

$$T_{\text{avg}} = \Omega\left(k \sum_{j=0}^{m-1} \log_{d-1}(m+n-j(1+k))\right)$$

$$= \Omega(\lceil n/m \rceil \, m \log_{d-1}(m+n)) = \Omega(n \log_{d-1}(m+n)). \qquad \square \tag{10}$$

Corollary 4. *The minimum number of edges of a bounded degree m-to-n permutation is bounded below by* $\Omega(n \log(m+n))$.

Proof. It follows from the previous lemma since d is a constant and the number of edges is always greater than or equal to the average total path length. $\square$

3. Proposed designs and layouts for *m*-to-*n* permutations

In this section, we are going to propose theoretical and practical designs for m-to-n permutations, $m \leqslant n$. The number of programmable switches used in these designs is within a constant factor of the lower bound established earlier. Furthermore, they are characterized by average utilizations of $\Theta(1)$. The performance of the designs measured in relation to the longest permutation path can be fine-tuned by allowing a slightly larger number of programmable switches. The design of a hybrid model employing the advantages of the Beneš network [9,11] and the crossbar switch is also presented. Here, we should point out that the asymptotic optimality stems from the fact that we have assumed that m can take any value up to n.

In [9,10] we have proposed the r-dimensional k-ary Beneš network and the r-HBC network as attractive designs for 1-to-1 permutations which combine asymptotically optimal average utilization, flexible routing delays, quadratic (on the number of switches) two-layer layout area, and an asymptotically optimal number of programmable switches. Here, we show that the $[(m+n)/2]$-input, $[(m+n)/2]$-output binary Beneš network is also an m-to-n permutation.

Theorem 3. *Given $r = \log s$, where $s = (m+n)/2$, the r-dimensional Beneš network is an $(m, n)_{\leqslant k}$ permutation network.*

Proof. In [11] it has been shown that the Beneš network is a 1-to-1 permutation where all permutations connecting nodes of level 0 to nodes of level r are possible using node-disjoint paths. We will extend the proof to include all input-to-input and output-to-output permutations.

We will use induction on r. If $r = 1$, the network is two 2×2 back-to-back switches and the proof follows trivially. For the inductive hypothesis we assume that the proof holds for up to $(r-1)$-dimensional Beneš networks. We will show it holds for r-dimensional Beneš networks.

We observe that if we remove the first and last level of nodes, we will be left with two $(r-1)$-dimensional Beneš networks, the upper and lower subnetwork. Each node in the first or last level is connected to both the upper and lower subnetwork using one edge for each connection. Specially, nodes $\langle i, 0\rangle$ and $\langle i + s/2 \bmod s, 0\rangle$ are both connected to nodes $\langle i, 1\rangle$ and $\langle i + s/2 \bmod s, 1\rangle$ and nodes $\langle i, r\rangle$ and $\langle i + s/2, r\rangle$ are both connected to nodes $\langle i, r-1\rangle$ and $\langle i + s/2, r-1\rangle$, $i < s$. (A node is labeled $\langle i, j\rangle$ if it is a node in row i, level j).

We will deal with level 0 only, since a proof for level r will be identical by symmetry. Since any node at the first level of the upper (lower) subnetwork can be connected to any other node in the first level of that subnetwork (inductive hypothesis), we can connect any node $\langle i, 0\rangle$ to any node $\langle j, 0\rangle$. To ensure simultaneous connections of all permutations we have to satisfy constraints that occur due to routing of specific pairs. For example, we may assume that the connection of node $\langle i, 0\rangle$ to node $\langle j, 0\rangle$ was realized using the upper subnetwork and nodes $\langle i', 1\rangle$ and $\langle j', 1\rangle$ of the upper subnetwork were used to make the connection. Then, the connection of node $\langle i'', 0\rangle = \langle i + s/2 \bmod s, 0\rangle$ to any other node must be realized using the lower subnetwork. The same holds for the connection of node $\langle j'', 0\rangle = \langle j + s/2 \bmod s, 0\rangle$ to any other node. Node $\langle i'', 0\rangle$ is free to connect to node $\langle i' + s/2 \bmod s, 1\rangle$ of the lower subnetwork. Similarly, node $\langle j'', 0\rangle$ can connect to node $\langle j' + s/2 \bmod s, 1\rangle$ of the lower subnetwork. Since the nodes to which they are going to be connected can connect to both the upper and lower subnetwork, these connections are always possible. Subsequent constraints can always be met the same way.

