

---

Retrospective Theses and Dissertations

---

Fall 1982

## An Expandable Architecture for a Conferencing Digital Communications Switch

Timothy A. Mitchell  
*University of Central Florida*

 Part of the [Engineering Commons](#)

Find similar works at: <https://stars.library.ucf.edu/rtd>

University of Central Florida Libraries <http://library.ucf.edu>

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact [STARS@ucf.edu](mailto:STARS@ucf.edu).

---

### STARS Citation

Mitchell, Timothy A., "An Expandable Architecture for a Conferencing Digital Communications Switch" (1982). *Retrospective Theses and Dissertations*. 646.

<https://stars.library.ucf.edu/rtd/646>

AN EXPANDABLE ARCHITECTURE FOR A  
CONFERENCING DIGITAL COMMUNICATIONS SWITCH

BY

TIMOTHY A. MITCHELL  
B.S., University of Central Florida, 1981

THESIS

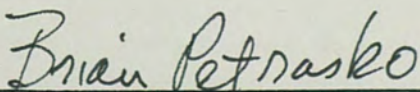
Submitted in partial fulfillment of the requirements  
for the degree of Master of Science in Engineering  
in the Graduate Studies Program of the College of Engineering  
University of Central Florida  
Orlando, Florida

Fall Term  
1982



## ABSTRACT

This paper architecturally describes the switching portion of a digital communications system that is dedicated to conferencing. The basic ideas and methods of circuit switching and packet switching are introduced. The conferencing function is described, and some resulting design considerations are discussed. The architecture of the switch is then presented. Circuit switching techniques are used throughout the architecture of the switch, coupled with arithmetic processing to accomplish the conferencing function. The architecture is developed in such a way that it is expandable in all directions to meet a given set of requirements. The requirements include the number of users the system supports and the number of conference channels provided. The processing stages of the switch can be sized based on these requirements and the chosen component speeds. The basic timing of each stage is given to describe its operation and establish the critical delay paths. The resulting switching architecture is then examined in terms of the circuit switching methods first introduced. The switch is also tested to see if it fits the criteria for being a distributed processing system. It is concluded that if the provision for dynamic reconfiguration is added, the switch fits the criteria. Finally, further topics of study are suggested.



---

DR. BRIAN E. PETRASKO  
Director of Thesis



## TABLE OF CONTENTS

LIST OF FIGURES . . . . .	iv
 Chapter	
I. INTRODUCTION . . . . .	1
II. DIGITAL SWITCHING . . . . .	3
Circuit switching . . . . .	3
Packet switching . . . . .	11
III. CONFERENCING INTRODUCTION . . . . .	17
IV. STRUCTURE INTRODUCTION . . . . .	20
Structure Outline . . . . .	20
Considerations . . . . .	23
V. PROCESSING SECTION DETAILS . . . . .	27
Data transfer between stages . . . . .	27
Switch processing stages . . . . .	29
The concentrator . . . . .	32
The pipeline adder . . . . .	39
The deconcentrator . . . . .	47
The output buffer . . . . .	54
Summary . . . . .	58
VI. CONCLUSIONS AND FURTHER STUDY . . . . .	61
Conclusions . . . . .	61
Further Study . . . . .	62
 Appendix	
A. VARIABLE DEFINITIONS . . . . .	64
B. DERIVATION OF EQUATION [3] . . . . .	66
 LIST OF REFERENCES . . . . .	 67



## LIST OF FIGURES

1.	Non-blocking switching network . . . . .	5
2.	Centralized non-blocking network . . . . .	6
3.	Centralized blocking network . . . . .	7
4.	Packet switching network . . . . .	13
5.	Division of data and control . . . . .	19
6.	Ping-pong memory detail . . . . .	28
7.	Effective ping-pong memory representation . . . . .	30
8.	Processing stage interconnection . . . . .	31
9.	Internal concentrator detail . . . . .	33
10.	Concentrator ideal timing . . . . .	36
11.	Internal pipeline adder detail . . . . .	40
12.	Pipeline adder ideal timing . . . . .	43
13.	Pipeline adder expansion structure for many concentrators . . . . .	46
14.	Internal parallel-type deconcentrator detail . . . . .	50
15.	Parallel-type deconcentrator ideal timing . . . . .	53
16.	Internal output buffer detail . . . . .	55
17.	Output buffer ideal timing . . . . .	57



CHAPTER I  
INTRODUCTION

Telecommunication is often thought of in terms of the telephone system. Two people can communicate with each other over a large distance. The feature of conferencing allows several people in different locations to speak as if they were in the same room.

Conferencing is a popular feature of a private branch exchange (PBX). The PBX is effectively a small telephone system that provides service to one user organization. In the long run, it becomes cheaper for an organization to purchase such a system than to rent the service from the local telephone utility. An example of a modern PBX is the ROLM Corporation's CBX (Computerized Branch Exchange). The system can support 150 simultaneous two-way conversations (Kasson, 1979).

An organization such as the Kennedy Space Center (KSC) has a need for such an exchange. KSC desires a conferencing capability of at least 500 conferences to take place simultaneously, with no restriction on the number of users that have the potential to speak in or hear a conference. The space center also desires the feature of a user being able to hear more than one conference at once. The degree of conferencing required by KSC is far beyond any device that is on the market (MacDonald, 1982). For example, the ROLM CBX only supports 8 conferences with 8 users in each conference.



The motivation for the development of the architecture for such a conferencing system came from KSC's needs. Chapter II of this paper provides an overview of switching methods and strategies. Chapter III is an introduction to the conferencing situation. Chapters IV and V present some design considerations and the architecture of the switch portion of the conferencing system. Chapter VI discusses conclusions about the developed architecture and suggests further areas of study.



## CHAPTER II

### DIGITAL SWITCHING

#### Circuit Switching

A circuit switch is a device which provides for the exchange of messages (Joel, 1977). The messages can be voice, video, or electronic data. The exchange takes place between an input port (the message sender) and an output port (the message receiver). The circuit switch sets up the communication path, which may be a set of lines, by controlling the interconnection circuits. The communication path, once set up, is a dedicated connection for the duration of the message transmission. The most familiar example of a circuit switch is the telephone exchange. The input and output ports are together in one device (the telephone), and all telephones are ultimately connected to the system that sets up the talk paths between the users.

An ideal circuit switch has the capability of connecting all input ports and output ports in any arbitrary pattern. How completely arbitrary the connections between ports can be accomplished depends on the complexity of the network. Anyone who has had difficulty in making a long distance phone call on Christmas Day is aware that the phone company's circuit switch is not ideal. Most circuit switches have a saturation point where all possible communication paths are in use and any request for a message transmission by an input port will be



refused. The structure of the switch determines where, or if, this point is reached.

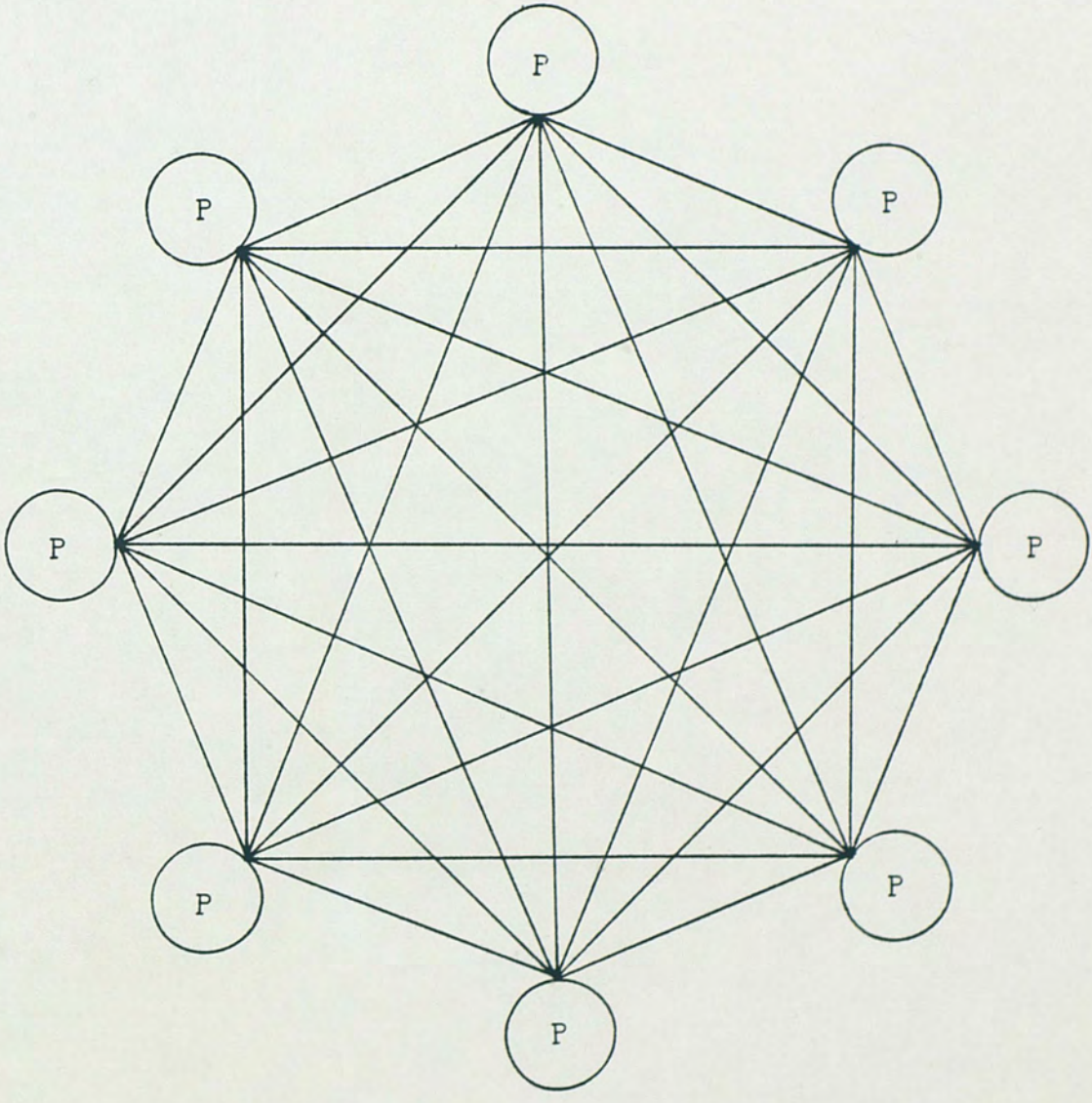
If a structure allows the ideal of all input/output ports in use with any connection pattern, the structure is said to be a non-blocking network. Figure 1 shows such a structure.  $N$  ports are connected by  $N*(N-1)/2$  lines. Each line is a two-way path, and the switching operations take place at or near each port. Disadvantages with this network are that the number of lines increases as the square of  $N$ , and the switching function is not centralized in one location. The structure as shown does not easily expand to service more ports, and has application only when  $N$  is small.

Figure 2 shows an improvement on the first structure.  $N$  ports are connected by  $N-1$  Lines, and all the connections can be performed at a central location. This network is also non-blocking, as all possible connection combinations can be accomplished.

Each interconnection point where a line could potentially be connected to a port is called a crosspoint. The crosspoints are controlled by the switching network to create the desired communication paths. The network in Figure 2 has  $N*(N-1)/2$  crosspoints. The network in Figure 1 has twice as many:  $N-1$  crosspoints at each port, for a total of  $N*(N-1)$  crosspoints.

The network of Figure 3 has less crosspoints, and the interconnection paths have been lengthened. Each circular path is called a link, and one link is required for two ports to be connected. When there are less links ( $L$ ) than  $N/2$  ports, all ports cannot be in use simultaneously. Note that in Figure 3,  $N=8$  and  $L=3$ . This configuration

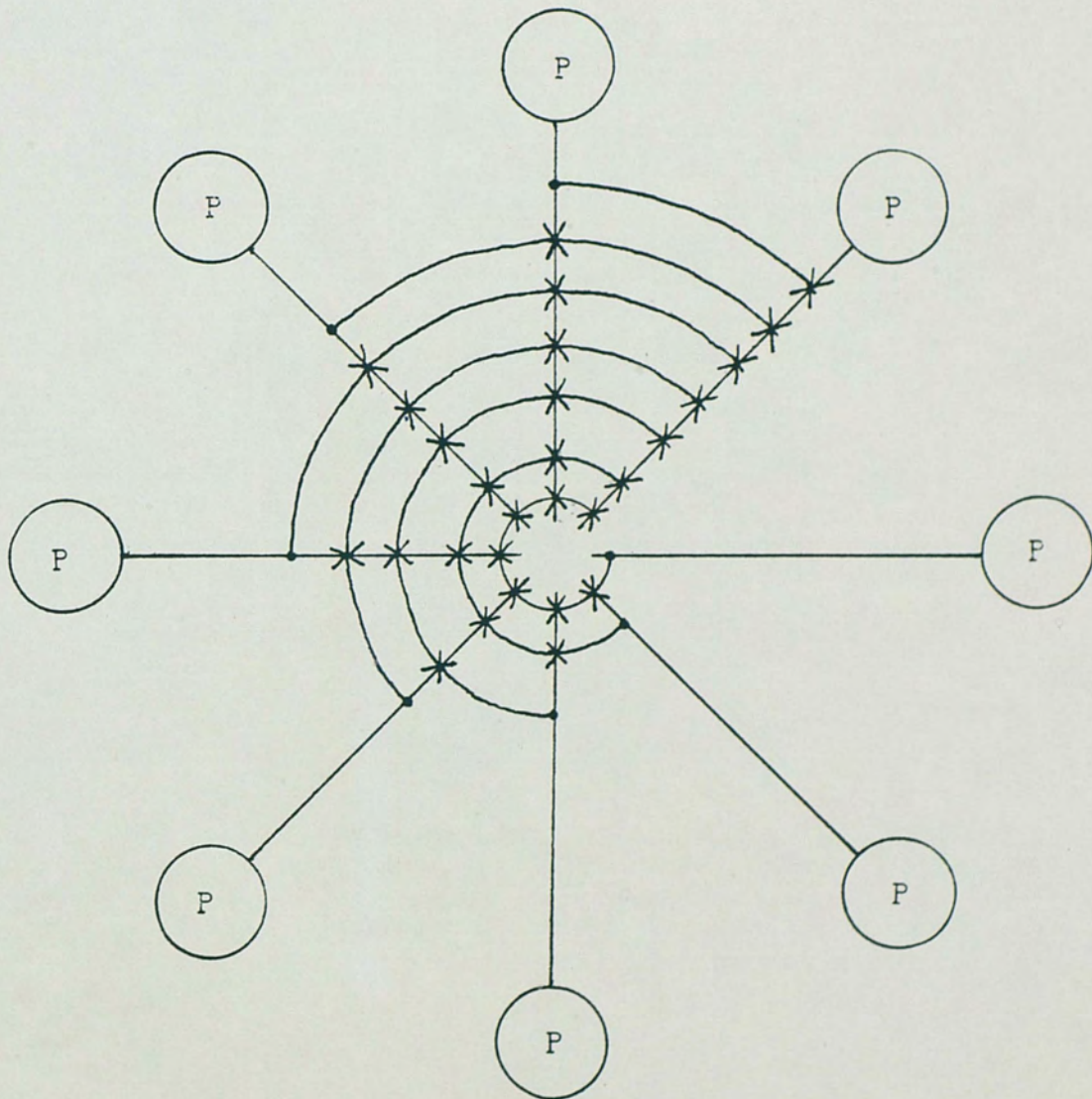




P = port

Figure 1. Non-blocking switching network.





