ELECTRICAL ENGINEERING DEPARTMENT

FINAL REPORT. Contract NSG-5002

Architecture and Data Processing Alternatives for the Tse Computer

VOLUME 2: Extraction of Topological Information from an Image By the Tse Computer

J. R. Jones

R. E. Bodenheimer

TECHNICAL REPORT TR-EE/CS-76-2

September 1976

UNIVERSITY OF TENNESSEE

KNOXVILLE
TN 37916

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

FINAL REPORT. Contract NSG-5002

Architecture and Data Processing Alternatives for the Tse Computer

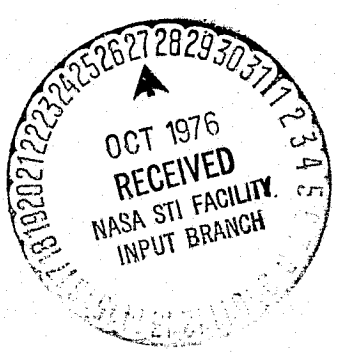
VOLUME 2: Extraction of Topological Information from an Image By the Tse Computer

J. R. Jones

R. E. Bodenheimer

TECHNICAL REPORT TR-EE/CS-76-2

September 1976



ARCHITECTURE AND DATA PROCESSING ALTERNATIVES FOR THE TSE COMPUTER

VOLUME 2: EXTRACTION OF TOPOLOGICAL INFORMATION FROM AN IMAGE BY THE TSE COMPUTER

Robert E. Bodenheimer - Principal Investigator
James R. Jones - Co-Investigator
Department of Electrical Engineering
The University of Tennessee
Knoxville, Tennessee 37916

Final Report. NSG-5002
Period: May 1974 - August 1976

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND 20771

ABSTRACT

A new computing concept, called Tse Computing, has been introduced by D. H. Schaefer and J. P. Strong at Goddard Space Flight Center (GSFC), Greenbelt, Maryland. Tse Computing is an optical, digital, image processing technique which utilizes an entire image as its basic computational entity. The purpose of this investigation was the extension of the research being conducted at GSFC by organizing tse computing machines capable of extracting topological information from an image. The desired information consisted of global and local maxima, first and second partial derivatives, and the gradient of the image.

After summarizing the research at GSFC, a simple programmable tse processor organization was introduced and arithmetic operations necessary for extraction of the desired topological information were developed for this organization. In order to improve the processing performance of the simple tse processor, hardware additions to this organization were introduced with a discussion of the trade-offs peculiar to the tse computing concept. An improved organization was then proposed, with the complementary software for the various arithmetic operations. The performance of the two organizations was then compared in terms of speed, power, and cost.

Utilizing the instruction set of the improved tse processor, software routines were developed to extract the desired information from an image.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

CHAPTER I

INTRODUCTION

Since the 1940's, when the first relay computers were introduced, there has been rapid improvement in the processing capabilities of these machines. Most of these improvements may be attributed to the development of faster and more sophisticated hardware. However, considering the pace of hardware development in recent years, the possibility exists that the maximum velocity at which electrical signals can propagate will soon establish a limit on the amount of data that future generation computers can process in a given time frame. There have been many methods devised to circumvent this physical limitation. These methods, in the areas of "Multiprocessing" and "Parallel Processing," have become especially important in applications for which the data flow would tax even the fastest of the present generation computers. One such application occurs in array processing tasks in which the data is two-dimensional.

The two-dimensional data processing problem was addressed by Unger [3] in 1958, and later by Kruse [4] and others [5,6]. Parallel processing machines, essentially of the type shown in Figure 1.1, were proposed. These machines were conceived as an array of processing elements controlled by a single control unit. Each processing element is a self-contained computer, capable of performing at least a minimum number of logical operations. To provide data exchanges and inter-communication between adjacent cells in the array, each element is

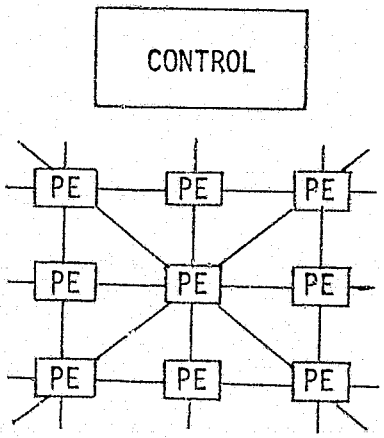


FIGURE 1.1. Organization of a Parallel Processor

connected to its nearest neighbor. These parallel processing machines offered a significant advantage over the conventional computer when addressed to array processing tasks; the instruction execution time or simply the processing speed was independent of the size of the array. Thus these machines were capable of processing large arrays many times faster than a conventional computer. Unfortunately, as the size of the array increased, so did the hardware costs. Due to the hardware costs, these machines were feasible only for the smallest arrays. However, a small version of a computer of this type was produced in 1965 and called the Solomon computer [5]. Studies of the Solomon computer led to the development of Burroughs' Illiac IV array processing computer in 1972 [5]. The Illiac IV system is composed of four reconfigurable arrays of the type shown in Figure 1.1, each containing 64 processing elements. The control unit is a Burroughs' B6500 general purpose computer. This system is capable of greatly reducing the computation time necessary for matrix manipulations compared to conventional computers. However, for a picture or image processing application in which very large arrays are required for reasonable resolution, even this "super-computer" is strained.

Since some computational tasks, such as fourier transforms, are being performed optically, the utilization of optical methods for image processing tasks seems a natural, and efficient method of computation. However, although many optical analog methods have been successfully implemented, optical digital computing is not so well developed.

When the National Aeronautics and Space Administration began its Earth Resources program in 1967 [7], the processing of the large amount

of data in image form that was gathered by satellite was of major concern. In order to properly account for and manage the earth's resources, the data must be processed and disseminated in a reasonable amount of time. NASA projects that by 1980, 50,000 images per day will be acquired. With this in mind, new processing methods capable of much greater performance than that available with conventional computers are being investigated. Research of one group at Goddard Space Flight Center, (GSFC), (investigation is not complete) generated the concept of a new family of computers, called tse[1] computers [3]. These computers, conceived at NASA, GSFC, utilize an entire image as its basic computational entity, instead of a single bit as in a conventional digital computer.

The purpose of this study is to investigate the data processing, and consequently the architectural alternatives in the organization of a tse computer to extract certain topological information from an image. The desired information consists of the global and local maxima, first and second partial derivatives, and the gradient of the image. Chapter II discusses the tse concept at its present stage of development, and the remainder of this study is devoted to the organization of a machine to perform the desired operations.

---

[1] tse is the English transliteration of the Chinese word for a pictograph character.

CHAPTER II

TSE CONCEPT

To facilitate the processing of pictorial information rapidly and at reasonable cost, Schaefer and Strong [1], at Goddard Space Flight Center, (GSFC), have exploted the feasibility of performing conventional logical operations on entire images simultaneously using optical techniques. Their research concludes that a computer organization composed of electro-optical devices capable of performing logical operation on binary images has enormous potential in the area of image processing. This chapter summarizes the research at GSFC.

The basic computational entity of the tse computer is a binary image, called a tse. Since pictorial information is naturally represented by the distribution of a reflectance or density parameter, imaged information is acquired in an analog form. In order to put this pictorial information in a form useful for tse computer processing, the image must be digitized into a specified number of "grey-levels" to give a string of binary images, or tses. This process is illustrated in Figure 2.1. The digitization process, performed by image threshold devices, characterizes the tse as a binary image whose positional elements contain either a "0" (black - the absence of light) or a "1" (white - the presence of light), depending on the "grey" content of the corresponding positional elements in the original image. For example, an original image composed of a 1024 × 1024 array of picture elements in which the grey level of each element is quantized to six

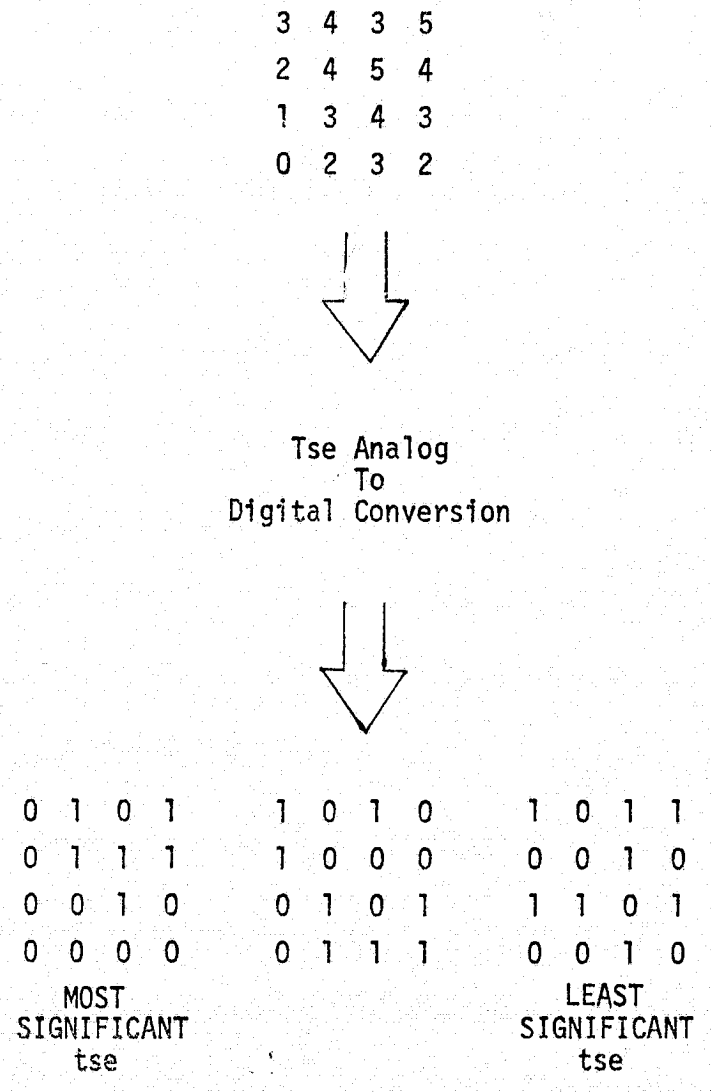


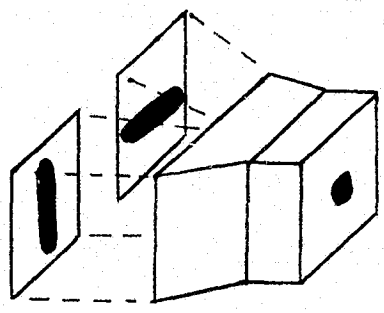
FIGURE 2.1. Tse Digitization Process

bits, would contain over 6×10^6 bits of information. Analogous to a conventional digital computer, the performance of various computations is achieved by the manipulation of the tse data word (string of binary images) in the tse computer.

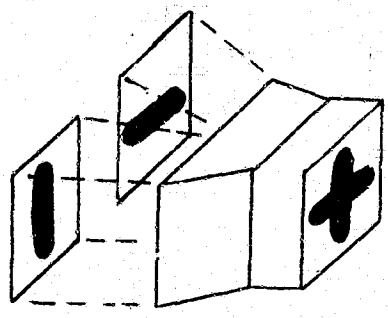
Tse logic components. Schaefer and Strong have proposed a family of tse logic devices which perform basic logical operations on all elements of one or two input tses simultaneously by utilizing electro-optical technology. The basic operations are AND, OR, NOT, EXOR, and SLIDE. These devices are illustrated in Figure 2.2. The method of data transfer that has been chosen is an optical fiber bundle. This bundle contains one optical fiber per element in the array. Optical transport methods were chosen to facilitate the interconnection of components.

Tse logic devices are implemented by bringing the proper fibers of each input tse into adjacency, and then utilizing photoconductors and semiconductor technology to perform the indicated operation. Electroluminescence techniques are then used to derive a light output from the device. The component used to bring the correct fibers of the two input images into adjacency is called an "interleaver." The operation of this strictly optical device is illustrated in Figure 2.3. An important structural advantage to the interleaver is that it can be used in reverse operation as a duplicator of images. A prototype interleaver is shown in Figure 2.4. Its actual length is six centimeters and its weight is 0.7 grams.

