Chapter 8

Implementation of the Fast Packet Switch

The performance of the fast packet switch has been investigated by means of a simulation model. Confidence in the accuracy of the simulation results has been gained by comparison with a number of analytical models with which close agreement has been demonstrated. The error introduced into the simulation model has been investigated to yield upper and lower bounds upon the performance results. All of these results, however, assume the availability and performance of the crossbar switching element. This chapter presents an investigation of the design and implementation of an experimental crossbar switching element [114] and also of a simple input port controller from which a fast packet switch may be constructed. Measurements of the experimental implementation demonstrate that its performance is very close to that assumed by the simulation model. A discussion then follows of the various enhancements required and options available for the full-scale implementation of a fast packet switch.

8.1 An Experimental Implementation

To investigate the hardware realisation of a crossbar switching element and to demonstrate the simplicity of the design an implementation in gate array technology was undertaken. A low volume prototype gate array fabrication service was available in which an electron beam was used to write the final layer metal interconnection pattern directly onto a wafer of gate array devices. The target gate array family was the Texas Instruments TAHC series in 3 μm HCMOS of which the largest device, the TAHC10, was selected. This device contains the equivalent of 1120 two input NAND gates with 40 input/output buffers. Of the raw gates available a maximum of 896 could be accessed using the computer aided design software and of these, few designs could use much more than 50% due to the layout and routing restrictions. (With only a single layer of metal interconnection available it was in general necessary to sacrifice a large number of transistor cells in order to provide interconnections across

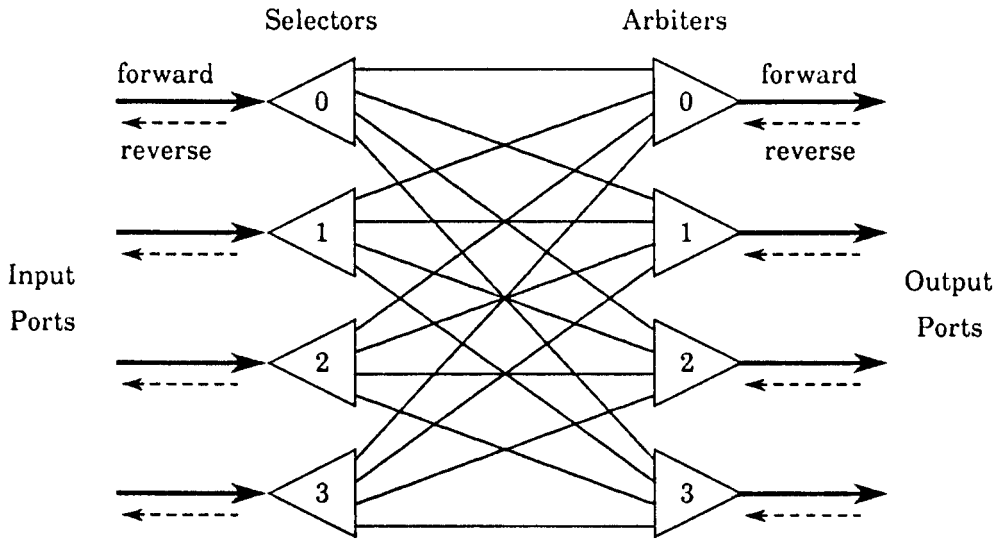

Figure 8.1: Structure of the 4×4 crossbar switching element.

the device.) The computer aided design software included a complete suite of design, layout, testing and fault location programs with an adequate library of basic logic elements, e.g. gates, flip-flops and latches. Two separate devices were constructed on separate TAHC10 gate arrays: a 4×4 crossbar switching element, and a single port input port controller with a general purpose 8 bit microprocessor bus interface.

8.2 The Switching Element

Although the simulation results indicated that 8×8 would be the preferred size of switching element the limited size of the gate array available restricted the implementation of the switching element to a 4×4 design. The complete structure of the 4×4 crossbar switching element, implemented on a single TAHC10 gate array, is illustrated in fig. 8.1. The structure consists of a set of identical selectors and a set of identical arbiters in a fully interconnected topology. Each selector and arbiter operates independently and asynchronously at the packet level but synchronously at the bit level with the use of a common system clock. Increasing the size of the switching element to 8×8 and beyond is merely a matter of increasing the size of the selectors and arbiters and interconnecting one selector per input port and one arbiter per output port in a completely interconnected topology. Crossbar switching elements of any integer dimension may be constructed in this manner and non-square, concentrating or expanding switches, are also possible.