X = crosspoint P = port

Figure 2. Centralized non-blocking network.



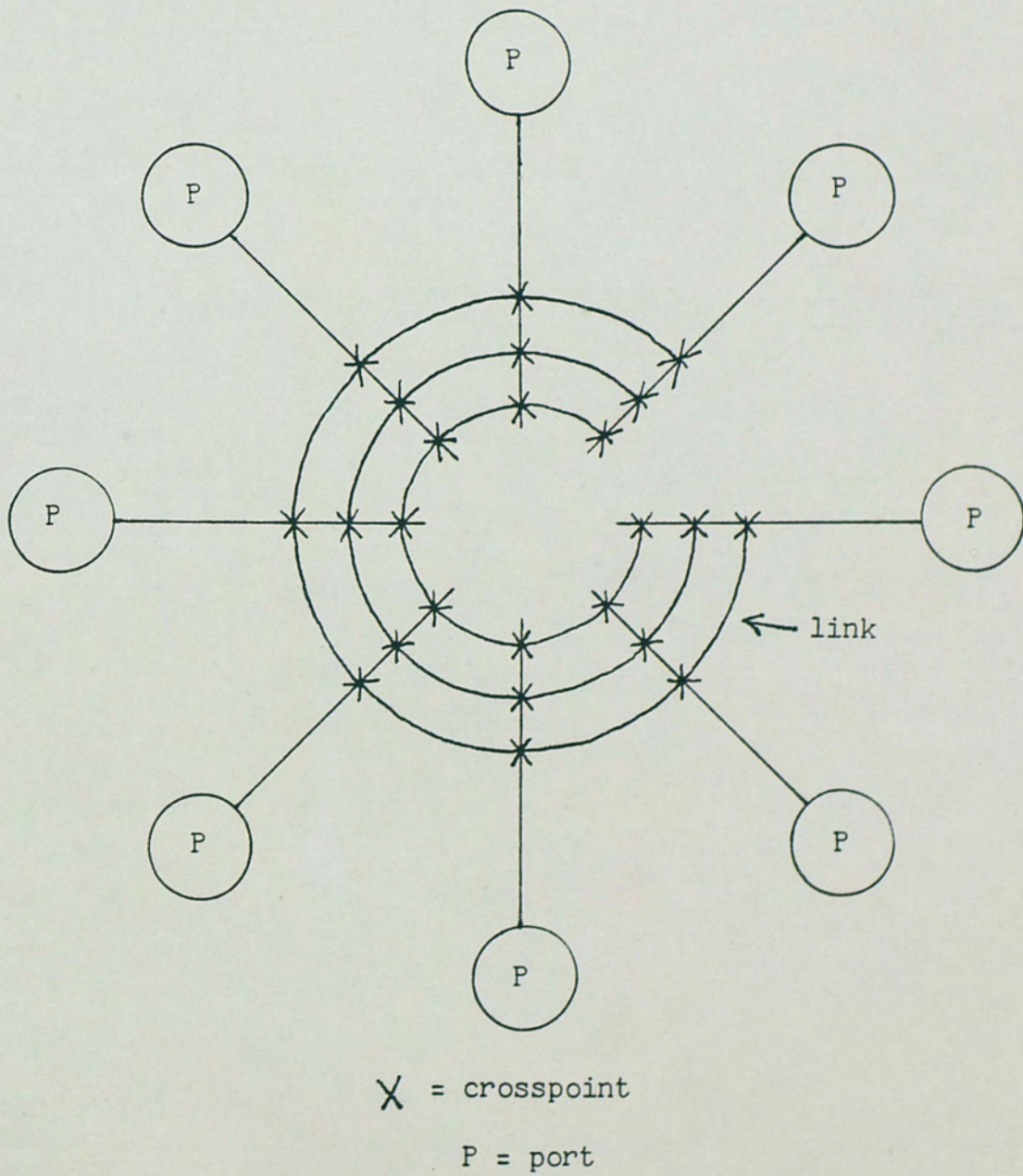


Figure 3. Centralized blocking network.



is thus known as a blocking network. When  $2L$  users are connected, the remaining  $N-2L$  users are blocked from being serviced and must await free links. How many simultaneous communication paths are required will determine  $L$ .

The networks considered have been all examples of space division multiplexing (SDM). A place in space (a circuit) was dedicated to one message transmission for the entire message time. Two other methods of sharing a communication medium exist: frequency division multiplexing (FDM) and time division multiplexing (TDM).

FDM involves dedicating a specific band of frequencies to one message. Many messages can then be simultaneously transmitted over a broad band medium. The most familiar examples are radio and television. Signals that share the same bandwidth (video and audio information) are all transmitted through the air at once. The radio or television receiver selectively tunes in on one and effectively ignores the rest. FDM has been investigated for switching but is more expensive than TDM.

TDM involves dedicating the medium to the message for a specific amount of time. Communication sampling theory states that if a signal is sampled at the Nyquist frequency or higher, the original signal can be reconstructed from the samples at the receiver. The switching and sampling can take place in an analog fashion, where source and destination ports are actually connected for a moment. Most recently, with the advances in digital electronics, the time samples are digitally encoded and stored in a memory. Each sample can then later be transmitted over the medium, when the proper time comes. The proper time is



called a time slot. Each message source's sample arrives at the switch at some previously agreed time, or a sample has some source identification with it. The samples are then stored in the central switch's memory. Then the samples can be read out of memory and sent to the proper destination. The switching function can occur 1) where an incoming sample is stored, 2) where an outgoing sample is retrieved from, or 3) both. By "shuffling" the sample data in terms of what time slot it ends up in, communication switching can occur. This method of TDM switching is called time slot interchange (TSI).

If there are many users, not enough time may exist to process all the samples before the users need to be sampled again (because of bandwidth requirements). The switch can be split up, with parts of it dedicated to certain users. Or the switch can be constructed so that one memory is filling with samples and another is sending out samples that were taken in on the last frame time. "Frame time" means a period in time between when all senders must be sampled. New samples to be processed are presented to the switch each frame time. Sections of the switch can process a frame of data (samples) and pass it on to the next section for further processing. The routing required here is simply to read out the samples in such a way as to properly swap the time slots to establish the right communication paths. Flow through the switch is then maintained, with the bandwidth requirements of the message met. A delay is introduced from input to output depending on how many routing steps are involved. The idea is that the structure can be changed to accommodate the amount of routing that must be done. To handle more users, the system can expand by duplicating routing and sampling



functions and having them operate in parallel.

Another time division idea arises when a space division system is time switched. Then many more users can be serviced than with a TSI system since there is more than one common transmission medium. This switched SDM system is called a time multiplexed switching (TMS) network.

TMS and TSI systems that use memory staging as previously discussed will generate a delayed transmission path. Both systems come under the broader term of store and forward switching. The storage action is what causes the delay. Different types of store and forward networks require memory for different reasons. They are:

1. The transmitter may be in use when a message arrives.
2. Multiple transmissions of the same message are required.
3. The called unit is busy.
4. The speed of the transmitter or medium is less than the received rate.
5. The format or coding of a received message differs from how it is transmitted.

These reasons come into play when networks are not required to handle a steady flow of information. When computers transmit data to other computers, the message is sent in a large block at a high speed. Then the connection is idle for a very long time until another block is sent. A SDM system would be wasting a line if the connected devices communicated only intermittently. A TDM system would be wasting a time slot if it had to sample a silent device. A receiver could contain a buffer memory and accept messages at any time, whether the computer it was



connected to was busy or not. How often the computer is transmitted to is one factor that determines how large the buffer needs to be. Packet switching takes advantage of a device's idle time and only transmits messages when specifically asked to.

### Packet Switching

As suggested by one author (Kleinrock, 1978), our privately owned cars are a waste of money. They sit parked most of the time, not in use. The value of the convenience seems to outweigh the cost of the use efficiency we get out of it. There is some point where the cost of the idle time is too high (few of us own a private jet) or where the cost is so low we are almost forced to pay for the convenience (we all own idle pencils).

Communication devices exhibit the same kind of idleness. Consider a computer listening to a set of time shared terminals. The amount of time that passes between a user's keystrokes is large compared to the amount of time it takes for the computer to process a keystroke. A system like this could use TDM of sorts, where the computer would poll each device for new data. But time would be wasted if a device had no data. A more efficient use of time would be accomplished if the computer was connected to a terminal only if the terminal had new data. For terminals that are close to the host computer, each could interrupt the host when a key was hit.

Now consider a group of computers (hosts) separated by miles. The hosts desire communication with each other with blocks of data. As with a time sharing network, the ratio of peak demands to average demands is very high, that is, the hosts are "bursty" and work among

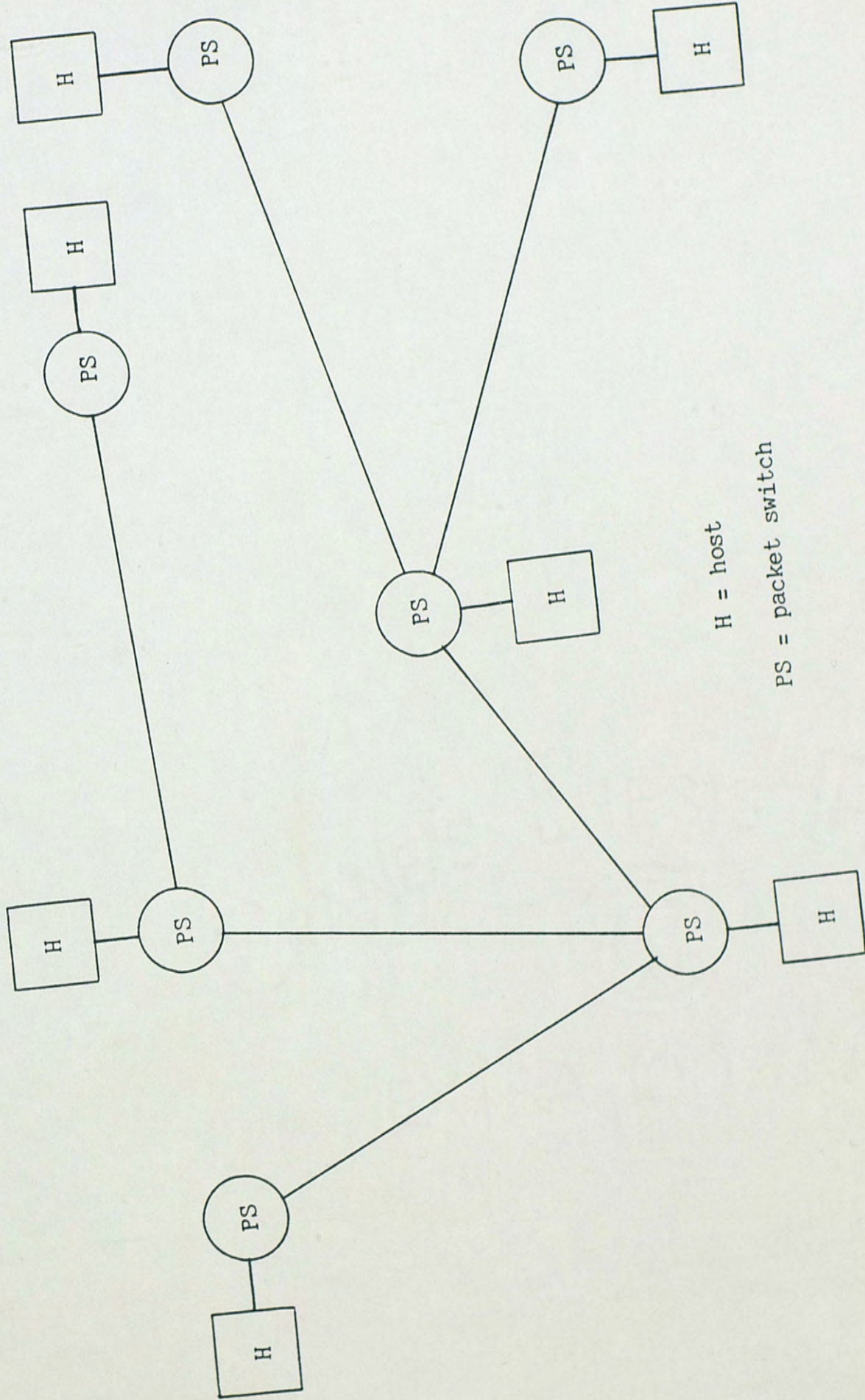


themselves most of the time. Because of the distance between them, a dedicated network of lines connecting each pair would be costly. If hosts were connected such that a message could be sent to any other host provided there always was a path through intermediate hosts, communication could be accomplished by packet switching. Such a network might look like Figure 4.

The idea of packet switching involves a set of bursty (usually idle) transmitters (hosts) connected together. Each host is connected to the network through a packet switch. Suppose a host wants to send a block of data to another. The block of data and the destination host's address is given to the packet switch tied to the sending host. The packet switch divides that block into smaller messages, usually of uniform size. The packet switch knows how the hosts are interconnected, so it starts sending the message packets towards the destination host. If there is an intermediate packet switch, that switch receives the packet and forwards it on again until the packet reaches the destination. Each message packet must have a destination identification along with the data. The packet switches act like a virtual channel between hosts.

The packet switch can also have a certain amount of smartness to improve on transmission speed. If more than one possible route exists from source host to destination host, packet switches could route the data along paths that had less traffic than others. A measure of the traffic a packet sees is how full the message buffers are at the other packet switches along the possible routes. Each switch could broadcast its status from time to time, to update the other's routing tables. A packet in this case might need source information also, to prevent





H = host

PS = packet switch

Figure 4. Packet switching network.



looping around a "low-use" path.

A packet switch might not be able to process a packet fast enough before another comes in. Thus, there is a buffer memory in a packet switch on the receiving end. A buffer is needed to save the part of the message that has not been divided and sent out yet from the connected host. A buffer on the transmitter side is needed also if retransmission is required because of an error flagged at the receiver. Once an acknowledgement is sent, the sending packet switch can discard the message. Note that a packet switch could simply be a minicomputer with special software.

One advantage of breaking up the data block into packets is that each message sees a relatively uniform waiting time. Delays are shared by all users rather than short messages waiting on long ones. Each packet's addressing directs it completely independent of another through the network. The size of the packets as well as a measure of overall traffic density impact the packet buffer size. Packet switching allows complete asynchronous use of transmission resources. Again, packet switching becomes feasible when messages are bursty and switching functions (done by a minicomputer) are cheap compared to a network of lines.

Routing and flow control are very important ideas in packet switching. To take advantage of the host's idle time, the system must be able to deliver the packets to their proper destinations efficiently. For reliability, the switches must know if the packets arrived correctly at the destination. Flow control and routing algorithms, as well as error checking schemes are the "smarts" that



allow the simple scheme of packet switching to function. These subjects all come under the broader heading of protocols (Pouzin and Zimmerman, 1978).

The classic example of a network of computers connected by packet switches is the ARPANET. The ARPANET uses minicomputers interconnected by 50 kbit/s leased lines in a distributed network. ARPANET became operational in late 1969 with 4 hosts and as of 1977, 111 hosts were supported (Roberts, 1978).

