A COMPUTING MEMORY

A COMPUTING MEMORY: DESIGN AND APPLICATIONS WITH SPECIAL REFERENCE TO CORRELATION

By

DAVID A. LAWRENCE, B.ENG.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

For the Degree

Master of Engineering

McMaster University

May 1970

MASTER OF ENGINEERING (1970) McMASTER UNIVERSITY
(Electrical Engineering) Hamilton, Ontario.

TITLE: A Computing Memory: Design and Applications with Special Reference to Correlation.

AUTHOR: David A. Lawrence, B. Eng. (McMaster University).

SUPERVISOR: Professor R. Kitai

NUMBER OF PAGES: ix: 142.

SCOPE AND CONTENTS:

The philosophy of parallel processing within computers is discussed and a word-organized memory array is described in which each word location includes an independent, autonomous, arithmetic and logical processor. Several examples of computations within the memory are suggested and application in the solution of potential field problems is discussed. The concept of the arithmetic memory cell is introduced and applied in a design outline for a digital instrument capable of measuring correlation functions and mean values of fluctuating voltages.

ACKNOWLEDGEMENTS

The author would like to express his sincere appreciation to Professor R. Kitai and Dr. E. Della Torre for their encouragement and guidance in the preparation of this thesis.

Thanks are also due J. C. Majithia and K. H. Siemens for the many useful discussions and suggestions.

The financial assistance provided by the Department of Electrical Engineering is gratefully acknowledged.

TABLE OF CONTENTS PAGE

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

With the advent of the monolithic integrated circuit, the new degree of freedom offered design engineers through reduced size and cost has heralded in a new era for the digital computer. No longer is it necessary to record measured results for later analysis by a large general purpose computer. Smaller, more efficient, special purpose computers can now be taken directly to the source for a considerable saving in time and cost. More recent technological advances in the fabrication of integrated circuits have led to the point where greater than 100 gates on a single chip is now possible. The development of large scale integration (L.S.I.) has created new and unique problems for the manufacturers. The circuits must be complete independent units in themselves so that the input and output connections may be minimized. It is this necessity for entire functions to be completed within a single integrated circuit that exemplifies the need for cellular system design. A digital system will be an assemblage of cells or blocks each of which is independent within itself, and it is the duty of the system designer to suggest of what these cells must consist. It is the proposal

of such a block, the computing memory, that is the subject of this thesis.

The computing memory is essentially a word organized storage device with sufficient gating included with each memory location so that simple arithmetic operations and logical functions can be performed entirely within the confines of the word location. The autonomous nature of each memory element permits highly parallel processing. Whereas a conventional central processor must approach identical operations on many items in a large data base in a sequencial manner, the computing memory could simultaneously execute the same operations. As a special peripheral unit to a conventional computer, the computing memory would then free the central processor of the menial, repetitive operations encountered in many problems. In other applications, the computing memory may even replace a central processor and its peripherals.

Considerable investigation of the concept of the computing memory has been done in recent years. Perhaps the first efforts at including logic capabilities within the memory block occurred with the content-addressable or associative memory.[1] Such a memory was built by Rux[2] using glass delay lines as the storage unit. Although the memory was an independent self-administrating block it still relied on a central control unit to sequentially search the 2048 word memory every 100 microseconds. A qualitative study of

the relative advantages of the parallel and sequential computers along with design concepts and software considerations was made by Koczela and Wang.[3] Huttenhoff and Shively[4] proposed a computer system composed of perhaps thousands of independent blocks each of which possessed the capabilities of storage, arithmetic, and input-output interfacing. The acceptance of such a system arrangement would necessarily have to coincide with the development of new software packages that would break down a large problem into subprograms that could be simultaneously executed in each of the computer blocks. The principle of the block oriented computer was further extended by Campeau[5] in a paper that draws a distinction between multiprocessors and array processors. Whereas the multiprocessor concept would be similar to that of Huttenhoff and Shively, the array processor approach would have the arithmetic capability associated with each and every word in the memory, permitting the cellular redundancy desirable with the imperfect yield inherent in L.S.I. circuit fabrication. An example of a cellular logic-in-memory array is suggested by Kautz[6] to solve the problem of data sorting. A device is proposed that will arrange the data stored within it in either ascending or descending order of magnitude without administration from a central processor, effectively leaving this processor free to proceed to other operations.

The computing memory described in this thesis is

most comparable to the array processors of Campeau. The array is composed of identical cells and only the interconnection lines between them distinguishes one from the other. The arithmetic operations of addition, subtraction, multiplication, and division can be simultaneously executed between many pairs of cells, and in addition, shifting of cell contents within the confines of the array is permitted. Since the entire range of capabilities is a part of every cell and each is autonomous, then the elements may be termed universal arithmetic memory cells. An array composed of these cells was built and tested and found to perform well within the concepts of the design. Solution of a potential field problem by the method of relaxation is presented as an example of a problem most efficiently approached by a highly parallel computer.

The philosophy of the arithmetic memory cell is not to be restricted to application in a memory array for a computer. In any situation where a large number of identical operations are to be performed on unique groups of data all of which become available at once, the arithmetic elements enjoy several advantages. As a case in point, a design proposal for a digital instrument for the measurement of correlation function is included in the second part of this thesis. Correlation was chosen because of its general familiarity and usefulness in noise analysis and in extraction of a signal from a noisy source. The major part of the

digital correlator was found to be the arithmetic and storage sections, and because of the adaptability to array processing, it is dealt with in considerable detail.

Early instruments for the measurement of correlation function were analogue in nature and incorporated a variety of optical, mechanical and electronic methods.[7] The advantages of digital processing for slowly fluctuating waves has been recognized and in more recent years the digital computer has been utilized in a number of ways. The inconvenience involved in recording data for later processing by a general purpose computer was recognized, however, and special purpose computers as an integral part of the measuring instrument evolved. The first commercially available machines such as the Princeton Applied Research Corporation Model 100 Signal Correlator were essentially hybrid devices. In this instrument, one channel is converted to a series of binary pulses, random in nature, but with a probability of being on equal to the normalized amplitude of the input waveform. Effective time delay of this channel is accomplished through a 100 stage shift register and at the output of each stage a hybrid multiplier is used to multiply the delayed pulse train by the analogue input of the second channel. The voltage product of the multiplication is then used to charge capacitive storage elements through low-pass filters.

The principle of the probability computer[8] was further exploited by Jespers, Chu, and Fettweis,[9] in their method for computing correlation functions. Both inputs to their correlator are converted to a series of binary pulses and multiplication is accomplished in an exclusive or gate. Averaging of the gate output is performed in a conventional up-down counter so that the contents of the counter contain the digital coded value of correlation function at the completion of the measuring run.

In more recent years the techniques of conventional computer processing have been applied to the measurement of correlation function. Kitai and Masuko[10] describe an instrument that initially samples and quantises the input waveform into a binary code and then employs a unique digital multiplier to carry out the multiplication and time averaging process. It is on this last principle that the design of the correlator in this thesis is based, and the proposal may in fact be considered a modification of the Kitai and Masuko instrument to make use of the concept of parallel processing.

The philosophy and design basis for the arithmetic memory cell are presented in Chapter 2. A matrix of these cells is organized in Chapter 3 to form a computing memory array and solution of a potential field problem is discussed as a possible application. Chapter 4 introduces the design theory for the digital correlator and the pertinent algorithms are developed for the digital realization of correlation

functions. These algorithms are used in Chapter 5 as the design criterion for the hardware implementation of an instrument that measures mean values and the autocorrelation or cross correlation function of fluctuating waveforms. Chapter 6 summarizes the contents and offers areas for future improvement and development of the concepts laid down in this thesis.

CHAPTER 2

THEORY AND DESIGN OF ARITHMETIC MEMORY CELLS

2.1 Introduction

In most modern computers, the arithmetic operations of addition, subtraction, multiplication, and division are carried out in a device known as a central processor. As the name implies, this processor is central to the entire computer and is shared by all the various peripheral devices. For instance, if two words located in the memory core of the computer are to be added, then they must both first be loaded into the processor. The adding operation is then carried out and the result is returned to some memory location. This operation occupies the entire capability of the processor and while it is being carried out, no other arithmetic operations may commence. In the case where several groups of numbers are to be added and several sums obtained, the processor must be time shared so that the sums are obtained one at a time. The same limitation carries over into the remaining operations of subtraction, multiplication, and division.

The central processors of modern computers are extremely fast devices and in most cases the limitation of

single step operation may seem insignificant. In some circumstances, however, notably correlation, fourier analysis, and relaxation, a very large number of identical operations are to be performed in parallel and the elapsed time for computation can become rather extensive. Consider, for example, solution of Laplacian field problems by the method of relaxation.[11] Figure 2:1 shows a potential field divided into 36 matrix points identified by their relative positions on the x and y axis. In this method, the potential at a given point, say P(x,y), within the boundaries of the field, is a function of the potentials at the four corners of an imaginary square surrounding that point. This relationship may be expressed as

$$P(x,y) = \frac{1}{4}\,[P(x-1,y) + P(x+1,y) + P(x,y+1) + P(x,y-1)] \qquad .....(2.1)$$

Equation 2:1 is applied to all the interior points to complete one relaxation. The process is repeated again and again until the solution converges or relaxes to the desired accuracy. If the boundary conditions of the 36 point field of Figure 2:1 are taken to be x = 0 and x = 5, then there are 24 interior points to be analyzed. Each point involves four additions and one division for each relaxation. There are then 120 arithmetic operations per relaxation. In this method of solution the number of relaxations performed is typically 50. For the small 36 point field then there

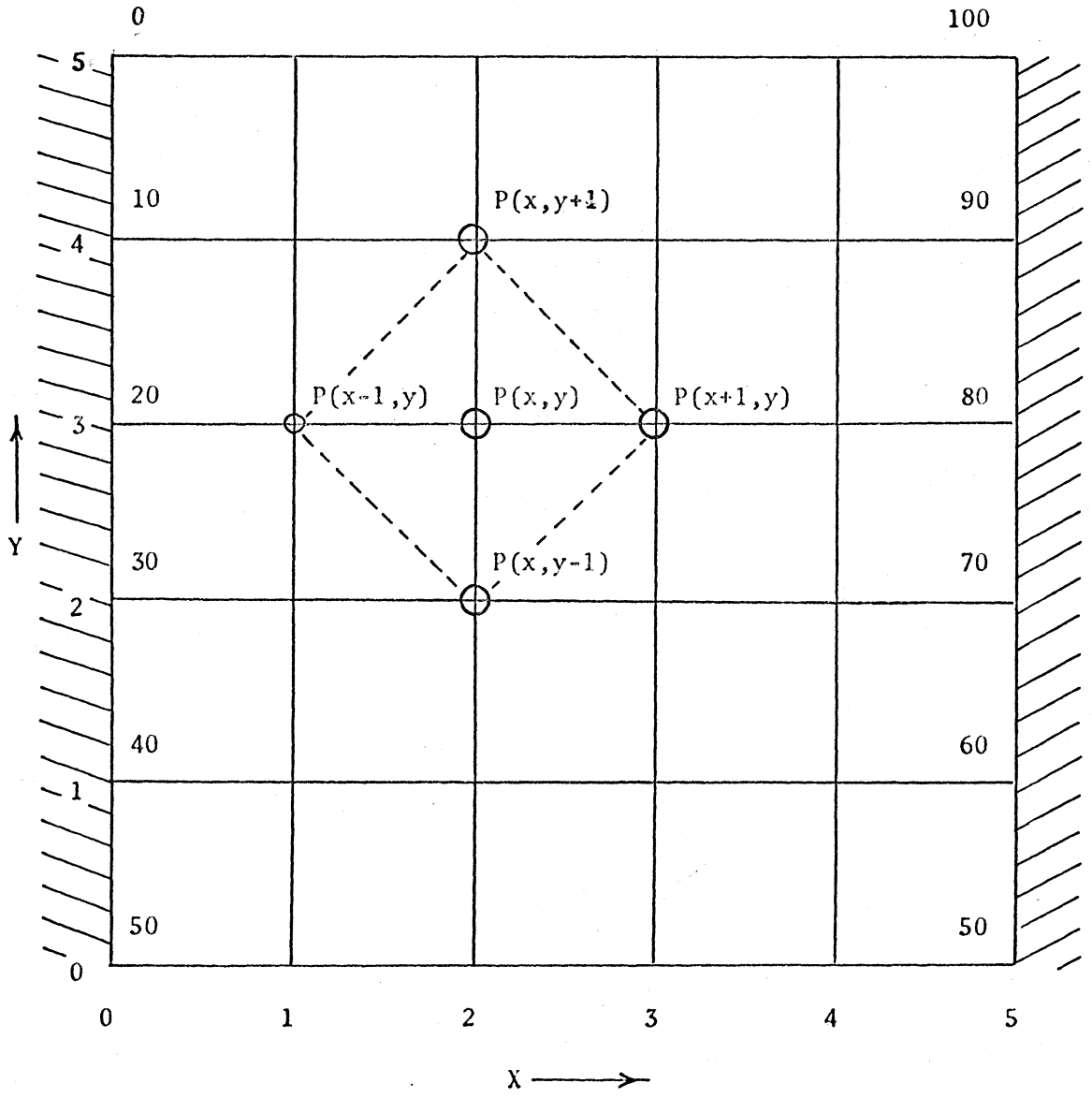


FIGURE 2.1 POTENTIAL FIELD MATRIX

will be a total of 6000 arithmetic operations. For solution by contemporary computers, therefore, the central processor will be called upon to carry out 6000 steps in addition to the loading of words from memory and the returning of words to memory. In practise a 36 point field is of little interest and for a reasonable degree of accuracy fields in the neighbourhood of 1,000 points are used. One can clearly see how a central processor can be used inefficiently.

In order to ease the burden on the central processor, it would be preferable to create an active computer memory. That is, instead of merely passively storing information, an active memory would have the added capability of computing, for example sums, differences, products, quotients, and do logic. Each memory location would have the arithmetic facility built in. In this way two or more words stored in memory can be added together directly without the necessity of unloading and reloading. Since each location is its own processor, there is no limit to the number of operations that can be performed simultaneously. In the example of solution by relaxation one complete relaxation will consist of five steps, four additions and one division. The time saving over conventional methods can be enormous.

This chapter suggests a special memory array capable of the mass logic described above. It is built with present day integrated circuits and is readily adaptable to large

scale integration. Several applications with examples are given and possible extensions are suggested.

2.2 The Arithmetic Memory Cell

The basic cell consists of a 16 bit memory register capable of serial addition. That is, any new word can be added to the word stored in a particular cell location in a single step without first removing the word from memory. This is accomplished by including a full adder with each memory location. Data in a memory location is addressed one bit at a time and is available serially, as in a shift register. Addition is therefore performed in a serial manner starting with the least significant bit, l.s.b., and progressing toward the most significant bit, m.s.b. The carry between bits is held in a delay as the adder is shifted. Figure 2:2 shows the simplified block diagram of the process with the three basic units, the memory, the adder, and the carry delay. Any number of memory cells can be addressed simultaneously and thus there is no limit to the number of additions that can be carried on in parallel. Serial addition is inherently slower than parallel addition but there must necessarily be some trade-off between speed and complexity of the single cell. In a large problem where a very great number of identical additions are to be performed, there is a point where the many slower serial additions all performed at once become faster in total

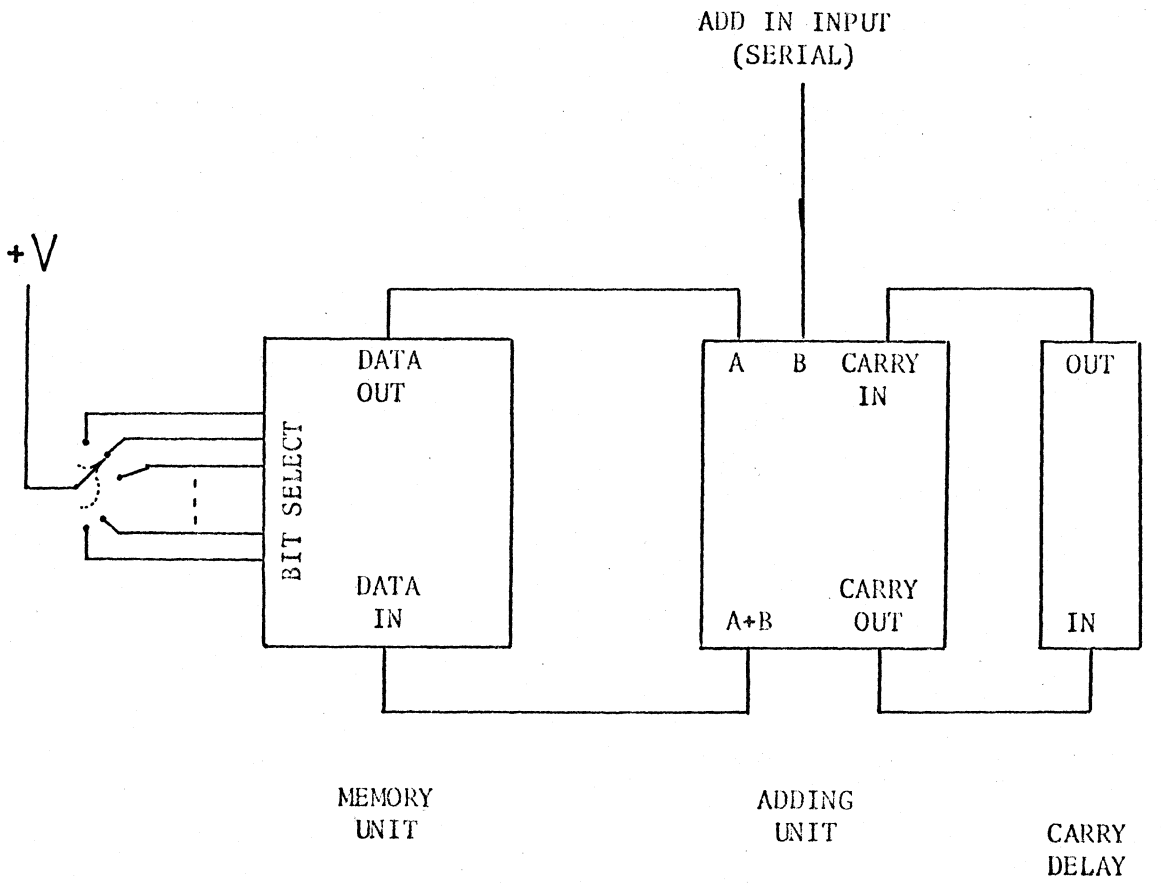


FIGURE 2.2 BLOCK DIAGRAM OF ARITHMETIC MEMORY CELL

elapsed time than the same additions executed in parallel but sequentially. Assume that a single addition takes T seconds regardless of the word length. Executed serially, an n bit addition will take nT seconds to complete. If parallel addition is employed, then the same n bit addition will produce a sum in T seconds. For a single addition, the parallel technique is faster than the serial by a factor equal to the bit length of the word. Assume now that m unique pairs of words are to be added each having n bits. If m serial adders are used then the total time required for the addition remains at nT. If a single parallel adder is utilized, however, then the total time for the m additions increases to mT. Clearly, then, when the number of similar additions to be performed becomes greater than the number of bits in the words, a system of numerous serial adders is, in fact, faster than a single high speed parallel adder.

The question naturally arises at this point as to why one should not use a system of many parallel adders to gain even greater speed. The answer, of course, lies in the complexity of the system. In a serial adder only a single bit adder is used for n bits of a single word. In the parallel case, an adder must be assigned to each of the n bits. The parallel adder must be n times larger than when serial addition is used. The interconnection between cells must also be n times more complex since all bits must be

compared simultaneously instead of one bit at a time. Rapid advances in large scale integration technology, however, may in the near future make parallel addition more practical. In the system proposed here, hardware availability limits consideration to serial addition.

Consider again Figure 2:2. For a serial adder, a memory is required having a system for addressing a number of flip flops in such a manner that single data in and data out lines can be used to selectively write information into a unique location and also read the information held in the location. Texas Instruments Model SN7480 integrated circuit contains 16 flip flops arranged in a matrix form so that only a single matrix position is activated at a time. Appendix A contains specifications and circuit diagrams for all integrated circuits utilized in this thesis. The bit positions of the memory word are addressed in a cyclic manner from the least significant bit to the most significant bit. In a typical addition the l.s.b. is addressed first. The data stored in this position is added at the A input of the adder to information entering the B input. The sum of A & B is then directed to the data-in input of the memory and this sum is written into the position previously occupied by word A. If a carry is generated as a result of the A & B addition this information is held in the carry delay. When the writing operation has been completed the cyclic bit selector selects the next m.s.b. and the corresponding bit position

of the new word to be added is applied to the B input of the adder. The adder then compares the information stored in the memory, the new information to be added and the stored carry from the lesser significant bit and generates a new sum and carry. The new sum is written into the memory and the carry is again delayed for the next m.s.b. The process is repeated until all 16 bits have been added. The cyclic bit selector then resets itself to the l.s.b. to be prepared for the next addition.

To read out the information stored in the memory, the process is repeated with the exception that no new information is directed into the B input of the adder. The sum is then simply the information appearing at the A input and the word stored in memory is unaltered. Readout is thus non-destructive. New data may be entered directly into memory through the adder's B input when the data out line from the memory to the adder is disabled. In this case the sum is merely the data entered at B. If no new data is applied to B and the data out line from the memory is disabled then all bit positions in the memory are cleared.

Let the memory register be represented by A and the register containing the word to be added by B. Then the basic arithmetic memory cell is capable of 4 different operations. These may be termed "set" operations,

1. set A = 0
2. set A = A
3. set A = B
4. set A = A+B.

Operation 1 is a clear, 2 a readout of stored data, 3 a loading of data and 4 an addition of two numbers or more correctly an accumulated total of numbers. The basic cell then is no more than an accumulator and has no more capability than addition. It will be shown later that through proper sequencing, the operations of subtraction multiplication and division can be reduced to simple additions, well within the ability of the basic cell.

2.3 Circuit Description of the Basic Cell

Figure 2:3 shows the schematic of the basic adding memory cell. Texas Instruments Series SN74 integrated circuits are used throughout and are referred to by the 4 digit package number 74XX. Circuit diagrams for these packages are listed in Appendix A. Symbology for all gates follows mil 806B specifications and is listed in Appendix A.

The 7481 memory array employs a direct internal connection between the write inputs and the sense outputs. Data already stored in a memory location cannot therefore be used to write new information into the location without a suitable delay. This delay is realized in a D-type flip

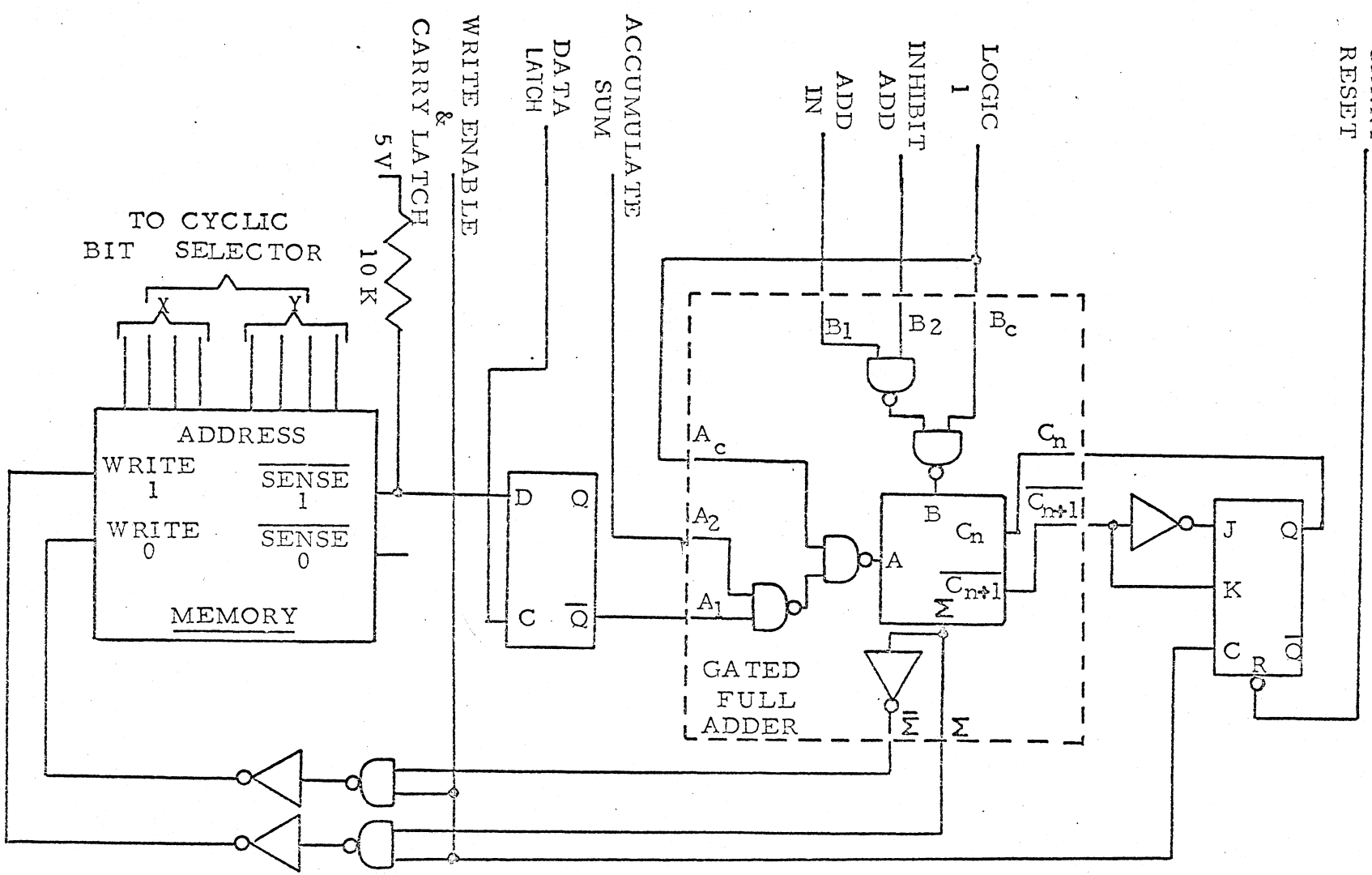


FIGURE 2.3 SCHEMATIC OF BASIC CELL

flop clocked with a pulse termed DATA LATCH. The sense lines of the memory, in practice, yield the complement of the data stored in the bit position and inversion of this data is accomplished by using the $\overline{Q}$ output of the flip flop. The final stage of the sense amplifiers feature open collector transistors and a 10 kilohm resistor is used, tied to +5 volts to ensure the required logic levels to drive the flip flop.

The full adder employed here, type 7480, incorporates a range of gating inputs rendering it very useful for this application. A simple adder utilizes A, B, and CARRY-IN inputs and provides A + B and CARRY-OUT outputs. These functions are the central core of the gated full adder of Figure 2:3. In the 7480, 3 gated inputs are provided to each of the A and B inputs and in addition the complement of the sum is available. This extra logic is shown within the dashed section of Figure 2:3. If the A_c input is held at logic 1, then the A_2 input may be used to disable the A_1 input. Only when A_2 is held at logic 1 will data at A_1 be passed through to the adding core. The same holds for the B inputs. When the $\overline{Q}$ output of the flip flop is fed to the A_1 input and the B_1 input is used to direct new data to the basic cell, then the A_2 and B_2 inputs function as accumulate sum and inhibit add respectively.

It is convenient to use both the sum and complement-

of-sum outputs of the adder. New information, either a 1 or a 0, is written into a memory bit location by momentarily raising the corresponding write input to the logic 1 level. In order to maintain control over this write function, 2 input NAND gates and inverters are used to link the sum outputs of the adder to the write inputs of the memory. One input of each NAND gate is then taken to a common WRITE ENABLE line. Only when a logic 1 is applied to this line will the sum be written into the memory.

The carry delay is realized in the basic cell by a J-K flip flop. A D-type flip flop is unsuitable in this application since the state of the CARRY OUT output of the adder is determined in part by the data at the CARRY IN input. As long as the clock input of a D-type flip flop is held high, whatever information appearing at the D input will be passed through to the Q input. Typical propagation delay from the rising edge of a clock pulse to the $\overline{Q}$ output is in the neighbourhood of 7 ns. Delay from C_n to $\overline{C}_{n+1}$ through the adder is typically 8 ns. Set-up time at the D input is approximately 7 ns. The total delay in the loop is thus about 22 ns. This, however, is a typical figure and in practice, could be considerably less. The maximum clock pulse width must therefore be of the order of 15 ns, for correct functioning of the system. Pulses of such a small duration are difficult to achieve and in addition it is

desirable that the basic cell operate independently of clock pulse duration. Unlike the D-type flip flop, the J-K triggers on the trailing edge of the clock pulse. That is, whatever information appears at the J and K inputs when the clock input is in the logical 1 state, will be transferred to the Q output when the clock is returned to the 0 state. Since the CARRY OUT is complemented in the adder, it is inverted at the input to the flip-flop. The clock input, termed CARRY LATCH, is joined to the WRITE ENABLE line so that at the same time as the new sum is written into the memory, the carry to the next most significant bit is stored. The clear input of the carry latch, called CARRY RESET, is provided to prevent the carry from the m.s.b. being returned to the l.s.b. As long as the l.s.b. is being addressed, there must be no CARRY IN.