Operation of the Switching Element

Each link of fig. 8.1 consists of two paths, a forward path which carries the traffic and a reverse path used to pass collision information back through the switch fabric. Each selector monitors an input port and when an idle to active transition is observed on the forward path it removes the first two digits from the tag at the head of the incident packet. These digits are used to select the required output port and the state of the relevant arbiter is inspected. If the arbiter is free, the remainder of the packet is passed over the forward channel to the output port. The reverse channel is also connected transparently from output port back to input port so that collision information from later stages in the switch fabric may be quickly transmitted back through the switch fabric to the input port controllers. If the arbiter indicates that the output port is busy, the selector asserts the collision (busy) signal on the reverse channel of the input port. The selector returns to the idle state when an active to idle transition is detected on the forward path.

The arbiter monitors the forward channels from each of the selectors. As soon as a channel goes active it is switched through to the output port and the reverse channel is connected transparently. The reverse channels to all other selectors are then set to the busy state. The arbiter returns to the idle state as soon as the forward path of the selected channel goes idle.

Implementation of the Selector

The selector is required to distinguish between an active and an idle forward channel. Normally this would be achieved using some form of line code but with a total budget for the switch of less than 500 gates, the selector must be implemented with less than 60 gates which leaves little spare logic to handle a line code. The device does however have an excess of input and output pins thus a simple solution was adopted for the forward path. The forward path was formed from two signals on separate I/O pins: the data signal which carried the information component and the active signal which defined when the data signal carried valid information. Each input (and output) port thus carried three signals: the data and active signals forming the forward path and the busy signal which was the reverse path.

The implementation of the selector is shown in detail in fig 8.2. When the incoming active signal is asserted the route selector latches the first two bits of the data signal. These are applied to the selector switches which connect the 'busy' and 'active' signals of the appropriate arbiter to the collision controller. The route selector then issues an active signal which causes the collision controller to examine the state of the incoming busy signal from the selected arbiter. If this signal indicates 'not-busy' it is connected transparently through to the outgoing busy signal of the input port and the collision controller asserts the active signal which propagates through the selector switches to the selected arbiter. If the incoming busy signal indicates 'busy' then the collision controller asserts the outgoing busy signal of the input port and keeps the outgoing active signal in the 'inactive' state. The selector is reset to the idle state when the

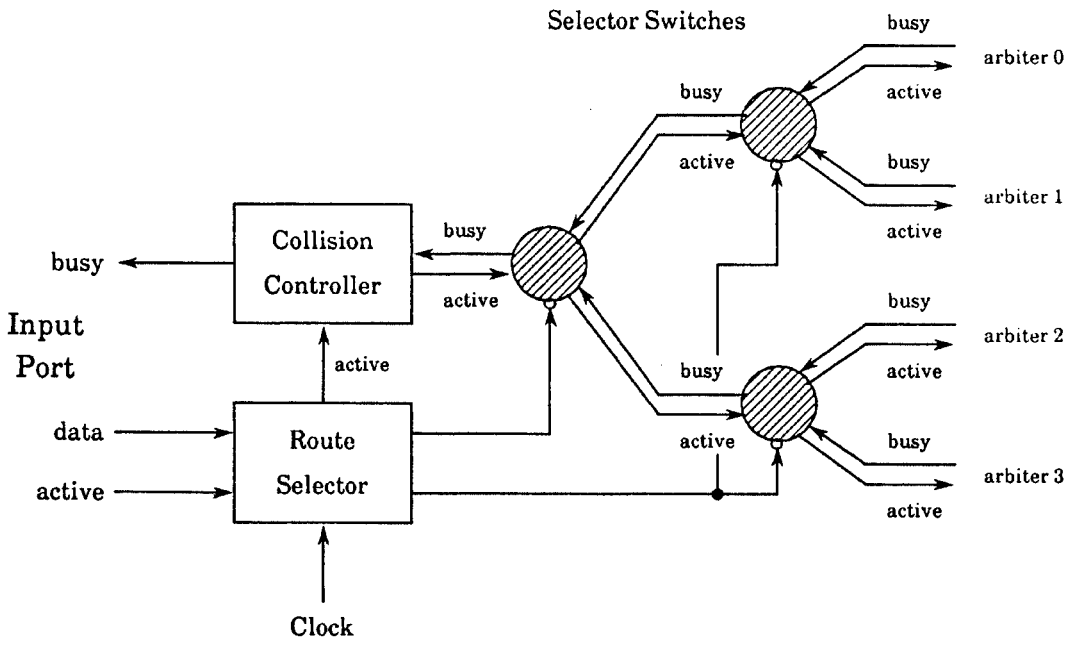

Figure 8.2: Implementation of a 1 to 4 selector.

incoming active signal of the input port indicates the idle condition. The complete 1 to 4 selector required a total of 42 gates.

