A Systematic Analysis of Equivalence in Multi-Stage Networks

Martin Collier

Keywords— **networks, multistage interconnection networks, optical interconnections.**

Abstract— **Many approaches to switching in opto-electronic and optical networks decompose the switching function across multiple stages or hops. This paper addresses the problem of determining whether two multi-stage or multi-hop networks are functionally equivalent. Various ad-hoc methods have been used in the past to establish such equivalences. A systematic method for determining equivalence is presented based on properties of the link permutations used to interconnect stages of the network. This method is of use in laying out multistage networks, in determining optimal channel assignments for multi-hop networks, and in establishing the routing required in such networks. A purely graphical variant of the method, requiring no mathematics or calculations, is also described.**

I. Introduction

THE growth in demand for communications switching capability is set to exceed the capacity of all-electronic switching systems. Switch designers have responded by incorporating opto-electronic and optical elements in switches and routers (e.g., [1], [2]). The use of optical elements offers potentially larger bandwidth and reduced crosstalk compared to purely electronic systems. The different capabilities of optical systems, such as bend radius constraints in waveguides, space variance considerations in free-space optics, and the absence of low-cost optical memory, mean that novel switch architectures are required to implement optical and opto-electronic packet switching [3].

Many designs for optical and opto-electronic switching distribute the switching function across multiple small switching elements, whose interconnection pattern dictates the overall switch properties. Examples include Shufflenet [4] and GEMNET [5]. These are implemented either as multi-stage networks (where there is a one-to-one correspondence between links in the logical switch architecture and links in the physically realised network) or as multi-hop networks (where links in the logical architecture correspond to WDM channels in the implementation, and where physical interconnection is typically achieved using a passive optical star).

Many such designs are based on architectures originally proposed for electronic switching such as the omega network [6] and the butterfly network. This paper addresses the question of when two such networks can be said to be functionally equivalent. Two networks are regarded as functionally equivalent in this paper if they cannot be distinguished on the basis of the response at the output ports to any pattern of inputs. This topic is important for a number of reasons.

- It is of theoretical importance, since if a class of networks can be shown to be functionally equivalent, theoretical results obtained for one member of the class may be applied to all members of the class.
- It is of practical importance in the construction of such networks, since it is rarely possible to lay out such networks exactly in accordance with their formal description.
- It provides insight into the properties of such networks.

This problem has been addressed many times in the past, using a variety of ad-hoc methods (e.g., [7], [8], [9], [10], [11]). However, the interconnections between stages in optical multi-stage switches pose unique problems, such as optical crosstalk in systems based on directional couplers [12], and image rotation in systems featuring polymer fibre-image guides [2]. These issues can lead switch designers to seek alternative interconnection patterns, which result in unchanged functionality of the switch, but which avoid the problems associated with implementing classical interconnection patterns optically.

Hence a demand exists for a systematic procedure for investigating functional equivalences in multi-stage networks. Such a procedure is presented here, which may readily be implemented as a computer program, and which applies to a wide class of multi-stage networks. The extended procedure to be applied in the case where a network does not belong to this class will be presented in another paper.

II. Notation and Approach

A prerequisite for the comparison of two multi-stage networks is a common method of description. The approach here uses the notation of Chapter Two of [13] with some refinements required for the problem in hand. We consider regular networks comprising n stages of $k \times k$ switch elements, and which thus have k^n inputs and outputs. For the remainder of the paper, it shall be assumed that $k = 2$. This is in the interests of notational simplicity — the extension of the method described here to other values of k is straightforward.

An obvious tool to use in investigating the properties of multi-stage networks is graph theory, since the links and nodes of the switch map readily into the edges and vertices of the graph. However, whilst graph theory allows us to explore the *connectivity* of a network, it tells us little about *routing* in the network (and thus blocking), unless the vertices are labelled. A distinction may be made between three levels of equivalence: *topological equivalence*, *isomorphism*

M. Collier is with the Research Institute for Networks and Communications Engineering (RINCE), Dublin City University, Dublin 9, Ireland. E-mail: martin.collier@rince.ie. This paper is scheduled to appear in September 2002

and *functional equivalence* ([13], section 2.1.1). Two networks are functionally equivalent if they support the same permutations, i.e., if they have the same blocking properties. They are isomorphic if a re-ordering of the switch elements, inputs and outputs of one network gives rise to the second. They are topologically equivalent if they have the same graph.

For example, consider the simple networks shown in Fig. 1. The two networks shown have the same graph (shown in Fig. 1(c))and so by definition are topologically equivalent, but the routing algorithm in each is different.[1]

Labelling the vertices circumvents this difficulty, allowing the two networks to be distinguished. The same result will be achieved in this paper by introducing the concepts of *link addresses* and *link permutations.* Note that 2^n links enter and leave each stage of the switch. The link address is simply the number of the link on which a packet leaves the stage, with address 000. . . 0 being the address of the lowest link.

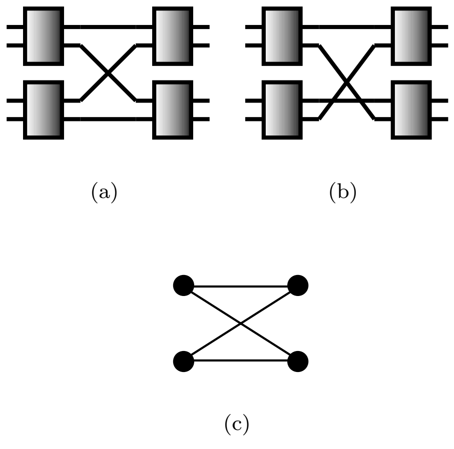

Fig. 1. (a) A simple network. (b) Another network. (c) the corresponding graph.

The link address can thus be represented by a binary number of the form $b_{n-1}b_{n-2}\dots b_0$. When a packet exits a stage of the switch, the interconnection pattern between it and the downstream stage causes the packet to enter the downstream stage with a (typically) new link address. This change of address is regarded as the result of a *link permutation* which may be defined as a transformation of the form $LP : x \to y$ where $x, y \in \{0, \dots, 2^{n-1}\}$ and where $LP(x_1) = LP(x_2) \Leftrightarrow x_1 = x_2$.

