

## Chapter 1. Introduction

### 1.1 Two-sided Nonblocking Switching Networks

The need of a switching network first came from the need to interconnect pairs of telephones. At first, when there were not that many phones, a direct wire was installed between every two phones. But with the increase in the number of phones, the transmission cost of these wires became overbearing and the notion of switching was born. Every phone in a given locality was then connected to a “switching” center where the wires from these phones were interconnected through a network called *switching networks*. Later, it was reinvented for the parallel computer to interconnect a set of processors with a set of memories. Currently, it is intended for many other applications, data transmission, video rental, conference calls, broadcast, satellite communication . . . . It is safe to say that the need of switching network is expanding fast.

A switching network can either interconnect one group of users, called a 1-sided network, or two groups, called a 2-sided network. While there are applications for the one-sided network and a theory has been developed for it, the dominant applications and theory for switching networks are 2-sided. For many applications, the two sides represent two genuinely different types of entity; so input  $x$  connecting to output  $y$  is not the same as input  $y$  connecting to output  $x$ . Even for the telephone network which seems to interconnect only one group of users, the real networks could still be 2-sided because they might connect customers’ wires to line equipments, or two different sets of customers, or two sets of time slots as in time division switching (here again, input  $i$  connecting to output  $j$  doesn’t imply input  $j$  connecting to output  $i$ ). Note that a 2-sided network can be used as a 1-sided network by putting the same set of entities on both sides, although this is less economical from the switching viewpoint. In this book, we will only deal with 2-sided networks.

In the 2-sided case we assume that the network has a set of input terminals and a set of output terminals, while the former generate requests to be connected to the latter through the network. Theoretically, an input terminal can request to be connected to any output terminal, just as one phone can call any

other phone. Therefore the network must provide access from any input terminal to every output terminal. Furthermore, once a connection is established, it could last for a period of time, while other input terminals may generate their own requests during this period. What a switching network does is to simultaneously connect these requests, the pattern constantly changing by some terminals hanging up and others making new requests. For economic reason, the network usually doesn't provide a dedicated path from an input terminal to an output terminal, but provides a common pool of connection links. Thus the connection of one request may block that of another. It was an amazing achievement that Clos (1953) first showed that through some clever design supported by some mathematical principle, there exist nonblocking networks with significantly less hardware than a network with dedicated lines.

For telephone networks, perhaps "nonblockingness" is too high a goal. First of all, not all customers are calling simultaneously, so even a blocking network can be satisfactory for most practical purposes. Secondly, when a call is blocked, the customer usually accepts the situation with grace and makes a second attempt at a later time. It is only frequent blocking that may lead to a complaint, and perhaps, a loss of revenue. However, many other applications have a lower tolerance for blockage, and the cost of blockage can be higher. For example, when a customer moves to a nearby place and requests the same phone number, the network interconnecting the customer's wires with line equipments (phone numbers) is not expected to reject the request due to blockage. Military or high-security networks are usually nonblocking. Blockage in a data-network can lead to a serious problem if the data happens to be important. Customers of a video-rental network would not wait beyond the starting time of the program. On the other hand, the cost of hardware has been dropping fast, which makes nonblocking networks more and more a realistic alternative, and even a necessity sometimes.

While we recognize that most switching networks in the world are still the blocking kind, and there is a very interesting theory developed for it, this book will deal with nonblocking, or almost nonblocking networks only because the theory of nonblocking networks is very different from that of blocking networks. By doing so, we will miss many important topics like packet switching, queueing networks, buffers, traffic balancing, blocking probability, etc. However, we think that setting up a framework for nonblocking networks to accommodate their many existing beautiful results is itself a worthwhile task.

Not many books have been published on the theory of switching networks since the Beneš (1965) classic. Varma-Raghavendra (1994) edited a collection

of benchmark papers with a detailed introduction before every topic. Hui's book (1990) covers not only switching, but also traffic theory. It does not focus on the mathematical theory as our book intends to. Li's book (2001) emphasizes on the engineering and application side of switching network, but touches only the most fundamental part of mathematical theory.

## 1.2 Networks

The basic components of a switching network are crossbar switches, or just *crossbars* (occasionally other switches are used and will be referred to as *switches*), and links which connect crossbars. A crossbar with  $n$  inlets and  $m$  outlets, denoted by  $X_{nm}$ , is said of size  $n \times m$ . Inlets (outlets) on the same crossbar are called *co-inlets* (*co-outlets*). Any matching (one-to-one mapping) between the inlets and the outlets of a crossbar is considered routable, i.e., a crossbar is nonblocking. Some crossbars are connected to the outside world. For a 2-sided network, one set of such crossbars will be called *input crossbars* and another set *output crossbars*. The links on an input (output) crossbar linking to outside world are called *inputs* (*outputs*) of the network, and often drawn by open-ended lines. They are also referred to as *external links*, while other links are *internal links*. An  $(N, M)$ -network has  $N$  inputs and  $M$  outputs, which will be called an  *$N$ -network* if  $M = N$ . Although a request is originally generated by a pair of input-output, it can be treated as if generated by a pair of input-output crossbars since the crossbar is nonblocking. A request is connected by a path in the network, while two connections do not block each other if their paths are link-disjoint.

We shall warn the reader that when the switching network is intended to be used as a computer network such as a processor-memory network, then it is customary to assume that the input and output crossbars have no external links. So the numbers of inputs and outputs are simply the numbers of input and output crossbars, respectively. We would not attempt to use different names to differentiate networks with or without external links, since the mathematical theory for them is usually the same. We will call the reader's attention whenever a difference is relevant, and draw the input and output crossbars in circles when no external links are intended, as versus the usual squares for crossbars.

In an  $s$ -stage network, the crossbars are lined up into  $s$  columns, each called a *stage*. Sometimes  $s$  is not specified and the network is called a *multistage interconnection network* (MIN). Crossbars in the same stage have the same

size. Links exist only between crossbars in adjacent stages. Links between a stage- $i$  crossbar and a stage- $(i+1)$  crossbar connects an outlet of the former to an inlet of the latter. Often, only the information of which stage- $i$  crossbars are connected to which stage- $(i+1)$  crossbars are needed. Then the linking pattern between stage  $i$  and stage  $i+1$  can be represented by a bipartite graph  $L_i$  with the crossbars as vertices. Crossbars in the first (last) stage are the input (output) crossbars and its inlets (outlets) are the input (output) terminals, sometimes just called inputs (outputs) of the network connected to external lines. The notation for an  $s$ -stage network is that stage  $i$  has  $r_i$  crossbars of size  $n_i \times m_i$ . Necessarily,  $r_i m_i = r_{i+1} n_{i+1}$  for  $i = 1, \dots, s-1$ . Figure 1.2.1 shows a 3-stage network with 8 inputs and 6 outputs where  $r_1 = r_2 = 4$ ,  $r_3 = 2$ ,  $n_1 = 2$ ,  $m_1 = 3$ ,  $n_2 = 3$ ,  $m_2 = 1$ ,  $n_3 = 2$ ,  $m_3 = 3$ , and a crossbar is represented by a square.

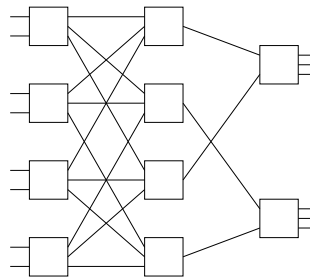


Figure 1.2.1: A 3-stage network

Two networks are called equivalent if one can be obtained from the other through permuting crossbars in the same stage.

Let  $X_{ab} \times X_{cd}$  denote the 2-stage network where stage-1 consists of  $c$  copies of  $X_{ab}$ , stage-2 consists of  $b$  copies of  $X_{cd}$ , and the linking pattern between the two stages is a complete bipartite graph. Let  $[X_{ab}, X_{cd}, X_{be}]$  denote the 3-stage network where stage-1 consists of  $c$  copies of  $X_{ab}$ , stage-2 consists of  $b$  copies of  $X_{cd}$ , stage-3 consists of  $d$  copies of  $X_{be}$ , and the linking patterns between adjacent stages are complete bipartite graphs. These notations, due to Cantor (1971), are often used to easily describe the structure of a MIN. Figure 1.2.2 illustrates these networks. The 3-stage network  $[X_{ab}, X_{cd}, X_{be}]$  was first proposed by Clos (1953) and is now known as the 3-stage Clos network. The traditional notation for 3-stage Clos network is  $[X_{n_1 m_1}, X_{r_1 r_2}, X_{m_2 n_2}]$ , where  $N_1 = n_1 r_1$  is the number of inputs and  $N_2 = m_2 r_2$  is the number of outputs.

Note that the use of  $m, n_2, r_2, N_2$  is not consistent with the notation for MIN.

