Chapter 5

Design of the Cambridge Fast Packet Switch

In this chapter the design of the Cambridge Fast Packet Switch is considered in detail. A review is first presented of some early work on binary routing networks. Although this work was developed in the context of the local area network it introduces some ideas on high-speed switching mechanisms which were to influence the design of the fast packet switch. The discussion is then widened to introduce the general principles which guided the switch design. Finally, the switch itself is described with some of the design options whose influence on the performance of the switch will be presented in the following chapter.

5.1 Binary Routing Networks

The Buffered Binary Routing Node

Binary routing networks were first proposed about ten years ago by Hopper and Wheeler [69] as a new class of local area network with simpler hardware and better performance than the existing bus or ring designs. The networks were based upon the buffered binary routing node as shown in fig. 5.1. The value of the first bit in the route field of an arriving packet was used to direct the node to switch the packet either to the upper port if zero or to the lower output port if one. The whole route field of the packet was rotated by one bit so as to present the next bit of the route field to the succeeding node. If the selected output was free the packet was transmitted across the node with little delay but if it was busy the packet was buffered and a backward busy signal asserted to prevent the transmission of further packets, on the relevant input port, until the buffered packet was transmitted. This formed one of the first proposals to employ a buffered 2×2 self-routing crossbar packet switch in a communications network. However, as the application envisaged was that of local area networks, most of the topologies considered for constructing binary routing networks were buses, rings and trees which distributed the switching nodes across the local

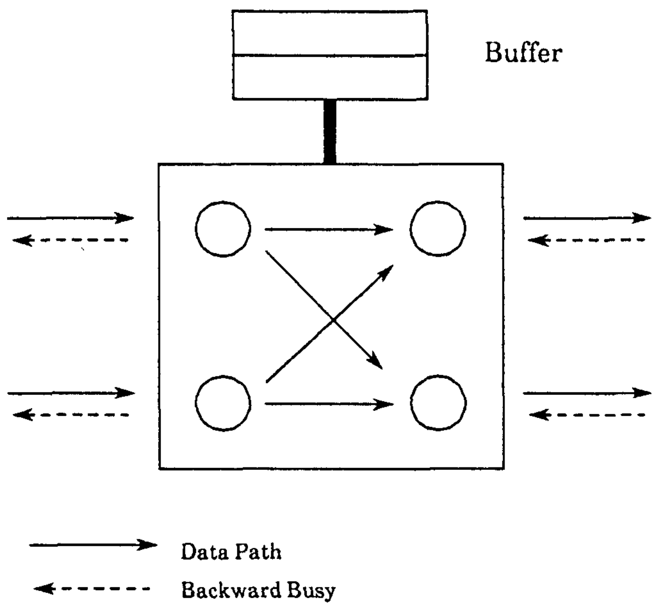

Figure 5.1: The structure of a buffered binary routing node.

area. The switching mechanism thus became source routing rather than self-routing and the networks were suitable for operation only at relatively low loads. A binary routing network based upon a shuffle exchange network, to form a fast packet switch, was foreseen but the idea was not developed.

Non-Buffered Networks

Between 1981 and 1983 the author was involved in a project to investigate possible switching mechanisms for the interconnection of high-speed optical fibre transmission links. The work of Hopper and Wheeler was taken as a starting point but it soon became clear that the implementation of a buffered switching node would prove expensive and difficult, at the speed required, in the technology available at that time. If possible, a solution capable of implementation in gate array devices was sought. A switching mechanism based upon a non-buffered binary routing node was therefore proposed in [110, 112]. The buffer was removed from the binary routing node and the backward busy signal modified to become a reverse path through the network, back to the originating node, established in parallel with the forward path. If a packet incident at a node should find that the desired output was busy the reverse path was used to inform the originating node of the contention with very little delay. On receiving a contention signal the originating node removed this attempt to transmit the packet and tried again a short while later.

In [111] a self-routing crossbar switch was proposed according to the design presented in fig. 4.6 and the use of this switching element in both delta and Beneš networks was suggested but no attempt was made to follow this up with any performance investigations. A detailed investigation into the design and construction of a non-buffered binary routing node was reported in [113] in which a hardware model was constructed in TTL operating at 10 MHz. From this model it was clear that a non-buffered binary crossbar switching element could be fabricated with less than 400 gates and that crossbar switching elements of degree 8 or possibly 16 could be achieved.

Binary Shuffle Exchange Networks

The suggestion in [69] that a fast packet switch might be constructed from a binary routing network in a shuffle exchange topology was investigated by Milway [100] for use in local area network applications. He considered the performance of both buffered and non-buffered designs and concluded that within the context of the local area network the performance of the non-buffered design was adequate. He also considered the improvement in performance that could be obtained by a form of input queue by-pass algorithm and also by adding additional stages of switching to the front of networks which selected at random from the available paths to the destination. As the work was directed to the use of binary routing networks in the context of the local area network, much of the investigation was directed towards operation at low loads and only switching elements of degree 2 were considered. The hardware implementation of a non-buffered binary routing node was investigated using programmable logic array devices operating at 10 MHz and the performance of a 4×4 network of binary routing nodes was shown to agree well with performance results derived from a simulation model.