Furthermore, all such connections are pairwise node-disjoint (by induction on the subnetworks and the fact that no two connections share nodes at level 1).

Connections between nodes of level 0 and between nodes of level r may both use nodes of level $r/2$, therefore, we have to show that they can be realized with no conflict. We observe that any connection using node $\langle 2i, r/2\rangle$ may also use node $\langle 2i+1, r/2\rangle$ and vice versa. Therefore, in case two connections one between level 0 nodes and the other between level r nodes both use the ith one-dimensional Beneš subnetwork, one connection can use node $\langle 2i, r/2\rangle$ while the other node $\langle 2i+1, r/2\rangle$.

Let $U = \{u_i \mid i = 1, 2, \ldots, m\}$ and $V = \{v_j \mid j = 1, 2, \ldots, n\}$ be two disjoint sets of nodes which are going to be the designated input and output nodes of the Beneš network, respectively. Any node in the first or last level of the network can be a member of U or V, i.e., an input or output node.

We have to show that, given a multipath connecting node $u_i \in U$ to nodes $v_j \in V_i \subseteq V$, any connection of node $u_i' \in U - \{u_i\}$ to nodes in $V - V_i$ is feasible simultaneously.

Since the Beneš network is a connected graph, the multipath $P^{u_i}_{V_i}$ connecting u_i to nodes in V_i always exists. We analyze multipath $P^{u_i}_{V_i}$. It is composed of $|V_i|$ possibly partially overlapping paths P_{u_i,v_j} connecting node u_i to nodes $v_j \in V_i$.

The Beneš network is also an 1-to-1 permutation network, thus, there always exist $|U \cup V|$ disjoint paths connecting nodes $U \cup V$ to nodes $U \cup V$. We construct $P^{u_i}_{V_i}$ from paths that would be selected in implementing permutations (u_i, v_j). We claim that any node $u_i' \in U - \{u_i\}$ can then be connected simultaneously to any node $v_{j'}' \in V - V_i$. Let $P_{u_i',v_j'}$ be such a connection and assume that $P_{u_i',v_j'}$ is overlapping with some path P_{u_i,v_j} of multipath $P^{u_i}_{V_i}$. That would mean that simultaneous connection of u_i to v_j and of u_i' to v_j' is not possible, which contradicts the fact that the Beneš network is an 1-to-1 permutation network. □

An illustration of the implementation of a specific m-to-n permutation on the Beneš network is given in Fig. 3. The following multipaths are used to implement connections:

$$P^{\langle 1,0\rangle}_{\{\langle 3,0\rangle,\langle 5,8\rangle,\langle 12,8\rangle\}} \qquad P^{\langle 2,0\rangle}_{\{\langle 15,0\rangle,\langle 7,8\rangle,\langle 8,8\rangle,\langle 16,8\rangle\}}$$
$$P^{\langle 6,0\rangle}_{\{\langle 8,0\rangle,\langle 12,0\rangle,\langle 3,8\rangle\}} \qquad P^{\langle 9,0\rangle}_{\{\langle 16,0\rangle,\langle 15,8\rangle\}}$$
$$P^{\langle 14,0\rangle}_{\{\langle 1,8\rangle,\langle 9,8\rangle,\langle 13,8\rangle\}} \qquad P^{\langle 2,8\rangle}_{\{\langle 4,0\rangle,\langle 7,0\rangle,\langle 10,0\rangle,\langle 14,0\rangle\}}$$
$$P^{\langle 4,8\rangle}_{\{\langle 5,0\rangle,\langle 13,0\rangle,\langle 6,8\rangle,\langle 10,8\rangle\}} \qquad P^{\langle 11,0\rangle}_{\{\langle 11,8\rangle\}}$$