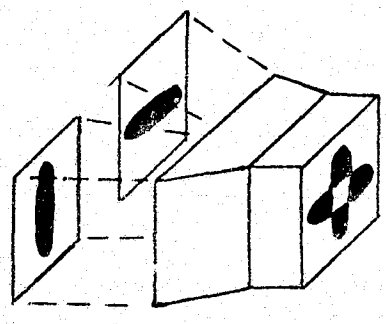
The SLIDE operations are implemented by four separate devices which provide the options of sliding an image UP, DOWN, RIGHT, or



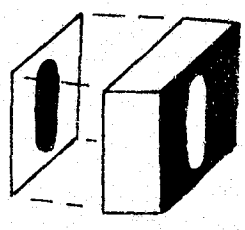
"AND"



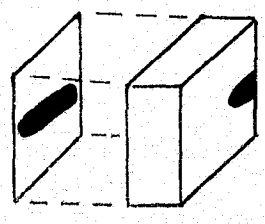
"OR"



"EXCLUSIVE-OR"

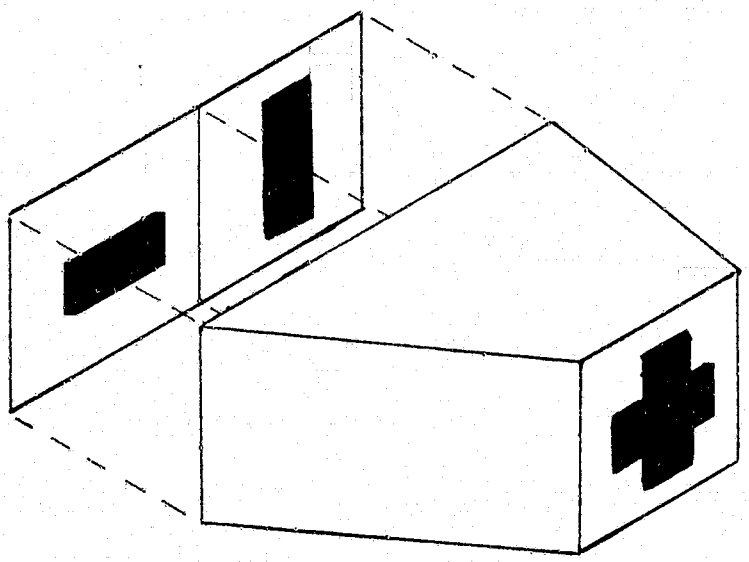


"NOT"

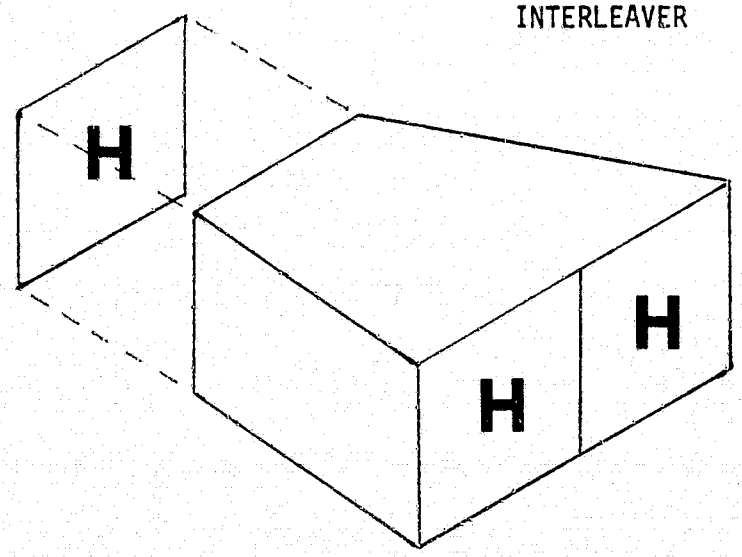


"SLIDE"

FIGURE 2.2. Tse Logic Devices



INTERLEAVER



DUPLICATOR

FIGURE 2.3. Interleaver Operation

FIGURE 2.4. A Prototype Interleaver

(Courtesy of Earth Observation Systems Division, Goddard Space Flight Center)

LEFT in the X and Y directions. These devices facilitate the intercommunication of elements of the array in different coordinate positions. Actual structural details of the sliding devices are shown in Figure 2.5. Sliders are implemented by allowing an interface between the input optical bundle and the output optical bundle which is offset with respect to the input bundle.

Several specialty tse devices have also been proposed. One important example of these specialty devices is called a "contractor." The output of a contractor is a single bit while the input is a binary image. If the input image to the contractor contains at least one element which is logical 1 (white), the single bit output is logical 1. However, if the input image is all logical 0 (black), the single bit output will be logical 0. This device is useful for intercommunication with conventional digital systems.

The interconnection of tse logic devices in the organization of a tse computer to perform the operations discussed in Chapter I must be achieved with the performance specifications of the tse devices in mind. GSFC projects a propagation delay through each tse logic component of five milliseconds and a power requirement of not more than three watts [1]. Unless one recalls that each device is performing simultaneous operations on many bits of information, these specifications seem rather pessimistic. Nevertheless, the designer of an optimal tse computer must be concerned with the normal trade-offs necessary to perform the desired tasks as fast as possible and at a reasonable cost. Unfortunately, speed and cost have been traditionally contradictory.

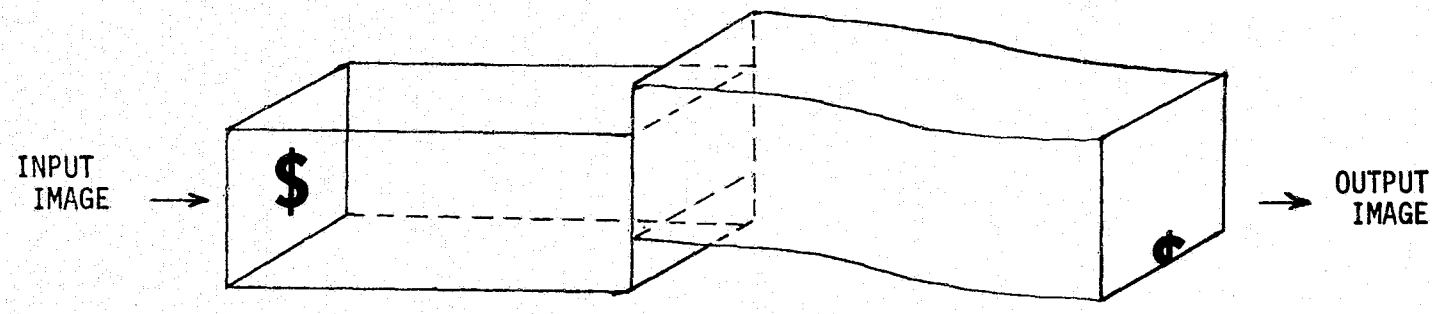


FIGURE 2.5. SLIDE Device Structure

In this respect, there is a similar trade-off in the organization of the tse computer.

Another very important consideration in the design of a tse computer is the fan-out/fan-in restriction. In the present stage of development of the tse components, a fan-out of one has been specified. This requires duplication of images in any tse computer organization. As discussed previously, this duplication is accomplished by the tse hardware devices called an "interleaver" coupled with what is essentially a gain device called a "reformatter." Also, a fan-in of not more than two has been specified. This requires, in some applications, the use of more than one tse logic device when one device with a greater fan-in would be sufficient. In addition to the extra hardware required as a result of these specifications, additional tse logic devices add propagation delay to the data path. Simplicity of design in the tse logic networks required is an important consideration. In order to better evaluate the relative merits of one organization to another, a tse hardware cost function has been proposed.

Cost function. For a cost function to relate meaningfully to the tse computer concept, active devices (with gain) should be distinguised from passive devices (without gain). For example, the AND, OR, NOT, EXOR, and "reformatter" tse devices are active, while the "interleaver" is a passive device. A distinction must be made between these devices because active devices consume power while the passive device does not. However, the passive device contributes size and weight to any organization of the tse computer. Also, the number of inter-

connecting optical fiber bundles is a factor contributing to cost and size. Therefore, the proposed cost function is as follows:

$$\text{cost} = Ax + By + Cz$$

where

x = number of active tse logic devices
y = number of passive tse logic devices
z = number of interconnecting fiber bundles
A, B, C are appropriate weights attached to each cost factor.

The tse computer concept is summarized nicely by the illustration of Figure 2.6. The remainder of this study is devoted to the development of a tse computer to perform certain arithmetic operations necessary to extract topological information from an image.

TSE COMPUTER CONCEPT

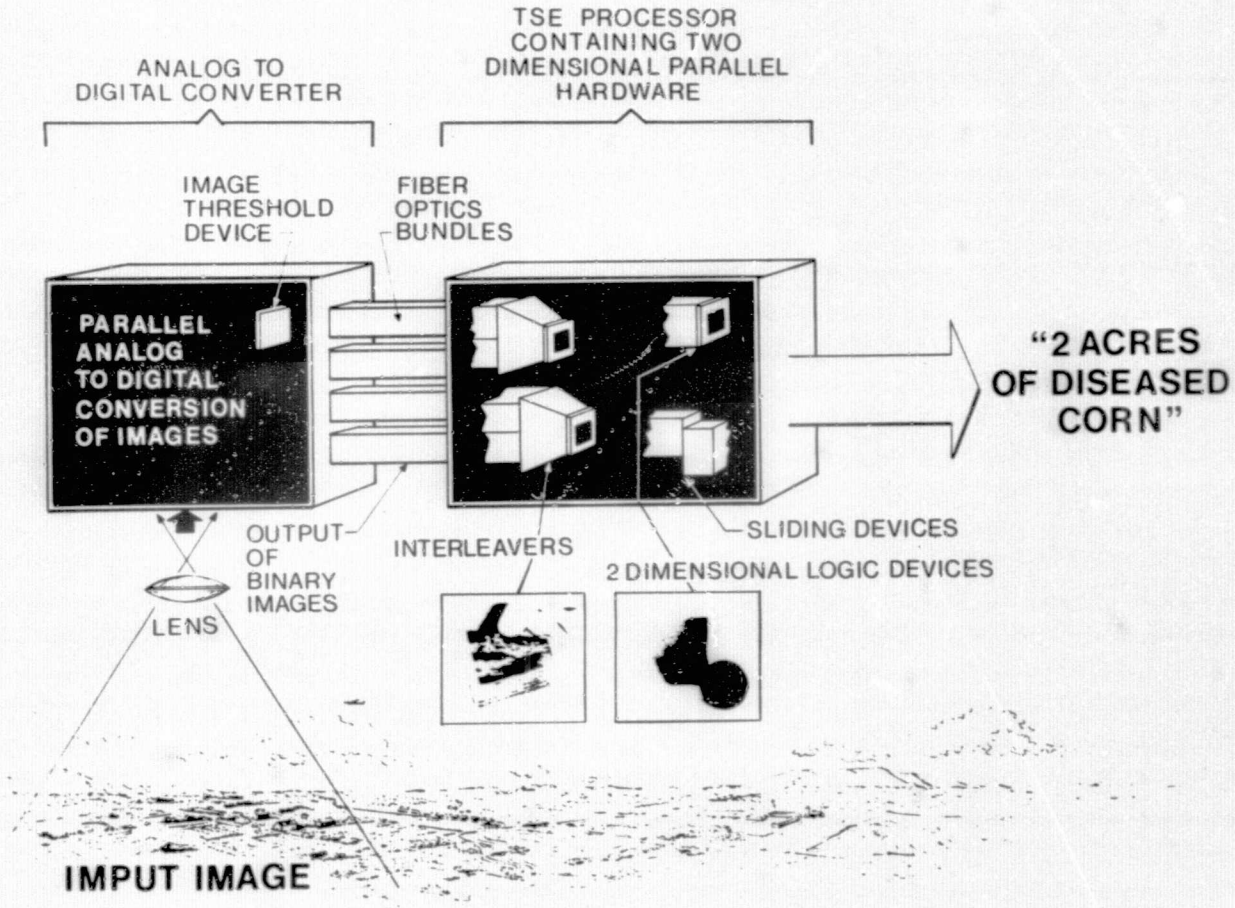


FIGURE 2.6. Tse Computer Concept

(Courtesy of Earth Observation Systems Division, Goddard Space Flight Center)

CHAPTER III

ARITHMETIC OPERATIONS FOR THE TSE COMPUTER

In order to extract the topological information from an image, certain arithmetic operations are required. These operations are basic, and essential to the structure of any machine which manipulates data in a digital manner. As with conventional logic, each of these arithmetic operations may be implemented by software using only the universal set of operators, AND, OR, NOT, and the slide operators of Chapter II. This approach allows a simple ALU organization such as the one shown in Figure 3.1. Starting with this simple structure, more complex tse logic networks may be added to achieve greater performance parameters. However, the restrictions of power requirement, fan-in and fan-out as discussed in Chapter II must be considered when making additions. The possibility exists, that due to the rigid fan-in and fan-out restrictions, the simple ALU structure could prove to have greater processing capability than a more complex structure. This chapter will investigate different organizations to implement the necessary arithmetic operations and compare them with respect to computation times, hardware costs, etc.

Threshold/Compare. The first operation to be discussed is the threshold or comparison operation. This operation is useful in extracting the information from an image which can be determined by a relationship to a specified threshold. For example, areas of an image A whose reflectance is greater than 5, but less than 10, may be

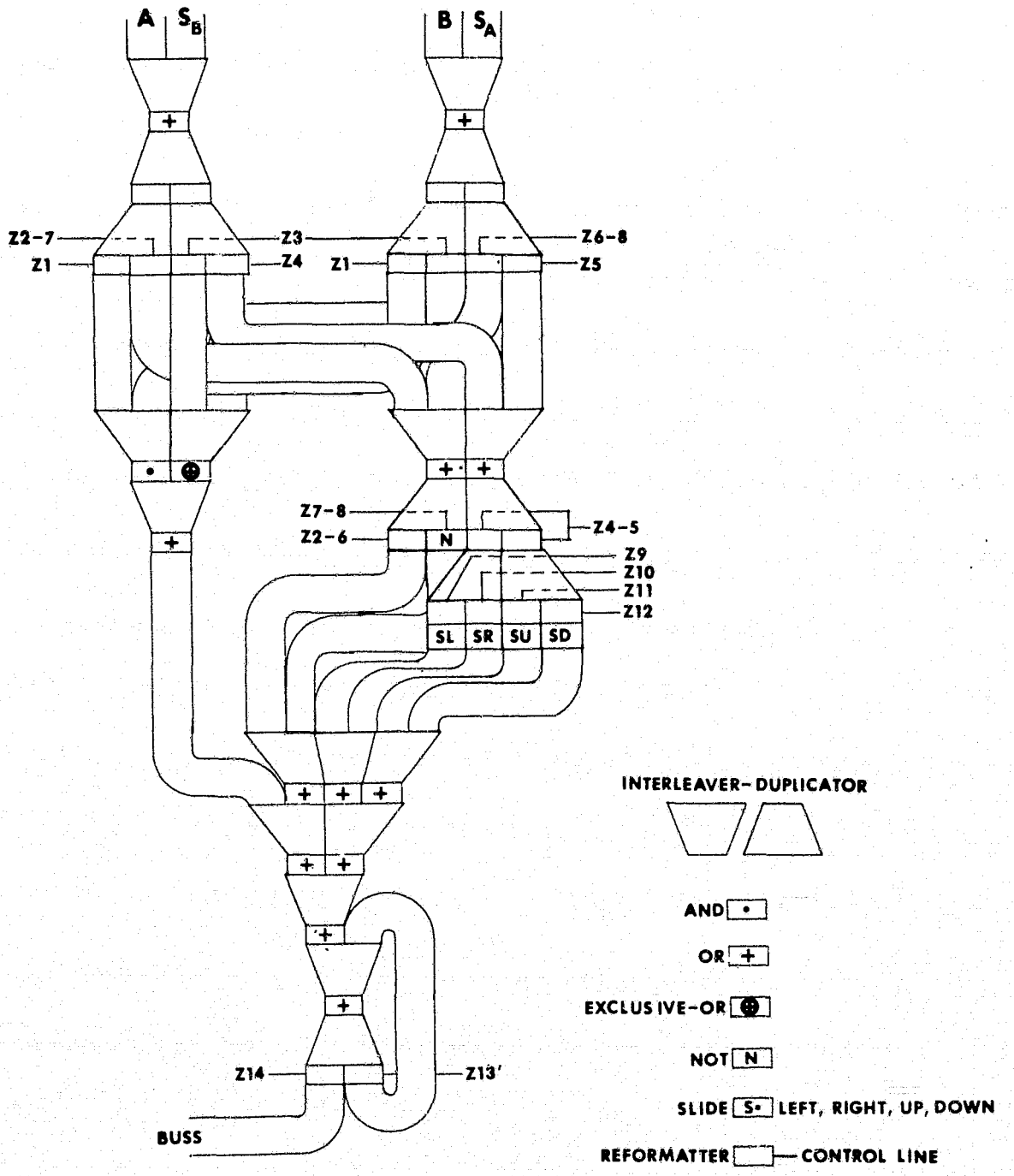


FIGURE 3.1. Simple ALU Organization (Maching 1)