5.2 Design Issues

Output Buffered Switching Elements

A fast packet switch must be constructed from either buffered or non-buffered switching elements. Considering the existing switch designs about half of them propose the use of a buffered switch fabric using output buffered switching elements [141, 120, 4, 36]. This results in a very complex design of switching element requiring a large number of buffers on each output of every switching element in the switch fabric if packet loss is to be kept to acceptable levels. Such a design is not unreasonable considering the current state of the art in VLSI technology yet a simpler design of switching element may be worth investigating for a number of reasons. First, a simple design of switching element is likely to be suitable for fabrication in a wide range of implementation technologies. Thus a simple but high-speed switch may well be capable of a greater capacity. Looking toward the long term, with a simple design of switching element it may be possible to implement the data paths of the switch fabric using integrated optics [14, 134, 64] leading to very high-speed operation. If it is possible to implement the switching element using gate array technology a more flexible switch design will result. Thus the resulting switch design may be much more easily customised to fit each environment that it is commissioned to serve. Hence, parameters such as line code, priority levels, packet length, etc. may be modified much more easily than in a fixed VLSI design. Finally, the fact that most other switch designs employ a VLSI implementation makes a design capable of implementation in small gate arrays an interesting subject for investigation.

Input Buffered Switching Elements

A design based upon an internally buffered switch fabric may still be practicable with the requirement for a simple design of switching element if the switching element is input buffered. Two designs [148, 128] based upon input buffered switching elements have already been investigated. Also, Milway has shown [100] that even with infinite buffers at both inputs of a 2×2 buffered switching element, the performance of the resulting switch is only slightly in excess of that of an input buffered non-blocking switch fabric. If a simple buffered design is selected, the size of the switching element is likely to be limited to 2×2. Thus even if the buffering were implemented in memory external to the logic of the switching element, performance is unlikely to greatly exceed that of the input buffered non-blocking switch fabric.

A switch fabric constructed from switching elements of degree two presents another difficulty. The number of switch stages is maximised, hence the number of interconnections within the switch fabric is also maximised. It is likely that in any implementation, the number of interconnections required in the switch fabric is going to limit the maximum size of switch that can be realised. Thus, not only would a simple switching element design be desirable but also the size of that switching element would preferably be of a degree greater than two. If not, the switch fabric should be capable of partitioning such that a standard array of switching elements may be implemented in a single device from which the switch fabric may be constructed. This is the approach taken in the Batcher-banyan switch fabric [34] but it has resulted in the design of several different VLSI devices to implement the switch fabric.

Non-Buffered Switching Elements

Considering the possibility of a non-buffered switch fabric, it is clear that if simplicity of implementation is a major requirement an input buffered switch is a much more suitable candidate than an output buffered switch. The only major design of input buffered fast packet switch so far proposed for communications applications, the three phase Batcher-banyan [71], uses a non-blocking switch fabric. This requires a much larger switch fabric than is needed for a simple blocking switch fabric and thus necessitates implementation in VLSI. Also the algorithm required to ensure non-blocking operation renders the use of short packet lengths extremely inefficient. Hence the question of the performance of the various possible blocking switch fabrics in comparison with that of the non-blocking switch fabric is of considerable interest. Further, a number of techniques are available to enhance the performance of an input buffered switch fabric, for example input queue by-pass and a multi-plane switch