Once the new sum has been written into a bit position, addition is carried on to the next most significant bit by incrementing the bit selector by unity. In its simplest form, the bit selector is simply a binary counter with appropriate decoding to drive the X and Y address lines. Incrementing of the counter may occur at the same time as WRITE ENABLE and CARRY LATCH. There are two steps to each bit addition:

1. DATA LATCH, and
2. WRITE ENABLE, CARRY LATCH, and RE-ADDRESS.

This can be accomplished with a single pulse generator by gating the pulses alternately to one line or the other. After the m.s.b. has been modified the bit selector returns to the l.s.b. and ceases to count until another add command is received. The control circuitry can be made quite simply and has the advantage that any number of cells can be driven from a single control. Since each cell is an independent adder and storage element in itself there is no need to time-share the control unit and there is no limit to the number of additions that can be carried out simultaneously. Figure 2:4 shows the control circuit required to drive the basic cell. The cyclic bit selector is simply a 4 bit binary counter, type 7493, with the A and B outputs used to drive one decoder for the X address lines of the memory and the C and D outputs taken to a second decoder for the Y address lines. These decoders are BCD to decimal decoders, type 7442, used as 2 line to 4 line decoders by taking the C and D inputs of each to ground potential. The decoders feature complementary outputs, and descrete component address drivers may be designed to simultaneously drive as many memories as desired. As a matter of convenience the counter code for the least significant bit is taken as a high state at all 4 counter outputs. The type of code and starting point are completely arbitrary so long as they are consistent. All high is chosen for the l.s.b. since this code is easily detected using a 4 input NAND gate. Previous to the start

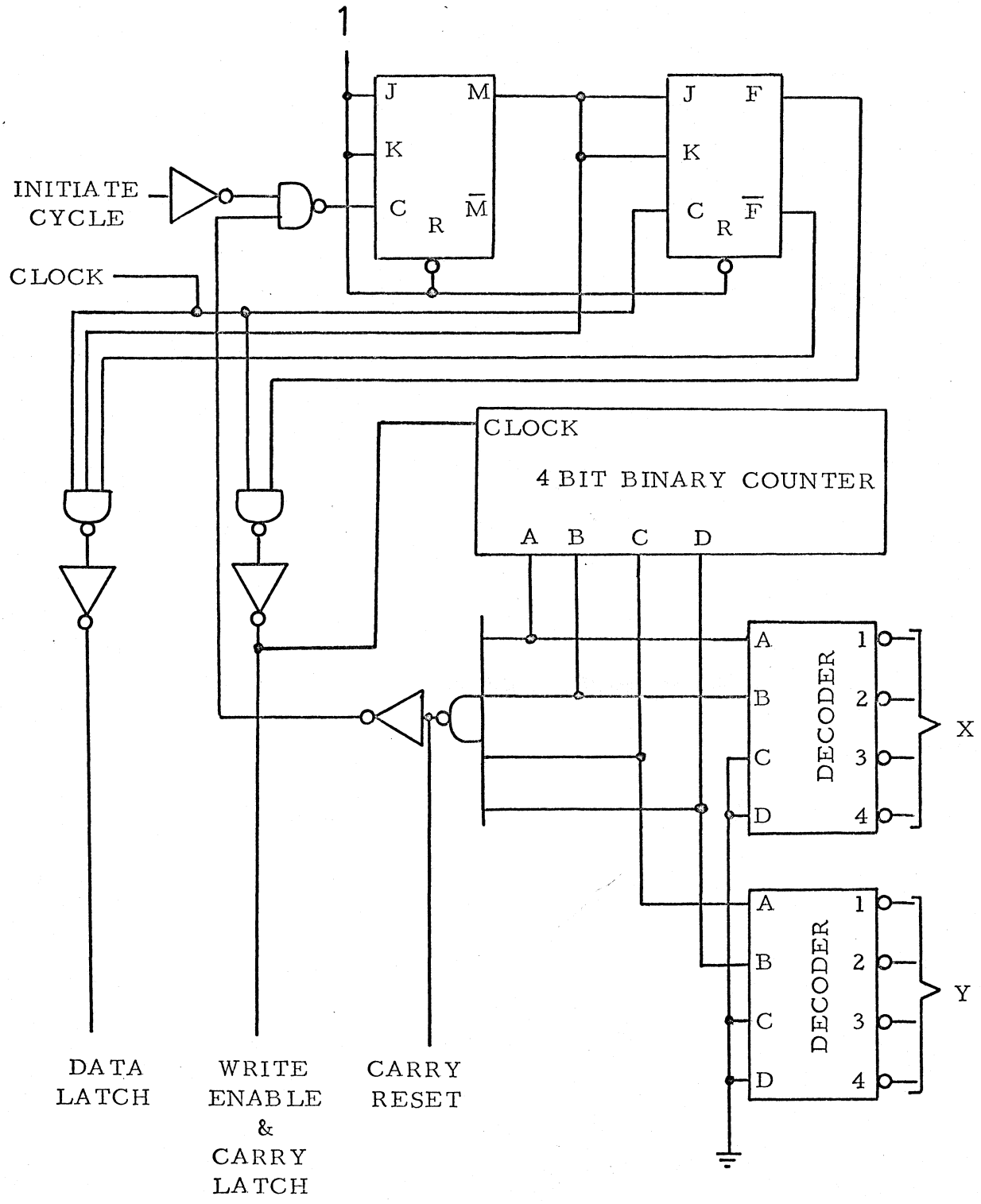


FIGURE 2.4 CONTROL CIRCUIT FOR BASIC CELL

of an adding cycle, control flip flops M and F are both in the 0 state. The continuous train of clock pulses have no effect on flip flop F since both J and K inputs are low. When the INITIATE CYCLE input to the control is raised to a logic 1 the clock input to flip flop M is raised from 0 to 1. As the INITIATE CYCLE input is raised to 0 the trailing edge at the clock of M triggers the flip flop, and since both J and K inputs are tied permanently to logic 1 the flip flop complements. Raising M to a logic 1 in turn applies a high to both J and K inputs of F. Flip flop F will then complement for every succeeding clock pulse. The first clock pulse received following the raising of M to logic 1 is directed through a 3 input NAND gate and inverter to the DATA LATCH inputs of the various memory cells. The second is carried through a 2 input NAND gate and inverter to the WRITE ENABLE and CARRY LATCH inputs. At the same time, the counter is incremented by unity to address the second least significant bit. The third pulse is again directed to the DATA LATCH. This is repeated for 32 clock pulses, 2 pulses for each bit position, until the counter recycles to the l.s.b. When the counter was first incremented the output of the 4 input NAND gate was raised to a logic 1 causing the clock input to the M flip flop to be raised from 0 to 1. As the counter returns to the l.s.b. the output of the 4 input NAND gate goes low and the trailing edge applied to the clock of M causes the flip flop to complement again

and all further clock pulses are blocked. The adding cycle is now complete and the control is reset and ready for the next adding cycle. The output of the 4 input NAND gate is also used to reset the carry latch so that overflow from the m.s.b. cannot be carried back to the l.s.b.

Now that the principle and operation of the arithmetic memory cell has been established, it remains to show how these basic cells can be incorporated into a large memory array. Chapter 4 describes such a computing memory and suggests how such a device may be interfaced with existing machines.

CHAPTER 3

THE COMPUTING MEMORY

3.1 Introduction

The basic arithmetic memory cell described in Chapter 2 is the fundamental block of the computing memory. If many cells are arranged in some matrix array addressable individually by matrix location, then the memory resembles conventional passive memories but has an additional capability. Any new number can be added to a number already stored in memory directly without first removing the word from its location and returning the sum after the addition has been completed. This is an interesting feature but of questionable advantage when consideration is made of the extra gating required to make every location an accumulator. If, however, additional interconnection between matrix locations is provided and provision is made for direct addition between locations, then the capabilities of the computing memory become enormous. Such a computing memory is described in this chapter. The memory is small, able to store only 64 15-bit words, but the concept is easily extendable and is particularly adaptable to large scale integration. Only 8

words of the memory have been built and tested at the time of writing, but the design will incorporate the remaining 56 with no alteration.

In the memory described here, the 64 words are subdivided into 32 PAIRS, labeled EVEN and ODD, each being an independent basic arithmetic memory cell. All arithmetic operations are performed within the PAIRS and the result is stored in either the EVEN or the ODD depending on the command. For example, a command ADD EVEN TO ODD results in the number stored in the EVEN cell being added to that stored in the ODD cell and the sum stored in the ODD location. It must be understood that there are 32 EVEN and 32 ODD locations and thus a command ADD EVEN TO ODD will effect 32 independent additions. That is, all EVENS will be added to all ODDS and all sums will be stored in the ODDS. A similar command exists for ADD ODD TO EVEN. In this case the same numbers are added but the sum is stored in the EVEN location. An additional important built-in feature of both EVEN and ODD cells is a RIGHT SHIFT and LEFT SHIFT. These commands effect a shift of the word stored in a cell location by a single bit position to the right or left respectively. Since the words are stored in a binary code, such a shift is effectively division and multiplication by 2 respectively. By proper sequencing of the adding and shifting operations, each PAIR becomes an independent multiplier. Provision for forming the two's complement is included in all cells and thus subtraction can also be

realized as simply as addition.

It is noteworthy that when an addition between EVENS and ODDS is performed, one of the words added is lost in order to store the sum in the location previously occupied by that word. In some cases this is undesirable and to prevent unwanted operations an inhibit add facility is built in. This feature is in the form of a bit position in the stored word. At the commencement of any operation this bit position is first surveyed. If a 1 is stored in the position then the operation proceeds. If, however, a 0 appears in the inhibit bit position then the word remains unchanged. Again, all cells are independent and one operation inhibited for one PAIR has no effect on the remaining PAIRS. For each individual PAIR the decision whether to add resides in the inhibit bit of that pair.

The capabilities of a PAIR can be listed as SET commands. That is SET EVEN = EVEN + ODD means add the number stored in the EVEN location to the number stored in the ODD location and place the sum in the EVEN location. Only those operations in which the result is stored in the EVEN location are listed here. It must be understood that exactly the same capabilities are permitted where the result is stored in the ODD location.

1. EVEN = 0
2. EVEN = A
3. EVEN = EVEN + A
4. EVEN = - EVEN
5. EVEN = ODD
6. EVEN = EVEN + ODD
7. EVEN = EVEN
8. EVEN = EVEN x N
9. EVEN = EVEN/N
10. EVEN = ODD x A
11. EVEN = EVEN x ODD

A is a number stored in the control unit, and N is an integer such that $N = 2^x$, x = 1, 2, 3,... . These operations are described in detail later in this chapter. It will suffice here only to point out that these capabilities exist for each and every PAIR and that the particular operation is simultaneously carried out in all PAIRS. For example, operation number 10 implies that all 32 numbers stored in ODD locations are simultaneously multiplied by a common number A and all 32 products are stored in the corresponding EVEN locations. 32 operations are performed at once.

In order that arithmetic operations between locations outside the PAIR be permitted, a system of shifting of words from location to location is provided. Figure 3:1 shows the 32 PAIR memory organized into a rectangular array of 4 by

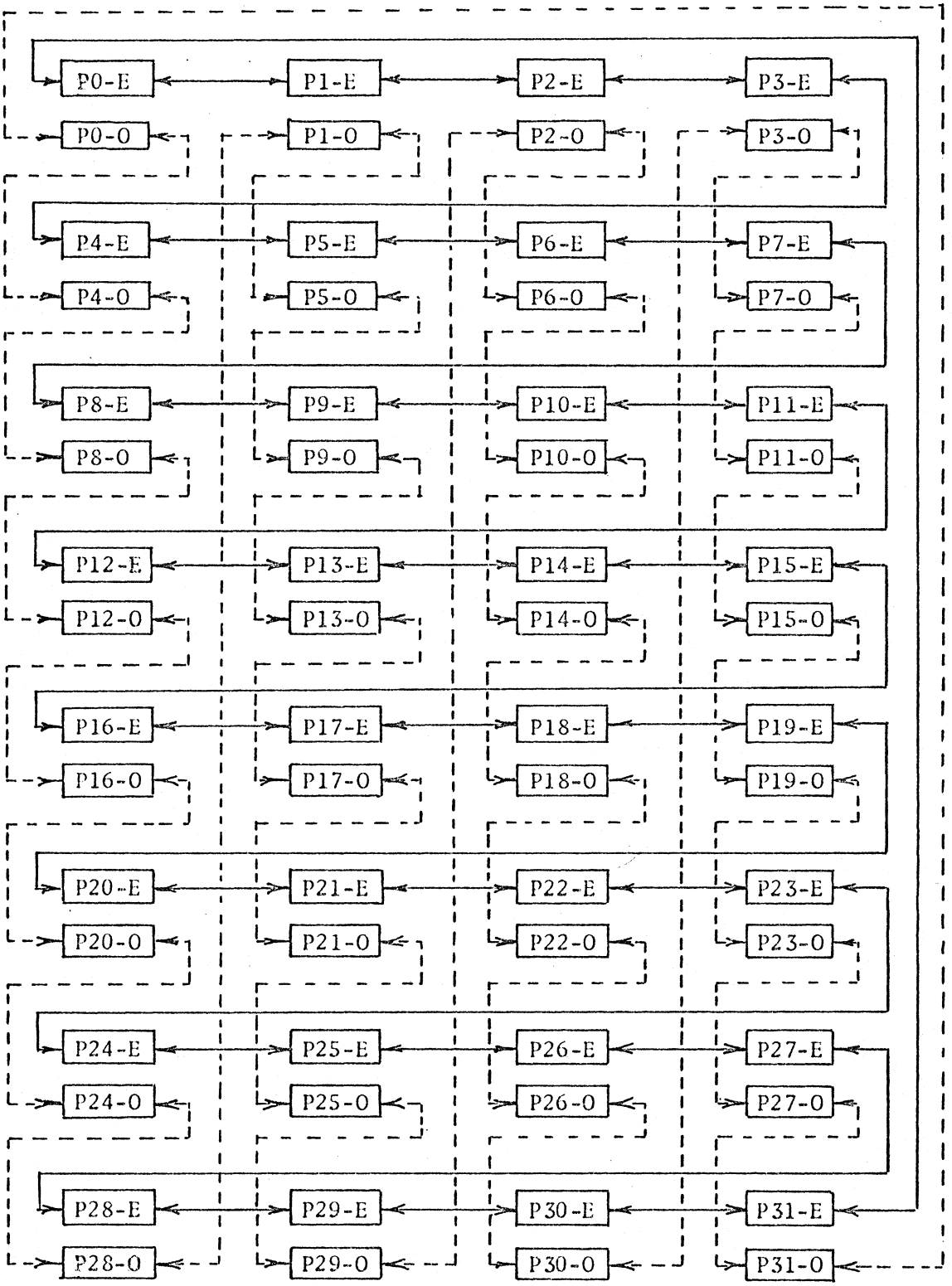


FIGURE 3.1 INTERNAL DATA SHIFTING PATTERN

8 PAIR locations, each location with an EVEN and ODD cell. Each PAIR location possesses the capabilities previously listed. In addition, all EVEN or all ODD words can be shifted in two directions. If the array of Figure 3:1 is considered to lie in the X-Y plane then the EVENS may be said to be shifted in the plus X or minus X directions. Likewise the ODDS can be shifted in the plus or minus Y direction. To prevent loss of information at the end of a row or column, an end around link is included so that the word shifted beyond the last position of a row or column is carried over to the first position of the next row or column. The solid lines of Figure 3:1 designate the shifting pattern for EVEN cells while that for ODD cells is shown dashed. Position in the array is described by the letter P followed by the PAIR number. Each position is suffixed by the letter E or O to indicate EVEN or ODD cells respectively.

Once the general configuration of the computing memory is understood it remains to describe the circuit blocks in detail and explain how the 11 operations previously listed are to be realized. In addition, the following sections present the control circuit required to drive and manipulate the memory. The final section of this chapter includes a discussion of the design and suggestions for improvement.

3.2 Circuit Description

All 64 arithmetic memory cells of the computing memory are identical. It is only the system of cell interconnection that differentiates EVEN and ODD cells of a particular PAIR. Hence, all PAIRS are also similar and it is only their position in the array that makes them unique. Figure 3:2 illustrates a typical cell. In order that this may be regarded as a general cell, the interconnection lines are shown unterminated. They are identified by a code which describes their function. Once the code is understood, the joining of interconnection lines between corresponding cells will become obvious.

All codes consist of 4 uppercase characters. The first is an identifier which classifies the line into one of 3 subgroups. These subgroups are:

WXXX pulsed input command

ZXXX instruction step

TXXX data transfer

A complete listing of the code is recorded in Table 3.1. It can be noted from Table 3.1 that some codes are common to both EVEN and ODD cells while others are peculiar to each. To avoid confusion the typical cell of Figure 3:2 will be considered an EVEN cell for the purpose of coding. This choice is completely arbitrary and the cell might just as easily be made ODD by replacing the codes by those

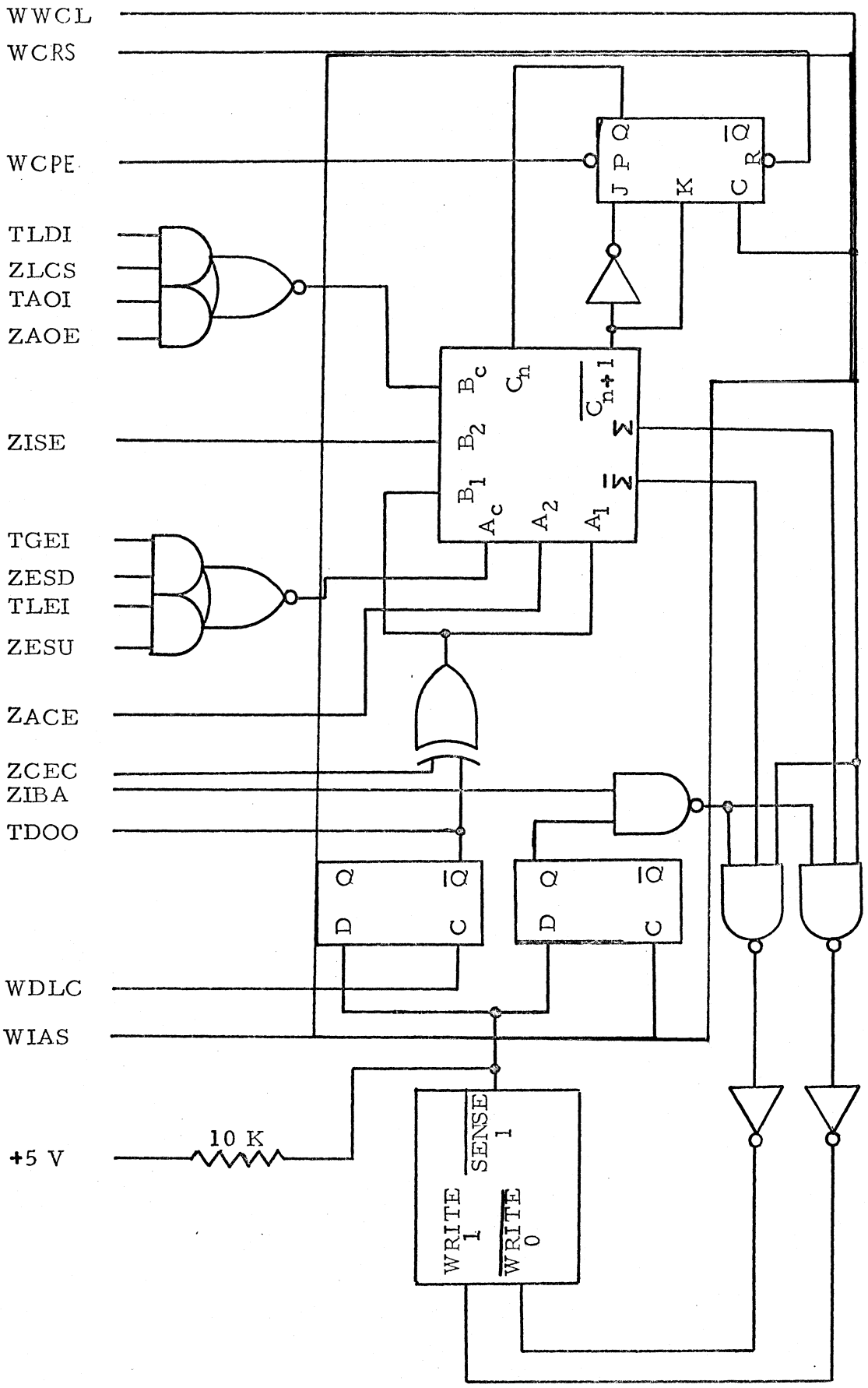


FIGURE 3.2 TYPICAL ARITHMETIC MEMORY CELL

WWCL		WRITE ENABLE AND CARRY LATCH
WDLC		DATA LATCH
WIAS		INHIBIT ADD BIT SET
WCRS		CARRY RESET
WCPE	(WCPO)	CARRY IN PRESET EVEN (ODD)
ZCEC	(ZCOC)	COMPLEMENT EVEN (ODD) DATA
ZIBA		INHIBIT ADD BIT ENABLE
ZACE	(ZACO)	ACCUMULATE EVEN (ODD) DATA
ZESU	(ZOSU)	EVEN (ODD) SHIFT UP
ZESD	(ZOSD)	EVEN (ODD) SHIFT DOWN
ZAOE	(ZAEO)	ADD ODD TO EVEN (EVEN TO ODD)
ZLCS		LOAD DATA
ZISE	(ZISO)	SHIFT EVEN (ODD) WORD 1 BIT
TDOO		DATA OUT
TLEI	(TLOI)	LESSOR EVEN (ODD) IN
TGEI	(TGOI)	GREATER EVEN (ODD) IN
TAOI	(TAEI)	ADD ODD (EVEN) IN
TLDI		LOAD DATA CELL SELECT

TABLE 3.1 CELL INTERCONNECTION IDENTIFICATION CODE

corresponding to ODD cells. For example ZAOE for the EVEN cell would be replaced by ZAEO for the ODD cell.

The circuit is essentially the same as that for the basic arithmetic cell of Figure 2:3 with additional gating provided to expand its capabilities. Data appearing in complemented form at the $\overline{\text{SENSE 1}}$ output of the memory is first latched by the upper D-type flip-flop. The lower latch is used to detect and decode the inhibit add bit if this mode of operation is chosen. A 1 contained in the inhibit add bit position will allow addition to be completed. If, however, a 0 is detected in this position then it is required that a flag be set which prevents any change of the word in the memory location. This feature is accomplished through a system of latching and gating. The inhibit add option is first activated by applying a logical 1 to the inhibit add bit enable, ZIBA. The inhibit add bit must then be the first bit selected by the cyclic bit selector. This means simply that the inhibit add bit is the starting point of the selector. At the same time as this data is latched in the upper D-type flip flop it is also stored in the lower flip flop. Simultaneous pulsing of both WDLC and WIAS will accomplish this end. It should be noted here that while WDLC is pulsed for every bit position, WIAS is pulsed for only the inhibit bit. If a 1 appears in the inhibit add bit then a 0 will be taken to the Q output of the lower latch due to inversion at the memory output. This

0 is further inverted at the 2 input NAND gate and then applied to one input each of the two 3 input NAND gates. Upon receipt of a write enable pulse at WWCL the 3 input NAND gates will transmit the information appearing at the sum outputs of the adder to the write inputs of the memory. If, however, a 0 is located in the inhibit add bit position, this low level will be carried through a similar path to deactivate the 3 input NAND gates. When these gates are deactivated, no new word can be written into the memory location. If the inhibit add option is not to be used then the ZIBA line is maintained in the 0 state and the output of the 2 input NAND gate remains high regardless of the state of the inhibit bit latch. The 3 input NAND gates remain activated and writing is permitted. It should be noted, that although the inhibit add bit will in most cases be the first bit selected, this is not invariable. It will be shown later in this chapter that there is some advantage in making this bit position flexible. In some operations the bit position will change many times in the course of a single operation.

The $\overline{Q}$ output of the data latch is taken directly to an exclusive or gate. The inputs to this gate are the instruction step ZCEC and the data transfer line TD00. The output of the exclusive or gate will then be the function

$$F = ZCEC \oplus TDOO$$

$$= \overline{ZCEC} \cdot TDOO + ZCEC \cdot \overline{TDOO}$$

If the ZCEC command line is held in the logical 0 state then the data from the latch will be passed through the exclusive or gate unchanged. If, however, the ZCEC line is taken to the logical 1 state then the data appearing at the output of the gate will be the complement of that from the D-type latch. When ZCEC is held high through an entire cycle of the bit selector, all bit positions of the word in memory will be complemented. Thus the one's complement of a number stored in memory can be formed. This feature is particularly useful in the processing of negative numbers. Negative numbers are most conveniently handled in two's complement form. Subtraction can then be performed as simply as addition. The two's complement is just the one's complement plus unity. Once the one's complement is formed by the exclusive or gate, it remains only to add unity to this number by inserting a carry in to the least significant bit. This is readily accomplished using a presetable J-K flip flop for the carry latch. When EVEN or ODD numbers stored in memory are to be negated, the associated complement command lines ZCEC or ZCOC respectively are raised to logical 1 and the presetable J-K flip flops are set to 1 by the same pulse used to latch the data in the l.s.b. position. In this manner the number is negated in a single cycle of the bit selector.

By using two and-or-invert gates in conjunction with the built in gating to the full adder, three distinct sources are provided to both the A and B input of the adder. Refer to Figure 2:3 or Appendix A for the meaning of the subscripted A and B inputs. The AND functions permit passage of data when the appropriate instruction line is held at a logical 1 voltage. Thus for each input source there is one Z coded instruction line and one T coded data transfer line. The three A inputs provide for either accumulation, up shifting or down shifting. These may be expressed in functional form

$A = (TDOO \oplus ZCEC)\cdot ZACE + TLEI\cdot ZESU + TGEI\cdot ZESD$

Similarly the three B inputs allow for left and right shifting of the stored word, addition of a second word, or data loading.

$B = (TDOO \oplus ZCEC)\cdot ZISE + TAOI\cdot ZAOE + TLDI\cdot ZLCS$

Since each input is an OR function only one source for each input may be selected at a time. It is the responsibility of the central control unit to select the proper combination of sources for the operation desired.

The sum output of the adder is based on the A and B inputs as expressed above and on the carry in. A J-K flip flop is used for the carry delay as was explained for Figure 2:3. The addition of the presentable feature has already been examined. Again it is required of the control unit to properly sequence the clear and preset pulses as

necessary. The design and operation of this control unit is discussed in a later section. It will suffice here to investigate the memory array itself and the steps involved in carrying out each operation.

The ten operations listed earlier in this chapter will now be examined individually. It must first be understood that a step consists of a single cycle of the bit selector during which appropriate instruction step lines must be activated. At the end of the cycle, other instruction step lines must be raised to a logical 1 for the next step. The nature of the cyclic bit selector insures that all steps require an equal amount of time. This time is predictable and is based on the propagation delay of data around the processing loop. Some operations, notably those involving multiplication, however, require many steps for completion. In other words, while all steps require the same amount of time, this is not the case with all operations. This is of no real disadvantage since a given operation will always involve an integral number of steps and the total time for completion is again predictable. The operations can be discussed from the point of view of both realization and application.

3.3 Circuit Operations

The circuit operations are listed again for convenience

1. EVEN = 0
2. EVEN = A
3. EVEN = EVEN + A
4. EVEN = - EVEN
5. EVEN = ODD
6. EVEN = EVEN + ODD
7. EVEN = EVEN
8. EVEN = EVEN x N
9. EVEN = EVEN/N
10. EVEN = ODD x A
11. EVEN = EVEN x ODD

Again only the EVEN cells are considered here. Since EVEN and ODD cells are identical, separate consideration of ODD cells would be redundant.

3.3.1 EVEN = 0

This operation is effectively setting all the bit positions of all EVEN cells to 0. It is a clear memory operation that must precede any loading of new data. A single step operation, it consists of simply ensuring that nothing is applied to either A or B input of the adder and no carry is permitted. In this case the sum will always be zero and zeros will be written into all bit positions of the memory. Care must be taken that the inhibit add option does not prevent the writing of these zeros. This is easily accomplished by maintaining the inhibit add bit enable line

ZIBA in the low state. The clearing operation is then a single cycle of the bit selector with all Z instruction lines held in the logical 0 level.

3.3.2 EVEN = A

This is a load operation where the number represented by A is loaded serially into one of the EVEN cell locations. Words are loaded into a previously cleared memory cell through the B input of the appropriate adder when the accompanying A input is held in the low state. As in the clear operation, the inhibit add bit enable line ZIBA must be maintained low to prevent this option interfering with the write operation. For a word to be loaded the control unit selects the desired cell by raising the particular load data cell select line, TLDI, to the logical 1 state and entering data through the load data line, ZLCS. This latter line is common to all memory locations both EVEN and ODD, but because of the AND gate at the B input only one cell is loaded at a time. One limitation of the computing memory should be pointed out here. Since all EVEN or ODD locations are cleared simultaneously the word in a particular cell can not be changed without losing the contents of all other similar locations. This limitation may be alleviated by incorporating additional gating in the accumulate data lines, ZACE and ZACO. Like the clear operation, the load operation is one of the fundamental single step operations.

3.3.3 EVEN = EVEN + A

This operation is a simple extension of the load data operation. A single step operation, it involves addition of a new number to a number already stored in a memory location. As in the load operation, the new number is serially added in and can be added to only one cell at a time. The number is entered by the same path used for EVEN = A, but it is also required that the number already stored in the cell be re-entered at the A input to the adder. Raising the accumulate data line, ZACE, to the logical 1 level will ensure re-entry of the EVEN word. Use may be made here of the inhibit add option by simply maintaining ZIBA to the high level, otherwise it is held low.

3.3.4 EVEN = - EVEN

This operation is the number negation described earlier. When negative numbers are expressed in two's complement form, subtraction becomes a simple matter of addition. As pointed out previously, the two's complement of a number is found by first complementing all bit positions to form the one's complement and then adding unity. The most significant bit position becomes a sign bit: a 0 represents a positive number and 1 a negative number. The number of bits available for storage of absolute numbers is reduced by one but the overall range of storage is kept constant by allowing equal positive

and negative values. In practise all EVEN or all ODD cells are negated simultaneously in a single step. The complement data line ZCEC is held high for a complete cycle and the carry delay flip flops are preset to 1 for the l.s.b. position by the same pulse that latches the data for this bit. All B inputs to the adder are disabled and the accumulate data line, ZACE, is held high. The inhibit add option is available here but a check must be made to prevent the inhibit add bit position from being complemented by the exclusive or gate. The control unit ensures this by blocking the write enable pulse for this bit position.

3.3.5 EVEN = ODD

In this single step operation, all words located in ODD cells are simultaneously transferred into the adjacent EVEN locations. Readout from the ODD locations is non-destructive as the ODD words are entered into the previously cleared EVEN cells through the B inputs to the adder. The instruction step line ZAOE, corresponding to add ODDS to EVENS, is held at a logical 1 voltage for a complete cycle and data is entered through the add ODD in transfer line, TAOI. This line is directly connected to the data out line, TDOO, of the adjacent ODD cell. The ODD word is then directly loaded into the EVEN location one bit at a time. No data enters through the A input to the adder and the inhibit add bit enable line must be kept low.

3.3.6 EVEN = EVEN + ODD

Known as add ODDS to EVENS, this single step operation simultaneously adds all the numbers in ODD cells to the numbers in adjacent EVEN cells and stores the result in the EVEN cells. ODD data enters the cell through the adder's B input as in the EVEN = ODD operation, but data already in the cell is applied to the A input by raising the accumulate data line, ZACE, to high. A sum and carry are generated as in normal addition and this information is used to update the EVEN cells. The inhibit add feature is available if desired and is activated in the usual manner. Since two's complement arithmetic is employed either or both numbers may be negative without interfering with the addition.

3.3.7 EVEN = EVEN

This operation may be thought to be trivial, but it will be seen in a later section that certain combinations of steps are permissible, and are more easily understood if this fundamental step is considered. The operation is essentially a reading of data and a rewriting of the data without change. All B inputs to the adder are disabled and the latched data is fed to the A input by maintaining a logical 1 on the accumulate data line. This step may be used for nondestructive readout of data and also as an insurance that EVEN data is not lost while operations are performed in ODD locations. Since stored data is not to be changed, the inhibit add

feature has no effect.

3.3.8 EVEN = EVEN x N

This is the first of the multiple step operations, the number of steps required depending on the value of N. If N is an integral number given by $N = 2^j$, $j = 0, 1, 2, \ldots$, then the number of steps required for the operation is j. Multiplication of a binary number by 2 is a simple shifting of all bits to the left by one bit position. Shifting by two bits corresponds to multiplication by 4, three bits to x 8 and so on. Each step effects a shift by one bit position, hence a multiplication by 2. In practise, such a multiplication is accomplished by adding the number to itself. Data from the data latch is simultaneously entered through both the A and B inputs to the adder. The sum output is then just twice the original number stored and this new number is written into the memory location. Both the accumulate data line ZACE and word shift line ZISE must be held high so that the data is channeled to both adder inputs. All EVEN words or all ODD words are shifted at once and the inhibit add option is available.

3.3.9 EVEN = EVEN/N

This is a second multiple step operation closely parallelling that of the previous section. Division of a binary number by $N = 2^j$, $j = 0, 1, 2, 3, \ldots$ is simply a

shifting of the number toward the l.s.b. by a number of bit positions corresponding to j. Such a shift can be easily accomplished in the computing memory if a simple technique is employed. Consider again the process of multiplication by 2. Since the data applied to the A and B inputs of the adder is always identical, regardless of whether the bit position contains a 0 or a 1 the sum will always be zero. If two 1's occur there will be a carry out; otherwise there will be no carry for two 0's. It is this carry delayed to the next bit position that determines if the sum output of the adder will be a 1 or a 0. The process is effectively then a delay of the data word by one bit position as the bit selector sweeps through its cycle. Consider now the effect of reversing the direction of the bit selector. Instead of initiating with the l.s.b. and progressing toward the m.s.b., the selector will now start with the m.s.b. and sweep towards the l.s.b. The state of each bit will again be stored in the carry delay, but in this case the stored data will be written into the next l.s.b. position. The end result will be a shift toward the l.s.b. and hence a division by two. As pointed out in an earlier section, the bit selector is nothing more than a binary counter with decoder drivers. By incorporating an up-down count feature the bit selector can be made reversible. The inhibit add operation is applicable in this operation and all EVENS or all ODDS are shifted at once. No adding operations are permitted when the selector is reversed since the

carry is toward the lessor significant bit.