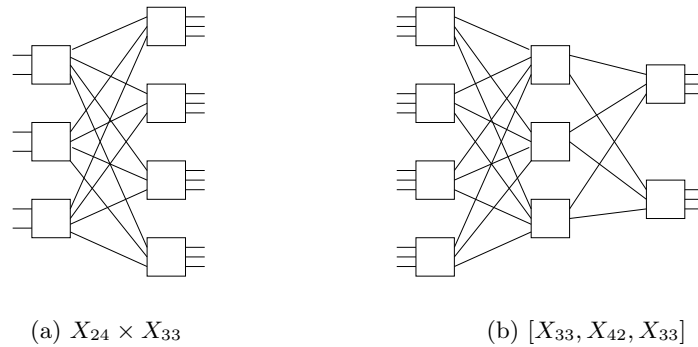


Figure 1.2.2: Two constructions of networks

The notation introduced in the last paragraph can be extended by replacing the crossbars by networks. For example, the well known  $(2n - 1)$ -stage *Beneš network*  $B_n$  is defined recursively by  $B_2 = [X_{22}, X_{22}, X_{22}]$  and  $B_n = [X_{22}, B_{n-1}, X_{22}]$ , where the number of the first(last)  $X_{22}$  equals the number of inputs(outputs) in  $B_{n-1}$ , which is  $2^{n-1}$ . Another network operator often used is *concatenation*. Let  $\nu$  denote an  $s$ -stage network and  $\nu'$  an  $s'$ -stage network such that the last stage of  $\nu$  is identical to the first stage of  $\nu'$ . Then  $\nu \circ \nu'$  denote the  $(s + s' - 1)$ -stage network obtained by identifying the last stage of  $\nu$  with the first stage of  $\nu'$ . Figure 1.2.3 illustrates the concatenation of the two networks in Fig. 1.2.2.

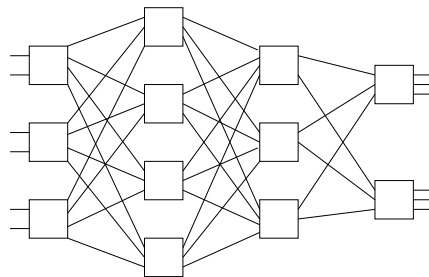


Figure 1.2.3: Concatenation of the two networks in Fig. 1.2.2.

A  $d$ -nary network (MIN) is simply a network (a MIN) using only crossbars of size  $d \times d$ . In a  $d$ -nary MIN of size  $N$ , a power of  $d$ , it is customary to use the

notation  $n = \log_d N$  (but if input and output switches have no external link, then  $n = \log_d N + 1$ ), which we adopt in this book. Note that in a  $d$ -nary MIN every stage has the same number of crossbars. The term *almost  $d$ -nary* is used if exceptions are allowed for the input and output stages. The  *$d$ -nary Beneš network*, denoted by  $B_n^d$ , is similarly defined as the Beneš network except using  $d \times d$  crossbars.

Figure 1.2.4 illustrates two  $d$ -nary MINs.

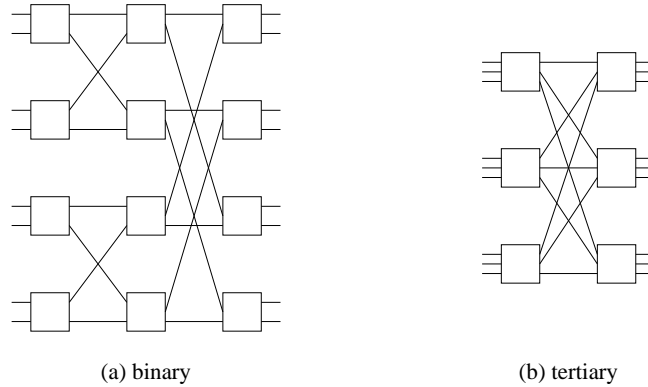
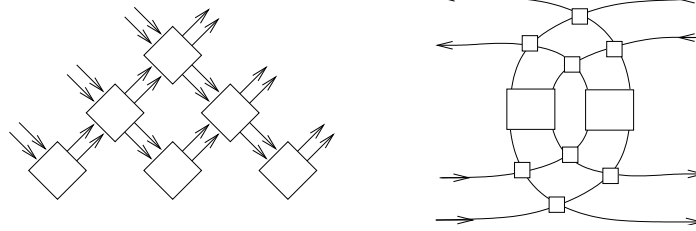


Figure 1.2.4: Two  $d$ -nary MINs

The notion of a *staged* network was first proposed by Halpenny (1990). The only requirement is that the network can be laid out in the plane with input terminals on one side, output terminals on another, and a link, represented by a straight line, does not bypass a crossbar. A network is *planar* if it can be laid out in the plane such that links do not cross each other. Figure 1.2.5 illustrates these networks. The relevance of the planar network to the optical switch was first called into attention by Spanke and Beneš (1987).

### 1.3 Traffic and Nonblockingness

When we say that a network is nonblocking, it is in reference to a certain type of traffic. We can classify traffic according to whether the requests come one by one, like phone calls, or they are scheduled into sessions, and all requests in a given session, called a *frame*, are given simultaneously for routing, like video-rental or satellite communication. We call the former type *dynamic* and the latter *scheduled*. For convenience, we also refer to all traffic currently carried



(a) A staged (also planar) network with six input (output) terminals

(b) A planar network with four input (output) terminals

Figure 1.2.5: A staged network and a planar network

by the network plus the new request in the dynamic traffic as a frame.

The traffic can also be classified as *point-to-point*, like 2-party phone calls, or *broadcast*, which is one to many. If there is a restriction on the maximum number of receivers per request, then broadcast is called *multicast*, or *f-cast*, if that number is specified to be  $f$ . The dynamic multicasting traffic can be further divided into two types according to whether additional receivers can be added after a multicast request is already connected. We will use *open-end traffic* (which allows additions) and *closed-end traffic* (which does not allow) to differentiate the two types.

Finally, we call the traffic *classic* if each link (terminal) carries (generates) one request and *multirate*, if each request is associated with a load, while each link (terminal) has a capacity and can carry (generate) as many requests as desirable as long as the total load does not exceed the capacity. Such kind of traffic can be generated by networks which integrate various types of requests like phone calls, data transmission, video signals with different bandwidth requirements (loads). If a network is designed for a special type of traffic, we can also transfer the classification of traffic to networks. Note that for the classic traffic, capacity = load = unity.

Define the *states* of a network as the set of all possible routings of all legitimate frames, *legitimate* means the load generated by each input and output terminal does not exceed its capacity. The empty state is simply the state of carrying no traffic. The states can be partially ordered by containment where  $s$  containing  $s'$  means  $s$  can be obtained from  $s'$  by adding the routings of more requests (but still legitimate). A set of requests is routable if there exists a set of link-disjoint paths connecting the requests. A state is *blocking* if there exists

a legitimate new request not routable in the current state; and is *nonblocking* otherwise.

Traditionally, there are different levels of nonblockingness: strictly, wide-sense and rearrangeable. A network is *strictly nonblocking* (SNB) if it has no blocking state. Beneš (1965) proposed the concept of *wide-sense nonblocking* (WSNB). A network is WSNB under a routing algorithm  $A$  if there exists a closure  $CL(S)$  of states called *safe states*, closed under the operation of adding or deleting a connection, such that

- (i) a safe state is a nonblocking state,
- (ii) the empty state is a safe state,
- (iii) any legitimate request generated in a safe state can be routed according to  $A$  into another safe state.

Sometimes  $A$  is given as a class. What it means is that the network is WSNB for any algorithm in  $A$ . Other times  $A$  is not mentioned, what it means is that there exists an algorithm under which the network is WSNB. With the same logic, a network is not WSNB means under no algorithm is it WSNB.

Smyth (1988) gave a method to find  $CL(S)$  under a routing algorithm  $A$ . First delete all blocking states. Then iteratively delete all states which may be forced to go to a deleted state under  $A$  by a deletion or an addition. The undeleted states, if any, constitute  $CL(S)$ . If  $CL(S)$  is empty, then the network is not WSNB under  $A$ .

Since a routing is done by setting (the internal connection of) the switches, a state can also be defined as a setting of switches. The difference between SNB and WSNB is actually more subtle than it appears since what are the possible settings of a switch is often hidden. Take the crossbar  $X_{nm}$ , which is usually considered SNB and represented as a grid of  $n$  rows and  $m$  columns. At each row-column intersection, there is a crosspoint with two states “straight” and “bend”, which controls the routing (see Fig. 1.3.1).

If each crosspoint is free to go straight or bend, then we claim that a crossbar is not SNB. Figure 1.3.2 shows an  $X_{33}$  where the (2, 2) path blocks the (1, 3) and the (3, 1) path.

Of course, by not allowing a path to make right turns, we can avoid the above situation and show nonblockingness. However, then a crossbar is merely WSNB, because there exist blocking states; it is only that we don't get into them by directing traffic cleverly. On the other hand we can wire a crosspoint

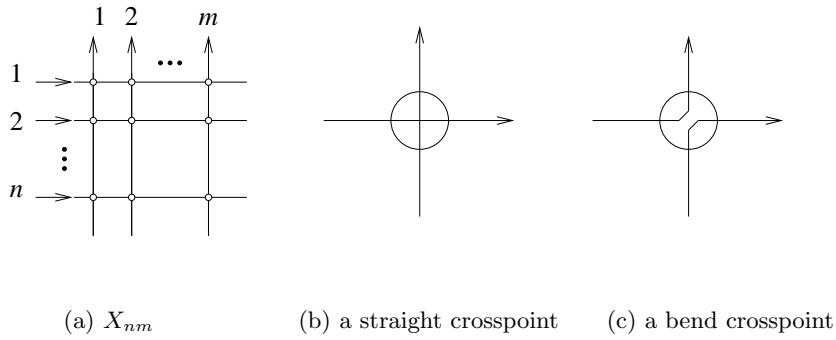


Figure 1.3.1: Crossbar and crosspoint

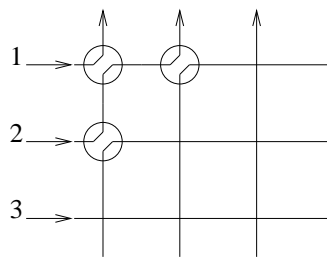


Figure 1.3.2:  $X_{33}$  is not SNB

differently to disallow the right turn (see Fig. 1.3.3), then a crossbar remains SNB.