Consider the $2^n \times 2^n$ network shown in Fig. 2. It is completely defined by the set of $n + 1$ link permutations $\{LP_n, LP_{n-1}, \dots, LP_0\}$. In this paper, attention shall be restricted to a class of link permutations called bit-shuffling

[1]The routing algorithm is the same in both cases if output ports are defined in terms of the downstream switch elements to which they are connected. However, high-speed routing requires that the routing decision in a 2×2 switch should reduce to "exit via the upper port" or "exit via the lower port", the outcome of which decision is different for the switch element in the southwest corner of Fig. 1(a) and Fig. 1(b) respectively.

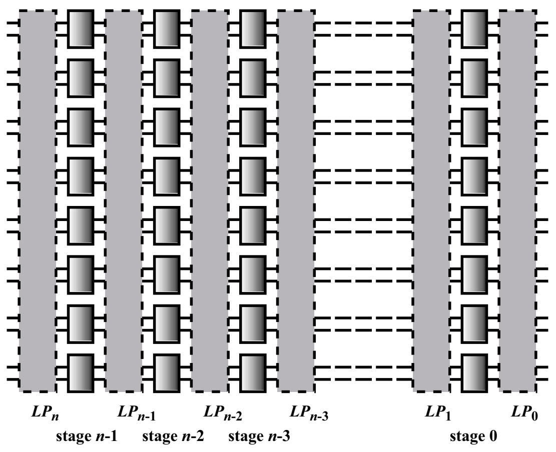

Fig. 2. A generic multi-stage network using 2×2 switch elements. The dashed boxes are replaced by link permutations to construct a specific network. Stages are numbered from right to left - this simplifies the description of the routing for some common multistage networks. Ports are numbered from the bottom up so that the term "upper port" is unambiguous.

permutations. A link permutation LP is a *bit-shuffling permutation* if $LP(b_{n-1}b_{n-2}\dots b_0) = b_{s_{n-1}}b_{s_{n-2}}\dots b_{s_0}$ where $s_j \in \{0, \dots, n-1\}$ and $s_j = s_k \Leftrightarrow j = k$ (the value of s_j indicates the *source* of the bit in position j at the link permutation output). The link permutations used in the majority of multistage networks using 2×2 switching elements belong to this family of link permutations. Some examples are presented below.

The *perfect shuffle [14] of scope* 2^k, denoted σ_{k-1}, is a bit-shuffling permutation where:

$$s_j = \begin{cases} k-1 & \text{if } j = 0; \\ j-1 & \text{if } 0 < j < k; \\ j & \text{if } j \geq k. \end{cases}$$

The *inverse perfect shuffle of scope* 2^k, denoted $\bar{\sigma}_{k-1}$, is a bit-shuffling permutation where:

$$s_j = \begin{cases} j+1 & \text{if } 0 \leq j < k-1; \\ 0 & \text{if } j = k-1; \\ j & \text{if } j \geq k. \end{cases}$$

The *butterfly permutation of scope* 2^k, denoted β_{k-1}, is a bit-shuffling permutation where:

$$s_j = \begin{cases} k-1 & \text{if } j = 0; \\ 0 & \text{if } j = k-1; \\ j & \text{if } 0 < j < k-1, j \geq k. \end{cases}$$

The *bit-reversal permutation of scope* 2^k, denoted ρ_{k-1}, is a bit-shuffling permutation where:

$$s_j = \begin{cases} k-1-j & \text{if } 0 \leq j < k; \\ j & \text{if } j \geq k. \end{cases}$$

The *straight-through* or *identity permutation*, denoted I is a bit-shuffling permutation where:

$$s_j = j \ \forall j.$$

One of the key benefits of the link permutation notation above is that it can be used to investigate what happens if one link permutation is immediately followed by another. This can be described mathematically using a product notation, essentially the same as that used for matrix multiplication. For example, suppose that a link permutation LP_a is immediately followed by a link permutation LP_b. A signal entering the first link permutation on link x will emerge on link $LP_a(x)$, and enter the second link permutation on that link, emerging at link $LP_b(LP_a(x))$. The concatenation of these two link permutations is equivalent to a single link permutation LP_c defined by:

$$LP_c(x) = LP_b(LP_a(x))$$

written in reverse order as

$$LP_c(x) = xLP_aLP_b$$

so as to correspond to the order of link permutations in diagrams.

Hence LP_c may be described as the product

$$LP_c = LP_aLP_b.$$

A link permutation LP_a is said to be the *inverse transformation* of a link permutation LP_b if, and only if:

$$LP_aLP_b = I.$$

Then LP_a may be written as:

$$LP_a = (LP_b).^{-1}$$

Some properties of concatenated link permutations used later in the paper are given in the Appendix. The framework of notation above can now be used to compactly define many well-known multi-stage networks.

The *baseline network* [7] is defined by:

$$LP_k = \begin{cases} I & \text{if } k = 0, k = n; \\ \bar{\sigma}_k & \text{if } 0 < k < n. \end{cases} \quad (1)$$

A 16 × 16 baseline network is shown in Fig. 3 (the significance of the paths through the switch indicated by dashed lines will be explained later).

The *omega network* [6] is defined by:

$$LP_k = \begin{cases} I & \text{if } k = 0; \\ \sigma_{n-1} & \text{if } 0 < k \leq n. \end{cases} \quad (2)$$

A 16 × 16 omega network is shown in Fig. 4.

The *butterfly network*[2] or *n-cube network*[15] is defined by:

$$LP_k = \begin{cases} I & \text{if } k = 0; \\ \beta_k & \text{if } 0 < k < n; \\ \sigma_{n-1} & \text{if } k = n. \end{cases} \quad (3)$$

A 16 × 16 butterfly network is shown in Fig. 5.

[2]Many authors define the butterfly as the corresponding *reverse* network

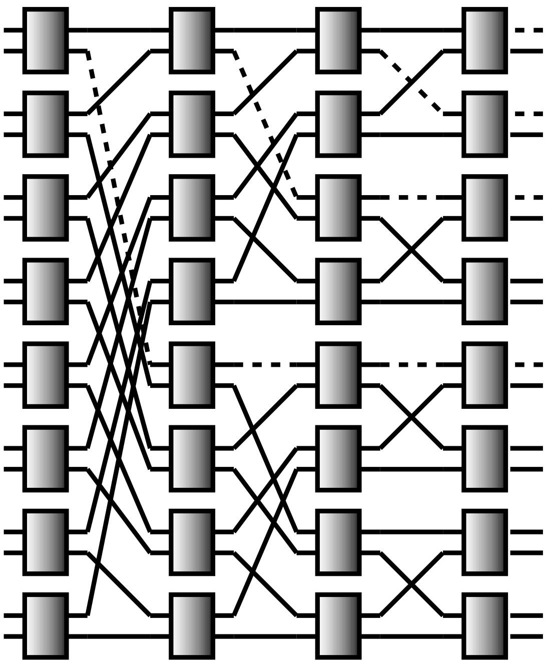

Fig. 3. A 16 × 16 baseline network.