characteristic of the reflectance of a wheat field. The area of this region is desired. This region may be isolated by applying the threshold operation twice; first by extracting areas of image A greater than the specified threshold B_1 ($= 5_{10} = 0101_2$), and then areas of the resulting image which are less than the second threshold B_2 ($= 10_{10} = 1010_2$) are extracted. This isolation of specific areas of the input image would then allow the calculation of the physical area of the regions of the image characterized by the values of certain parameters. This process is illustrated in Figure 3.2. In the above application, the threshold consisted of a digital representation of a real number by either all black (0's) or all white (1's) tses. A more general comparison operation is achieved by letting the individual threshold tses take on any possible matrixed combination of 0's and 1's. With this general comparison function, the threshold/compare operation becomes useful for many other arithmetic applications such as convergence tests, etc.

As stated, all arithmetic operations may be software-implemented on the simple ALU structure of Figure 3.1. A complete CPU organization for the tse computer is shown in Figure 3.3. The individual networks are enclosed by dotted lines and are designated by letters. Each network performs the specific function listed below:

Block A. Provides data paths to the ALU and transfer network B for "A" and "B" accumulators, and the ten intermediate registers of block E.

4	6	7	5		0	0	0	0	1	1	1	1	0	1	1	0	0	0	1	1
5	8	8	6		0	1	1	0	1	0	0	1	0	0	0	1	1	0	0	0
3	9	10	11	=	0	1	1	1	0	0	0	0	1	0	1	1	1	1	0	1
2	11	12	12		0	1	1	1	0	0	1	1	1	1	0	0	0	1	0	0

Image A

5	5	5	5		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
5	5	5	5	=	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
5	5	5	5		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
5	5	5	5		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

B_1 MS LS

0	1	1	0
0	1	1	1
0	1	1	1
0	1	1	1

$A > B_1$ tse

10	10	10	10		1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
10	10	10	10	=	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
10	10	10	10		1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
10	10	10	10		1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0

B_2

1	1	1	1
1	1	1	1
1	1	0	0
1	0	0	0

$A < B_2$

FIGURE 3.2. Isolation of Regions in an Image

0	1	1	0		1	1	1	1		0	1	1	0
0	1	1	1	.	1	1	1	1	=	0	1	1	1
0	1	1	1		1	1	0	0		0	1	0	0
0	1	1	1		1	0	0	0		0	0	0	0
	$A > B_1$					$A < B_2$				Region Isolated			

FIGURE 3.2. (continued)

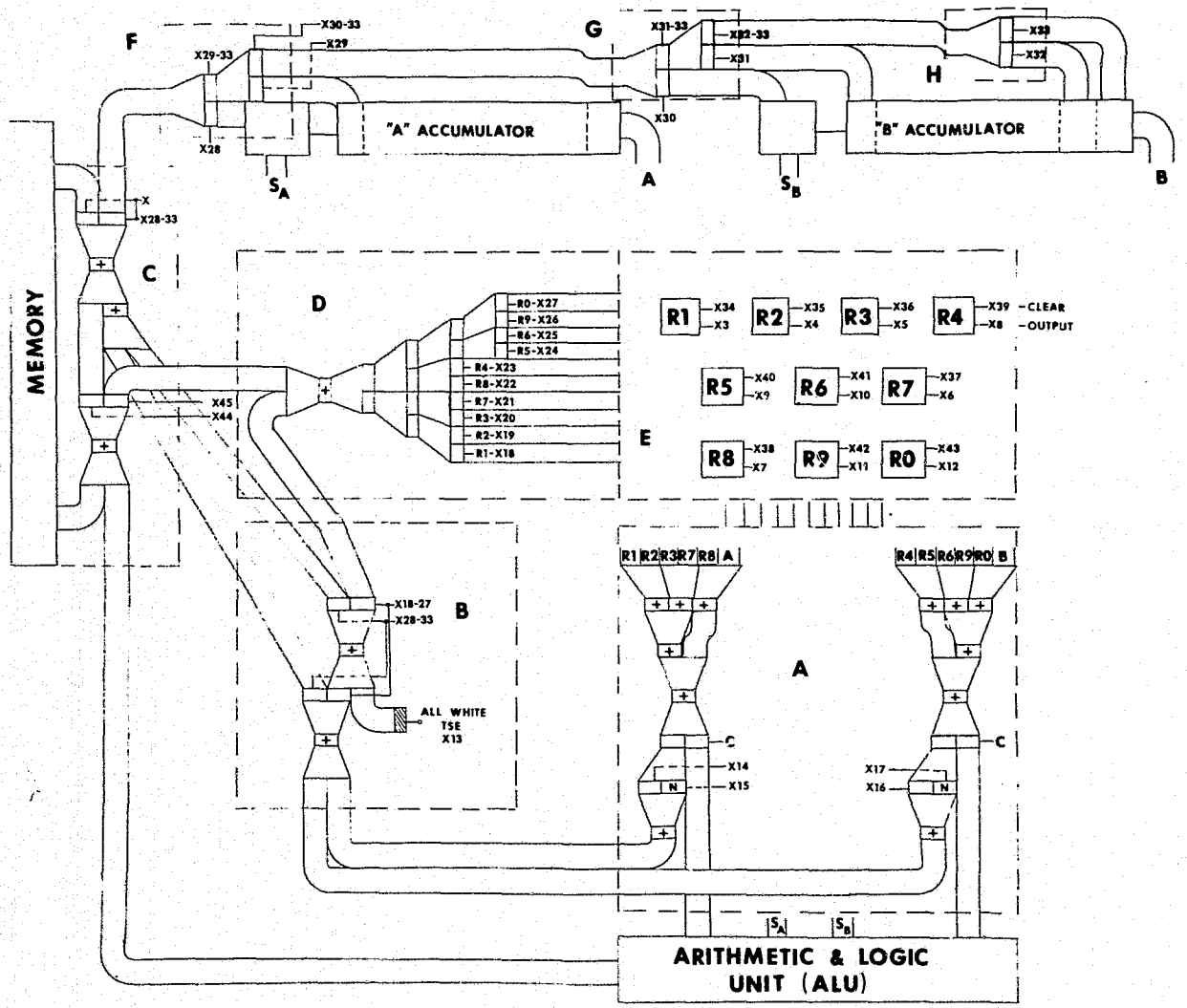


FIGURE 3.3. Organization of the Tse Processor (CPU)

Block B. A transfer network which routes information from network A to the appropriate destination. An all 1's (white) tse is also provided for special purpose loading.

Block C. A transfer network which allows data flow to and from memory, and selects the appropriate path for ALU output or transfer data from network B.

Block D. Provides data paths and selection capability for the ten intermediate registers.

Block E. Ten intermediate registers.

Blocks F-H. Provides data paths and selection capability to "A" and "B" accumulators.

Accumulator organizations for the CPU are shown in Figures 3.4 and 3.5. The subtleties of the CPU organization will become more apparent later. The basic blocks of the CPU organization are as follows:

a. "A" accumulator - Right-shift register - Figure 3.4.

b. "B" accumulator - Right-shift register - Figure 3.5.

c. 10 one-tse latches - Figure 3.6.

d. ALU.

e. Support Hardware.

Note that a significant percentage of the hardware cost of the organization is the support hardware. This is directly attributable to the fan-in and fan-out restrictions.

With the simple ALU structure of Figure 3.1 (page 17) and the above CPU organization, the threshold/compare operation may be software implemented by the routine of Table 1. A complete listing of tse computer