Thus we see that the difference between SNB and WSNB hinges on more details about the hardware, which is unfortunate because we now have to provide the details of hardware, for example, which type of crosspoints, before discussing the nonblocking property of a network. (This is especially awkward for a mathematician who usually doesn't know the engineering details.) Of course, if we know a component can be implemented to be SNB, then instead of giving the engineering details, we can simply specify a SNB component. Another alternative, which seems to be the common practice as far as the crossbar is concerned, is to assume a component is SNB if it can be implemented that way.

Perrier–Prucnal (1989) proposed a notion of nonblockingness for point-

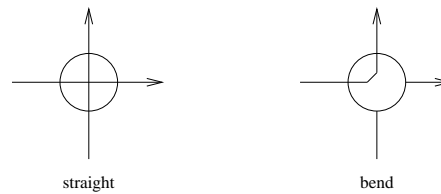


Figure 1.3.3: A differently wired crosspoints

to-point traffic which also blurs the difference between SNB and WSNB. A network is a *standard path network* (also called *fixed routing*) if it provides a standard path for each pair of input-output terminals to route their request, and the standard paths in any legitimate frame are all disjoint. Note that this doesn't mean that the standard paths are all disjoint, since it is possible that the standard paths of  $(1, 3)$  and  $(2, 3)$  overlap, but these two requests cannot coexist in a legitimate frame. A standard path network is clearly WSNB where the set of nonblocking states are those states which use standard paths. On the other hand, a standard path network can often be hard-wired into a SNB network by physically handicapping all paths except the standard ones. For example, the crossbar as shown in Fig. 1.3.1 is a standard path network where the standard path for the  $(i, j)$  request is to turn at the  $(i, j)$  intersection. This standard path crossbar can be turned into SNB by hard-wiring the crosspoints as shown in Fig. 1.3.3. Note that a general WSNB network cannot be hard-wired into SNB because the connection paths, unlike the standard paths, are traffic dependent.

For scheduled traffic, a network which can route all legitimate frames is called rearrangeably nonblocking (RNB), or simply *rearrangeable*. Such a network can also route dynamic traffic by rerouting some existing connections to make room for a newly blocked request, hence the name "rearrangeable". Note that by rerouting all existing calls, dynamic traffic is turned into scheduled traffic. A rearrangeable network with point-to-point traffic is often called a *connector*. We will also call one with multicast traffic a *multicast connector*, which sounds more specific than the term *generalizer* used in the literature.

A related notion first proposed by Ackroyd (1979) is the *repackable* network. For dynamic traffic, the rerouting is done not when a new request is blocked, but when an existing connection is deleted, with the purpose to balance the load carried by the intermediate switches. The rerouting is usually

limited to one connection. A network is *repackable* if it is nonblocking by following the rerouting rule. Conceptually, repackability is also similar to WSNB except the traffic being directed is the existing connections rather than the new arrivals.

#### 1.4 Objectives

Traditionally, nonblocking networks are designed to minimize the number of crosspoints since that is the most expensive part in a network. Even though in many new technologies, the cost of crosspoints is no longer a dominant issue, the number of crosspoints still remains a popular measure of network performance since it serves a figure of merit for some other important features, like the physical size and control complexity of a network, or the number of ports required when a network is partitioned into chips or boards. We define the *cost* of a network as the number of crosspoints in it.

In a multistage network, the number of stages represents the length of a path. A shorter path of course means a faster connection, less likely to encounter a faulty element or to have the signal being transmitted weakened, and to consume less power. Under the current technology, only networks with a very small number of stages are practical.

Routing algorithms tend to be overlooked in SNB networks since a free path is guaranteed for any request even no routing algorithm is given. However, an efficient routing algorithm still serves to speed up the connections. Routing algorithms of course play a more fundamental role in WSNB and rearrangeable networks since the very claim of nonblockingness depends on the existence of a routing algorithm. Such routing algorithms are evaluated by their time complexity. Any algorithm requiring more than linear time would be too complicated for real-time use. Two remedies are to use parallel processors to route requests and to use self-routing algorithms, which provide standard paths in a blocking network. The trade-off is a higher cost for the former remedy, and giving up nonblocking in the latter.

To partition a switching network and to lay it out into boards or chips, the criteria are the modularity of the network, the number of parts, the number of ports on a part, the number of knock-knees and crossings, and planarity. For optical switches, the crosstalk problem is of particular importance and the planar network is one solution to that problem. Research on these problems are emerging, but will not be covered in this book (except planar networks).

The literature on switching networks shows two different lines of approach

in constructing nonblocking networks with the number of crosspoints as objective. One is to consider networks of practical sizes and to analyze their performances. The other is a theoretical pursue of networks with the minimum orders of complexity. While the latter approach usually produces better theoretical results, one should be warned that some of the complexity results are proved by existence without explicit constructions, some of the constructions may have very large coefficients which are not shown in the orders of magnitude complexity, some of the results can be achieved only with a very large number of stages, or an unbounded number of terminals. For these reasons, we will focus on the practical networks, but provide the theoretical results as a background. In particular, some constructions rely heavily on the constructions of bipartite graphs with certain properties. While these graph constructions are very clever and interesting of their own right, we choose not to delve into them since they are more graph-theoretic than network-theoretic. If we need them in a network, we simply quote their properties from known sources.

### 1.5 Self-routing Networks and Their Extra-stage Version

For computer networks, delays more than polylog time are generally unacceptable. Therefore centralized routing algorithms which usually require  $O(N \log N)$  time are out. Instead, a bunch of  $\log_2 N$ -stage networks with self-routing property have been invented; here, *self-routing*, first proposed by Lawrie (1976) for the Omega network, means that a request can be routed by only knowing its input and output, and nothing about other requests. These networks are usually recognized as the banyan-type by the following features.

- (i) The network is an  $n$ -stage binary network ( $n = \log_2 N$ ).
- (ii) Each input has a unique path to each output.

Agrawal (1983) called attention to another common property of these self-routing networks. Let  $u$  and  $v$  be two stage- $k$  crossbars, and let  $V_j(u)$  and  $V_j(v)$  denote the two sets of stage- $(k + j)$  crossbars  $u$  and  $v$  can reach. Then the network is said to have the  $(k, j)$  *buddy property* if either  $V_j(u) \cap V_j(v) = \emptyset$  or  $V_j(u) = V_j(v)$  for all  $u, v$ . If a network has the  $(k, j)$  buddy property for all  $k$  and  $j$ , it is a *buddy network*.

Some well-known self-routing networks which have the buddy property, are shown in Fig. 1.5.1.

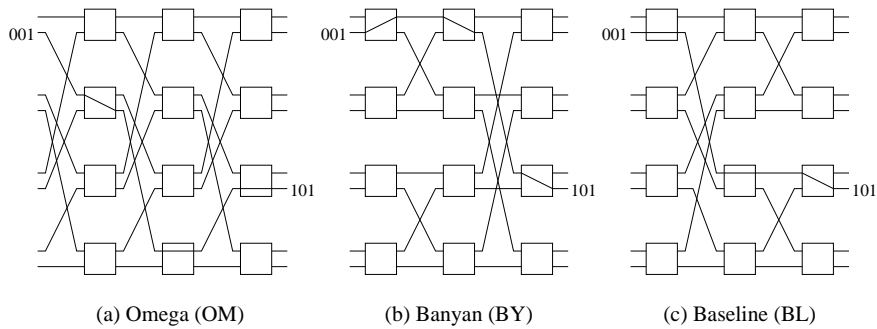


Figure 1.5.1: Some self-routing networks

The above class of binary networks can be extended to  $d$ -nary by replacing  $(i)$  with  $(i')$   $N = M = d^n$ . The network consists of  $n$  stages of  $X_{dd}$ 's.

Note that an  $s$ -stage  $d$ -nary network is completely determined by the linking patterns between stages. Two such networks are called *equivalent* if the crossbars can be labeled such that the linking functions of the two networks become identical. Parker (1980) established the equivalence of some  $n$ -stage binary networks. Wu–Feng (1980) systematically studied a set of popular  $n$ -stage binary networks and proved them to be all equivalent. Agrawal (1983) proposed that the buddy property characterizes this equivalence class, but Bermond–Fourneau–Jean-Marie (1987) gave a counter-example. Instead, they gave the following characterization.

A binary  $n$ -stage network is said to have the  $P(i, j)$  property if the sub-network from stage  $i$  to stage  $j$ , viewed as a graph, has exactly  $2^{n-1-(j-i)}$  components.