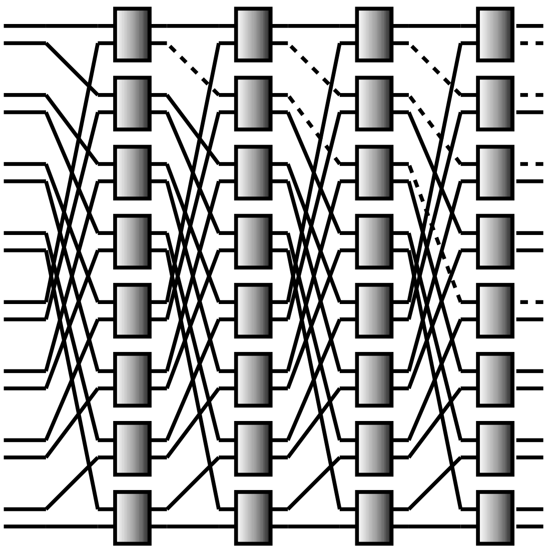

Fig. 4. A 16 × 16 omega network.

The reverse network (network B) of network A may be obtained by setting $LP^B_k = (LP^A_{n-k})^{-1}$for $0 \leq k \leq n$. Hence the *reverse butterfly network* is defined by:

$$LP_k = \begin{cases} \bar{\sigma}_{n-1} & \text{if } k = 0; \\ \beta_{n-k} & \text{if } 0 < k < n; \\ I & \text{if } k = n. \end{cases} \quad (4)$$

The *reverse baseline network* is defined by:

$$LP_k = \begin{cases} I & \text{if } k = 0, k = n; \\ \sigma_{n-k} & \text{if } 0 < k < n; \end{cases} \quad (5)$$

The *segmented two-shuffle network* is a rather more complex network, proposed by Cloonan *et al.* [11] for use with

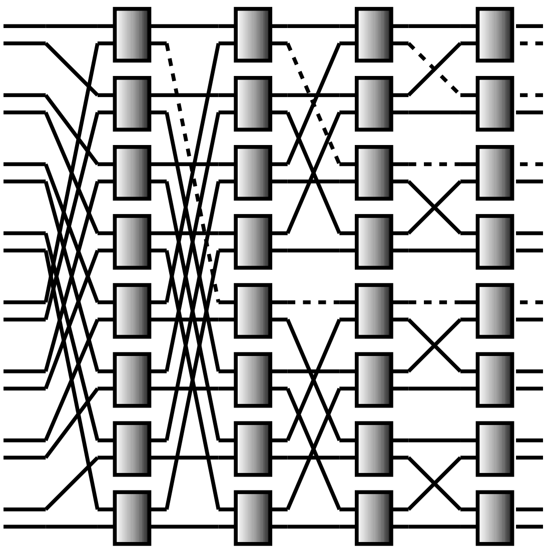

Fig. 5. A 16×16 butterfly network.

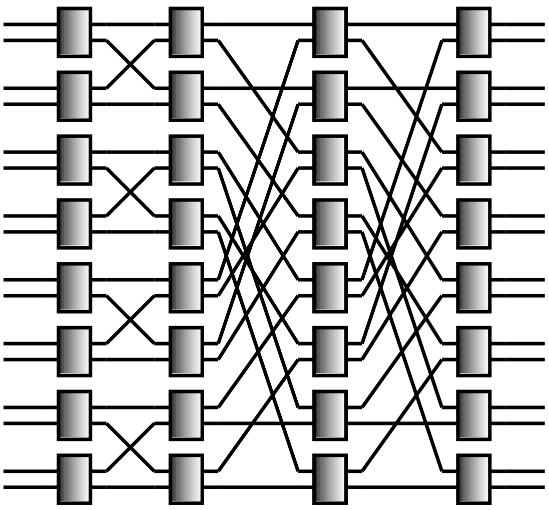

Fig. 6. A segmented two-shuffle network with 16 inputs and two columns.

free-space optical interconnects. This is an example of a 3-D network, where each switch element is defined by its position in a row and column within each stage. Hence the link address is defined by a triple (R, C, b_0) identifying the row and column occupied by the switch element, and to which of its input or output ports the link is connected. The number of columns (called the segment size in [11]) is $H = 2^h$. Hence the number of rows is 2^{n-1-h} for a $2^n \times 2^n$ switch. The switch illustrated in Figs. 5(b) and 7(c) of [11] features the parameters $n = 5$ and $h = 2$. Fig. 6 shows such a switch with parameters $n = 4$ and $h = 1$.

The switch uses the perfect shuffle as a link permutation in every stage. However, for h stages of the switch, the perfect shuffle is performed horizontally (across columns) and in the other stages it is performed vertically (across rows). Formally the switch (network A) may be defined by the following link permutations:

$$LP_k^A = \begin{cases} I & \text{if } k = 0, k = n; \\ \sigma_{n-h-1} \text{ applied to } (R, b_0) & \text{if } 0 < k < n - h; \\ \sigma_h \text{ applied to } (C, b_0) & \text{if } n - h \leq k < n. \end{cases}$$

Writing the link address (R, C, b_0) as a uni-dimensional n-bit address $R.2^{h+1} + C.2 + b_0$, this may be rewritten as:

$$LP_k^A = \begin{cases} I & \text{if } k = 0, k = n; \\ \sigma_{n-1}\bar{\sigma}_{h+1}\beta_{h+1} & \text{if } 0 < k < n - h; \\ \sigma_h & \text{if } n - h \leq k < n. \end{cases} \tag{6}$$

III. Functional Equivalence

A. The Tramline Transformation

The approach taken to establishing functional equivalence of two networks A and B takes as a starting point that used by Wu and Feng in their classic papers [7],[8] on this topic. They used the concept of a logical numbering of switch elements in a stage to establish this equivalence — if the logical numbers describing a switch element and the two downstream elements to which it is connected in network A correspond to the physical numbers of switch elements in network B, then the two networks are isomorphic.

Here, the switch elements in network A are *physically moved* to effect the reordering. If, after this operation, the link permutations between stages are identical to those in network B, isomorphism has been established.

