Department of Applied Analysis
and Computer Science

Technical Report CS-73-09

November 26, 1972

SWITCHING MATRICES FOR PROGRAMMABLE TIME-DIVISION MULTIPLEXING

by

Eric G. Manning, W.M. Gentleman
C.E. Køhn and D.E. Morgan
University of Waterloo

Faculty of Mathematics

University of Waterloo

Waterloo, Ontario

Canada

Department of Applied Analysis

&

Computer Science

Department of Applied Analysis
and Computer Science

Technical Report CS-73-09

November 26, 1972

SWITCHING MATRICES FOR PROGRAMMABLE TIME-DIVISION MULTIPLEXING

by

Eric G. Manning, W.M. Gentleman
C.E. Køhn and D.E. Morgan
University of Waterloo

This report in the University of Waterloo Computer Science Departmental Series is also available under separate cover, for the benefit of readers in the Computer Communications area who do not receive the Computer Science Series.

I INTRODUCTION

This paper discusses one aspect of Programmable Time-Division Multiplexing (PTDM), a proposed new switching discipline for data communications. PTDM is a generalization of packet switching and circuit switching, and is intended for future data networks which share facilities (transmission links, local distribution facilities and floor space) with public telephone networks. The shared transmission links are expected to be digital links using Time-Division Multiplexing (TDM) with fixed frame formats[1], arranged in the present hierarchical topology. The motivations for developing such a switching discipline have been described by Manning[2] and are briefly summarized in Section II.

In this paper, the problem of designing switching matrices for the intermediate and higher levels of such a data network is studied. (Local distribution problems are not considered.) Two designs are presented, and each is evaluated with respect to cost and performance.

II THE PROBLEM

Data traffic is growing rapidly[3], and the majority of this traffic is carried on public telephone and telegraph networks. These networks were not designed for data traffic, and well-known problems of connection times, error rates and costs have arisen. Moreover, there are at least two different types of data traffic - transaction traffic and file traffic. Transaction traffic[4] is characterized by small volumes of data per call, high volumes of calls, a requirement for short connection times, and bursty use of transmission links. File traffic is characterized by large volumes of data, reduced need for short connection times, and smooth use of links.

Transaction traffic is generated by interactive terminals (for example, credit-card checking, seat reservations and general-purpose time-sharing), whereas file traffic is typically generated by computers or remote batch terminals.

The different needs of these two traffic types have given rise to two fundamentally different switching disciplines. Common carriers have developed circuit switches using space-division switching and electro-mechanical crosspoints; these are reasonably adequate for file traffic.* Also, packet switches which use mini-computers to multiplex short messages onto high-speed lines have been developed, and these are well-suited to transaction traffic.+ A present trend is for common carriers to offer both types of service via functionally separate networks. This is an adequate short-term solution to the problem but it possesses serious long-term defects. It is expensive to design, build, and maintain two networks which are functionally separate (although they may share physical facilities). Also, new types of traffic may well appear, which are neither transaction-like nor file-like in their switching needs. Hence a single data network, able to cater to transaction traffic, file traffic or intermediate traffic types, at the user's request, would be a better long-term solution. The concept of PTDM switching is aimed at providing such a generalized data network.

---

* The Trans-Canada Telephone System's MULTICOM service[1] provides one example.

+ Packet switching was developed by Baran[5], Roberts et. al.[6] and by Davies et. al.[7] The service proposed by Packet Communications Ltd. provides an example of a public packet-switched service.

III ASSUMPTIONS

The following assumptions are made;

a) Digital transmission links employing fixed frame formats will be common in future public voice networks. (The T-1, T-2 and T-4 links of AT&T[8] provide examples). These links will be globally synchronized to hold frame slippage to an acceptable level.

b) For economic reasons, public data networks should share physical facilities - notably trunks - with the switched voice network. This assumption will tend to impose the tree-like or hierarchical structure of the switched voice network on public data networks as well.

c) Packet switching may not be a feasible discipline for future public data networks. The CPU of a packet switch must actively process each character of the data transmitted, and this creates problems of switch saturation. Also, it is not clear that packet-switched networks can be expanded to contain several thousand switches.*

d) Time-Division multiplexed switching of digital lines is now feasible. This is demonstrated by the announcement of the No. 4 Electronic Switching System by Bell Telephone Laboratories[9].

e) The control CPU of a feasible switch must not be required to process data. In order to prevent CPU saturation under high transmission rates, the CPU should only intervene in the processing of state changes (set-up, teardown or bandwidth change of calls).

f) Hardware costs are as stated in Appendix A.

To summarize, we assume that the switched voice network will provide frame-formatted digital transmission links arranged in a hierarchical or

---

* There are approximately 8500 switches in the U.S. Bell System.

tree-like topology. There are compelling economic motivations to exploit these facilities to provide a functionally separate but physically overlapped data network, able to cope with a wide range of traffic types. The purpose of this paper is to propose a new switching discipline aimed at this goal.

Extensive research has been done on problems of data switching; work on loop structures[4] and Asynchronous Time-Division Multiplexing[10] (ATDM) is of particular interest here. Loop techniques are well-suited to carriage of bursty traffic and can be used in conjunction with fixed-length frame formats[11]. Moreover, the problem of reliability in a multi-loop hierarchical network may not be as serious as is often supposed[12]. However, the topology of a multi-loop network does not resemble the tree-like topology of the switched voice network, and this would complicate the sharing of trunks between the two networks. We therefore believe that local distribution is the most promising area of application for loops in public networks.