A binary  $n$ -stage network is said to have the  $P(\cdot, \cdot)$  property if it has the  $P(i, j)$  property for all  $1 \leq i \leq j \leq n$ .

**Theorem 1.5.1.** *A banyan-type network is equivalent to the Omega network if and only if it has the  $P(\cdot, \cdot)$  property.*

The proof of Theorem 1.5.1 is withheld as we are going to prove a more general result, which does not restrict the network to  $n$  stages.

Kruskal–Snir (1986) defined the *bidelta* network  $BD_n$  as one which can be recursively constructed in both directions, i.e.,  $BD_1 = X_{dd}$ ,  $BD_n = X_{dd} \times BD_{n-1} = BD_{n-1} \times X_{dd}$ . They independently proved the following result which can now be stated as a corollary of the  $d$ -nary version of Theorem 1.5.1.

**Corollary 1.5.2.** *All  $n$ -stage bidelta networks are equivalent.*

Let  $W^{-1}$  denote the inverse network of  $W$ , i.e., reversing the order of the stages. An  $(n + 1)$ -stage buddy network was first proposed by Siegal-Smith (1978) for increasing the connection power and for fault tolerance. Shyy-Lea (1991) considered adding  $k$  extra stages to  $BY^{-1}$  and specified that the extra  $k$  stages should be identical to the mirror image of the first  $k$  stages. Represent a  $k$ -extra-stage buddy network by  $B(k, n)$  or  $B(k, N)$ . The specified way of addition has the advantage that  $BY^{-1}(k, n)$  can be sequentially decomposed  $j$  times,  $1 \leq j \leq k$ , namely the subnetwork of  $BY^{-1}(k, n)$  from stage  $j + 1$  to stage  $n + k - j$  decomposed into  $2^j$   $BY^{-1}(k - j, n - j)$  such that each input (output) switch of the  $BY^{-1}(k, n)$  has a unique path to each  $BY^{-1}(k - j, n - j)$  (see Fig. 1.5.2 in which the external terminals are not drawn). Denote this way of adding extra stages by  $F^{-1}$ . Hwang-Liaw-Yeh (1998) observed that there are three other natural ways of addition.

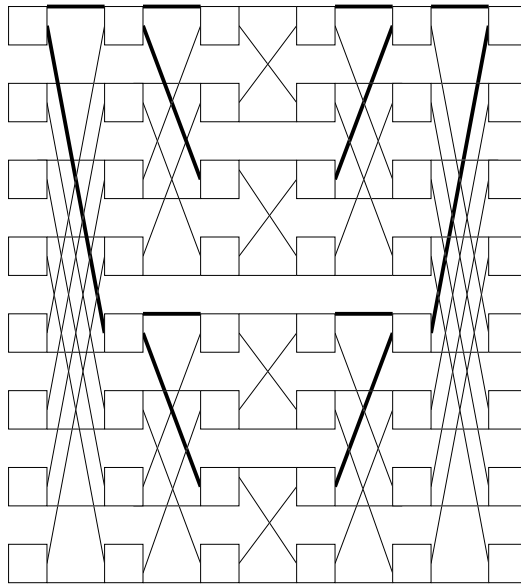


Figure 1.5.2: Decomposition of  $BY^{-1}(2, 4)$

- (i)  $F$ : The extra  $k$  stages are identical to the first  $k$  stages.
- (ii)  $L$ : The extra  $k$  stages are identical to the last  $k$  stages.

- (iii)  $L^{-1}$ : The extra  $k$  stages are identical to the mirror image of the last  $k$  stages.

The various ways of addition result in different networks with different connection capabilities in general. Extra-stage/Omega networks are known as *shuffle exchange* (SE) networks. Hwang–Liaw–Yeh determined the equivalence classes among the  $k$ -extra-stage networks  $SE(k)$ ,  $SE^{-1}(k)$ ,  $BY(k)$ ,  $BY^{-1}(k)$ ,  $BL(k)$ ,  $BL^{-1}(k)$  for all  $k$  and under each of  $F$ ,  $F^{-1}$ ,  $L$ ,  $L^{-1}$ .

Chang, Hwang and Tong (1999) proposed the class of bit permutation networks. Label the crossbars in a stage by distinct  $d$ -nary  $(n - 1)$ -sequence. A bit- $i$  group (or simply  $i$ -group) consists of the  $d$  crossbars whose labels differ only in bit  $i$  (there are  $d^{n-2}$  bit- $i$  groups for each  $i$ ). Let  $G_i$  denote the bipartite graph connecting the crossbars of stage  $i$  and stage  $i + 1$ . An  $s$ -stage network is a bit permutation network if every  $G_i$ ,  $1 \leq i \leq s - 1$ , corresponds to a mapping from bit- $u_i$  groups of stage  $i$  to bit- $v_{i+1}$  groups of stage  $i + 1$  for some  $u_i, v_{i+1}$ . They proved that a bit permutation network is equivalent to one whose  $G_i$  corresponds to a mapping with  $v_{i+1} = u_i$  for all  $i$ . Such a network can be characterized by the vector  $(u_1, \dots, u_{s-1})$ . Further, they showed that two characterizing vectors are equivalent if one can be obtained from the other through permuting the  $n - 1$  bits.

Recently, Li (2001) proposed to view  $G_i$  as a bipartite graph between the outputs of stage  $i$  and the inputs of stage  $i + 1$ . Label the outputs (inputs) by distinct  $d$ -nary  $n$ -sequences. Then  $G_i$  gives a bijection from the  $d^n$  outputs to the  $d^n$  inputs, and hence can be treated as a permutation. Such a permutation is called a *bit permutation* if it can be characterized by a permutation  $\Pi_i$  of the  $n$  bits. We will call a network an *edge bit permutation network* if each  $G_i$  corresponds to a  $\Pi_i$ . Since a permutation  $\Pi$  mapping bit  $n$  to bit  $n$  will cause all outputs of a stage  $i$  crossbar connected to the same stage- $(i + 1)$  crossbar (highly undesirable), we assume  $\Pi(n) \neq n$  throughout this section.

Hwang (2003) established a framework to properly place the above-mentioned four classes of networks: buddy ( $B$ ),  $P(\cdot, \cdot)$ , bit permutation ( $BP$ ) and edge bit permutation ( $EBP$ ). Since  $P(\cdot, \cdot)$  is defined only for  $n$ -stage networks, he generalized it to the class of power-of- $d$  networks ( $d^P$ ). An  $s$ -stage network is in this class if for any  $i, j$ ,  $1 \leq i < j \leq s$ , the number of components in  $G_{ij}$  is a power of  $d$ . The intersection of the power-of- $d$  class and the buddy class is called the power-of- $d$  buddy class ( $d^P B$ ).

The permutation in Fig. 1.5.3 (the stages are drawn in horizontal to save space) illustrates a bit permutation (1432) where  $x_1, x_2, x_3, x_4$  are mapped to  $x_2, x_3, x_4, x_1$ .

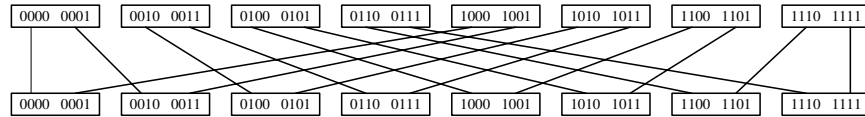


Figure 1.5.3: a bit permutation

We now show that a bit permutation  $T_i$  defines a mapping from  $u$ -groups of stage  $i$  to  $v$ -groups of stage  $i + 1$ . In fact, we can pinpoint  $u$  and  $v$ .

**Lemma 1.5.3.** *Suppose  $G_i$  is represented by the bit permutation  $\Pi_i$ . Then  $G_i$  induces a mapping from  $\Pi_i^{-1}(n)$ -groups to  $\Pi_i(n)$ -groups.*

*Proof.* Note that the label of a crossbar can be obtained from the labels of its  $d$  edges by dropping the last bit. Since  $\Pi^{-1}(n)$  is mapped to  $n$  and get dropped in the crossbar label of stage  $i + 1$ , the  $d$  stage- $i$  crossbars differing only in bit  $\Pi^{-1}(n)$ , i.e., the  $\Pi^{-1}(n)$ -group, are mapped to the same set of stage- $(i + 1)$  crossbars.

On the other hand, the stage- $i$  crossbar containing  $d$  edges whose left endpoints differ only in bit  $n$  is mapped to the  $\Pi_i(n)$ -group of stage  $i + 1$ . Lemma 1.5.3 is proved.  $\square$

For the example in Fig. 1.5.3, the mapping is from  $(\Pi_i^{-1}(4) = 1)$ -groups to  $(\Pi_i(4) = 3)$ -groups.

**Corollary 1.5.4.**  $EBP \subset BP$ .

*Proof.* That  $EBP \subseteq BP$  is trivial. The strict containment is shown by Fig. 1.5.4 which gives a network in  $BP$  ( $G_1$  maps 3-groups to 3-groups,  $G_2$  maps 2-groups to 3-groups), but not in  $EBP$  ( $\Pi_2$  is not a bit permutation).  $\square$

Next we give a simple proof of vector-characterization of an  $EBP$  network.