Implementation of the Arbiter

The implementation of the arbiter is shown in detail in fig. 8.3. The active signals from each of the selectors are fed into a priority encoder. If multiple active signals are asserted during the same clock period the priority encoder selects one of them according to a simple priority scheme. The event is rare so a simple priority scheme should be sufficient otherwise a round robin or random scheme could be employed. When an incoming active signal is asserted and selected, the priority encoder sets the arbiter switches to connect the incoming data signal, corresponding to the selected port, to the retiming flip-flop on the data line of the output port. The priority encoder then issues the outgoing active signal which enables the connection of the incoming busy signal of the output port through the arbiter switches to the appropriate selector. The arbiter switches also set all other busy lines to the busy state which prohibits other input ports from attempting to access a busy arbiter. When the active signal of the selected input port drops to inactive, the outgoing active signal returns to the inactive state after a delay of one bit time. Thus the tail of the active signal propagates across the switch fabric together with the tail of the data signal. The arbiter is ready for access by another input port after a further delay of only one bit time. The complete 4 to 1 arbiter required a total of 50 gates.

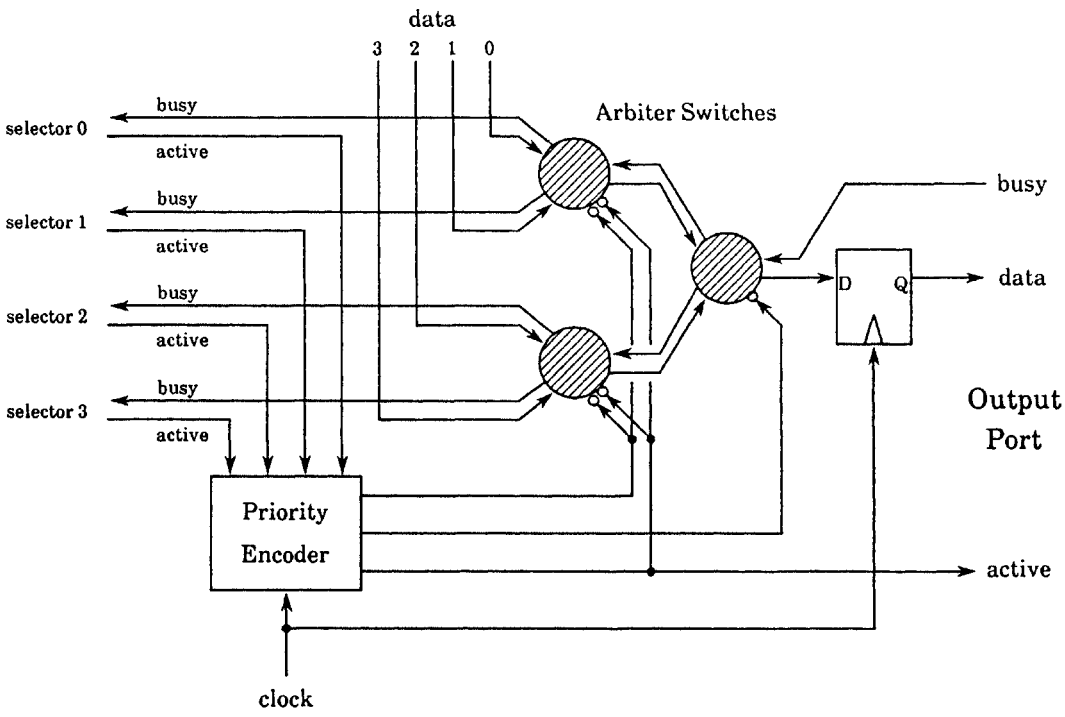

Figure 8.3: Implementation of a 4 to 1 arbiter.

Implementation of the Switching Element

The critical path in the set-up of a packet across the switching element is from the assertion of the active signal at the route selector to the selection of the appropriate data signal by the arbiter switches in time to be clocked by the output flip-flop. In the existing design this must occur within one bit time which limits the clock frequency of the experimental design to 8 MHz. The inclusion of one or two bits delay at the input ports of the switch would ease this restriction. In the existing implementation, each 4×4 switching element inserts a delay of only one bit time into the data path and the data path passes through no more than two gates and a flip-flop.

The complete 4×4 crossbar switching element required a total of 378 gates. This represents a 42% utilisation of the nominal size of the gate array which ought not to have caused any layout difficulties. Several difficulties were however encountered, most of them resulting from the fact that only a single layer of metal interconnection was available. This required interconnections in the vertical direction to proceed via polysilicon tunnels of which there were generally insufficient and the use of which increased the delay across the interconnections and required the sacrifice of transistor cells. The use of a gate array with two layers of metallisation would overcome these difficulties.