ATDM or statistical multiplexing is an elegant technique which has been studied extensively (10,13,14,15). The problem solved by ATDM is that of efficiently multiplexing the traffic flow between a computer and a collection of remote terminals onto a single transmission link, rather than the efficient use of the higher-level links of a tree-like public network. ATDM and PTDM are both multiplexor-concentrator schemes, however, ATDM is a message-with-address scheme using variable-length frame formats. Also, the design of an efficient ATDM scheme depends heavily on the scheduling algorithm used by the attached computer. For all of these reasons we support Doll's comment[15], that "ATDM's primary utility will be found, not in replacing Synchronous Time Division Multiplexing en masse, but in totally new kinds of applications...".

It is therefore clear that PTDM, loop structures and ATDM are complementary rather than competitive techniques. We now discuss the functional specification of a PTDM network followed by an analysis of PTDM switching matrices.

IV FUNCTIONAL SPECIFICATION OF PTDM NETWORK

This section describes the services to be offered by a network of PTDM switches, from the user's point of view. The central concept is that a user can negotiate with the network to obtain the transmission bandwidth desired. Moreover, the user can request changes in the bandwidth assigned to his call as the call progresses. This capability implies packet-switched performance and circuit-switched performance as special cases, thus meeting the objective of a generalized data network.

To obtain circuit-switched performance, a calling party A asks for a connection to a called party B at a fixed bandwidth. After the network grants the request a file is sent and the call is terminated. To obtain packet-switched performance, A asks for a connection to B at a relatively high bandwidth. After the request is granted A sends a packet followed immediately by a command to reduce the assigned bandwidth to zero.* This mode of operation gives A the exclusive use of a high-bandwidth channel for short periods and is therefore functionally identical to packet switching.

---

*A packet souce such as a transaction terminal would not be connected directly to a PTDM switch. Rather, a large number of such sources would be multiplexed onto a single frame-formatted line by a serving central office, and the aggregate traffic would feed a PTDM switch. Hence the PTDM switches would respond to statistical averages of such commands, not individual instances.

PTDM also has sufficient flexibility to handle traffic types which have not yet emerged, as the following hypothetical example shows. Consider interprocess communication between process A_2 of machine A and process B_{26} of machine B, which might proceed as follows. Initially, A_2 asks the network for a connection to B at low bandwidth - perhaps 2400 baud. The supervisory process B_1 of B answers the call and advises that B_{26} will be run in twenty minutes. A_2 then reduces the assigned bandwidth to zero and suspends itself. When B_{26} enters the running state it increases the bandwidth and signals the supervisory process A_1 of A. A_1 awakens A_2 and interprocess communication between A_2 and B_{26} begins. The call bandwidth can now be arbitrarily varied to optimally meet the communication needs of the inter-process communication. File transfer, for example, can be done at high bandwidth whereas control signalling or protocol can be done at lower bandwidth.

In summary, note that the bandwidth requirement varies dynamically; that the call resembles neither file traffic nor transaction traffic; and that the PTDM discipline would allow optimal resource allocation for the purpose. Finally, it is clear that this type of bandwidth allocation is not readily available from existing public switched networks.

V A PARALLEL SWITCHING MATRIX

A. Design

A design for a parallel switching matrix is given in Figure V.1. The incoming frames are F bits wide* and arrive at

* Please see Appendix B for the definition of notation.