**Lemma 1.5.5.** *Suppose  $G_i$  corresponds to a bit permutation  $\Pi_i$  which maps  $\Pi_i^{-1}(n)$ -groups of stage  $i$  to  $\Pi_i(n)$ -groups of stage  $i + 1$ . Suppose we permute the crossbars of stage  $i + 1$  such that the  $j^{\text{th}}$  crossbars of the  $\Pi_i(n)$ -groups are lined up with the  $j^{\text{th}}$  crossbars of the  $\Pi_i^{-1}(n)$ -group,  $j = 0, 1, \dots, d - 1$ . Then the permutation  $\Pi$  (of stage  $i + 1$  crossbars) can be obtained from  $\Pi_i^{-1}$  by dropping  $n$ .*

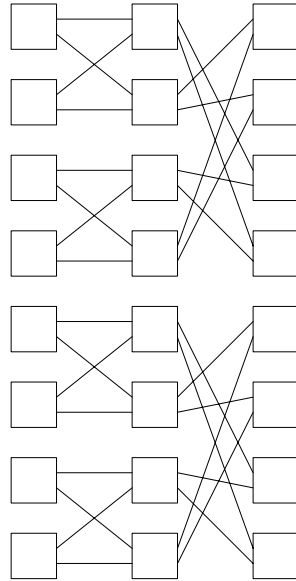


Figure 1.5.4: A BP network

*Proof.* Take a  $\Pi_i^{-1}(n)$ -group of stage  $i$ . Then the  $j^{th}$  crossbars in this group is mapped (lined up) to the  $j^{th}$  crossbars of the corresponding  $\Pi(n)$ -group of stage  $i$  to the crossbars of stage  $i + 1$  under this lined-up operation. Then the only difference between  $\Pi_i$  and  $\Pi'_i$  is that in  $\Pi_i$ ,  $\Pi_i^{-1}(n)$  maps to  $n$  and  $n$  maps to  $\Pi_i(n)$ , while in  $\Pi'_i$ ,  $\Pi_i^{-1}(n)$  maps to  $\Pi_i(n)$ . Therefore  $\Pi'_i$  can be obtained from  $\Pi_i$  by dropping  $n$ . But what we look for is the mapping  $(\Pi'_i)^{-1}$ , namely, to move each stage  $i + 1$  crossbar to the position its stage- $i$  mate occupies. Hence Lemma 1.5.5.  $\square$

In Fig. 1.5.5,  $\Pi_i = (123)$ ,  $G_i$  is a bit permutation from 2-groups to 1-groups. The lining up of the stage- $(i + 1)$  crossbars corresponds to a permutation  $\Pi_i = (12)$  of stage- $(i + 1)$  crossbars. Note that  $\Pi'_i$  can be obtained from  $\Pi_i^{-1} = (132)$  by dropping 3.

**Corollary 1.5.6.** *A group in stage  $i + 1$  remains a group after  $\Pi'_i$ .*

*Proof.* Since  $\Pi'_i$  merely permutes the  $n - 1$  bits of the crossbar labels, a  $u$ -group is transformed to a  $\Pi'_i(n)$ -group.  $\square$

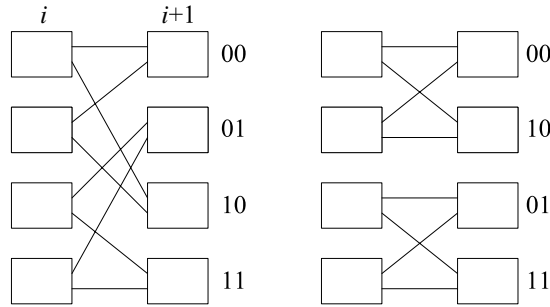


Figure 1.5.5: Lining up stage-(i + 1) crossbars

This leads to a proof of the vector characterization of the *BP* network simpler than the original proof by Chang, Hwang and Tong.

**Theorem 1.5.7.** *Consider an  $s$ -stage bit permutation network. By permuting the crossbars at stage 2, 3,  $\dots$ ,  $s$ , each  $G_i$  can be characterized by mapping  $u'_i$ -groups to  $u_i$ -groups,  $i = 1, \dots, s - 1$ .*

*Proof.* We prove Theorem 1.5.7 induction on  $s$ . Theorem 1.5.7 is trivially true for  $s = 2$  since we can permute the crossbars of stage 2 to line up with their mates in stage 1. Then  $G_1$  becomes mapping  $u_1$ -groups to  $u_1$ -groups. Suppose Theorem 1.5.7 holds for up to  $s - 1$  stage. We prove for  $s$  stages.

Again, permute the crossbars at stage 2 according to  $(\Pi'_1)^{-1}$  to line up with their mates at stage 1. By Corollary 1.5.6, a  $u_2$ -group at stage 2 remains to be a group, say, a  $u'_2$ -group. Thus we may apply induction on the  $(s - 1)$ -stage bit permutation network such that  $G_i$  is characterized by mapping  $u'_i$ -groups to  $u_i$ -groups for  $i = 1, 2, \dots, s - 1$ .  $\square$

**Corollary 1.5.8.**

$$u'_i = (\Pi'_1)^{-1}(\Pi'_2)^{-1} \dots (\Pi'_{i-1})^{-1}(u_i)$$

For the example in Fig. 1.5.3,  $\Pi_i = (1432)$  and the mapping is from 1-groups to 3-groups and  $(\Pi'_i)^{-1} = (123)$  for all  $i = 1, 2, 3$ . Thus  $u'_1 = u_1 = 1$ ,  $u'_2 = (\Pi'_1)^{-1}(1) = 2$ ,  $u'_3 = (\Pi'_1)^{-1}(\Pi'_2)^{-1}(1) = (\Pi'_1)^{-1}(2) = 3$ .

Although we showed  $BP \subset EBP$  in Corollary 1.5.4, the two classes are equivalent.

**Theorem 1.5.9.** *A BP network is equivalent to an EBP network.*

*Proof.* The vector characterization of a  $BP$  network actually means that a  $BP$  network is equivalent to another  $BP$  network whose mapping in  $G_i$  is bit- $u_i$  group for  $i = 1, \dots, s-1$ . Since the mapping bit- $u_i$  group to bit- $u_i$  group corresponds to the edge bit-permutation  $(u_i, n)$ , Theorem 1.5.9 follows.  $\square$

Chang, Hwang and Tong proved

**Theorem 1.5.10.** *Suppose a  $d$ -nary bit permutation network is characterized by the vector  $(u_1, \dots, u_{s-1})$  which contains  $k$  distinct elements. Then the network has  $d^{n-1-k}$  components.*

*Proof.* Let  $i$ ,  $1 \leq i \leq n-1$  be a number not in  $(u_1, \dots, u_{s-1})$ . Then the crossbars on a path never change their  $i^{\text{th}}$  bit, i.e., all crossbars in a component have the same  $i^{\text{th}}$  bit. Since there are  $n-1-k$  such  $i$  not in  $(u_1, \dots, u_{s-1})$ , they generate  $d^{n-1-k}$  different combinations and hence that many components.  $\square$

**Corollary 1.5.11.**  $BP \subseteq d^P$ .

**Theorem 1.5.12.**  $BP \subseteq B$ .

*Proof.* Consider an  $s$ -stage  $BP$  network characterized by  $(u_1, \dots, u_{s-1})$ . Let  $v$  be a crossbar in stage  $i$  which reaches a set  $V_j(v)$  of crossbars in stage  $j$ . Then  $V_j(v)$  consists of crossbars whose labels are same as  $v'_s$  in bits in the set  $I = \{1, \dots, n-1\} \setminus \{u_i, u_{i+1}, \dots, u_{j-1}\}$ . Let  $v'$  be another crossbar in stage  $i$ . If  $v'$  differs from  $v$  in a bit in  $I$ , then clearly,  $V_j(v') \cap V_j(v) = \phi$ ; if not, then  $V_j(v') = V_j(v)$ . Since  $i, j, v, v'$  are arbitrary, the network is in  $B$ .  $\square$

**Theorem 1.5.13.**  $BP \subset d^P B$ .

*Proof.* That  $BP \subseteq d^P B$  follows from Corollary 1.5.11 and Theorem 1.5.12. That the containment is strict follows from Fig. 1.5.6.  $\square$

**Theorem 1.5.14.** *A  $d^P B$  network is equivalent to a  $BP$  network.*

*Proof.* We prove Theorem 1.5.14 by induction on the number of stage. Consider an  $s$ -stage  $d^P B$  network.

- (i)  $s = 2$ . Suppose  $v$  of stage 1 is connected to the set  $V_2(v)$ . Let  $w$  be another crossbar in stage 1 and connected to  $w \in V_2(v)$ . By the buddy property,  $V_2(w) = V_2(v)$ . Since there are  $d-1$  choices of  $v'$ , these  $v'$  together with  $v$  form a  $d \times d$  complete bipartite graph  $K_{d,d}$  with  $V_2(v)$ . Furthermore,  $V_2(v') \cap V_2(v) = \phi$  for any  $v'' \in v \cup \{v'\}$ . Since  $v$  is arbitrary,  $G_{12}$  consists of  $d^{n-1} K_{d,d}$  whose equivalence to a  $BP$  network is clear.