The binary Beneš network of $(m+n)/2$ inputs and $(m+n)/2$ outputs has a total of $2(m+n)\log[(m+n)/2]$ edges and a maximum permutation path length equal to $2\log[(m+n)/2]$ both within a constant factor of the lower bound. Furthermore, the network has an average total path length of $O((m+n)\log(m+n))$ which is asymptotically optimal given that $m \leqslant n$. Since the number of edges in the network is $2(m+n)\log[(m+n)/2]$, the average utilization is O(1).

Alternatively, the r-dimensional k-ary Beneš network has $2rk^{r+1}$ edges and a maximum path length equal to $2r$. Increasing the maximum degree of the Beneš network from 4 for the binary to $2r$ for the r-dimensional network, we can reduce the maximum path length with a trade-off in the number of edges of the network. Although, increasing the performance is highly desirable, this theoretical alternative is difficult to implement due to the increasingly higher node degrees (two- and three-layer layouts are very costly and inefficient). Use of a hybrid model (HBC) in which

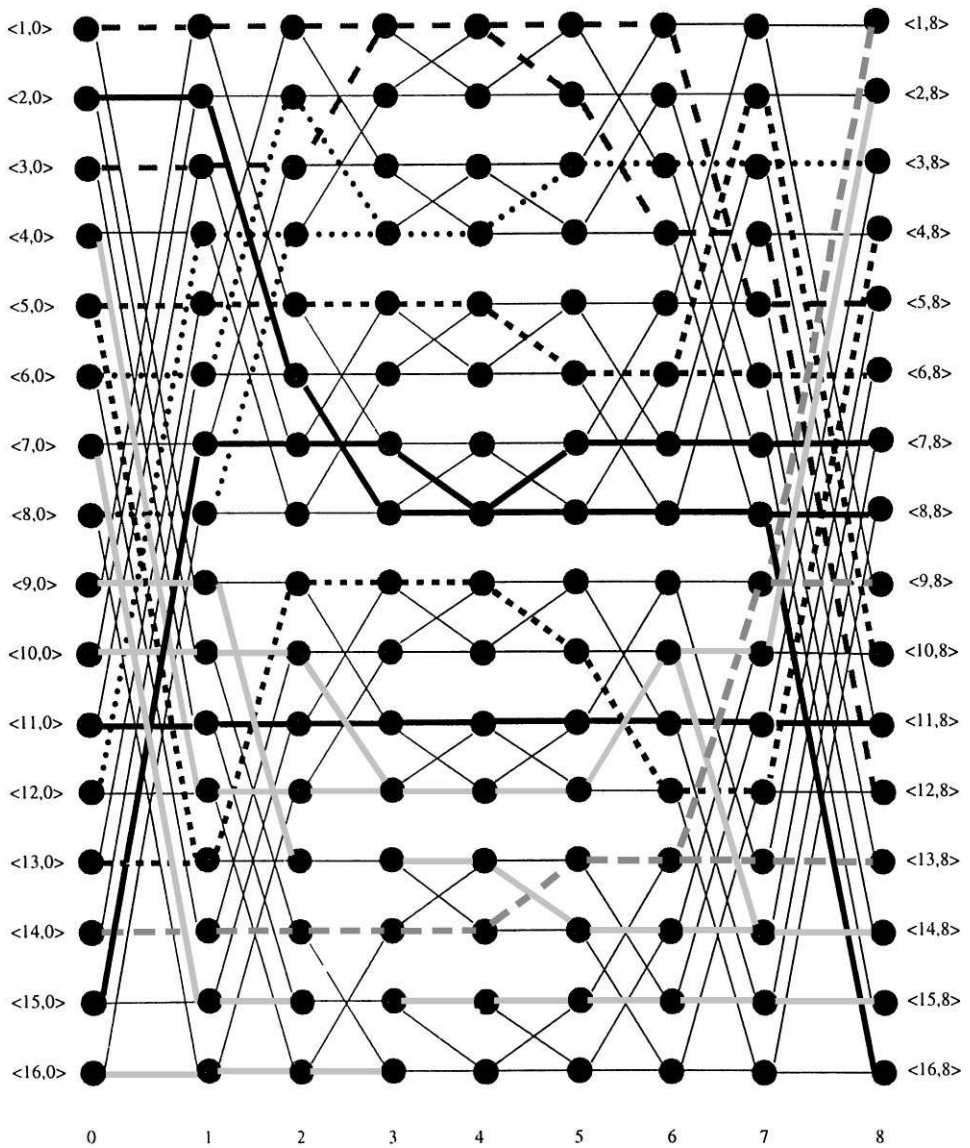

Fig. 3. Routing specific permutations in 16 input Beneš network.

the binary Beneš network is combined with the crossbar switch is a rather attractive solution. The performance-area trade-off is easily controlled by substituting a variable number of levels in the Beneš network with functionally equivalent crossbar switch components. A detailed description of the HBC permutation network is given in [9,10]. An example of a hybrid model is depicted in Fig. 4.

A two-layer layout for the binary Beneš network and the HBC network has been designed in [9,10]. In Fig. 5, we provide the two-layer layout for the three-dimensional binary Beneš network.