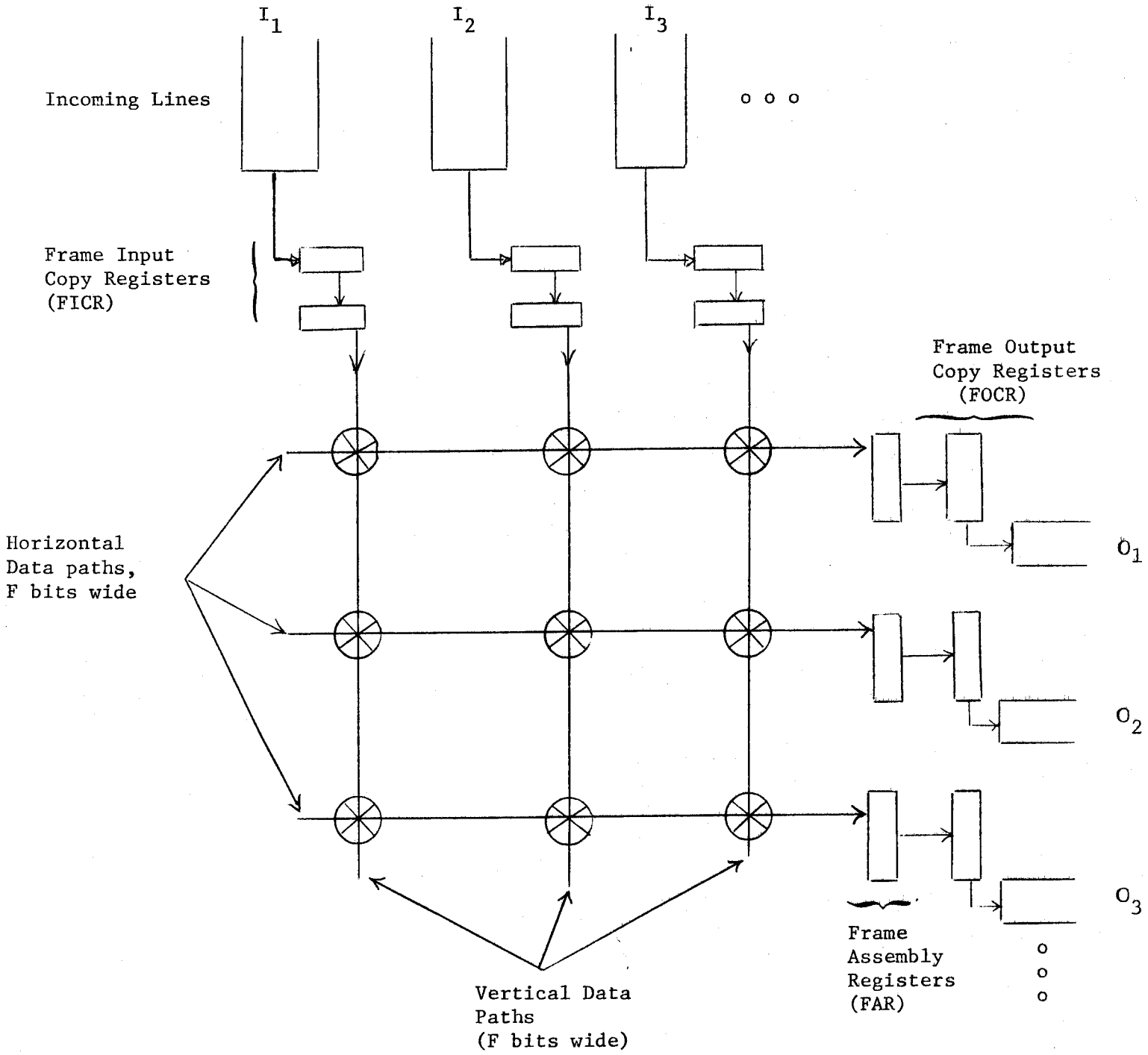

Figure V. I - Parallel Switching Matrix

a rate of R per unit time. Each incoming line is terminated in a swinging pair of shift registers, called Frame Input Copy Registers (FICR), which are F bits wide. The crosspoint array is able to gate bit j of FICR(i) to bit ℓ of FAR(k), for all i,j,k and ℓ. Each intersection of a horizontal and vertical line comprises a crosspoint M(i,j,k,ℓ) having a one-bit memory. Hence,

FAR(k,ℓ) <- FICR(i,j) implies

M(i,j,k,ℓ) = 1.

The control processor (not shown) can read and write the crosspoint M-bits and can therefore set up and tear down switching paths through the crosspoint array.

B. Operation

As a frame arrives on line I_i, it is shifted into one of the pair of registers FICR(i). Detection of the framing pulse causes the pair FICR(i) to be swapped, providing an empty register to receive the next frame. The full register is connected to the crosspoint array, and its contents are gated through the array to the appropriate FARs. This process happens concurrently on all input lines. Periodically, at rate R, a master output event occurs which copies all FARs to the corresponding FOCRs, and output begins on the outgoing lines. Frame slippage due to timing variations between input lines is held to an acceptable level due to assumption (1), together with double buffering on inputs. Moreover, CPU intervention only occurs when the values $\{M(i,j,k,\ell)\}$ must be altered, i.e. when the bandwidth of a call has to be changed.

C. Cost/Performance of the Parallel Matrix

To assist in evaluating costs and performance we study a specific example, namely

$$F = 200$$

$$I = O = 10$$

$$T = 133\ \mu sec$$

which closely corresponds to a switch serving 10 Bell System T-1 lines incoming and outgoing.

Assuming the component characteristics of Appendix A and standard designs for the shift registers and crosspoints, the switching time T_{sw}^{*} is about 150 nsec. The constraint on T_{sw} is

$$T_{sw} < T = 133\ \mu sec.$$

Hence the parallel matrix is "over-designed".

Moreover, about 10^6 Dual In-Line Packs would be required for the Parallel Matrix studied, implying 7×10^3 sq. ft. of board space and 10^5 watts of D.C. power. Hence the Parallel Matrix is not a feasible design, and is useful only as a conceptual model of the PTDM switching discipline.

* defined as the interval between arrival of a frame in FICR(i), and the beginning of transmission from FOCR(k).

VI A SERIAL SWITCHING MATRIX

A. Switching Primitives

It is useful to decompose the PTDM switching process into two fundamental operations or primitives:

1) ROUTE selected bits of an incoming frame to the appropriate output lines, keeping bit positions in the frame (time slot assignments) constant;
2) PERMUTE the bits of a frame to achieve the required outgoing time-slot assignments.

Informally, the ROUTE or R-primitive operates on an incoming frame, breaks it into partial frames* and assigns them to outgoing lines as illustrated in Figure VI.I. Hence the R-primitive specifies a form of space-division switching. The PERMUTE or P-primitive operates on a partial frame and permutes its bits appropriately. Finally, the set of permuted partial frames associated with each outgoing line are merged to create the outgoing frame.

More formally,

$$O(k) \leftarrow \bigcup_i P(i,k) \cdot \left[R(i,k) \cdot I(i) \right] \qquad 5.1$$

describes the serial switching process, where

R(i,k) denotes the routing operator applied to the input frame I(i);

P(i,k) denotes the permutation operator applied to R(i,k)·I(i); and

O(k) denotes the frame to be output on the k^{th} output line.

