

Puzzling Permutations

Mary Sheeran

Computing Science Department, Chalmers Tekniska Högskola
Göteborg, Sweden

Abstract

We show how to describe and analyse multistage interconnection networks in a relational framework. We give simple elegant descriptions of butterfly networks, using only two functions for building networks from smaller networks. By studying the algebra of these functions, we can give a clear account of the properties of networks, without resorting to the standard methods of reasoning in terms of binary representations of the addresses of data elements. The paper is intended as a tutorial introduction to multistage interconnection networks.

1 Introduction

This paper shows how to describe and reason about multistage interconnection networks. It introduces some standard networks and shows how they are related. I hope that the paper will be useful as an introduction to multistage interconnection networks for computing scientists. At the end, I present a puzzle from network theory. Ideally, I should also present the solution, but I have not been able to solve the puzzle! Since the question has been open for many years, perhaps I should not be surprised. But maybe you, dear reader, will be able to find the solution after reading the paper.

2 Permutations and switches

We consider permutations on sequences whose length is a power of two. A sequence is either a unit sequence containing a , written $[a]$, or the concatenation of two equal length sequences, written $r \ddagger s$. For convenience, we also write sequences using the usual $[a, b, \dots]$ notation, so that $[a] \ddagger [b]$ is written $[a, b]$.

We view a permutation as a binary relation between sequences. The identity relation id_n represents the identity permutation on sequences of length 2^n . In this paper, all the relations discussed relate a given sequence only to sequences of the same length. In the relation R_n , the subscript n indicates that the length of the sequences in both domain and range is 2^n ; it can be viewed as a type annotation.

The exchange permutation is defined to be the smallest relation satisfying

$$[a, b] \text{ } ex \text{ } [b, a]$$

for all a and b .

The converse of a relation is defined by $a R^{-1} b \equiv b R a$. Both id_n and ex are their own converses. An example of a permutation that is not is the cyclic shift to the left. For example, $[a, b, c, d] \text{ } lc_2 \text{ } [b, c, d, a]$, but $[a, b, c, d] \text{ } lc_2^{-1} \text{ } [d, a, b, c]$. Another example of a permutation that is (in general) not its own converse is the perfect shuffle, which we call \mathcal{S} .

$$\begin{aligned} [a, b] \mathcal{S}_1 [a, b] &\equiv \text{true} \\ (a \ddagger b) \ddagger (c \ddagger d) \mathcal{S}_{n+1} (e \ddagger f) &\equiv (a \ddagger c) \mathcal{S}_n e \quad \& \quad (b \ddagger d) \mathcal{S}_n f \end{aligned}$$

The permutation \mathcal{S}_2 relates $[0, 1, 2, 3]$ to $[0, 2, 1, 3]$ and is its own converse. The permutation \mathcal{S}_3 relates $[0, 1, 2, 3, 4, 5, 6, 7]$ to $[0, 4, 1, 5, 2, 6, 3, 7]$, whereas \mathcal{S}_3^{-1} relates $[0, 1, 2, 3, 4, 5, 6, 7]$ to $[0, 2, 4, 6, 1, 3, 5, 7]$.

We use ordinary relational composition to compose permutations. Any permutation composed with its converse gives an identity. So, for example $ex ; ex^{-1} = ex ; ex = id_1$ and $\mathcal{S}_n ; \mathcal{S}_n^{-1} = id_n$.



Figure 1: two ex , two sw and the permutations realised by two sw

Repeated composition is abbreviated as $R^1 = R$ and $R^{n+1} = R; R^n$. We write $(R^n)^{-1}$ as R^{-n} . For a permutation P , it is the case that $P^{-q}; P^q = P^0$ is an identity, but beware of assuming that this kind of arithmetic in the indices works for more general relations.

A two-input two-output switch is defined as

$$sw = id_1 + ex$$

where $+$ is relational union. The switch is capable of realising two different permutations, id_1 and ex . In our network modelling method, the semantics of a switch or network of switches is exactly the set of permutations that it can realise (viewed as a binary relation). We say that two networks are equivalent if and only if they realise the same set of permutations. This is the only notion of network equivalence that we need. As a first example of a pair of equivalent networks, let us calculate

$$\begin{aligned} sw ; sw &= (id_1 + ex) ; (id_1 + ex) \\ &= (id_1 ; id_1) + (id_1 ; ex) + (ex ; id_1) + (ex ; ex) \\ &= id_1 + ex + ex + id_1 \\ &= id_1 + ex \\ &= sw \end{aligned}$$

So the composition of two switches in series realises the same permutations as a single switch. When reasoning about networks, we will feel free to replace $sw ; sw$ by sw .