Packet switching could be applied to voice communication although the advantage is not as great as with computer communication. Speech requires more constant attention than true bursty flow. In normal speech, the ratio of active voice to silence is from 50% (Barberis and Pazzaglia, 1980) to 33% (Roberts, 1978). Such a system would require an "intelligent" design using speech detectors and a packet voice receiver. The detector would indicate when a silent period had occurred, and then the previously gathered packet of active voice could be transmitted. In computer communications, every bit of data is considered essential, but a voice system that occasionally lost a few voice samples would be tolerated. The human mind can easily interpret less than perfect speech (such as a phone conversation), where a computer cannot guess missing pieces of data. It has been suggested (Dally, Gold and Seneff, 1980) that a packet voice system could detect traffic variations and handle a system overload by dropping overall voice quality. The disadvantage is that it would take time to detect the overload and broadcast the commands to change the flow protocols. The flow control commands could be lost in a near-saturated system.



The difficulty with a packet voice switch is that the packet delays through the system are a function of how heavily loaded the system is. Unguaranteed delays could upset the conversations between the users. Given a set of users, the degree of interconnection required to guarantee a minimum delay would probably approach that of a circuit switch. There is a high cost assumed with the use of packet voice receivers. Inefficiency of the transmission medium exists in that speech samples would have to have address routing information accompanying them. A collection of speech devices would also tend to be in a much denser layout than computers using packet switches are.



## CHAPTER III

### CONFERENCING INTRODUCTION

The usual extent of voice telecommunication is two individuals talking to each other. Existing switching systems are two-user intensive and provide for only a limited degree of a conferencing function. Conferencing allows several users, each connected to the system through telephone devices, to talk to each other. The conferencing function allows people in different locations to speak as if they were in the same room. A vendor survey done by the MITRE Corporation indicates that any existing systems must be extensively modified to handle a large conferencing structure such as the one present at the Kennedy Space Center (MacDonald, 1982). The space center also has the requirement of a user to be able to listen to more than one conference at a time.

Packet switching and circuit switching were described in Chapter II. Because of the difficulties associated when packet switching is applied to voice communication, circuit switching has been chosen as the basic method of switching for the conferencing system.

As discussed before, a circuit switch sets up a path between the users who desire communication. The path may be digital in nature, where a user's voice sample is stored in memory locations until it is delivered to the receiver. There is a section of the circuit switch that controls the addressing of the voice data memories, and thus,



controls the set up of the talk paths. This section of the switch will be referred to as the master control unit. Master control has other functions that will be identified later.

Figure 5 shows the division of the signalling data and the voice data that arrive from each user. All users deliver to the system digital voice and signal data. The data is multiplexed and the signal data is stripped off to be sent on to master control. Voice data is sent to the switch. The switch processes and routes the user voice samples as directed by master control. The output of the switch is recombined with any signaling data that may be required for output functions (e.g., dial tones). The digital output is then demultiplexed back to the receivers.

The scope of the remainder of this paper is to architecturally design the switch portion of the conference communication system. Functions required by master control and the nature of the switch's voice data input and output will be specified. The structure will be described parametrically, where the user specifications and hardware speeds will size the switch's processing stages. The goal of this paper is to present an architecture that can provide a non-blocking conferencing function for any number of users. The flexibility of the architecture to handle any range of specifications will rely on a great deal of parallelism in processing. The architecture will expand as required by duplicating its parts and having them operate at the same time.



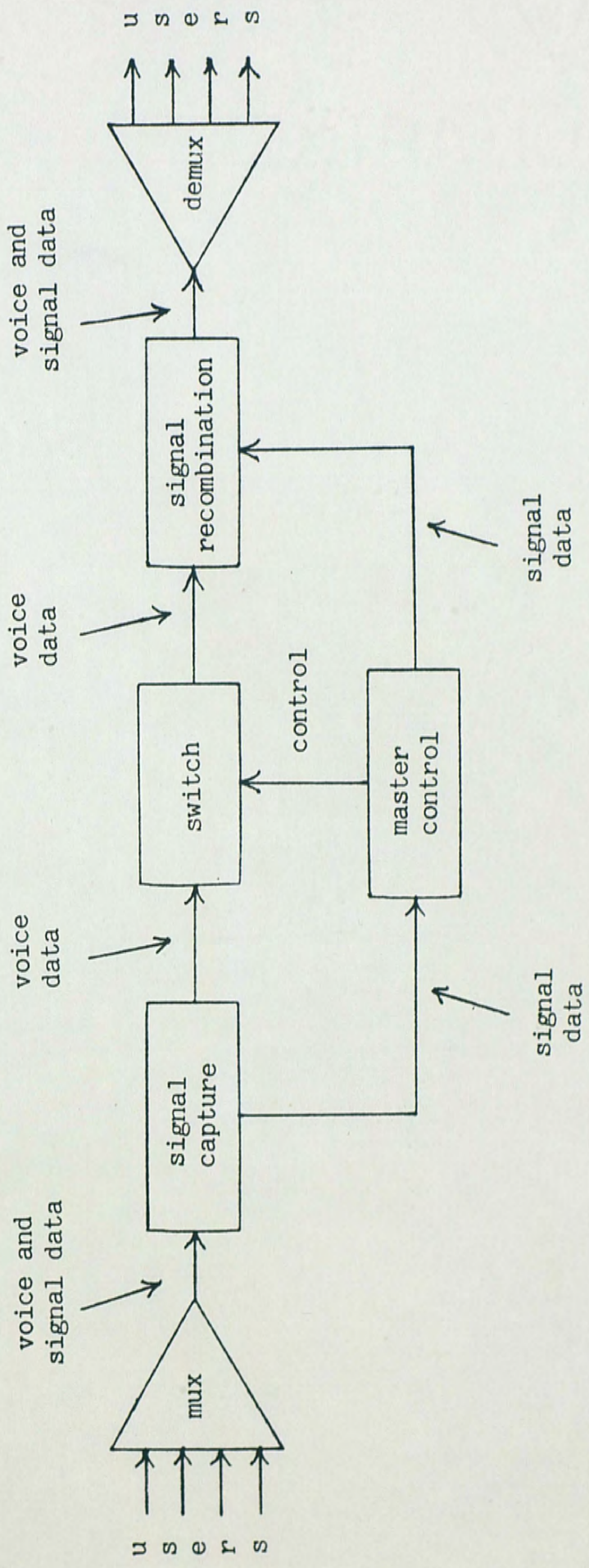


Figure 5. Division of data and control.



## CHAPTER IV

### STRUCTURE INTRODUCTION

#### Structure Outline

As implied before, multiplexed digital data arrives at the switch, and multiplexed digital data leaves the switch. The input and output formats will be the same. All the user's voices will be time sampled periodically, and converted into a digital word of  $B$  bits. The code used for the conversion does not have to be linear. For example, a typical device used for voice sampling is a Codec. The Codec uses an 8 bit nonlinear code (MacDonald, 1982). The switch will convert it to a linear code to allow for easy arithmetic manipulation. As required by sampling theory, a user presents a new digital sample every  $T$  seconds. Note that the users will not all be sampled at the same time, but every user will be sampled periodically. Data arriving at the switch will be further multiplexed, in that  $C$  sets of  $I$  user's digital voice samples will arrive at the switch after every  $T$  seconds. The digital words will arrive in parallel on  $C$  sets of  $B$  bit lines. Data in each set will arrive at the same time as another set. The digital words will arrive in an order known to master control, such that a voice sample's arrival time and which set contains it will identify that user. The total number of users processed by the system,  $TU$ , is  $C \cdot I$ .

The purpose of the switch is to allow conferencing. As the ear adds up many voices in a room, the switch must add up the voice data of



user's speaking on the same channel. If a user is listening to more than one channel, those channels must be added and the digital total delivered to the user. The most commercially available hardware addition is done on linear codes, so the  $B$  bit word will be converted on a  $L$  bit linear code by the switch.  $L$  is a direct function of the input code format.

The data input to the switch will be processed in stages. Each of the  $C$  sets of the  $TU$  users will be processed at the same time. Speed of the components of the system will determine how many users a set  $C$  will contain ( $I$ ). Each stage of the switch will receive new digital voice information every  $T$  seconds. Thus, each stage must finish its processing before the next time period  $T$ . Note that  $T$  corresponds to a frame time. At the end of each frame time, the processed data will be passed on to the next section for processing. Data will pass through the stages until the voice sample that each user is scheduled to hear has been computed and routed.

There will be four processing stages in the architecture to be described. Master control will influence the operation of all of them. The first processing stage is accomplished by a concentrator.  $C$  concentrators will be in the switch. Each concentrator receives  $I$  digital voice samples every period of  $T$  seconds. A user's sample arrival time identifies that user to master control. Which conference the user is speaking to is then known, and then that sample can be added to that conference total that is being accumulated during that sample period. A specific conference that is taking place will be constructed each sample period in a place called a channel. Channels in the switch are



digital locations in memory. All of the digital samples of all the people speaking on the same channel will ultimately be added and stored in one location in memory. A concentrator accomplishes part of this addition. All users being processed by a specific concentrator and who are speaking on the same channel will have their voice samples added to a memory location in the concentrator. At the end of the current sample period, each of the  $C$  concentrators has a partial sum of each channel. These partial sums are passed to the next stage for processing. The concentrators then begin the addition function on the new set of incoming digital voice samples from the next sample period.

The second processing stage is accomplished by the pipeline adder. The pipeline adder must finish the task of adding all the voice samples destined for one channel to one specific memory location. What it has to start with is all the partial sums of each channel that were formed by the concentrators. The pipeline adder gets its name from the method by which the channel sums are completed. Let the conferencing system be capable of handling  $CH$  channels. Then the pipeline adder must be able to perform  $CH*(C-1)$  additions in a sample period. The speed that is implied requires fast hardware if the system supports a reasonable number of users and conference channels (such as 16,000 users and 2,000 channels). Rather than driving  $C-1$  2-input adders with  $C$  partial sums of a channel and waiting for the digital sum to propagate to the end, sums can be buffered at each adder, forming a controlled flow pipeline. The operation of the pipeline adder will be described in more detail later. When the sample period is finished, all the voice samples on the same channel have been added and stored in one memory location.



The third processing stage is done by a deconcentrator. A deconcentrator accomplishes the task of adding up those channels a user wishes to hear. There will be  $D$  deconcentrators serving  $O$  users each. Note that  $D*O=TU$ . Master control knows which channels a user is listening to, and controls the channel additions. A volume adjustment is performed on each channel before it is added to the user's final sum. Volume adjust data is part of the signal data that is stripped off and routed to master control. The architecture is flexible enough so that any number of channels may be added to form a user's final sum.

The fourth processing step is simply an output buffering stage that passes the completed sums to the users served by the deconcentrator. No processing is performed, other than converting the channel sums from a linear format to the output format of  $B$  bits, to match the input format. The stage introduces another delay of  $T$  seconds from the input of the switch to the output. The buffering stage resulted from the architecture of the deconcentrator and the method of how the stages pass data to the subsequent sections.

#### Considerations

The four processing stages of the switch will introduce a delay of  $3T$  seconds from when a user's voice sample is entered and when it exits as part of a channel summation. Since a user will normally listen to the channel he is speaking on, he will hear his own voice. MITRE studies found that the psychological effect on a user hearing his own voice delayed was a function of the amount of delay (MacDonald, 1982). Reactions of test subjects ranged from complete tolerance to annoyance and even nausea. However, the delays that caused the undesir-



able affects on the test subjects are much longer than those expected to be encountered here. MITRE judged a delay of 50 ms or more was undesirable. A typical voice sample rate is 8 kHz, implying a delay of .375 ms through the switch portion of the communications system. If the delay of the entire system approaches the undesirable delay, the structure described here would have to be modified. Each user's voice sample would have to be saved until the sum he was to hear was created. Then his sample would be subtracted from that sum. Since the user's sample is available four sample periods before the sum he is to hear is finished, additional memory acting as a buffer would be required. The structure presented here will result in a user hearing his own voice.

When sizing the components of the four processing stages, the classic "cocktail party" problem arises: How many separate, distinct voices can a user distinguish between? If  $V$  digital binary numbers are to be added, each encoded in  $L$  bits (sign and magnitude), the answer will require at most  $W$  bits (sign and magnitude), where  $W$  is given by the integer part of

$$\text{LOG}_2[(2^{L-1}-1)*V]+2. \quad [1]$$

The number of simultaneous voices that must be heard distinctly will be  $V$ .  $V$  will partially determine internal dimensions such as memory size, internal buses and hardware adder widths. If more than  $V$  voices could be ultimately summed and delivered to a user (perhaps in a period of emergency or during a loud argument), the switch architecture could handle the situation in different ways. One is to test if the sum will overflow, and then just not add in the sample(s) that would produce the overflow. Since voice samples are both positive and negative, the



order of addition would be a factor in what final sum resulted. If many negative samples were added, the negative limit would be reached, causing the system to discard further negative samples. These samples might have been added if some positive samples had been added first. To best implement such a scheme, the samples would have to be sorted so they would be added in the order of ascending magnitude, with a secondary requirement of a maximum arrangement of sign changes in the sorted list. The specially sorted list of samples could then be added in order with a minimum amount of samples discarded. However, the implications of such a scheme demand a large overhead in terms of hardware and processing time. An acceptable alternative would be to simply perform the addition in any order. Then simple overflow hardware logic would become a part of each adder. Again, extra time would be required for the overflow signal of the adder to control what answer was saved.

Another way to handle the problem of overflow is to ignore it. Then the question arises, what does a digitally "rolled over" signal sound like? The sound of an analog amplifier distorting as it saturates is familiar, but a digital number being fed to a digital-to-analog converter does not cause saturation of the amplified signal. Since human beings have adapted to interpreting voice from a saturated analog device, perhaps users of the conferencing system would become accustomed to interpreting a signal that resulted from digital overflow. The architecture presented for the switch network relies on that premise. If more than  $V$  voices occur simultaneously within the same channel summation and a digital sum overflows, it will go undetected through the system. The switch architecture only guarantees undis-



torted delivery of  $V$  simultaneous voices. Tests would have to be performed to justify the additional hardware to make digital addition overflow behave like analog saturation. The tests would have to show that listeners could not adapt to the sound produced by digital overflow.



## CHAPTER V

### PROCESSING SECTION DETAILS

#### Data Transfer Between Sections

As described earlier, a concentrator stores channel partial sums in its memory. At the end of a frame time, all the concentrators pass those sums to the pipeline adder. Likewise, the pipeline adder passes all of the completed channel sums to the deconcentrators at the end of each frame time. The way a large block of data is passed to another processing stage is with a structure known as a ping-pong or A-B memory. Suppose a concentrator is filling a memory with partial channel sums during frame  $i$ . Then the pipeline adder will be processing the partial channel sums of frame  $i-1$ . That data is in an identical memory as that of the concentrator. At the end of the frame, the concentrator has completed all of its channel partial sums and the pipeline adder has finished processing the channel partial sums from the previous frame. If the memories could change places, both processing sections would be ready to work on the next frame of data.