When a switch element is moved to a new position with respect to its neighbours in a switching stage, the change in the switch layout which results may be described using the concept of the *tramline transformation.*

A *tramline transformation* of 2^n links is a link permutation $TT : x \to y$ where y is even for x even and:

$$\begin{aligned} TT(2x) &= 2y \quad \Leftrightarrow \\ TT(2x+1) &= TT(2y+1), \quad \text{for } 0 \leq x < 2^{n-1}. \end{aligned}$$

If a switching stage is preceded by a tramline transformation and followed by the *inverse* of that transformation, then the ordering of switch elements in that stage is changed, but the connectivity to the rest of the network is unaffected. This is illustrated in Fig. 7.

B. Verifying equivalence

This suggests a scheme for verifying the equivalence of two networks A and B. Each stage k of network A is preceded by a link permutation T_k^{AB} and followed by the inverse permutation $(T_k^{AB})^{-1}$. Hence, signals exiting from stage k of the modified network A traverse $(T_k^{AB})^{-1}$, then LP_k and finally T_{k-1}^{AB} before entering stage $k-1$. This is illustrated in Fig. 8. If a set $\{T_k^{AB}\}$ can be found for which each link permutation in the set (for $0 \leq k < n$) is a tramline transformation, and for which

$$LP_k^B = (T_k^{AB})^{-1} LP_k^A \; T_{k-1}^{AB} \text{ for } 0 < k < n. \tag{7}$$

then networks A and B are isomorphic.

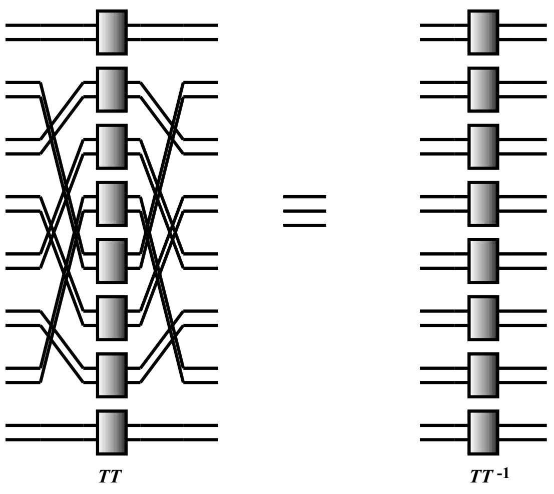

Fig. 7. An example of a tramline transformation.

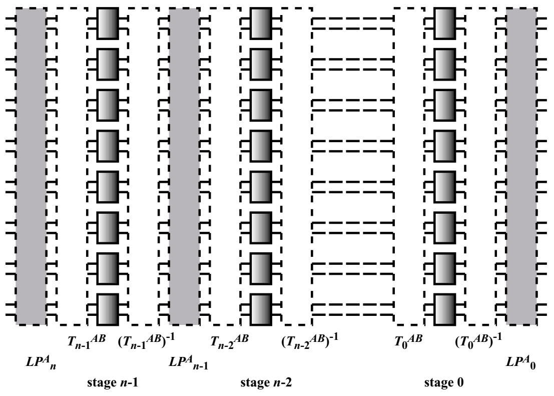

Fig. 8. A network (A) with the switch elements repositioned using tramline transformations so as to replicate the layout of another network (B).

It can easily be shown, by manipulating the above identity using the rules of matrix multiplication, that these transformations satisfy the recurrence relation:

$$T_k^{AB} = LP_k^A T_{k-1}^{AB}\ (LP_k^B)^{-1} \text{ for } 0 < k < n. \qquad (8)$$

It may be shown by induction that:

$$T_k^{AB} = DP_k^A T_0^{AB}\ (DP_k^B)^{-1} \text{ for } 0 < k < n, \qquad (9)$$

where DP_k is called the *downstream permutation*[3] for stage k and is defined as:

$$DP_k = LP_k LP_{k-1} \dots LP_1 \text{ for } 0 < k < n. \qquad (10)$$

DP_k^A and DP_k^B may be determined from a knowledge of the link permutations in networks A and B, and since, by assumption, these are all bit-shuffling permutations, so too are the respective downstream permutations.

Defining $lsb^A(k)$ as the bit position to which bit zero is shuffled by DP_k^A (with a corresponding definition for network B), it follows that T_k^{AB} performs the following operation on bit zero of its input if it is a tramline transformation:

$$b_0 \xrightarrow{DP_k^A} b_{lsb^A(k)} \xrightarrow{T_0^{AB}} b_{lsb^B(k)} \xrightarrow{(DP_k^B)^{-1}} b_0, \text{ for } 0 < k < n$$

Hence, and observing that T_0^{AB} must also be a tramline transformation, it must satisfy the following properties:

$$\begin{aligned} &T_0^{AB}(b_{n-1}b_{n-2}\dots b_0) = b_{s_{n-1}}b_{s_{n-2}}\dots b_{s_0}, \\ &\qquad \text{where:} \\ &s_0 = 0; \\ &s_{lsb^B(k)} = lsb^A(k) \quad \text{for } 0 < k < n. \end{aligned} \qquad (11)$$

This transformation must be bijective if T_0^{AB} is to be a link permutation. If this property is satisfied, it may be concluded that the two networks are isomorphic. It will be shown presently that, for a network of the form of Fig. 2, where all the link permutations are bit-shuffling permutations, a necessary and sufficient condition for all network outputs to be reachable from the inputs is that the set $\{lsb(k)\}$, $0 < k < n$, be a permutation of the set $\{1, 2, \dots, n-1\}$.

C. The downstream permutation

A physical interpretation of the downstream permutation and the function $lsb(k)$ may now be given. It indicates the link address at the output of stage 0 at which packets will appear, denoted $d'_{n-1}d'_{n-2}\dots d'_0$. To see this, consider a packet entering the network with a routing tag $r_{n-1}r_{n-2}\dots r_0$. It first enters stage $n-1$, where it is switched on the basis of bit r_{n-1}, is routed in stage $n-2$ on the basis of bit r_{n-2}, and so forth. How is its destination determined by the routing tag? Consider the possible destinations of the packet after leaving a switch element in stage $n-1$ (the input stage). Suppose that the switch elements downstream from stage $n-1$ are all configured in the straight-through position, so that the switch may be considered to perform the DP_k permutation on the packet's link address between the output side of stage $n-1$ and the output side of stage 0. The least significant bit of the link address at the output side of stage $n-1$ is dictated by r_{n-1}, the switching decision at stage $n-1$. The least significant bit of the link address becomes bit $lsb(n-1)$ of the link address $d'_{n-1}d'_{n-2}\dots d'_0$ at the output of stage 0, by definition of the $lsb(\ )$ function. In summary,

$$d'_{lsb(n-1)} = r_{n-1} \qquad (12)$$

for certain settings of the downstream switch elements.