3 Building networks

We introduce *combinators* for building larger networks from smaller ones. The first is called *two* and it places two identical networks one above the other.

$$(a \ddagger b) \text{ two } R (c \ddagger d) \equiv a R c \& b R d$$

For example, the network *two sw*, shown in figure 1, relates the sequence $[0, 1, 2, 3]$ to each of $[0, 1, 2, 3]$, $[1, 0, 2, 3]$, $[0, 1, 3, 2]$ and $[1, 0, 3, 2]$, and vice versa.

The network *two(two sw)* is made up of two copies of *two sw* and it realises 16 distinct permutations. It is convenient to abbreviate this kind of repeated application of a combinator:

$$\begin{aligned} \text{two}^0 R &= R \\ \text{two}^{n+1} R &= \text{two}(\text{two}^n R) \end{aligned}$$

The *two* combinator interacts nicely with composition and converse.

$$\begin{aligned} \text{two } R ; \text{two } R &= \text{two}(R ; S) \\ (\text{two } R)^{-1} &= \text{two } R^{-1} \end{aligned}$$

The second combinator is called *ilv* and it interleaves two identical networks, so that one of them works on the even elements of the domain and range, and the other on the odd elements.

$$ilv R_n = \mathcal{S}_{n+1}^{-1} ; \text{two } R_n ; \mathcal{S}_{n+1}$$

We abbreviate repeated composition of *ilv* in the same way as we do for *two*. The wiring network $ilv^n ex$ swaps the two halves of a sequence, while $two^n ex$ swaps adjacent elements. The network *ilv sw* relates $[0, 1, 2, 3]$ to $[0, 1, 2, 3]$,



Figure 2: The network $\text{two ilv } sw$, which is the same as $\text{ilv two } sw$

$[2, 1, 0, 3]$, $[0, 3, 2, 1]$ and $[2, 3, 0, 1]$. For P_n a permutation, the permutation $\text{ilv } P_n$ can be thought of as pairing adjacent elements, permuting the resulting sequence and then unpairing. In other words, it “preserves” adjacent pairs. This means that

$$\text{ilv } P_n ; \text{two}^n sw = \text{two}^n sw ; \text{ilv } P_n$$

and similarly

$$\text{two } P_n ; \text{ilv}^n sw = \text{ilv}^n sw ; \text{two } P_n$$

The properties of ilv derive from those of two .

$$\begin{aligned} \text{ilv } R ; \text{ilv } S &= \text{ilv}(R ; S) \\ (\text{ilv } R)^{-1} &= \text{ilv } R^{-1} \end{aligned}$$

Figure 2 shows the network $\text{two ilv } sw$. If you look carefully, you will see that it also shows $\text{ilv two } sw$! The two networks are equivalent and realise the same permutations.

In fact for any R_n ,

$$\text{two ilv } R_n = \text{ilv two } R_n$$

We will not prove this fact here. (It is proved in reference [11].) We say that two and ilv commute and this is the key insight in our analysis of butterfly and related networks. A consequence is that

$$\text{two}^i \text{ilv}^j R_k = \mathcal{S}^{-j} ; \text{two}^{i+j} R_k ; \mathcal{S}^j \quad (1)$$

Here we have omitted subscripts where size can be deduced from context. We will often do this.

We can now use the combinators to give *recursive* descriptions of permutations and networks. For example, to reverse a sequence, swap the halves and reverse each half:

$$\begin{aligned} rev_0 &= id_0 \\ rev_{n+1} &= \text{ilv}^n ex ; \text{two } rev_n \end{aligned}$$

By induction, we can also show that

$$rev_{n+1} = \text{ilv } rev_n ; \text{two}^n ex$$

which corresponds to reversing the even and odd elements in place and then exchanging those two subsequences by swapping adjacent elements.