8.3 The Input Port Controller

The function of the input port controller of the experimental implementation was to demonstrate the functional operation of the switching element and to measure its performance. To achieve this it was not considered necessary to send real data packets across the switch but merely to set up a connection to the requested output port and to hold the connection for the length of a packet while sending a preset bit pattern across the connection. This greatly simplified the design of the experimental input port controller whilst allowing the functional operation of the crossbar switching element to be investigated using an oscilloscope and a logic analyser. To have attempted to transmit real data packets across the switching element would have required a much more complex experimental set-up and would have yielded no further results on the performance of the switching element than the simple arrangement to be described.

The experimental input port controller was implemented in a single TAHC10 gate array and controlled a single input port of the switching element. Four of these devices were therefore required to investigate the operation and performance of a single 4×4 switching element, one on each of the four input ports. A functional diagram of the experimental input port controller is given in fig. 8.4. The device is connected to the outside world via a standard 8 bit microprocessor bus interface and to the switching element via the three signals data, active and busy of the output port. Three status outputs indicate the state of the input port controller and their condition may also be read across the bus interface on the lowest three lines of the data bus. To initiate operation of the input port controller an 8 bit word is written to the bus interface, passed through the latch and into the recirculating shift register. The most significant bits define the required destination while the remaining bits form the bit pattern to be transferred across the data line. The active signal is asserted and the data transferred across the data switch to the output port. If the incoming busy line is asserted within 16 bit times the active signal is dropped and the sequencer waits for a retry delay of 32 bit times from the beginning of the set-up attempt before commencing the next set-up attempt. If no busy signal is received within 16 bit times of the beginning of a set-up attempt the path set status bit is established and the path held for a total packet length of 256 bits including the routing tag. On completion of packet transmission the input port remains idle for 16 bit times to prevent an input port with a large number of packets to the same destination from excluding other input ports from obtaining access to that output port.

A second mode of operation was added in which a path once set remains held until released by a specific instruction received across the bus interface. In this mode the data path, once established, is switched by the data switch to connect an external data signal to the output port. This permits packet data to be transmitted across the switching element using an external framing and line coding device such as an HDLC chip. It also permits the operation of the switching element to be slowed down for observation. The experimental input port controller required a total of 292 gates and presented no implementation or layout difficulties on the TAHC10 gate array.