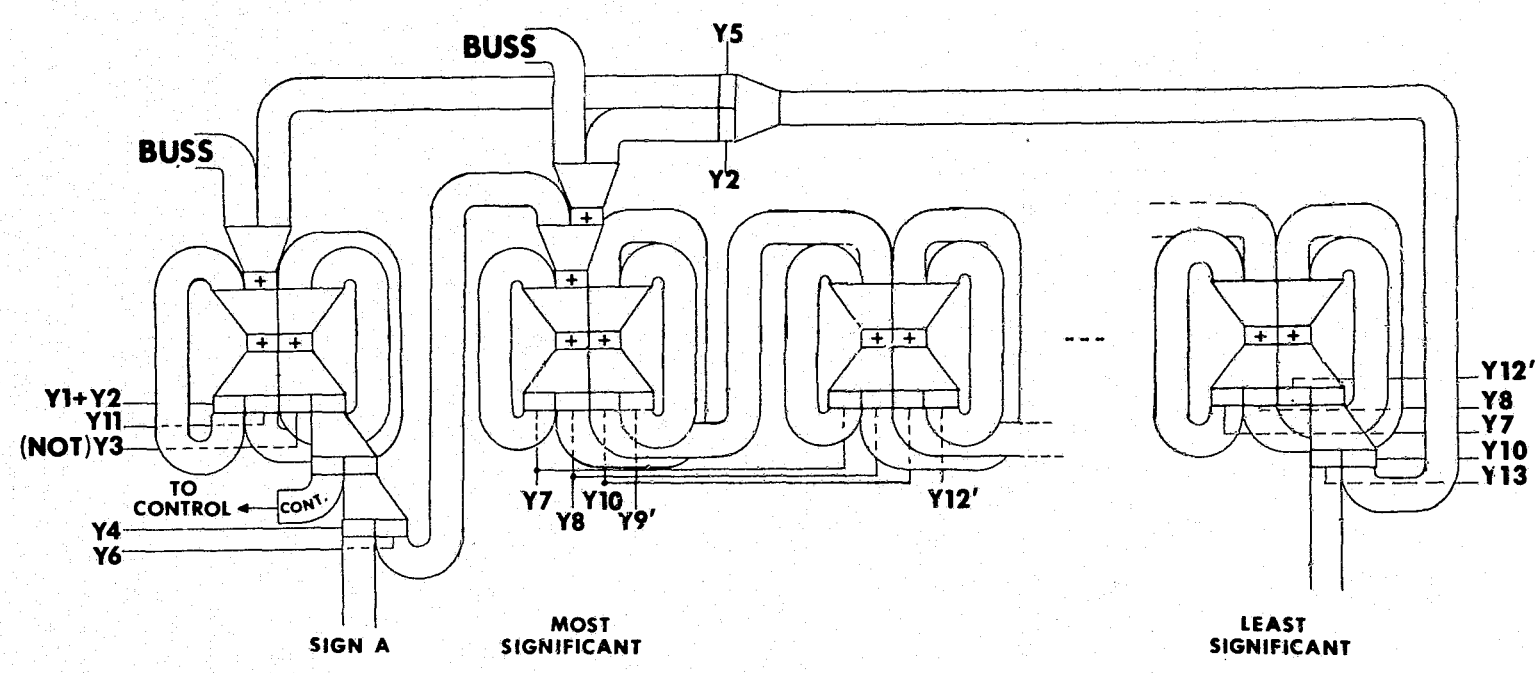


FIGURE 3.4. "A" Accumulator

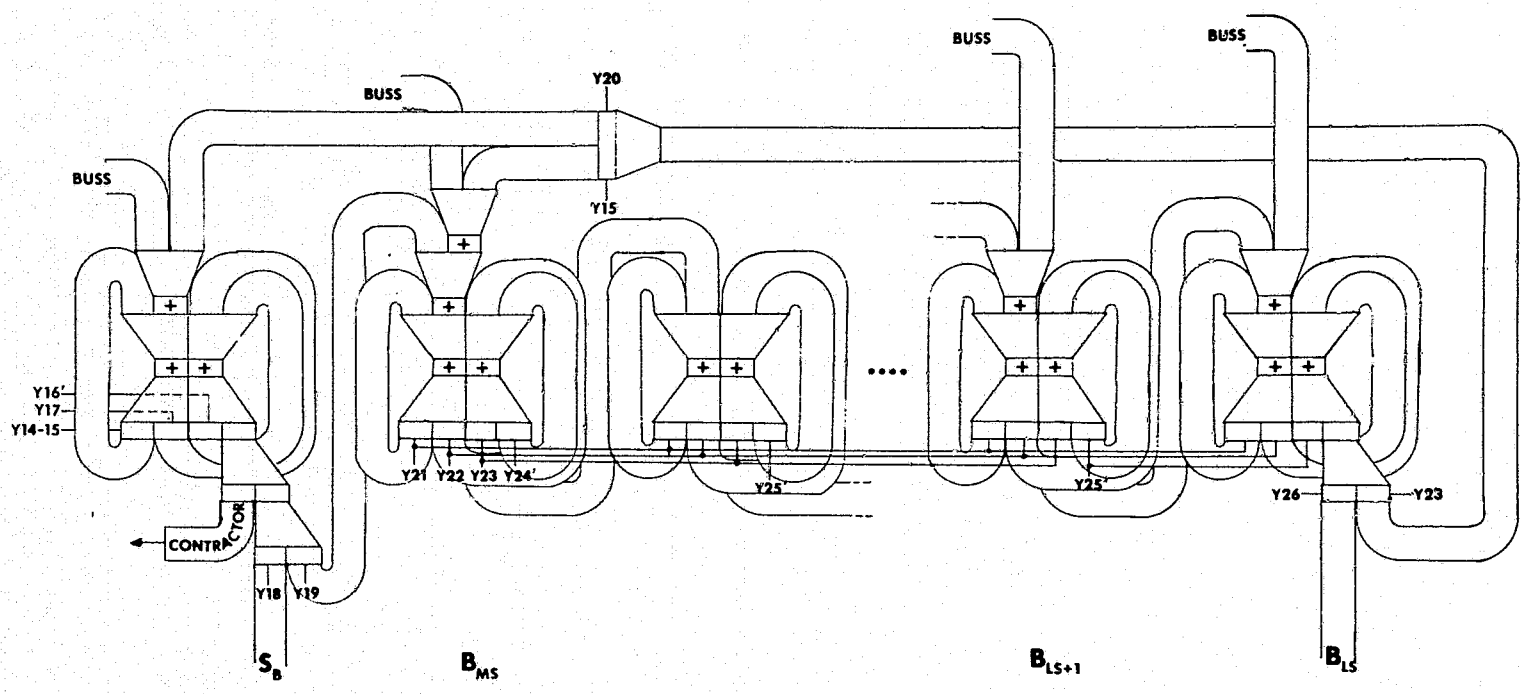


FIGURE 3.5. "B" Accumulator

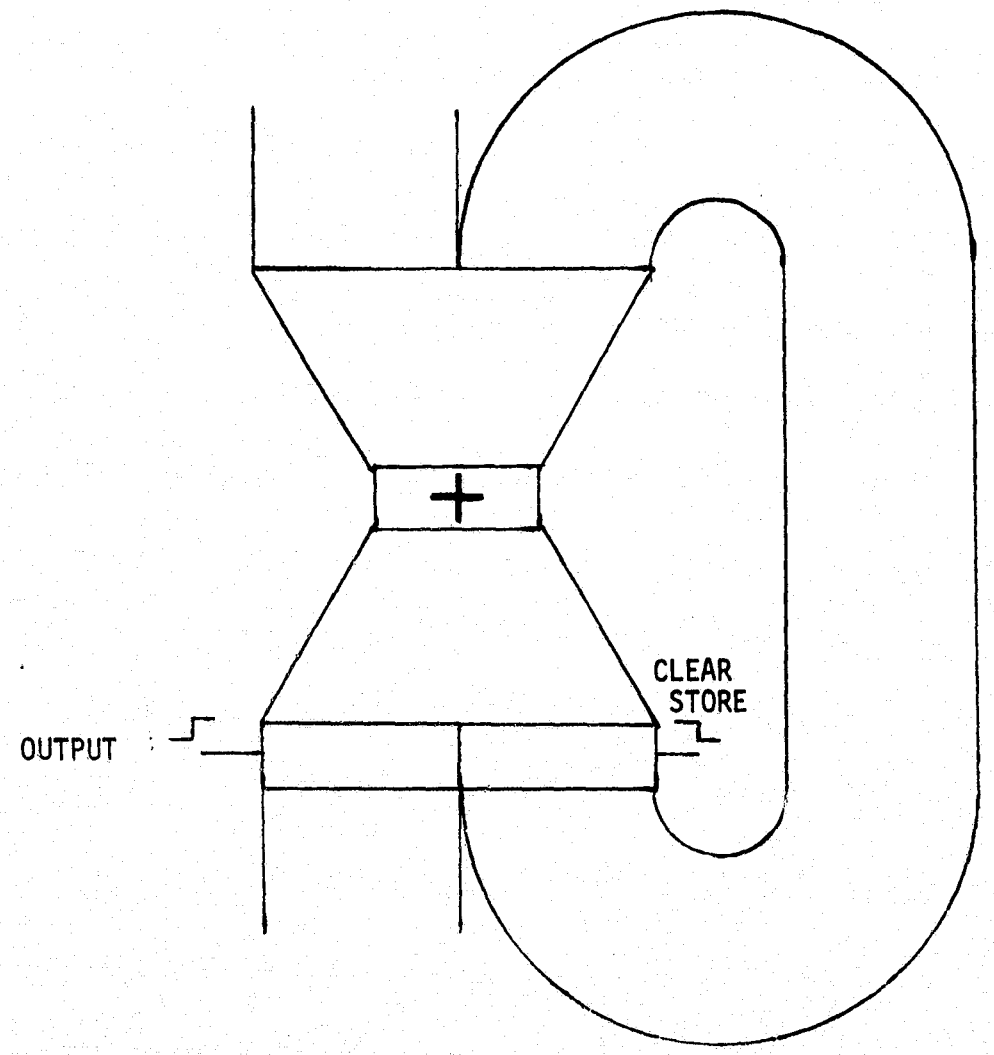


FIGURE 3.6. One-tse Latch

TABLE 1

PROGRAM TO THRESHOLD/COMPARE THE CONTENTS OF
A AND B ACCUMULATORS

(See Appendix for Complete Instruction Set Definition)

NUMBER	INSTRUCTION	TYPE	COMMENTS
1	CONT 5	CONTROL	SUBROUTINE WHICH SETS UP AN ITERATION COUNTER IN CONTROL
2	TRNN B, R4	tse	
3	LAND A, R4, R5	tse	
4	LORE A, R4, R1	tse	
5	LAND R1, R6, R1	tse	REGISTER R6 CONTAINS ALL 0's (BLACK) FOR FIRST ITERATION
6	LORE R1, R5, R6	tse	
7	TRNN A, R1	tse	
8	LAND B, R1, R2	tse	
9	LORE B, R1, R4	tse	
10	LAND R3, R4, R4	tse	REGISTER R3 CONTAINS ALL 0's (BLACK) FOR FIRST ITERATION
11	LORE R2, R4, R3	tse	
12	SHFA 1		
13	SHFB 1		
14	DCR	CONTROL	DECREMENT COUNTER
15	JNZ 2	CONTROL	JUMP TO INSTRUCTION NUMBER 2 WHEN COUNTER IS NON-ZERO
16	RET	CONTROL	

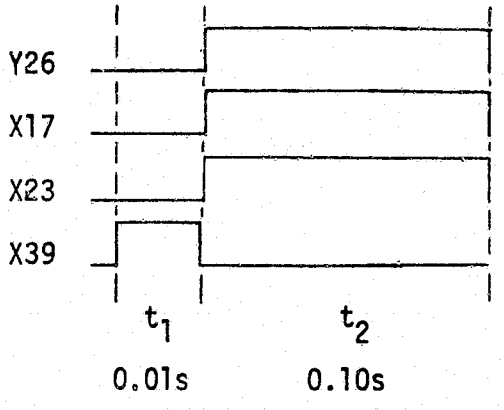
The A < B tse is available in register R3.
The A > B tse is available in register R6.

instructions is provided in the Appendix. Within this routine, there are two types of instructions. The first type of instruction is called a tse instruction. A tse instruction is implemented by sequencing one-bit control signals directly to the active tse logic devices. The second type of instruction is the control instruction. Control instructions do not directly affect the tse data, but sequence the program to assure proper processing. A more detailed discussion of tse computer control is found in Chapter V.

The threshold/compare operation is initialized by loading the digitized tse word into the A accumulator. The threshold tse word is then loaded into the B accumulator. The program then sequences through a number of iterations equal to the number of significant tses of the digitized tse words. Proper control timing for representative instructions needed in the threshold/compare operation is shown in Figure 3.7. These control timing diagrams are for the ALU structure of Figure 3.1 (page 17). Also, an example of the tse data manipulation is shown in Figure 3.8. Note that all comparison operations (A < B, A > B, A > B, A < B) are user-programmable. With the simple ALU of Figure 3.1 (page 17), the threshold/compare operation requires 320 tse logic device delays per iteration. Assuming a propagation delay of 5 milliseconds per tse gate, the threshold/compare operation requires 1.6 seconds per iteration. For a typical tse word length of 6 tses, the result tses are available in 9.6 seconds.

A hardware threshold/compare network of the type shown in Figure 3.9 may be added to the simple ALU structure to improve performance of the operation. The additional hardware cost of this

TRNN B, R4: TRANSFERS THE COMPLEMENT OF THE CONTENTS OF THE LEAST SIGNIFICANT tse POSITION OF THE B ACCUMULATOR TO LATCH R4.



LAND A, R4, R5: FORMS THE LOGICAL AND OF THE CONTENTS OF THE LEAST SIGNIFICANT tse POSITION OF THE A ACCUMULATOR AND LATCH R4, AND STORES THE RESULTING tse IN LATCH R5.

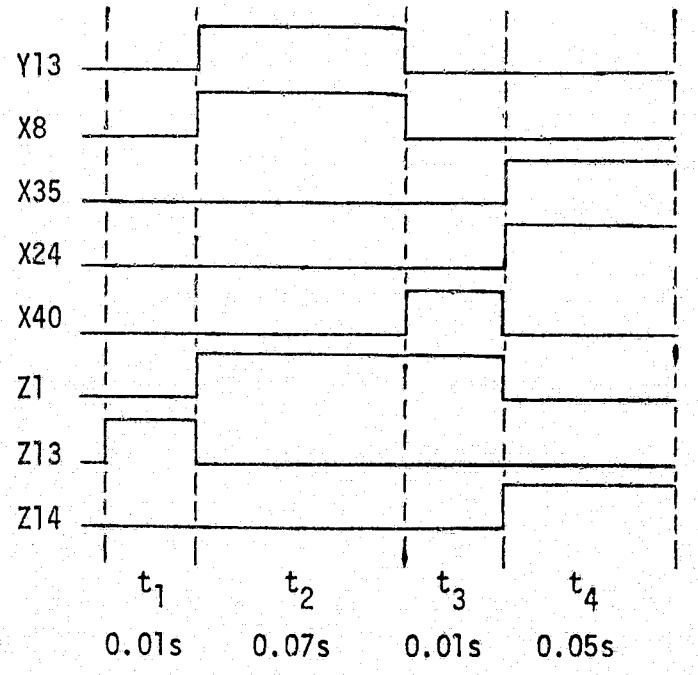


FIGURE 3.7. Timing for Representative Instructions on Machine 1

3	4	5		0	1	1	1	0	0	1	0	1
5	4	5	=	1	1	1	0	0	0	1	0	1
6	5	4		1	1	1	1	0	0	0	1	0
	(A)				A_2			A_1			A_0	
4	5	6		1	1	1	0	0	1	0	1	0
3	4	6	=	0	1	1	1	0	1	1	0	0
5	5	3		1	1	0	0	0	1	1	1	1
	(B)				B_2			B_1			B_0	

AFTER INSTRUCTIONS 1-4,

1	0	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
	(R1)			(R2)			(R3)	
1	0	1	1	0	1	0	0	0
0	1	1	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0
	(R4)			(R5)			(R6)	

FIGURE 3.8. Example of Threshold/Compare Operation

AFTER INSTRUCTIONS 5,6 R2, R3, R4, R5 REMAIN UNCHANGED

0	0	0	1	0	1
0	0	0	0	0	1
0	0	0	0	0	0
	(R1)			(R6)	

AFTER INSTRUCTIONS 7-9, R3, R5, R6 REMAIN UNCHANGED

0	1	0	0	1	0	0	1	0
0	1	0	0	0	0	1	1	0
1	0	1	1	0	1	1	1	1
	(R1)			(R2)			(R4)	

AFTER INSTRUCTIONS 10, 11 R1, R2, R5, R6 REMAIN UNCHANGED

0	1	0	0	0	0
0	0	0	0	0	0
1	0	1	0	0	0
	(R3)			(R4)	

AFTER THREE ITERATIONS OF THE ABOVE, THE RESULTS ARE:

1	1	1	0	0	0
0	0	1	1	0	0
0	0	0	1	0	1
	(A < B)			(A > B)	

FIGURE 3.8. (continued)