The perfect shuffle has a similar recursive decomposition:

$$\begin{aligned} \mathcal{S}_{n+2} &= \text{ilv}^n \mathcal{S}_2 ; \text{two } \mathcal{S}_{n+1} \\ &= \text{ilv } \mathcal{S}_{n+1} ; \text{two}^n \mathcal{S}_2 \end{aligned}$$

Because $\mathcal{S}_2^2 = id_2$, we can compose $\text{two}^n \mathcal{S}_2$ on both sides and get

$$\mathcal{S}_{n+2} ; \text{two}^n \mathcal{S}_2 = \text{ilv } \mathcal{S}_{n+1} \quad (2)$$

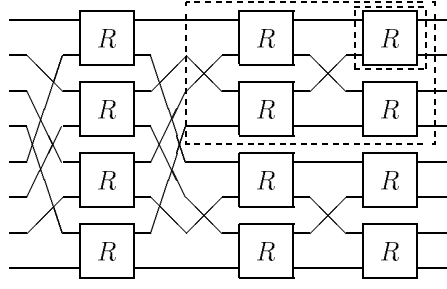


Figure 3: the first recursive decomposition of the butterfly

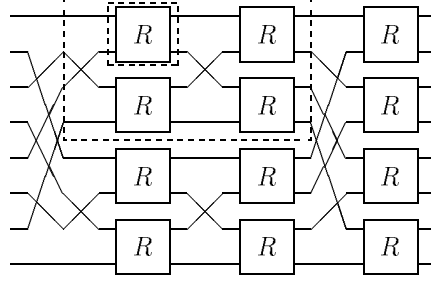


Figure 4: a second recursive decomposition of the butterfly

4 Describing the butterfly network

The pattern of twos and ilvs that appears in both reverse and the perfect shuffle is exactly the pattern used to build a butterfly network. Define, for $k > 0$,

$$\begin{aligned} \bowtie_0 R_k &= id_{k-1} \\ \bowtie_{n+1} R_k &= ilv^n R_k ; \text{two}(\bowtie_n R_k) \end{aligned}$$

This first recursive decomposition of the butterfly is illustrated (for $k = 1$) in figure 3.

We have seen that

$$\begin{aligned} rev_n &= \bowtie_n ex \\ \mathcal{S}_{n+1} &= \bowtie_n \mathcal{S}_2 \end{aligned}$$

From the first recursive decomposition of the butterfly, we can show by induction that

$$\bowtie_{n+1} R = ilv(\bowtie_n R) ; \text{two}^n R \quad (3)$$

This is the second well-known recursive decomposition of the butterfly and it is shown in figure 4.

There are also many possibilities in between these two. It can be shown, by induction, that

$$\bowtie_{i+j} R = ilv^i(\bowtie_j R) ; \text{two}^j(\bowtie_i R) \quad (4)$$

It is also useful to consider iterative descriptions of the butterfly. We use a continued version of relational composition to compose indexed sequences of relations: $\mathop{\bullet}_{i=0}^n R(i) = R(0) ; R(1) \dots R(n)$. Then

$$\bowtie_{n+1} R = \mathop{\bullet}_{i=0}^n ilv^{n-i}(\text{two}^i R) \quad (5)$$

If we now use equation 1 to replace each term in the repeated composition, we find that

$$\mathcal{S}^{n+1} ; \bowtie_{n+1} R_k = \mathcal{S}^{n+1} ; \mathop{\bullet}_{i=0}^n (\mathcal{S}^{-(n-i)} ; \text{two}^n R_k ; \mathcal{S}^{n-i})$$

$$\begin{aligned}
&= \mathop{\textcircled{\text{;}}}_{i=0}^n (\mathcal{S}^{n+1-i} ; \mathcal{S}^{-(n-i)} ; \text{two}^n R_k) ; \mathcal{S}^0 \\
&= \mathop{\textcircled{\text{;}}}_{i=0}^n (\mathcal{S} ; \text{two}^n R_k) \\
&= (\mathcal{S} ; \text{two}^n R_k)^{n+1}
\end{aligned}$$

This expresses the relationship between the butterfly network and the so-called ‘shuffle network’, which consists of a series of identical slices, each of which is a perfect shuffle composed with $\text{two}^n R$ [16].