3.3.10 EVEN = ODD x A

In this operation all ODD numbers are multiplied by a common multiplier, A, and the individual products are stored in the corresponding EVEN cells. Binary multiplication may be regarded as a process of shifting and adding, both readily accomplished using the ODD/EVEN cell pairs. The multiplicand is entered into the ODD location and is either added or ignored depending on the l.s.b. of the multiplier. The multiplier must thus be held in the control unit. If the l.s.b. is a 1 then an EVEN = EVEN + ODD command is given. If on the other hand the bit contains a 0 then a EVEN = EVEN blank command is called for. The ODD word is then shifted toward the m.s.b. by one bit position and the next greater significant bit of the multiplier is checked to determine whether a second addition is to be performed. The process of shifting and adding is repeated until the multiplication is complete. Figure 3:3(a) illustrates a general multiplication of two 4 bit numbers $x_3x_2x_1x_0$ and $y_3y_2y_1y_0$. The multiplicand denoted by the subscripted x's is rewritten 4 times each time displaced toward the m.s.b. by one bit position. At the right adjacent to the shifted multiplicand is written the corresponding bit of the multiplier that will determine if the shifted multiplicand is to be added or ignored. It is noted that due to the nature of the multiplication

(a)

$$
\begin{array}{rcccccc}
 & & X_3 & X_2 & X_1 & X_0 & \\
 & & Y_3 & Y_2 & Y_1 & Y_0 & \\
\hline
 & & X_3 & X_2 & X_1 & X_0 & -Y_0 \\
 & X_3 & X_2 & X_1 & X_0 & & -Y_1 \\
X_3 & X_2 & X_1 & X_0 & & & -Y_2 \\
X_3 & X_2 & X_1 & X_0 & & & -Y_3 \\
\hline
Z_6 & Z_5 & Z_4 & Z_3 & Z_2 & Z_1 & Z_0
\end{array}
$$

(b)

$$
\begin{array}{rcccccc}
 & & X_3 & X_2 & X_1 & X_0 & \\
 & & Y_3 & Y_2 & Y_1 & Y_0 & \\
\hline
X_3 & X_2 & X_1 & X_0 & & & -Y_3 \\
 & X_3 & X_2 & X_1 & X_0 & & -Y_2 \\
 & & X_3 & X_2 & X_1 & X_0 & -Y_1 \\
 & & & X_3 & X_2 & X_1 & X_0 \quad -Y_0 \\
\hline
Y'_6 & Y'_5 & Y'_4 & Y'_3 & Y'_2 & Y'_1 & Y'_0
\end{array}
$$

FIGURE 3.3 BINARY MULTIPLICATION OF TWO 4-BIT NUMBERS
(a) USING 3 REGISTERS
(b) USING 2 REGISTERS

operation the total number of bits in the numbers to be multiplied is restricted to the total length of storage capability of the memory. For a 16 bit memory storage, the multiplicand and multiplier may have a combined total of only 16 bits.

3.3.11 EVEN = EVEN x ODD

This is perhaps the most complex operation and exerts the greatest demands on the control unit. It involves the multiplication of all ODD numbers by the neighbouring EVEN numbers and the storage of all products in the EVEN locations. Most multiplication operations require 3 registers for the multiplicand, multiplier and product. By proper sequencing of steps it will be shown here that the operation can be performed using only two registers, one of which must serve the double purpose of storing both the multiplier and the product. Consider first Figure 3:3(b) showing the general multiplication of two 4-bit numbers. A simple rearrangement of Figure 3:3(a), this method examines the m.s.b. of the multiplier first and initially displaces the multiplicand by 3 bit positions to the left. After each addition step, the multiplicand is shifted toward the lessor significant bit and the next l.s.b. of the multiplier is examined. Consider now how this multiplication may be performed between an ODD and EVEN cell of the computing memory.

The multiplicand and multiplier are initially loaded

into the ODD and EVEN locations respectively. The ODD multiplicand is then shifted 3 bit positions toward the m.s.b. Whether or not this shifted word will contribute to the accumulated sum depends on the data in the m.s.b. of the multiplier stored in the EVEN cell. In order to accumulate this sum in the EVEN location without losing all the information already stored there, a function known as the split cycle must be employed. For part of the cycle of the bit selector the memory is called upon to perform one step and for the remainder it is commanded to execute a different operation. In the case of the 4-bit multiplication above, the three least significant bits of both EVEN and ODD cells must be swept through without change. When the fourth bit is selected, however, the inhibit add operation of the EVEN location is activated and this bit is regarded as the inhibit add bit. If this bit contains a 1 then the shifted word in the ODD cell is added to the EVEN cell. Since the ODD word has been left shifted three bit positions with trailing zeros, addition from the fourth bit onward is acceptable and correct. Once the fourth bit of the multiplier is checked for a 0 or a 1 on the data latch pulse it is no longer required and can be lost. On the fourth bit position then, the accumulate data line of the EVEN cells, ZACE, is taken to the 0 voltage level and the bit is destroyed. The data contained in the fourth bit of the ODD cell is entered into its place. For the fifth and succeeding bit

positions, however, normal addition takes place and the accumulate data instruction is returned to the high level. This completes the first of the four additions required for the 4-bit multiplication. Before the second addition, the multiplicand in the ODD location is right shifted one bit position and the control unit readjusts itself so that the third bit of the EVEN cell will be accepted as the inhibit add bit. The first two bits are swept by without alteration and on the third bit position the inhibit add function is again activated and the accumulate data instruction is dropped for EVEN cells. An add ODD to EVEN attempt is made the success of which depends on the data contained in the third bit of the EVEN cell. If this is a 1 then the l.s.b. of the multiplier is directly entered into the third bit position. For the fourth and following bits, normal ODD plus EVEN addition is commanded. At the end of two additions bits y_3 and y_2 of the multiplier in Figure 3:3(b) have been lost and the EVEN cell will contain part of the y' accumulated sum as well as the two least significant bits of the multiplier. The final two additions are carried out in a similar manner as the first two with the inhibit add bit being taken as the second l.s.b. and finally the l.s.b. itself. At the completion of this last cycle the multiplication is complete. The product has replaced the multiplier in the EVEN cell location and the multiplicand resumes its unshifted position in the ODD.

Demands which at first glance may appear rather severe on the control unit relax considerably when a deeper study of the process is made. Since the left shifted multiplicand is always followed by trailing zeros, an add odd to even command can be made without fear of losing the multiplier stored in the EVEN cell. The instruction can then be activated for the entire cycle and only in the case of the current inhibit add bit must any modification be made. For this bit position the accumulate even data command must be dropped so that this bit of the multiplier is destroyed. Such a feature is easily incorporated into the control unit. Again note that each EVEN and ODD pair acts as an independent unit and thus all multiplications are carried out simultaneously.

3.3.12 Location Shifting

In addition to the operations between EVEN and ODD cells of a particular PAIR, facilities are provided for interaction between adjacent PAIRS. The entire contents of all EVEN or ODD cells can be transferred in one cycle of the bit selector. The bidirectional shifting pattern is shown in Figure 3:1. Referring again to Figure 3:2, the AND-OR-INVERT gate provides two additional sources to the A input of the adder, one for up-shifting, the other for down-shifting. When the even shift up instruction line, ZESU, is held at the high voltage level, data enters the adder through the TLEI data transfer line. This line is permanently wired to the

TDOO data out line of the adjacent lessor EVEN cell. Similarly data enters from the greater EVEN cell when the instruction even shift down, ZESD, is raised to logical 1. Since there is no B input or carry the sum output of the adder is simply the shifted word and this new word is written into the memory. The inhibit add option is not normally used for this operation since the blocking of any shifted word would result in the loss of that word.

3.3.13 Combinational Operations

Since some operations involve only the A input of the adder while others the B input, certain combinations of operations are permitted. Still others do not involve the adder at all and are used only to retrieve data from a cell to be used by another cell. Consider again for example, the multiplication EVEN = ODD x A. In its simplest form this multiplication involves the alternate repetitive use of two fundamental steps; EVEN = EVEN + ODD, and ODD = ODD x 2. There is no reason why these two steps cannot run concurrently. The multiplication of the ODD word by 2 in no way effects the simultaneous readout of the word already stored in the location. By combining these two steps, the total time for the entire multiplication is reduced by one half. In general, different operations for EVEN and ODD cells can be run concurrently since each is an independent unit. In computer solutions of lengthy problems, combining operations can add

up to a considerable saving of time and in such cases an analysis of the memory capabilities should be made so that the most efficient sequence of steps is utilized combining as many steps as possible.

3.4 Control Unit

The control unit for the computing memory is basically an interface between the memory and the device that it is to be a part of. This device will in general be a computer for which the computing memory is to be a special purpose peripheral unit. Since the exact requirements of the control unit are not clearly defined in the general case, it is of little use to design a completely all purpose unit. The control unit described here contains the circuitry necessary to drive the memory including the loading of data and the extraction of information stored in the array. The design is reasonably flexible so that interfacing with some central device is a simple matter adding a few gates and instruction decoders.

3.4.1 Bit Selector and Pulse Switch

The heart of the control unit is a 4-bit up down counter and a two position switch. The counter with decoders and line drivers acts as the cyclic bit selector while the switch steers clock pulses alternately to the data latch and write enable lines of the memory cells. The synchronous

reversible 4-bit counter is shown in Figure 3:4. Consisting of four 7473 J-K flip flops, four 7400 2 input NAND gates, and six 7430 3 input NAND gates, the entire counter can be condensed into five I.C. packages. In keeping with most recent developments in I.C. technology where monolithic up-down counters are becoming available, this circuit will be referred to only as a block in subsequent figures.

The counter, decoders and switch are shown schematically in Figure 3:5. This circuit is similar in form and operation to that of Figure 2:4 described in Chapter 2 with additional gating included to provide for the inhibit add and negation features. In this simple control unit, the inhibit add bit is fixed, always the first bit in the bit selector cycle, and so the operation EVEN = EVEN x ODD is not readily available.

As noted earlier, care must be taken to prevent alteration of the inhibit add bit in certain operations. For example, forming the two's complement of a number must not result in the inhibit add bit of that number being inverted. Nor must the inhibit add bit be altered when the contents of a cell are right or left shifted. Finally, when an ODD and EVEN cell are added together, neither inhibit add bit should be affected. To ensure that these three cases for both EVEN and ODD cells are provided for, a condition is imposed on the write enable pulse, WWCL. The condition

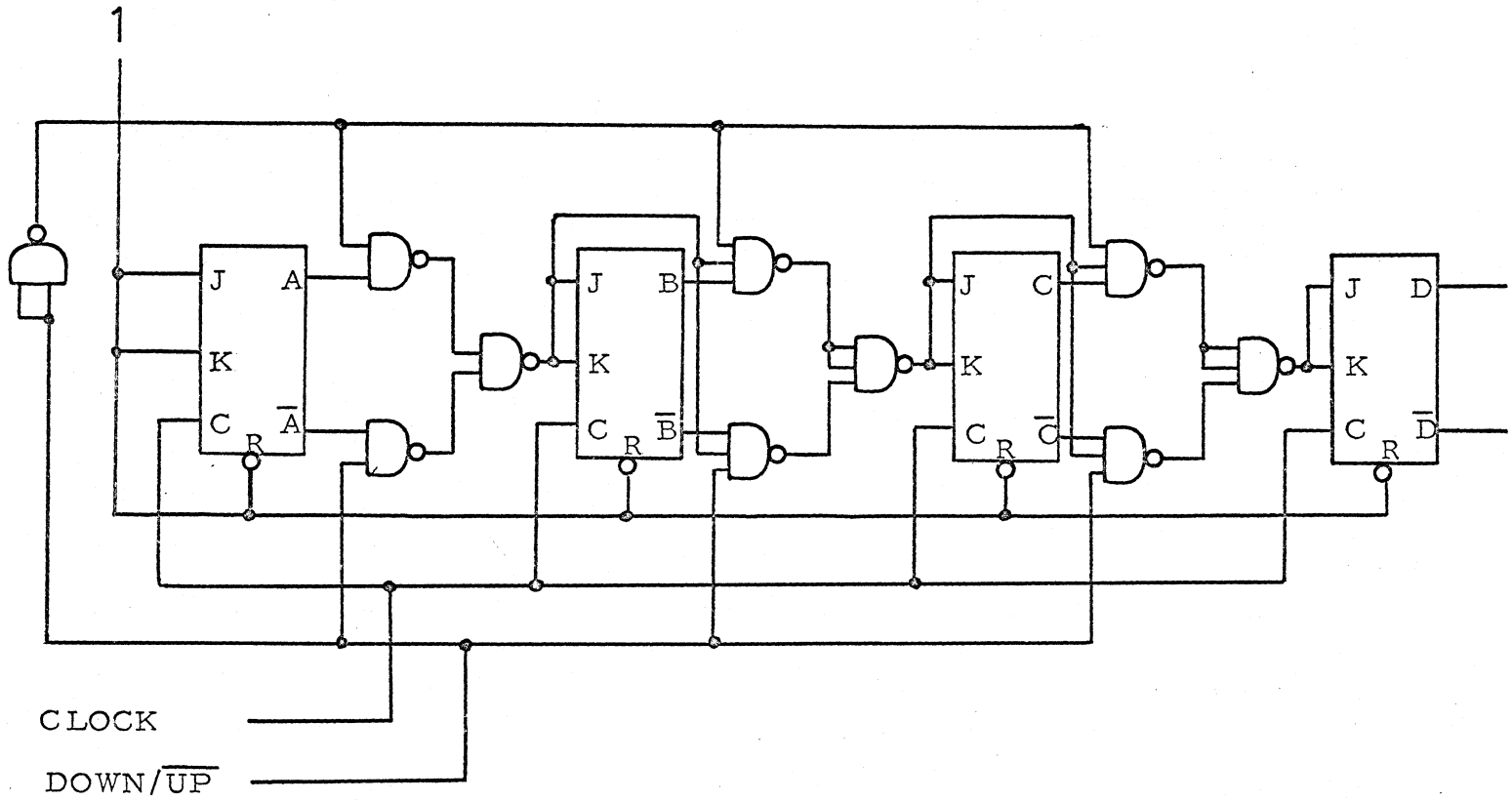


FIGURE 3.4 4 BIT UP-DOWN COUNTER

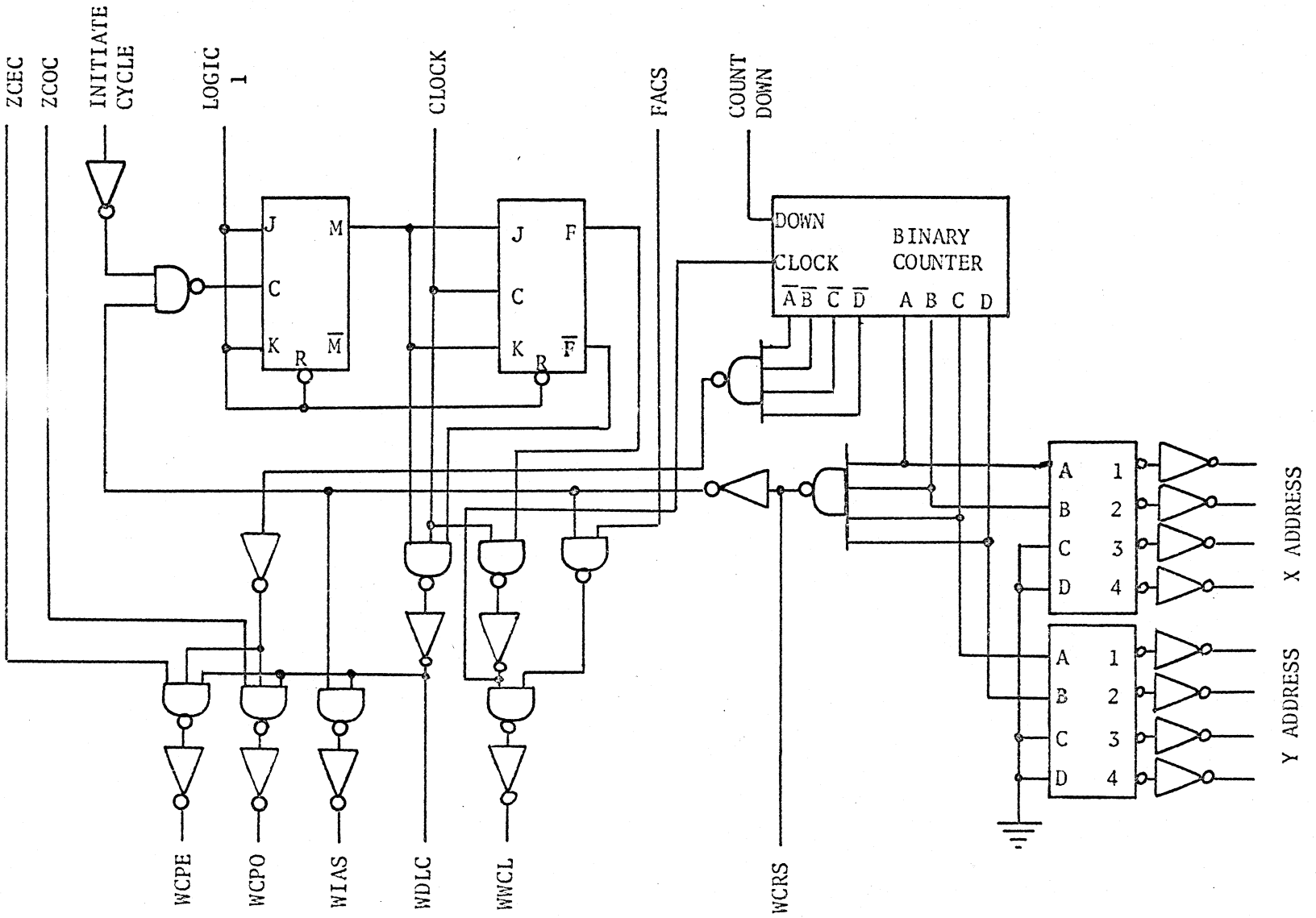


FIGURE 3.5 BIT SELECTOR AND PULSE DIRECTOR

allows that when the inhibit add bit position is detected by the 4 input NAND gate at the counter and any of the above instructions are in force then the write enable pulse will be blocked. A check on these instructions can be expressed as the OR function FACS such that

FACS = ZAEO + ZAOE + ZISE + ZISO + ZCEC + ZCOC.

Realization of FACS is discussed later in this section where all instruction lines are covered.

Also included in Figure 3:5 is the gating required to trigger the inhibit add bit latches at the same time as the data latches. This is provided for in a simple 2 input NAND gate and inverter that passes the data latch pulse only when the inhibit add bit is selected.

As discussed in Chapter 2, the starting point of the reversible counter is always with all bits in the logical 1 voltage level. For an up count then, the next state will be with all bits in the low level. Since the first bit position selected is always the inhibit add bit, the second will be the l.s.b. of the word stored in the memory. When the two's complement of the word is to be formed, it is required to direct an inverted pulse to the carry preset of the corresponding memory cell at the same time as the data is latched. Detection of the l.s.b. position is provided by a 4 input NAND gate and an inverter, the output of which is combined with the data latch pulse, WDLC, and either the complement

even data, ZCEC, or complement odd data, ZCOC, to preset the necessary carry flip flops, WCPE for EVEN cells and WCPO for ODD cells.

The only additional feature shown on Figure 3:5 is the instruction line ZRCS which when taken to the high level will reverse the direction of the counter, for use when words are to be shifted toward the lessor significant bit. The inverters at the output of the decoder utilize descrete transistors to supply the heavy current necessary to drive the address lines of the memory.

3.4.2 Data Load and Data Out Control

Since only a single cell can be loaded or read at a time, some system of addressing each cell location individually must be provided for in the control unit. Figure 3:6 illustrates a partial schematic of this cell identification system. Each of the 64 cell locations is assigned a six bit binary number represented by the subscripted characters $C_5C_4C_3C_2C_1C_0$, even numbers 0 through 62 corresponding to EVEN cell locations and odd numbers 1 to 63 for the ODD locations. A common cell address is used for both loading of data and reading out of stored data. For convenience, the entire 64 word array is subdivided into groups of 8 words, 4 EVENS and 4 ODDS. Each group in turn possesses its own binary to octal decoder for entering information and 8 bit multiplexer for selecting the proper cell to be

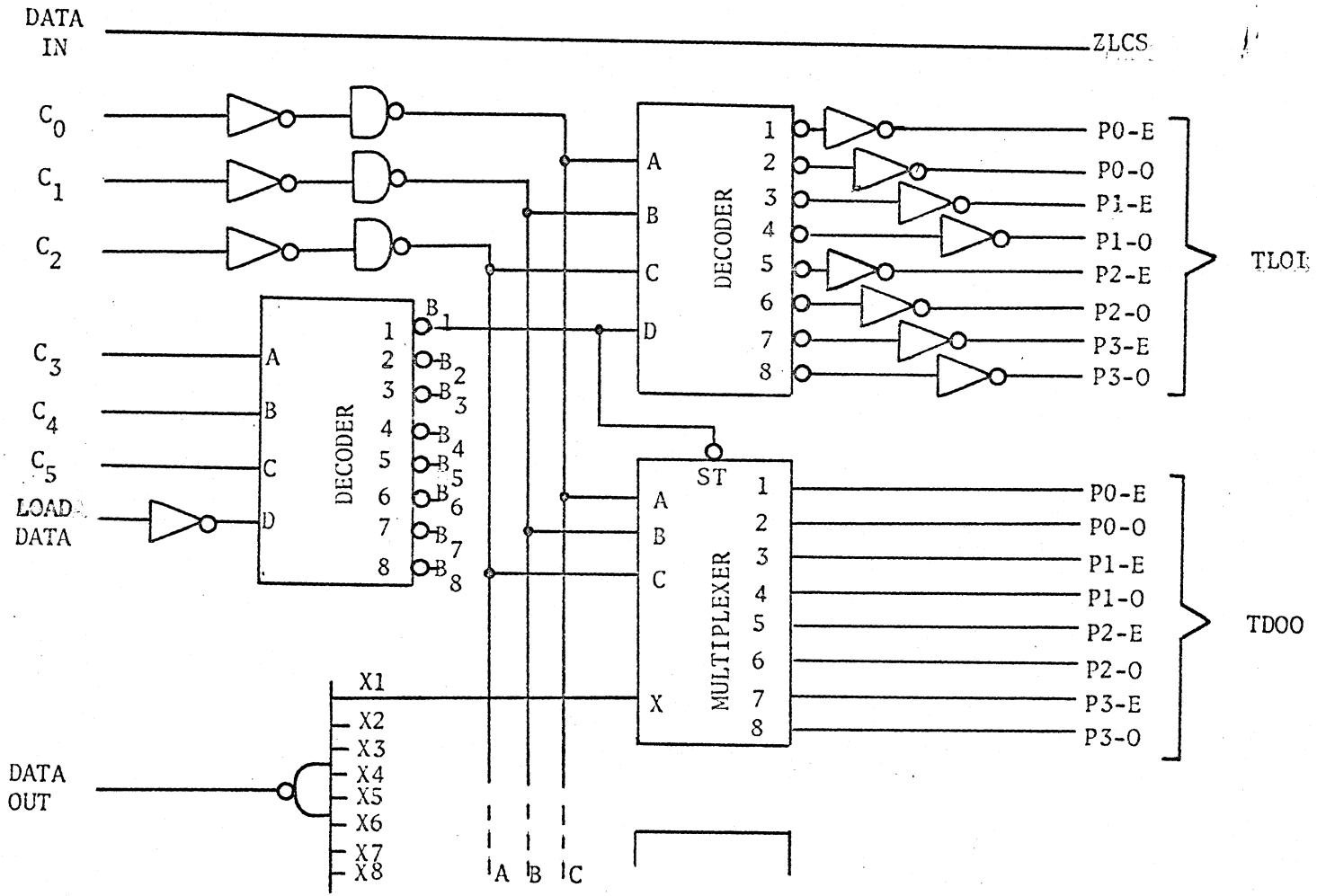


FIGURE 3.6 DATA LOAD AND DATA OUT CONTROL

read. Only one such group is shown in Figure 3:6 but since all other groups are identical the total system is easily understood by examining a typical section. Recall again Figure 3:4 showing the single cell. As described earlier, data is entered to the cell through the B input to the adder through the ZLCS load data line when the appropriate TLDI cell select line is raised to the high logic level. This high level is derived from the decoder-inverter combination of the group that the cell in question belongs to. The inverters are required since the 7442 decoders feature complementary outputs. One of the 8 cells in each group is thereby selected by the binary number represented by the 3 least significant bits of the cell address. Inverters and buffers are provided in these lines to reduce the load of the eight decoders and eight multiplexers. Since every decoder shares the common address lines, one cell for every decoder would be selected at once. To prevent the simultaneous loading of 8 cells with identical information, the three most significant bits of the cell address are decoded in such a manner that only one group decoder is activated at a time. The 7442 decoders are in fact BCD to decimal decoders used as 3 bit binary to octal decoders. If the D input to these decoders is raised to the logical 1 level then none of the octal outputs are driven. This D input therefore acts as a group select when it in turn is driven by the complemented output of another binary to octal decoder. This new decoder

is coded by the three most significant bits of the cell address just as the other 8 decoders are coded by the three least significant bits. The 8 outputs of the m.s.b. decoder, labelled B_1 through B_8, each lead to the D input of the corresponding group decoder. Since the m.s.b. decoder is also a B.C.D. to decimal decoder the D input can be used to activate the load data function. Once the cell corresponding to the unique coded address is selected, data is entered through the DATA IN line shown in Figure 3:6 which is directly connected to all 64 ZLCS load data lines.

Reading out of data is a simple matter of addressing the desired cell and running through a cycle of the bit selector. A 74151 multiplexer is used for each of the 8 groups of cells, with one input tied to the TDOO data out line of each cell. The multiplexers are coded in a similar manner to the decoders by the three least significant bits of the cell select. Each multiplexer is also provided with a strobe input which must be held in the low state before any data can be transmitted. When this strobe input is joined to the D input of the adjoining decoder, then only one of the 8 multiplexers will be activated at a time, and only one of the 64 cells will be considered. The X output of the multiplexer yields the complement of the data selected by the cell address code, and when all 8 outputs are led into a common 8 input NAND gate, the output of this NAND gate becomes the data stored in the desired cell. In this manner,

any one of 64 inputs can be selected to appear at a single output terminal.

3.4.3 Instruction Line Driver

This section of the control unit is simply a terminal point for all instruction lines. All lines to the memory array are brought to the control unit so that the computing memory may be easily adapted to any other device. Since the instruction lines may drive as many as 64 cells at once a system of buffers is required. These buffers play no part in the logic of the control unit and will therefore not be included in this discussion. The instruction line driver is shown in Figure 3:7. Included in this figure is the gating required for the generation of the FACS function discussed in Section 3.4.1 and used in Figure 3:5.

3.5 Physical Layout

With the exception of memory address drivers and pull up resistors, the entire computing memory and control unit is composed of integrated circuits mounted on nine 70 socket boards. The cell arrangement places 4 EVEN and 4 ODD cells on a single board for eight of the boards and an extra board for the control unit. No particular care was taken to minimize the size of the device since a 64 word memory is too small to be of much practical use in any case. Future developments in large scale integration technology directed

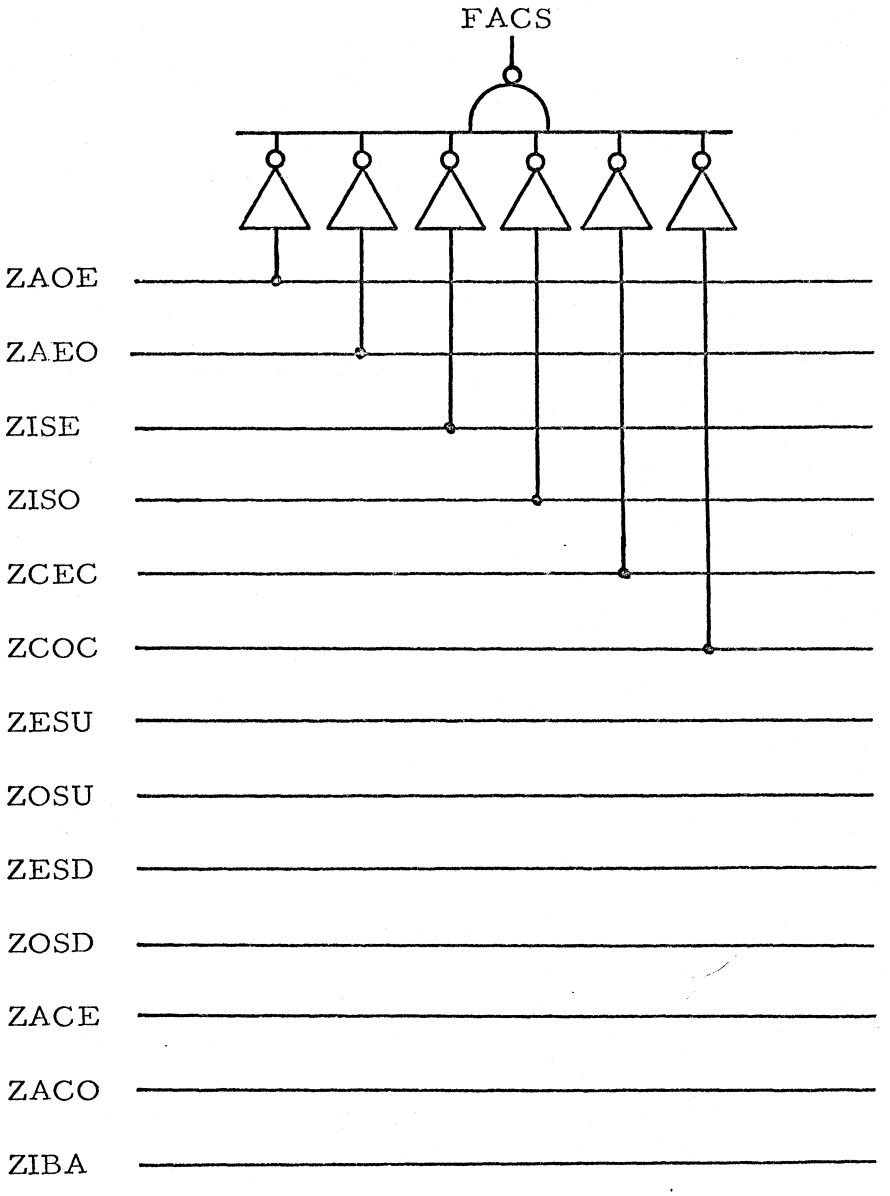


FIGURE 3.7 INSTRUCTION LINE DRIVER

toward inclusion of an entire cell in a single package should reduce the size and cost of the system enormously. All integrated circuits are either 14-pin or 16-pin dual in line packages mounted to the boards with wire-wrap sockets. All pin connections are wire wrapped with the exception of V_{cc} and ground. Due to the heavy load imposed by the memory address lines, the power requirements are rather high at approximately 10 amp. 5 volt D.C. The circuit diagrams and pin connections are included in Appendix B and need not be mentioned here.

3.6 Summary

In Chapter 2, the philosophy of the computing memory was introduced: certain frequently encountered numerical problems involving a myriad of identical arithmetic operations may be very efficiently solved by an arithmetic computer memory array. Solution of potential field problems by the method of relaxation was presented as a case in point. Recall again the conventional computer solution by relaxation. The equation for a single point relaxation is repeated for convenience.

$$P_{x,y} = \frac{1}{4}(P_{x-1,y} + P_{x+1,y} + P_{x,y-1} + P_{x,y+1}) \ldots . 2.1$$

For a single relaxation equation 2.1 must be applied to each and every point in the descrete field individually, a tedious and time consuming exercise.

Consider now how the computing memory developed in this chapter might be applied to the relaxation problem. Figure 3:1 illustrated the cell arrangement and shifting pattern for the 64 word array. Assume that the potential field is rectangular and can be divided into a uniform 4 x 8 matrix of 32 points. Any shape of field is permissible, the rectangular field is chosen as it conveniently matches the entire memory array. For the complete solution the EVEN cells will be used to store the current approximation to the potential at that point and the ODD cells will be used for computation of new values. As a preliminary step all memory locations are cleared and boundary values are loaded into the EVEN cells at the appropriate points. Since these boundary values must remain fixed, the contents of these cells are protected by inserting zeros in the inhibit add bit of each location. All interior points both EVEN and ODD cells, have ones in their inhibit bit positions. The memory is now ready for its first relaxation.