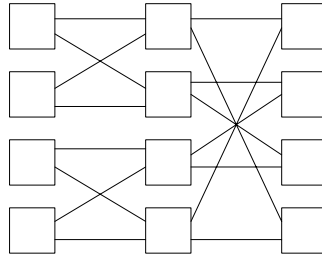


Figure 1.5.6: A  $d^P B$  network which is not a  $BP$ -network

- (ii)  $s = 3$ . By the  $d^P$  property, the network has  $d^{n-k}$  components for some  $1 \leq k \leq n$ . Recall that from (1) the subgraphs  $G_{12}$  and  $G_{13}$  must each consist of  $d^{n-1} K_{d,d}$ . For  $k = 1$ , then no two  $K_{d,d}$  in  $G_1$  can be connected through  $G_2$ . Therefore  $G_{13}$  must consist of  $d^{n-1}$  copies of concatenation of two  $K_{d,d}$ , with the outputs of the former identified with the inputs of the latter (see Fig. 1.5.7).

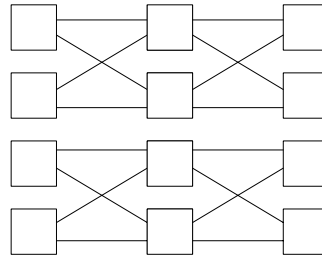


Figure 1.5.7: Concatenation of  $K_{2,2}$

Clearly,  $G_{13}$  is equivalent to a  $BP$  network.

For  $k = 2$ , first suppose  $G_{13}$  is obtained by connecting each  $d$ -set  $D = \{D_1, \dots, D_d\}$ , when each  $D_i$  is a  $K_{d,d}$  in  $G_1$ , into one component in  $G_{13}$ . Note that the connection is done by a  $d$ -set  $D' = \{D'_1, \dots, D'_d\}$  of  $K_{d,d}$  in  $G_2$ . If two crossbars of the same  $D_i$  are connected to a  $D'_j$ , then one member of  $D \setminus D_i$  will not be connected to  $D'_j$ , violating the buddy property. Therefore, the  $d$  crossbars in a  $D_i$  must go to distinct  $D'_j$ , or all  $D'_j$ . Since we can permute the stage-2 crossbars in a  $D$  arbitrarily,

and independently for each  $D$ , the stage-2 crossbars in each  $D$  can be ordered such that the  $k^{th}$  one goes to the  $k^{th}D'$ , which is clearly a  $BP$  network. Figure 1.5.8 illustrates how to permute.

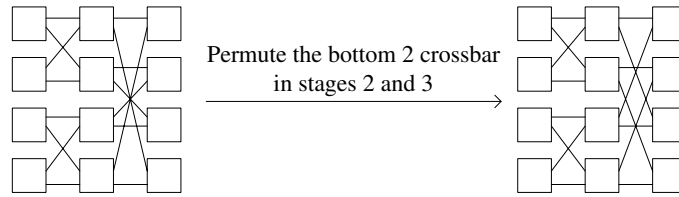


Figure 1.5.8: A permutation to achieve  $BP$

Suppose  $G_{13}$  is obtained otherwise. There must exist a  $d'$ -set of  $K_{d,d}$ ,  $d' > d$ , in  $G_1$  connected in  $G_2$  through a  $d'$ -set of  $K_{d,d}$  in  $G_2$ . Note that an input in this component touches only  $d^2$  among the  $dd'$  outputs. Hence there must exist another input reaching some, but not all, of these  $d^2$  outputs, violating the buddy property.

For  $k \geq 3$ , then the situation described in the last paragraph must also happen.

- (iii)  $s \geq 4$ . Consider the two subnetworks  $G_{13}$  and  $G_{2s}$ . By induction,  $G_{13}$  can be represented by a vector  $(u_1, u_2)$  and  $G_{2s}$  by  $(u'_1, \dots, u'_{s-2})$ . By Corollary 1.5.6, we can permute the crossbars in stage  $k$ ,  $2 \leq k \leq s$ , such that  $u'_1 = u_2$  and  $u'_k = u''_k$  for  $2 \leq k \leq s-2$ . Therefore  $G_{1s}$  is represented by the vector  $(u_1, u_2, u''_3, \dots, u''_{s-1})$ , i.e.,  $G_{1s}$  is a  $BP$  network.

□

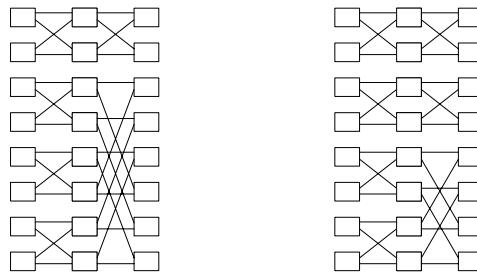
**Corollary 1.5.15.** *Two  $d^P B$  networks are equivalent if the characterization vector of one can be obtained from the other through a permutation.*

Note that Corollary 1.5.15 generalizes Theorem 1.5.1.

Figure 1.5.9(a) gives a  $d^P$  network which is not equivalent to a  $BP$  network (since it has a component of size  $\geq d^2$ ). Therefore the buddy condition can not be dropped from Theorem 1.5.13.

Figure 1.5.9(a) is also an example of a  $d^P$  network which is not equivalent to a buddy network, while Fig. 1.5.9(b) gives a buddy network which is not equivalent to either a  $d^P$  or a  $BP$  network.

Finally, we have

Figure 1.5.9: A  $2^P$  network and a buddy network

**Theorem 1.5.16.** *The number of equivalent classes among  $s$ -stage bit-permutation networks is  $\sum_{t=1}^{s-1} \sum_{i=1}^t \binom{t}{i} (-1)^{t-i} i^{s-1}$ .*

*Proof.* The number of canonical sequences of length  $m$  with  $t$  distinct elements is simply the number of ways of labeling  $m$  cells with exactly  $t$  symbols. Using the principle of inclusion and exclusion, this number is

$$\sum_{i=1}^t (-1)^{t-i} \binom{t}{i} i^m.$$

Let  $i_1, \dots, i_t$  denote the ordering of the  $t$  symbols according to their appearances in the  $m$  cells. Then the  $t!$  possible orderings reduce to the same canonical sequence, which has the ordering  $(1, \dots, t)$ . Therefore we have to divide by  $t!$  to obtain the number of canonical sequences with exactly  $t$  symbols. Summing over  $t$  and setting  $m = s - 1$ , we obtain Theorem 1.5.16.  $\square$

Note that the number of equivalent classes is independent of  $d$ .

For  $B(0, n)$ , Li also introduced a notion called guide to facilitate the self-routing. A *guide* is an  $n \times n$  upper-left triangular matrix  $(g_{ij})$  where the set  $\{1, \dots, n\}$  labels the rows (from top to bottom) and the columns (from left to right). Then  $g_{ii} = n$  for  $1 \leq i \leq n$ , and  $g_{ij}$  represents the number that  $g_{i,j-1}$  permutes to  $\Pi_{j+1}$ . For example, for  $SE_4$ ,  $\Pi_i = (1432)$  for all  $i$ . Then  $g_{12} = 3$  since 4 permutes to 3 in  $\Pi_3$ .  $g_{13} = 2$  since 3 permutes to 2 in  $\Pi_2$ , and so on. The guide is shown in Fig. 1.5.10(a). For  $BY_4^{-1}$ ,  $\Pi_i = (i, 4)$  for  $1 \leq i \leq 3$ ; its guide is shown in Fig. 1.5.10(b). For  $BL_4$ ,  $\Pi_i = (i, i+1, \dots, n)$ ; its guide is shown in Fig. 1.5.10(c). For  $BY_4$ ,  $\Pi_i = (n-i, 4)$  for  $1 \leq i \leq 3$ ; its guide is shown in Fig. 1.5.10(d).

4 3 2 1	4 1 1 1	4 1 1 1	4 3 3 3
4 3 2	4 2 2	4 2 2	4 2 2
4 3	4 3	4 3	4 1
4	4	4	4
(a)	(b)	(c)	(d)

Figure 1.5.10: Guides for  $SE_4$ ,  $BY_4$ ,  $BL_4$  and  $BY_4^{-1}$ 

To route  $(x_1, \dots, x_n)$  to  $(y_1, \dots, y_n)$  in a BP network represented by the canonical vector  $(u_1, \dots, u_{n-1})$ , set  $x_n = y_{g_{in}}$  at the stage- $i$  switch, i.e., the path goes to the upper (lower) output if  $y_{g_{in}} = 0(1)$ .  $\Pi_i$ ,  $1 \leq i \leq n-1$ , then moves  $y_{g_{in}}$  from position  $g_{ij}$  to position  $g_{i,j+1}$ . So at stage  $n$ ,  $y_{g_{in}}$  is moved to position  $g_{in}$ . Namely, the outcome vector is  $(y_1, \dots, y_n)$ . We illustrate by some examples.

$$SE_4 : (x_1x_2x_3x_4) \xrightarrow{g_{14}} (x_1x_2x_3y_1) \xrightarrow{\Pi_1} (x_2x_3y_1x_1) \xrightarrow{g_{24}} (x_2x_3y_1y_2) \xrightarrow{\Pi_2} \\ (x_3y_1y_2x_2) \xrightarrow{g_{34}} (x_3y_1y_2y_3) \xrightarrow{\Pi_3} (y_1y_2y_3x_3) \xrightarrow{g_{44}} (y_1y_2y_3y_4).$$

$$BY_4^{-1} : (x_1x_2x_3x_4) \xrightarrow{g_{14}} (x_1x_2x_3y_3) \xrightarrow{\Pi_1} (x_1x_2y_3x_3) \xrightarrow{g_{24}} (x_1x_2y_3y_2) \xrightarrow{\Pi_2} \\ (x_1y_2y_3x_2) \xrightarrow{g_{34}} (x_1y_2y_3y_1) \xrightarrow{\Pi_3} (y_1y_2y_3x_1) \xrightarrow{g_{44}} (y_1y_2y_3y_4).$$

## 1.6 Graphs and Channel Graphs

A graph  $G$  consists of a set  $V$  of *vertices* and a set  $E$  of pairs of vertices called *edges*. If the pairs are ordered pairs, then a graph is known as a *digraph*, a vertex is called a *node* and an edge an *arc*. The degree of a vertex  $v$ ,  $d_G(v)$ , is the number of edges it is incident to. Two vertices (edges) are *adjacent* if they are incident to the same edge (node). A *coloring* (an *edge-coloring*) of a graph is to assign a color to each node (edge) such that all adjacent nodes (edges) have different colors. We say that the graph can be  $c$ -(edge-)colored if  $c$  colors suffice. These terms have their counterparts for digraphs. In particular, the modifier “in” or “out” is used to associate with arcs coming into or going out from a node.