Now we can use the fact that the perfect shuffle can be expressed as a butterfly to prove by induction that $\mathcal{S}_n^n = id_n$, that is, that if you do enough perfect shuffles on a list of length 2^n , then you get back to where you started. The base case is $\mathcal{S}_1^2 = id_1^2 = id_1$. For the step:

$$\begin{aligned}
\mathcal{S}_{n+2}^{n+2} &= \mathcal{S}_{n+2}^{n+1} ; \mathcal{S}_{n+2} \\
&= \mathcal{S}_{n+2}^{n+1} ; \text{\textcircled{\text{;}}}_{n+1} \mathcal{S}_2 \\
&= \{ \text{above calculation} \} \\
&\quad (\mathcal{S}_{n+2} ; \text{two}^n \mathcal{S}_2)^{n+1} \\
&= \{ \text{equation 2} \} \\
&\quad (i|v \mathcal{S}_{n+1})^{n+1} \\
&= i|v \mathcal{S}_{n+1}^{n+1} \\
&= \{ \text{inductive hypothesis} \} \\
&\quad i|v id_{n+1} \\
&= id_{n+2}
\end{aligned}$$

And this, combined with the previous calculation allows us to conclude that in the special case where R relates 2-sequences

$$\text{\textcircled{\text{;}}}_{n+1} R_1 = (\mathcal{S}_{n+1} ; \text{two}^n R_1)^{n+1} \quad (6)$$

If we replace \mathcal{S}_{n+1} by \mathcal{S}_{n+1}^{-n} and then use equation 1, we find that

$$\text{\textcircled{\text{;}}}_{n+1} R_1 = (i|v^n R_1 ; \mathcal{S}_{n+1})^{n+1} \quad (7)$$

If the component of the butterfly is a two-input, two-output comparator, then we got the circuit known as Batcher’s bitonic merger [1]. It sorts an input consisting of a increasing followed by a decreasing sequence, and is the merger component of Batcher’s bitonic sorter [15]. If the component is a two-point Fourier transform, then we get the Fast Fourier Transform [7, 10]. So the pattern of recursion captured in the butterfly is a common one.

5 A butterfly of switches

One of the most studied interconnection networks is the butterfly of switches. Define

$$\Omega_n = \text{\textcircled{\text{;}}}_n s w$$

The relation Ω_n characterises exactly those permutations realised by a butterfly network of two-input two-output switches. There is no need for a separate description of that set. We describe network structure (using recursion and the combinators two and $i|v$) and behaviour (the allowed permutations) at the same time. To build both networks and permutations with the same combinators is an important simplification, and one that is made surprisingly rarely in the literature on interconnection networks. We would argue that there is no neater way than this to describe such networks.

Drawing the butterfly network using the notation for switches introduced earlier makes it easier to see the recursive structure of the network. You could think of taking the diagram in figure 3, say, and pulling hard on the wires at each end so that the R components are stretched vertically. These straight line diagrams are both easier to draw, and, I think,

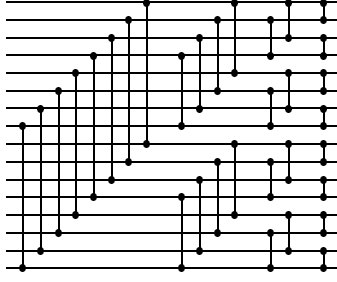


Figure 5: Ω_4

easier to read. Not many of the papers in the literature use this notation. One that does is Bilardi's excellent discussion of merging and sorting networks [4].

We know from equation 6 that

$$\Omega_{n+1} = (\mathcal{S}_{n+1}; \text{two}^n \text{ sw})^{n+1}$$

and this network is usually called the Omega network. The component $\mathcal{S}_{n+1}; \text{two}^n \text{ sw}$ is known as a shuffle-exchange stage.

The relation Ω is not in general its own converse:

$$\Omega_n \neq \Omega_n^{-1}$$

This can easily be checked by enumerating all 16 permutations realised by $\Omega_2 = \text{ilv sw}; \text{two sw}$. Setting the lower switch in the *ilv* part, and the upper switch in the *two* part to *ex*, while leaving the other switches as *id*₁, gives the permutation that relates [0, 1, 2, 3] to [2, 1, 3, 0]. (Sequences are numbered from bottom to top in the diagrams.) There is no way to set the switches to realise the converse of this permutation.