A similar argument may be used to show that after the packet passes through the next stage of the switch (stage $n-2$), the switching decision there, for certain settings of the downstream switch elements, is such that

$$d'_{lsb(n-2)} = r_{n-2}.$$

[3]so-called because it is the permutation which results between stage k and the switch outputs if all switch elements downstream of stage k are configured in the "straight-through" position.

However, if $lsb(n-1) \neq lsb(n-2)$, it follows that the value of $d'_{lsb(n-1)}$ is independent of r_{n-2}. Thus, (12) holds regardless of the setting of the switch elements in stage $n-2$. Applying this argument iteratively to the downstream stages, it may be concluded that (12) holds for all settings of the downstream switch elements, if there is no stage k for which $lsb(k) = lsb(n-1)$. Applying the same logic to the other switch stages, it follows that a packet can be routed to any link address $d'_{n-1}d'_{n-2}\ldots d'_0$ at the output of stage 0 by setting :

$$r_0 \leftarrow d'_0,$$

and

$$r_j \leftarrow d'_{lsb(j)}, \text{ for } 0 < j < n,$$

provided that

$$lsb(j) = lsb(k) \Leftrightarrow j = k \quad \text{for } 0 < j < n, 0 < k < n$$

and

$$lsb(j) \neq 0 \text{ for } 0 < j < n,$$

or, equivalently, provided that the set

$$\{lsb(1), lsb(2), \ldots, lsb(n-1)\}$$

is a permutation of the set $\{1, 2, \ldots, n-1\}$. Since stage 0 is followed by the link permutation LP_0, the actual destination of the packet is:

$$d_{n-1}d_{n-2}\ldots d_0 = LP_0(d'_{n-1}d'_{n-2}\ldots d'_0).$$

Hence, the routing strategy for the switch may readily be determined from a knowledge of LP_0 and the set $\{lsb(k)\}$.

D. Exact equivalence and blocking

Note that, from Fig. 8, in order for the numbering of input and output ports to be identical in networks A and B, it is required that:

$$LP^A_n T^{AB}_{n-1} = LP^B_n$$

and

$$(T^{AB}_0)^{-1} LP^A_0 = LP^B_0.$$

If this is the case for two isomorphic networks, they are said to be *exactly equivalent.* Exact equivalence implies functional equivalence, although it is a stronger property. If network A is not exactly equivalent to network B it may be preceded by a link permutation

$$T^{AB}_{in} = LP^B_n (T^{AB}_{n-1})^{-1} (LP^A_n)^{-1} \tag{13}$$

and followed by a link permutation

$$T^{AB}_{out} = (LP^A_0)^{-1} T^{AB}_0 LP^B_0 \tag{14}$$

to obtain a network exactly equivalent to B. These link permutations are *not* required to be tramline transformations.

If two networks are functionally equivalent, not only are the routing algorithms for the two networks the same, but the stages in the switch at which internal contention will occur between two packets will be identical. There are few contexts in which functional equivalence between two networks is required [4]. Thus, it is rarely necessary to introduce the link permutation T^{AB}_{in}. T^{AB}_{out} may also be dispensed with, if the routing algorithm of network A is chosen to emulate the routing in network B.

The permutation T^{AB}_{out} may be used to determine the routing algorithm required in network A, if the routing algorithm for network B is known, and both networks are isomorphic. For example, suppose that it is known that the appropriate routing tag for network B is $r_{n-1}r_{n-2}\ldots r_0 = d_{n-1}d_{n-2}\ldots d_0$, i.e., that the switching decision required at stage j is $r_j = d_j$. It follows from the concept of exact equivalence that, if network A is followed by a link permutation T^{AB}_{out}, it will also route packets to the output port with address $r_{n-1}r_{n-2}\ldots r_0$. If T^{AB}_{out} is omitted, data will be routed to the destination:

$$d_{n-1}d_{n-2}\ldots d_0 = (T^{AB}_{out})^{-1}(r_{n-1}r_{n-2}\ldots r_0).$$

It follows that the required pattern of bits in the routing tag for network A is given by:

$$r_{n-1}r_{n-2}\ldots r_0 = T^{AB}_{out}(d_{n-1}d_{n-2}\ldots d_0). \tag{15}$$

E. Multi-hop networks

The discussion up to this point applies to multi-stage rather than multi-hop networks. Logically multi-hop architectures derived from multi-stage networks may be distinguished fr om the latter by three properties:
1. Each switch element acts as a source and sink of packets.
2. The outputs of the switch are looped back to the inputs. Thus the numbering of stages is arbitrary.
3. The destination of a packet is defined by the stage number and switch element address, not by the link address.

Clearly, two multi-hop networks A and B can be shown to be isomorphic by breaking the loop between stages 0 and $n-1$ and establishing the equivalence of the resulting multi-stage networks. The routing strategy which routes a packet to switch element i in stage k of network B will route a packet to switch element $(T^{AB}_k)'(i)$ in network A, where $T'_k(x) = T_k(2x)/2$.

IV. Some examples

A. Downstream permutations for some popular networks

A.1 The butterfly network

The downstream permutation for this network is, from (3) and (10), $DP_k = \beta_k\beta_{k-1}\ldots\beta_1$. We know from Identity 1 in the Appendix that $DP_k = \bar{\sigma}_k$. This moves bit zero to bit k. Hence $lsb(k) = k$. Recall that bit $lsb(k)$ of a packet's link address at the outputs of stage zero is determined by the switching decision at stage k (for $0 < k < n$). Since $LP_0 = I$ for the butterfly network, it

[4]A notable exception is when data is sorted by destination address prior to being submitted to a multi-stage network. The resulting switch is strictly nonblocking only if the multi-stage network is functionally equivalent to the butterfly network.

follows that the routing strategy is to switch the packet at stage k based on bit k of the destination address. Hence $r_{n-1}r_{n-2}\dots r_0 = d_{n-1}d_{n-2}\dots d_0$.

A.2 The Omega network

Here, using (10) and (2), $DP_k = (\sigma_{n-1})^k$. Hence $lsb(k) = k$ and the routing strategy required is identical to that of the butterfly network.