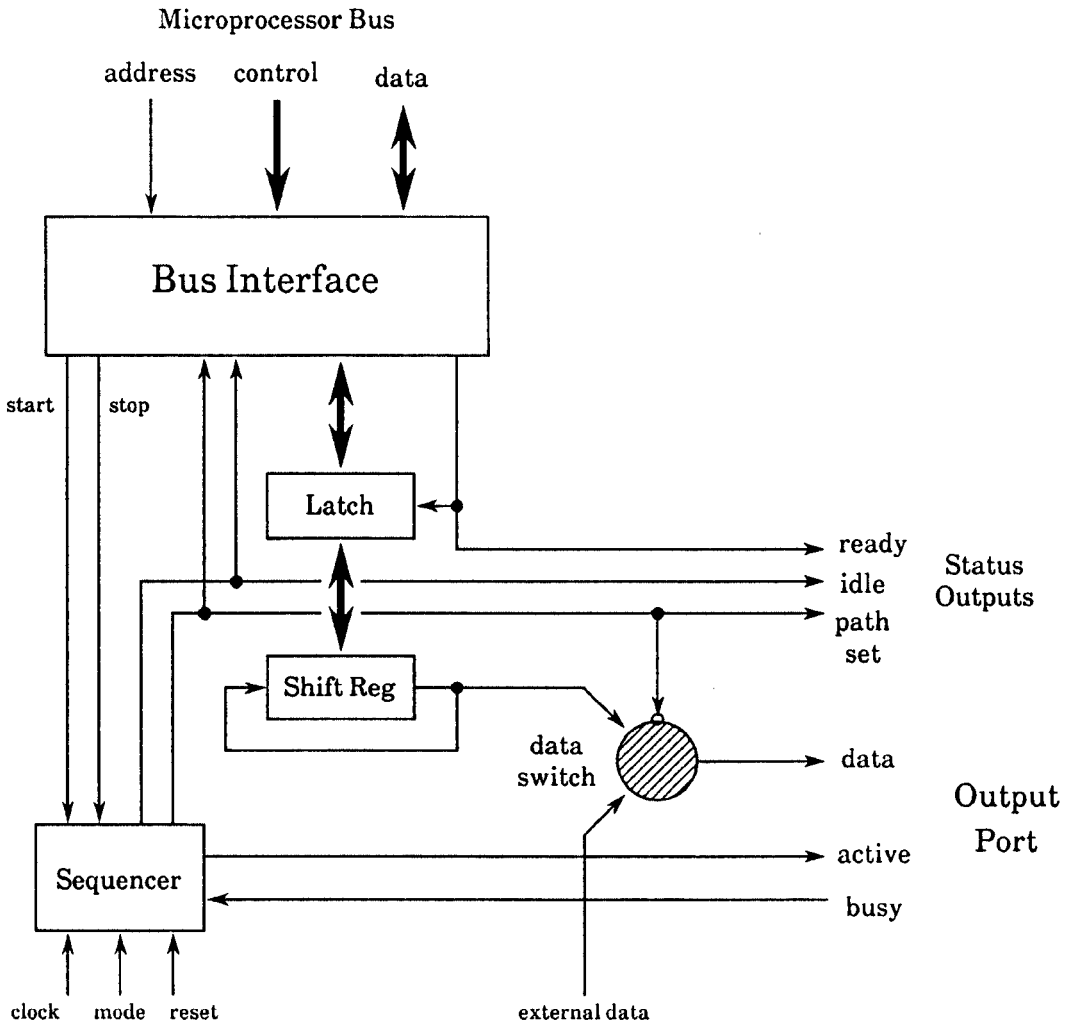

Figure 8.4: The experimental input port controller.

8.4 Performance Measurements

Each of the four input ports of a 4×4 crossbar switching element was connected to an experimental input port controller which were in turn connected to the system bus of a microcomputer. The microcomputer could thus request the transmission of test packets from any input port to any output port and inspect the status of each of the input port controllers. The busy signals of the output ports of the switching element were also interfaced to the system bus so that each output port could be enabled or disabled. The functional operation of the switching element was investigated using an oscilloscope and a logic analyser in the test packet mode and by transferring external data patterns across the switching element. The measured operation of the switching element was in agreement with that predicted by the logic level simulator of the CAD software at a clock rate of up to 8 MHz. The input port controller from input port

number 3 of the switching element was disconnected and that input port connected to output port number 3 of the switching element. This allowed test packets to be routed through two switch stages in cascade demonstrating that a four bit routing tag was correctly interpreted and indicating that the routing mechanism would operate successfully with an arbitrary number of stages of switching elements in cascade.

To measure the throughput at saturation performance of the switching element the system clock was decreased until the microcomputer was capable of keeping all input ports saturated with test packet requests with a uniform random distribution of destination tags. The measurement was repeated eight times for a total of 200,000 packets each to yield a throughput at saturation result of 0.6204 ± 0.0004. The same measurement was taken using the simulation model set up to simulate the characteristics of the 4×4 crossbar switching element. The simulation model gave a throughput at saturation of 0.6160 ± 0.0001 which differs from the measured value by 0.7%. This difference is of the same order of magnitude as that between the analytical [76] and simulation results for the 4×4 crossbar switch. It originates from the slight non-uniformity of the random number generators used.

The active line of one of the input ports was connected to a counter so that the average number of retry attempts per packet could be measured for a switching element at saturation. This figure could also be calculated from the number of packets generated within the period of measurement. The measurement of the average number of retries per packet yielded a value of 4.30 ± 0.02 which agreed with the calculated value to within 0.25% demonstrating that the switching element was indeed operating at saturation and behaving as predicted. The measurement also agrees closely with the result of the simulation model for the average number of retries per packet at saturation, to within 0.5%. This, however, is hardly surprising as the average number of retries per packet at saturation is closely related to the throughput at saturation.

8.5 Towards a Full-Scale Switch Implementation

The experimental switch implementation has demonstrated that a crossbar switching element may be constructed in current gate array technology using very few gates and also that its performance agrees closely with that predicted by the simulation model. The developments required to construct a full-scale implementation of the fast packet switch, based upon the results of the experimental model, will now be discussed.

The basic structure of the fast packet switch configured for general purpose communications applications is shown in fig. 8.5. The separate input and output port controllers of every k^{th} input and output port of the switch fabric have been combined to form I/O port controllers which handle full-duplex ports. The overall structure closely resembles a PABX or a current telecommunications circuit switch consisting of a control plane, a switch plane and some line cards.