Figure 6 shows how the memories can effectively change places by changing which processing stage controls the address and data lines. The control line PING determines the address and data connections. When PING is low, memory A is connected to system 0 and memory B is connected to system 1. When PING is high, the opposite is true. The tri-state outputs on the data input multiplexers were needed because it is assumed that



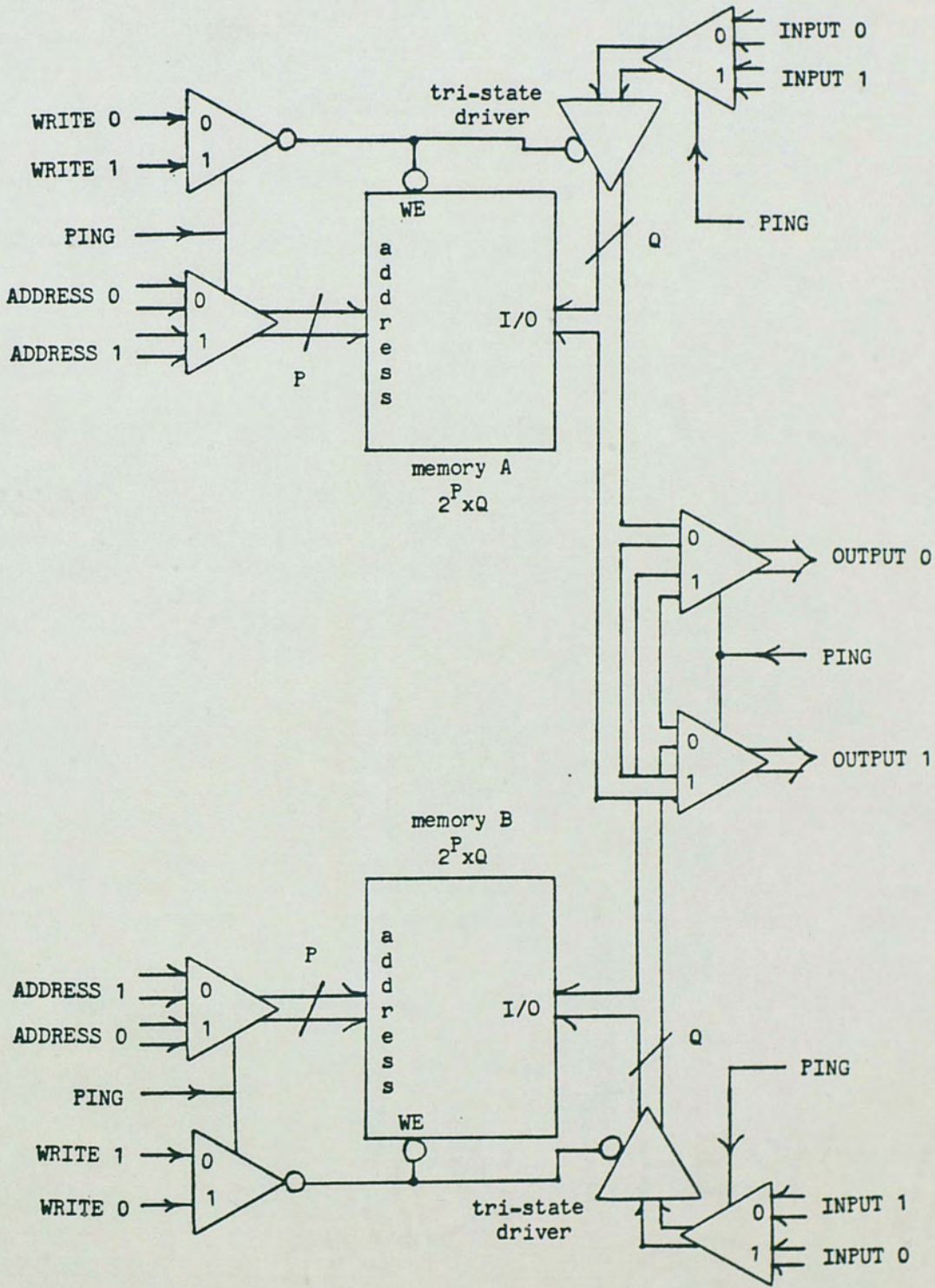


Figure 6. Ping-pong memory detail.



the memory devices have bidirectional input/output lines. The directionality of the memory data lines is controlled by write enable (WE) which is typically active low. When a new frame time passes, the logic state of PING will toggle. The memories will effectively change positions, and the action will be transparent to the different processing stages.

Note that the components in Figure 6 allow for the worst case of a ping-pong memory: read and write operations can occur in both systems. If system 1 only read data and system 0 only wrote data, the multiplexers connected to OUTPUT 0 and the INPUT lines could be eliminated. Both of the inputs to the tri-state drivers would be connected to INPUT 0. The requirements of the two systems will determine the minimum amount of extra parts to make two memories into a single ping-pong memory.

The processing stages will see only one memory, with separate input and output lines, a set of address lines, and an active high write enable line. During a write cycle, the output lines will show the data being fed to the input lines. The read and write delays will be a function of the memory delay and the multiplexing delay. Typically, the memory delay will be significantly larger than the multiplexing delays. The delay of such a ping-pong memory will be understood to account for the extra switching involved. Further on in this paper, a ping-pong memory will appear as in Figure 7, without the extra detail of Figure 6. It will be assumed that the PING line will change state each frame time, unless otherwise noted.

#### Switch Processing Stages

Figure 8 shows the interconnection of all the processing stages for clarity. C concentrators each receive I digital samples of B bits



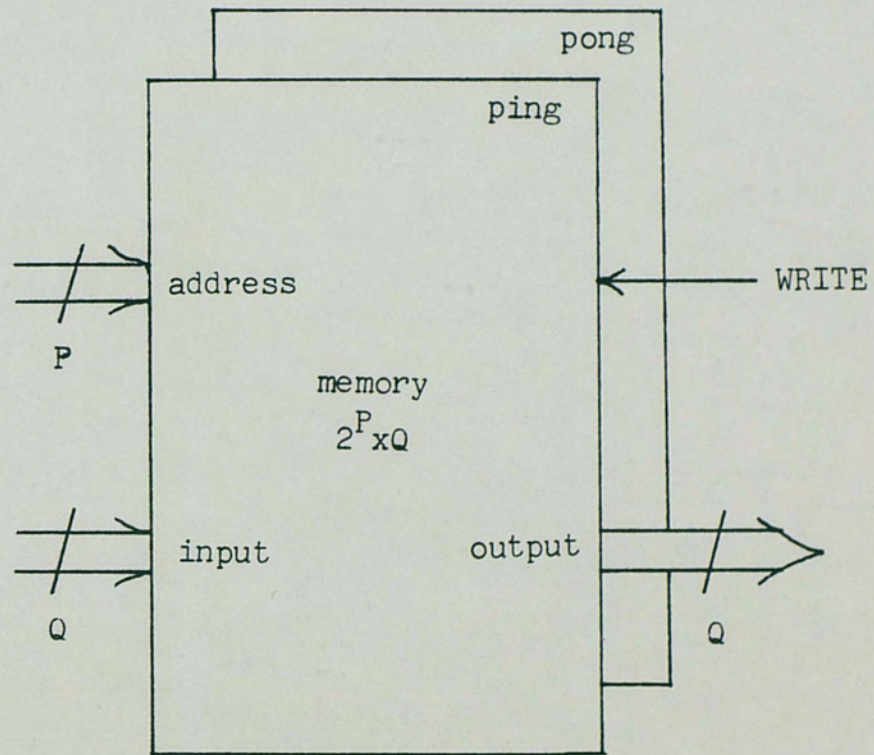


Figure 7. Effective ping-pong memory representation.



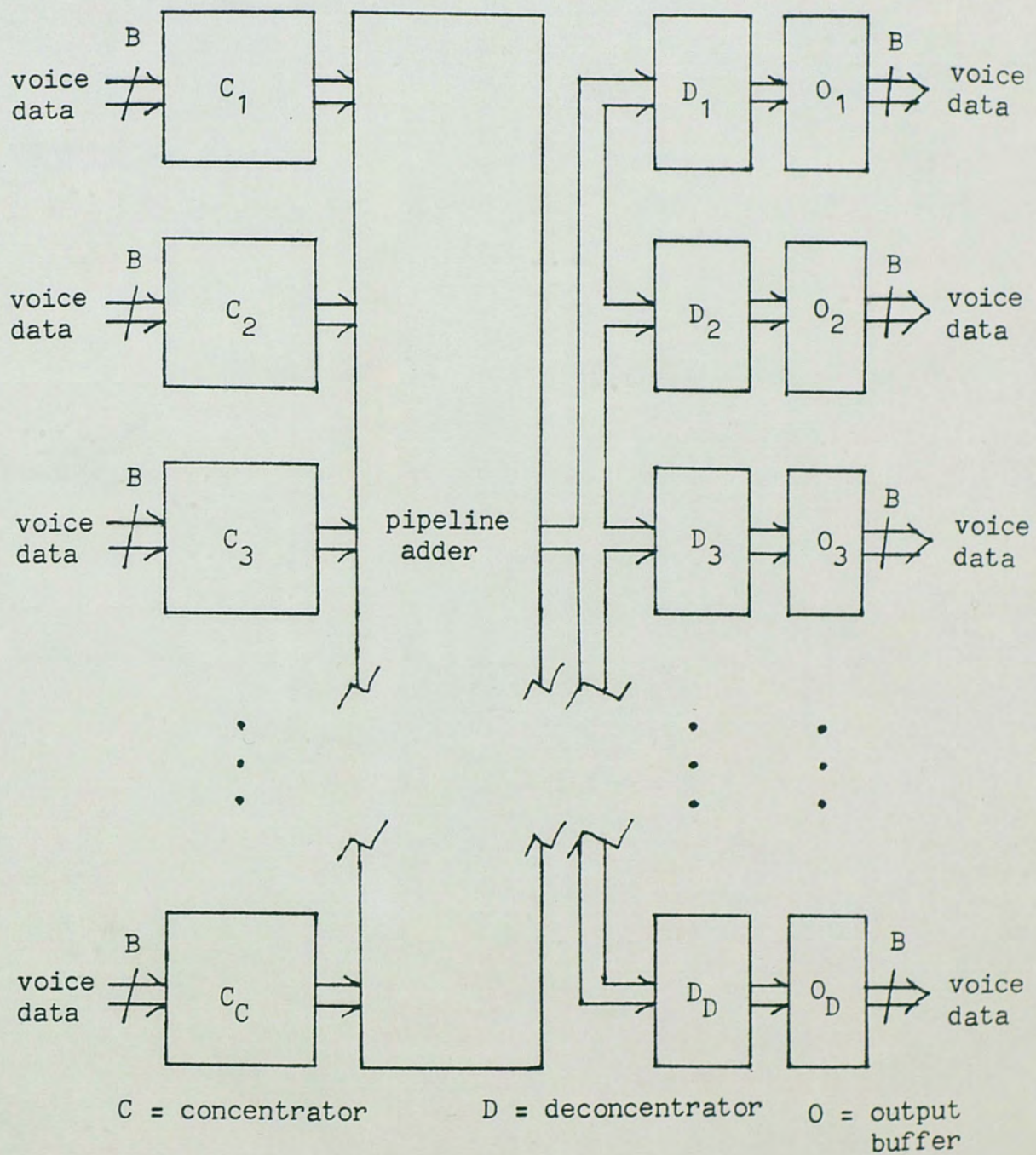


Figure 8. Processing stage interconnection.



each during a frame time. The concentrators pass their CH partial channel sums to the pipeline adder. The pipeline adder broadcasts the CH channel totals to the deconcentrators. Each of the D deconcentrators passes O user sums (sums of channels) to an output buffer. Each output buffer sends its O sums out, each made up of B bits. Data enters and leaves the switch in a parallel sequence. A new digital sample appears on the input lines every  $T/I$  seconds. Likewise, a digital output of B bits occurs every  $T/O$  seconds. Data is passed from the output of a concentrator to the input of an output buffer via sets of ping-pong memories.

#### The Concentrator

Figure 9 shows the internal architecture of a concentrator. The devices and their functions are described as follows:

- CNTR: A K bit counter (where  $2^K \geq I$ ) who's binary count will indicate which of the I user's voice data is present at the parallel input. A low to high transition on line COUNT will advance the counter after a delay of CNTRdel seconds.
- CR: A K bit register. The register stores the count on a low to high transition on line CLOAD. The output is valid after CRdel seconds.
- TA: A  $2^K \times (M+1)$  (where  $2^M \geq CH$ ) ping-pong memory. The talk address memory maps a user's count to the channel number that he is active on. The TA's ping line is not toggled each frame time; it is changed by master control. The memory has a delay of TAdel seconds. The extra bit is low if a user is the first of the I users to be speaking on his channel.



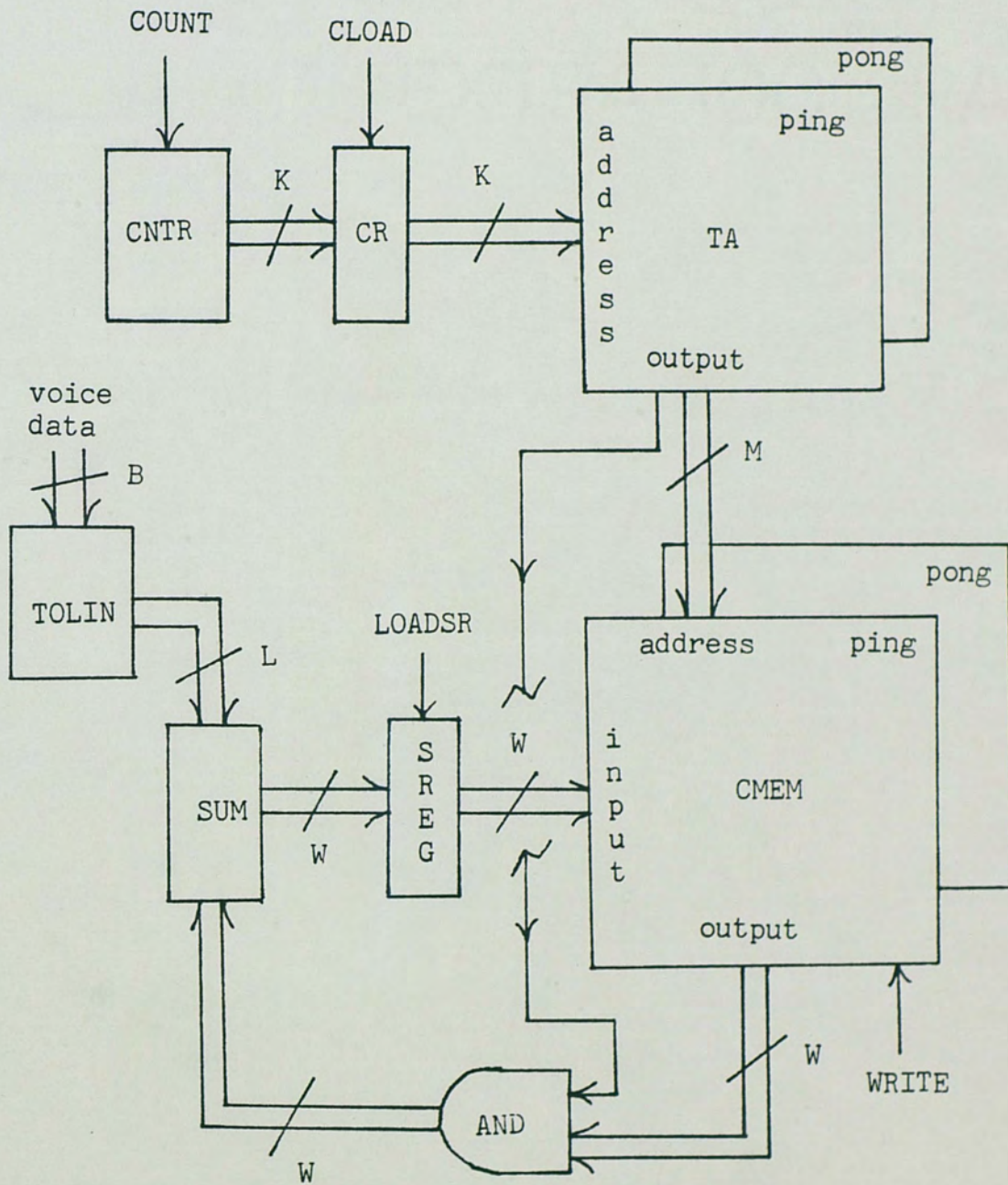


Figure 9. Internal concentrator detail.