In a weighted graph, each edge has a weight assumed to be a fraction. An edge-coloring of a weighted graph satisfies the requirement that the sum of weights of all edges of the same color incident to a vertex  $v$  must not exceed 1

for all  $v$ .

By treating a crossbar as a node and a link as an arc, a switching network is very much like a digraph except that each input (output) switch has external links dangling without connecting to any nodes and hence cannot be considered as arcs (but the computer network version does not have this problem). To remedy this irregularity, the graph theorist prefers to define a true digraph  $G(V, I, O, E)$  from a network by converting each link as a node including the inputs  $I$  and the outputs  $O$ , while a crosspoint connecting two links in the network becomes an arc in the graph. Note that a crossbar is represented by a complete bipartite subgraph whose recognizability may depend on the drawing of  $G(V, I, O, E)$ . Figure 1.6.1 shows the digraph representation of the network in Fig. 1.2.1.

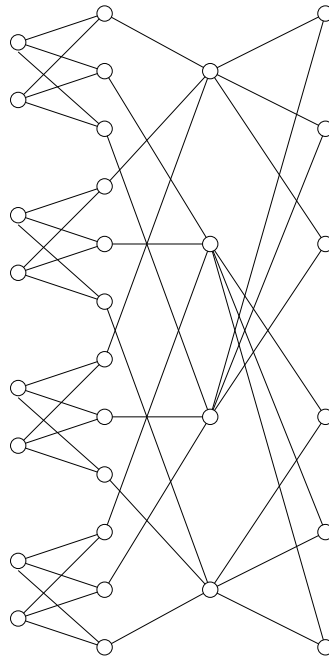


Figure 1.6.1: Digraph of Fig. 1.2.1.

While a telephone network does have the external links to connect to outside world, a processor-memory network can use the input crossbars to represent processors, the output crossbars to represent memories, so that requests

are internally generated. Such a network is often drawn without the external links, and itself interpreted as a digraph.

A graph is called *bipartite* if its vertices can be partitioned into two disjoint parts such that edges exist only between parts. A frame  $F$  can be modelled by a bipartite graph  $G(F)$  with the input crossbars and output crossbars as the two parts, and an edge  $(X, Y)$  for each request  $(x, y)$  with  $x$  an input on  $X$  and  $y$  an output on  $Y$ . For a 3-stage Clos network, the requirement that only one request from each input crossbar and output crossbar can be connected through a given middle crossbar can then be translated into an edge-coloring of  $G(F)$  where each color corresponds to a middle crossbar. Namely, we can assign each middle crossbar with a distinct color which will route all edges of that color.

Bipartite graphs are also used to describe the connection pattern between two adjacent stages. We define some bipartite graphs with special properties useful to nonblocking networks. Let  $N$  denote the number of inputs and  $M$  the number of outputs.

**Expander.** For a given family  $F$  of input-subsets, each input-subset  $S$  can reach an output subset  $S'$  with  $|S'| > |S|$ . Depending on different constraints on  $F$  and different requirements on  $|S'|$ , various expanders can be defined. The original definition is: An  $(N, d, p)$ -expander is a  $d$ -regular graph with  $M = N$  such that every set of  $x$  inputs,  $1 \leq x \leq N$ , has at least  $x + p(1 - x/N)x$  outputs as neighbors. Pinsker (1973) first proved the existence of a linear-cost  $(N, d, p)$ -expander and Margulis (1975) gave the first construction but leaving  $p$  unspecified. Gabber–Galil (1981) provided this constant. The currently best construction requires a cost about  $30N$ .

In a multiplicative  $(\alpha, \beta)$ -expander, each subset  $S$ ,  $|S| \leq \alpha N$ , of inputs can reach  $\beta|S|$  outputs with  $\beta > 1$ . In an additive  $(p, q)$ -expander, each subset  $S$ ,  $|S| \leq p$ , of inputs can reach  $p + q$  outputs,  $q > 0$ .

**Concentrator.** For any  $k$ -subset  $K_1$  of inputs, there exists a matching between  $K_1$  and a  $k$ -subset  $K_2$  of outputs ( $K_2$  is not prespecified) as long as  $k \leq \min\{N, M\}$ . Pinsker (1973) showed that a concentrator can be constructed from an expander with essentially the same cost.

**Partial concentrator.** A concentrator becomes a partial concentrator, denoted by  $(N, M, c)$ , when the above condition holds only for  $k \leq$  a constant  $c$ .  $c$  is called the *capacity* of the partial concentrator.

**Superconcentrator.** A concentrator with  $K_2$  also specified (but note that the matching is not specified). Valiant (1976) first proved the existence of a linear-cost superconcentrator. Gabber–Galil gave a method to convert an expander to a superconcentrator with the same cost complexity. The currently best construction requires a cost about  $100N$ .

**Hyperconcentrator.** A superconcentrator with a consecutive set  $K_2$ .

**Infra concentrator.** The set of the first  $k$  inputs can be mapped to some set of  $k$  outputs preserving the order.

In particular

**Infra connector.** The set of the first  $k$  inputs can be mapped to any set of  $k$  outputs prescribing the order.

The reader is referred to Alon–Galil–Milman (1987), Tanner (1987), Pippenger (1990) for more detailed discussions on these bipartite graphs.

The shuffled  $BY^{-1}(n)$  differs from  $BY^{-1}(n)$  in the fact that inputs are assigned to the input crossbars in a shuffled pattern, i.e., suppose the  $N$  inputs are lined up at “stage 0”, then inputs  $i$  and  $i + N/2$  are connected to input crossbar  $i$  for  $1 \leq i \leq N/2$ . Gfman (1978) proved

**Theorem 1.6.1.** *The shuffled  $BY^{-1}(n)$  is an infra connector.*

*Proof.* The restrictions on routable permutations for a shuffled  $BY^{-1}(n)$  is that if inputs  $i$  and  $j$  have the same last  $l$  bits, then  $\Pi(i)$  and  $\Pi(j)$  cannot have the same first  $n - l$  bits. On the other hand, when inputs are consecutive and the mapping to output is monotone, then  $i < j$  implies  $\Pi(j) - \Pi(i) \geq j - i$ . Hence if  $i$  and  $j$  have the same last  $l$  bits, then  $\Pi(i) - \Pi(j) \geq 2^l$ , i.e.,  $\Pi(i)$  and  $\Pi(j)$  cannot have the same first  $n < l$  bits. Consequently, such a permutation is routable by the shuffled  $BY^{-1}(n)$ .  $\square$

For a given MIN  $\nu$ , the channel graph  $CG(x, y)$  between an input  $x$  and an output  $y$  is the union of all paths connecting  $x$  and  $y$ . The channel graph, first proposed by Lee (1955) and LeGall (1956) independently, has been widely used in studying the blocking probability of MIN. Recently, Shyy and Lea (1991) showed that it can also be useful in the study of nonblocking networks.

Consider an  $s$ -stage channel graph  $CG$ . For odd  $s$ ,  $CG$  is called a *double-tree channel graph* if  $CG$  can be partitioned at some stage into two trees whose leaves are identified at that stage. For even  $s$ , there exists a stage of

links connecting the identified pairs of leaves. A double-tree is *series parallel* if the leaves of the two trees in their natural orders are mapped by an identity mapping; otherwise, it is *spiderweb*. Figure 1.6.2 illustrates both types. A symmetric  $s$ -stage series-parallel channel graph can be sequentially decomposed into  $\lfloor (s-1)/2 \rfloor$  shells where shell  $i$  consists of links between stages  $i$  and  $i+1$ , and between stages  $s-i$  and  $s-i+1$ .