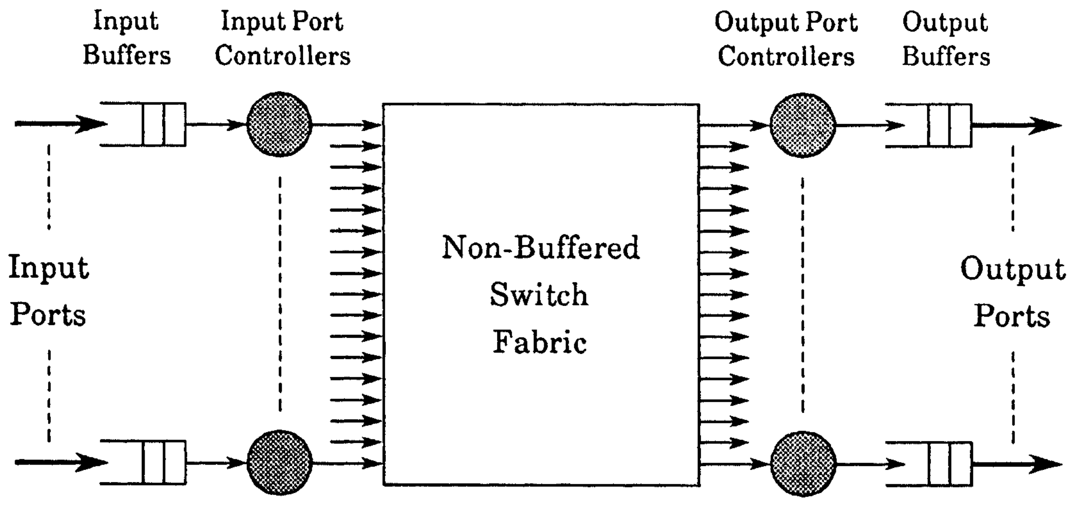

Figure 5.2: The basic structure of the Cambridge Fast Packet Switch.

fabric with output buffering across the multiple switch planes. It may be interesting to investigate how close the use of these techniques may improve the performance of a blocking switch fabric towards that of an output buffered switch. (The output buffered switch with infinite output queues represents the best possible theoretical switch performance.)

5.3 The Switching Mechanism

At the lowest level, a fast packet switch will generally use a connection-oriented switching mechanism in which a virtual circuit is established between source and destination before communications traffic is exchanged. Thus at each input port of the switch a table is maintained which lists the parameters of each active connection and assigns a label to each connection. The label appears in the header of every incoming packet and is used to identify the connection to which each packet belongs. In general a connection will traverse several switches. Hence, to avoid problems of global label allocation, each switch allocates its own labels to refer to the connections that it supports. The label of every packet must therefore be translated as it traverses each fast packet switch in the path. This operation is generally performed at the input port with the replacement label stored as a field in the connection table. Other fields in the table will include the output port of the switch through which the connection is routed and possibly other parameters such as priority and traffic type.

Connections are set up across a network of fast packet switches using a message based signalling system similar to that of a modern telephone network. However, as connections are virtual, no system bandwidth is exclusively allocated to any connection, so connections may lay idle for substantial periods of time at little cost to the network.

The basic structure of the Cambridge Fast Packet Switch is given in fig. 5.2. The connection tables are housed in the input port controllers while in a simple switch the output port controllers are required for little more than line conditioning and similar tasks. In the basic switch, output buffers would only be required in order to interface to output lines running at a different rate from that of the switch fabric. An incoming packet arrives in the input buffer which is a first in first out (FIFO) queue. When free, the respective input port controller extracts the label from the packet at the head of the queue and uses it to reference the connection table. Each input port controller operates asynchronously, at the packet level, and independently of all other controllers. From the table it receives two components, an outgoing label and a tag. The outgoing label is used to replace the incoming label within the packet. The tag specifies the required destination output port of the switch and is attached to the front of the packet. The input port controller then initiates a set-up attempt by launching the packet into the switch fabric, tag first and in bit serial form. There are two possible outcomes, either the packet will be successful and reach the desired output buffer, or it will fail. A set-up attempt may fail either because it is blocked by other traffic within the switch fabric or because the requested output port is busy serving another packet. If the set-up attempt fails, the switch fabric will assert a collision signal which is returned to the input port controller, along a reverse path, typically within a few bit times of emission of the packet tag. On receiving the collision signal the input port controller removes the set-up attempt from the switch fabric and waits for a delay typically equivalent to 10% of the length of a packet. This is the retry delay and at the end of this period the input port controller begins a fresh attempt to transmit the packet. It continues to do so until it is successful or until it exceeds a limit designed to detect fault conditions.

A slightly more complex algorithm that offers an improvement in performance at high loads does not repeatedly attempt to transmit the same packet but on the failure of a set-up attempt searches through the input queue and attempts to transmit the second packet. If that attempt fails the third packet on the queue is attempted and so on cyclically through the queue until a successful transmission is achieved. This overcomes the so called 'head of the line' blocking problem [71, 76] but care has to be taken not to get packets on the same virtual circuit out of sequence. This algorithm will be referred to as input queue by-pass [19].

A simple model of the operation of the fast packet switch may be drawn by analogy with the operation of a well known local area network: Ethernet. Ethernet may be considered as a fast packet switch which distributes the switching function across the local area using a single shared medium switch fabric. The fast packet switch described above merely confines the switching function within a box so that a multi-path medium of much higher bandwidth may be implemented. The input port controller of the fast packet switch corresponds to the media access controller of Ethernet and in both cases the controller launches a packet into the switch fabric and if it is unsuccessful the switch fabric informs it immediately. The difference between the two lies in the fact that in Ethernet a collision destroys both colliding packets therefore an exponential random back-off algorithm is required. In the fast packet switch, however, collisions are non-destructive in the sense that one of the colliding packets always survives, so a simple retransmission algorithm is sufficient.