- CMEM: A  $2^M \times W$  (Where  $W$  is given by [1]) ping-pong channel memory that stores the partial channel sums as they are accumulated. A high signal on line WRITE will cause the data on the input lines to be fed to the output lines (because of the ping-pong memory structure) and to be written to memory. CMEM has a delay of CMEMdel seconds.
- AND:  $W$  and gates. The and gates will pass either a zero or CMEM's output to the adder. They each have a delay of ANDdel seconds.
- SUM: A  $W$  bit adder. The adder adds the linearized incoming voice sample and the previous channel partial sum after a delay of SUMdel seconds.
- SREG: A  $W$  bit register. A low to high transition on line LOADSR will cause the output of SUM to be stored after a delay of SREGdel. The register is needed to isolate the input and the output of CMEM, since the output lines will change when a write is occurring.
- TOLIN: A decoder that inputs the  $B$  bit voice sample code and outputs the  $L$  bit linear equivalent. Realizations could use combinational logic or a ROM (read only memory) look-up table. It has a delay of TOLINdel seconds.

As implied in the description of CMEM,  $V$  distinct voices are to be able to be added to one memory location without overflow. The required word size is  $W$  bits, given by Equation [1] as a function of  $L$  and  $V$ .

Among the incoming  $I$  users, those speaking on the same channel will be accumulated in one location in CMEM. When the frame of data has



completely finished, the channel partial sums will be ping-ponged to the pipeline adder. The pipeline adder only reads those partial sums, so the same sums are switched back to the concentrator after two frame times. The and gates are needed for when a new sum is being started in a location, and thus, a zero will be switched to the adder. The first accumulated sum will be exactly the first user's data who is on that channel.

The talk address map (TA) is a ping-pong memory so new routing information can be switched into the system periodically. Master control processes incoming signalling information by updating one of its own tables. That table (or a portion of it) is periodically loaded into the pong side of TA. The loading occurs separately for each concentrator, since the map's contents will differ for each set of users. The TA memory is ping-ponged after this operation to switch in the updated talk paths.

The sequence of events during the time a user's voice sample is valid on the input lines is described by the timing diagram of Figure 10. Of course, the user's voice sample must be linearized, added to and stored in memory in T/I seconds. Therefore, the time described by the critical time given by

$$TC = CR_{del} + TA_{del} + CMEM_{del} + AND_{del} + SUM_{del} + SREG_{del} + CMEM_{del} \quad [2]$$

is less than or equal to T/I. When actual parts are chosen, slack might have to be added in to account for the set up time of the registers, recovery time of the memory, bus speeds and any other delays not accounted for in the architectural description. Equation [2] can easily be modified to account for additional delays. Also, a system clock will



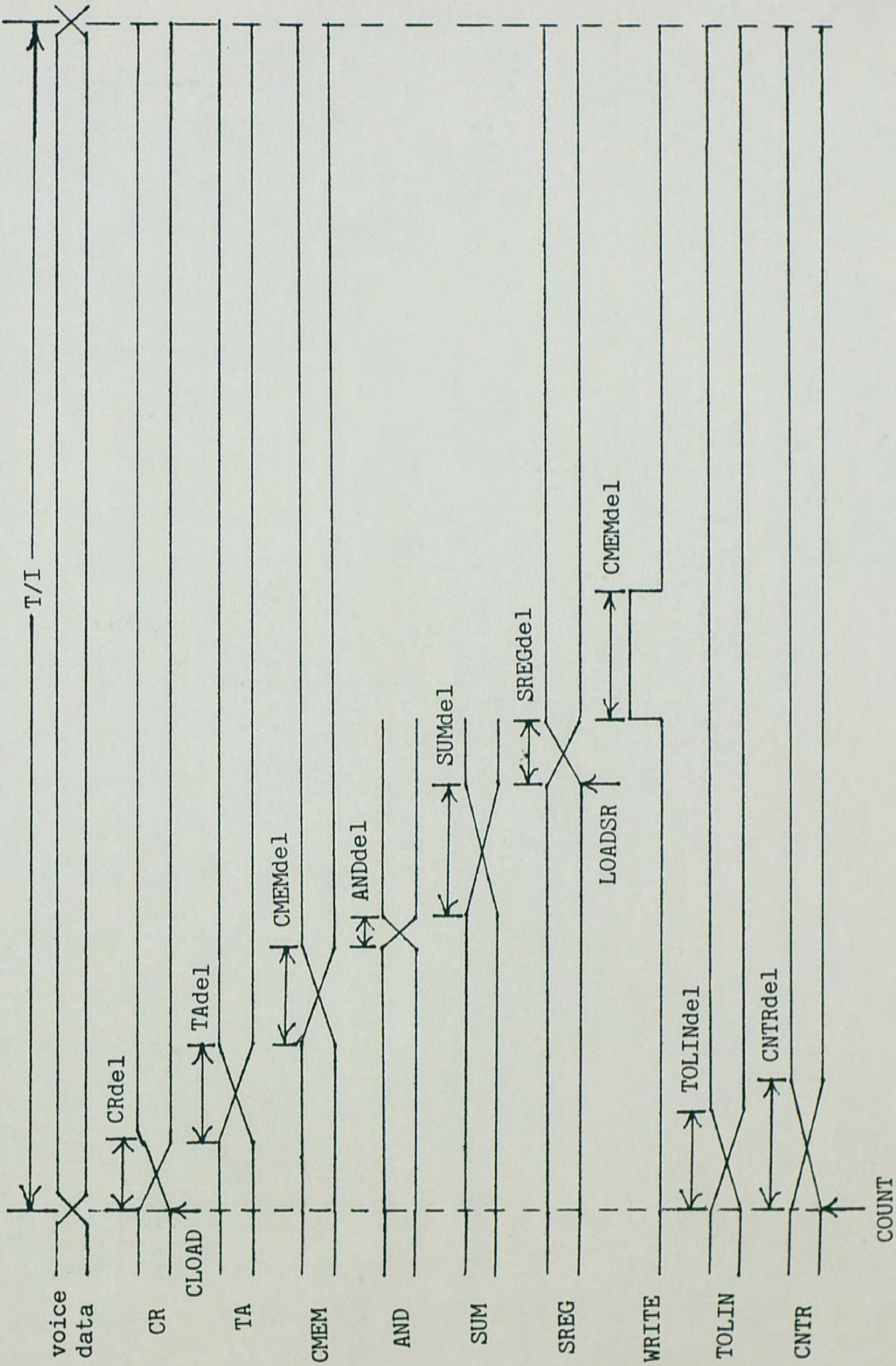


Figure 10. Concentrator ideal timing.



generate the load and write signals. The signals can only occur on edges in step with the clock signal edges. Extra time may have to be allowed for to take care of these "quantization" errors. In other words, the control signal transitions have been shown in Figure 10 to take place at the optimum times. Once the clock rate is set, these transitions will shift, so they will take place after a clock edge. There are two assumptions implied by the timing of Figure 10. One is that  $TOLINdel$  is smaller than the time from when CLOAD occurs and when the addition starts taking place. The other is that  $CNTRdel$  will be smaller than the critical time. Both assumptions are valid because  $TOLINdel$  should be close to a memory delay at most (depending on TOLIN's realization), and  $CNTRdel$  should be comparable to an addition delay. If, after an actual parts selection is made and these assumptions are invalid, a new timing diagram will yield the correct values for I and C.

The events described by Figure 10 occur I times during a frame period T. CNTR is upcounted, anticipating the next user coming in. The count is loaded into CR so CNTR may be upcounted as soon as possible. The CR register was added because typically  $CNTRdel \geq CRdel$ . The count in CR addresses the talk address map (TA). TA addresses the channel memory (CMEM) with the channel address the particular user is speaking on. An extra bit in the TA map either passes or zeroes the channel partial sum read out of CMEM. If the user is the first in his set to speak on that channel, the extra bit will be low, causing zero to be added to his sample. After the adder sums the linear equivalent of the user's voice and the partial channel sum, the result is loaded into SREG. Then the



result is stored back in CMEM. The cycle repeats for the next user. At the end of a frame, the new TA map could be switched in (if master control has finished updating it) and the counter will be reset.

Note that a user's data is added to a channel regardless of whether he is actually speaking or not. Ideally, his voice sample would be zero if he was not speaking. Since microphones will pick up non-zero background noise, all users should have a press-to-talk (PTT) type switch that must be activated in order for the user to be heard. A voice operated switch (VOX) is also a possible solution of eliminating background noise, but the VOX tends to clip off the beginning of a user's conversation. The idea is that there must be a device to keep the user's background noise out of the system if he is not speaking.

When the switch system is to be sized, the total number of users (TU), the total number of conference channels (CH) and the number of distinct voices to be heard (V) will determine the number of concentrators (C) and their internal size. V and B will determine some of the hardware delays, by setting W. SUMdel will then be known once an actual part is chosen, for example. A delay that is not known yet is CNTRdel, because the number of users serviced by a concentrator (and thus K) is not known. However, TC can still be found; a close estimate of TAdel can be made even though the length of TA is not known. Then once C and K are found from TC, the counter delay assumption can be checked. Also, the length of TA is known ( $2^K$ ), so its actual delay can replace the one assumed, which will generate a new TC. When the value of C does not change after an iteration, the concentrator has been sized.



The concentrator structure is then flexible to handle any reasonable requirements of operation. The structure is non-blocking to this point because all users can be serviced, whether they are using the system or not.

### The Pipeline Adder

Figure 11 shows the internal architecture of the pipeline adder. Note that the pipeline adder is spread across the entire switch; a chain of adders are on the side of the concentrators, and a bus broadcasts to all the deconcentrators. The devices and their functions are described as follows:

- CHCNT: A M bit counter whose binary count indicates which channel partial sum is about to be introduced into the pipeline. Line CHCOUNT advances CHCNT after a delay of CHCNTdel seconds.
- CNTREG: A M bit register that stores the channel count. CNTREG adds the same kind of timing margin that register CR did in the concentrator. When line CNLOAD is raised, the input is stored after CNTREGdel seconds.
- CMEM: The pong side of the ping-pong memory in each concentrator, with identical dimensions and delays. The channel partial sums are stored here.
- CHREG: A W bit register that stores the channel partial sum that is about to be summed into the pipeline. When line CHLOAD is raised, the input is stored after CHREGdel seconds.
- ADD: A W bit adder that forms the sum of the previous concentrators' total of partial sums and the local concentrator's partial sum. ADD has a delay of ADDdel seconds.



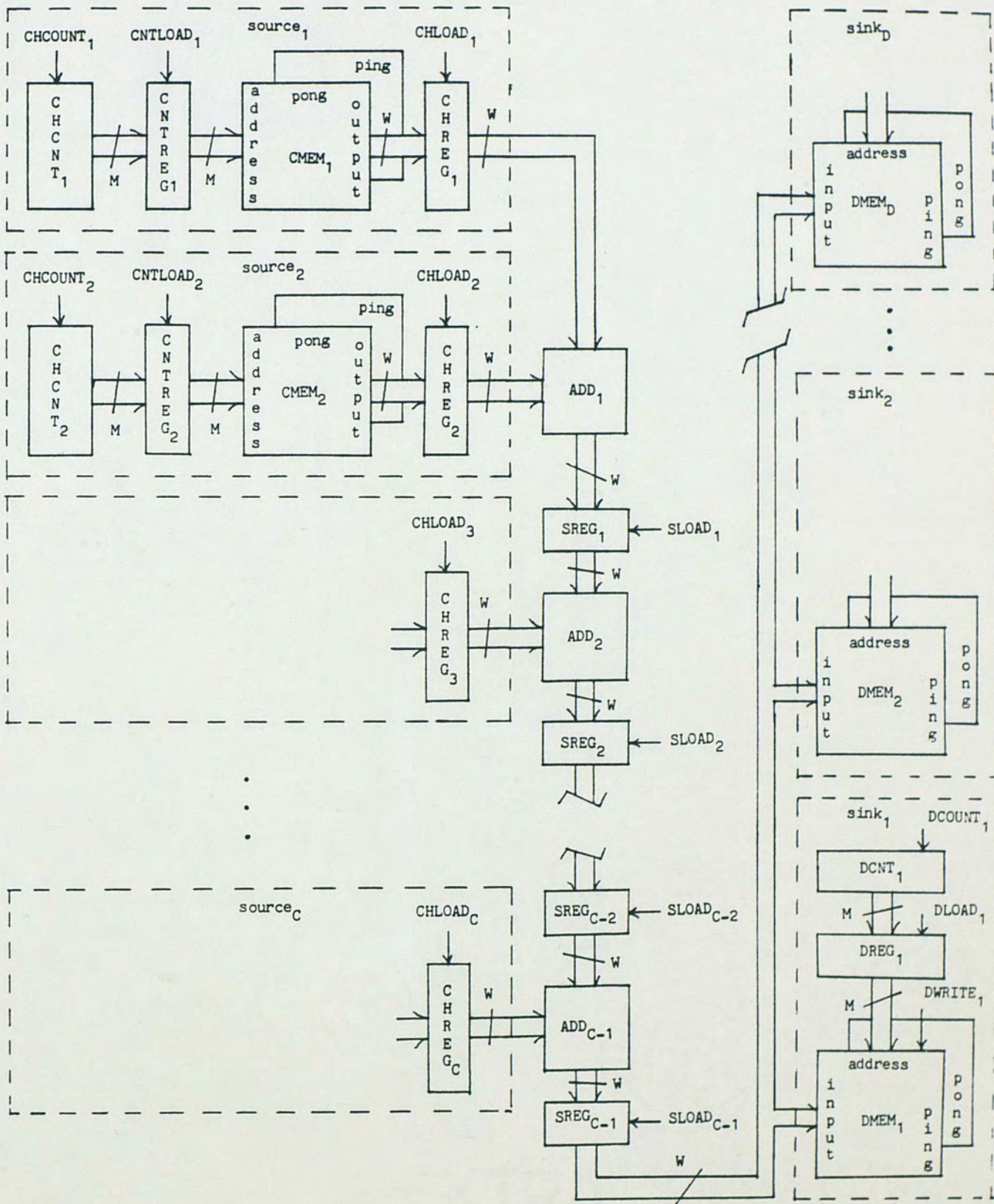


Figure 11. Internal pipeline adder detail.