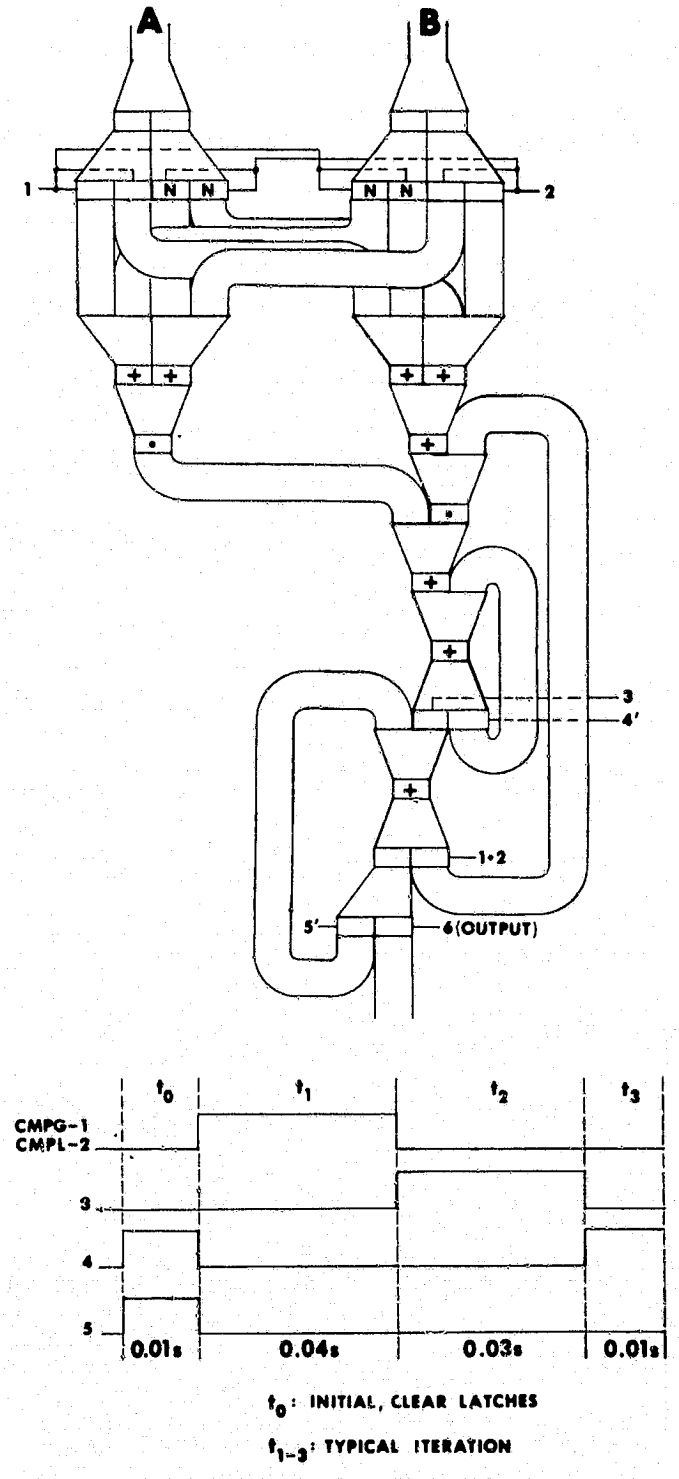


FIGURE 3.9. Hardward Threshold/Compare Network

network is 40 A + 24 B + 16 C. However, one iteration of the threshold/compare operation now requires only 42 tse gate delays or 0.21 seconds per iteration. For the typical tse word length of 6 significant tses, the result is now available in 1.26 seconds. The hardware network has allowed a decrease in computation time of approximately 7.6 times that of the software approach. This network will be advantageous in later arithmetic and tse operations.

Add/Subtract. In order to perform add/subtract operations, the tse word structure must be defined. The proposed tse word must contain a sign tse and N magnitude tses. The positions of the sign tse which are characterized by the absence of light (0) represent a positive data word, while those positions characterized by the presence of light (1) represent a negative data word. The sign tses must be directly accessible to the ALU, without altering the accumulator organization as right-shift registers. This accessibility will afford a computation savings. For ease of computation, a two's complement representation is chosen instead of the magnitude representation. The two's complement representation allows less complexity in the sign control problem than a one's complement representation.

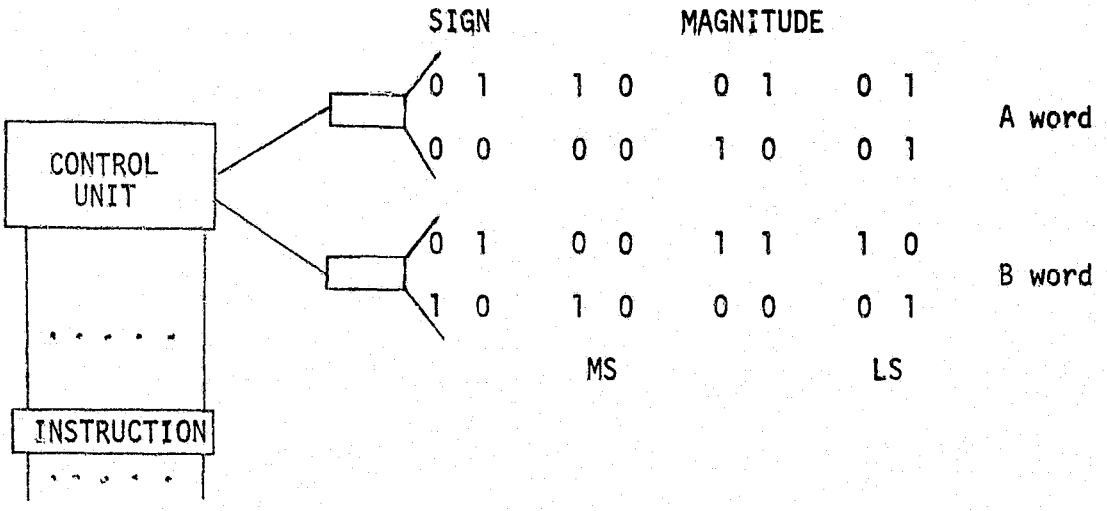
When the add or subtract instruction is read by the control unit, the sign tses are routed through contractors located at the output of the sign positions to the control unit. The contractor device was introduced in Chapter II. The output of the contractor is a single-bit signal which is logical 1 if and only if there is at least one

position of the tse that is logical 1. This single-bit output controls the logical disposition of a flag bit in the control unit. By checking this flag bit when an add or subtract instruction is received, the control unit determines whether the tse data word requires two's complement representation. Another control flag is used to monitor the tse word representation. When the tse word has been complemented, the flag is set; when the tse word is in magnitude form, the flag is cleared. An example of this computation is shown in Figure 3.10.

To better illustrate the add/subtract and two's complement operations, algorithms for implementing these operations are shown in Figure 3.11. These algorithms are converted to software form for the ALU of Figure 3.1 (page 17) in Tables 2 and 3. Computation times for these routines are 188 gate delays/0.94 seconds per iteration and 180 gate delays/0.90 seconds per iteration, respectively.

Due to the computational methods required for the two's complement and add/subtract operations, there exists an interrelationship which allows a single hardware tse logic network that will perform both operations in a minimum amount of time. This network is shown in Figure 3.12. Using this network, the add/subtract routine requires 44 gate delays/0.22 seconds per iteration and the two's complement routine 40 gate delays/0.20 seconds per iteration.

There are many possible versions of tse logic networks to implement the threshold/compare, add/subtract, and two's complement operations. However, the above networks have been introduced because of an interrelationship which exists between these three operations and their corresponding hardware implementations. Because of this



TWO'S COMPLEMENT REPRESENTATION:

SIGN

0 1 1 1 0 0 0 1

0 0 0 0 1 0 0 1

0 1 0 1 1 1 1 0

1 0 1 0 0 0 0 1

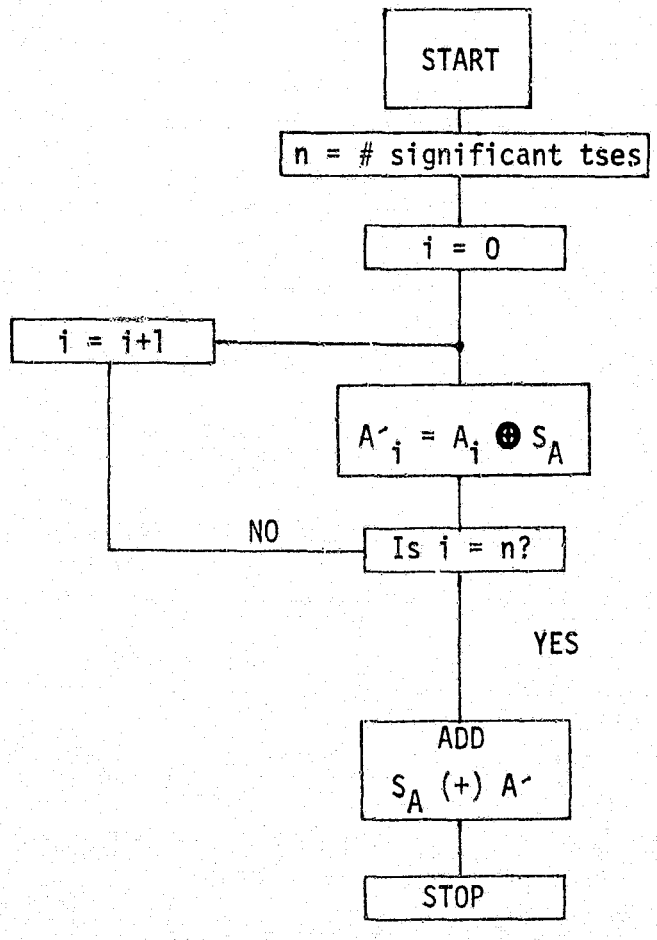
For positions in which the sign tse is 0, the magnitude remains unchanged.

For positions in which the sign tse is 1, the magnitude is complemented and the sign tse is added to it.

FIGURE 3.10. Example of Two's Complement Operation

$S_A = A_{n+1}$ A_n A_{n-1} ... A_0 $S_B = B_{m+1}$ B_m B_{m-1} ... B_0

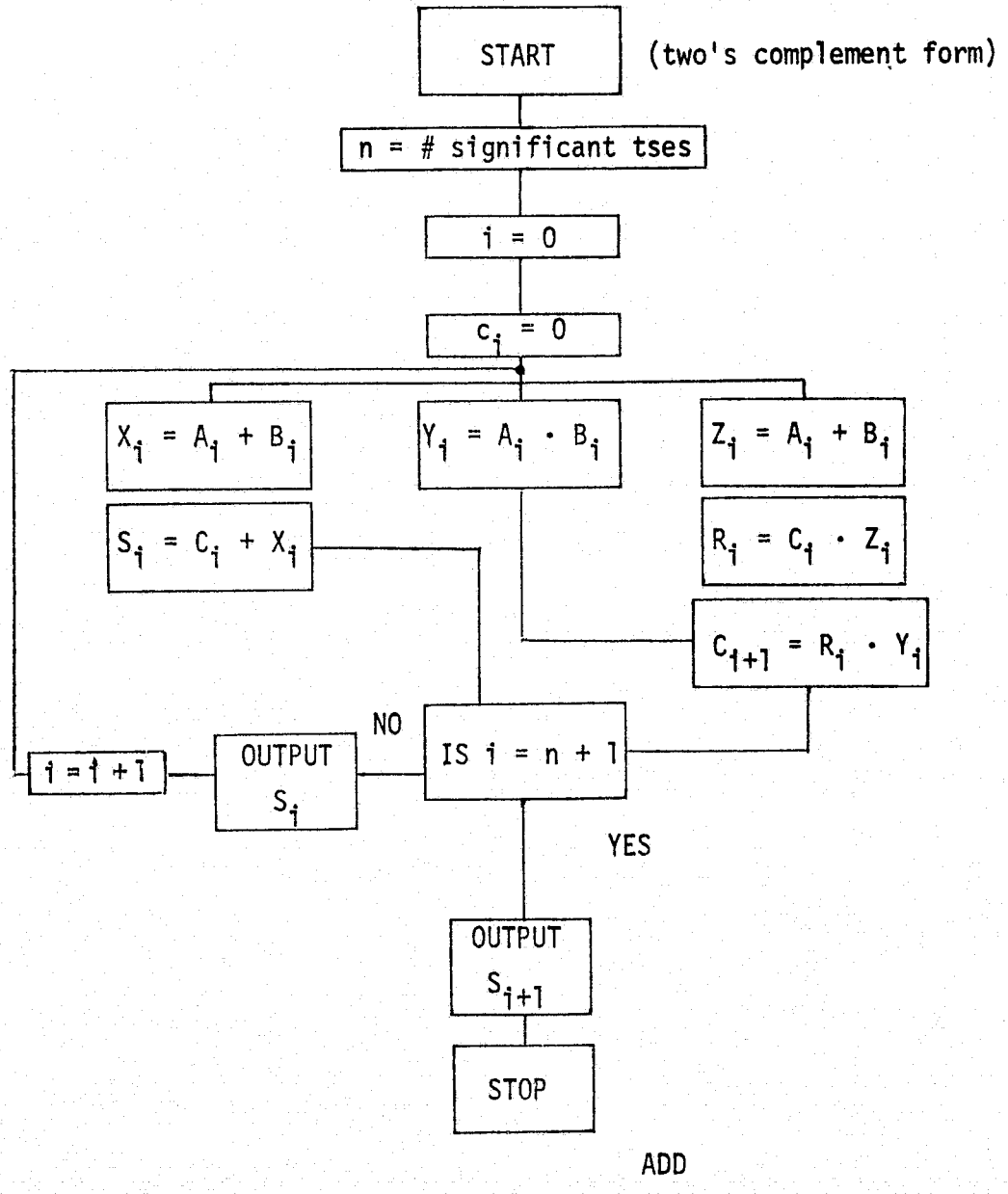
tse WORD REPRESENTATION



TWO'S COMPLEMENT

(a)

FIGURE 3.11. Add/Subtract and Two's Complement Algorithms



(b)

FIGURE 3.11. (continued)

TABLE 2

ADD/SUBTRACT SOFTWARE FOR THE MACHINE OF FIGURE 3.1

(See Appendix for Complete Instruction Set Definition)

ADDU N

NUMBER	INSTRUCTION	TYPE	COMMENTS
1	CONT N+2	CONTROL	SETS ITERATION COUNTER TO N+2
2	LAND A, B, R2	tse	
3	LORE A, B, R4	tse	
4	LAND R3, R4, R5	tse	
5	EXOR A, B, R4	tse	
6	SHIA 1	tse	
7	CLRC S_A	tse	
8	EXOR R3, R4, S_A	tse	
9	SHIB 1	tse	
10	DCR	CONTROL	
11	JNZ 2	CONTROL	JUMPS TO INSTRUCTION NUMBER 2 ON NON-ZERO COUNTER
12	RET	CONTROL	