5.4 The Switch Fabric

The switching mechanism described above requires a self-routing switch fabric. The switch fabrics proposed for investigation are therefore constructed from multi-stage interconnection networks of crossbar switching elements. The switching mechanism relies upon the ability of the switch fabric to inform the input port controllers of a collision between a packet attempting set-up and one already established. This is achieved by setting up a reverse path through the switch fabric in parallel with the forward path. Every link in the interconnection network consists of two paths, a forward path to carry the data and a reverse path for the collision signal. Every switching element sets up both paths in parallel provided that the required output on the forward path is free. If blocked, it will return a collision signal on the reverse path and all switching elements from the partially established path will return to the idle state as soon as the input port controller removes the failed packet from the switch fabric.

Three classes of self-routing switch fabric present themselves as deserving of investigation under the above switching mechanism: non-blocking, rearrangeable non-blocking and blocking. Although several methods of constructing a non-blocking switch have been demonstrated they tend to be expensive in terms of their hardware requirements. This class of switch fabric will thus be used as a standard against which other switch fabrics may be compared.

The Beneš network belongs to the class of rearrangeable non-blocking networks when controlled by a centralised switching algorithm. Its performance under the various possible distributed algorithms is of interest. The delta network seems the most appropriate class of blocking, self-routing switch fabric to bear closer scrutiny.

Whilst the majority of research interest has been expended upon multi-stage interconnection networks constructed from 2×2 switching elements, previous investigations suggested that it might be possible to implement crossbar switching elements of up to degree 16 in gate array technology [113]. It is therefore of considerable interest to investigate what effect the degree of the switching element has upon the performance of both delta and Beneš switch fabrics.

Delta Networks

An example of a 64×64 delta network constructed from switching elements of degree 8 is given in fig. 5.3. The self-routing property is easily demonstrated. The output ports of each switching element are numbered from 0 to 7 with the uppermost port 0. The output ports of the switch fabric are numbered from 0 to 63 with the uppermost port 0. If the number of the required output port is expressed to the base 8, two digits will be required. If the most significant digit is used to select the output of the