- SREG: A W bit register that stores the channel partial sum from all the previous concentrators. When line SLOAD is raised, the input is stored after SREGdel seconds.
- DMEM: A ping-pong memory with identical dimensions as that of CMEM. DMEM stores the final channel sums. It has a delay of DMEMdel seconds. Line DWRITE controls the write cycle. Note that the pong side of DMEM is part of a deconcentrator, and a deconcentrator may have one or more identical DMEM's each.
- DREG: A M bit register with the same purpose as CNTREG. The input is stored after a delay of DREGdel seconds when clocked by DLOAD.
- DCNT: A M bit counter who's count indicates which channel's final sum is coming off the pipeline. It upcounts after a delay of DCNTdel seconds when clocked by line DCOUNT.

The function of the pipeline adder is to total all the partial channel sums from all the concentrators and broadcast those totals to all the deconcentrators. The broadcasting is done by the long bus on the right of Figure 11. At the end of the frame period, the deconcentrators receive the final sums as DMEM ping-pongs.

The registers around the adders in the pipeline are to allow a large time savings. Without them, C-1 addition delays would have to pass before each channel total was ready to be written. When the registers are included, the time between when totals are completed is just a single add delay and a register delay. The sums are created in a pipeline fashion, and the clocking of the registers around the adders



control the flow. The timing of Figure 12 shows how the pipe flow is built up. There is an overhead in terms of the delay that occurs while DMEM is awaiting the first sum. Afterwards, the pipeline delivers the remaining channel sums in sequence with an add delay and a register delay separating them. Note that assumptions are implied again, as only relative times have been estimated. Parts with the same operation have been shown to have similar delays. Register delays have been shown to be shorter than memory delays and memory delays are shorter than addition delays. An actual parts selection will verify the assumptions.

The first two sources of partial sums (the pong side of  $C_1$  and  $C_2$ ) are clocked together, such that the channel counters are upcounted at the same time and the channel partial sums are fed to the first adder at the same time. The third source of partial sums (the pong side of  $C_3$ ) is started at a point in time where its partial channel sums are loaded into  $CHREG_3$  at the same time as  $SREG_2$  is loaded. After another addition delay,  $SREG_2$  is loaded with the partial sum of the first channel from the first three concentrators. At the same time,  $SREG_1$  is loaded with the partial sum of the second channel from concentrators 1 and 2. The partial sum sources are started in sequence until finally the last source drives  $ADD_{C-1}$  with its first channel partial sum. At this time,  $SREG_{C-2}$  holds the channel partial sum of the first channel from all the other sources. After an add delay, the complete channel sum is loaded into  $SREG_{C-1}$ .  $SREG_{C-1}$  drives the broadcast bus with the channel total. It is then written to DMEM in the channel location indicated by DREG. While DMEM is being written to,  $ADD_{C-1}$  is forming the final sum of the second channel. The process continues until the



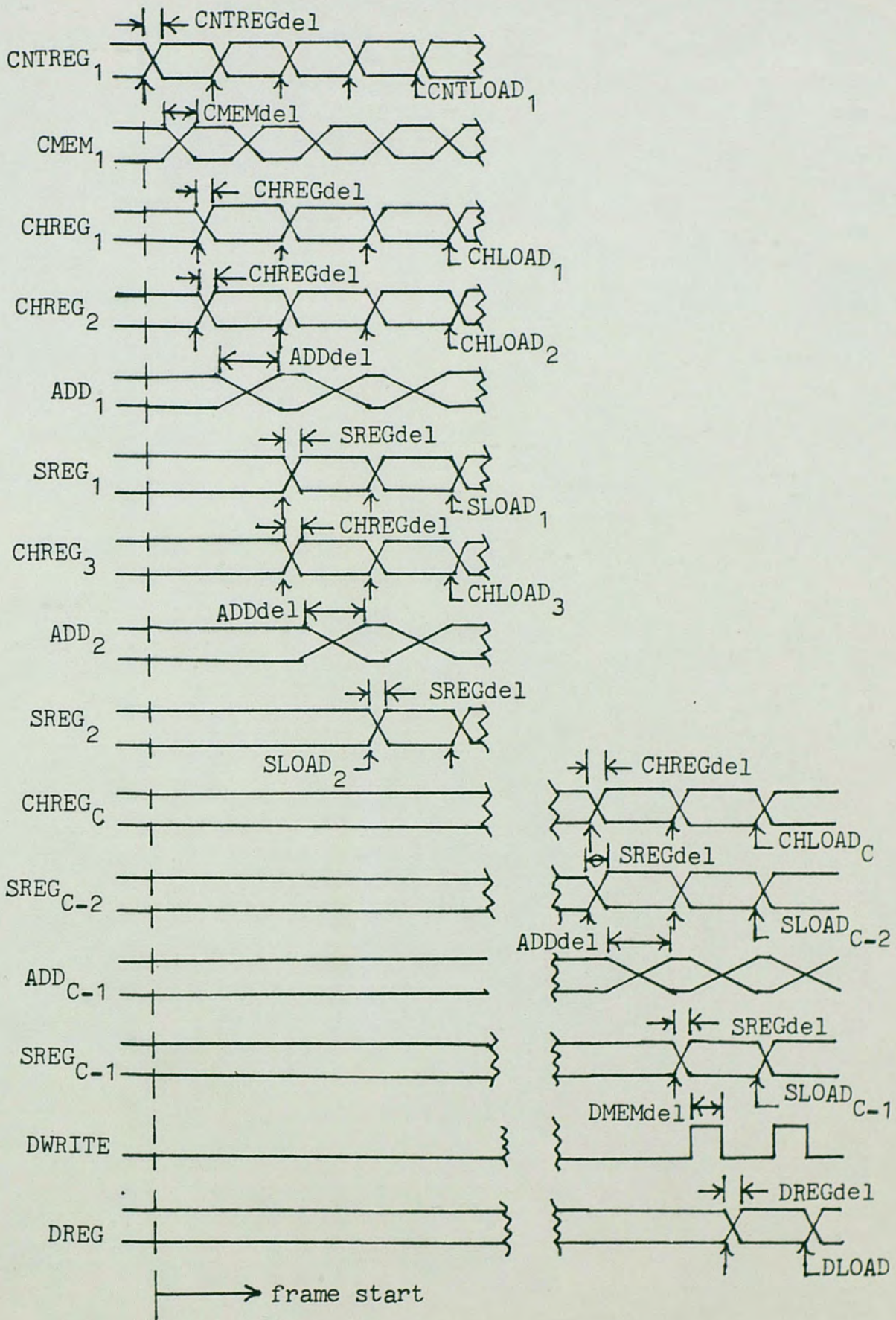


Figure 12. Pipeline adder ideal timing.



last channel total is written into DMEM. The frame period then ends and the channel totals are switched to the deconcentrators.

The timing of Figure 12 illustrates the sequence of events. Only the critical timing has been shown for clarity. Because of the registers that store the channel counts, the counters can be upcounted as soon as their old counts have been stored. The same idea prevails here as in the concentrator; typically the counter delay is more than a register loading delay, but less than a memory delay and a register delay. As with the concentrator, when actual parts and a clock rate are chosen, another timing diagram must be drawn to absolutely size the pipeline adder.

The timing is bounded by the fact that all the channel totals must be stored before the frame period  $T$  is over. Thus, the constraining Equation is

$$T \geq \text{CNTREGdel} + \text{CMEMdel} + \text{CHREGdel} + (C-1) * (\text{ADDdel} + \text{SREGdel}) \\ + (\text{CH}-1) * (\text{ADDdel} + \text{SREGdel}) + \text{DMEMdel}. \quad [3]$$

The derivation of Equation [3] appears in appendix B. Since  $C$  is already known from when the concentrator was sized, parts can be chosen and the Equation can be checked.

As with the concentrators, any other delays (such as bus delay) would have to be added to the timing in Figure 12. If the total number of conference channels ( $\text{CH}$ ) is large, Equation [3] may not be satisfied. If so,  $\text{CMEM}$  could be divided into  $P$  pages and the structure of the pipeline adder would be repeated  $P$  times.  $\text{DMEM}$  would also have to be divided into  $P$  pages. All counters could drive the memory pages at once. The hardware that would be duplicated are the adders and partial



sum registers. Each pipeline adder would work on its own page of all the CMEMS and create CH/P channel totals. The timing would be the same, as the page sections would all be running together. Equation [3] would change to

$$T \geq \text{CNTREGdel} + \text{CMEMdel} + \text{CHREGdel} + (C-1) * (\text{ADDdel} + \text{SREGdel}) \\ + [(CH/P)-1] * (\text{ADDdel} + \text{SREGdel}) + \text{DMEMdel}. \quad [4]$$

If the number of users processed by the system forces C to be so large that Equation [3] cannot be satisfied, the structure will have to be modified. The concentrators can be split into G groups with C/G concentrators in each group. G pipeline adders can process the channel addition, with one per group. Instead of storing what will again be channel partial sums in DMEM, the partial sums will be stored in G intermediate ping-pong memories (IM). Figure 13 shows basically how the structure must be expanded, where G=2 and C=3. The pong sides of the intermediate memories will be processed by another pipeline adder that will store the final sums into the deconcentrator memories. Since another ping-pong memory stage has been added, an additional frame period delay has been added to the switch. Equation [3] can be changed to show the affect of dividing the concentrators into groups:

$$T \geq \text{CNTREGdel} + \text{CMEMdel} + \text{CHREGdel} + [(C/G)-1] * (\text{ADDdel} + \text{SREGdel}) \\ + (CH-1) * (\text{ADDdel} + \text{SREGdel}) + \text{DMEMdel}. \quad [5]$$

If the number of concentrators is so large that the number of intermediate memories is too large for one pipeline adder, the pong side of the IMs can be divided further. Then more than one pipeline adder would run in parallel, driving a second set of intermediate memories. The idea can expand in this direction, adding more delay to



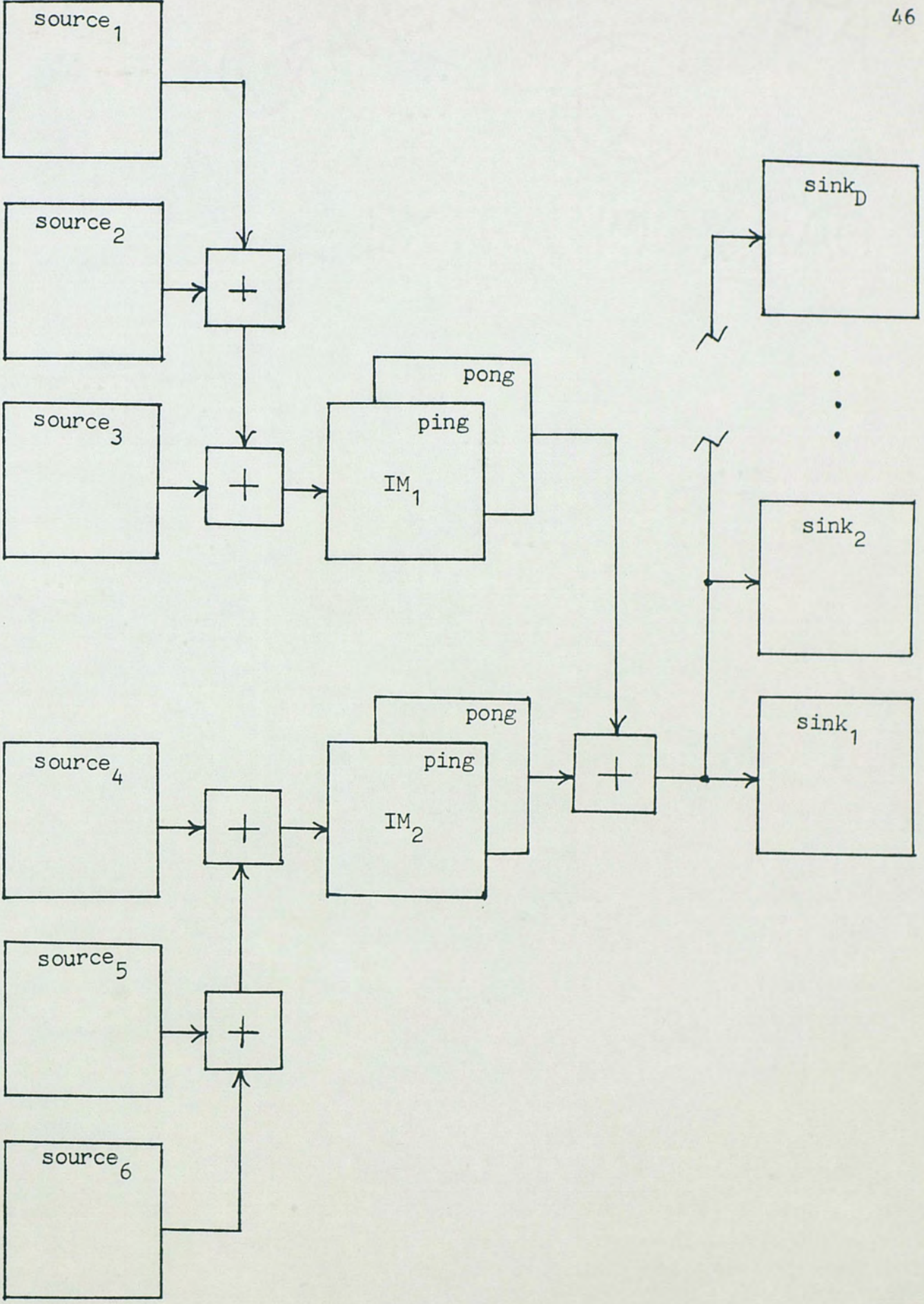


Figure 13. Pipeline adder expansion for many concentrators.



the switch until the channels can finally be totalled.

Thus, the structure of the pipeline adder can expand in either direction to accommodate more channels or more users or both. Finally, Equation [3] can include the possibility of expansion in both directions:

$$T \geq \text{CNTREGdel} + \text{CMEMdel} + \text{CHREGdel} + [(C/G) - 1] * (\text{ADDdel} + \text{SREGdel}) \\ + [(CH/P) - 1] * (\text{ADDdel} + \text{SREGdel}) + \text{DMEMdel}. \quad [6]$$

the switch delay will be longer if more users are processed, but delay is assumed not to be significant as mentioned earlier. The switch is still non-blocking as all channel partial sums from all the concentrators have been added to one location.

#### The Deconcentrator

A discussion of the deconcentrator is needed before the architecture is presented. The function of the deconcentrator is to create user sums which will be the sums of the channels a user is listening to. The channel total sums are passed to the deconcentrators from the pipeline adder. As with the concentrator, each deconcentrator will service a group of users. Each user has sent signalling information to the system, informing master control which channels he desires to hear. He sends additional information so master control can determine what relative volume level each of these channels has before they are added together.

The volume adjustment must be done on a linearly encoded digital word of  $W$  bits. Since the possibility of no adjustment exists, the result of the adjusted word must have at least  $W$  bits. To amplify a voltage level represented by a digital word and maintain the same quan-



tization levels, the word size would have to increase. The basic word size,  $W$ , was already set based on how many "full volume" voices could be added to one memory location. It seems reasonable then that the volume adjustment should only involve an attenuation of the digital word. Then the basic bus width ( $W$ ) would stay the same. Since analog amplification will be available at the output device, the volume adjust done here should be all relative level adjustments of each channel with respect to the others. Thus, the volume adjust function will take in  $W$  bits and  $VC$  volume control bits and produce  $W$  adjusted bits. With  $VC$  control bits, a maximum of  $2^{VC}$  different levels are possible.