Substituting register names yields

---

* A partial frame is an F-bit string containing one subframe of the incoming frame; the remainder of the bits have no significance.

$$FAR(k) \leftarrow \bigcup_i P(i,k) \cdot \left[R(i,k) \cdot FICR(i)\right] \qquad 5.2$$

$$FAR(k) \leftarrow \bigcup_i P(i,k) \cdot PFR(i,k) \qquad 5.3$$

Here, Frame Assembly Register (k) is an F-bit register which holds outgoing frames before transmission on O(k); Partial Frame Register (i,k) is an F-bit register which holds the contribution of I(i) to O(k). Also, note that the operators U and P are not commutative; partial frames must be permuted before they can be merged in order to prevent conflict in bit locations.

The above material is equally applicable to conventional TDM switching and to PTDM. However, PTDM imposes an additional design constraint: It must be possible to easily redefine the R and P-primitives realized by the matrix, due to the frequent re-definition of these primitives which the PTDM discipline implies.* We refer to this requirement as controllability.

B. Implementation of the R-primitive

An implementation of the R-primitive is given in Figure V1.2. All registers are F-bit shift registers and the data paths are one bit wide. The crosspoints are controlled by XPTR registers whose contents are defined by

$$XPTR(i,k;j) = 1 \text{ iff} \qquad 5.4$$

bit j of FICR(i) is to be routed to PFR(i,k). Thus the content of XPTR (i,k) at time t defines the R-primitive R(i,k) implemented at time t.

---

*Indeed, the need for controllability as defined here is the only major additional design requirement which PTDM imposes, over and above the requirements for conventional TDM. The latter we assumed to have been solved, by Assumption (d).