Figure 3:8 shows a magnified five point section of the 32 point array. In order that this section be assumed general and to conform to the notation of equation 2.1, cells are labelled with a uppercase letter P and two subscripts corresponding to the x and y matrix position. Only one point $P_{x,y,}$ is considered but it must be remembered that this is a general point and every step taken here is simultaneously executed at every other point in the matrix.

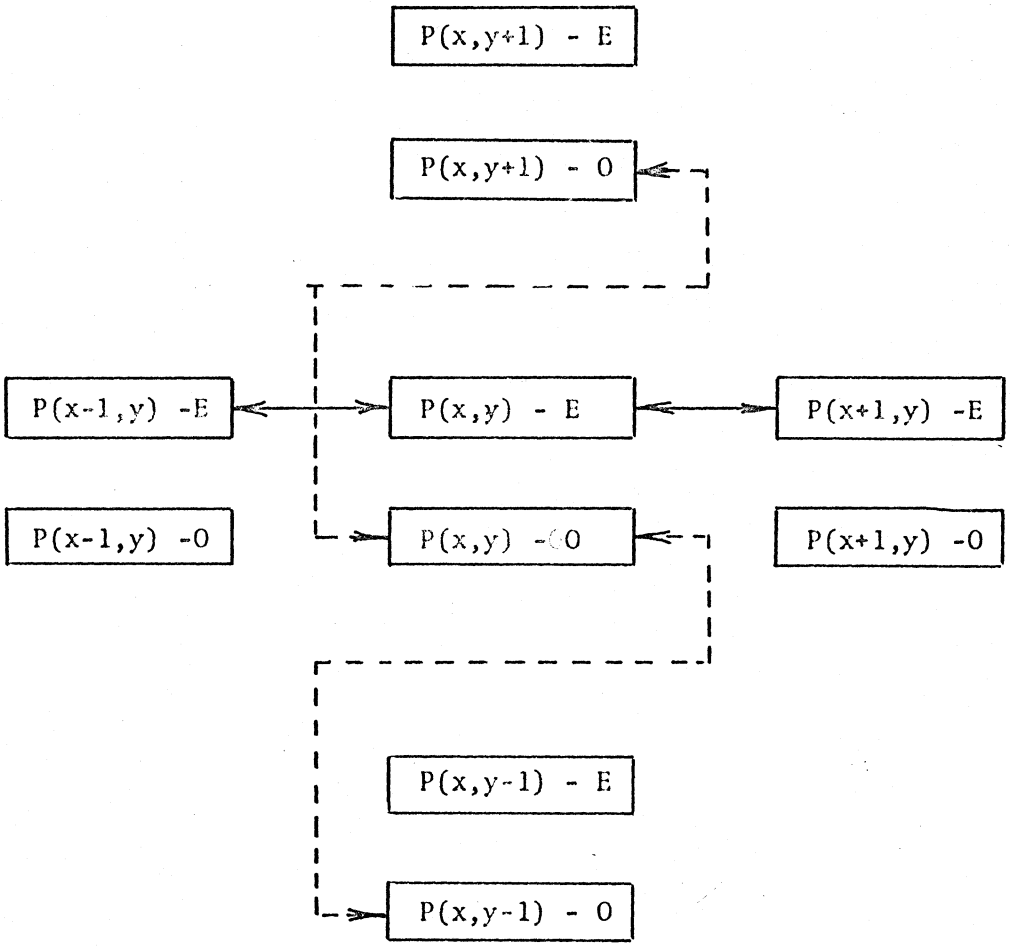


FIGURE 3.8 MAGNIFIED SECTION OF COMPUTING MEMORY ARRAY

The first step is an up-shift of all ODD cells. The word previously contained in cell $P_{x,y}$ will now reside in $P_{x,y+1}$. Since all ODD locations are initially cleared this step is in fact unnecessary. It is included here so that the movement of point $P_{x,y}$ may be better understood. An add EVEN to ODD operation followed immediately by a down-shift of all ODD cells will effect a transfer of the value of the potential at point $P_{x,y+1}$ into the ODD location of $P_{x,y}$. This may be expressed as

$$P_{x,y} = P_{x,y+1}$$

A down-shift of all ODD locations will transfer this value previously located in the $P_{x,y}$ position to the $P_{x,y-1}$ position. Again EVENS are added to ODDS and the ODDS are up-shifted back to their undisturbed positions. The word in the ODD $P_{x,y}$ cell may now be expressed

$$P_{x,y} = P_{x,y+1} + P_{x,y-1} \quad .$$

ODD cells are now retained in their positions and all EVEN cells are up-shifted. The value of the potential at point $P_{x-1,y}$ is now stored in the EVEN cell of point $P_{x,y}$. Simultaneous addition of EVENS to ODDS and down-shifting of EVEN cells will return all potential values to their proper points and update ODD cells to include yet another value.

$$P_{x,y} = P_{x,y+1} + P_{x,y-1} + P_{x-1,y}$$

The next step is obviously to down-shift EVEN cells, again add EVENS to ODDS, and return the EVENS with a single up-shift. The contents of the ODD cell at point $P_{x,y}$ may now be expressed

$$P_{x,y} = P_{x,y+1} + P_{x,y-1} + P_{x-1,y} + P_{x+1,y}$$

It remains only to divide this number by 4 by shifting the contents of all ODD cells by two bit positions toward the lessor significant bit. At the completion of this division the ODD cells of the memory contain the approximations to the potential updated by one relaxation. The numbers previously contained in the EVEN cells can now be destroyed, values in the ODD cells transferred into the EVEN, and the ODD cells cleared, all in a single step. The memory will then be set up for the next relaxation. The complete relaxation cycle can be summarized into 10 steps:

1. shift ODDS up
2. add EVENS to ODDS
3. shift ODDS down
4. shift ODDS down
5. add EVENS to ODDS
6. shift ODDS up and shift EVENS up
7. add EVENS to ODDS and shift EVENS down
8. shift EVENS down
9. add EVENS to ODDS and shift EVENS up
10. clear EVENS, add ODDS to EVENS, and clear ODDS.

Since all boundary points are guarded by the inhibit add feature and the memory encorporates an end around shift, there is no danger of losing these values. A complete relaxation involves only 10 cycles of the memory bit selector and the same time is required to relax 10 points or 10,000 points. For fields with many points then, the computing memory is very efficient.

Chapters 2 and 3 have traced the development of the computing memory from the stage of a mere principle to a full scale application. A single application, however, would scarcely seem to warrant the development necessary to bring a practical memory into production. References thus far have been made solely toward applying the computing memory as a special peripheral unit for a programmable data processing computer. Recent advances in computer miniaturization and in analogue to digital conversion techniques have brought the special purpose computer into the realm of instrumentation. More and more sophisticated measuring instruments have been demanded, designed, and produced and inadvertently the instrument designers have found themselves deep in the field of computer design. That is not to say that this is an undesirable situation. General purpose computers are costly, must be programmed and are in some cases very inefficient. There is considerable advantage in a computer that needs no programming and can be readily transported for on line measurement or control.

It is just such a computer "instrument" to which the remainder of this thesis is devoted. A major portion of the digital correlator described is the storage facility. Many individual operations must be performed and many results must be retained, an excellent application for the computing memory. Again, since this is a special application of the memory where only a few of its many features will be required, the general memory described in this chapter may well prove to be uneconomical. In such a case it is advisable to return to basics and consider the basic arithmetic computing memory cell described in Chapter 2 and utilize this as the unit building block for the instrument memory. It was with this in mind that the control unit for the computing memory was dealt with in such general terms. Once the principle of the memory is established and justified the general unit can be tailored to specific applications.

CHAPTER 4

THEORY OF DIGITAL CORRELATOR

This chapter describes the theory of amplitude and time quantized signals and applies this theory to the problem of correlation. An algorithm is developed whereby the correlation function may be realized using digital techniques. The method of development of this algorithm follows essentially that of Masuko[12] in his design of a digital autocorrelator. The principles used in this derivation were originally proposed by Deist and Kitai[13] in a paper describing a technique for obtaining the mean square value of a fluctuating voltage by digital methods.

4.1 Theory of Amplitude and Time Quantised Signals

Consider the voltage waveform shown in Figure 4:1. For simplicity in this case it is assumed that this voltage is always positive. The voltage may be quantized into n descrete levels of amplitude designated by the integers 0 to n. At a particular instant in time, say t*, the voltage is sampled and found to lie somewhere in the interval bounded by m and m+1. In the processing technique considered here, the probability of the voltage falling within a particular interval

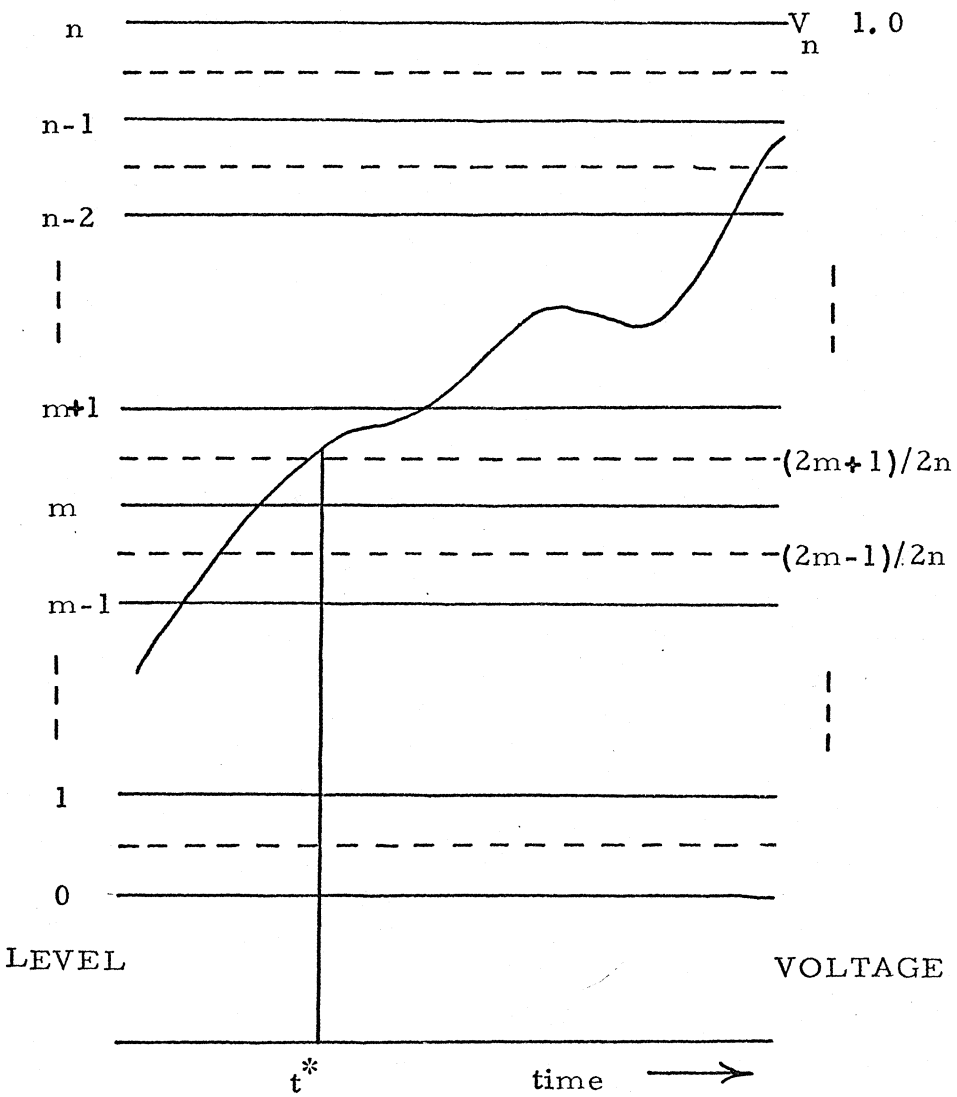


FIGURE 4.1 QUANTISED VOLTAGE

is of concern. The probability distribution of the voltage within each interval must necessarily be assumed uniform with a mean at the mid interval value. If the voltage at the nth level is taken as unity, then the sample taken at time t* will be considered $\frac{2m+1}{2n}$, the mid-interval value.

A linear quantiser may now be employed to transform the sampled voltage to m, an integer representing the highest level exceeded. In processing this sample, m must be converted to the mid-interval value $\frac{2m+1}{2n}$. This is the technique used by Masuko in the derivation of the algorithm realized in his auto-correlator.

A somewhat simpler algorithm results if the half interval shift is incorporated into the quantiser itself. In this case for a sample falling somewhere between the m and m+1 levels the quantiser will be called upon to designate an integer representing the mid-interval value directly. If we call this integer r, then

$$\frac{r}{n} = \frac{m}{n} + \frac{1}{2n}$$

$$r = m + \frac{1}{2} \qquad \text{.....4.1}$$

The transformation suggested by equation 4.1 can be regarded as a simple up-shift of the input to the quantiser by a half level. In Figure 4:2 the solid lines indicate the quantiser levels and the broken lines the shifted input. For a

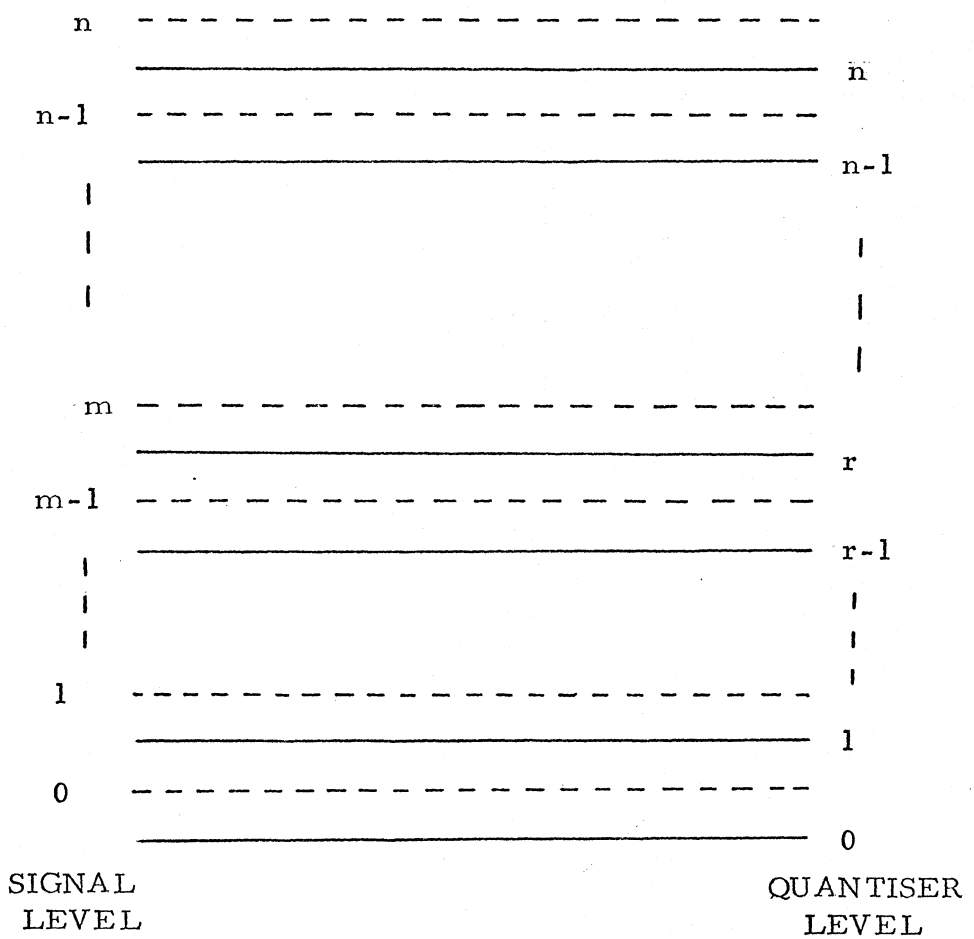


FIGURE 4.2 SHIFTED INPUT TO QUANTISER

sample now falling in the interval m - $\frac{1}{2}$ to m + $\frac{1}{2}$ the quantiser will indicate the integer r. This integer r is in fact the new mid-interval value of the shifted sample and can be used in processing as representative of this value.

4.2 Derivation of Algorithm for Correlation

The most general form of correlation function is the cross correlation function and is therefore considered in this derivation. The autocorrelation function is merely a special case where one signal alone is examined.

Consider two processes represented by the fluctuating voltages v_1 and v_2. If the two processes are ergodic with respect to their correlation function,[14] then the time averaged correlation function $R'_{v_1 v_2}(\tau)$ is expressed as

$$R'_{v_1 v_2}(\tau) = \underset{T\to\infty}{\text{Lim}} \frac{1}{T} \int_0^T v_1(t)v_2(t+\tau)dt.$$

Let $p(x,y:\tau)$ be the probability that v_1 lies within an interval dx about x at time t and v_2 within an interval dy about y at a time τ seconds later. Then the ensemble averaged correlation function is written

$$R_{v_1 v_2}(\tau) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xyp(x,y:\tau)dxdy \qquad .....4.2 \quad .$$

In the case of a sampling system, v_1 and v_2 are quantised in amplitude and time and the double integral of equation 4.2 is replaced by a double summation over the

quantised range of v_1 and v_2. In figure 4:3 both v_1 and v_2 are considered always positive and quantised into n intervals, the nth being unity. Let $p(r,s:\tau)$ be the probability that v_1 lies within $\frac{1}{2n}$ of level r at time t and v_2 within $\frac{1}{2n}$ of level s at a time τ seconds later. If these values of v_1 and v_2 are then represented by $\frac{r}{n}$ and $\frac{s}{n}$ respectively, the ensemble averaged correlation function is given by

$$R_{v_1v_2}(\tau) = \frac{1}{n^2} \sum_{r=0}^{n} \sum_{s=0}^{n} rsp(r,s:\tau) \qquad .....4.3$$

As shown previously the probability that a voltage lies within $\frac{1}{2n}$ of a level can be equated to the probability that a particular level is the highest level exceeded if the voltage under observation is shifted by $\frac{1}{2}$ level. If the signal voltages are both shifted positive by $\frac{1}{2}$ level then $P_{r,s}$ is the probability that r is the highest level exceeded by $v_1 + \frac{1}{2n}$ at time t and s is the highest level exceeded by $v_2 + \frac{1}{2n}$ at a time τ seconds later. For voltages confined in the range 0 to $1 - \frac{1}{2n}$ volts, equation 4.3 may be rewritten

$$R_{v_1v_2}(\tau) = \frac{1}{n^2} \sum_{r=1}^{n-1} \sum_{s=1}^{n-1} rs\, P_{r,s} \qquad .....4.4$$

If n is large, then the half level restriction on the range of v_1 and v_2 will be of little concern. In practise it is convenient to superimpose v_1 and v_2 on a d.c. voltage to permit processing of positive and negative voltages. A d.c. voltage bias of $\frac{n+1}{2n}$ will permit near equal positive and negative excursions of input voltage as well as realizing the

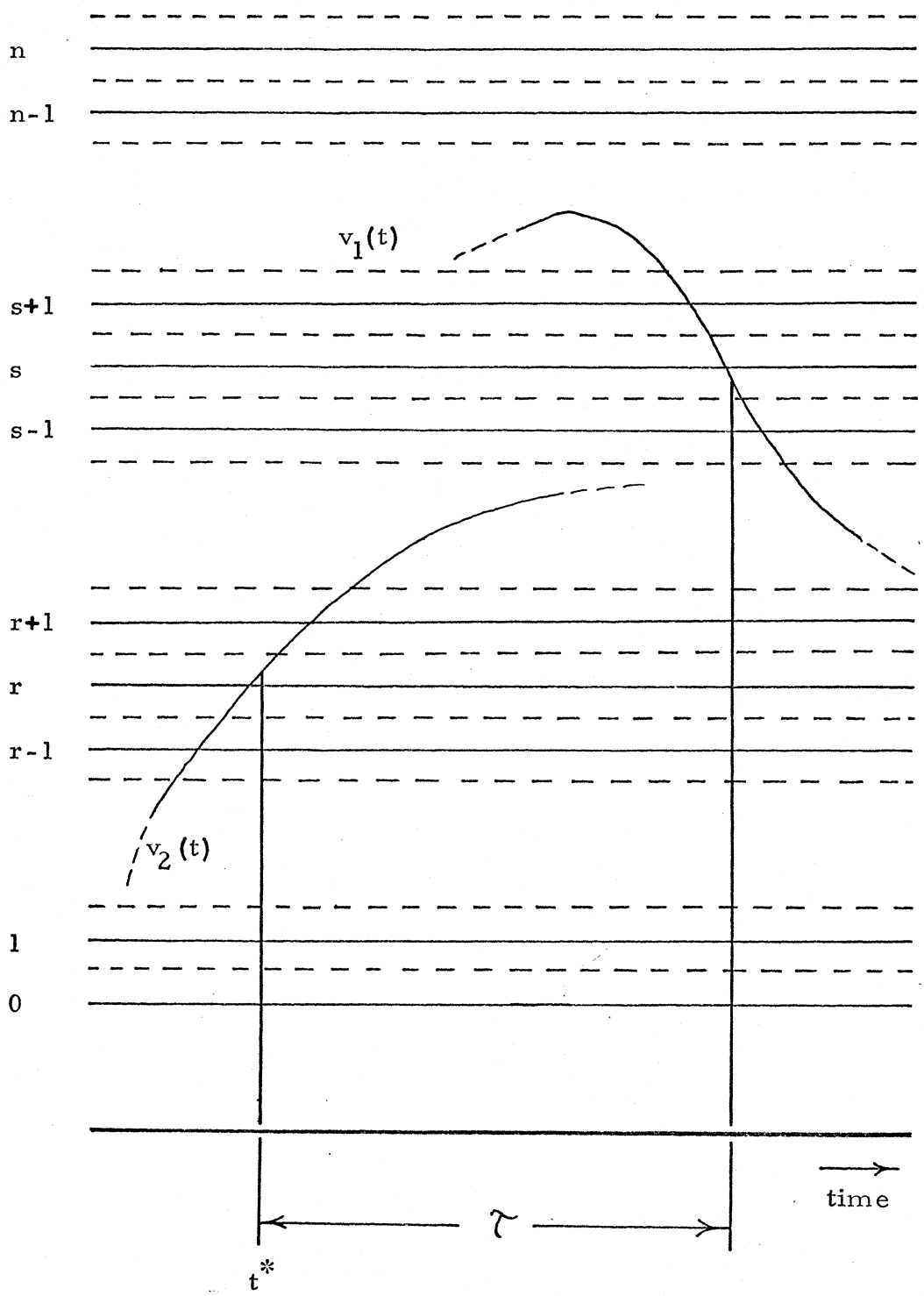


FIGURE 4.3 AMPLITUDE AND TIME QUANTISATION OF v_1 AND v_2

necessary half level shift. In this way, the need for precision rectifiers, subtractors, and up-down counters is eliminated. However, the d.c. bias must be accounted for in the final processing for the correlation function.

For a process monitored for a finite time, the probabilities of equation 4.4 may be replaced by the fraction of the total measuring time that the voltage spends in the particular interval. If T_0 is the total time for which the process is observed and $T_{r,s}$ is the total time for which r is the highest level exceeded by $v_1 + \frac{1}{2n}$ and s the highest level exceeded by $v_2 + \frac{1}{2n}$ τ seconds later, then

$$P_{r,s} = \frac{T_{r,s}}{T_0} \quad .$$

Equation 4.4 can be further adapted to a time quantised system if T_0 is replaced by C_0, the total number of samples taken within the measuring time, and $T_{r,s}$ is replaced by $C_{r,s}$, the number of samples taken for which the conditions of r and s apply. Equation 4.4 may now be regarded as

$$R_{v_1 v_2}(\tau) = \frac{1}{n^2 C_0} \sum_{r=1}^{n-1} \sum_{s=1}^{n-1} rs\, C_{r,s} \qquad .....4.5$$

Equation 4.5 implies that a separate count must be kept for each $C_{r,s}$ over the time of measurement. That is, (n-1)(n-1) separate counts are to be maintained and added at the completion. In order to avoid such mass storage, it would be preferable, at each sampling instant, to calculate the contribution of that particular sample pair to the

$R_{v_1v_2}(\tau)$ sum. This is readily accomplished if for each sample pair collected, a weighting number is calculated equivalent to the contribution of that particular sample pair to the $R_{v_1v_2}(\tau)$ sum. This weighting number, a function of r and s, is then fed to a single accumulator prior to collecting the next sample pair for processing.

For a sample pair collected, represented by the r and s levels exceeded, a count of unity is added to the $C_{r,s}$ count. The contribution of this pair to the final sum then is simply rxsxl and the weighting factor is merely rs. At the completion of the measuring time when all C_0 weighting factors have been accumulated it remains only to divide the accumulated sum by n^2C_0 to obtain $R_{v_1v_2}(\tau)$, the correlation function.

4.3 Derivation of Algorithm for Mean Value

The time average of a fluctuating process represented by a voltage v is

$$\bar{v} = \lim_{T\to\infty} \frac{1}{T} \int_0^T v(t)dt \ .$$

If p(x) is the probability that v lies within an interval dx about x then the ensemble average is expressed by

$$\bar{v} = \int_{-\infty}^{\infty} xp(x)dx \qquad .....4.6$$

Again, for a sampling system, where v is quantised in amplitude and time, the integral of equation 4.6 is replaced by a summation over the range of v. If v is assumed always

positive and quantised into n levels of amplitude, the nth being unity, then a voltage at the rth level may be represented by $\frac{r}{n}$. p(r) is then the probability that v lies within an interval $\frac{1}{2n}$ about r and the ensemble average is now written,

$$\bar{v} = \frac{1}{n} \sum_{r=0}^{n} r\, p(r) \qquad \ldots\ldots 4.7$$

If v is shifted in a positive direction by a half interval then the probability that the voltage lies within $\frac{1}{2n}$ of level r can be expressed as P_r the probability that r is the highest level exceeded by $v + \frac{1}{2n}$. If this transformation is now applied to equation 4.7 and v is confined in the range 0 to $1 - \frac{1}{2n}$, $\bar{v}$ now takes the form

$$\bar{v} = \frac{1}{n} \sum_{r=1}^{n-1} r\, P_r \qquad \ldots\ldots 4.8$$

As in the case of correlation function, when the process is measured for a finite time, the probability P_r may be replaced by the fraction of the total measuring time the voltage $v + \frac{1}{2n}$ spends within the rth level. For a process observed for time T_0 where $v + \frac{1}{2n}$ spends a total time of T_r within the rth level equation 4.8 becomes

$$\bar{v} = \frac{1}{nT_0} \sum_{r=1}^{n-1} r\, T_r \ .$$

Replacing times by number of samples taken for which r is the highest level exceeded equation 4.8 takes the form

$$\bar{v} = \frac{1}{nC_0} \sum_{r=1}^{n-1} r\, C_r \qquad \ldots\ldots 4.9$$

C_0 is the total number of samples taken and C_r is the total number for which r is the highest level exceeded by $v + \frac{1}{2n}$.

Rather than maintaining the n-1 separate counts and summing at the end of the measuring run as implied by equation 4.9, it is preferable to calculate the contribution of each sample to the final product. In this way the processing of samples will keep pace with the sampling frequency and the $\bar{v}$ will be available immediately at the end of the measuring run. The contribution of each sample takes the form of a weighting factor which is a function of r the highest level exceeded. For one sample a single count is added to the appropriate C_r in equation 4.9. The contribution of this sample to the final sum, and hence the weighting factor, is simply rxl. It remains only to add this integer number to an accumulator before the next sample is collected for processing. At the end of the measuring run the accumulated sum is divided by nC_0 to give the mean value $\bar{v}$.

4.4 Effect of d.c. Bias

In order to handle both positive and negative going signals it is necessary either to use a rectifier or to add a d.c. bias to the input signal. A d.c. bias eliminates the need for rectifiers, up-down counters, and subtractors

but has the disadvantage that the final value of correlation function and mean value will be in error by an amount dependent on the d.c. bias voltage used.

Assume that a d.c. bias of W volts is added to both v_1 and v_2 inputs of a correlator. The measured time averaged correlation function then becomes

$$R'_0(\tau) = \frac{1}{T}\int_0^T [v_1(t) + W]\,[v_2(t+\tau) + W]\,dt$$

$$= \frac{1}{T}\int_0^T [v_1(t)\,v_2(t+\tau) + Wv_1(t) + Wv_2(t+\tau) + W^2]\,dt \qquad \text{.....4.10}$$

The first term in the integral of equation 4.10 is seen to be $R'_{v_1v_2}(\tau)$, the time averaged correlation function of v_1 and v_2. The second and third terms are the mean values of v_1 and v_2 respectively multiplied by a constant W. The last term is simply a constant set by the d.c. bias. Equation 4.10 may then be rewritten

$$R'_0(\tau) = R'_{v_1v_2}(\tau) + W\bar{v}_1 + W\bar{v}_2 + W^2 \ .$$

Rearranging the equation, a corrected expression for correlation function is obtained.

$$R'_{v_1v_2}(\tau) = R'_0(\tau) - W(\bar{v}_1 + \bar{v}_2) - W^2 \qquad \text{.....4.11}$$

$R'_0(\tau)$ is the measured correlation function taking into account the d.c. bias. To obtain the true correlation function of v_1 and v_2 it is then necessary to perform the

operations shown in equation 4.11.

The time average of a fluctuating voltage v(t) superimposed on a d.c. bias W is simply

$$V = \frac{1}{T} \int_0^T [v(t)+W]\, dt$$

$$= \bar{v} + W \quad ,$$

where $\bar{v}$ is the time average of the voltage v(t). To extract the mean value of v from the measured V it is then necessary only to subtract the d.c. bias voltage from the measured mean.

4.5 Summary

In this chapter the basic algorithms have been developed whereby the correlation function of two fluctuating processes and their mean values can be realized by digital means. This may be regarded as the software aspect of the problem. It now remains to design the hardware required to take this software and manipulate it in such a way as to produce the desired result. The hardware will take the form of a special purpose computer that has the capability of yielding the mean values and correlation function without any prior programming. Chapter 5 presents the detailed design of an instrument capable of measuring both the mean values and correlation function of two fluctuating voltages. If only a single voltage is examined then the

instrument yields the autocorrelation function of that voltage.

CHAPTER 5

DESIGN OF DIGITAL CORRELATOR

5.1 Correlation Section

The basic algorithm for the digital correlator, derived in Chapter 4, implies that for each time delay two numbers represented in binary form are to be multiplied together and their product, also in binary form, is to be stored in an accumulator. This is repeated for every sample pair collected and at the end of the measuring run it remains to divide the accumulated total by a constant to obtain the value of the correlation function for that particular time delay. If this is implemented for all desired time delays at once, then at the end of a single measuring run the complete correlation function will be available.