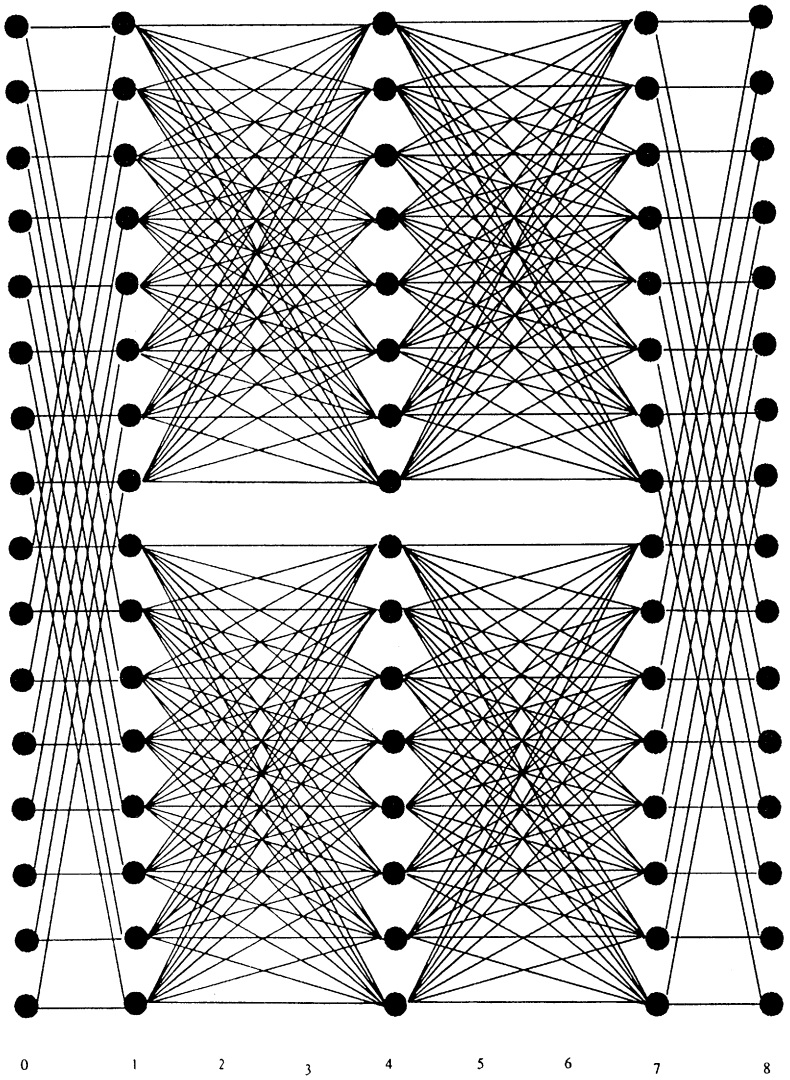

Fig. 4. An example of HBC permutation network.

In [9,10] we have also shown that both layouts use an area equal to $O(n^2)$, where n indicates the number of inputs/outputs of the network. Therefore, we can express the area of the layout for the $(m, n)_{\leqslant k}$-permutation as $O((m+n)^2)$.

A different design is proposed for $(m, n)^{*}_{\leqslant k}$-permutations. We construct the network as a collection of $\lceil n/m \rceil$ m-input, m-output (log m)-dimensional Beneš networks. All $\lceil n/m \rceil$ copies share the input level. An illustration of the design for a $(m, 3m)^{*}_{\leqslant k}$-permutation is given in Fig. 6. A two-layer layout can be constructed by placing the copies side by side