We can, however, find a permutation, ρ , for which $\rho_n; \Omega_n = \Omega_n^{-1}; \rho_n$. We know from the recursive decompositions of the butterfly that

$$\Omega_{n+1} = \text{ilv } \Omega_n; \text{two}^n \text{ sw}$$

so it is tempting to choose $\rho_0 = \text{id}_0$ and $\rho_{n+1} = U; \text{ilv } \rho_n$ where the U is an unknown that we will fill in later. Then

$$\begin{aligned} \rho_{n+1}; \Omega_{n+1} &= U; \text{ilv } \rho_n; \text{ilv } \Omega_n; \text{two}^n \text{ sw} \\ &= \{ \text{inductive hypothesis} \} \\ &\quad U; \text{ilv } \Omega_n^{-1}; \text{ilv } \rho_n; \text{two}^n \text{ sw} \\ &= \{ \text{ilv } \rho_n; \text{two}^n \text{ sw} = \text{two}^n \text{ sw}; \text{ilv } \rho_n \} \\ &\quad U; \text{ilv } \Omega_n^{-1}; \text{two}^n \text{ sw}; \text{ilv } \rho_n \end{aligned}$$

Now we can see that we would like $U; \text{ilv } \Omega_n^{-1}; \text{two}^n \text{ sw}$ to be equal to $\text{two}^n \text{ sw}; \text{ilv } \rho_n; U$. Luckily, the perfect shuffle satisfies this equation and we get

$$\rho_n; \Omega_n = \Omega_n^{-1}; \rho_n$$

where

$$\begin{aligned} \rho_0 &= \text{id}_0 \\ \rho_{n+1} &= \mathcal{S}_{n+1}; \text{ilv } \rho_n \end{aligned}$$

This ρ permutation is the ubiquitous bit-reversal permutation. If you view the indices into the sequence as binary numbers, and view those numbers in turn as sequences of bits, then ρ moves an element at address i (a sequence of bits) to address $\text{rev}(i)$, the reverse of that sequence of bits. We have not needed to think in these terms. Perhaps a

more useful way to think about it is as a permutation that swaps twos and ilvs. If $\rho_k ; R_k = R_k ; \rho_k$ then we can show by induction that

$$\rho_{i+j+k} ; \text{ilv}^i \text{two}^j R_k = \text{two}^i \text{ilv}^j R_k ; \rho_{i+j+k}$$

Then, since the perfect shuffle is a butterfly of \mathcal{S}_2 components and $\rho_2 ; \mathcal{S}_2 = \mathcal{S}_2 ; \rho_2$, we can use ρ to ‘reverse’ a perfect shuffle

$$\rho_n ; \mathcal{S}_n = \mathcal{S}_n^{-1} ; \rho_n$$

The permutation ρ is its own converse. We show by induction that $\rho_n^2 = id_n$. The base case is $\rho_0^2 = id_0^2 = id_0$. For the step:

$$\begin{aligned} \rho_{n+1}^2 &= \rho_{n+1} ; \mathcal{S}_{n+1} ; \text{ilv} \rho_n \\ &= \{ \rho_n ; \mathcal{S}_n = \mathcal{S}_n^{-1} ; \rho_n \} \\ &\quad \mathcal{S}_{n+1}^{-1} ; \rho_{n+1} ; \text{ilv} \rho_n \\ &= \{ \text{definition of } \rho \} \\ &\quad \mathcal{S}_{n+1}^{-1} ; \mathcal{S}_{n+1} ; \text{ilv} \rho_n ; \text{ilv} \rho_n \\ &= \text{ilv} \rho_n ; \text{ilv} \rho_n \\ &= \text{ilv} \rho_n^2 \\ &= \{ \text{inductive hypothesis} \} \\ &\quad \text{ilv} id_n \\ &= id_{n+1} \end{aligned}$$

This means that the network $\rho_n ; \Omega_n$ is its own converse, since it is the same as $\Omega_n^{-1} ; \rho_n$, which is $\Omega_n^{-1} ; \rho_n^{-1}$.