Figure 5:1 shows the simplified system block diagram for the digital correlator. Both $v_1(t)$ and $v_2(t)$ waveforms are continuously sampled and amplitude quantised by the A-D converters and converted to binary representations r and s respectively. These digital numbers are then fed into two delay lines to provide the suitable time lag between the v_1 and v_2 waveforms. The v_1 delay line is exactly one half

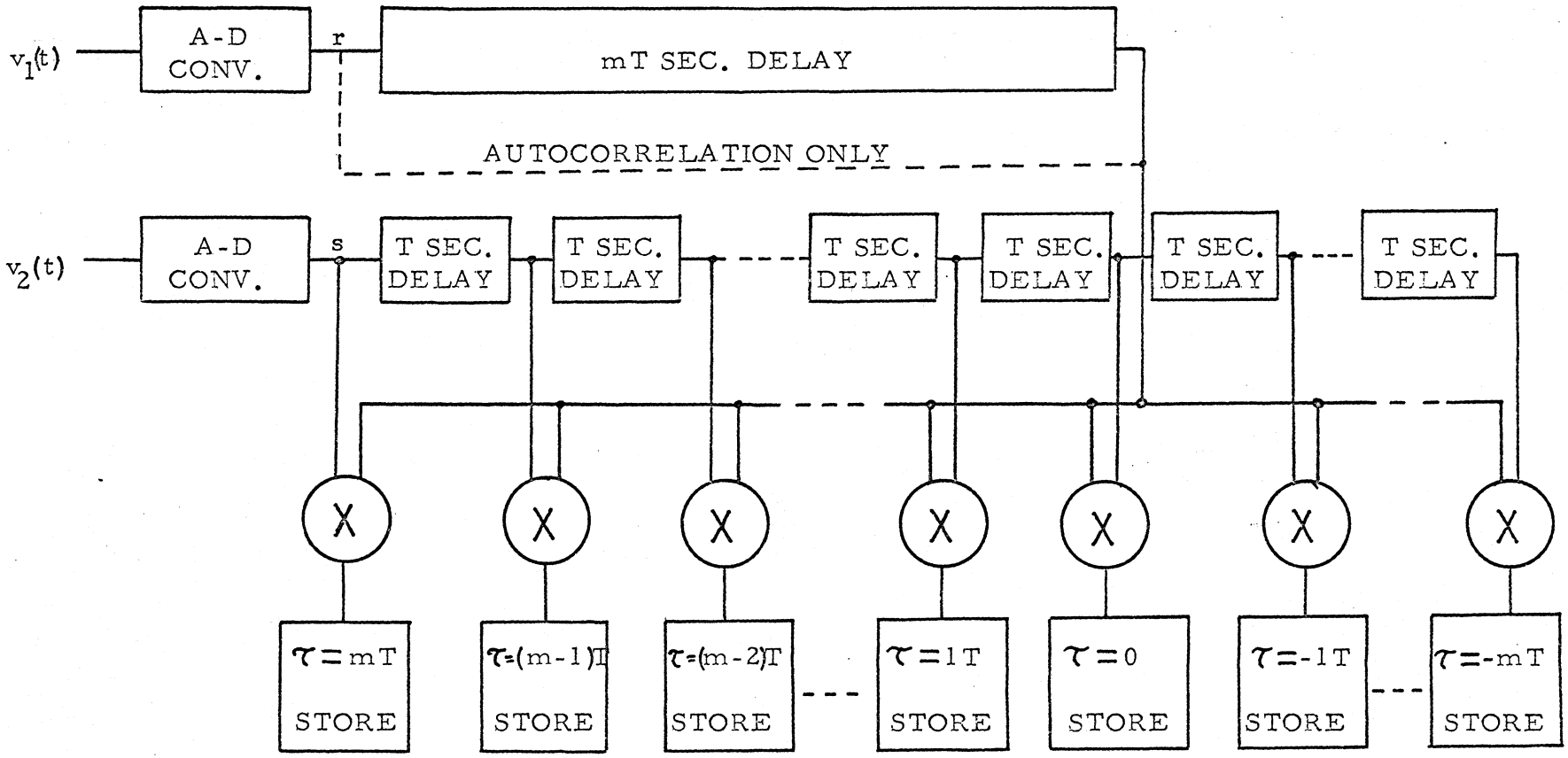


FIGURE 5.1 SIMPLEFIED SYSTEM BLOCK DIAGRAM FOR CORRELATOR

the v_2 delay so that both positive and negative values of τ may be plotted on the graph of $R(\tau)$. The v_2 delay is tapped at uniform intervals along its length corresponding to the values of τ on the correlation graph. At each tapped position the binary representation of the delayed v_2 sample is multiplied by the common v_1 sample and the product is added to an accumulated sum. In Figure 5:1 the increment of delay is assumed to be T seconds and there are 2m+1 unique sums to be accumulated corresponding to the range of delays

$$\tau = kT \quad \text{where} \quad k = 0, \pm 1, \pm 2, \ldots, \pm m.$$

The v_1 samples are delayed a fixed amount, mT seconds, so that when multiplied by the v_2 sample delayed through m stages of T seconds each, the effective time lag between the two samples is zero. When the system is used to determine the autocorrelation function, the v_1 delay may be eliminated so that only positive values of time delay are considered. If this were not done, then one half the storage accumulators would be redundant, since the autocorrelation function possesses even symmetry.

An inspection of Figure 5:1 reveals that the only complex part of the system lies in the multipliers and storage units. A digital time delay is very easily realized in a clocked shift register. What is required then is a device that will simultaneously multiply many digital numbers by a common multiplier and add the products into corresponding

accumulators, an operation well within the capability of the arithmetic memory cells. The cells will not only implement the required multiplication but also provide storage for the accumulated sums. A single central control unit will manipulate all cells at once and since all cells are independent, there is no limit to the number of delays that can be driven by the basic control.

We recall now the technique for binary multiplication discussed in Chapter 3. It is essentially a shifting and adding operation where the appropriate bit of the multiplier determines whether a particular addition will take place. Assume now that in Figure 5:1, r, the digitized sample of the v_1 waveform, will represent the common multiplicand and the 2m+1 delayed samples of the v_2 waveform will be considered the multipliers, s_k. For each delay position then the product rs_k will be added to the accumulation of all previous products. The multiplicand r is held in a register R that has the capability of shifting the contents one significant bit to the left after each addition. Whether or not this addition is carried out in each delay position is determined by the contents of the appropriate bit of the corresponding multiplier s_k. The multipliers are continually shifted down the delay line, S, so that after each addition step, the next m.s.b. is made available. At the end of the last addition a further shift of the multipliers will bring the l.s.b. of the next sample into position so that the next

multiplication may commence.

Figure 5:2 shows the flow chart for the process. To ensure that negative delays are available at the start of the measuring run, the A-D converter and shift register delays are continuously active even before the computing is begun. For the eight bit word length considered here, the converters must obtain a new sample after every eight shifts of the delay line so that the register is always filled. C_0 is the total number of samples to be taken for a complete measuring run and COUNT is a record of the number of samples processed thus far. A measuring run is initiated by resetting COUNT to zero and terminated when the number of samples taken reaches C_0.

Figure 5:3 shows a section of the correlator in block form. Only one delay position is illustrated, but since all positions are identical and similar operations are simultaneously carried out for all positions, examination of one constitutes a survey of the whole system. We recall again that the basic arithmetic memory cell introduced in Chapter 2 is essentially a serial adding unit. Since it is a serial unit, words to be added must be presented in a serial form. Register R, containing the multiplicand of the rs product, is therefore a 17 bit ring counter with parallel load capabilities in the 8 least significant bits. Sixteen bits of this shift register are used to deliver the

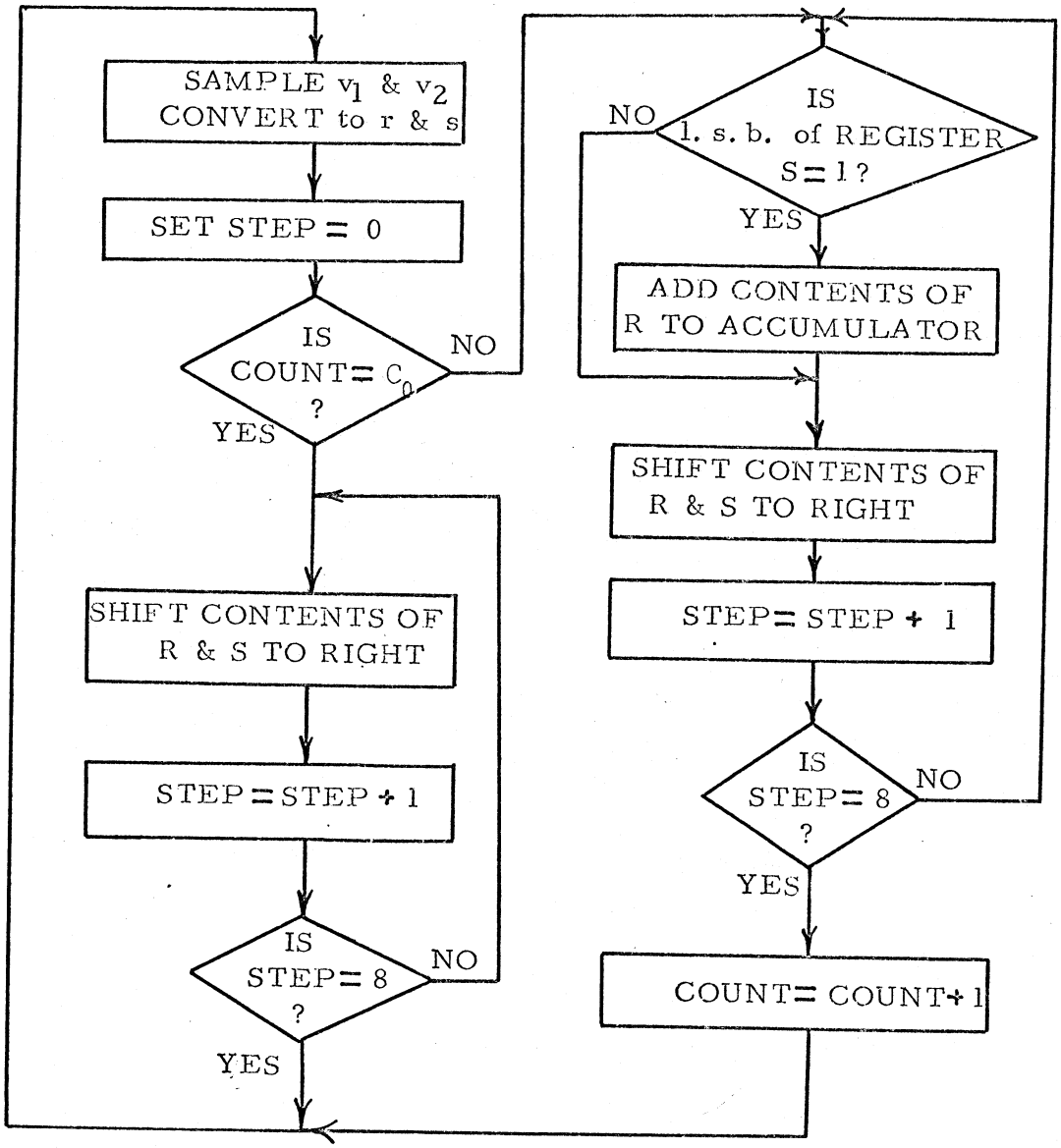


FIGURE 5.2 FLOW CHART FOR CORRELATION

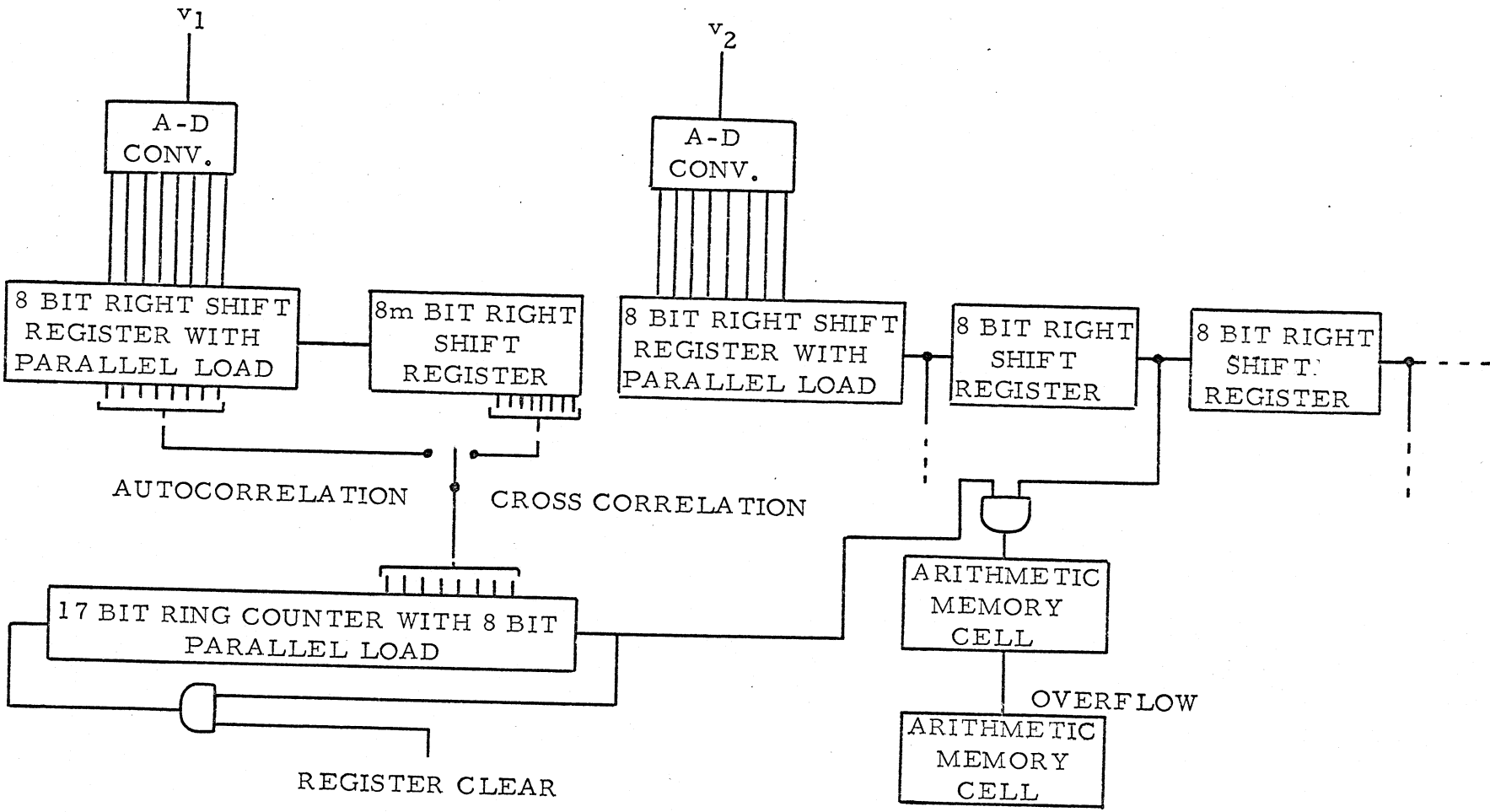


FIGURE 5.3 CORRELATION SECTION BLOCK DIAGRAM

multiplicand to the memory cell while the seventeenth bit is used to effect a one bit left shift of the number after each addition. The ring counter is pulsed 16 times corresponding to the 16 bit addition, and at the sixteenth pulse, the S register is pulsed once to address the next m.s.b. of the multiplier s. A second addition of the left shifted multiplier is attempted depending on the contents of this bit of the multiplier. The process is repeated until the entire multiplication has been completed. On the eighth and final cycle of the ring counter the ring is broken at the AND gate so that the register is automatically cleared in preparation for the parallel entry of the next r sample. The counter may be loaded from two sources; either directly from the A-D converter for autocorrelation, or from the end of a 8m bit shift register for cross-correlation. This shift register is clocked at the same time as the S register so that respective r and S time lags are kept in their proper perspective. The S shift register itself is simply a (2m+1)x8 bit shift register tapped at every 8 bits and possessing parallel load capabilities in the first 8 bits. It is the information contained in this register that determines whether or not the word in the ring counter is to be added to the sum in the memory cell. This is very simply accomplished through a AND gate at the input to the cell such that when the appropriate bit of the multiplier contains a 1 then the multiplicand is added and when a 0 appears no addition takes place. Since

a single 8 bit multiplication may require up to 15 bits to store the product, one 16 bit memory cell could become overloaded after very few samples are processed. To expand the capacity, a second or even a third cell may be cascaded to the first cell to count the overflow. This joining of several cells to increase the word length will be discussed in a later section.

At the end of the measuring run when all the sample products have been collected it is required to divide all these stored sums by $C_0 n^2$ where n is the number of quantization levels and C_0 is the total number of sample pairs taken. For 8 bit binary representation there are 256 quantization levels and a division by 256^2 is simply a shifting of the number 16 bit positions toward the l.s.b. Alternatively, the n^2 division can be made to occur simultaneously with sample processing if only the overflow from the first 16 bit memory cells are considered. The number stored in the second cells will then be the accumulated products already divided by n^2. Since the number of samples required for a given run may vary between problems, sufficient flexibility must be built in so that division by C_0 can be easily accomplished. If C_0 is made an even power of 10 and readout is in either decimal or BCD form then division by C_0 is simply a shift of the decimal point. For example if 10^6 sample pairs are collected then the decimal point must be shifted 6 positions to the left.

5.2 Mean Value Section

Measurement of mean value, as discussed in Chapter 4, is simply a matter of feeding a weighting number representing the sample value to an accumulator at each sampling instant. This accumulated total is then divided by a constant $C_0 n$ where C_0 is the number of samples collected and n is the number of quantization levels. Division by C_0 is a simple shifting of the decimal point, while for 8 bit binary representation, division by n is a matter of shifting the binary number 8 bits toward the l.s.b. As in the case of the sample product accumulation discussed in the previous section a more convenient method of this division is the counting of the overflow from an 8 bit accumulator. If successive approximation or cyclic analog to digital converters are used then a very simple 8 bit accumulator can be constructed. The device depends on the number being entered, serially, m.s.b first, as pulses from the appropriate bit position. If a particular bit position contains a 1 then a pulse is fed to the corresponding input to the accumulator. If the bit is a 0 then no pulse is delivered. Figure 5:4 shows the circuit diagram of the accumulator. The device ACTIVATE line must be held in the logical 1 position for any new information to be added. The overflow from the accumulator is then fed to an arithmetic memory cell for storage. When displayed the number in this cell must be divided by C_0 to

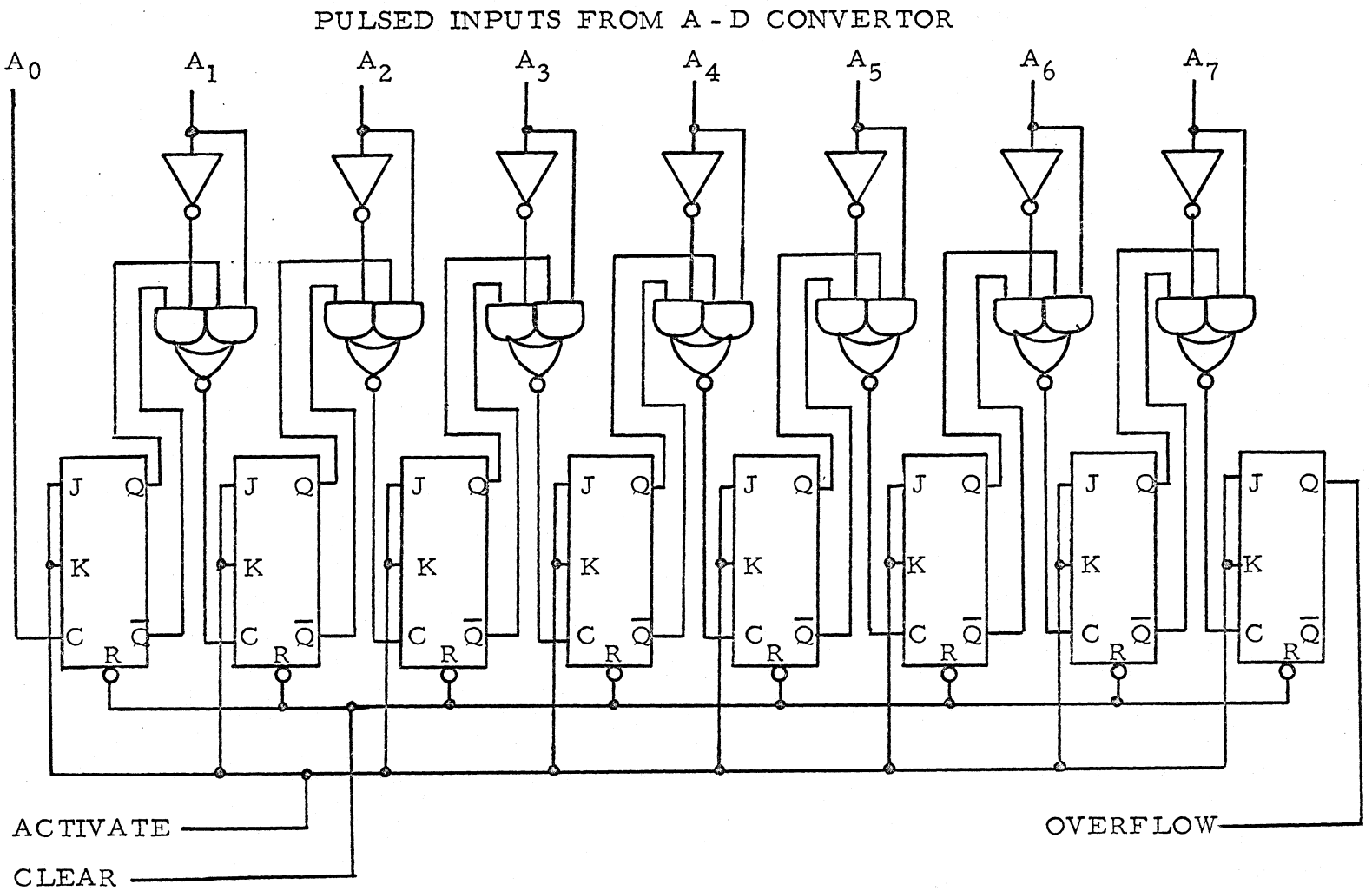


FIGURE 5.4 MEAN VALUE ACCUMULATOR

yield the mean value.

Identical circuits are used for measurement of mean value of the v_1 waveform and the v_2 waveform. In the case of autocorrelation, only one would be required; however, two are included for a general correlator.

5.3 Removal of D.C. Bias

As discussed in section 4 of Chapter 4 the numbers stored in both the mean value and correlation section of the instrument will be in error due to the effect of the half range d.c. shift imposed on the input waveforms. Equation 4.11 is restated here for convenience.

$$R'_{v_1v_2}(\tau) = R'_0(\tau) - W(\bar{v}_1 + \bar{v}_2) - W^2 \quad .....4.11$$

where $R'_{v_1v_2}(\tau)$ is the correlation function of two voltages v_1 and v_2; $R'_0(\tau)$ is the function as measured by the correlator; $\bar{v}_1$ and $\bar{v}_2$ are the mean values of v_1 and v_2 respectively; and W is the d.c. bias voltage superimposed on the input waveforms. Consider again the derivation of this relation. The v_1 and v_2 input voltages are quantised and converted to binary representations r and s respectively. Because of the bias voltage W, however, the outputs of the A-D converters is correspondingly modified to r+B and s+B respectively, where B is the binary representation of W. After suitable scaling to account for the number of quantisation levels, n, the number fed to the storage unit is

$$S = \frac{1}{n^2}(r+B)(s+B)$$

$$= \frac{rs}{n^2} + \frac{B(r+s)}{n^2} + \frac{B^2}{n^2} \quad .....5.1$$

Equation 5.1 expresses the contribution of a single sample pair to the accumulated sum of all sample pairs. For a d.c. bias of exactly half the range of the instrument $B = \frac{n}{2}$ and the equation may be rewritten

$$S = \frac{rs}{n^2} + \frac{1(r+s)}{2n} + \frac{1}{4} \quad .....5.2$$

Applying a similar approach to the mean value section of the instrument the output of the mean value accumulators E_1 and E_2 may be expressed as

$$E_1 = \frac{r+B}{n}$$

$$= \frac{r}{n} + \frac{1}{2} \quad .....5.3(a)$$

and

$$E_2 = \frac{s}{n} + \frac{1}{2} \quad .....5.3(b)$$

If the outputs of these accumulators are both fed to a common counter then the input to this counter will be the sum of E_1 and E_2

$$E = E_1 + E_2$$

$$= \frac{1}{n}(r+s) + 1$$

The insertion of an additional flip flop at the input to the counter may be made to effect a division by 2. The contribution of each sample pair to the count will then be

$$P = \frac{1}{2n}(r+s) + \frac{1}{2} \qquad .....5.4$$

A comparison between equations 5.2 and 5.4 shows that P is very close to the correction factor required to obtain the unbiassed correlation function. If the counter is initially preset to $-\frac{1}{4}$ using two's complement notation then at the end of the measuring run, the exact correction factor will be available. It remains only to subtract this correction factor from each of the correlation accumulators to obtain the correlation function.

It should be recognized here that the previous analysis represents an approximation only. Equation 5.2 considers samples r and s that are delayed through a shift register. The samples used in equation 5.4, however, are taken directly from the A-D converters, not delayed, and are not therefore necessarily equal to those in equation 5.2. In a long measuring run where many thousands of samples are taken, the effects of a change in input voltages over the time of the delay will be negligible and the previous analysis will be valid.

Figure 5.5 illustrates the block diagram for the calculation of the correction factor. The arithmetic memory

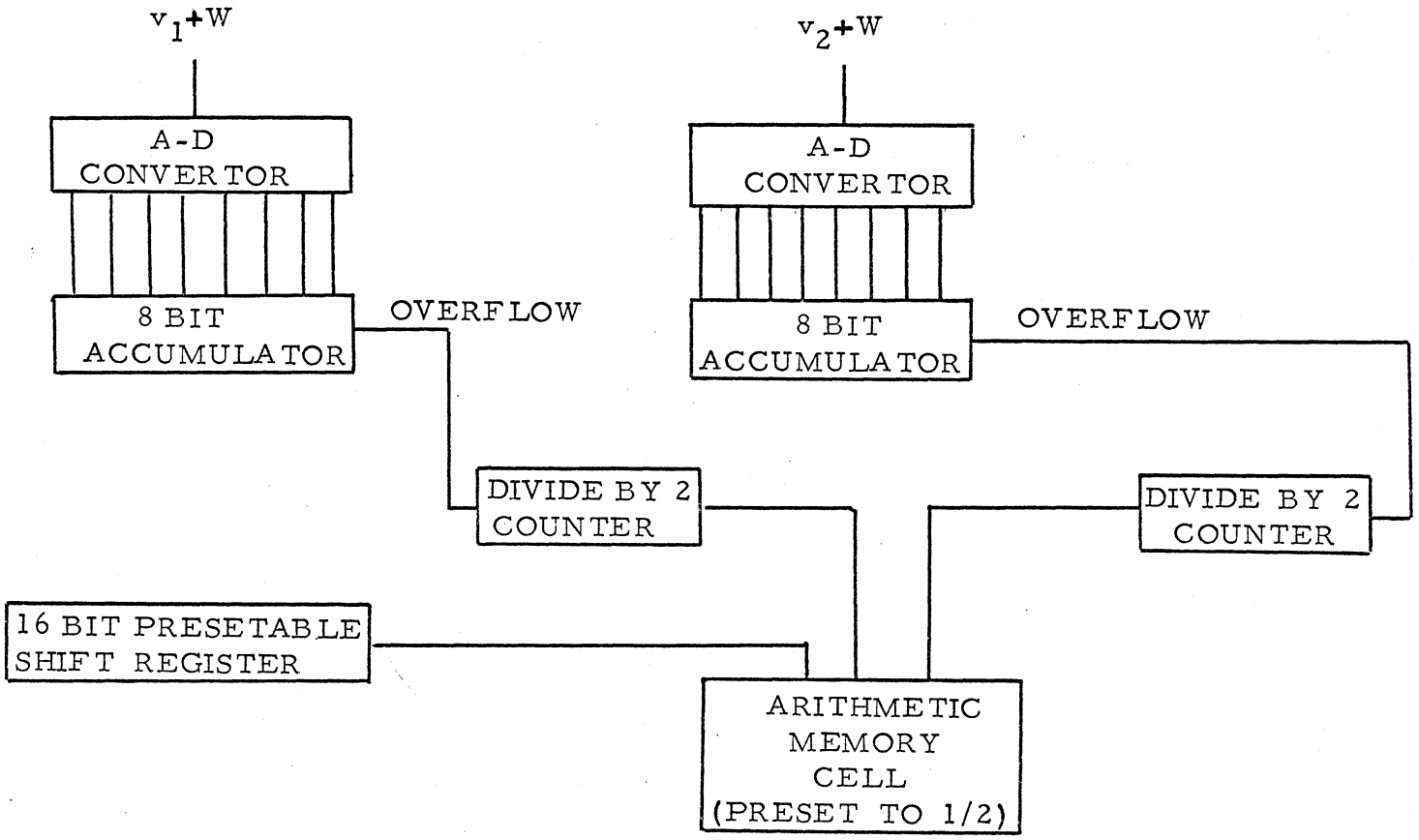


FIGURE 5.5 BLOCK DIAGRAM FOR STORAGE OF CORRECTION FACTOR

cell acts as the counter and is preset prior to the start of the measuring run by serially loading the cell from a presetable 16 bit shift register. This shift register is preset to the two's complement representation of $\frac{1}{4}$ suitably scaled to the number of samples to be taken.

Removal of the W term in the mean value memory cells is a simple matter of presetting these cells to $-\frac{1}{2}$ at the same time as the correction factor is preset. At the end of the measuring run the number contained in these cells will be the exact time average of the input waveforms.

5.4 Complete System

When the entire correlator is examined as a system, it becomes evident that certain devices can be shared by different sections of the instrument. For example, the A-D converter represents a major expense in any system and it would therefore be economical to utilize a single converter for both v_1 and v_2 inputs. A sample and hold circuit would then be used in one of the input lines to permit simultaneous sampling of both inputs. While v_1 is being sampled and quantised by the converter, v_2 is sampled and stored in the sample and hold circuit until the A-D converter is again free. Such sample and hold devices are considerably less complex and more economical than quantisers and in a system where speed of conversion is not important they represent significant

savings. An electronic switch driven by the control logic would choose which of the two inputs is to be converted.

Figure 5:6 shows the complete correlator in block form. The system blocks will not be described in any greater detail than has already been done. Any further details such as specification of components or construction techniques are beyond the scope of this thesis. Suffice it to say that all components required for a satisfactory implementation of the design are currently available in integrated circuit form. The principle of digital measurement of correlation functions has been proven by Kitai and Masuko[10] and the design proposed here may be considered an extension of their instrument.

5.5 Cascading of Memory Cells

In cases where a very large number of sample pairs are to be processed, it may be found that some of the accumulated products exceed the storage capacity of a single 16 bit memory cell. In such cases a second cell can be used to count the overflow from the first. Recall again the carry delay incorporated in the typical arithmetic memory cell of Figure 3:2. The carry between adjacent bit positions was stored in a presetable J-K flip flop. Any carry from the m.s.b. was destroyed with the data latch pulse so that it would not effect the l.s.b. during following additions. If this carry, before being cleared, is used to preset the carry flip flop of a second memory cell then this second cell will

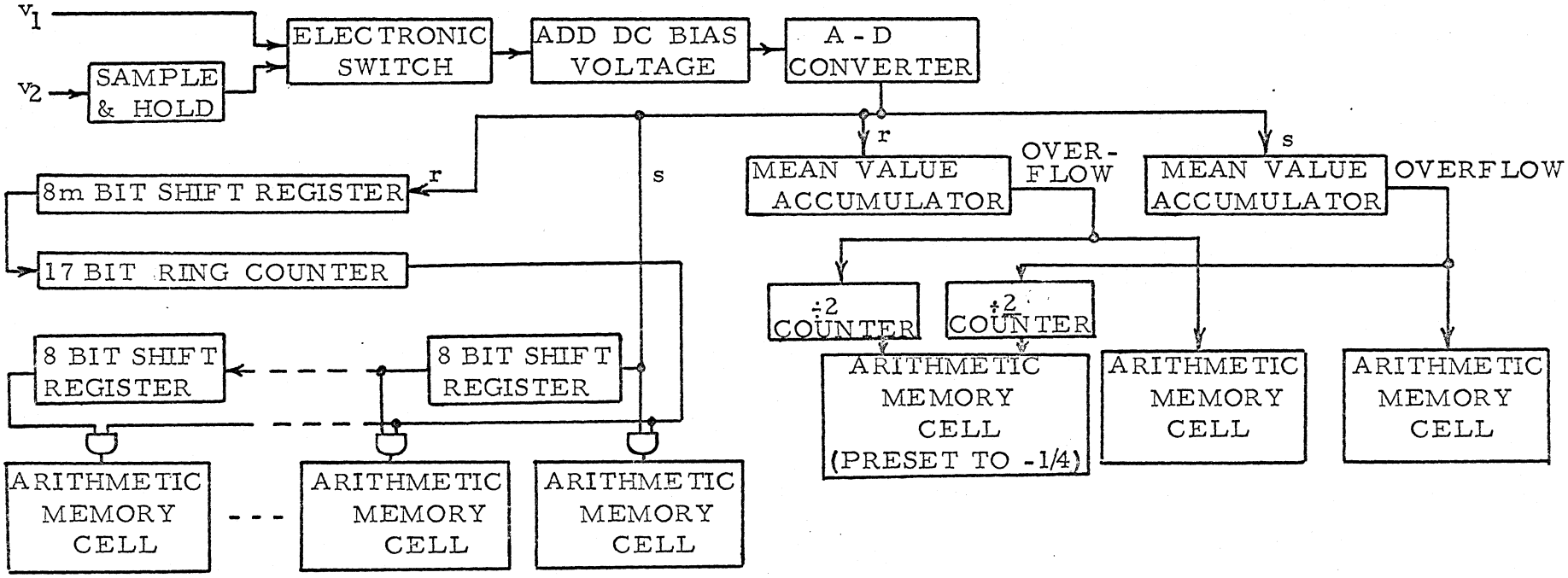


FIGURE 5.6 COMPLETE CORRELATOR BLOCK DIAGRAM

count the overflow from the first. A very simple logic circuit can be used to recognize the l.s.b. position and ensure that this carry between cells occurs at the proper time. The reading of information from cascaded cells is handled in a similar fashion to a single cell. Information from the first is fed out serially followed immediately by the second resulting in 32 bits of storage. Any number of cells can be cascaded in this manner without increasing the processing time beyond that for a single cell.