and connecting the corresponding inputs. An additional m vertical tracks are sufficient to provide the necessary connections for the inputs in this implementation. The layout of an example network can be seen in Fig. 7. The area of

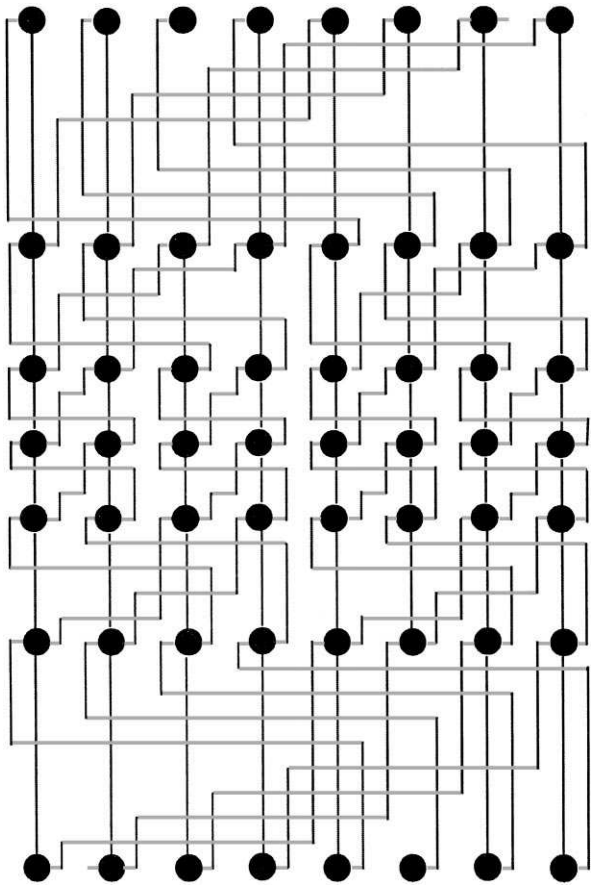

Fig. 5. A two-layer layout for the three-dimensional binary Beneš network.

the layout is $O((m + m^2)\lceil n/m \rceil) = O(mn)$. The maximum path length is equal to $2 \log m$ which is an improvement over the $2 \log[(m + n)/2]$ path length of the design for the $(m, n)_{\leqslant k}$-permutation. Further improvement of the performance of this network with a trade-off in the area of the layout can be achieved by redesigning all $\lceil n/m \rceil$ copies as hybrid models.