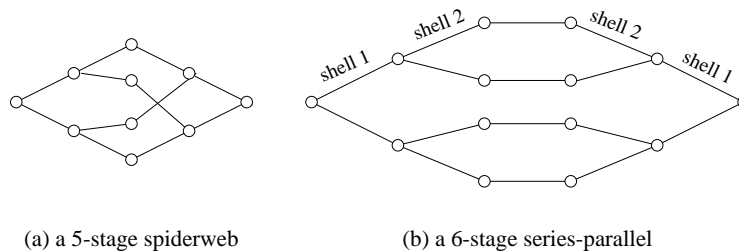


Figure 1.6.2: Two double-trees

Clearly,  $CG(x, y)$  plays an important role in determining whether the  $(x, y)$  request can be connected. It is particularly convenient if  $CG(x, y)$  is invariant to  $x$  and  $y$ , since then the connectability of one pair stands for the whole network. Often, the hardware of a network, namely, the number of stages, the number and the size of the crossbars at each stage, is given. The question is to determine the interconnection pattern inducing the best channel graph. It is well known (Hwang, 1979) that a spiderweb channel graph is usually better than a series-parallel channel graph for blocking networks. Among series-parallel channel graphs with fixed number of stages and fixed number of paths, it is better to have branching at an outer shell than an inner shell. It is of interest to note that in Shyy and Lea's analysis of channel graph for nonblockingness (see Sec. 2.2), it is also better to have more branching at outer stages, even though the two underlying models are totally different.

Since the nonblockingness analyses of bit permutation networks to be discussed in later chapters depend only on the (invariant) channel graphs of the networks, the question arises whether nonequivalent bit permutation networks can have isomorphic channel graphs. If that were the case, then the study of equivalent bit permutation networks becomes less relevant. Tong, Hwang and Chang (1998) gave the reassuring "no" answer. First they proved

**Theorem 1.6.2.** *The channel graph of a connected bit permutation network is invariant to  $x$  and  $y$ .*

*Proof.* Let  $CG(x)$  denote the union of all paths from an input  $x$ . Let  $V_i(x)$  denote the set of stage- $i$  nodes in  $CG(x)$ . Then  $V_i(x)$  consists of those nodes which may differ from (the  $d$ -nary representation of)  $x$  in bits  $I_j = j = 1, \dots, i$ . Therefore  $CG(x)$  is invariant to  $x$ . Let  $x' \neq x$  be another input. Since the bit permutation network is connected,  $G(x)$  intersects  $G(x')$  at some stage  $k$ . By the buddy property  $V_k(x) = V_k(x')$ . Therefore  $CG(x, y)$ , being the concatenation of the graph from  $x$  to  $V_k(x)$  and the graph from  $V_k(x)$  to  $y$  is isomorphic to the concatenation of the graph from  $x'$  to  $V_k(x')$  and the graph from  $V_k(x')$  to  $y$ , which is  $CG(x', y)$ . Similarly, we can prove  $CG(x, y) = CG(x, y')$ . Hence  $CG(x, y) = CG(x', y) = CG(x', y')$ .  $\square$

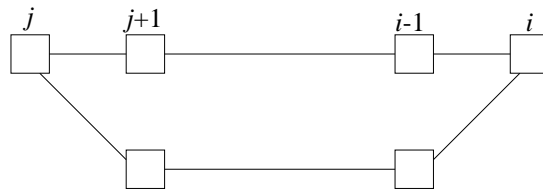


Figure 1.6.3: A cycle from stage  $j$  to stage  $i$

**Theorem 1.6.3.** *Two connected bit permutation networks are isomorphic if and only if their channel graphs are isomorphic.*

*Proof.* The “only if” part is trivial. We prove the “if” part.

Let  $\nu_1(n; I_{k_1}, \dots, I_{k_{s-1}})$  and  $\nu_2(n; I_{k'_1}, \dots, I_{k'_{s-1}})$  denote two nonisomorphic  $s$ -stage connected bit permutation networks. Then there exists a smallest stage  $i$  such that  $k_i \neq k'_i$ . Without loss of generality, assume  $k_i > k'_i$ . Then  $k'_i \in \{k'_1, \dots, k'_{i-1}\} = \{k_1, \dots, k_{i-1}\}$ . Let  $j$  be the largest subscript such that  $k'_i = k'_j = k_j$ . Then the channel graph of  $\nu_2$  from stage  $j$  to  $i$  contains a cycle as shown in Fig. 1.6.3, which is not present in the channel graph of  $\nu_1$ .  $\square$

**References**

- Ackroyd, M. H. 1979. Call repacking in connecting networks. *IEEE Trans. Commun.*, **27**, 589–591.
- Agrawal, D. P. 1983. Graph theoretical analysis and design of multistage interconnection networks. *IEEE Trans. Comput.*, **C32**, 637–648.
- Alon, N., Galil, Z., & Milman, V. D. 1987. Better expanders and superconcentrators. *J. Alg.*, **8**, 337–347.
- Beneš, V. E. 1965. *Mathematical Theory of Connecting Networks and Telephone Traffic*. New York: Academic.
- Bermond, J. C., Fourneau, J. M., & Jean-Marie, A. 1987. Equivalence of multistage interconnection networks. *Inform. Proc. Lett.*, **26**, 45–50.
- Cantor, D. G. 1971. On nonblocking switching networks. *Networks*, **1**, 367–377.
- Chang, G. J., Hwang, F. K., & Tong, L. D. 1999. Characterizing bit permutation networks. *Networks*, **33**, 261–267.
- Clos, C. 1953. A study of non-blocking switching networks. *Bell Syst. Tech. J.*, **32**, 406–424.
- Gabber, O., & Galil, Z. 1981. Explicit construction of linear size superconcentrators. *J. Comput. Syst. Sci.*, **22**, 407–420.
- Halpenny, L. 1990. Nonblocking staged and repeated stage networks. Unpublished manuscript.
- Hui, J. Y. 1990. *Switching and Traffic Theory for Integrated Broadband Networks*. Norwell, MA: Kluwer.
- Hwang, F. K. 1979. Superior channel graphs. *In: Proc. 9<sup>th</sup> Int. Teletraffic Cong.* Torremolinos, Spain.
- Hwang, F. K. 2003. Equivalence among extra-stage banyan-type networks, preprint.
- Hwang, F. K., Liaw, S. C., & Yeh, H. G. 1998. Equivalent classes of extra-stage networks, unpublished.
- Kruskal, C. P., & Snir, M. 1986. A unified theory of interconnection network structure. *Theo. Comput. Sci.*, **48**, 75–94.

- Lawrie, D. H. 1976. Access and alignment of data in an array processor. *IEEE Trans. Comput.*, **25**, 1145–1155.
- Lee, C. Y. 1955. Analysis of switching networks. *Bell System Tech. J.*, **34**, 1287–1315.
- LeGall, P. 1956. Étude du blocage dans les systèmes de commutation téléphonique. *Ann. Télécommun.*, **11**.
- Li, S.-Y. R. 2001. Algebraic Switching Theory and Broadband Applications. *New York: Academic*.
- Margulis, G. A. 1975. Explicit constructions of concentrators. *Prob. Inform. Transm.*, **9**, 325–332.
- Ofman, Y. P. 1965. A universal automation. *Trans. Moscow Math. Soc.*, **14**, 200–215.
- Parker, D. S. 1980. Notes on shuffle/exchange-type networks. *IEEE Trans. Comput.*, **29**, 213–222.
- Perrier, P., & Prucnal, P. 1989. Self-clocked optical control of a self-routed photonic switch. *J. Lightwave Tech.*, **7**.
- Pinsker, M. 1973. On the complexity of a concentrator. *Proc. 7<sup>th</sup> ITC*, Stockholm, Sweden, 318/1–318/4.
- Pippenger, N. 1990. Communication networks. *Pages 807–833 of: Leeuwen, J. V. (ed), Handbook of Theoretical Computer Science*. Amsterdam: Elsevier.
- Shyy, D.-J., & Lea, C.-T. 1991.  $\log_2(N, m, p)$  strictly nonblocking networks. *IEEE Trans. Commun.*, **39**, 1502–1510.
- Siegel, H. J., & Smith, S. D. 1978. Study of multistage SIMD interconnecting networks. *Pages 307–314 of: Proc. 5<sup>th</sup> Ann. Symp. Comput. Arch.*
- Smyth, C. Y. 1988. Nonblocking photonic switch networks. *IEEE J. Select. Areas Commun.*, **6**, 1052–1062.
- Spanke, R. A., & Beneš, V. E. 1987. N-stage planar optical permutation network. *Appl. Opt.*, **26**, 1226–1229.
- Tanner, R. M. 1987. Explicit concentrators from generalized N-gons. *SIAM J. Alg. Disc. Method*, **5**, 287–293.

- Tong, L. D., Hwang, F. K., & Chang, G. J. 1998. Channel graphs of bit permutation networks, *Theor. Comput. Sci.*, **263**, 139-143.
- Thompson, C. D. 1978. Generalized connection network for parallel processor interconnection. *IEEE Trans. Comput.*, **27**, 1119-1124.
- Valiant, L. G. 1976. Graph-theoretic properties in computational complexity. *J. Comput. Syst. Sci.*, **13**, 278-285.
- Varma, A., & Raghavendra, C. S. (eds). 1994. *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice*. IEEE Computer Soc., Los Alamitos, CA.
- Wu, C.-L., & Feng, T.-Y. 1980. On a class of multistage interconnection networks. *IEEE Trans. Comput.*, **29**, 694-702.