A.3 The baseline network

Here, $DP_k = \bar{\sigma}_k\bar{\sigma}_{k-1}\dots\bar{\sigma}_1 = \rho_k$ (using Identity 2). Hence $lsb(k) = k$, implying the same routing strategy as the two earlier networks.

A.4 The reverse butterfly network

From (4), $DP_k = \beta_{n-k}\beta_{n-k+1}\dots\beta_{n-1}$. It follows from Identity 3 that $DP_k = \bar{\sigma}_{n-k-1}\sigma_{n-1}$. The permutation $\bar{\sigma}_{n-k-1}$ moves bit zero to the position of bit $n-k-1$. The permutation σ_{n-1} moves the bit in position $n-k-1$ to bit position $n-k$. Thus, $lsb(k) = n-k$ for $0 < k < n$. It follows that packets appear at link address $r_0r_1\dots r_{n-1}$ at the output of stage 0, where r_k is the switching decision made at stage k. They are then re-ordered by LP_0, appearing at the switch output $r_{n-1}r_0r_1\dots r_{n-2}$.

A.5 The reverse baseline network

From (5), $DP_k = \sigma_{n-k}\sigma_{n-k+1}\dots\sigma_{n-1}$. It follows from Identity 4 that $DP_k = \rho_{n-k-1}\rho_{n-1}$. Hence $lsb(k) = k$. Thus the routing is the same as in the butterfly network.

B. *Equivalence of the butterfly network and the Omega network*

The simplicity of the method described here shall be demonstrated by verifying a well-known result. The equivalence of these two networks was established by Wu and Feng in [7][5]. Network A is the butterfly network. We have found above that $DP_k^A = \bar{\sigma}_k$ and $lsb^A(k) = k$. Network B is the Omega network, for which $DP_k = (\sigma_{n-1})^k$ and $lsb^B(k) = k$. It follows immediately that $T_0^{AB} = I$ and thus that both networks are isomorphic. Let us now calculate the tramline transformations.

$$\begin{aligned} T_k^{AB} &= DP_k^A T_0^{AB}\ (DP_k^B)^{-1} \quad (\text{for } 0 < k < n)\\ &= \bar{\sigma}_k\ I\ ((\sigma_{n-1})^k)^{-1}\\ &= \bar{\sigma}_k(\bar{\sigma}_{n-1})^k. \end{aligned}$$

In particular,

$$T_{n-1}^{AB} = \bar{\sigma}_{n-1}(\bar{\sigma}_{n-1})^{n-1} = I.$$

Hence

$$\begin{aligned} T_{in}^{AB} &= LP_n^B(T_{n-1}^{AB})^{-1}(LP_n^A)^{-1}\\ &= \sigma_{n-1}(\sigma_{n-1})^{-1} = I;\\ T_{out}^{AB} &= (LP_0^A)^{-1}T_0^{AB}LP_0^B = I. \end{aligned}$$

[5]They considered the modified data manipulator rather than the butterfly network - these networks differ only in LP_n.

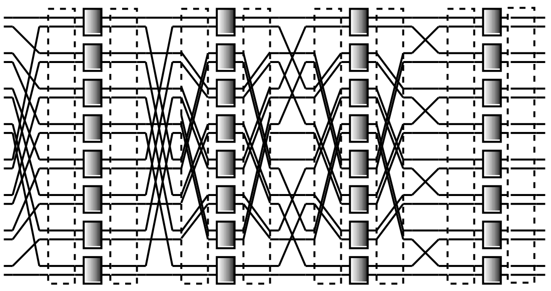

Fig. 9. Equivalence of butterfly and Omega networks.

Thus the two networks are exactly equivalent. The tramline transformations required for the 16×16 case are presented below and are illustrated in Fig. 9.

$$\begin{aligned} T_0^{AB} &: b_3b_2b_1b_0 \to b_3b_2b_1b_0;\\ T_1^{AB} &: b_3b_2b_1b_0 \to b_1b_3b_2b_0;\\ T_2^{AB} &: b_3b_2b_1b_0 \to b_2b_1b_3b_0;\\ T_3^{AB} &: b_3b_2b_1b_0 \to b_3b_2b_1b_0. \end{aligned}$$

In general, it may easily be shown that

$$T_k^{AB} : b_{n-1}b_{n-2}\dots b_0 \to b_kb_{k-1}\dots b_1b_{n-1}b_{n-2}\dots b_{k+1}b_0. \tag{16}$$

A computer program implementing the above formula can readily calculate the transformations for arbitrarily large switches.

C. *Equivalence of the baseline network and the butterfly network*

Network A is the baseline network. Here $DP_k^A = \rho_k$ and $lsb^A(k) = k$. Network B is the butterfly network so $lsb^B(k) = k$. Again it follows immediately that $T_0^{AB} = I$ and thus that both networks are isomorphic. Also:

$$\begin{aligned} T_k^{AB} &= DP_k^A T_0^{AB}\ (DP_k^B)^{-1} \quad (\text{for } 0 < k < n)\\ &= \rho_k\sigma_k. \end{aligned}$$

In particular,

$$T_{n-1}^{AB} = \rho_{n-1}\sigma_{n-1} = \delta_{n-1},$$

where δ_{n-1} is the *bit switch permutation* defined in [8]. Hence

$$\begin{aligned} T_{in}^{AB} &= LP_n^B(T_{n-1}^{AB})^{-1}(LP_n^A)^{-1}\\ &= \sigma_{n-1}\sigma_{n-1}^{-1}\rho_{n-1}\ I = \rho_{n-1};\\ T_{out}^{AB} &= (LP_0^A)^{-1}T_0^{AB}LP_0^B = I. \end{aligned}$$

Thus the two networks are isomorphic but not exactly equivalent. The tramline transformations required for the 32×32 case are presented below.

$$\begin{aligned} T_0^{AB} &: b_4b_3b_2b_1b_0 \to b_4b_3b_2b_1b_0;\\ T_1^{AB} &: b_4b_3b_2b_1b_0 \to b_4b_3b_2b_1b_0;\\ T_2^{AB} &: b_4b_3b_2b_1b_0 \to b_4b_3b_1b_2b_0; \end{aligned}$$

$$T_3^{AB} : b_4b_3b_2b_1b_0 \rightarrow b_4b_1b_2b_3b_0;$$

$$T_4^{AB} : b_4b_3b_2b_1b_0 \rightarrow b_1b_2b_3b_4b_0.$$

In general,

$$T_k^{AB} : b_{n-1}b_{n-2}\dots b_0 \rightarrow b_{n-1}b_{n-2}\dots b_{k+1}b_1b_2\dots b_kb_0. \tag{17}$$