We extend our design with a model for $(m, n)_{\leqslant k}$-permutations with a maximum of m input nodes. The Beneš network is a more general design where permutations of all values of m can be implemented. If we assume that m is a fixed upper bound on the number of inputs, we can design an interconnection network whose routing delay is improved by a factor of $\log(m + n)/\log m$. The proposed design is similar to the one proposed for $(m, n)^*_{\leqslant k}$-permutations. We use $\lceil n/m \rceil + 1$ copies of an m-input, m-output Beneš network connected at the output level. Nodes of the input levels are used as the $m + n$ input/output nodes of the interconnection structure.

The layout of this design is very similar to the one for $(m, n)_{\leqslant k}$-permutations with the addition of one copy of an m-input, m-output Beneš network. The area of the layout is $(m + m^2)(\lceil n/m \rceil + 1) = O(n + mn + m^2) = O(mn)$. The maximum path length is $4 \log m$, an improvement over $2 \log[(m + n)/2]$, the maximum path length for an $(\log[(m + n)/2])$-dimensional

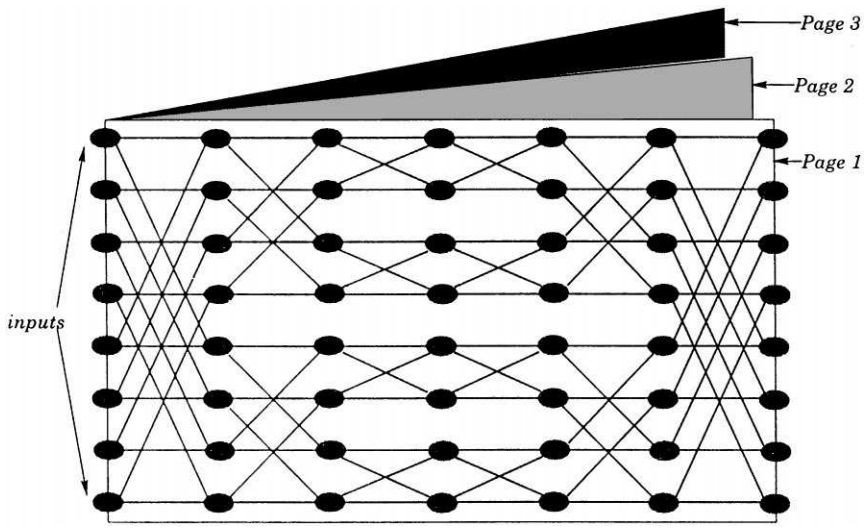

Fig. 6. A network for $(m, 3m)^{*}_{\leqslant k}$-permutations.

Beneš network, for all values of $m < \sqrt{(n/2)}$ ($m < (1 + \sqrt{1 + 8n})/4$ is the exact value). This design is suitable for high-fanout interconnections and it can be used to route power and ground connections.

In both our designs, we observe that the interconnection structure is the Cartesian product of a star graph of $\lceil n/m \rceil + 1$ nodes and an ($\log m$)-dimensional Butterfly ($(m, n)^{*}_{\leqslant k}$-permutations) or a star graph of $\lceil n/m \rceil + 2$ nodes and an ($\log m$)-dimensional Beneš network ($(m, n)_{\leqslant k}$-permutations), respectively. Recall that the $(n + 1)$-node star graph is the minimum size graph for all 1-to-n permutations. We relate this to the fact that an edge is the minimum 1-to-1 permutation and an m-input, m-output 1-to-1 permutation can be considered as the Cartesian product of an edge and an ($\log m$)-dimensional Beneš network.