Because Ω_n is not its own converse, it cannot realise all permutations on sequences of length 2^n . In fact, Ω_n simply has too few switches. It has n columns, each with 2^{n-1} switches. So at most $2^{n2^{n-1}}$ permutations are possible. In fact each new switch setting gives a different permutation, so Ω_n achieves exactly $2^{n2^{n-1}}$ permutations, but that is rather less than the $2^n!$ possible permutations. For example, Ω_3 achieves $2^{12} = 4096$ permutations, but there are $8! = 40320$ possible permutations on 8-element sequences. An example of a permutation that Ω_n cannot achieve is, rather suprisingly, the perfect shuffle \mathcal{S}_n itself.

A network that can realise all permutations is called universal or rearrangeable in the large literature on this topic, see Oruc’s recent survey [13].

6 Building universal networks

Perhaps the best-known universal permutation network is the Benes network [3], which can be described by the following recursion:

$$\begin{aligned} \mathcal{B}_1 &= sw \\ \mathcal{B}_{n+1} &= \text{two}^n sw ; (\mathcal{S}_{n+1})^{-1} ; \text{two} \mathcal{B}_n ; \mathcal{S}_{n+1} ; \text{two}^n sw \\ &= \text{two}^n sw ; \text{ilv} \mathcal{B}_n ; \text{two}^n sw \end{aligned}$$

Figure 6 shows a 16-input Benes network. This is an instance of a more general class of networks known as the Clos networks. Clos’ paper from 1953 seems to have been the seminal work in this field [6]. Both Clos and Benes were working in the context of telephone networks and this influenced the ways in which they thought of and described networks. That legacy seems to live on in the current literature, despite the fact that the emphasis has now shifted to the use of networks in parallel processing.

There are two standard ways to show that the Benes network achieves all permutations. The first is to use a looping algorithm to show how a given permutation can be achieved. One proves universality by giving a routing algorithm. Start with the top-left input. Choose a setting of the switch to which it is attached, a path through the recursive sub-networks and setting of one of the final column of switches so that this input is connected to the correct output. Then

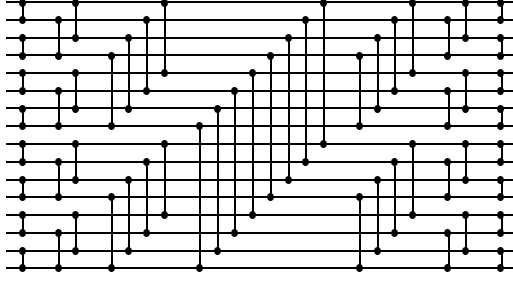


Figure 6: \mathcal{B}_4

take the second output of that final switch and repeat the procedure, going backwards through the network, to connect that output to the correct input, and so on. This is sometimes known as Waksman's construction [18].

The second method is to appeal to results from group theory [2]. The theorem that proves that the "all permutations" relation can be recursively defined as

$$\begin{aligned} \mathcal{P}_1 &= sw \\ \mathcal{P}_{n+1} &= \text{two}^n sw ; (\mathcal{S}_{n+1})^{-1} ; \text{two } \mathcal{P}_n ; \mathcal{S}_{n+1} ; \text{two}^n sw \\ &= \text{two}^n sw ; \text{ilv } \mathcal{P}_n ; \text{two}^n sw \end{aligned}$$

is now called the Slepian-Duguid theorem [12]. It is named after Slepian, who never published anything, and Duguid, whose work is only recorded in a technical report, but who was carefully acknowledged by Benes [3]. It follows that the Benes network achieves all permutations. This non-constructive method is much more appealing, though of course it does not result in a routing algorithm.

Looking at the diagram of the Benes network in figure 6, one can see the converse of a butterfly on the left, and a butterfly on the right, with one column of overlap. Indeed the network $\Omega_n^{-1} ; \Omega_n$ has the same behaviour as the Benes network, and so realises all permutations. We show by induction that $\Omega_n^{-1} ; \Omega_n = \mathcal{B}_n$, for $n > 0$. The base case is $\Omega_1^{-1} ; \Omega_1 = sw^2 = sw = \mathcal{B}_1$. For the step:

$$\begin{aligned} \Omega_{n+1}^{-1} ; \Omega_{n+1} &= \{ \text{definition of butterfly} \} \\ &\quad \text{two}^n sw ; \text{ilv } \Omega_n^{-1} ; \text{ilv } \Omega_n ; \text{two}^n sw \\ &= \{ \text{ilv } R ; \text{ilv } S = \text{ilv}(R ; S) \} \\ &\quad \text{two}^n sw ; \text{ilv}(\Omega_n^{-1} ; \Omega_n) ; \text{two}^n sw \\ &= \{ \text{inductive hypothesis} \} \\ &\quad \text{two}^n sw ; \text{ilv } \mathcal{B}_n ; \text{two}^n sw \\ &= \{ \text{definition of Benes network} \} \\ &\quad \mathcal{B}_{n+1} \end{aligned}$$