D. Equivalence of the baseline network and its reverse network

Network A is the baseline network so $lsb^A(k) = k$. Network B is the reverse baseline network so $lsb^B(k) = k$. Again it follows immediately that $T_0^{AB} = I$ and thus that both networks are isomorphic. Also:

$$\begin{aligned} T_k^{AB} &= DP_k^A T_0^{AB} \, (DP_k^B)^{-1} \text{ (for } 0 < k < n) \\ &= \rho_k(\rho_{n-k-1}\rho_{n-1})^{-1} \\ &= \rho_k\rho_{n-1}\rho_{n-k-1}. \end{aligned}$$

In particular,

$$T_{n-1}^{AB} = \rho_{n-1}\rho_{n-1}\rho_0 = I.$$

Hence

$$T_{in}^{AB} = LP_n^B(T_{n-1}^{AB})^{-1}(LP_n^A)^{-1} = I;$$

$$T_{out}^{AB} = (LP_0^A)^{-1}T_0^{AB}LP_0^B = I.$$

Thus the baseline network and its reverse network are exactly equivalent. The tramline transformations required for the 16×16 case are presented below.

$$T_0^{AB} : b_3b_2b_1b_0 \rightarrow b_3b_2b_1b_0;$$

$$T_1^{AB} : b_3b_2b_1b_0 \rightarrow b_1b_3b_2b_0;$$

$$T_2^{AB} : b_3b_2b_1b_0 \rightarrow b_2b_1b_3b_0;$$

$$T_3^{AB} : b_3b_2b_1b_0 \rightarrow b_3b_2b_1b_0.$$

It may be shown that

$$T_k^{AB} : b_{n-1}b_{n-2}\dots b_0 \rightarrow b_kb_{k-1}\dots b_1b_{n-1}b_{n-2}\dots b_{k+1}b_0.$$

Note that these are the same transformations required to convert the butterfly network to the Omega network.

E. Equivalence of the segmented two-shuffle network and the butterfly network

Attention is now given to a rather more complex problem, which is to establish that the segmented two-shuffle network is isomorphic to the butterfly network. From (6),

$$DP_k^A = \begin{cases} (\sigma_{n-1}\bar{\sigma}_{h+1}\beta_{h+1})^k & \text{if } 0 < k < n-h; \\ (\sigma_h)^{k-n+h+1}DP_{n-h-1}^A & \text{if } n-h \le k < n. \end{cases}$$

Working out the bit-shuffles performed by these downstream permutations is straightforward, if tedious. It may be shown that

$$DP_k^A(b_{n-1}b_{n-2}\dots b_0) = b_{s_{n-1}^k}b_{s_{n-2}^k}\dots b_{s_0^k}$$

where, for $0 < k < n-h$:

$$s_j^k = \begin{cases} n-k & \text{if } j = 0; \\ j & \text{if } 0 < j \le h; \\ n-h+j-k & \text{if } h < j < h+k; \\ 0 & \text{if } j = h+k; \\ j-k & \text{if } h+k < j < n, \end{cases}$$

and for $n-h \le k < n$:

$$s_j^k = \begin{cases} h+1 & \text{if } j = 0; \\ j-k+n & \text{if } 0 < j < \delta_k; \\ j-\delta_k & \text{if } \delta_k \le j \le h; \\ j+1 & \text{if } h < j < n-1; \\ n-k & \text{if } j = n-1, \end{cases}$$

where $\delta_k = k-n+h+1$. To verify equivalence with the butterfly network, all that is required of the above results is the values of j for which $s_j^k = 0$. This gives:

$$lsb^A(k) = \begin{cases} h+k & \text{if } 0 < k < n-h; \\ k-n+h+1 & \text{if } n-h \le k < n. \end{cases}$$

Since $lsb^A(k) = lsb^A(j) \Leftrightarrow k = j$ it follows immediately that the segmented two-shuffle is isomorphic to the butterfly network. This result was obtained far more compactly than that presented in [11], which applied only for a single value of h. It also provides more insight into the operation of the switch, since it allows the routing algorithm to be inferred.

It was established earlier that for network B (the butterfly network), $lsb^B(k) = k$. Hence, T_0^{AB} is defined by:

$$\begin{aligned} &T_0^{AB}(b_{n-1}b_{n-2}\dots b_0) \\ &= b_hb_{h-1}\dots b_1b_{n-1}b_{n-2}\dots b_{h+1}b_0. \end{aligned}$$

Also $LP_0^A = LP_0^B = I$ and so $T_{out}^{AB} = T_0^{AB}$. Applying (15) it follows that the required pattern of bits in the routing tag of the segmented two-shuffle network is given by:

$$\begin{aligned} r_{n-1}r_{n-2}\dots r_0 &= T_{out}^{AB}(d_{n-1}d_{n-2}\dots d_0) \\ &= d_hd_{h-1}\dots d_1d_{n-1}d_{n-2}\dots d_{h+1}d_0. \end{aligned}$$

Thus for the network in Fig. 6 the routing tag required is $d_1d_3d_2d_0$ and for the network in [11] it is $d_2d_1d_4d_3d_0$.

The general form for T_k^{AB} is rather complex. However it may easily be evaluated using (9) for specific networks. For example, the tramline transformations relating to the network in Fig. 6 are:

$$T_0^{AB} : b_3b_2b_1b_0 \rightarrow b_1b_3b_2b_0;$$

$$T_1^{AB} : b_3b_2b_1b_0 \rightarrow b_1b_2b_3b_0;$$

$$T_2^{AB} : b_3b_2b_1b_0 \rightarrow b_1b_3b_2b_0;$$

$$T_3^{AB} : b_3b_2b_1b_0 \rightarrow b_1b_3b_2b_0.$$

V. A graphical method for checking equivalence

The rule for any $2^n \times 2^n$ network constructed using n stages of 2×2 switch elements to be equivalent to a butterfly network, if the interconnection patterns between stages are all bit-shuffling link permutations, is that $lsb(k) = lsb(j) \Leftrightarrow k = j$. In other words, the least significant bit of the input link address should map to a unique bit in the output link address, for each downstream permutation. This may be checked graphically as follows. At each stage of the switch, draw a line from the switch element output port corresponding to link address $00\ldots01$ (or $11\ldots10$), following the links between stages and passing straight through each downstream switch. If none of the resulting n lines converge, the network is equivalent to a butterfly network. This is illustrated in Figs. 3 and 4. The butterfly network itself (Fig. 5) also has this property. This method may also be applied in synthesising new networks, for example by tweaking a single link permutation until the property holds.