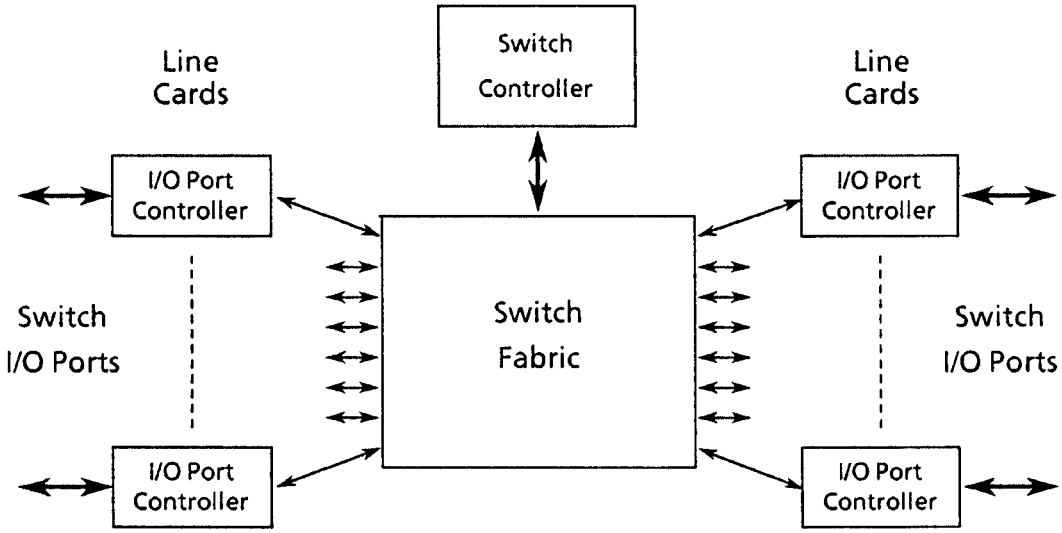

Figure 8.5: Structure of a fast packet switch implementation.

The Line Card

Each line card contains one or more I/O port controllers, the basic structure of which is given in fig. 8.6. Packets arrive in the input FIFO with a label at the head of the packet. This label is here referred to as the virtual circuit indicator (VCI) as it identifies the virtual circuit to which the packet belongs. The VCI of the packet at the head of the FIFO is latched and used to address a memory called the map. The map contains a replacement VCI for the next stage of the virtual circuit and thus maps the incoming VCI address onto the outgoing VCI address space. It also contains a tag which identifies the required output port of the switch. It may also contain information such as the priority of the connection, the type of traffic associated with the connection and details of a reverse connection, this information being available to the control hardware. The tag is prefixed to the packet, the new VCI replaces the old, and the packet is launched into the switch fabric. If the packet set-up attempt fails, the retry counter is incremented which may cause the most significant digits of the tag to change if the input port controller is searching through multiple paths. The contents of the map are updated by the switch controller possibly by the use of a separate control bus. In a large switch this method of control would be cumbersome thus an alternative is to define a special VCI that would allow the switch controller to update the map of each I/O port controller via special packets transmitted across the switch fabric.

The line code and packet framing function has been indicated in fig. 8.6 both on the I/O side of the port controller and also on the switch fabric side. The very different requirements on either side of the I/O port controller would almost certainly ensure that different line codes were used. If the switch port were interfaced to a local or a metropolitan area network the line code would be defined by the network and the switch interfaced to the network at the input and output FIFOs. If transmission