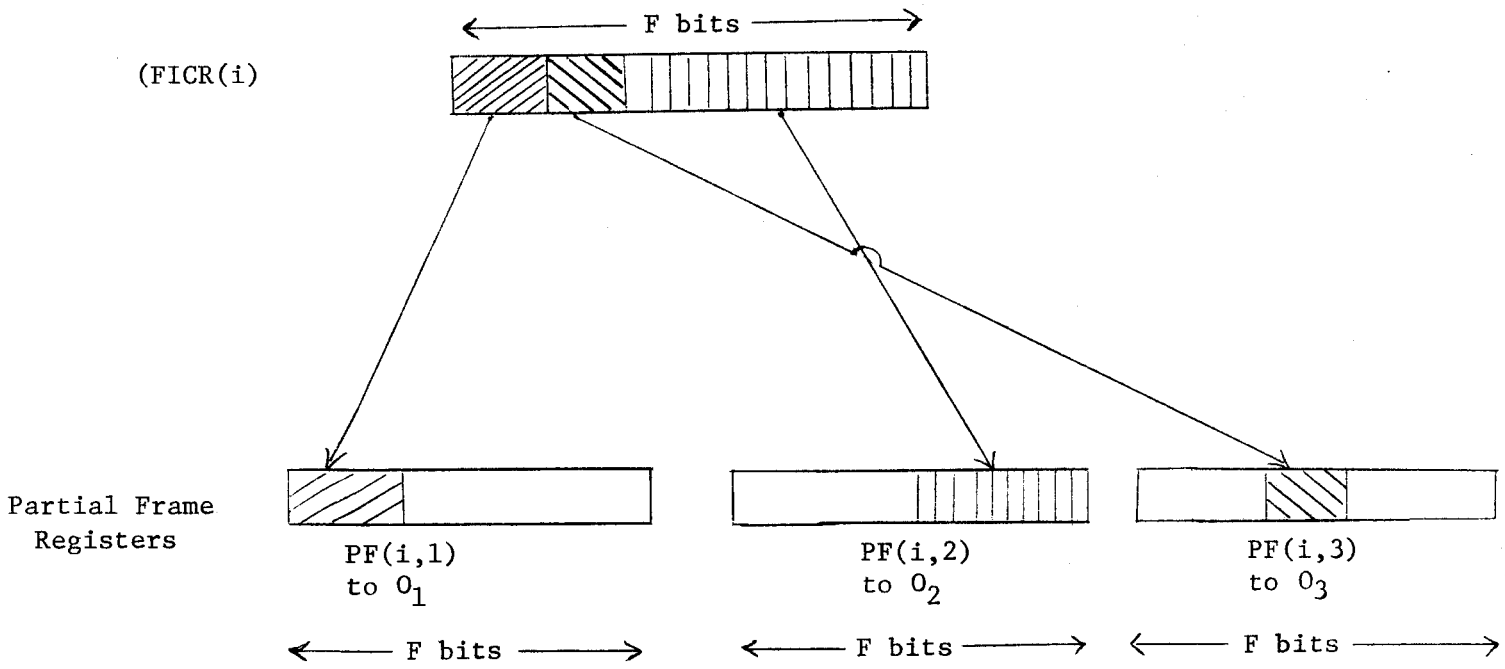

Figure VI.1 - Example of the ROUTE Primitive

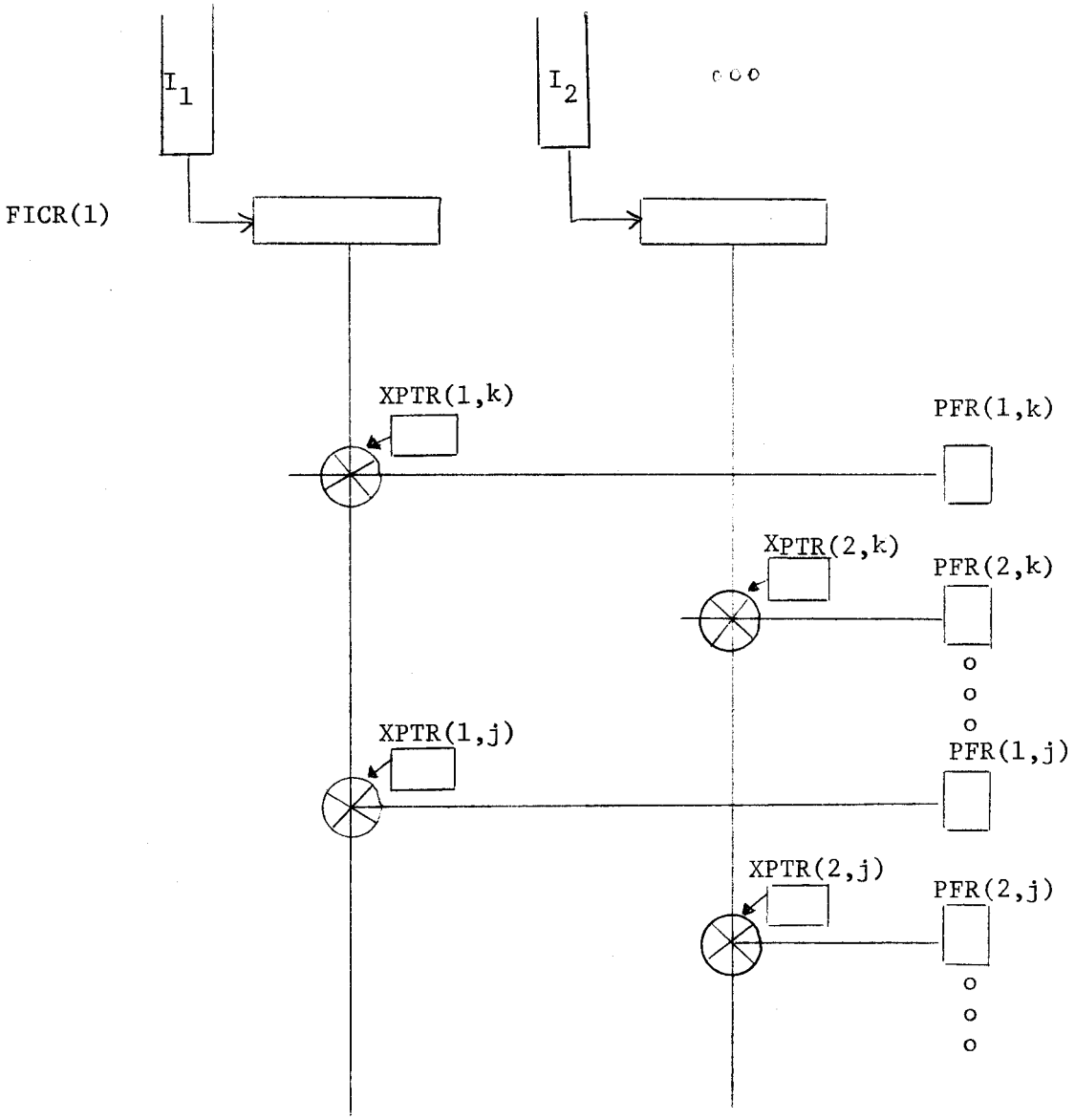

FIGURE VI.2

Implementation of the R-Primitive

The routing matrix operates as follows. Associated with the i^{th} input line I_i is a swinging pair of Frame Input Copy Registers {FICR(i)}. The bits of an arriving frame are shifted into an empty register FICR(i) at the input line rate. Upon detection of the framing pulse the pair {FICR(i)} are swapped.

The bits of FICR(i) are transmitted serially down the i^{th} vertical and are distributed among the n horizontals. This is accomplished by shifting FICR(i), {XPTR(i,1), ... XPTR(i,n)} and {PFR(i,1), ..., PFR(i,n)} in synchronism. (The presence of a '1' in the leftmost bit of XPTR(i,k) causes the associated crosspoint to be enabled.) Hence the partial frames from FICR(i) are gated to the appropriate registers PFR(i,1), ..., PFR(i,n), keeping bit positions constant.

C. Implementation of the P-primitive

The P-primitive serves to permute the bits of an F-bit string. For the Serial Matrix, the strings to be permuted are the set

$$\{PFR(i,k)\}_{i,k} = \{R(i,k) \cdot FICR(i)\}_{i,k} \qquad 5.5$$

(That is, the contents of each Partial Frame Register are permuted, in order to realize the desired time-slot assignments in the outgoing frames.)

Several approaches to the problem of implementing the P-primitive were tried and discarded. The first approach assumed the use of an F x F crosspoint array to permute the bits of a PFR. However, the Serial Matrix contains IO PFRs, implying F^2IO crosspoints. This expression is of the same order as the crosspoint count required for the Parallel Matrix.