One possible realization of the volume adjust function is with a  $2^{W+VC} \times W$  ROM look-up table. However, as an example, the Codec 8 bit code mentioned earlier has a 14 bit (sign and magnitude) linear equivalent (MacDonald, 1982). If 4 voices are allowed to occur at once, then Equation [1] gives  $W=16$ . With only one volume control bit, the ROM dimensions would be 128K words long and 16 bits wide, which is impractical.

An alternative to the ROM realization of volume adjust is to shift the digital word. Shifting the word  $S$  positions to the right corresponds to a division by  $2^S$ . Since the voice samples are in two's complement notation, the shift will allow the sign bit to extend over to the most significant positions. Shifting can be done with multiplexers. A  $W$  bit volume adjust on  $W$  bits with  $VC$  control bits would require  $W-1$   $2^{VC}$ -to-1 multiplexers. The multiplexing realization of the volume adjust is more reasonable than the ROM realization because of component size and lower typical delay. Note that since a voltage



signal is being divided in half, there will be a 6 dB drop in the signal with each shift.

Once the volume adjustment is performed, the channel can be added to the other adjusted channels that a user is going to hear. The user totals (sums of channels) could be constructed as the user partial sums were created in a concentrator. A location in memory would be added to, channel by channel. Assuming an approximate timing of a concentrator, a deconcentrator could perform  $I$  sums in a frame period  $T$ . If each user could listen to  $H$  channels, each deconcentrator would service  $I/H$  users and there would be  $C \cdot H$  deconcentrators in the switch system. A typical conferencing net might allow a user to listen to another channel besides his own. If one person wanted to be able to be heard by all users, he could speak on a channel that would be added to all user's channels. With this limited example, the user would be hearing 3 channels, resulting in  $3C$  deconcentrators in the system. The architectural approach will be referred to as the accumulator-type deconcentrator.

The deconcentrator structure of such a general organization where every user could listen to  $H$  channels could be done another way. The channel memory (DMEM) could be repeated  $H$  times in each deconcentrator. Then a tree of adders could perform the sum all at once. The tree would have  $H-1$  adders and have a delay of

$$\text{LOG}_2(H) \cdot (W \text{ bit addition delay}), \quad [7]$$

where the "LOG"() function is rounded up to the next highest integer.

Such a structure with tree addition is shown in Figure 14. The devices and their functions are as follows:



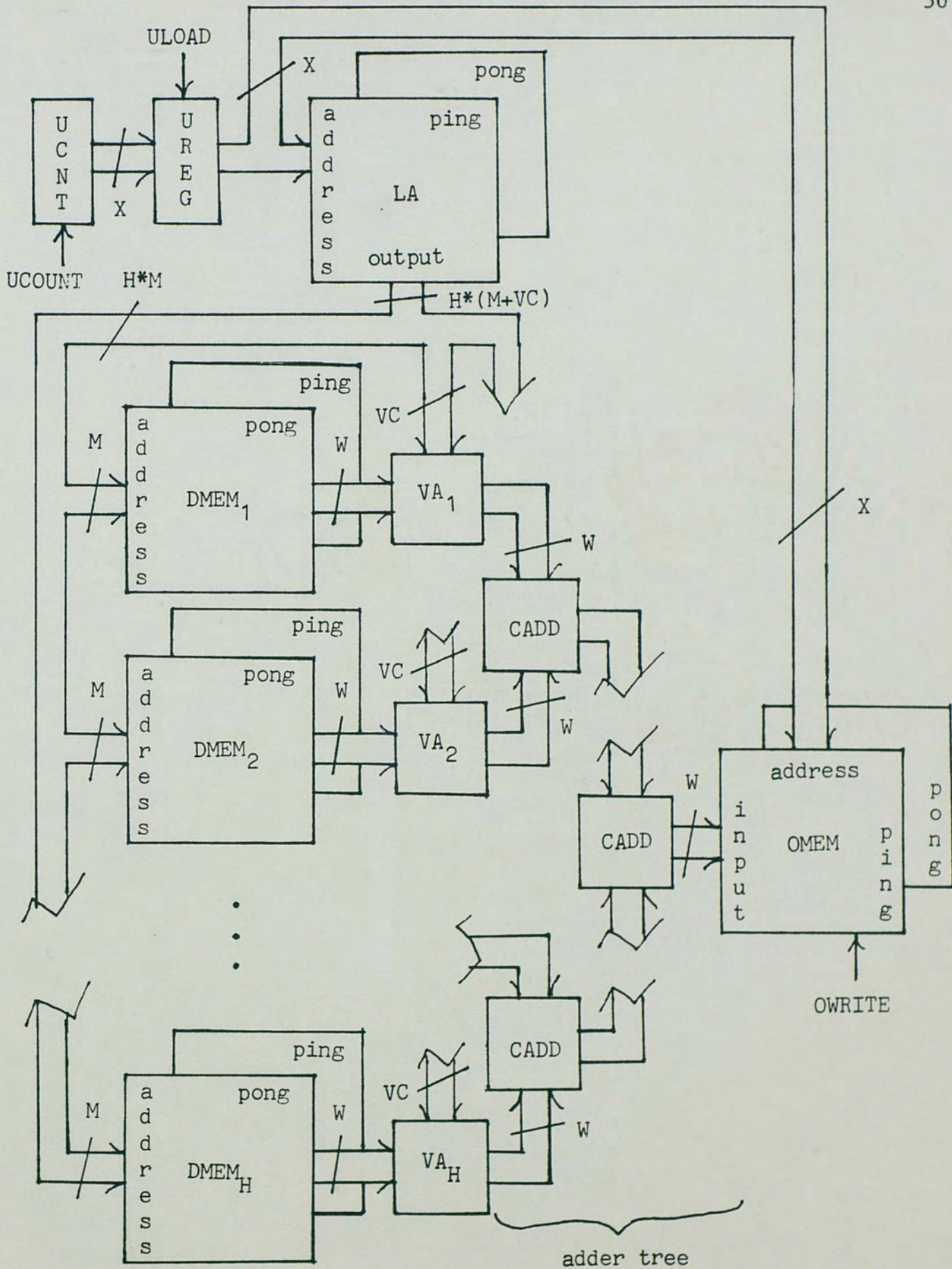


Figure 14. Internal parallel-type deconcentrator detail.



- UCNT: A  $X$  bit counter (where  $2^X \geq 0$ ) who's count will indicate which user's sum is being constructed. UNCT is upcounted by UCOUNT after UNCTdel seconds.
- UREG: A  $X$  bit register. UREG stores the input count UREGdel seconds after a leading edge on ULOAD. UREG provides a timing margin like CR does in the concentrator.
- LA: A  $2^X \times H \times (M+VC)$  ping-pong memory. The listen address memory indicates the addresses of the  $H$  channels the current user wants to hear along with  $VC$  volume adjust bits for each. The LA's ping line is not toggled each frame time. As with the talk address map in the concentrator, master control determines when the pong side of LA is to be loaded and switched in. LA has a delay of LAdel seconds.
- DMEM: The pong side of ping-pong memory DMEM with identical dimensions and delay. The channel total sums are stored here.
- VA: Volume adjust unit. As discussed above, the VA takes the channel's  $W$  bit total and provides a  $W$  bit output adjusted according to the  $VC$  adjust bits. VA has a delay of VAdel seconds.
- CADD: A  $W$  bit adder that helps construct the user's total. Each CADD has a delay of CADDdel seconds. The entire tree has a delay of TREEdel seconds, given by Equation [7].
- OMEM: A  $2^X \times W$  ping-pong memory. OMEM stores the user totals when they are completed by the adder tree. OWRITE controls the write cycle of the memory. OMEM's delay is OMEMdel seconds.



The timing of Figure 15 shows the sequence of events. UREG stores UCNT's output and addresses OMEM and LA. All the DMEMs are addressed at once. The channel contents are then volume adjusted according to the adjust bits in LA. The adjusted channels are then passed to the tree of adders. When the sum is completed, it is stored in OMEM.

The user's sum must be created and stored in  $T/O$  seconds. The deconcentrator critical time is given by

$$TD = UREGdel + LAdel + DMEMdel + VAdel + TREEdel + OMEMdel, \quad [8]$$

which is less than or equal to  $T/O$ . As before, slack might have to be added in to account for delays not explicitly shown. As with the concentrator,  $X$  is not known, but the register and memory delays can be estimated. When  $TD$  is found,  $X$  will be known (from 0) and  $TD$  can be checked when the actual delays are then known. Again, when actual parts are chosen, another timing diagram should be drawn to verify time  $TD$ .

The sequence of events in Figure 15 occurs  $O$  times in the period  $T$ , for each of the  $D$  deconcentrators. Note again that  $D*O = TU$ . At the end of each frame period, the counter is reset and OMEM, full of the linear user sums, is ping-ponged to the output buffer. If master control has completed updating the LA map, new listening paths can be switched in also. This architectural approach is a parallel-type deconcentrator.

As can be seen, the architecture in Figure 14 gets quite large if  $H$  is large because of the  $H$  sets of DMEM's that each deconcentrator has. However, the conferencing structure may not require  $H$  to be large for all users. Imagine groups of users on different channels. They



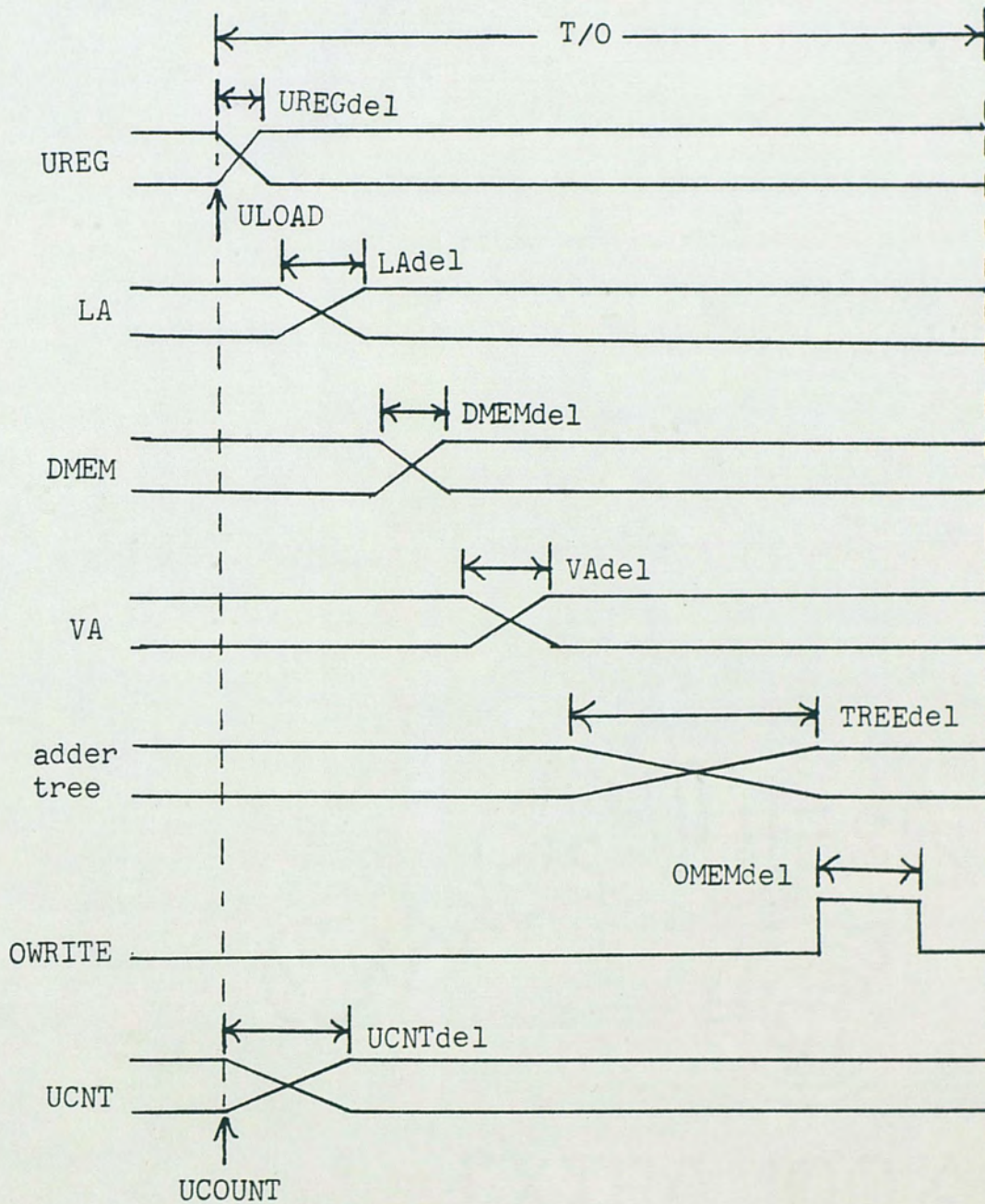


Figure 15. Parallel-type deconcentrator ideal timing.



might only need to hear the channel they are speaking on, the channel their supervisor is on and perhaps a broadcast channel that is normally quiet. Then for most of the users,  $H=3$ . There may be a small group of "super users" that can listen to a large number of channels. They might be able to be handled by the accumulator-type of deconcentrator. Since the pipeline adder broadcasts the channel total sums over a bus, more deconcentrators could be added easily. A complete system could be made up of a few of the accumulator-type deconcentrators serving the "super users" and many of the parallel-type deconcentrators serving the other users. Note that even though a user may only speak on one channel, two users could converse while being active (speaking) on different channels. Both user's could be listening to each other's active channel. The idea continues on to allow conversation between  $H$  users on  $H$  different channels. Note also that in the way the architecture has been set up, a user does not have to hear the channel he speaks into. The channels a user hears are solely determined by how the listen addresses are stored in the memory map (LA). Master control could easily cause a user to hear a sum of channels that did not include his active channel, if the need arises.

The switch is still expandable to this point since the number of deconcentrators can increase to handle more channels being listened to by a user. The system is still non-blocking because each user can hear all the channels that are desired.