4. Conclusions

In this paper, we presented a lower bound on the number of programmable switches needed to implement an FPIC capable of providing m-to-n permutations. To our knowledge, this is the first lower bound on m-to-n permutation networks. Nevertheless, assuming that $m = \Omega(n)$, existing networks that realize the lower bound within a constant factor are well known (butterfly, Beneš network). Furthermore, we have shown that m-to-n permutations whose underlying graph is of bounded degree d show routing delays equal to $\Omega(\log_d(m + n))$. For networks of dedicated input and output nodes a new design which asymptotically realizes the lower bound has been proposed. We enhance these designs by proposing a combination of the Beneš network and the crossbar switch, a design which we call the HBC network. We show that these designs have an asymptotically optimal average utilization. Two-layer layouts implementing the proposed designs using an area equal to $O((m + n)^2)$ and $O(mn)$

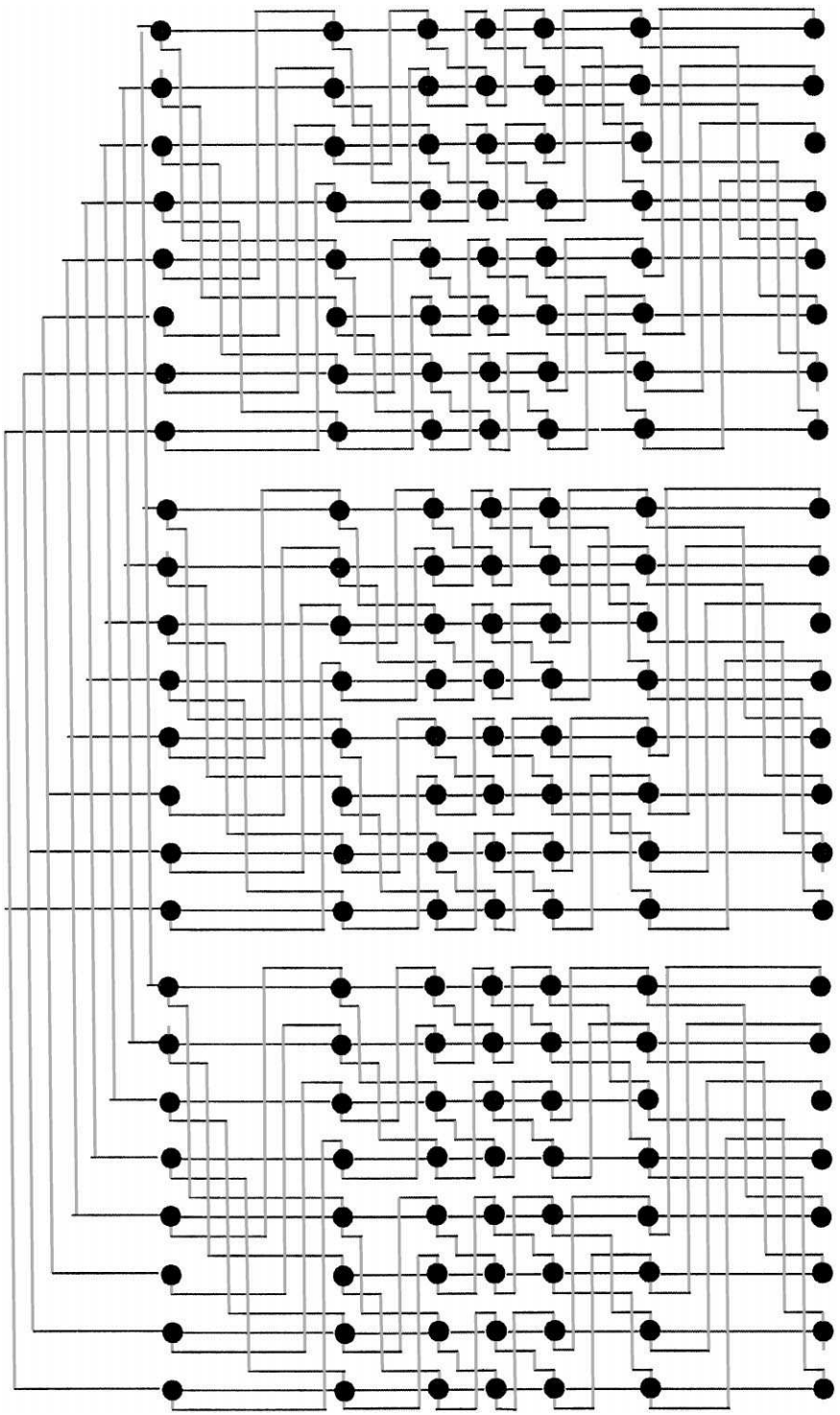

Fig. 7. A two-layer layout of network for $(m, n)^{*}_{\leqslant k}$-permutations.

for m-to-n permutations with non-designated and designated terminals, respectively, are also given.

An interesting open problem is the improvement of the lower bound for m-to-n permutations with no designated terminals, specifically for cases where m is of lower order than n. Finally, interconnection-specific designs for $(m, n)_{\leqslant k_i}$ and $(m, n)^*_{\leqslant k_i}$ permutations that dramatically lower the area of the layout or the number of programmable switches are yet to be explored.

References

[1] J. Varghese, M. Butts, J. Baatcheller, An efficient logic emulation system, IEEE Transactions on VLSI Systems 1 (2) (1993) 171–174.
[2] L. Maliniak, Multiplexing enhances hardware emulation, Electronic Design (1992) 76–78.
[3] ICUBE Data Book, 1993.
[4] Aptix Data Book, Aptix Corporation, 225 Charcot Avenue, San Jose, CA 95131, 1993.
[5] W.K. Mak, D.F. Wong, On optimal board-level routing for FPGA-based logic emulation, Proceedings of 32nd Design Automation Conference, DAC-95, San Francisco, June 1995, pp. 552–556.