5.6 Summary

The digital correlator is intended primarily for low frequency measurement where sampling intervals need not necessarily be small. The upper limit on sampling frequency is governed by the time required for the multiplication process, or in the case of the instrument proposed, the time for eight cycles of the arithmetic memory cells. At a clock frequency of 2 MHz, a single cycle of the memory cell would require 16 microseconds and a sampling period of 128 microseconds could be achieved. The design is such that the increment of time delay between points on the correlation graph is equal to the sampling period. There is, of course, no low frequency limit to the range of input waveforms to be measured and the upper frequency limit is governed by the minimum increment of time desired in the plotting of the correlation graph.

The feasibility of the design from a commercial

standpoint hinges on the availability of arithmetic memory cells in a low cost, compact form. The remainder of the system consists of integrated circuits presently on the market and no difficulty would be anticipated in the construction of this section of the instrument.

CHAPTER 6

CONCLUSION

The computing memory described in this thesis is intended only to be a guide for future development of arithmetic memory cells and not an ultimate design in its present form. The memory was built using currently available integrated circuits and hence is not particularly compact or fast. A clock frequency of only 2 MHz was achieved, permitting a one microsecond per bit addition. This was taken to be satisfactory since the purpose of the design was merely to introduce the active memory and prove its feasibility. It is to be expected that if the arithmetic memory cell is to gain acceptance, then new circuits will be designed which are suitable for large scale integration and higher speed. Once such devices are available it must necessarily follow that many more applications than those suggested here would be explored. The problems of correlation and field plotting are only two applications arbitrarily chosen because of their general familiarity. Many other possibilities exist such as Fourier analysis and Walsh-Fourier analysis[15] where a large number of counters are required to operate simultaneously.

Many areas for improvement in the basic design

naturally exist and although these have been considered, restrictions on expense and availability have forced the compromise discussed here. One rather obvious shortcoming of the cell is the need to provide two write circuits for the memory, one for each of the two possible binary states. A second weakness lies in the need for two clock pulses for each bit of addition, one to latch the data in memory and a second to write new data into storage. It would be preferable to use a memory circuit with a single data input line and a separate write enable input that would act like the clock input of a J-K flip flop. In such a case it would be possible to use the information already in a memory location to write new data in using only one clock pulse. Such a memory is available in the form of the Fairchild type MμL 9035 integrated circuit. This device incorporates four 16-bit words in a single 36 pin package and could form the basis for a group of four basic arithmetic memory cells. One such cell is illustrated in Figure 6:1. Since the write enable line acts as a clock input there is no need for the D-type flip flop to latch the data in the memory. Combined with the single data-in line the saving in circuitry is very considerable. Because of the restrictions on cost and availability, however, it was not possible to encorporate this device into the memory, but its potentials are apparent.

One other question is raised in the investigation of the correlator design: In many cases, the arithmetic

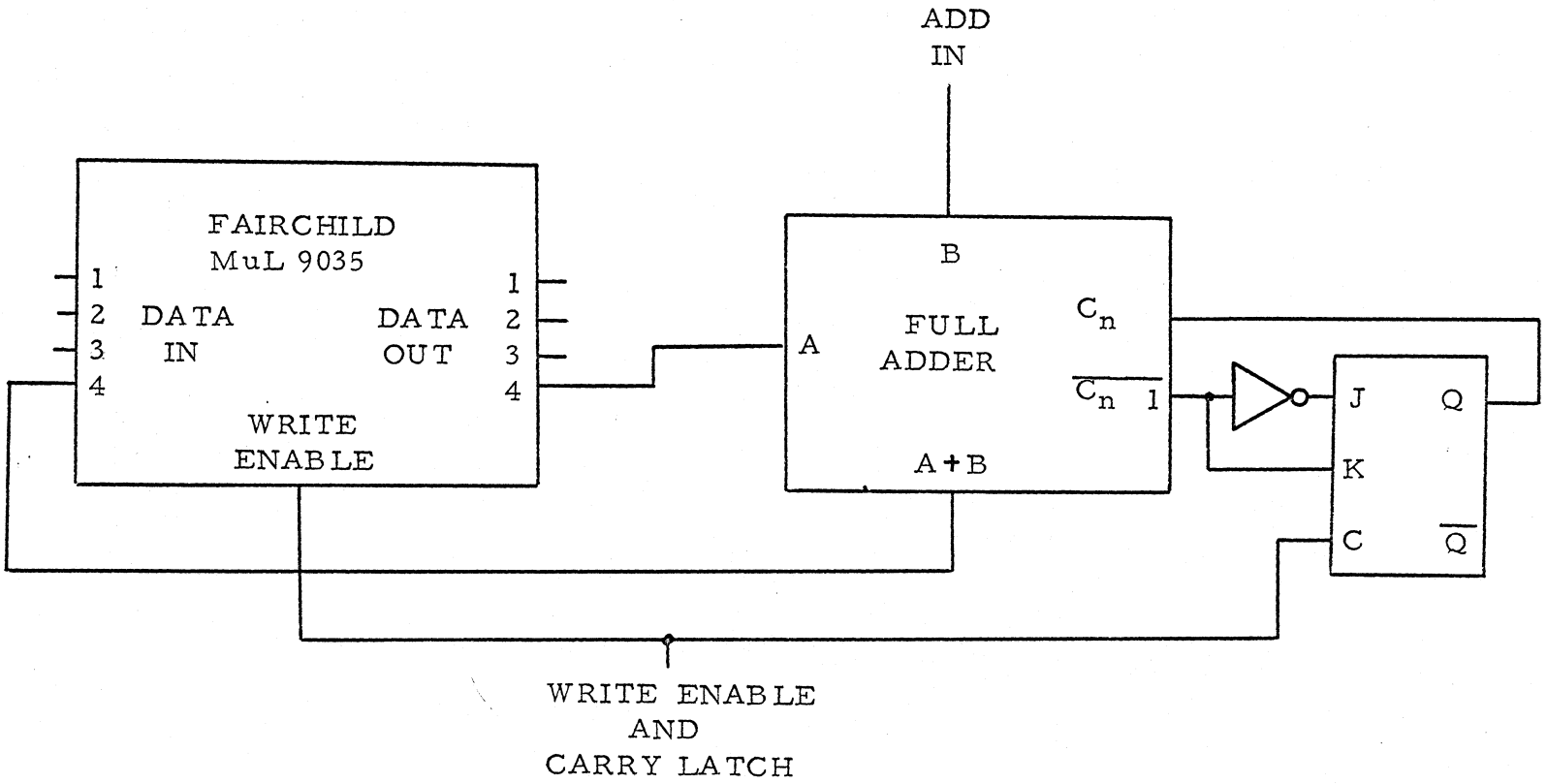


FIGURE 6.1 SIMPLEFIED ARITHMETIC MEMORY CELL

memory cells are not required to add as such, but rather to count uniformly time-quantized overflows from a second memory cell. In such a case it would appear that a considerable portion of the memory cell is unnecessary and perhaps a new device, the counting memory cell, should be investigated. In the interest of simplicity such a design was not included in the correlator proposal; however, it does warrant consideration. A possible design for a counting cell is shown in Figure 6:2 using the Fairchild 9035 memory. When the l.s.b. of the counter is addressed a count in will preset the J-K flip flop to a logical 1. This voltage level will then complement the data in this bit position of the memory through the exclusive-or gate and apply the new level to the data input of the memory. If the flip flop is left in the 0 state then the data will remain unaltered. If the flip flop is at the 1 level and a 1 state is encountered in the memory, then the 2-input NAND gate and inverter will apply a 1 and a 0 to the J and K inputs of the flip flop respectively and upon receipt of a clock pulse a 1 will be entered into the flip flop. If either of these conditions does not hold then the flip flop will contain a 0 following the clock pulse. At the same time as the flip flop is clocked, the memory write-enable line is pulsed and the next bit is addressed. The flip flop acts as a carry storage and the same analysis applies as for the l.s.b. The cycle continues through all 16 bits and at the end of the sixteenth bit the carry latch is reset to

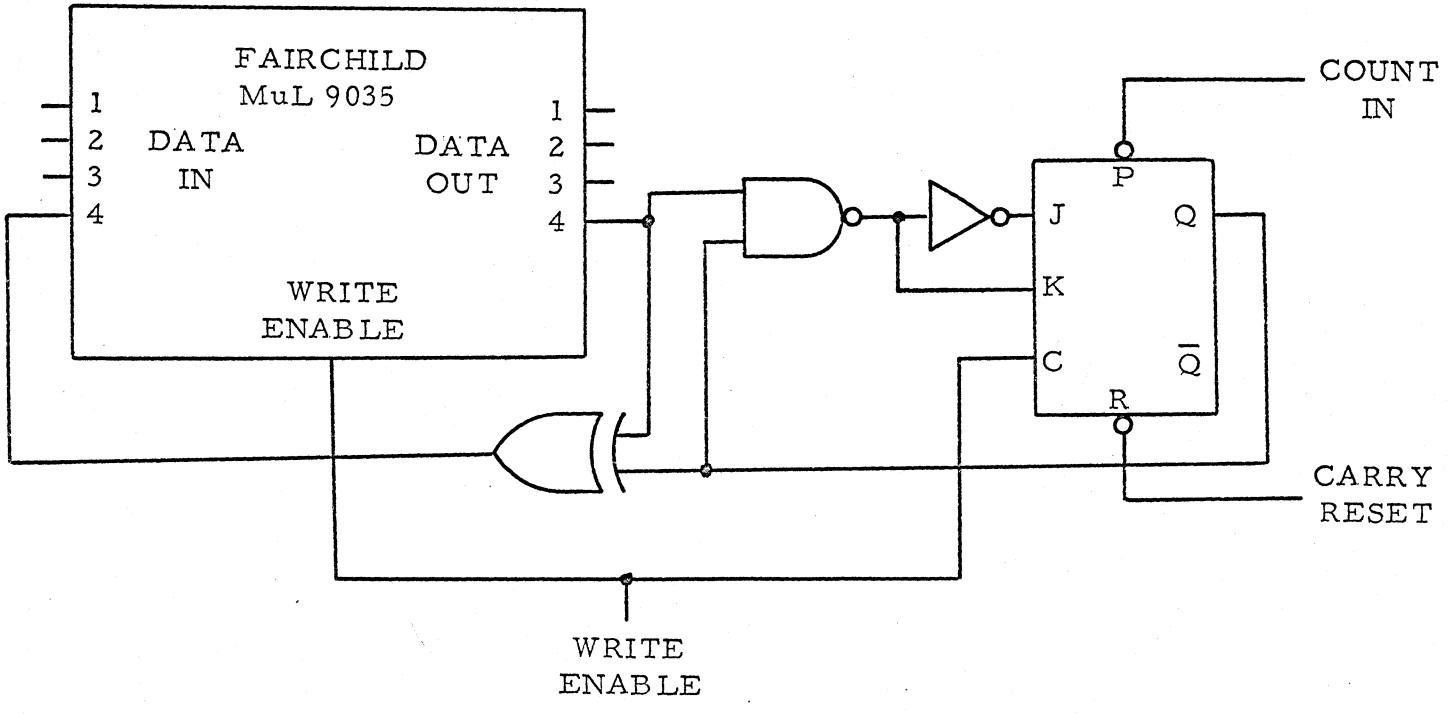


FIGURE 6.2 COUNTING MEMORY CELL

0 in preparation for a further count.

Counters of this type are restricted in that inputs may not be random, but must occur at fixed periods in time. In the case of the correlator, however, the overflow from the adders do occur at predictable times and all counters can be operated simultaneously.

It is not the purpose of this thesis to present the computing memory solely as a system to be associated with a general purpose computer; rather it is an attempt to introduce the concept of the arithmetic memory cells as a complete and independent unit in itself. It is through these qualities that the cell must gain its widest acceptance. As a special peripheral device to be used in conjunction with a general purpose computer, a memory array composed of these cells becomes a self-administering block, effectively freeing the sophisticated central processor of the menial task of repetitive but elementary arithmetic operations. When considered as an independent unit, however, the range of application of the cell broadens to encompass all areas of digital systems, and is not restricted to general purpose computers. It is the author's hope that this philosophy of universal applicability be the predominant impression left by this thesis so that the full potential of the cellular operators be realized.

APPENDIX A

Logic Symbology

Texas Instruments Integrated Circuits

Descrete Component Circuits

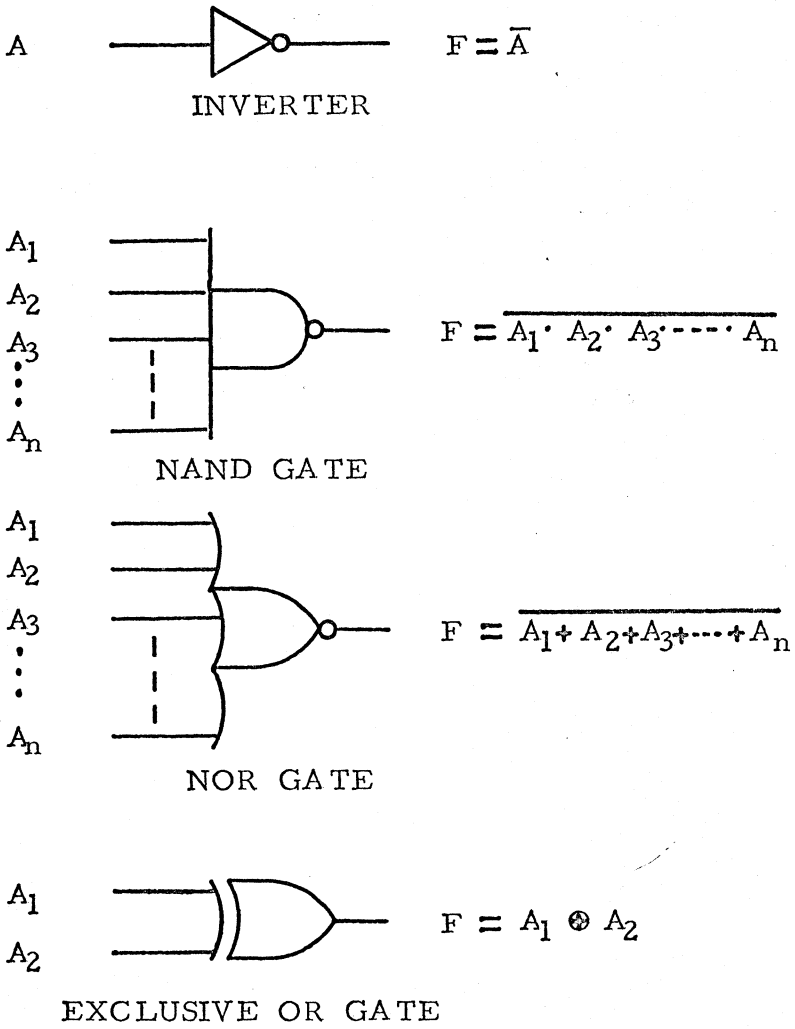
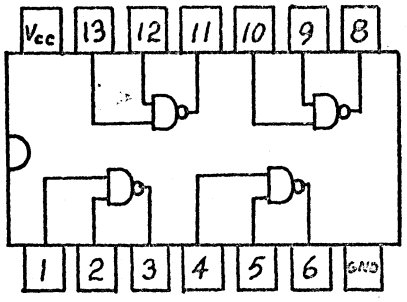


FIGURE A.1 LOGIC SYMBOLOGY

MIL 806B SPECIFICATIONS

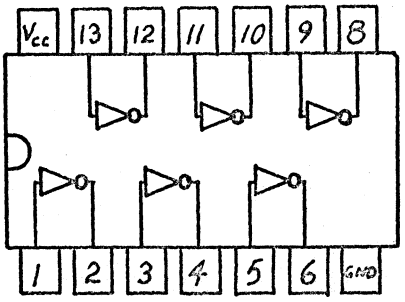
SN7400N

QUAD 2-INPUT NAND GATE



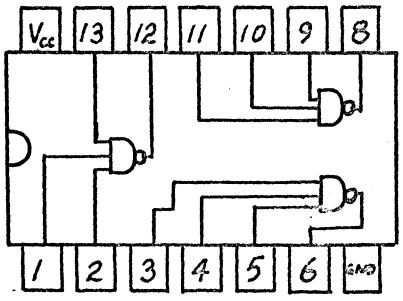
SN7404N

HEX INVERTER



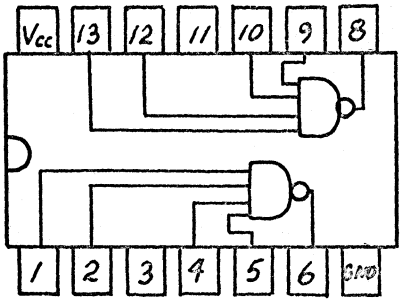
SN7410N

TRIPLE 3-INPUT NAND GATE



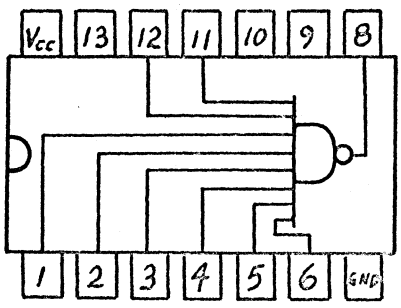
SN7420N

DUAL 4-INPUT NAND GATE



SN7430N

8-INPUT NAND GATE



SN7440N

DUAL 4-INPUT NAND BUFFER

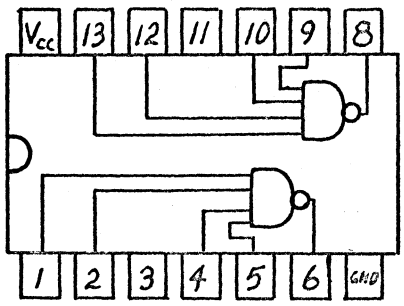
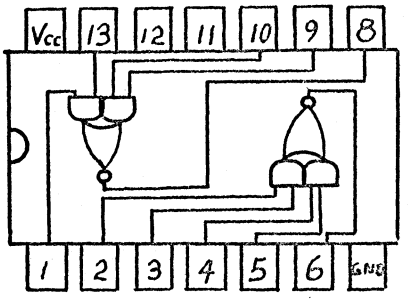


FIGURE A.2 TEXAS INSTRUMENTS INTEGRATED CIRCUITS

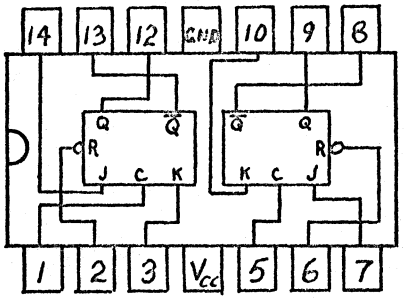
SN7451N

DUAL 2-WIDE 2-INPUT AND-OR-INVERT



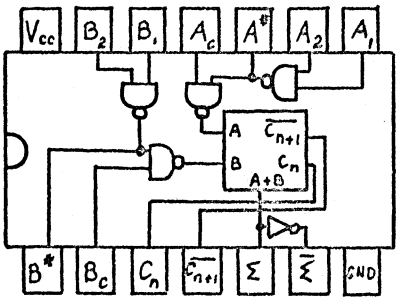
SN7473N

DUAL J-K MASTER-SLAVE FLIP-FLOP



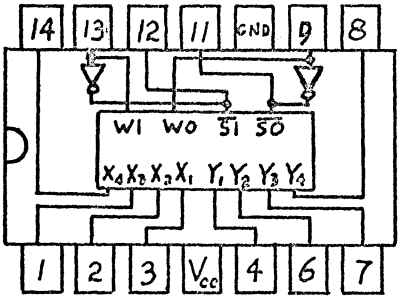
SN7480N

GATED FULL ADDER



SN7481N

16 BIT READ/WRITE MEMORY



SN7486N

QUAD 2-INPUT EXCLUSIVE-OR ELEMENT

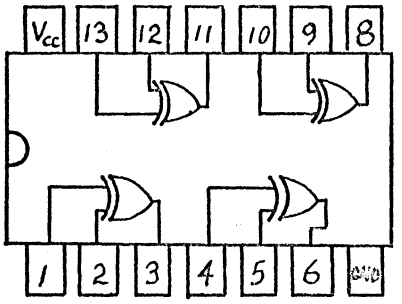
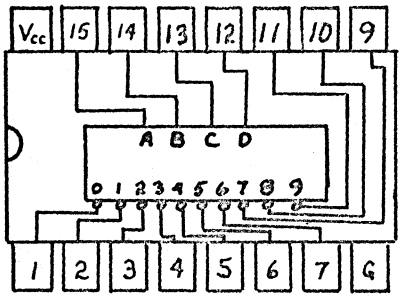


FIGURE A.2 TEXAS INSTRUMENTS INTEGRATED CIRCUITS

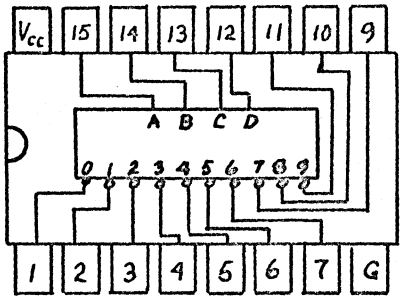
SN7442N

BCD TO DECIMAL DECODER



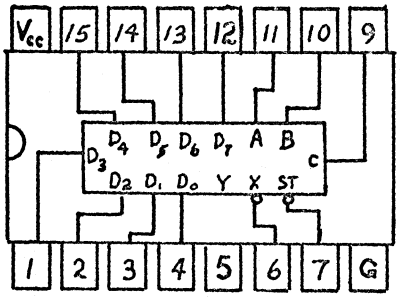
SN7445

BCD TO DECIMAL DECODER DRIVER
WITH OPEN COLLECTOR OUTPUTS



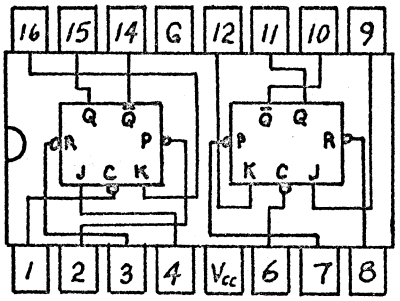
SN74151N

8 BIT DATA SELECTOR/MULTIPLEXER



SN7476N

DUAL J-K MASTER-SLAVE FLIP-FLOP
WITH PRESET AND CLEAR



SN7475N

QUADRUPLE BISTABLE LATCH

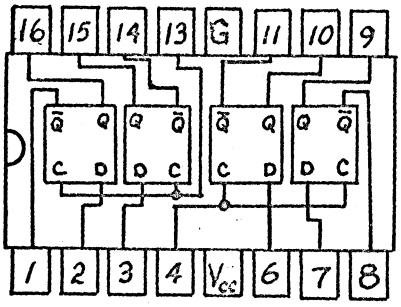
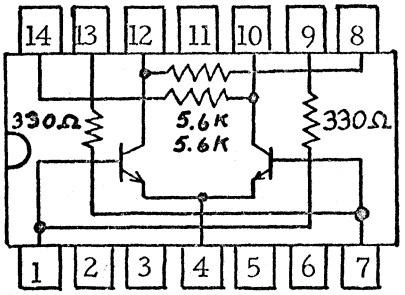
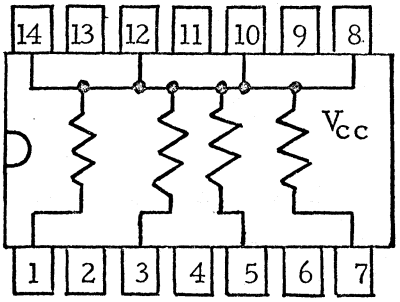


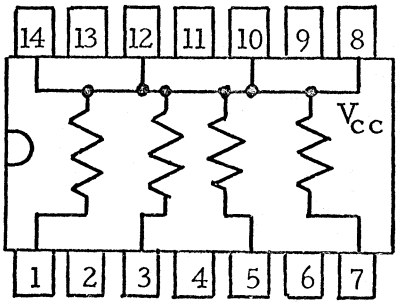
FIGURE A.2 TEXAS INSTRUMENTS INTEGRATED CIRCUITS



DP10
DUAL DRIVER PLATFORM
TRANSISTERS - 2N4227



DP20
QUAD 10 K RESISTER PLATFORM



DP30
QUAD 100Ω RESISTER PLATFORM

FIGURE A.3 DESCRETE COMPONENT PLATFORMS

APPENDIX B

This Appendix contains the wiring instructions for the 8 word memory array and the control circuit capable of driving up to 64 words of memory constructed for this thesis. The memory and control are mounted on separate circuit boards, Texas Instrument Part No. 10-000-PS, and are shown in Figures B.1 and B.13 respectively.

Each integrated circuit package location is designated by a two digit number representing the row and column number. For example in Figure B.1 package 46 is located in row 4 column 6 and is a SN7480N gated full adder. In the wiring diagrams that follow the location of each device is shown as a circled 2 digit number appearing inside the device representation. The numbers outside the device correspond to the pin number of the particular package, as shown in Appendix A.

Connections to the edge connector pins of the circuit boards are shown in the wiring diagrams as numbers enclosed in small squares. The codings of the output pin connections for the memory board and control board are shown in Table B.1 and B.2 respectively.