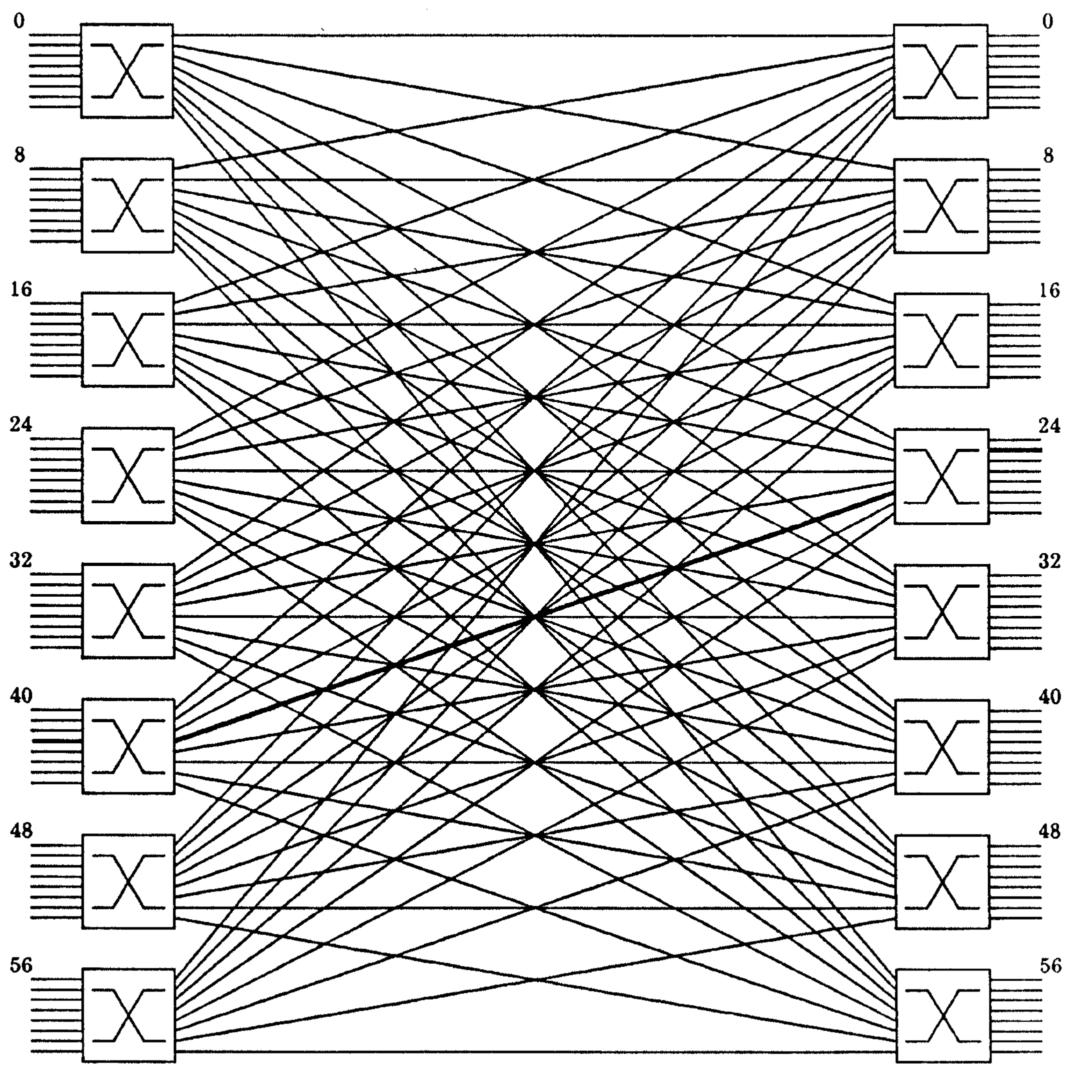

Figure 5.3: A 64×64 delta network of 8×8 switching elements.

switching element of the first stage and the least significant digit that of the second stage, a path to the required output port is established. This is true regardless of the input port from which the path originates. A path from input port 43 to output port 25, (i.e. output port number 31 to base 8,) is illustrated.

The use of switching elements of degree greater than 2 raises the problem that delta networks are only defined in sizes that are an integer power of the degree of the switching element. This would result in large increments between valid sizes of network. The proposed solution is to replicate the interconnection links between stages which permits networks to be built to any size that is an integer power of 2, from switching elements of any degree that is also an integer power of 2, [1, 85]. Thus a modified delta network of size N requires s stages, where $s = \lceil \log_d N \rceil$, of N/d switching elements per stage and each link of the pure delta network is replicated

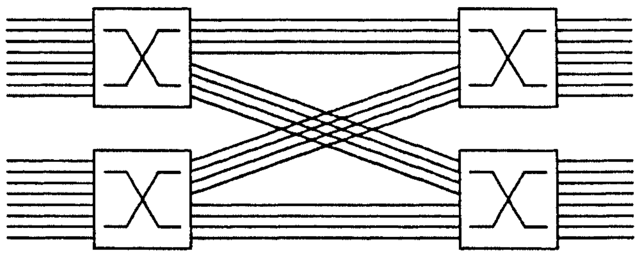

Figure 5.4: A 16×16 modified delta network of 8×8 switching elements.

d^s/N times.[1] Strictly speaking the modified delta network is no longer a member of the class of banyan networks and fig. 5.4 illustrates a 16×16 modified delta network of switching elements of degree 8. Clearly this network offers four equivalent paths between every input/output pair.

We now have the possibility of multiple paths existing between the same pair of input and output ports. This increases the performance and fault tolerance of the switch but requires an algorithm to select between equivalent paths. Fortunately, as there is no buffering within the switch fabric, each incident packet may be routed independently across the switch fabric without the risk of out-of-sequence errors between packets travelling on the same virtual circuit. Two algorithms have been investigated: searching and flooding. In the searching mechanism the input port controller attempts to transmit across each of the equivalent paths in sequence until it meets with success. It does this simply by incrementing the most significant bits of the packet tag. In the flooding method, the incoming packet is broadcast simultaneously over all free paths that lead to the destination such that the destination selects one of the incident copies. All other copies of the packet collide with each other and are removed from the switch fabric immediately, after the transmission of only a few bits. To implement this algorithm the switching elements in the first stage of the switch fabric would have to be modified to broadcast each incoming packet over all of the relevant paths. For any size of delta network modified to offer multiple paths in this way, all of the equivalent paths may always be selected from any switching element within the first stage of the switch fabric. Also there can never be more than $d/2$ equivalent paths to any single output port.

The Beneš Network

The delta network offers an acceptable performance for traffic which has a random destination distribution but its performance can be markedly impaired for incident traffic with a worst case distribution of destinations [159]. This is easily seen from fig. 5.3 as all connections between input ports 0 to 7 and output ports 0 to 7 share a single common link and likewise for ports 8 to 15 and so on. For this network the identity connection, in which every input port requests connection to every output

[1] $\lceil x \rceil$ signifies the smallest integer equal to or greater than x.

port of the same port number, represents the worst case traffic pattern. For this traffic pattern a total of only N/d connections may be established even though there is no contention for output ports. For some applications this sensitivity of the switch fabric to the destination distribution of the incident traffic may not be significant. For high performance switches, however, and in order to handle traffic sources which have an average bandwidth in excess of about 10% of the switch port bandwidth, extra stages of switching must be introduced to distribute the incident traffic across the switch fabric. These additional stages of switching are often termed the distribution fabric and the self-routing stages of the switch fabric are called the routing fabric. In order to fully distribute the incident traffic, with any arbitrary destination distribution, across an entire s stage delta network requires $s-1$ distribution stages and results in a Beneš topology. Thus the Beneš network is of interest because it offers increased throughput and also removes the sensitivity of the switch fabric to the destination distribution of the incident traffic. Due to the number of multiple paths through the Beneš network it may also increase the reliability of the switch fabric through fault tolerance.