The result is available in the A accumulator.
(Two's complement form)

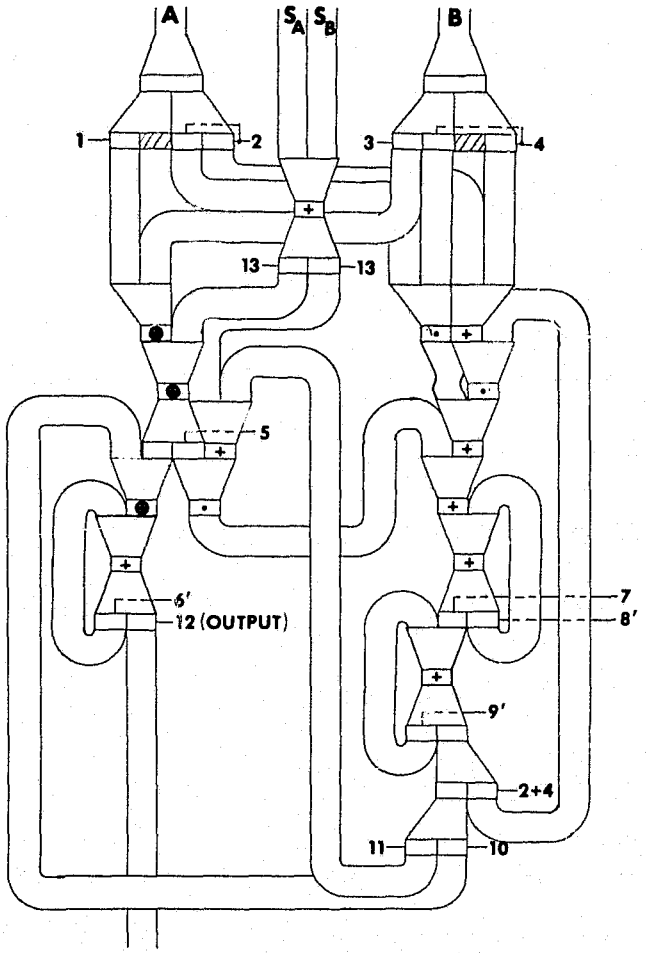
TABLE 3

TWO'S COMPLEMENT SOFTWARE FOR THE MACHINE OF FIGURE 3.1

(See Appendix for Complete Instruction Definition)

TCMA (Two's complement of the A accumulator)

NUMBER	INSTRUCTION	TYPE	COMMENTS
1	LORE S_A, 0, R6	tse	TRANSFERS THE S_A tse TO REGISTER R6
2	CONT N+1	CONTROL	
3	EXOR A, R6, -	tse	NO DESTINATION
4	SHFA 1	tse	
5	CLRC A_{MS}	tse	
6	MOVE A	tse	
7	DCR	CONTROL	
8	JNZ 3	CONTROL	JUMPS TO INSTRUCTION NUMBER 3 ON NON-ZERO COUNTER
9	CONT N+1	CONTROL	
10	EXOR A, R6, R1	tse	
11	LAND A, R6, R6	tse	
12	SHFA 1	tse	
13	CLRC A_{MS}	tse	
14	TRAN R1	tse	
15	DCR	CONTROL	
16	JNZ 10	CONTROL	JUMPS TO INSTRUCTION NUMBER 10 ON NON-ZERO COUNTER
17	RET	CONTROL	



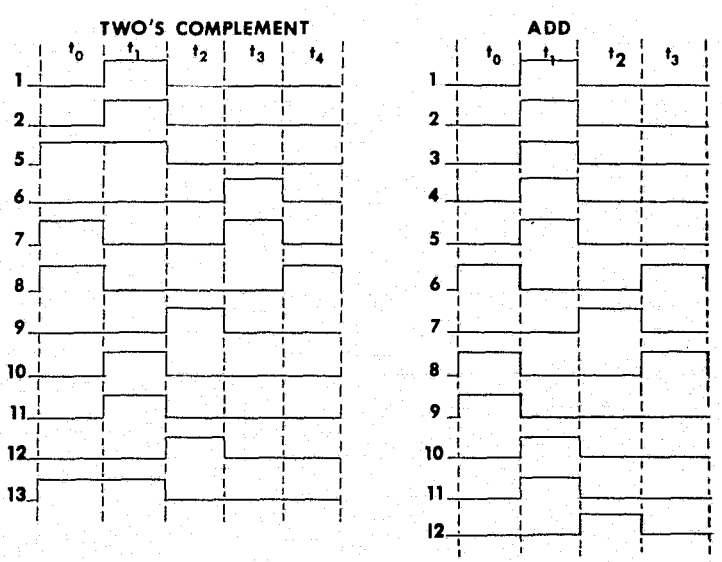


FIGURE 3.12. Hardward Add/Subtract/Two's Complement Network

interrelationship, there is no need to incorporate the threshold network of Figure 3.9 (page 31) into the tse processor; the threshold/compare operation may be implemented with the network of Figure 3.12 as well as the add/subtract and two's complement operations. For applications in which a fast threshold/compare operation is necessary, this results in a considerable hardware savings. There are some modifications to the network of Figure 3.12 that are necessary. For maximum flexibility, all four comparison operations should be available, and will be represented by the following mneumonics:

THRG: (A greater than B).

THRL: (A less than B).

THLE: (A less than or equal to B).

THGE: (A greater than or equal to B).

Since THLE is equal to NOT(THRG), and THGE is equal to NOT(THRL), only THRG and THRL need be generated. Table 4 is a summation of instructions needed to perform the threshold/compare, add/subtract, and two's complement operations on the modified processing unit of Figure 3.13. Table 4 is given in program type format. The corresponding timing diagrams for these operations on the processor of Figure 3.13 are shown in Figures 3.14, 3.15, and 3.16.

Square of an n-tse data word. The squaring operation is required in the computation of the magnitude of the gradient of an image. Due to the size and power requirements of the tse components, a hardware approach to the squaring operation is clearly not feasible. Also, algorithms exist which require only those operations already

TABLE 4

PROGRAM INSTRUCTION SUMMARY FOR THRESHOLD/COMPARE
ADD/SUBTRACT AND TWO'S COMPLEMENT OPERATIONS
FOR THE PROCESSOR OF FIGURE 3.13

NUMBER	INSTRUCTION	TYPE	COMMENTS
THRL N			
1	CLER	tse	
2	CONT N+1	CONTROL	
3	CMPL	tse	ON RECEIPT OF THIS INSTRUCTION, CONTROL SEQUENCES A AND B THROUGH THE PROPER ALU PATHS
4	DCR	CONTROL	
5	JNZ 3	CONTROL	JUMPS TO INSTRUCTION NUMBER 3 ON NON-ZERO COUNTER
6	TRNS	tse	A < B tse IS IN THE OUTPUT LATCH OF THE ALU
7	RET	CONTROL	
*	MOVE -	tse	STEERS THE RESULT TO THE DESIGNATED REGISTER
THGE N			
	SAME AS THRL, EXCEPT FOR:		
**	MCME -	tse	STEERS THE COMPLEMENT OF THE RESULT TO THE DESIGNATED REGISTER

TABLE 4 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
THRG N			
SAME AS THRL, EXCEPT FOR:			
3*	CMPG	tse	
THLE N			
SAME AS THRL, EXCEPT FOR:			
3**	CMPG	tse	
***	MCME -	tse	
ADDU N			
1	CLER	tse	
2	CALL	CONTROL	CALLS A SUBROUTINE TO CHECK THE STATUS OF THE CONTRACTOR OUTPUTS ON S_A tse AND S_B tse
3	TCMA	tse	

TABLE 4 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
4	TCMB	tse	INSTRUCTIONS 3 AND 4 ARE EXECUTED ON COMMAND FROM THE SUBROUTINE OF INSTRUCTION NUMBER 2. IF NEITHER TCMA OR TCMB ARE NEEDED, THEN THE CONTROL SUBROUTINE OF INSTRUCTION 2 JUMPS TO INSTRUCTION NUMBER 5
5	CONT N+1	CONTROL	
6	ADSU A, B	tse	
7	MOVE A	CONTROL	
8	DCR		
9	JNZ 6	CONTROL	JUMPS TO INSTRUCTION NUMBER 6 ON NON-ZERO COUNTER
10	RET	CONTROL	

SUBU N

SAME AS ABOVE, EXCEPT FOR:

INSERT AFTER INSTRUCTION 1 ABOVE:

(a)	LORE S_B, 0	tse	NO DESTINATION
(b)	MCME S_B	tse	COMPLEMENTS THE SIGN OF B ACCUMULATOR

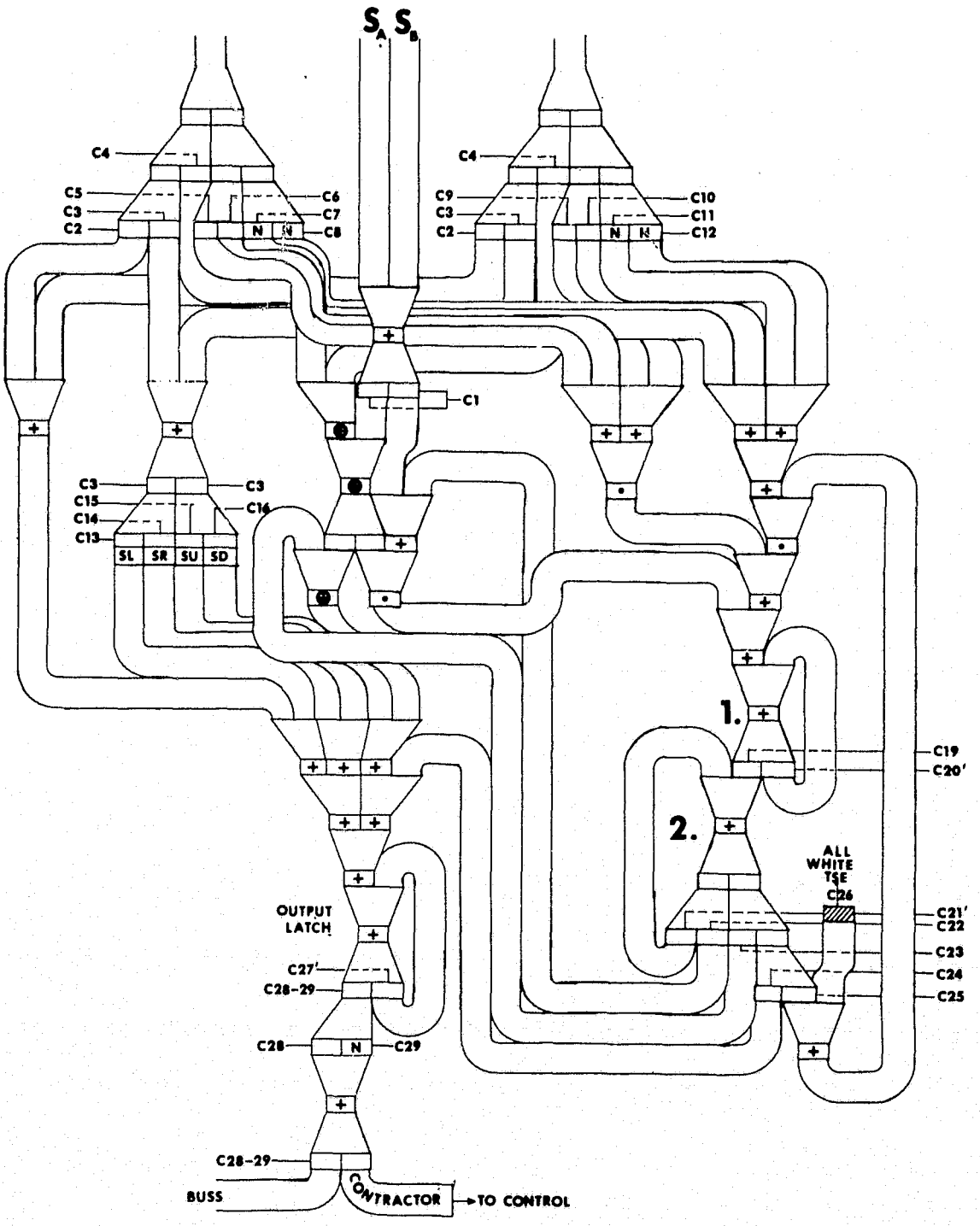
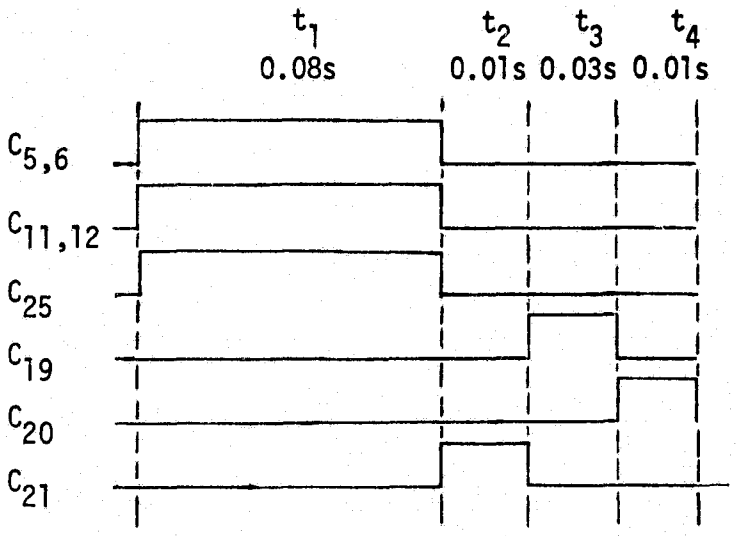


FIGURE 3.13. Modified ALU Organization (Machine 2)

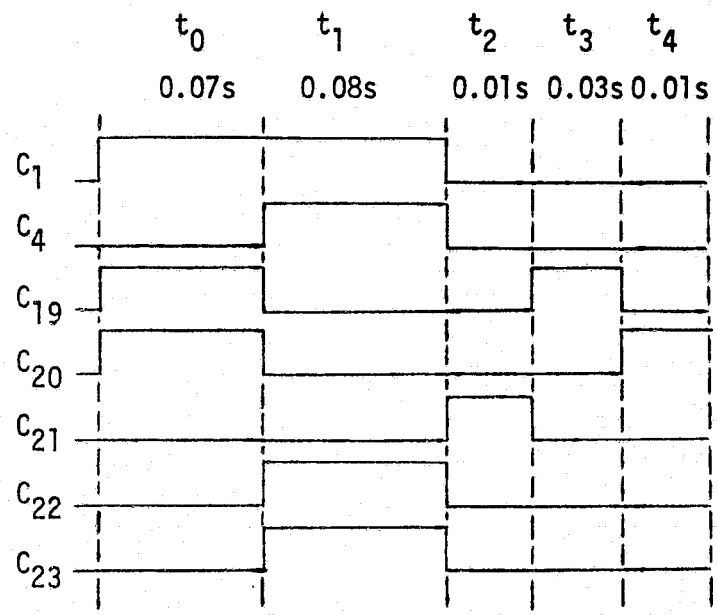
CMPG



CMPL - SAME AS ABOVE, EXCEPT SUBSTITUTE $C_{7,8}$ FOR $C_{5,6}$ AND $C_{9,10}$ FOR $C_{11,12}$.