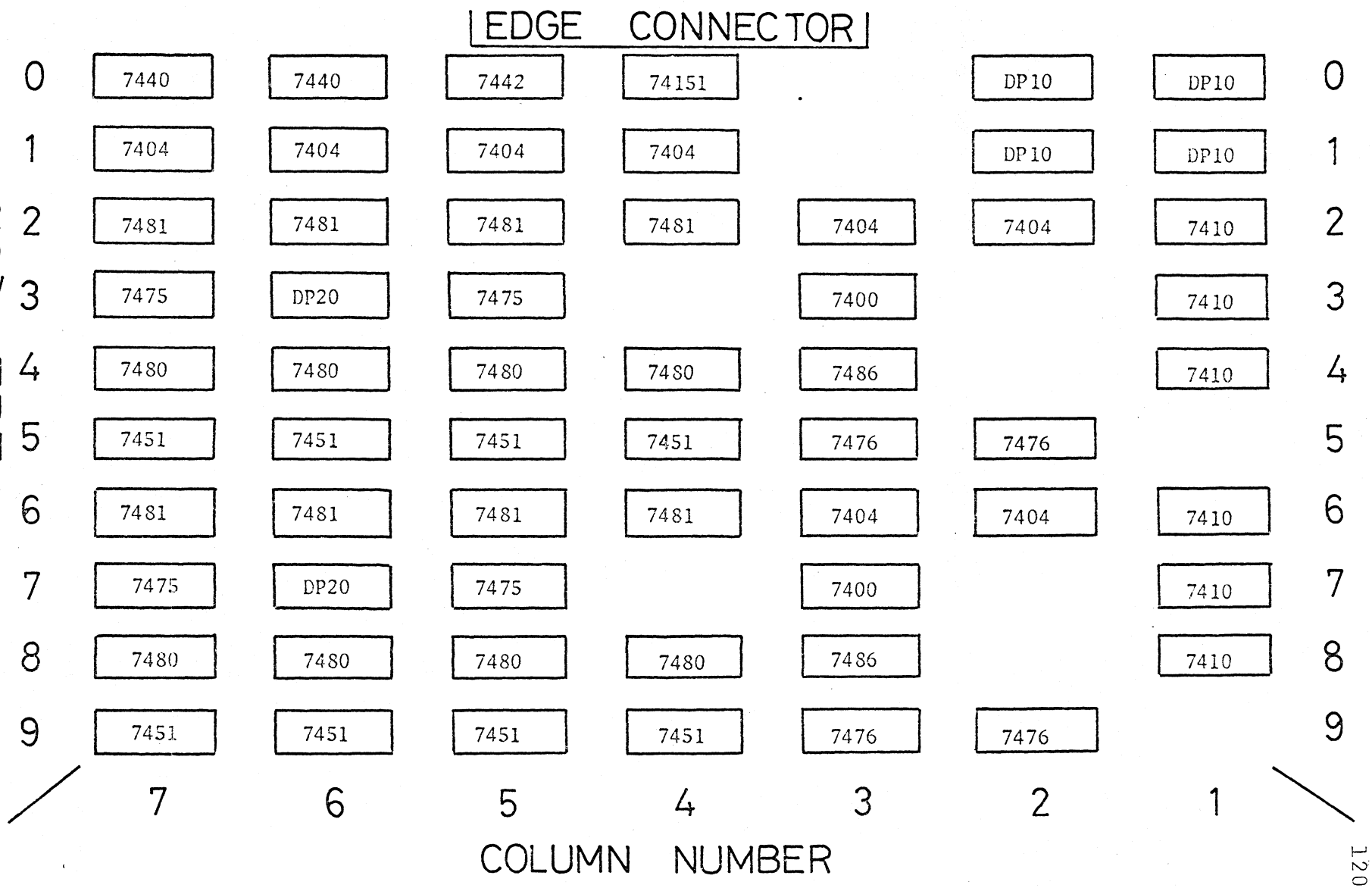


FIGURE B.1 MEMORY BOARD PACKAGE LOCATION CHART

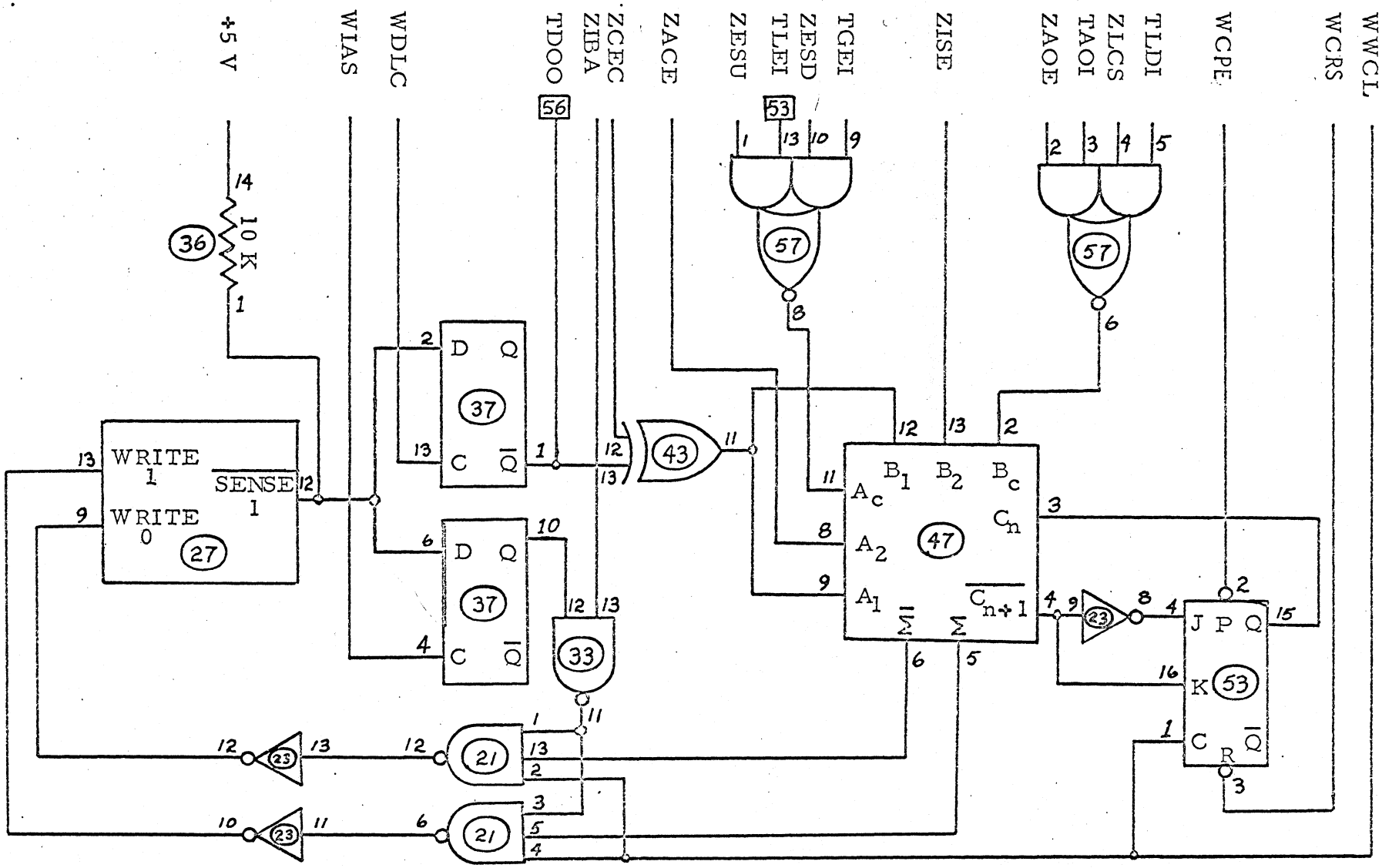


FIGURE B.2 MEMORY BOARD WIRING DIAGRAM: CELL P0-E

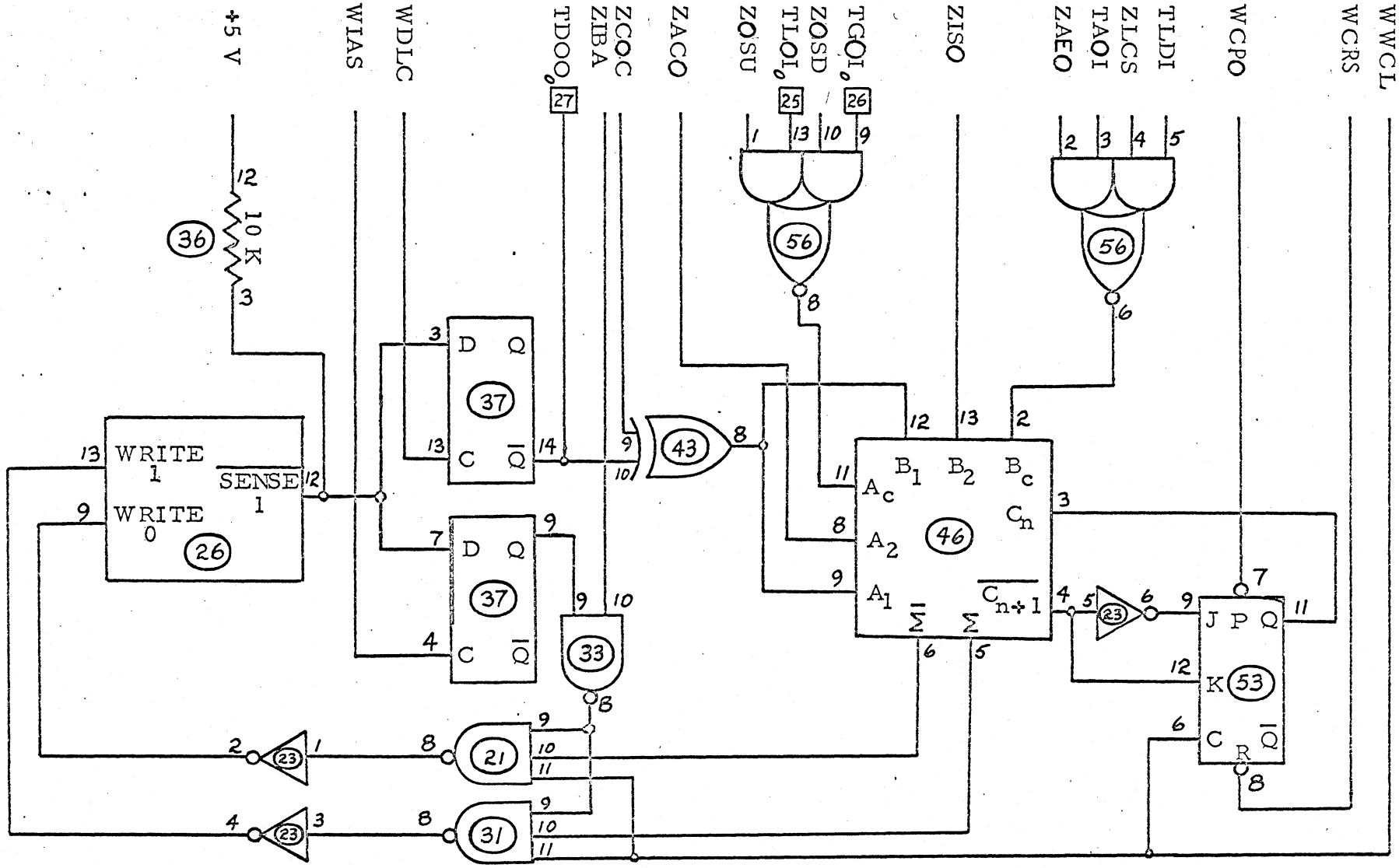


FIGURE B.3 MEMORY BOARD WIRING DIAGRAM: CELL P0-0

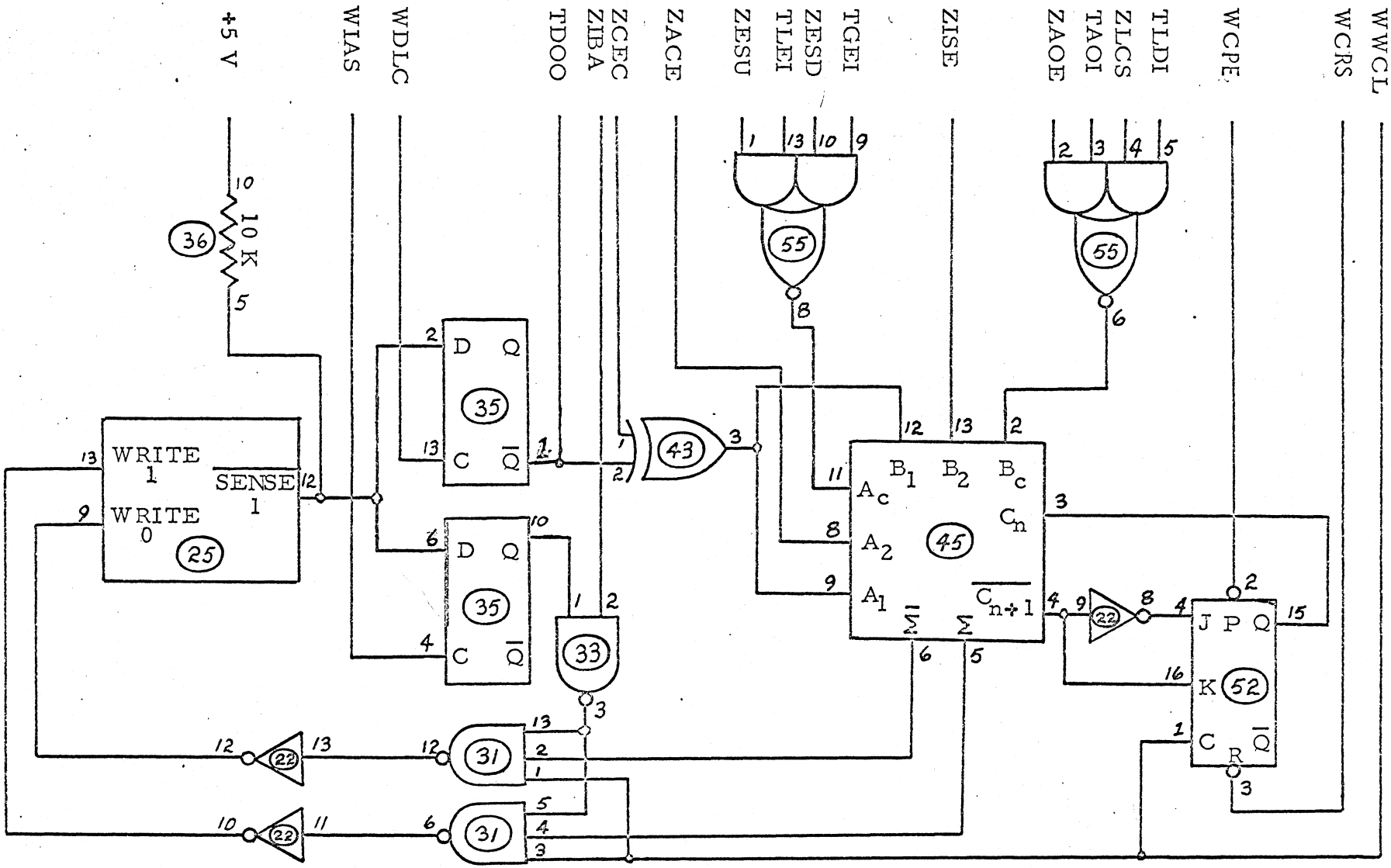


FIGURE B.4 MEMORY BOARD WIRING DIAGRAM: CELL P1-E

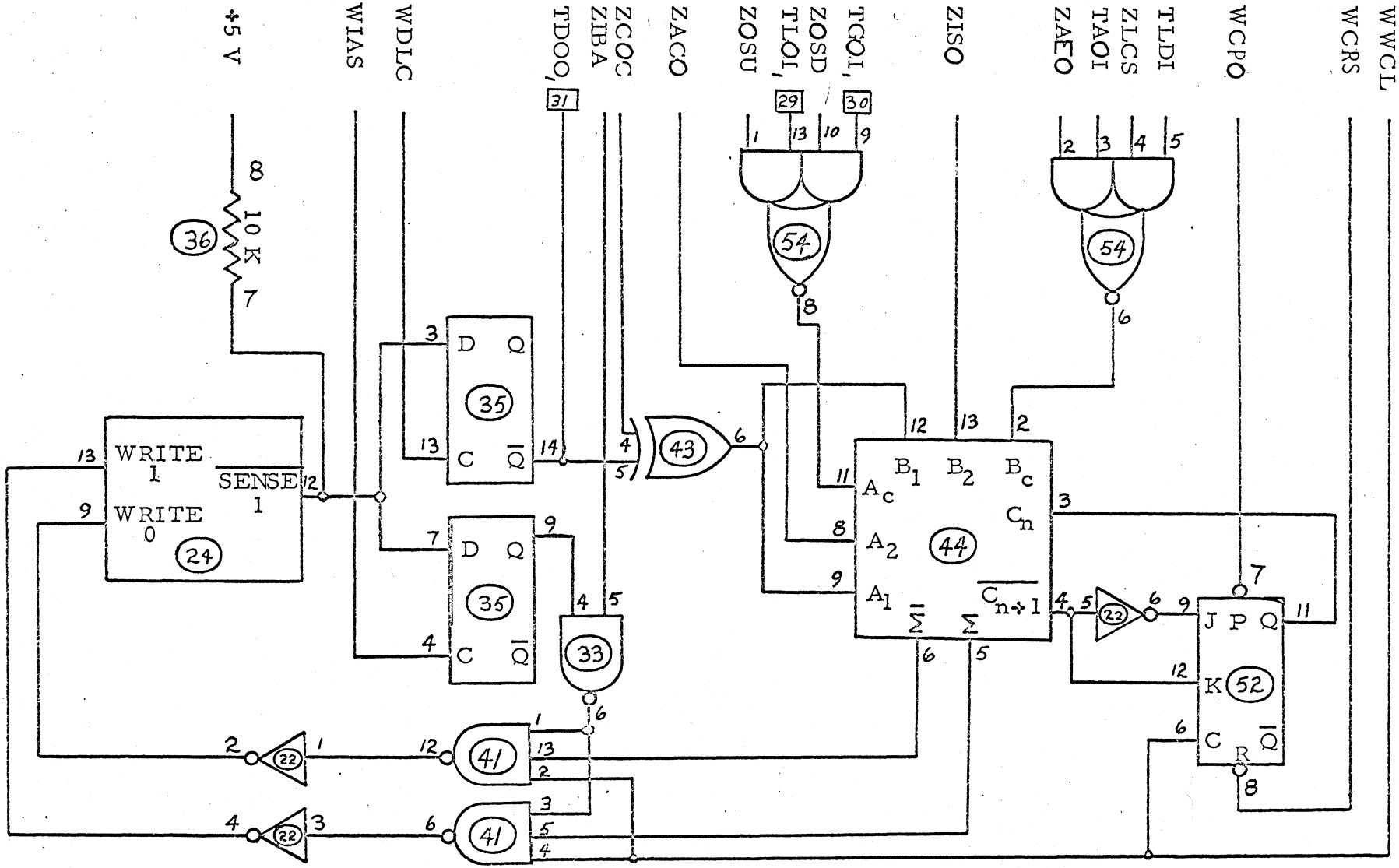


FIGURE B.5 MEMORY BOARD WIRING DIAGRAM: CELL P1-0

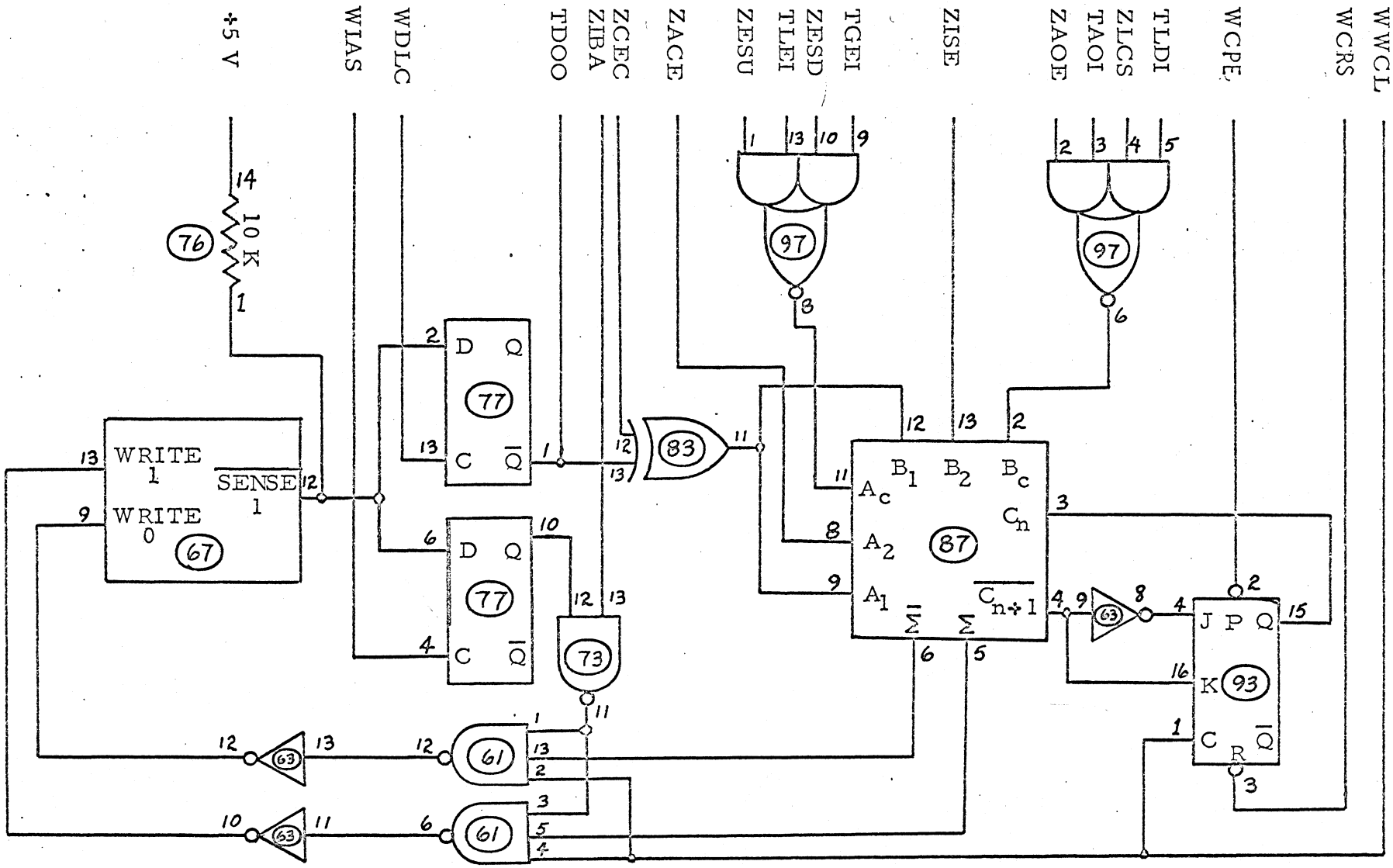


FIGURE B.6 MEMORY BOARD WIRING DIAGRAM: CELL P2-E

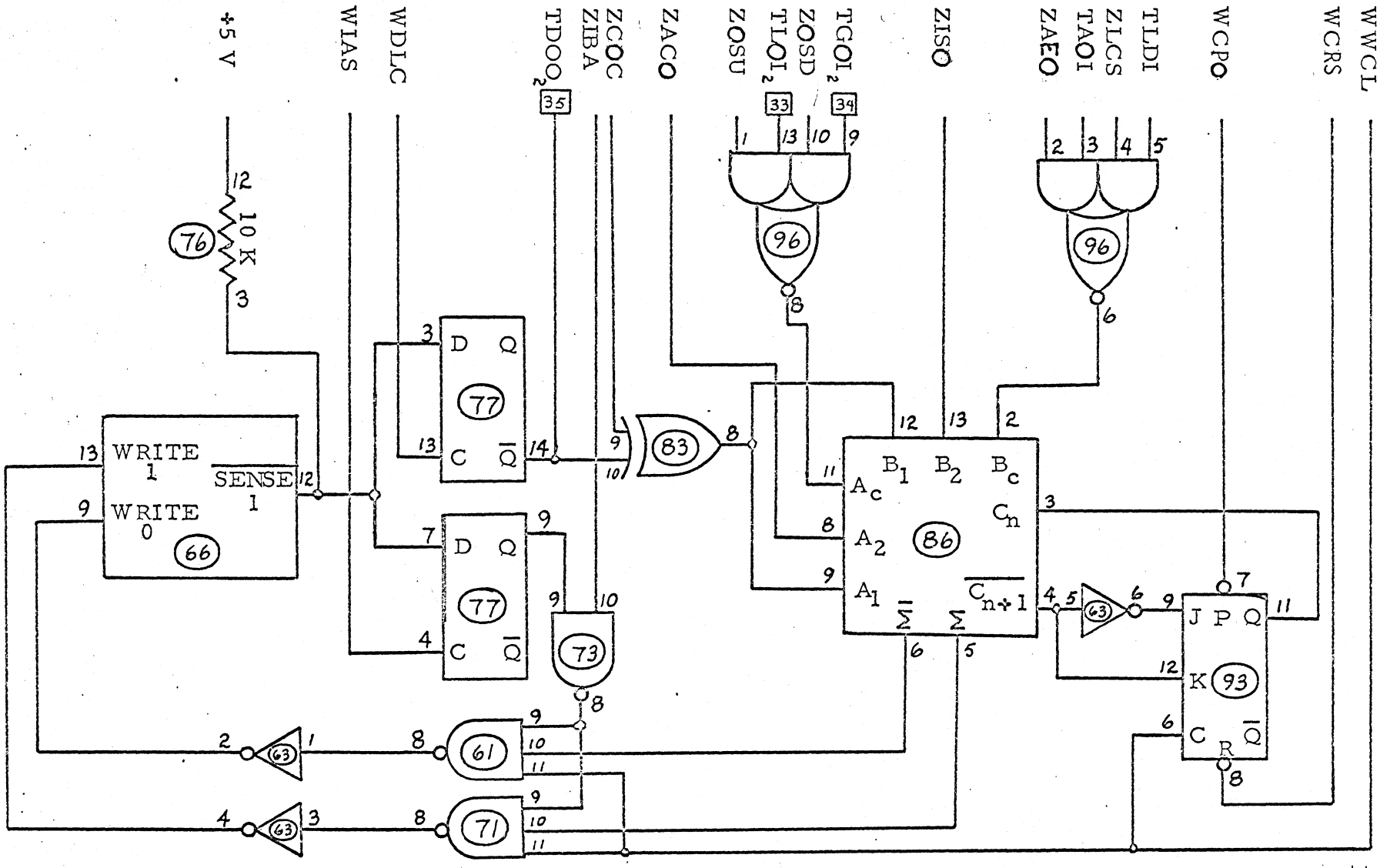


FIGURE B.7 MEMORY BOARD WIRING DIAGRAM: CELL P2-0

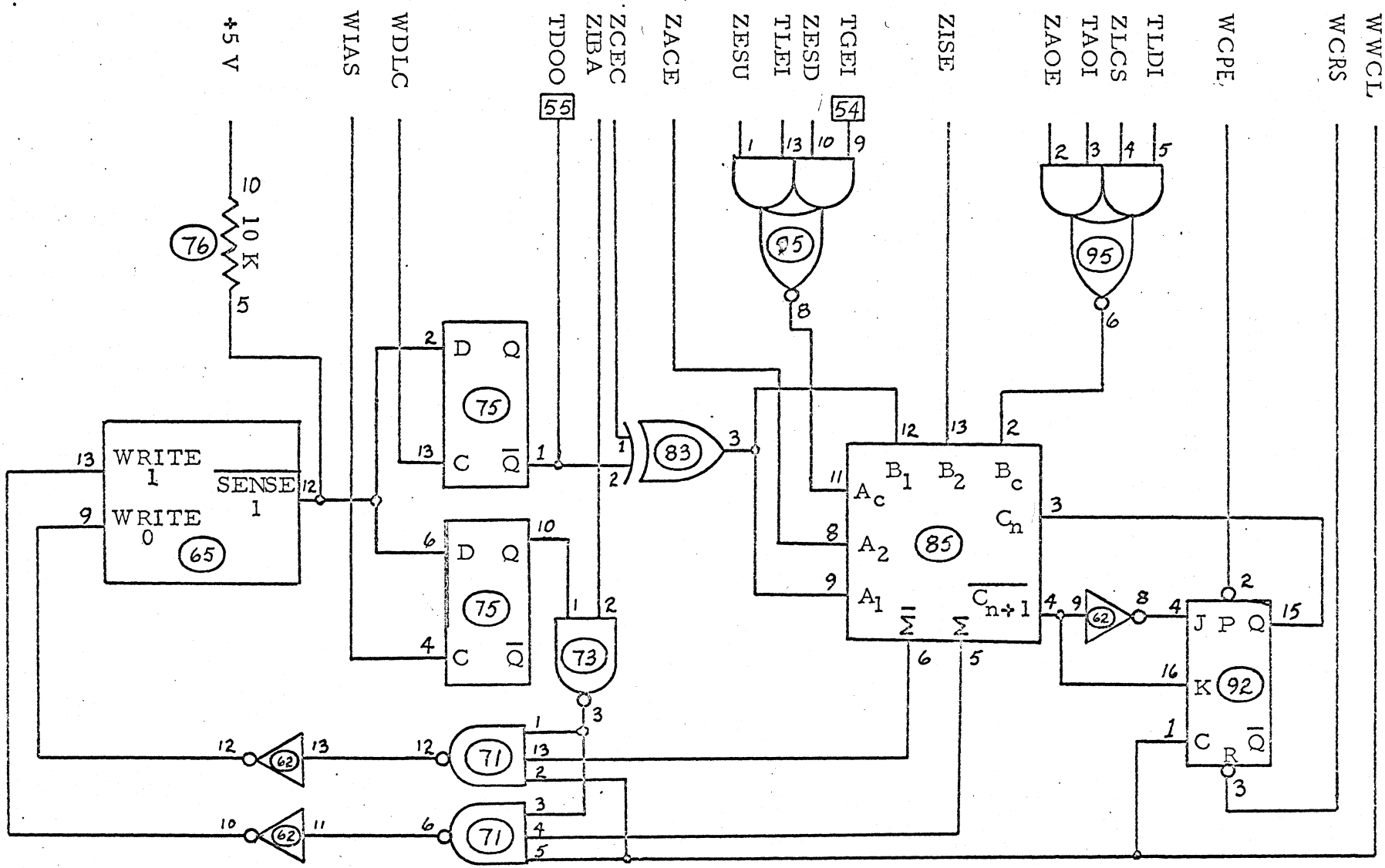


FIGURE B.8 MEMORY BOARD WIRING DIAGRAM: CELL P3-E

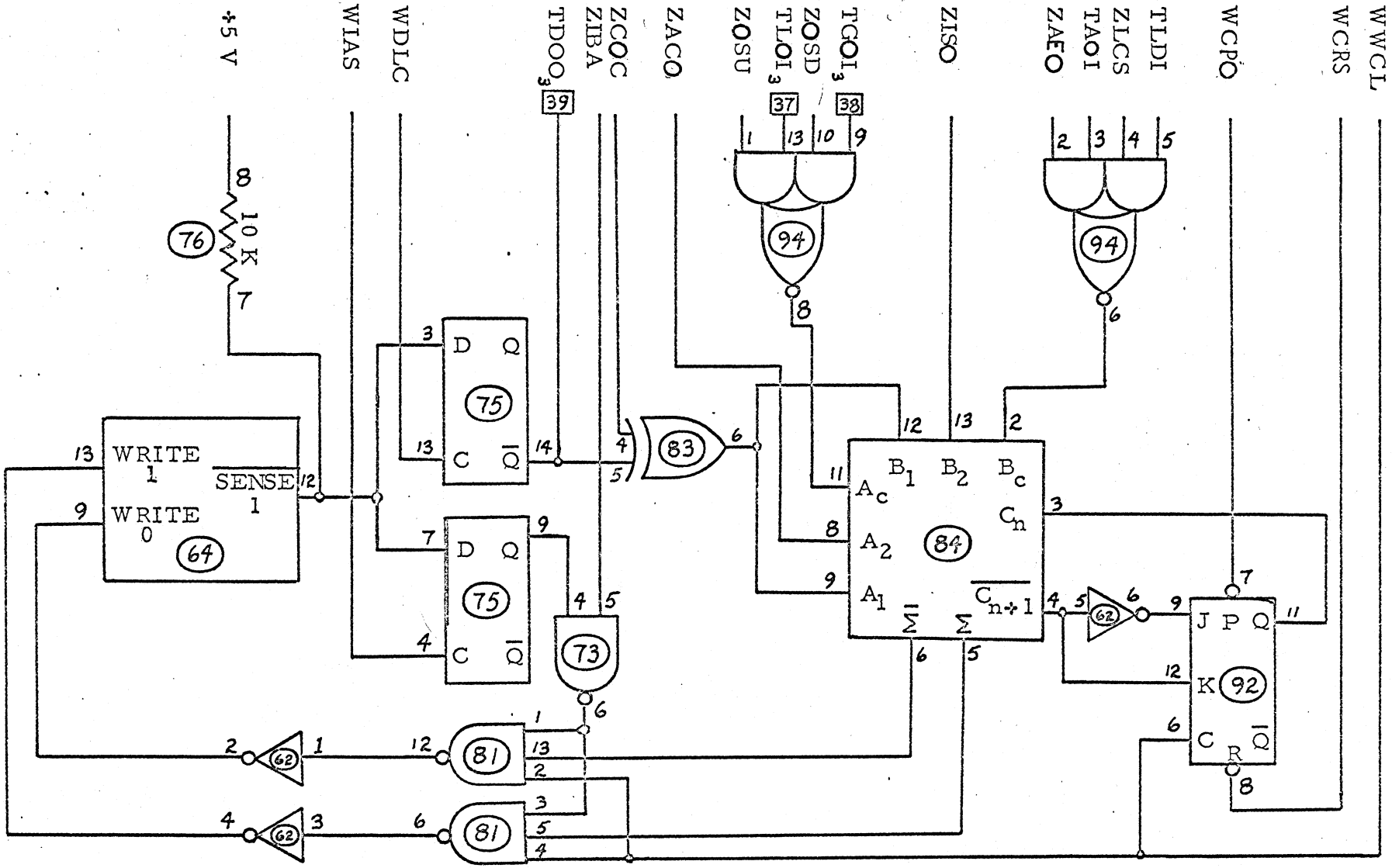


FIGURE B.9 MEMORY BOARD WIRING DIAGRAM: CELL P3-0

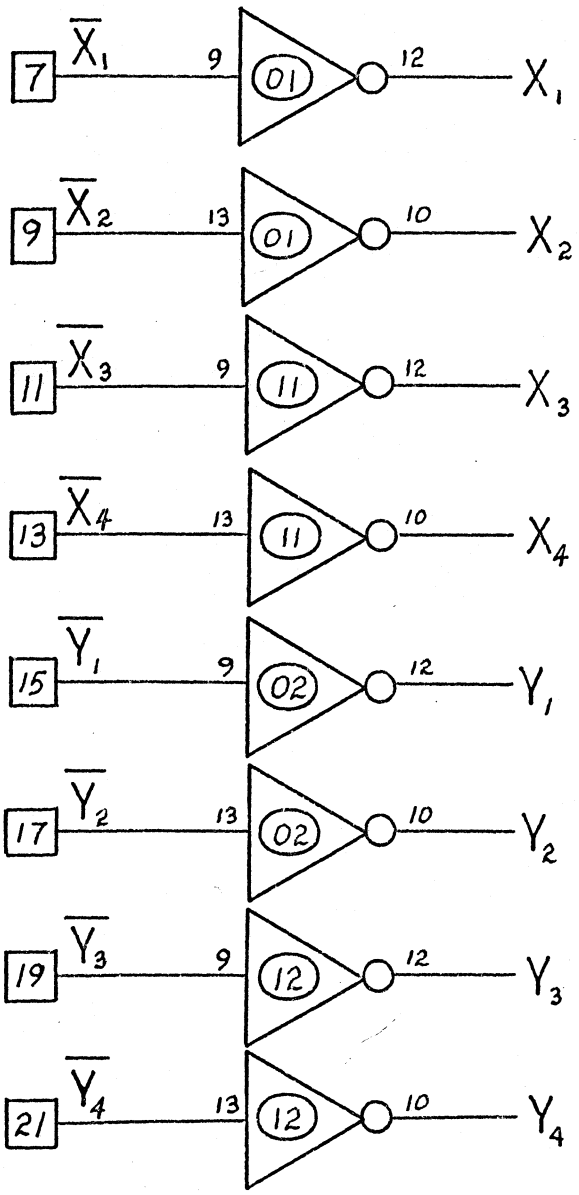


FIGURE B.10 MEMORY BOARD WIRING DIAGRAM:

BIT SELECT DRIVERS

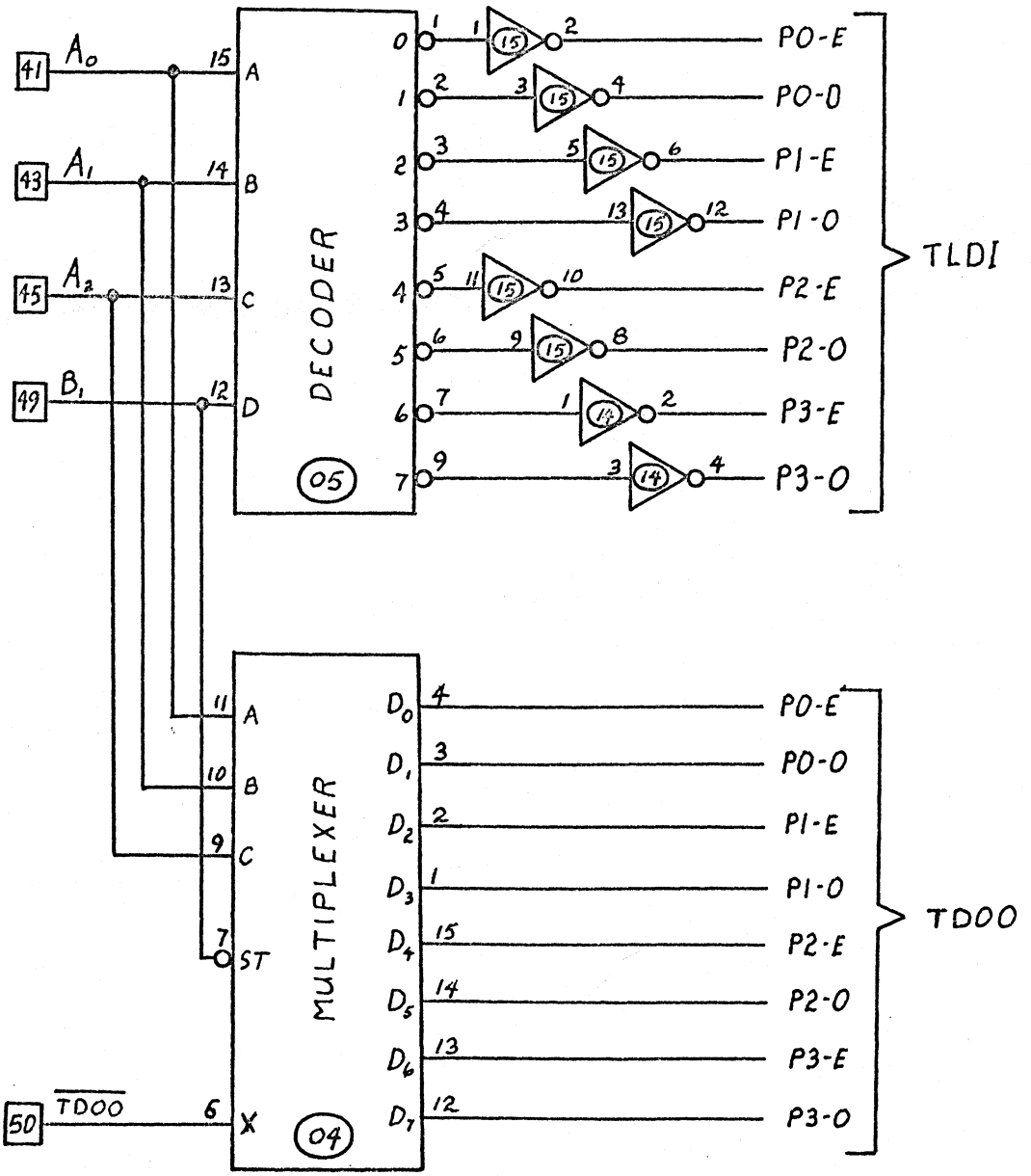


FIGURE B.11 MEMORY BOARD WIRING DIAGRAM:

DATA LOAD AND DATA READ CIRCUIT

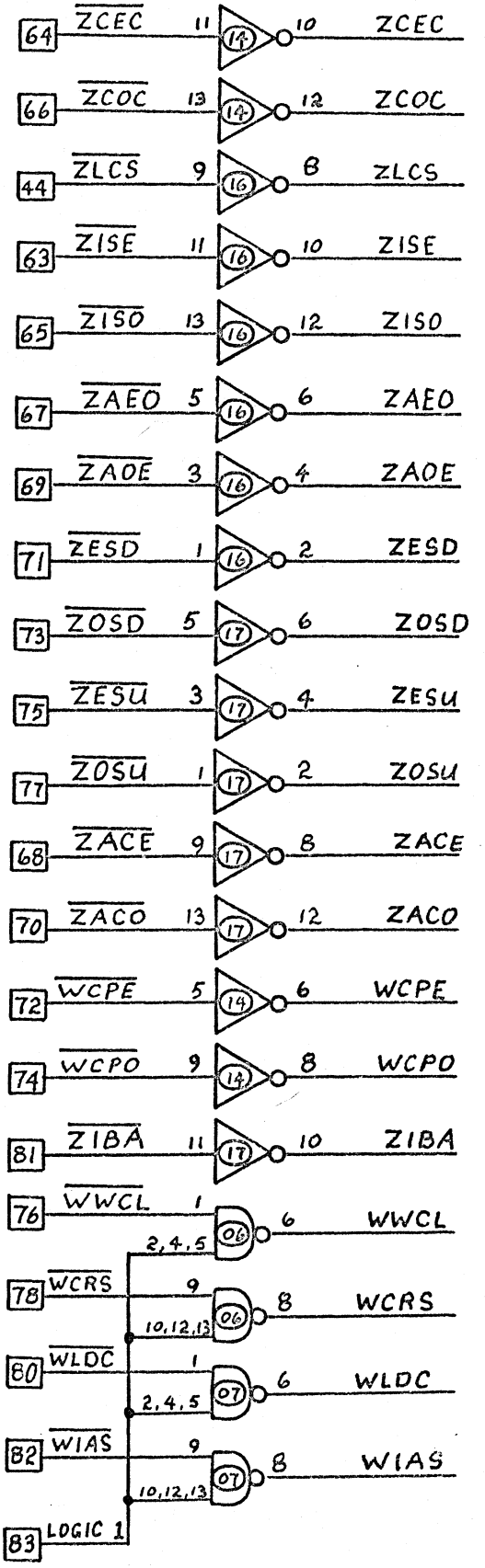


FIGURE B.12 MEMORY BOARD WIRING DIAGRAM:

INSTRUCTION AND PULSE BUFFERS

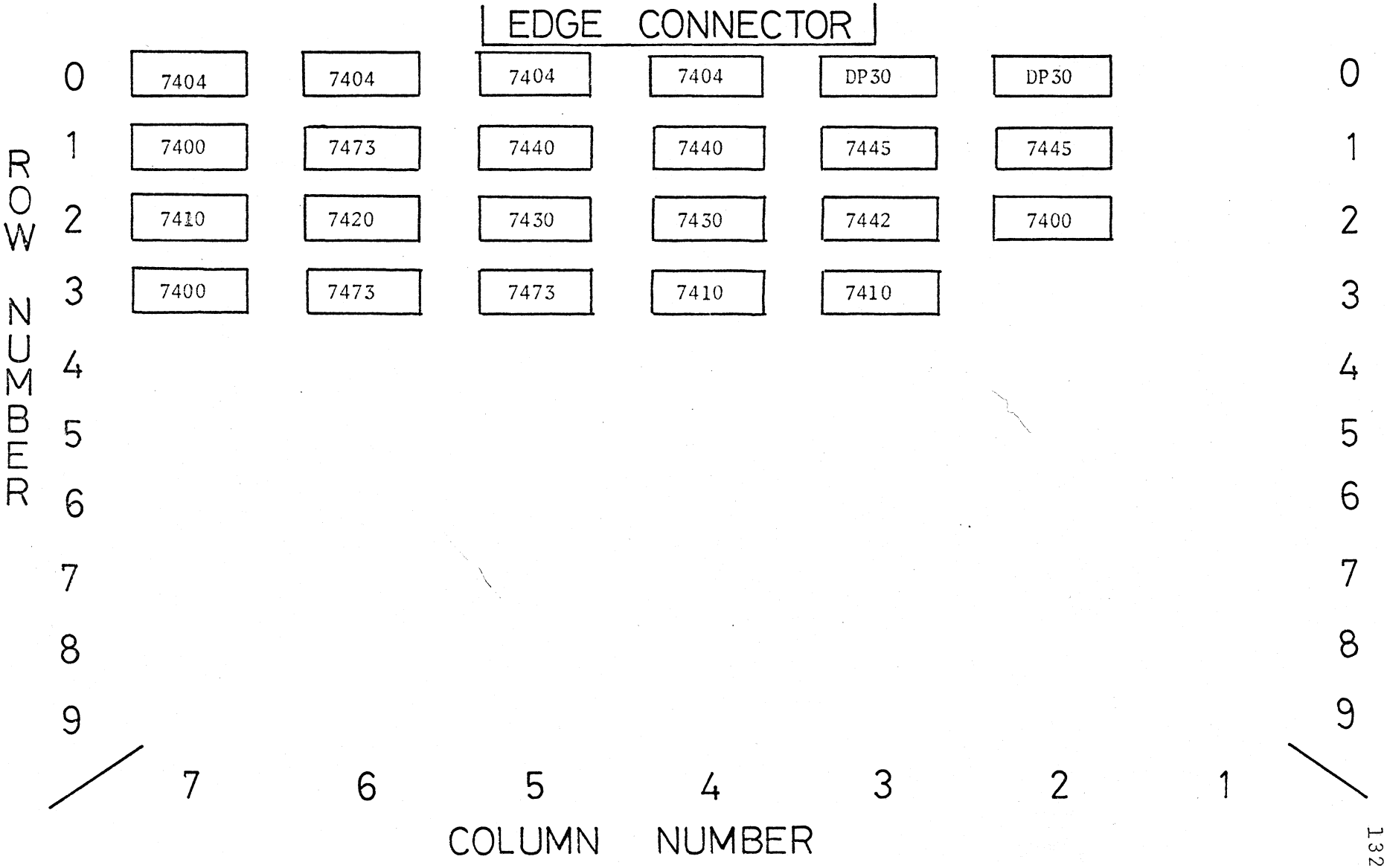


FIGURE B.13 CONTROL BOARD PACKAGE LOCATION CHART

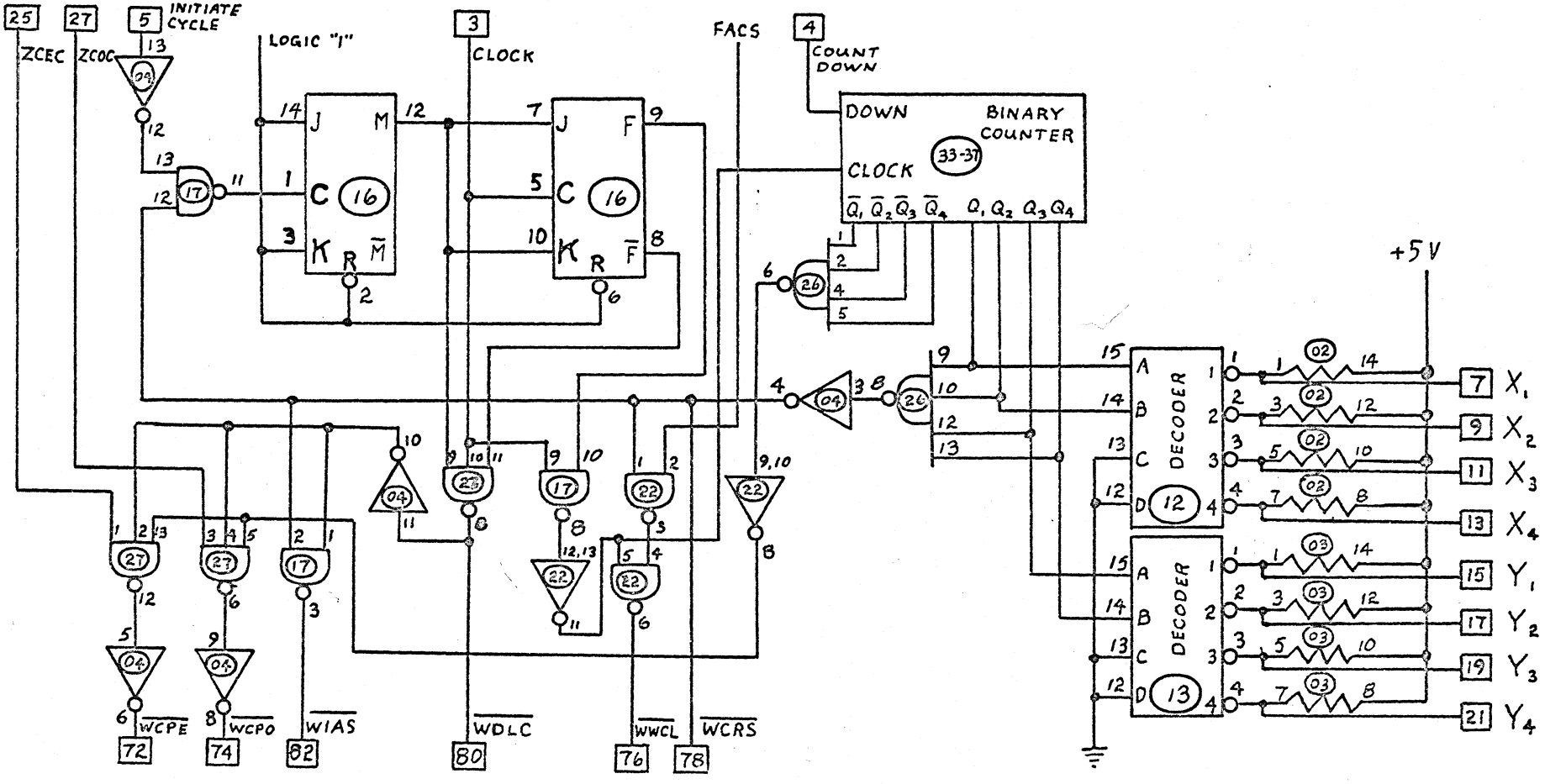


FIGURE B.14 CONTROL BOARD WIRING DIAGRAM: BIT SELECTOR AND PULSE DIRECTOR

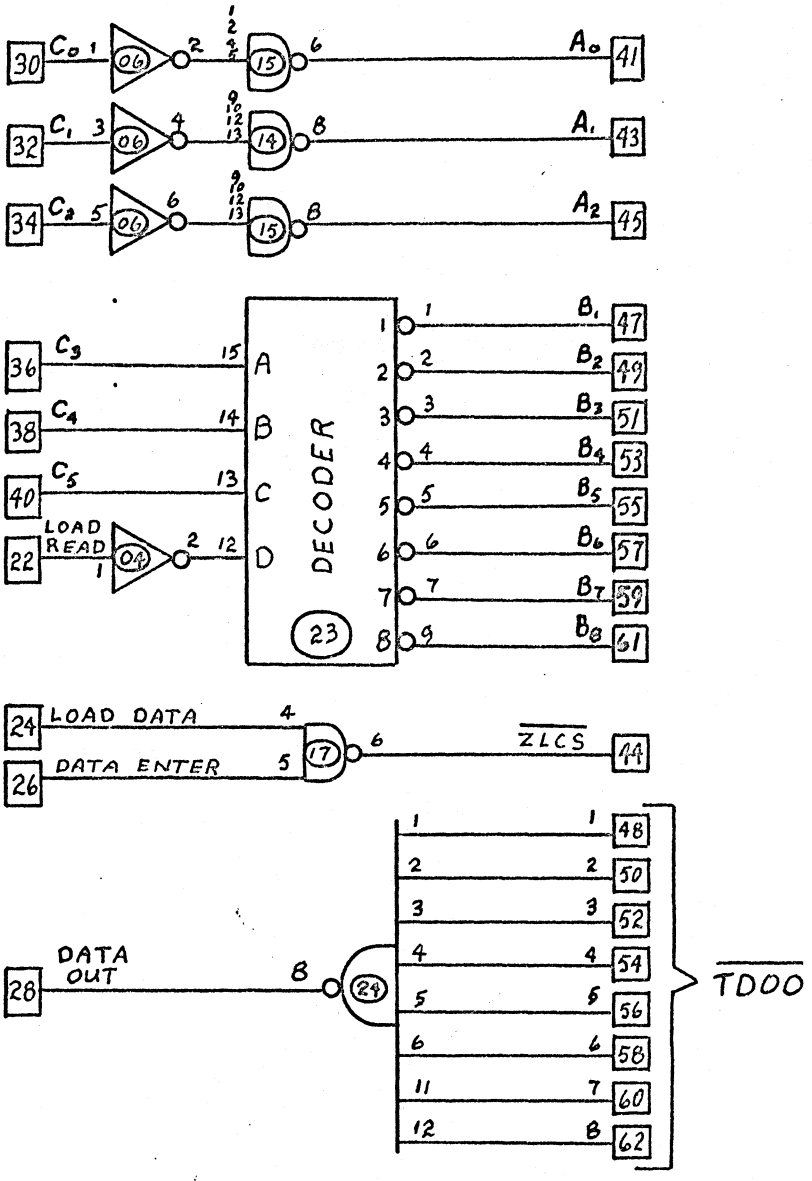


FIGURE B.15 CONTROL BOARD WIRING DIAGRAM:

DATA LOAD AND DATA READ CONTROL

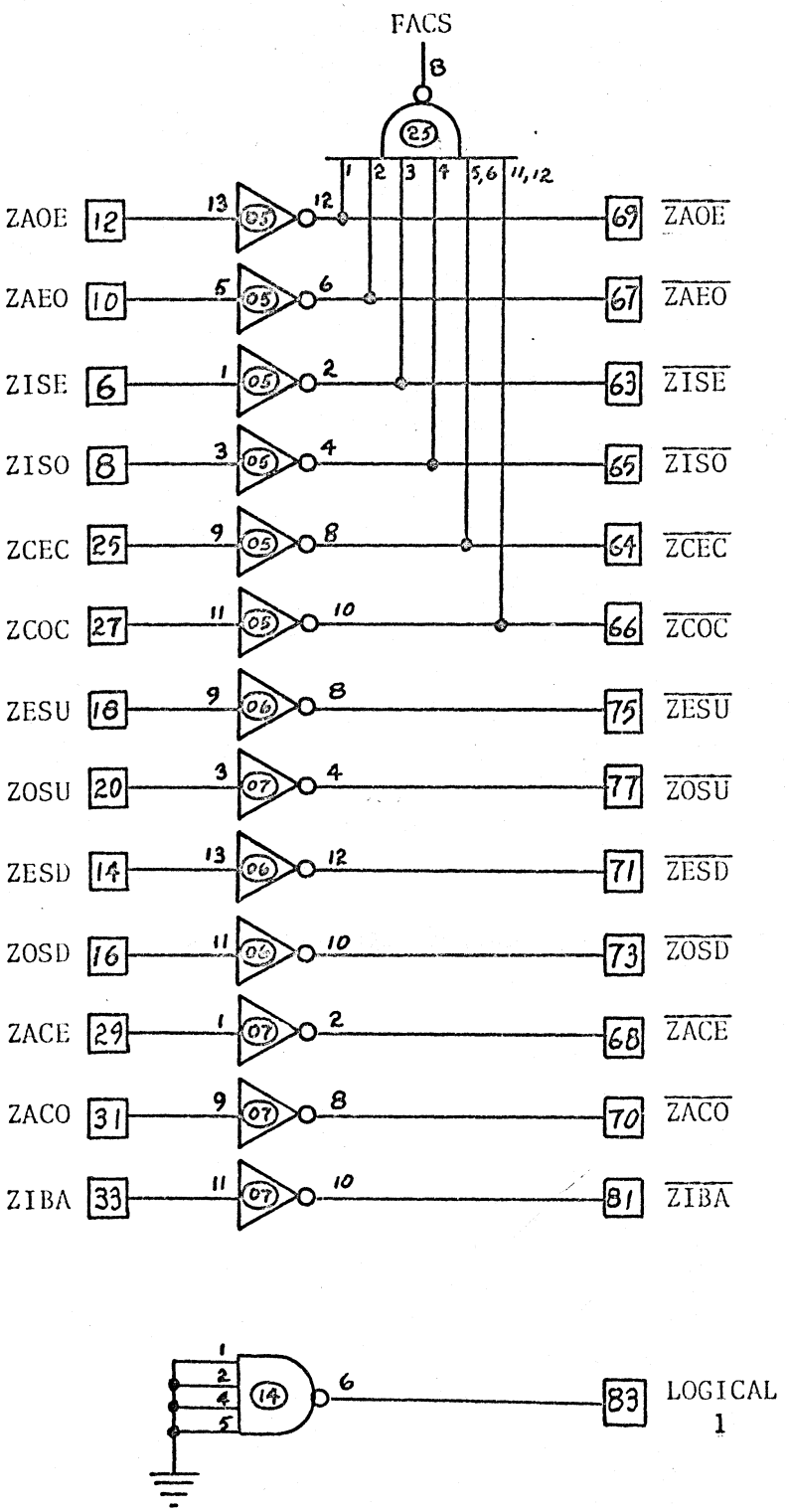


FIGURE B.16 CONTROL BOARD WIRING DIAGRAM:

INSTRUCTION LINE DRIVERS

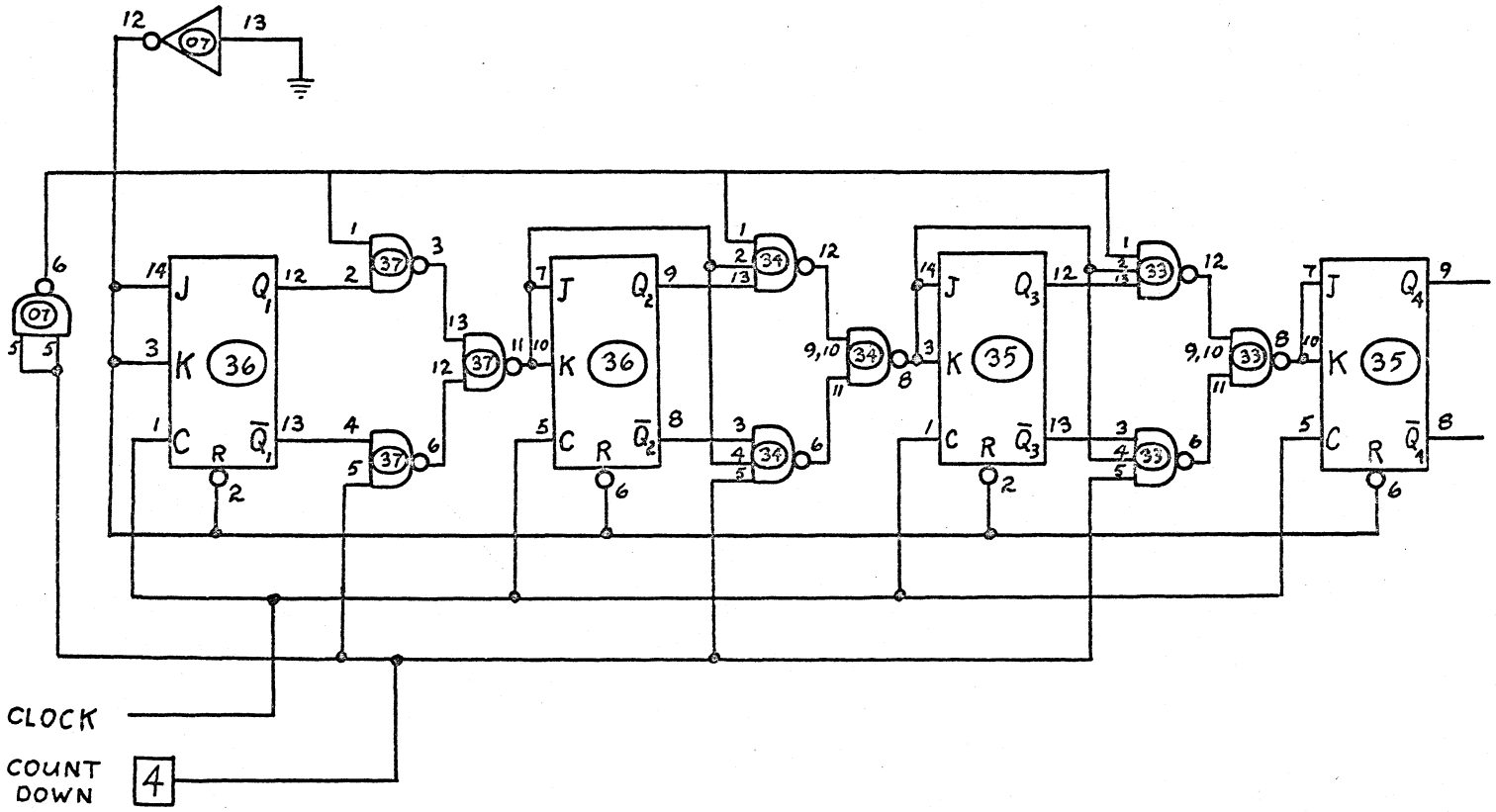


FIGURE B.17 CONTROL BOARD WIRING DIAGRAM: UP/DOWN COUNTER

1.	GROUND	2.	GROUND
3.		4.	
5.		6.	
7.	$\overline{X_1}$	8.	
9.	$\overline{X_2}$	10.	
11.	$\overline{X_3}$	12.	
13.	$\overline{X_4}$	14.	
15.	$\overline{Y_1}$	16.	
17.	$\overline{Y_2}$	18.	
19.	$\overline{Y_3}$	20.	
21.	$\overline{Y_4}$	22.	
23.		24.	
25.	$TLOI_0$	26.	$TGOI_0$
27.	$TDOO_0$	28.	
29.	$TLOI_1$	30.	$TGOI_1$
31.	$TDOO_1$	32.	
33.	$TLOI_2$	34.	$TGOI_2$
35.	$TDOO_2$	36.	
37.	$TLOI_3$	38.	$TGOI_3$
39.	$TDOO_3$	40.	
41.	A_0	42.	
43.	A_1	44.	$\overline{ZLCS}$

TABLE B.1 MEMORY BOARD OUTPUT PIN ASSIGNMENTS

45.	A_2	46.	
47.		48.	
49.	B_1	50.	$\overline{TDOO(1)}$
51.		52.	
53.	TLEI	54.	TGEI
55.	TLEO	56.	TGEO
57.		58.	
59.		60.	
61.		62.	
63.	$\overline{ZISE}$	64.	$\overline{ZCEC}$
65.	$\overline{ZISO}$	66.	$\overline{ZCOC}$
67.	$\overline{ZAEO}$	68.	$\overline{ZACE}$
69.	$\overline{ZAOE}$	70.	$\overline{ZACO}$
71.	$\overline{ZESD}$	72.	$\overline{WCPE}$
73.	$\overline{ZOSD}$	74.	$\overline{WCPO}$
75.	$\overline{ZESU}$	76.	$\overline{WWCL}$
77.	$\overline{ZOSU}$	78.	$\overline{WCRS}$
79.		80.	$\overline{WDLC}$
81.	$\overline{ZIBA}$	82.	$\overline{WIAS}$
83.	LOGIC "1"	84.	
85.	V_{cc}	86.	V_{cc}

TABLE B.1 (cont) MEMORY BOARD OUTPUT PIN ASSIGNMENTS

1.	GROUND	2.	GROUND
3.	CLOCK	4.	COUNT DOWN
5.	INITIATE CYCLE	6.	ZISE
7.	$\overline{X_1}$	8.	ZISO
9.	$\overline{X_2}$	10.	ZAEO
11.	$\overline{X_3}$	12.	ZAOE
13.	$\overline{X_4}$	14.	ZESD
15.	$\overline{Y_1}$	16.	ZOSD
17.	$\overline{Y_2}$	18.	ZESU
19.	$\overline{Y_3}$	20.	ZOSU
21.	$\overline{Y_4}$	22.	LOAD/READ
23.		24.	LOAD DATA
25.	ZCEC	26.	DATA ENTER
27.	ZCOC	28.	DATA OUT
29.	ZACE	30.	C_0
31.	ZACO	32.	C_1
33.	ZIBA	34.	C_2
35.		36.	C_3 CELL
37.		38.	C_4 ADDRESS
39.		40.	C_5
41.	A_0	42.	
43.	A_1	44.	$\overline{ZLCS}$

TABLE B.2 CONTROL BOARD OUTPUT PIN ASSIGNMENTS

45.	A_2	46.	
47.	B_1	48.	$\overline{TDOO}_1$
49.	B_2	50.	$\overline{TDOO}_2$
51.	B_3	52.	$\overline{TDOO}_3$
53.	B_4	54.	$\overline{TDOO}_4$
55.	B_5	56.	$\overline{TDOO}_5$
57.	B_6	58.	$\overline{TDOO}_6$
59.	B_7	60.	$\overline{TDOO}_7$
61.	B_8	62.	$\overline{TDOO}_8$
63.	$\overline{ZISE}$	64.	$\overline{ZCEC}$
65.	$\overline{ZISO}$	66.	$\overline{ZCOC}$
67.	$\overline{ZAEO}$	68.	$\overline{ZACE}$
69.	$\overline{ZAOE}$	70.	$\overline{ZACO}$
71.	$\overline{ZESD}$	72.	$\overline{WCPE}$
73.	$\overline{ZOSD}$	74.	$\overline{WCPO}$
75.	$\overline{ZESU}$	76.	$\overline{WWCL}$
77.	$\overline{ZOSU}$	78.	$\overline{WCRS}$
79.		80.	$\overline{WDLC}$
81.	$\overline{ZIBA}$	82.	$\overline{WIAS}$
83.	LOGIC "1"	84.	
85.	V_{cc}	86.	V_{cc}

TABLE B.2(cont) CONTROL BOARD OUTPUT PIN ASSIGNMENTS

REFERENCES

1. Hanlon, A.G.; "Content-addressable and associative memory systems, A Survey", IEEE Trans. Electronic Computers, Vol.EC-15, pp.509-521, August 1966.

2. Rux, P.T.; "A Glass Delay Line Content-Addressed Memory System", IEEE Trans. Computers, Vol.C-18, pp.512-520, June 1969.

3. Koczela, L.J.; Wang, G.Y.; "The Design of a Highly Parallel Computer Organization", IEEE Trans. Computers, Vol.C-18, pp.520-529, June 1969.

4. Huttenhoff, J.H.; Shively, R.R.; "Arithmetic Unit of a Computing Element in a Global, Highly Parallel Computer", IEEE Trans. Computers, Vol. C-18, pp.695-698, August 1969.

5. Campeau, J.O.; "The Block-Oriented Computer", IEEE Trans. Computers, Vol.C-18, pp.706-718, August 1969.

6. Kautz, W.H.; "Cellular Logic-in-Memory Array", IEEE Trans. Computers, Vol.C-18, pp.719-727, August 1969.

7. Solodovnikov, V.V.; "Introduction to the Statistical Dynamics of Automatic Control Systems", Dover, 1960, pp.116-122.

8. Gaines, B.R.; "Stochastic Computer Thrives on Noise", Electronics, McGraw-Hill, Vol.40, July 10, 1967, pp.72-79.

9. Jespers, P.; Chu, P.T.; Fettweis, A.; "A new method to compute correlation functions", presented at International Symposium on Information Theory, Brussels, September 3-7, 1962.

10. Kitai, R.; Masuko, A.; "Digital instrumentation for measurement of autocorrelation and moments", Proc. IEE, Vol.116, pp.1950-1956, November 1969.

11. Della Torre, E.; Longo, C.V.; "The Electromagnetic Field", Allyn and Bacon, 1969, pp.231-234.

12. Masuko, A.; "Digital Auto-Correlator: Design and Construction", McMaster University, 1969.

13. Deist, F.; Kitai, R.; "Digital Transfer Voltmeters: Principles and Error Characteristics", Proc. IEE, Vol.110, pp.1887-1904, October 1963.

14. Beckman, P.; "Elements of Applied Probability Theory" Harcourt, Brace, and World, 1967, pp.119-128.

15. Siemens, K.H.; "Digital Walsh-Fourier Analyser for Periodic Waveforms", McMaster University, 1969.