The second approach involved cellular arrays, which have been studied extensively in the theory of sequential machines (16,17,18,19). One well-known cellular permuter[18] consists of an upper-triangular matrix of cells, each of which is a 2-input permuter. For 200 input variables, the permutation delay is about 200 gate delays or 2 μsec., but the permuter requires

$$\sum_{1}^{200} i = 20{,}100$$

cells of 6 gates each or 20,000 Dual In-Line Packs. Moreover, the values of 20,100 control variables must be re-assigned to alter the permutation, which violates the requirement of controllability. Another cellular realization, due to Bandyopadhyay et. al.[19] was also tried. For 256 inputs, this design has 2,000 control variables but requires some 700,000 cells. Hence the use of cellular arrays was considered infeasible.

Finally, a new permuter based on standard logic modules was designed and evaluated. (Fig. VI.3). It consists of a Partial Frame Register which holds the input string, a Frame Assembly Register which receives the permuted string, a rotating control store and a one-out-of-F decoder. If P is the permutation desired*, word j of the control store contains P(j) encoded in binary.

Permutation is done by shifting bits out of the PFR serially and rotating the control store in synchronism. Hence P(j) is the control store output when bit j of the input string is shifted out of the PFR. P(j) is decoded and used to steer input bit j to bit position P(j) of the FAR.

---

* A permutation P of an F-bit string is a mapping

$$j \rightarrow P(j) \text{ where } \begin{array}{l} j = 1,2,\ldots,F \\ P(j) \in \{1,2,\ldots,F\} \quad \forall j \\ P(j) \neq P(K) \quad \forall j \ \forall K \end{array}$$

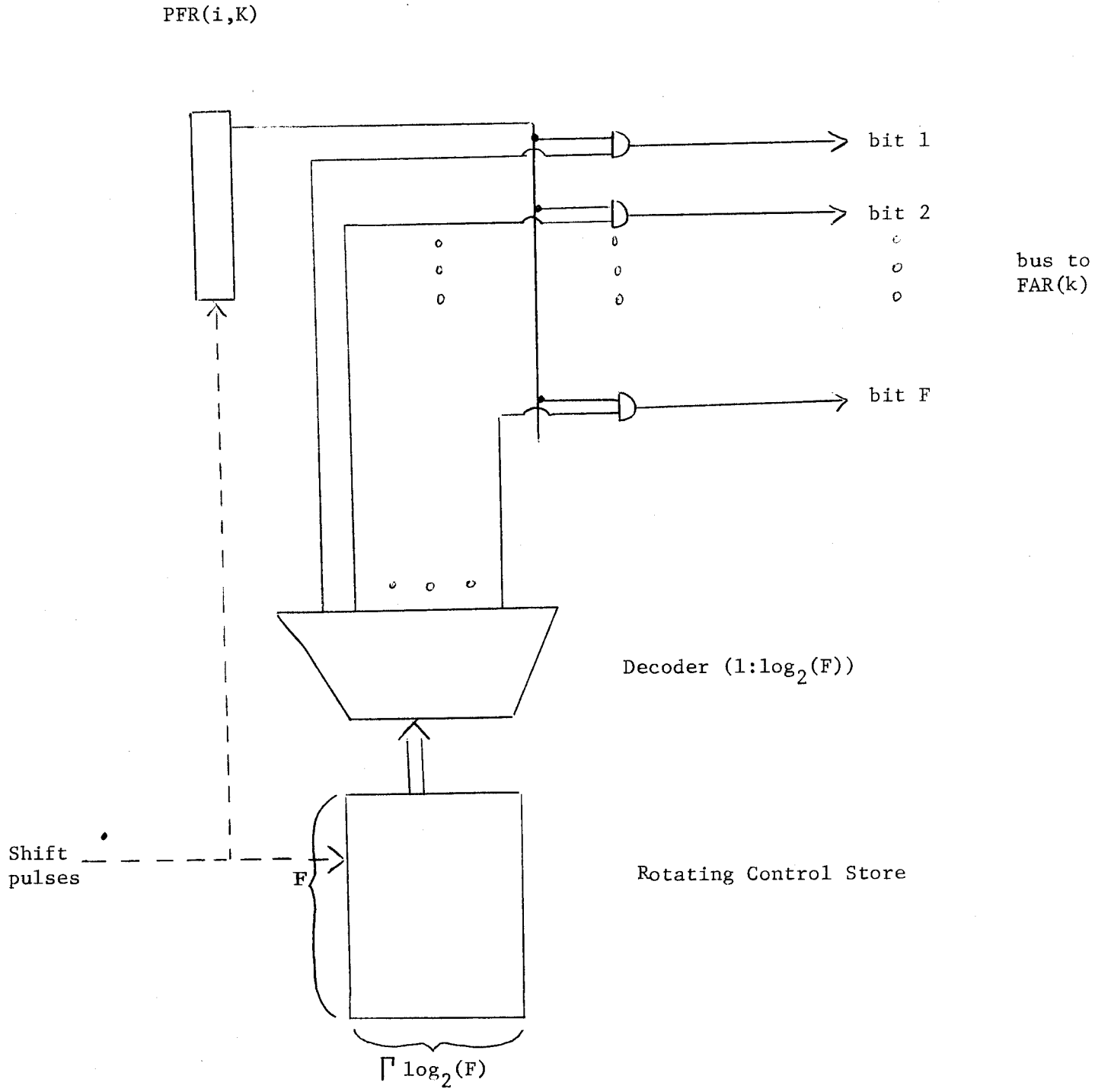

Figure V1.3- Permuter

The control store is therefore F words by $\lceil \log_2(F)$ bits in size; it can be conveniently realized with $\lceil \log_2(F)$ end-around shift registers, each F bits long. the PERMUTE primitive can of course be changed by simply rewriting the contents of the control store.