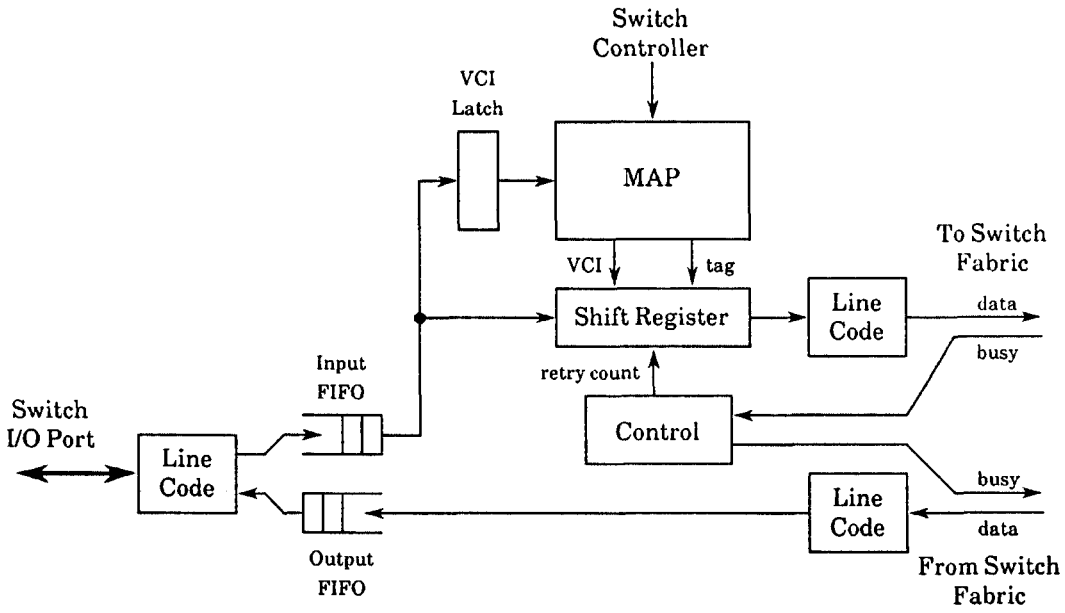

Figure 8.6: The I/O port controller.

across a point to point link is required three options are available for the line code:

- The packet may be delimited by a special bit pattern and the packet 'bit-stuffed' to avoid the occurrence of the special pattern within the packet, e.g. HDLC.
- Each N bit word may be coded using N+1 bits. The extra bit may be used to indicate the start and continuation of the packet, as in [100], or to provide a menu of special symbols as in FDDI [129].
- An alternative for fixed length packets is to divide the line into packet slots and to transmit a synchronisation pattern in the unused slots as proposed in the Prelude fast packet switch [141, 31], or to support the slotted structure in a conventional TDM frame.

It is probable that some form of error checking will be required on a per packet basis. The most suitable location for including this function is on entry to, or exit from, the input FIFO of the I/O port controller and not within every switching element within the switch fabric. Two levels of error checking may prove useful, one which protects the packet header and one for the information field. A packet with an error in the header should be discarded but if the error is in the information field it should be marked so that the destination may decide what action to take. Some classes of traffic are insensitive to occasional random errors but other classes will require retransmission of the packet.

If input queue by-pass is required to enhance the performance of the switch it must be implemented in the I/O port controller. It is possible, by the use of input

Size	*Gate Count*
2×2	250
4×4	600
8×8	1900
16×16	6000
32×32	21000

Table 8.1: Estimated complexity of crossbar switching elements.

queue by-pass, for packets arriving on the same virtual circuit to be delivered out of sequence. It may be possible to reconstruct the original sequence by the use of an end-to-end sequence number but this may prove undesirable. A simple way to avoid out of sequence errors would be to restrict the queue by-pass algorithm to just the first two packets on the queue that are routed to different ports. According to [100] this would offer about half of the performance improvement compared to applying the by-pass algorithm to all of the packets on the input queue. One possible method of implementing the full queue by-pass algorithm, while avoiding out of sequence errors, is to use a memory with one bit representing each of the output ports of the switch. The algorithm starts with the packet at the head of the queue and clears the memory. At every consecutive unsuccessful packet set-up attempt a note of the busy output port number is made in the memory and further attempts to that port are prohibited. The algorithm returns to the head of the queue and clears the memory on a successful packet transmission or on reaching the end of the input queue.

The Switch Fabric

The experimental model employed a very simple method of line coding and packet framing within each switching element. It required three signals for every link across the switch fabric, two forward and one reverse. While this may be acceptable for small switches it would in general be better to reduce the connections across the switch fabric to a single forward signal with a single reverse signal. This may be realised with a simple line code on the forward path in which a start bit is prefixed to the front of the tag at the head of the packet and the information field 'bit-stuffed' so that no more than N consecutive zeros occur. Whilst removing the relevant digits from the tag each switching element regenerates the start bit at the head of the packet. After the passage of the minimum packet length each switching element searches for a sequence of N+1 consecutive zeros to indicate the end of the packet. Assuming the use of a simple line code such as the above, an estimate of the complexity of the various possible sizes of crossbar switching element is presented in table 8.1 based upon the results of the experimental model. It is clear from the table that crossbar switching elements of up to size 16×16 may be fabricated in gate array technology.

For large sizes of switch, the use of two signals for every link in the switch fabric will be undesirable due to the constraints of interconnection and switch fabric par-

Technology	*Bandwidth per Port*
3 μm CMOS	10 Mbits/sec
2 μm CMOS	50 Mbits/sec
BiCMOS	250 Mbits/sec
ECL	500 Mbits/sec
GaAs	1 Gbit/sec
Photonic	$>$ 1 Gbit/sec

Table 8.2: Approximate maximum bandwidth per switch port for various implementation technologies.

titioning. The switch fabric may be constructed with a single connection for every link if it is operated in both directions in half-duplex mode. At the beginning of a set-up attempt all free switching elements in the path pass the packet tag forwards. When the tag has passed, all switches in the path pass the acknowledgement signal back across the path in the reverse direction. If the signal indicates collision the switching elements in the path are released. If the acknowledgement signal indicates successful connection to the output port, the switching elements pass the packet in the forward direction across the switch fabric. This 'ping-pong' packet set-up cycle slightly increases the time spent in packet set-up and may thus increase the internal blocking of the switch fabric but this impairment is likely to be outweighed by the reduction in the interconnections required within the switch fabric.