A 64×64 Beneš network of switching elements of degree 8 is illustrated in fig. 5.5. For a network whose total size is an integer power of the degree of the switching elements, all switching elements will be of the same degree d and the number of equivalent paths through the network will be d^{s-1}, (where $s = \log_d N$). Beneš networks can be constructed to any size that is an integer power of 2 but if the size is not an integer power of the degree of the switching element the network will require switching elements of two different degrees. Refer back to fig. 4.9 which describes the construction of a general square three-stage network. The Beneš condition requires that $m = n$. Hence, for example, to construct a 16×16 Beneš network $n = m = 2$ and $r = 8$. If $n = m < r$ a Beneš network which requires the least hardware is constructed. Consider, however, the network in which $n = m = 8$ and $r = 2$ which also gives a 16×16 network that satisfies the Beneš condition. It may require more hardware but it is of interest because it can be formed by sub-equipping a Beneš network that is an integer power of the degree of the switching element. Thus all switching elements are of the same degree, idle switching element ports in the central stage of the network are disabled, and the first and final stages of the network are equipped with only N/d switching elements. This structure will be referred to as the sub-equipped Beneš network. A 32×32 sub-equipped Beneš network of 8×8 switching elements is illustrated in fig. 5.6.

To reduce the number of devices required in the sub-equipped Beneš network of fig. 5.6, the central stage might be formed from 8×8 switching elements, each configured as two 4×4 devices as a selectable option within the standard device. A 16×16 sub-equipped Beneš structure would require switching elements in the central stage configured as four 2×2 devices. A network that is very similar to the sub-equipped Beneš may be formed by reflecting the modified delta network, illustrated in fig. 5.4, about its central stage. The central switching stage of this network, however, cannot be formed easily from a single standard part for all sizes of network and it cannot offer a significantly greater performance than that of the sub-equipped