Appendix A shows that for F=200, the logic cost of this permuter is about $50, the permutation delay is about 50 μsec, and the number of control variables is $F\log_2(F)$ or 1600. The use of one permuter per PFR is assumed, or I0=100 permuters in total.

D. Cost/Performance of the Serial Matrix

The evaluation was done for the example

$$F = 200$$
$$I = 0 = 10$$
$$T = 133 \text{ sec}$$

We obtain $T_{sw} \approx 100$ μsec

(see Appendix A) and costs of $\$5 \times 10^2$ for the routing array, $\$5 \times 10^3$ for permuters, and $\$3 \times 10^2$ for registers. To this must be added the costs associated with any type of TDM switch - the control CPU, the line termination units, frame detection logic and so forth. This indicates that the cost of a Serial Matrix would be a small (and diminishing) fraction of the total cost for a digital switch.

E. Control of A Serial Matrix

This subsection shows that no load is offered to the control CPU when the switching matrix is in steady-state, thus satisfying Assumption (e). Moreover, the number of new control values implied by a change of state is acceptably small.

1) Steady-State Operation

Arrival of a frame from input line I_i is signalled by frame detection logic. This signal triggers the swapping of the pair {FICR(i)}, followed

by synchronized shifting of FICR(i), $\{XPTR\ (i,k)\}_k$, and $\{PFR(I,k)\}_k$. Shifting is terminated by a counter which causes a 'data present' bit to be set on each PFR. The 'data present' bit starts the associated permuter and the permuted PFR contents are gated to the corresponding FAR. When all permuters homing on a FAR have OR'd their permuted data into it, a 'data present' bit is set on the FAR. Occurrence of a Master Output Event - a pulse of a train having repetition rate R - AND 'data present' causes transmission of the outgoing frames. Clearly, no CPU intervention is required during this process.

2) Processing of State Changes

Requests for state changes should be delivered to the Control CPU via a Common Channel Interswitch Signalling (CCIS) channel. Any change of state (setup, teardown, or bandwidth change) involving a call arriving on line i and departing on line k, can be processed by altering the contents of XPTR(i,k) and the contents of the permuter store associated with PFR(i,k). If m bits are to be changed, m 8-bit words of permuter store and one XPTR content of 200 bits must be rewritten.* Hence the Serial Matrix is considered to be controllable.

VII CONCLUSIONS AND FURTHER WORK

A. Conclusions

It has been argued that PTDM could be superimposed on a public switched voice network to provide a functionally separate data network able to handle file traffic, transaction traffic and intermediate types with efficient use of transmission bandwidth. These properties are desirable in any jurisdiction where a switched voice network satisfying

---

* Note that a permuter is off-line for about 133-60 = 73 μsec in every interval of 133 μsec, providing sufficient time to rewrite the contents of the rotating store. A similar remark holds for the XPTR registers of a given input line.

assumption (a) is evolving, and where transmission links are long and hence expensive. It has been shown that a switching matrix with low switching delay and adequate controllability can be realized at moderate cost with existing technology.

B. Further Work

The following problems are suggested for further work.

1) Control Processing

The total CPU workload caused by a request for change of matrix state must be determined. This can be investigated by designing an inter-switch protocol and developing the algorithms needed to service each command of the protocol. From this and the results of this paper, the time needed to respond to a request for state change can be estimated. These results in turn will determine the feasible limits of PTDM flexibility. (It has been noted that transaction sources should not be served directly by a PTDM switch.)

ii) Local Distribution

Research into means of building local distribution facilities for a hierarchical network, using PTDM at the higher levels, is required. Loop techniques using a fixed frame format are a prime possibility.

iii) Network Properties

The blocking probability and link utilization of PTDM networks must be evaluated under various sets of assumed traffic characteristics. Simulation is a promising technique for investigating this problem.

iv) Host Impact

The impact of the PTDM network on the design of operating systems for attached computers (hosts) is a final question of interest.

v) Acknowledgements

The assistance and comments of J. Majithia and the support of the Defence Research Board of Canada are gratefully acknowledged.

Appendix A

Costs, Power Requirements and Operate Times
for the Serial Matrix

A. R-Primitive

Item Name	Qty.	Cost($)	Speed	(mw) Power	Subsystem Name
200-bit MOS shift register	1	2	5 Mhz	40	one XPTR register
" "	100	2×10^2	"	4×10^3	IO=100 XPTR registers
2-input gate	1	5×10^{-2}	7 n sec	20	Crosspoint
"	100	5	"	2×10^3	IO=100 crosspoints
200-bit MOS shift register	100	2×10^2	5 Mhz	4×10^3	IO=100 PFR registers
200-bit MOS shift register	2	4	5 Mhz	80	pair of swinging FICR registers
" "	20	40	"	800	10 pairs of swinging FICR registers
		445	-	12×10^3	R-primitive for I=O=10, F=200

TABLE A.1

R-Primitive Costs & Power Requirements

Appendix A

B. P-Primitive

Item Name	Qty.	Cost($)	Speed	(mw) Power	Subsystem Name
1 out of 256 decoder	1	25	35 nsec	10^4	1 out of 256 decoder
200-bit MOS shift register	8	16	5 mhz	320	permutation control store 8 x 200
AND gates	200	10	7 nsec	$4x10^3$	bit selection (to FAR)
Permuter	1	51	-	$15x10^3$	one Permuter
Permuter	100	5100	-	$15x10^5$	IO = 100 Permuters