The internal design of the switching element presented here uses a large number of paths in a fully connected topology with each path operating in serial mode at the full speed of the switch fabric. The alternative, proposed in a number of fast packet switch designs, is to group the serial bit stream arriving at each switch port into word parallel form on entry to the switching element and to use some form of shared medium interconnection mechanism within the switching element. This allows the switching element to operate more slowly, at the word rate rather than at the bit rate of the switch fabric, but requires a more complex design.

The design of switching element presented in this chapter may be implemented in current 2 μm CMOS to achieve speeds of around 50 MHz. Beyond this speed, the simplicity of the design requires only the data path to operate at full speed. The majority of the logic in the switching element handles packet set-up and if a small increase in overhead is permitted in the packet set-up time then this logic can operate at a slower speed than that within the data path. Thus implementation may be possible in BiCMOS in which the data paths use bipolar technology while the packet set-up logic uses CMOS. Likewise for ECL, only the data paths will be required to operate at high speed thus reducing the power dissipation of the majority of the logic. Considering implementation in GaAs technology, [150] gives details of a device of 2200 gates with a power dissipation of 600 mWatts operating at 900 MHz, which is of the same dimensions as that required by an 8×8 switching element. A table of the approximate operating speed, and thus the maximum bandwidth per

switch port, is given for various technologies in table 8.2.

The design of the switching element may lend itself well to wafer scale integration [21]. Rows of selectors and arbiters may be located and interconnected on a single wafer of silicon to form a very large switch fabric. Care would have to be taken in the design to ensure that the effect of faulty nodes was localised and that sufficient redundancy was provided to cope with both faulty nodes and I/O pads. If sufficient multiple paths were provided across the switch fabric it would be possible to exploit the inherent fault tolerance of the design. Care would also have to be taken in the distribution of the clock signal. Using wafer scale technology it may be possible to construct switch fabrics of up to 1000 switch ports on a single wafer.

If the required operating speed of all switch ports is less than the maximum operating speed of the switch fabric then blocking within the switch fabric may be reduced by operating it at a higher speed than the switch ports. In this case buffering will be required at both input and output ports to dissociate the speed of the switch fabric from that of the switch ports. It will also be necessary to completely receive a packet in the input buffer before transmitting it across the switch fabric but this will be of no significance if the packets are short. If only a few ports are required to operate at very high speed, e.g. those terminating high speed trunks, then the high speed lines may be shared between a number of switch fabric ports. Packets from the high speed line may be served by any input port controller of the group and switching elements in the last stage of the switch fabric may be modified to direct a packet to any free output port controller of the group. The use of this technique is liable to cause out of sequence errors between packets travelling across the same virtual circuit.

The Switch Controller

The major function of the switch controller is to participate in the signalling protocol of a network of fast packet switches, to set up and clear down connections across the switch and to update the connection tables (maps) within each of the I/O port controllers. If a congestion control algorithm is to be implemented within a network of fast packet switches then the switch controller may be required to participate in congestion control. The switch controller will also be required to undertake maintenance and fault location functions. It may also be required to support a simple datagram service. A datagram service may easily be implemented on top of a fast packet switched network by using permanent virtual circuits between the switch controllers of the network to handle datagram transfer. If the datagram traffic load were to become too heavy for the switch controller it could be transferred to special purpose datagram servers.

8.6 Summary

The implementation of a 4×4 crossbar switching element in 3 μm HCMOS gate array technology has been described in detail together with an experimental input port controller in the same technology. The operation of the switching element has been measured and its throughput at saturation performance shown to agree with that predicted by the simulation model to within 1%. Thus confidence in the simulation results for larger switch structures is increased. Due to design and implementation technology limitations the operating speed of the 4×4 switching element was restricted to 8 MHz. Improvements to the design have been suggested which should enable operation at 50 MHz in 2 μm CMOS without great difficulty. Implementation in higher speed technologies has also been discussed.

Various possible developments to the basic design have been discussed to construct a full-scale switch for use in communications applications. There are some applications that call for a stand-alone switch, such as a high capacity bridge between local and/or metropolitan area networks, but many applications require fast packet switches to be interconnected to form a network. Thus the issues of access control, flow control, high speed protocol and congestion control within a network of fast packet switches become essential areas for further study.