The network $\Omega_n^{-1} ; \Omega_n$ has one more column of switches than the Benes network \mathcal{B}_n , but those switches are redundant and so can be removed.

$$\begin{aligned} \Omega_{n+1}^{-1} ; \Omega_{n+1} &= \text{two } \Omega_n^{-1} ; \text{ilv}^n sw ; \text{ilv}^n sw ; \text{two } \Omega_n \\ &= \text{two } \Omega_n^{-1} ; \text{ilv}^n (sw ; sw) ; \text{two } \Omega_n \\ &= \text{two } \Omega_n^{-1} ; \text{ilv}^n sw ; \text{two } \Omega_n \end{aligned}$$

7 The Puzzle

We have seen that the network $\Omega_n^{-1} ; \Omega_n$ can realise all permutations. A question that has fascinated researchers in interconnection networks is “how many shuffle-exchange stages must be connected in series to achieve all permutations?”. Parker showed in 1980 that $3 \log_2 N$ stages are sufficient, for N inputs [14]. That bound has been gradually reduced; to $3 \log_2 N - 1$ by Wu and Feng [19], to $3 \log_2 N - 3$ by Huang and Tripathi [9], and to $3 \log_2 N - 4$ by Varma and Raghavendra [17]. In his survey [13], Oruc cites a paper by Cam and Fortes that reduces the bound to $2 \log_2 N - 1$ [5], but I have so far been unable to get hold of that paper.

One of the reasons why I have been studying interconnection networks is that I would like to answer a different but related question: “does the series connection of two butterflies of switches, or, equivalently, of two omega networks, realise all permutations?”. Feng and Seo state in their 1994 paper that the routing problem for the Omega + Omega network has been an open question for decades [8]. Their paper presents a very complicated algorithm for routing the Omega + Omega network, which is first transformed into a symmetric network. The complexity of the algorithm is such that I have been unable to verify its correctness. In an earlier paper, Lee presents a method for proving the rearrangeability of networks [12]. She claims, plausibly, that the method applies to non-symmetric networks, like Omega + Omega, but, unfortunately, she applies the method only to the symmetric Omega + converse(Omega) network. So, again, I have not found the answer to my question in her paper. I would like instead to prove that

$$\Omega_n^2 = \mathcal{P}_n$$

relying only on the algebraic properties of two, inv , \mathcal{S} and ρ , and on the fact that $\Omega_n^{-1} ; \Omega$ achieves all permutations. I have spent some time trying to do this proof, but it never quite works! It is annoyingly easy to show that $\Omega ; \mathcal{S} ; \Omega$ achieves all permutations. (This is left as an exercise for the reader.) But so far, I have not managed to get rid of that extra shuffle. Note that proving that Ω^2 achieves all permutations would reduce the bound mentioned above to $2 \log_2 N$.

I have written about this puzzle here in the hope that someone will see the Eureka step that I have missed. As an encouragement, let me point out that there are some networks that when composed in series yield all permutations. For example,

$$\begin{aligned} \mathcal{P}_n &= \Omega_n ; \Omega_n^{-1} \\ &= \Omega_n ; \rho_n ; \rho_n^{-1} ; \Omega_n^{-1} \\ &= \Omega_n ; \rho_n ; \Omega_n ; \rho_n \\ &= (\Omega_n ; \rho_n)^2 \end{aligned}$$

My longer-term aim is to demonstrate that this combinator-based style of describing and analysing permutations and networks really works better than more standard methods. I feel that the field could benefit from some new ideas about programming and notation that have been developed by functional programmers and others working on the derivation of programs from specifications. For example, I would like to consider ways of deriving and presenting routing algorithms.