#### The Output Buffer

The structure of the output buffer is shown in Figure 16. The components and their functions are as follows:



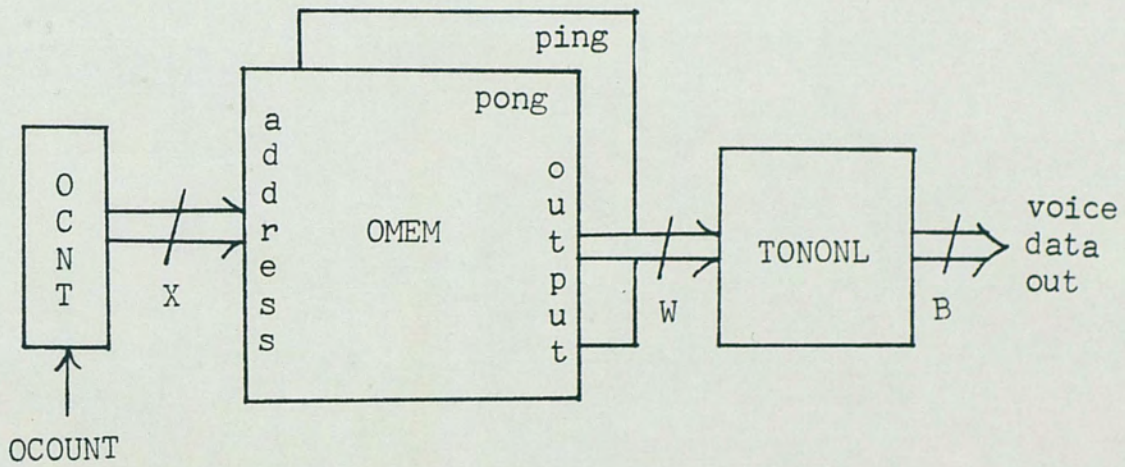


Figure 16. Internal output buffer detail.



- OCNT: A X bit counter. The output count is of the user who's final sum is to be sent out of the switch. OCNT is upcounted by line OCOUNT after OCNTdel seconds.
- OMEM: The pong side of ping-pong memory OMEM, with the identical size and delay. The user totals are stored here.
- TONONL: A function block that converts a W bit linearly encoded voice sample back to the original input format of B bits. It has a delay of TONONLdel seconds. Possible realizations are combinational logic or A  $2^W \times B$  ROM. The logic or the ROM's contents will be a function of the input code format. TONONL will handle the overflow that might occur if the number encoded in the W input bits is larger than the maximum allowed by L bits.

The sequence of events is described by the timing of Figure 17. The output user counter addresses the output memory. The output memory reads out the user's total and sends it to the format converter. The W bit user sum is changed to the B bit format (identical to the input) and that B bit code leaves the switch portion of the system. As described in Figure 5, the voice code will be recombined with signal data, and then the entire data frame will be demultiplexed to the users connected to the specific deconcentrator and output buffer. The output of the switch may be latched externally after the memory and format conversion delays occur. After the output code leaves the system, the counter is upcounted to address the next user total. At the end of a frame period, the counter is reset and OMEM ping-pongs in a new block of 0 user totals.



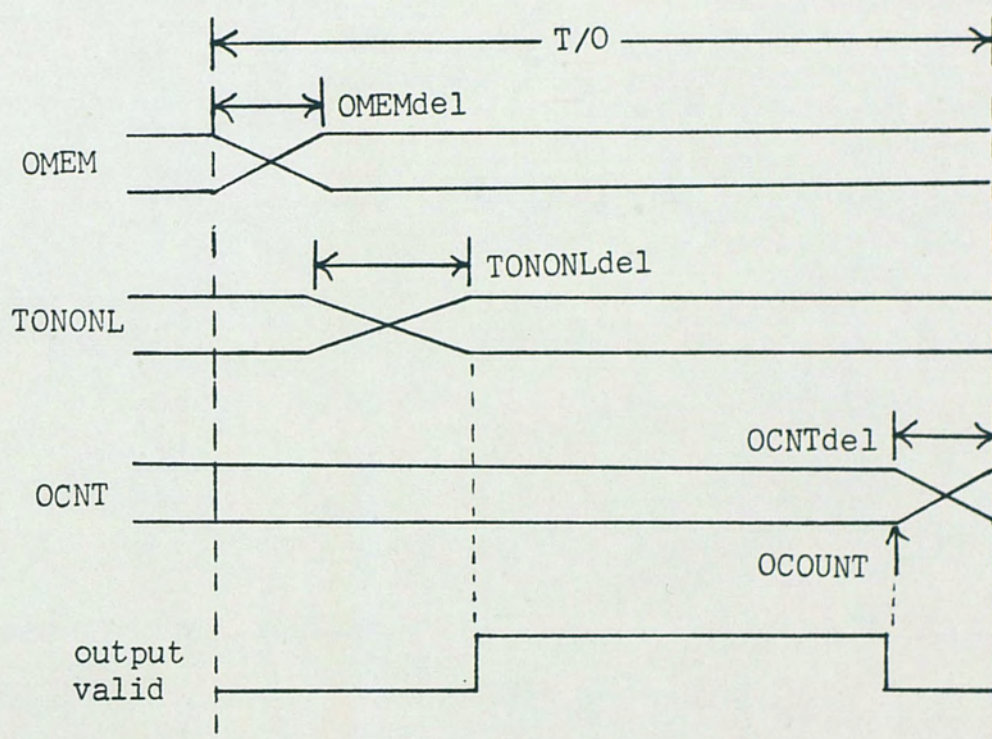


Figure 17. Output buffer ideal timing.



The sequence of events occurs  $O$  times within a frame period  $T$ . The timing of Figure 17 must then adhere to the Equation

$$T/O \geq \text{OMEMdel} + \text{TONONLdel} + \text{OCNTdel}. \quad [9]$$

Since  $O$  was fixed when the deconcentrator timing was done, this Equation should check. The basic delay of a deconcentrator was three memory delays and a long addition delay. If slower parts are chosen, the output buffer may be slower than the speed implied by Equation [9]. If so, OMEM can be divided into  $OP$  pages that the user totals will be read from and converted at the same time. The switch would then effectively have  $OP \cdot D$  outputs and  $C$  inputs, each of  $B$  bits. Since it is expected that the paging of OMEM should not be needed, a register between OCNT and OMEM was not included. When actual parts are chosen and a detailed timing analysis is done, a register at that location could provide an extra margin. No memory map is needed here since the user's output order is known by master control and the LA map in the deconcentrator was loaded accordingly.

The output buffer delivers user totals to all the users connected to it, so the switch is still non-blocking. The output buffer dimensions are controlled by how the deconcentrator is sized. If slower components than in the deconcentrator are chosen for the output buffer, the structure can still expand to handle the timing.

#### Summary

The switch portion of the conferencing digital communications system is now architecturally complete. The system parameters ( $B$ ,  $CH$ ,  $H$ ,  $T$ ,  $TU$ ,  $V$ ,  $VC$ ) and the chosen component delays will determine the size of the four processing stages. The necessary function of master



control to support this architectural description is to process the user signal data in order to update the talk and listen address maps in the concentrator and deconcentrator. The system should be investigated and sized in the order that it was presented, as the system parameters affect each stage in turn.



## CHAPTER VI

### CONCLUSIONS AND FURTHER STUDY

#### Conclusions

Chapter II described the different general schemes of circuit and packet switching. The conferencing switch architecture as presented was made totally from circuit switching ideas. The stages of the switch also act like a distributed processing system. First, the switching methods used will be analyzed and then the label "distributed processor" will be investigated.

Certainly, the conferencing switch is a circuit switch. A dedicated connection is set up from the message transmitter to the receiver for the duration of the transmission. The connection is made through a pathway in digital memories and logic switching. The receiver is a particular user of the system. The transmitter(s) are all the users that are communicating with the receiver, either on the same channel as he is, or one of the other H-1 channels he will hear. All of these channels (and thus users) are delivered to him at once as a total.

The C sets of lines coming into the switch and the O sets of lines leaving the switch (each set has B bits) have data on them that is time division multiplexed. A dedicated place in time is devoted to each user having information pass over those lines. A specific time slot on a specific incoming or outgoing B bit bus is dedicated to only one user. Because of that dedication, the switch is not a time multi-



plexed switched system. For the switch to act as a TMS system, memory sizes would remain the same, but the processing stages would have to work at least twice as fast. Suppose the total number of users were to triple after a system was sized. Then during a period  $T$ , three frames of data would arrive and leave the switch. The PING lines on the channel ping-pong memories would toggle three times during a period  $T$ . The switch would have to process at three times the rate discussed in Chapter V, with the added requirement of the memory maps being readied and switched in three times in a frame period. The maps would have to be switched that often since three patterns of switching are defined for the three groups of users that would be switched, by the same network. Even then, the three groups of users would be totally isolated from each other in their conferencing nets since each group is processed separately. Using TMS ideas to simply expand the total number of users on the same conferencing net just does not apply to the architecture done here.

Suppose that users connected to the conferencing switch are speaking in such a manner so that every user hears only a single voice. Then, just based on the inputs and outputs, the TDM switch appears to be using simple time slot interchange to achieve the talk paths. But when two users speak so that both voices are heard by one user, something more is going on. That something looks somewhat like distributed processing.

The conferencing digital switch clearly does arithmetic processing on voice data that passes through it. The processing sections are divided and repeated, with parallel parts performing the same task.



The processing sections are clearly distributed; they interact between each other with ping-pong memories. The stages are autonomous; they work alone during each frame period. There is a high level operating system (master control) that influences the entire operation of the network. Each processing section has its own operating system (done in terms of hardware controllers). The system is transparent, in that a user does not need to know which concentrator he is connected to. In the case where two types of deconcentrators are used to provide for a small group of "super users", a "super user" will only have to know where his special end communication device is. He will not be concerned which of the specific deconcentrators he is being serviced by.

These five criteria (multiplicity of components, distribution, processing autonomy, a high level master control and system transparency) almost fit the requirements for a system to be described as a distributed processing system (Enslow, 1978). The quality that is missing is that the processing stages are not dynamically reconfigurable. A concentrator could not be switched out of the system and another switched into its place. This feature could be added, with the cost of hardware delays to allow the switching to take place. Such a feature would be highly useful in the environment of communication where reliability is a major concern. If the flexibility of dynamic reconfiguration was added, the conferencing communications system could be viewed as a time slot interchange conferencing switch implemented with distributed processing.

#### Further Study

This paper has presented an expandable architecture that accomplishes the switching tasks of a non-blocking digital voice communica-



tions system. Ideal timing diagrams were presented as a guide to each processing stage's operation and constraining delay paths. Based on the constraining timing Equations, means of hardware replication and alternate structures were suggested to allow the switch to expand to handle any range of the system parameters (TU, CH, etc.).

Many tasks remain if the discussion here is further studied, and further steps are taken towards an actual implementation of such a communications system. Some are:

1. The possible affects or improvements on the architecture if dynamic reconfigurability is planned for.
2. Specific system parameters will cause a set of components to be chosen, requiring a detailed timing analysis to size the system.
3. When the system is sized, a physical size estimate is needed to check timing estimations such as bus delay, and assumptions about timing synchronization between physically separate sections.
4. How fast master control must act on a user's new signaling information and the size of the system will impact how master control is implemented (with dedicated hardware or with a general purpose processor).
5. Schemes on how the users are encoded, multiplexed and transmitted to the switch need investigation. Signal capture techniques would then have to be studied.
6. As discussed in Chapter IV, the question of what digital overflow sounds like (versus analog saturation) needs study.



## APPENDIX A

### VARIABLE DEFINITIONS

- B: The input and output voice data word size.
- C: The number of concentrators in the switch.
- CH: The number of conference channels supported by the switch.
- D: The number of deconcentrators and output buffers in the switch.
- del: Any variable with this suffix means the component delay in seconds of the part that is so suffixed.
- G: The number of groups of concentrators that each drive different pipeline adders.
- H: The number of channels that will be added together that a user will hear.
- I: The number of users input to a single concentrator.
- K: The integer that indicates the power of two that is just greater than I.
- L: The number of bits in the linear equivalent of the B bit input voice code.
- M: The integer that indicates the power of two that is just greater than CH.
- O: The number of users processed by a deconcentrator and output by an output buffer.
- OP: The number of equally sized pages that the output buffer memory (OMEM) is divided into.



- P: The number of equally sized pages that the input channel memory (CMEM) is divided into.
- T: The voice sampling period in seconds.
- TC: The critical time of one cycle of operation of a concentrator in seconds. TC is given by Equation [2].
- TD: The critical time of one cycle of operation of a deconcentrator in seconds. TD is given by Equation [8].
- V: The number of simultaneous voices speaking to one user that is guaranteed not to produce distortion.
- VC: The number of volume adjust bits that imply  $2^{VC}$  relative volume level adjustments a user can make on a channel he hears.
- W: The number of bits required to guarantee that V simultaneous voice samples can be added to one memory location without overflow. W is given by Equation [1].
- X: The integer that indicates the power of two that is just greater than 0.



APPENDIX B

DERIVATION OF EQUATION [3]

Referring to Figure 12, CH occurrences of the DWRITE signal must happen to store all of the final channel sums in the period T. The leading edge of the first DWRITE signal occurs

$$\begin{aligned} & \text{CNTREGdel}_1 + \text{CMEMdel}_1 + \text{CHREGdel}_1 + \text{ADDdel}_1 + \text{SREGdel}_1 + \text{ADDdel}_2 + \text{SREGdel}_2 \\ & + \text{ADDdel}_3 + \text{SREGdel}_3 + \dots + \text{ADDdel}_{C-1} + \text{SREGdel}_{C-1} \end{aligned}$$

seconds after the period T starts. This time can be rewritten as

$$\text{CNTREGdel} + \text{CMEMdel} + \text{CHREGdel} + (C-1) * (\text{ADDdel} + \text{SREGdel}).$$

The leading edges of the DWRITE signals occur

$$\text{ADDdel} + \text{SREGdel}$$

seconds apart, so the last leading edge occurs

$$\begin{aligned} & \text{CNTREGdel} + \text{CMEMdel} + \text{CHREGdel} + (C-1) * (\text{ADDdel} + \text{SREGdel}) \\ & + (CH-1) * (\text{ADDdel} + \text{SREGdel}) \end{aligned}$$

seconds after the period T starts. The last channel must then be written, and all the channels must be written within the period T.

This restriction implies

$$\begin{aligned} T \geq & \text{CNTREGdel} + \text{CMEMdel} + \text{CHREGdel} + (C-1) * (\text{ADDdel} + \text{SREGdel}) \\ & + (CH-1) * (\text{ADDdel} + \text{SREGdel}) + \text{DMEMdel}, \end{aligned}$$

which is identical to Equation [3].



## LIST OF REFERENCES

- Barbaris, Giulio, and Daniele Pazzaglia. "Analysis and Optimal Design of A Packet Voice Receiver." IEEE Transactions on Communications COM-28 (February, 1980):217-227.
- Dally, Theodore; Gold, Bernard; and Stephanie Seneff. "A Technique for Adaptive Voice Flow Control in Integrated Packet Networks." IEEE Transactions on Communications COM-28 (March, 1980):325-333.
- Enslow, Philip H., Jr. "What is a 'Distributed' Data Processing System?" Computer 11 (January, 1978):13-21.
- Joel, Amos E., Jr. "What is Telecommunications Circuit Switching?" Proceedings of the IEEE 65 (September, 1977):1237-1253.
- Kasson, James M. "The ROLM Computerized Branch Exchange: An Advanced Digital PBX." Computer 12 (June, 1979):24-31.
- Kleinrock, Leonard. "Principles and Lessons in Packet Communications." Proceedings of the IEEE 66 (November, 1978):1320-1329.
- MacDonald, A. N. "Digital Voice Communications System Study for Kennedy Space Center." MITRE Corporation, Bedford, Massachusetts, January, 1982.
- Pouzin, Louis, and Huber Zimmerman. "A Tutorial on Protocols." Proceedings of the IEEE 66 (November, 1978):1346-1370.
- Roberts, Lawrence G. "The Evolution of Packet Switching." Proceedings of the IEEE 66 (November, 1978):1307-1313.