FIGURE 3.14. Threshold/Compare Timing for Machine 2

ADSU



t_0 - Initialization

t_1-t_4 - Typical Iteration

FIGURE 3.15. Add/Subtract Timing for Machine 2

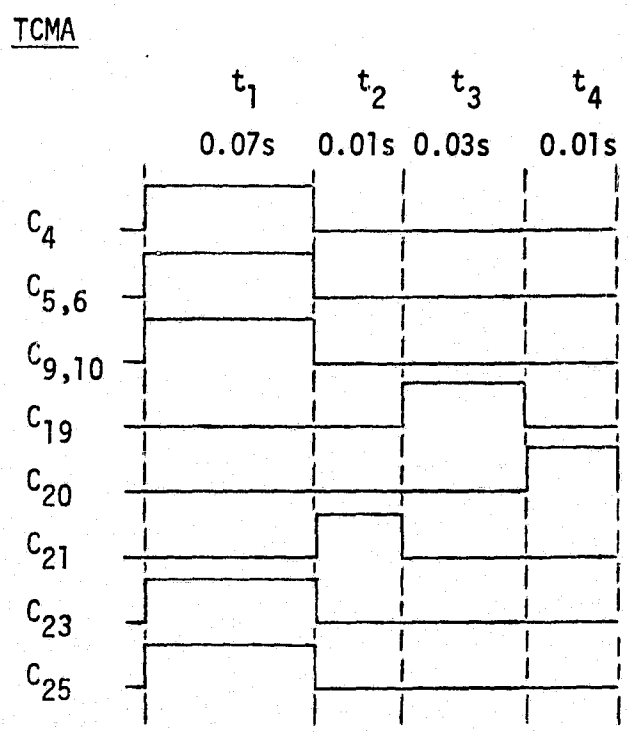
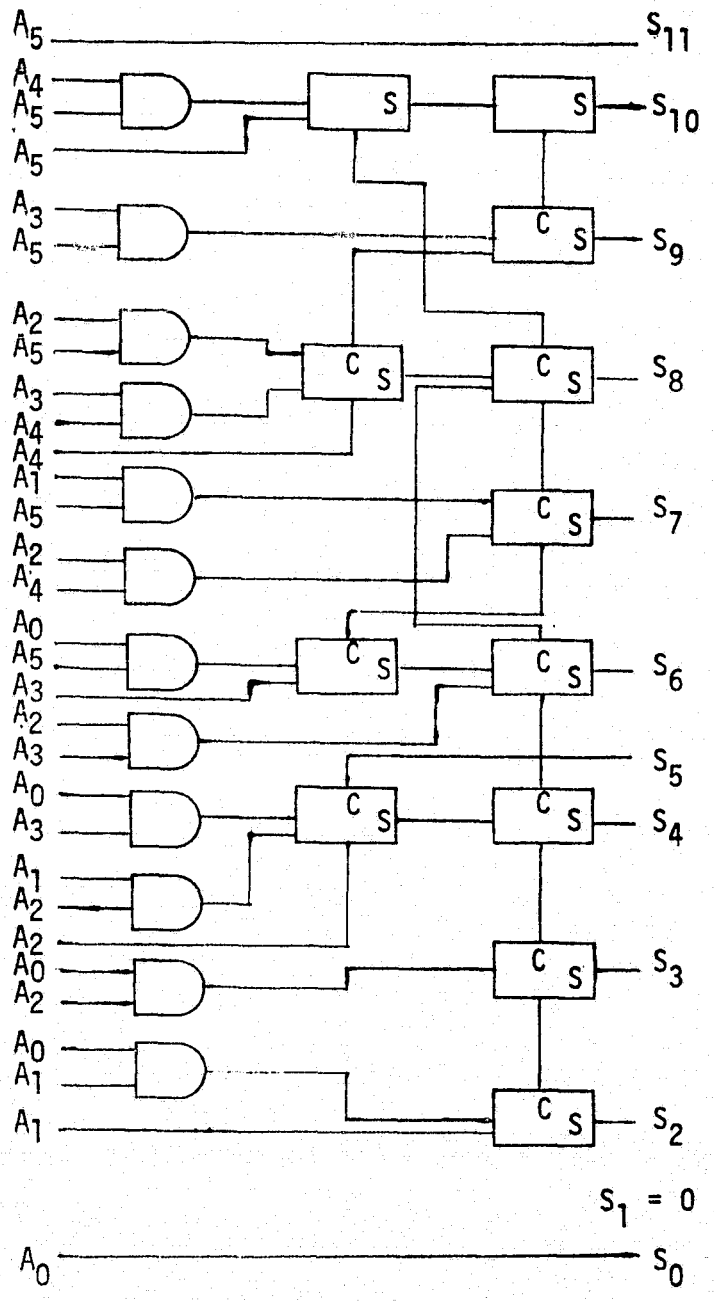


FIGURE 3.16. Two's Complement Timing for Machine 2

developed in the preceding sections to compute the square of an n-tse data word. One approach to the squaring problem would be to develop a software multiplication scheme and compute the tse-word "A" multiplied by itself. However, this approach would require many addition operations and extra storage to accomodate the partial products. A more elegant algorithm which utilizes the identity of the multiplicand and the multiplier has been developed [2] and will significantly reduce storage and computation time requirements. A conventional logic analogy of this algorithm is shown in Figure 3.17. Note that the algorithm is not an iterative technique.

A program to implement this algorithm is shown in Table 5, and may be performed on either the programmed machine (machine 1) of Figure 3.1 (page 17), or machine 2 of Figure 3.13. To implement this program, the specific number of significant tses in the data word to be squared is assumed to be 6. With this program, the square of a 6-significant-tse word is available in 3250 gate delays/16.25 seconds on machine 1, or 2000 gate delays/10 seconds on machine 2. The performance of the program may be gauged by the relative computation times of the squaring operation and the addition operation (since the square of a 6-tse-word is represented by a 12-tse word, a comparison to 12-tse addition is appropriate). The squaring operation requires approximately 4 times the computation time of the addition operation, which by conventional logic standards is very good indeed.



$$(A_5A_4A_3A_2A_1A_0)^2 = S_{11}S_{10}S_9S_8S_7S_6S_5S_4S_3S_2S_1S_0$$

FIGURE 3.17. Squaring Algorithm Shown in Conventional Logic

TABLE 5

PROGRAM TO COMPUTE THE SQUARE OF A 6-TSE DATA WORD LOCATED IN A ACCUMULATOR

SQUA $(S_{2048}\ S_{1024}\ S_{512}\ S_{256}\ S_{128}\ S_{64}\ S_{32}\ S_{16}\ S_8\ S_4\ S_2\ S_1)$

NUMBER	INSTRUCTION	TYPE	COMMENTS
1	CLER	tse	
2	TRAN A, M	tse	$M \leftarrow S_1$
3	TRAN A, R4	tse	
4	SHFA 1	tse	
5	LAND A, R4, R5	tse	
6	ADSU A, R5	tse	
7	SHFM 2	tse	$M \leftarrow S_2$
8	MOVE M	tse	$M \leftarrow S_4$
9	SHFM 1	tse	
10	TRAN A, R6	tse	
11	SHFA 1	tse	
12	LAND A, R4, R5	tse	
13	ADSU R5, 0	tse	
14	MOVE M	tse	$M \leftarrow S_8$
15	SHFM 1	tse	
16	LAND A, R6, R6	tse	
17	TRAN R6, R5	tse	
18	LAND R5, A, R5	tse	
19	TRAN A, R1	tse	
20	LAND R1, R4, R4	tse	
21	SHFA 1	tse	
22	LAND A, R4, R1	tse	
23	LORE R1, R6, R1	tse	
24	ADSU R5, 0	tse	

TABLE 5 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
25	MOVE M	tse	$M \leftarrow S_{16}$
26	SHFM 1	tse	
27	TRAN R1, M	tse	$M \leftarrow S_{32}$
28	SHFM 1	tse	
29	TRAN A, R4	tse	
30	SHFA 2	tse	
31	TRAN A, R5	tse	
32	SHFA 1	tse	
33	LAND A, R5, R1	tse	
34	EXOR R1, R4, R2	tse	
35	LAND R1, R4, R3	tse	
36	SHFA 2	tse	
37	LAND A, R4, R5	tse	
38	ADSU R2, R5	tse	
39	MOVE M	tse	$M \leftarrow S_{64}$
40	TRNS	tse	
41	MOVE R1	tse	
42	LDCA R3	tse	LOADS THE CONTENTS OF REGISTER R3 INTO CARRY LATCH 2
43	TRAN A, R5	tse	
44	SHFA 2	tse	
45	TRAN A, R2	tse	
46	LAND R2, R5, R6	tse	
47	SHFA 1	tse	
48	TRAN A, R5	tse	
49	SHFA 2	tse	

TABLE 5 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
50	LAND A, R5, R3	tse	
51	ADSU R3, R6	tse	
52	MOVE M	tse	$M \leftarrow S_{128}$
53	SHFM 1	tse	
54	SHFA 1	tse	
55	TRAN A, R3	tse	
56	LAND R3, R5, R3	tse	
57	TRNN R4, R6	tse	
58	LAND R2, R6, R6	tse	
59	EXOR R3, R6, R6	tse	
60	LAND R2, R4, R4	tse	
61	SHFA 2	tse	
62	TRAN A, R5	tse	
63	LAND R3, R5, R3	tse	
64	LAND R3, R4, R3	tse	
65	ADSU R1, R6	tse	
66	MOVE M	tse	$M \leftarrow S_{256}$
67	SHFM 1	tse	
68	SHFA 1	tse	
69	TRAN A, R4	tse	
70	SHFA 4	tse	
71	LAND A, R4, R5	tse	
72	LAND R3, R5, R6	tse	
73	EXOR R3, R5, -	tse	NO DESTINATION
74	MOVE M	tse	$M \leftarrow S_{512}$
75	SHFM 1	tse	

TABLE 5 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
76	SHFA 1	tse	
77	LAND A, R4, R1	tse	
78	ADSU R1, R4	tse	
79	MOVE R1	tse	
80	EXOR R1, R6, -	tse	NO DESTINATION
81	MOVE M	tse	$M \leftarrow S_{1024}$
82	SHFM 1	tse	$M \leftarrow S_{2048}$
83	RET	CONTROL	

Square root. The square root operation is the most difficult of the tse arithmetic operations. As was the case with the squaring operation, a hardware network to implement the square root is not feasible. Any type of iterative technique to compute the square root must account for the fact that the tse word is composed of N^2 positions (i.e., N = 512, 1024, ...), some of which may converge to the square root faster than others. Therefore, the positions that have converged must be masked from further operations, while continuing to perform the algorithm on the positions which have not converged. One algorithm which achieves this is shown in Figure 3.18. The S register contains the tse data word for which the square root is desired. The R register will eventually contain the square root of S. The algorithm begins with the R register containing all black tses. Then, the R register is squared and subtracted from S. The result is compared to a preset threshold. Positions of $(S - R^2)$ less than or equal to the fixed threshold have converged. Positions of $(S - R^2)$ greater than the threshold have not converged. The threshold/compare operation generates the $S > R^2$ tse, which consists of 1's in positions for which S is greater than R^2 and 0's elsewhere. The $S > R^2$ tse is then added to the contents of the R register. This is equivalent to incrementing those positions of R which have not converged. The algorithm continues the iterations until the $S > R^2$ tse contains all 0's.