[6] W.K. Mak, D.F. Wong, Board level multi-terminal net routing for FPGA-based logic emulation, Proceedings of International Conference on Computer Aided Design, San Jose, November 1995, pp. 339–344.
[7] M.A.S. Khalid, J. Rose, Experimental evaluation of mesh and partial crossbar routing architectures for multi-FPGA systems, Proceedings of 6th International Workshop on Logic and Architecture Synthesis, December 1997, pp. 119–127.
[8] M.A.S. Khalid, J. Rose, A hybrid complete-graph partial-crossbar routing architecture for multi-FPGA systems, Proceedings of 6th ACM Symposium of Field Programmable Gate Arrays, February 1998.
[9] D. Bhatia, J. Haralambides, Resource requirements for field programmable interconnection chips, Proceedings of the VLSI Design Conference, VLSI'95, New Delhi, India, January 1995.
[10] D. Bhatia, J. Haralambides, Bounds and practical layouts for field programmable interconnection chips, IEEE Transactions on VLSI Systems, to appear.
[11] F.T. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes, Morgan Kaufmann, San Francisco, 1992.

Dinesh Bhatia received a Bachelor's degree in Electrical Engineering from Regional Engineering College, Suratkal, India in 1985 followed by a MS and Ph.D. in computer science from University of Texas at Dallas in 1987 and 1990 respectively. Currently he is Associate Professor in the Department of ECECS at the University of Cincinnati. He also directs the Design Automation Laboratory. His research interests include all aspects of architecture and CAD for field programmable gate arrays, reconfigurable and adaptive computing, physical design automation of VLSI systems, and algorithms. His research is actively supported by Defense Advanced Research Projects Agency (DARPA) and the United States Air Force. He is a member of IEEE, ACM, and Eta Kappa Nu.

James Haralambides received the B.S. degree in Computer Engineering and Informatics from the University of Patras, Greece, in 1985, and the MS and Ph.D. in Computer Science in 1989 and 1991, respectively. He is currently an Associate Professor of Computer Science in the Department of Mathematics and Computer Science at Barry University, Miami Shores, Florida. His research interests include VLSI design, computer architecture, and applied graph theory.