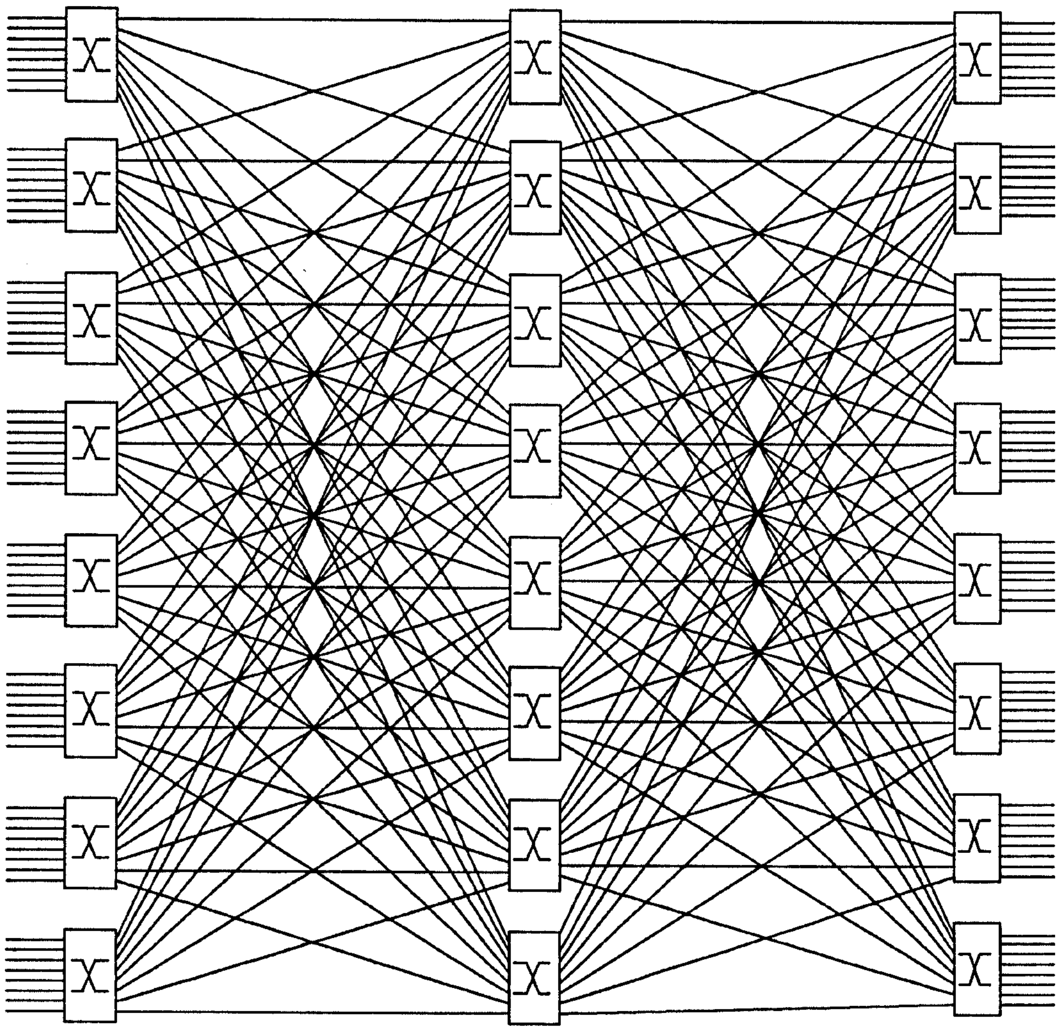

Figure 5.5: A 64×64 Beneš network of 8×8 switching elements.

Beneš structure.

The Beneš network introduces a large number of equivalent paths into the switch fabric. Once again, as there is no buffering within the switch fabric the path for each incident packet may be selected independently without fear of inducing out-of-sequence errors between packets belonging to the same virtual circuit. Three possible algorithms for selecting a path through the network have been investigated: searching, flooding and random. The searching and flooding algorithms operate in the same manner as for the delta network save that there are many more paths to search or flood. In the random algorithm the distribution stages of the switch are implemented with switching elements that select any free path to the succeeding stage at random.

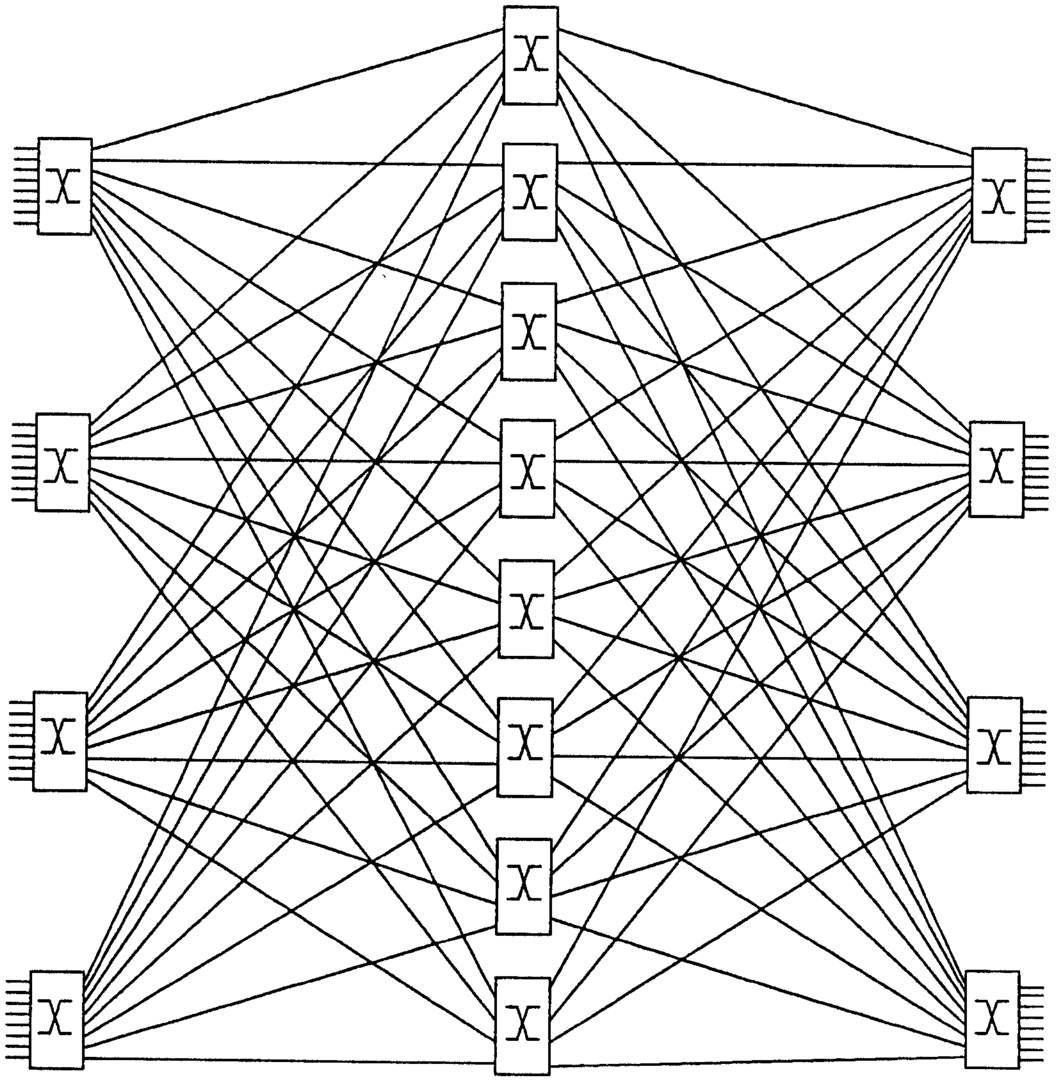

Figure 5.6: A 32×32 sub-equipped Beneš network of 8×8 switching elements.