TABLE A.2

P-Primitive Costs & Power Requirements

C. Registers

Item Name	Qty.	Cost($)	Speed	(mw) Power	Subsystem Name
200-bit MOS shift register	1	2	5 Mhz	40	-
" "	100	$2x10^2$	"	$4x10^3$	IO = 100 PFR registers
" "	10	$2x10^1$	"	$4x10^2$	O = 10 FAR registers
		220		$4.4x10^3$	I = O = 10, F = 200

TABLE A.3

Register Costs & Power Requirements

D. R-Primitive Operate Time

Δt	Event	Action
0	Framing pulse detected on I_i	exchange {FICR(i)}
15 ns	FICRs exchanged	gate lst bit onto vertical, shifting FICR, XPTR & PFR
0.2 μsec	first bit processed	shift FICR, XPTR & PFR
40 μsec	frame processed	start permuters

TABLE A.4

Approximate Operate Time for R-Primitive

E. P-Primitive Operate Time

Step	Δt	Event	Action
1	0		rotate control store and PFR one step
2	0.2 μsec	word available at store output	decoder begins
3	.035 μsec	decoder output available	gate PFR output to selection gates
4	.007 μsec	bit on bus to FAR	-
5	.015 μsec	bit registered in FAR	
6	.260 μsec	one bit processed	repeat steps 1-5, 200 times
7	52 μsec	frame processed	-

TABLE A.5

Approximate Operate Time for P-Primitive

F. Serial Matrix Operate Time

Δt	Event	Action
0	Framing pulse detected on I;	begin R-primitive
40 μsec	Frame routed	begin R-primitive
52 μsec	Frame permuted	-
20 μsec	Total estimated wire delay	-
112 μsec		

Appendix B

Notation

F : width of frames (200 bits in the example)

R : frame arrival rate in frames per second

I = # input lines

O = # output lines

T = frame time = 1/R

T_{sw} = switching time from arrival in FICR to begin out pulsing

$\{REG(i)\}_i$: $REG(i) \cup REG(2) \cup \ldots \cup REG(n)$

REG(i;j) : bit j of REG(i)

REFERENCES

1. Trans Canada Telephone System, "Computers, Communications and Canada - The TCTS Commitment", TCTS, Place Bell Canada, Ottawa, 1972.

2. Manning, Eric, "Newhall Loops and Programmable TDM", Proc. Intl. Conf. on Computer Communication", pp. 338-342, October 1972.

3. Data Transmission Company, "Comments to the Federal Communications Commission", FCC Docket No. 18920, Washington, April 1, 1970.

4. W.D. Farmer and E.E. Newhall, "An Experimental Distributed Switching System to Handle Bursty Computer Traffic," Proc. ACM Conf., Pine Mountain, Georgia, October 1969.

5. Baran, Paul, "On Distributed Communications", Series of Technical Memoranda published by the RAND Corporation, Santa Monica, California, August, 1964.

6. L.R. Roberts and B. Wessler, "Computer Network Development to Achieve Resource Sharing", AFIPS Conference Proceedings, Vol. 36, pp. 543-549.

7. Davies, D.W., "The Principles of a Data Communication Network for Computers", IFIP Congress Proceedings, 1968, pp. D11-D15.

8. Martin, James, "Telecommunications and the Computer", Prentice-Hall, Englewood Cliffs, New Jersey,

9. Vaughan, H.E., "An Introduction to No. 4 ESS", Proc. ISS, Cambridge, Mass., 1972, pp. 19-25.

10. W.W. Chu, "Design Considerations of Statistical Multiplexors", Proc. 1st ACM Symp. Data Comm., Pine Mountain, Georgia, Oct. 1969.

11. Zafiropulo, P. and E.H. Rothauser, "Signalling and Frame Structures in Highly Decentralized Loop Systems", Proc. Intl. Conf. on Computer Communication, pp. 309-315, October 1972.

12. Zafiropulo, P., "Reliability Optimization in Multi-Loop Communication Networks", IBM Research Report RZ 522, August 1972.

13. W.W. Chu, "Demultiplexing Considerations for Statistical Multiplexors", Proc. 2nd ACM Symp. Data Comm., Palo Alto, Calif., Oct. 1971.

14. H. Rudin, Jr., "Performance of Simple Multiplexor-Concentrators for Data Communication", IEEE Trans. Comm. Technology, Vol. COM-19, pp. 178-187, April, 1971.

15. D. Doll, "Multiplexing and Concentration", Proc. IEEE, Vol. 60, No. 11, pp. 1313-1321, Nov. 1972.

16. Minnick, R.C., 'Cutpoint Cellular Logic", IEEE Trans. Elect. Computers, Vol. EC-13, pp. 685-698, December 1964.

17. Dean, K.J., "Nonarithmetical Cellular Arrays", Proc. IEEE, Vol. 119, pp. 785-789, July 1972.

18. Kautz, W.H., K.N. Levitt and A. Waksman, "Cellular Interconnection Arrays", IEEE Trans. Computers, Vol. C-17, pp. 443-451, May 1968.

19. Bandyopadhyay, S., S.Basu and A.K. Choudhury, "A Cellular Permuter Array", IEEE Trans. Computers, Vol. C-21, pp. 1116-1119, October 1972.