One of my referees asked “what has this got to do with functional programming?”. My answer is that the development of libraries of combinators is a key activity in functional programming circles, and this is just another library. I am also interested in techniques for deriving functional programs to run on networks of processors, and this work is intended to provide the necessary basis. In fact, I have coded much of this in Haskell and viewed (smallish) permutation networks in action, but that is another story.

Acknowledgements

Bertil Svensson, from Computer Engineering at Chalmers, posed the puzzle which has led to many hours of pleasure and frustration. The survey by Oruc, which is available from the University of Maryland Interconnection Network Research Laboratory, was an invaluable source of information. Thanks to Jan Nicklisch, who encouraged me to complete the paper even though I have failed to solve the puzzle.

References

- [1] K.E. Batcher, *Sorting networks and their applications*, in Proc. AFIPS Spring Joint Computing Conference, Vol. 32, April 1969.
- [2] V.E. Benes, *Proving the rearrangeability of connecting networks by group calculations*, Bell Syst. Tech. J., vol. 54, pp. 421-434, Feb. 1975.
- [3] V.E. Benes, *On rearrangeable three-stage connecting networks*, Bell Syst. Tech. J., vol. pp. 1481-1493, 1962.
- [4] G. Bilardi, *Merging and sorting networks with the topology of the omega network*, IEEE Transactions on Computers 38(10) pp.1396-1403, 1989.
- [5] H. Cam and J.A.B. Fortes, *Rearrangeability of shuffle-exchange networks*, in Proc. IEEE Symp. on Frontiers of Massively Parallel Computing, pp. 303-314, 1990.
- [6] C. Clos, *A study of non-blocking switching networks*, Bell Syst. Tech. J., vo. 32, pp. 406-424, 1953.
- [7] J.W. Cooley and J.W. Tukey, *An algorithm for the machine computation of complex Fourier series*, Mathematics of Computation, 19, pp. 297-301, 1965.
- [8] T. Feng and S.-W. Seo, *A New Routing Algorithm for a Class of Rearrangeable Networks*, IEEE Transactions on Computers, 43(11) pp. 1270-1280, Nov. 1994.
- [9] S.T. Huang and S.K. Tripathi, *Finite state model and compatibility theory: mew analysis tools for permutation networks*, IEEE Transactions on Computers, vol. C-35, pp. 591-601, 1986.
- [10] G. Jones, *A fast flutter by the Fourier transform*, in G. Birtwistle (ed.), Proc. 4th Banff Workshop on Higher Order, Springer Workshops in Computing, 1991.
- [11] G. Jones and M. Sheeran, *The study of butterflies*, in G. Birtwistle (ed.), Proc. 4th Banff Workshop on Higher Order, Springer Workshops in Computing, 1991.
- [12] K.Y. Lee, *On the Rearrangeability of $2(\log_2 N) - 1$ Stage Permutation Networks*, IEEE Transactions on Computers, vol. C-34, No. 5, pp. 412-425, May 1985.
- [13] Y. Oruc, *Multiple Tracks of Research on Interconnection Networks*, Technical Report INRL-95-04, The University of Maryland Interconnection Network Research Laboratory (UMINRL), August 1995.
- [14] D.S. Parker, *Notes on shuffle/exchange-type switching networks*, IEEE Transactions on Computers, vol. C-29, pp. 213-222, 1980.
- [15] M. Sheeran, *Describing hardware algorithms in Ruby*, in Declarative Systems (David, Boute and Shriver eds.), Proc. IFIP TC 10/WG10.1 Workshop on Concepts and Characteristics of Declarative Systems, Budapest, 1988, North-Holland 1990.
- [16] H.S. Stone, *Parallel processing with the perfect shuffle*, IEEE Transactions on Computers, vol. C-20, pp. 153-161, Feb. 1971.
- [17] A. Varma and C.S. Raghavendra, *Rearrangeability of multistage shuffle/exchange networks*, IEEE Transactions on Communications, pp. 1138-1147, October 1988.
- [18] A. Waksman, *A Permutation Network*, J. ACM, vol. 15, pp. 159-163, Jan. 1968.
- [19] C.L. Wu and T. Feng, *The universality of the shuffle-exchange networks*, IEEE Transactions on Computers, vol. C-30, pp 324-332, 1981.