5.5 The Multi-Plane Switch Structure

It is common practice in the design of a telecommunications switch to duplicate or even replicate the switch fabric and control hardware for reliability and ease of maintenance. If this is achieved in a load sharing manner the performance of the switch is also enhanced. The general structure of a two-plane switch is shown in fig. 5.7 and may be extended to form a multi-plane switch of any arbitrary number of planes. It consists of two identical switch planes, each switch plane being a complete delta network with or without a distribution fabric. The two switch planes are connected in parallel to form a load sharing arrangement [85, 82].

Once again multiple paths are being introduced into the switch fabric and either a

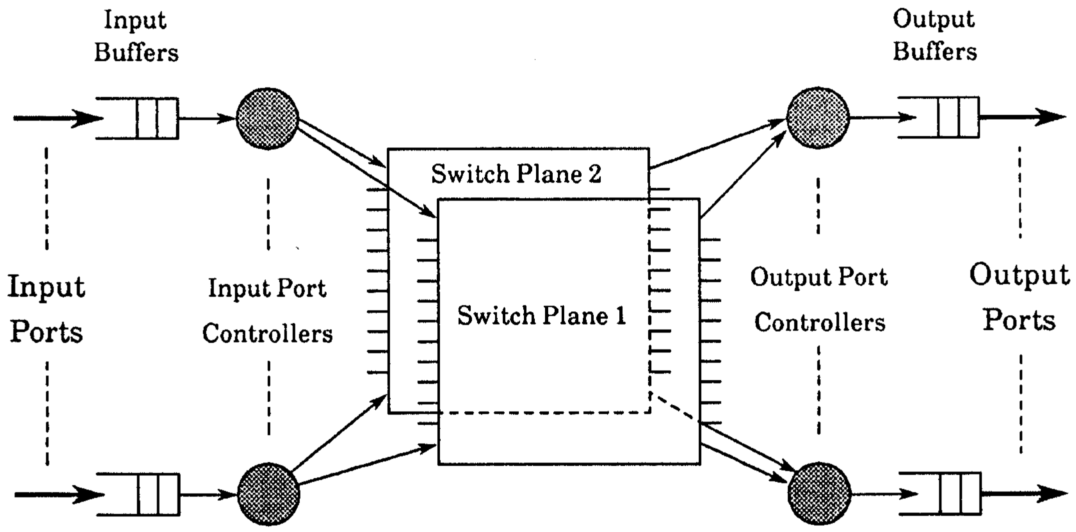

Figure 5.7: A two-plane switch structure.

searching or a flooding algorithm may be used to select a path across one of the switch planes. In the searching algorithm each plane is attempted in sequence until a path is established. For the flooding algorithm, the input port controller transmits a copy of the packet over each switch plane simultaneously and the output port controller selects one of the successful copies. All other copies will fail and will be removed from the switch fabric after the transmission of only a few bits.

A simple implementation of input port controller will only be capable of handling a single packet at once. Improved performance may be obtained if the input port controller is capable of transmitting multiple packets across separate switch planes simultaneously, at the expense of increased hardware complexity. The same is true for the output port controller. A simple implementation will only be able to handle a single packet at a time and thus must reject set-up attempts arriving across the free plane while it is busy serving an existing packet. A more complex output port controller will be capable of handling multiple packets arriving at the same time and buffering them in a first in first out manner in the output buffer. In this manner a measure of output buffering may be provided to increase the performance of the switch at the expense of a more complex output port controller.

5.6 Summary

For reasons of hardware complexity and flexibility of implementation, an input buffered switch design with a non-buffered switch fabric has been selected. For the same reasons the investigation of a blocking switch fabric, the delta network, has been proposed. To improve performance and to remove the sensitivity of the switch to the destination distribution of the incident traffic a switch fabric based upon a Beneš topology has been suggested. The majority of the existing work has concentrated on the use of 2×2 switching elements. Switching elements of higher degree will offer enhanced performance and reduce the number of interconnections required within the switch fabric which is a major factor limiting the maximum size of switch that may be implemented. Previous work suggests that non-buffered switching elements of up to degree 16 may be suitable for implementation in gate array technology.

To enhance the performance of the basic design, input queue by-pass and the use of multiple switch planes in parallel have been proposed. Output buffering across the multiple switch planes will further increase the performance as will the ability to transmit packets from the same input port across multiple switch planes simultaneously.