VI. Conclusions

A novel method has been presented for investigating the equivalences between multi-stage networks, where the interconnections between stages are bit-shuffling permutations. It has been demonstrated that complex networks may be easily and systematically analysed using the approach presented in this paper. The key to the method is to devise a set of tramline transformations to demonstrate that two networks are isomorphic. The method also allows the routing algorithm for the network to be easily determined and suggests a graphical approach for checking equivalences. Clearly, extending the method to networks of $k \times k$ switch elements is straightforward, requiring only some enhanced notation to record k. It also has other areas of application, such as in sorting networks. The generality of the method means that it will be of use to the designers of optical multi-stage and multi-hop networks, when they encounter practical constraints which preclude the use of classical multi-stage designs such as the butterfly network.

The method requires extension if it is to be applied to networks where the link permutations are not restricted to the class of bit-shuffling permutations, such as the crossover network [16], because of difficulties caused by a phenomenon the author calls tramline twisting. This will be the subject of a future paper.

Appendix

The following properties of concatenated link permutations are used in the paper.

Identity 1: $\beta_k\beta_{k-1}\ldots\beta_1 = \bar{\sigma}_k$.

Proof: It may easily be shown that $\bar{\sigma}_k = \beta_k\bar{\sigma}_{k-1}$. The result follows by induction. ■

Identity 2: $\bar{\sigma}_k\bar{\sigma}_{k-1}\ldots\bar{\sigma}_1 = \rho_k$.

Proof: It may easily be shown that $\rho_k = \bar{\sigma}_k\rho_{k-1}$. The result follows by induction. ■

Identity 3: $\beta_k\beta_{k+1}\ldots\beta_n = \bar{\sigma}_k\sigma_n$.

Proof: Inverting both sides of Identity 1, and evaluating for $k = n$ gives:

$$\beta_1\beta_2\ldots\beta_n = \sigma_n. \tag{18}$$

Multiplying Identity 1 and (18), and observing that $\beta_j\beta_j = I$, the result follows. ■

Identity 4: $\sigma_k\sigma_{k+1}\ldots\sigma_n = \rho_{k-1}\rho_n$.

Proof: It may be established by induction, since $\rho_k = \rho_{k-1}\sigma_k$, that

$$\sigma_1\sigma_2\ldots\sigma_k = \rho_k. \tag{19}$$

Hence

$$\begin{aligned}\sigma_k\sigma_{k+1}\ldots\sigma_n &= (\sigma_1\sigma_2\ldots\sigma_{k-1})^{-1}(\sigma_1\sigma_2\ldots\sigma_n)\\ &= \rho_{k-1}^{-1}\rho_n \text{ (using (19)).}\end{aligned}$$

The result follows. ■

Acknowledgment

The author wishes to acknowledge the constructive feedback of the reviewers.

References

[1] H. S. Hinton, T. J. Cloonan, F. B. McCormick, A. L. Lentine, and F. A. Tooley, "Free-space digital optical systems," *Proceedings of the IEEE*, vol. 82, pp. 1632–1649, November 1994.

[2] Y. Li, J. Ai, and J. Popelek, "Board-level 2-D data-capable optical interconnection circuits using polymer fiber-image guides," *Proceedings of the IEEE*, vol. 88, pp. 794–805, June 2000.

[3] S. Yao, B. Mukherjee, and S. Dixit, "Advances in photonic packet switching: an overview," *IEEE Communications Magazine*, vol. 38, pp. 84–94, February 2000.

[4] M. G. Hluchyj and M. J. Karol, "ShuffleNet: an application of generalized perfect shuffles to multihop lightwave networks," *IEEE Journal of Lightwave Technology*, vol. 9, pp. 1386–1397, October 1991.

[5] J. Iness, S. Banerjee, and B. Mukherjee, "GEMNET: A generalized, shuffle-exchange-based, regular, scalable, modular, multihop, WDM lightwave network," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 470–476, August 1995.

[6] D. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, vol. 24, pp. 1145–1155, December 1975.

[7] C.-L. Wu and T.-Y. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comput.*, vol. 29, pp. 694–702, August 1980.

[8] C.-L. Wu and T.-Y. Feng, "The reverse-exchange interconnection network," *IEEE Trans. Comput.*, vol. 29, pp. 801–811, September 1980.

[9] T. Cloonan, "Topological equivalence of optical crossover networks and modified data manipulator networks," *Applied Optics*, vol. 28, pp. 2494–2498, 1 July 1989.

[10] S. Even and A. Litman, "Layered cross product - a technique to construct interconnection networks," in *ACM Symposium on Parallel Algorithms and Architectures*, pp. 60–69, 1992.

[11] T. J. Cloonan, G. W. Richards, R. Morrison, A. Lentine, J. Sasian, F. McCormick, S. Hinterlong, and H. Hinton, "Shuffle-equivalent interconnection topologies based on computer-generated binary-phase gratings," *Applied Optics*, vol. 33, pp. 1405–1430, March 1994.

[12] Y. Pan, C. Qiao, and Y. Yang, "Optical multistage interconnection networks: new challenges and approaches," *IEEE Communications Magazine*, vol. 37, pp. 50–56, February 1999.

[13] A. Pattavina, *Switching Theory: Architecture and Performance in Broadband ATM networks*. John Wiley and Sons, 1998.

[14] H. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. 20, pp. 153–161, February 1971.

[15] H. Siegel and R. McMillen, "The multistage cube: a versatile interconnection network," *IEEE Comput.*, vol. 14, pp. 65–76, December 1981.

[16] J. Jahns and M. Murdocca, "Crossover networks and their optical implementation," *Applied Optics*, vol. 25, pp. 3155–3160, August 1988.

Martin Collier received the B.Eng. and M.Eng. degrees in Electronic Engineering at the National Institute for Higher Education, Dublin in 1986 and 1988 respectively. He was awarded a Ph.D. degree by Dublin City University in 1993. He is currently a lecturer at Dublin City University where he runs the Switching and Systems Laboratory. He established the Research Institute for Networks and Communications Engineering (RINCE) at DCU in 1999. His research interests include programmeable networks, quality of service, and advanced switching techniques.