The above algorithm introduces some problem areas in the computation of the square root. Firstly, a fixed threshold value to

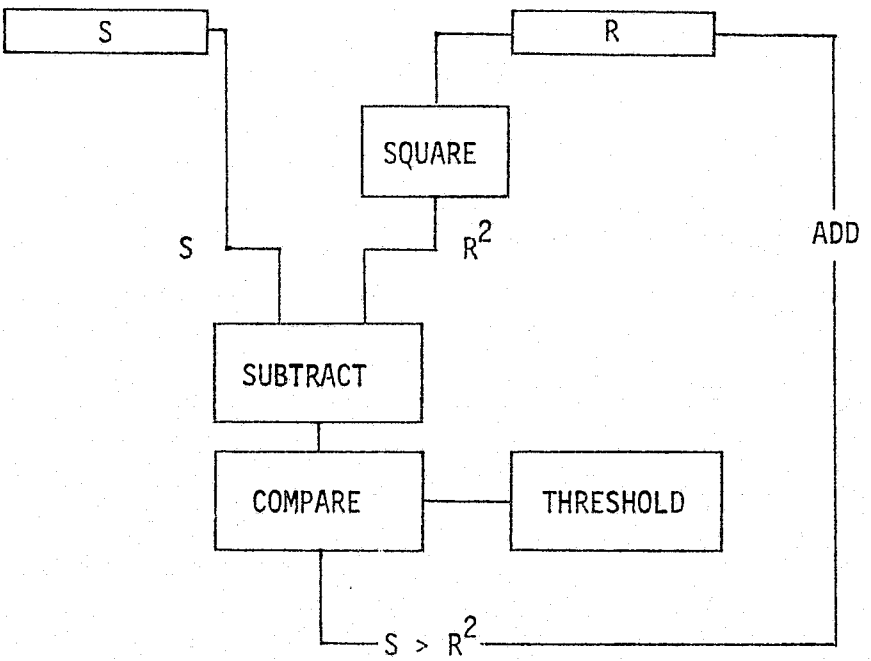


FIGURE 3.18. Square Root Algorithm One

test for convergence allows significant error in either the small number or large number positions when performing fixed-point computations. If the threshold is set to a small value, the possibility exists that the S register contains numbers in some positions which are large enough, that will not converge at all. This problem may be solved by either developing floating-point computations for the tse computer, or allowing the threshold to be changed with each iteration by tracking the relative magnitude of S in each position. However, since floating-point for the tse computer requires much more development and since the varying threshold introduces a more complex computation, the algorithm of Figure 3.18 is not feasible for the tse computer in its present stage of development.

A second method to compute the square root is based on the algorithm of Figure 3.19. This method is outlined in Table 6 in program form. The program of Table 6 will compute the square root of a 12 significant tse data word to the least significant integer. However, the algorithm may be extended to any degree of accuracy when floating-point is available. The result of the square root operation of Table 6 is available in 5780 gate delays/28.9 seconds on the machine of Figure 3.13 (page 44), or 20,950 gate delays/104.8 seconds on the machine of Figure 3.1 (page 17).

The results of the investigation of arithmetic operations on the tse computer are outlined in Table 7. These results clearly show the superiority of the organization of Figure 3.13 (page 44). For the discussion of tse operations in Chapter IV, only this organization will be considered.

$R^2 = S_{12}\ S_{11}\ S_{10}\ S_9\ S_8\ S_7\ S_6\ S_5\ S_4\ S_3\ S_2\ S_1$

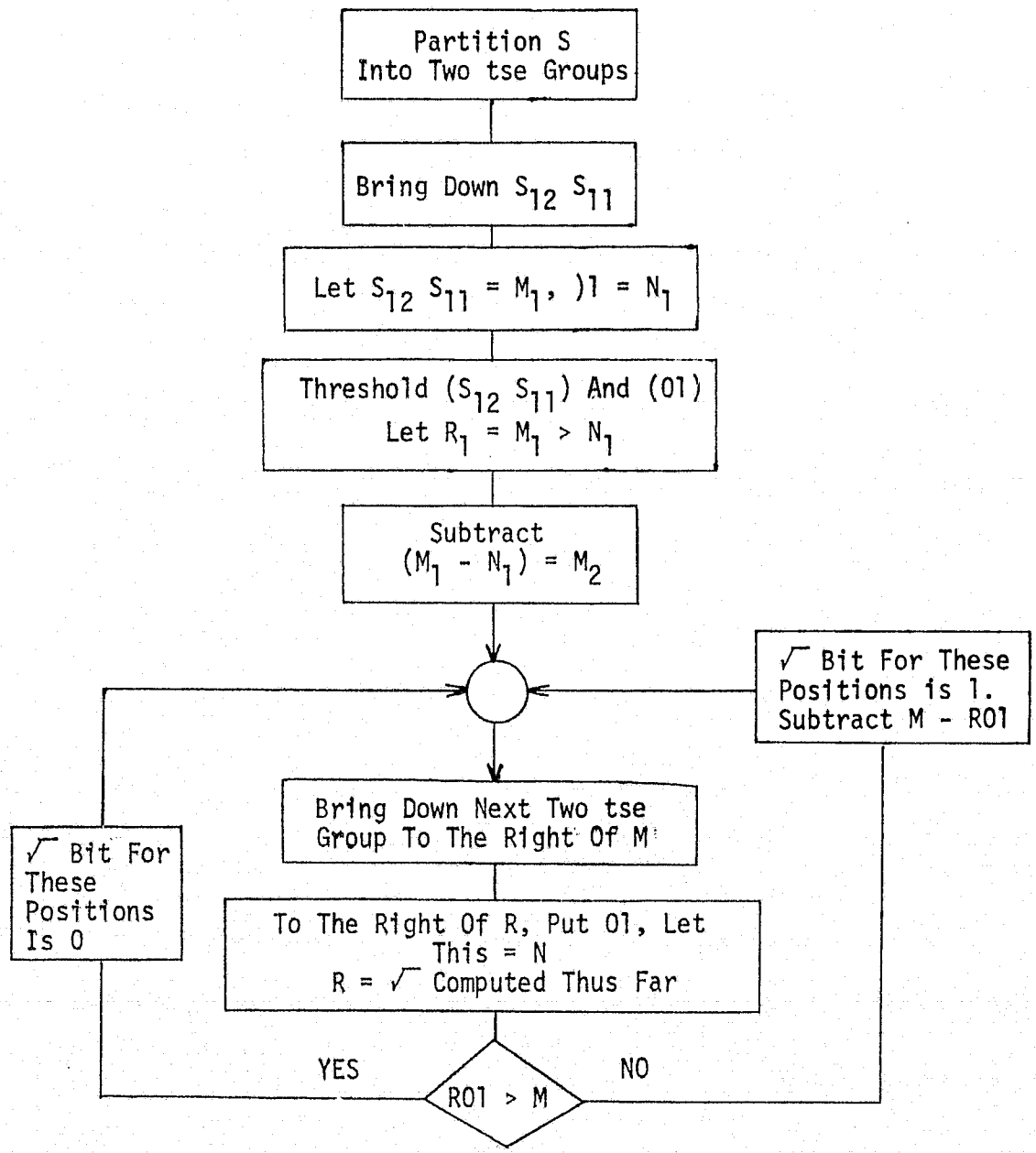


FIGURE 3.19. Square Root Algorithm Two

TABLE 6

PROGRAM TO COMPUTE THE SQUARE ROOT OF
A 12-TSE DATA WORD

SQUR $(S_{12}\ S_{11}\ S_{10}\ S_9\ S_8\ S_7\ S_6\ S_5\ S_4\ S_2\ S_1)$

(The word for which the square root is to be computed is assumed to be located in memory)

NUMBER	INSTRUCTION	TYPE	COMMENTS
1	CLER	tse	
2	CLRA	tse	
3	CLRB	tse	
4	TRAN M, R1	tse	TRANSFER THE CONTENTS OF THE MEMORY OUTPUT LATCH (S_{12}) TO REGISTER R1
5	TRAN M, R4	tse	TRANSFER THE CONTENTS OF THE MEMORY OUTPUT LATCH (S_{11}) TO REGISTER R4
6	LORE R1, R4, R6	tse	LOGICALLY AND THE CONTENTS OF R1 AND R4, AND PLACE THE RESULT IN R6. THE RESULT IN THE MOST SIGNIFICANT tse OF THE SQUARE ROOT WORD
7	TRNN R4, R5		TRANSFER THE COMPLEMENT OF R4 TO R5
8	LAND R1, R5, B		
9	SHFB 1		SHIFT THE MAGNITUDE OF B ACCUMULATOR RIGHT ONE POSITION
10	LAND R1, R4, B		
11	SHFB 8		SHIFT THE MAGNITUDE OF B ACCUMULATOR RIGHT EIGHT POSITIONS
12	TRAN M, B_{LS+1}		$B_{LS+1} \leftarrow S_{10}$

TABLE 6 (continued)

NUMBER	INSTRUCTION		TYPE	COMMENTS
13	TRAN	M, B_{LS}		$B_{LS} \leftarrow S_9$
14	TRAN	1, A		TRANSFER AN ALL WHITE tse INTO A
15	SHFA	2		
16	TRAN	R6, A		
17	SHFA	9		
18	THGE	4		THGE SUBROUTINE COMPARES A ACCUMULATOR AND B ACCUMULATOR AND GENERATES THE A B tse. THE SUBROUTINE WILL BE SEQUENCED 4 TIMES
19	MOVE	R9		R9 REGISTER NOW CONTAINS THE NEXT MOST SIGNIFICANT tse OF THE SQUARE ROOT
20	CLRA			
21	TRAN	1, A		
22	SHFA	1		
23	TRAN	1, A	tse	
24	SHFA	1	tse	
25	TRNN	R6, A	tse	
26	SHFA	1	tse	
27	TRAN	1, A	tse	
28	SHFA	8	tse	
29	ADDU	4	tse	THE A AND B WORD ARE ADDED. THE ADDU ROUTINE IS PERFORMED 4 TIMES WITH THE RESULT BEING LOADED BACK INTO THE A ACCUMULATOR
30	TRNN	R9, R1	tse	

TABLE 6 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
31	CONT 5	CONTROL	tse INSTRUCTION WHICH REFERENCES A CONTROL SUBROUTINE TO SET UP AN ITERATION COUNTER. THE SECOND BYTE CONTAINS THE NUMBER OF ITERATIONS TO BE PERFORMED, PLUS 1.
32	LAND R1, B, R2		
33	LAND R9, A, R5		
34	LORE R2, R5, B		
35	SHFB 1		
36	SHFA 1		
37	CLRC B_{MS}		
38	DCR	CONTROL	DECREMENTS COUNTER
39	JNZ 32	CONTROL	JUMPS TO INSTRUCTION NUMBER 32 WHEN COUNTER IS NON-ZERO
40	SHFB 6		
41	TRAN M, B_{LS+1}		$B_{LS+1} \leftarrow S_8$
42	TRAN M, B_{LS}		$B_{LS} \leftarrow S_7$
43	CLRA		
44	TRAN 1, A		
45	SHFA 2		
46	TRAN R9, A		
47	SHFA 1		
48	TRAN R6, A		
49	SHFA 8		
50	THGE 6		
51	MOVE R10		REGISTER R10 NOW CONTAINS THE NEXT MOST SIGNIFICANT tse OF THE SQUARE ROOT

TABLE 6 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
52	CLRA		
53	CLER		
54	TRAN 1, A		
55	SHFA 1		
56	TRAN 1, A		
57	SHFA 1		
58	TRNN R9, A		
59	SHFA 1		
60	TRNN R6, A		
61	SHFA 1		
62	TRAN 1, A		
63	SHFA 1		
64	TRAN 1, A		
65	SHFA 6		
66	ADDU 6		
67	TRNN R10, R1		
68	CONT 7	CONTROL	
69	LAND R1, B, R2		
70	LAND R10, A, R5		
71	LORE R2, R5, B		
72	SHFB 1		
73	SHFA 1		
74	CLRC B_{MS}		
75	DCR	CONTROL	
76	JNZ 68	CONTROL	JUMP TO INSTRUCTION NUMBER 68 ON NON-ZERO COUNTER
77	SHFB 4		
78	TRAN M, B_{LS+1}		$B_{LS+1} \leftarrow S_6$

TABLE 6 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
79	TRAN M, B_{LS}		$B_{LS} \leftarrow S_5$
80	CLRA		
81	CLER		
82	TRAN 1, A		
83	SHFA 2		
84	TRAN R10, A		
85	SHFA 1		
86	TRAN R9, A		
87	SHFA 1		
88	TRAN R6, A		
89	SHFA 7		
90	THGE 8		
91	MOVE R8		R8 REGISTER NOW CONTAINS THE NEXT MOST SIGNIFICANT tse OF THE SQUARE ROOT
92	CLRA		
93	CLER		
94	TRAN 1, A		
95	SHFA 1		
96	TRAN 1, A		
97	SHFA 1		
98	TRNN R10, A		
99	SHFA 1		
100	TRNN R9, A		
101	SHFA 1		
102	TRNN R6, A		
103	SHFA 1		
104	TRAN 1, A		

TABLE 6 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
105	SHFA 1		
106	TRAN 1, A		
107	SHFA 1		
108	TRAN 1, A		
109	SHFA 4		
110	ADDU 8		
111	TRNN R8, R1		
112	CONT 9	CONTROL	
113	LAND R1, B, R2		
114	LAND R8, A, R5		
115	SHFB 1		
116	SHFA 1		
117	CLRC B_{MS}		
118	DCR	CONTROL	
119	JNZ 111	CONTROL	JUMP TO INSTRUCTION NUMBER 111 ON NON-ZERO COUNTER
120	SHFB 2		
121	TRAN M, B_{LS+1}		$B_{LS+1} \leftarrow A_4$
122	TRAN M, B_{LS}		$B_{LS} \leftarrow S_3$
123	CLRA		
124	CLER		
125	TRAN 1, A		
126	SHFA 2		
127	TRAN R8, A		
128	SHFA 1		
129	TRAN R10, A		

TABLE 6 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
130	SHFA 1		
131	TRAN R9, A		
132	SHFA 1		
133	TRAN R6, A		
134	SHFA 6		
135	THGE 10		
136	MOVE R7		R7 REGISTER NOW CONTAINS THE NEXT MOST SIGNIFICANT tse OF THE SQUARE ROOT
137	CLRA		
138	CLER		
139	TRAN 1, A		
140	SHFA 1		
141	TRAN 1, A		
142	SHFA 1		
143	TRNN R8, A		
144	SHFA 1		
145	TRNN R10, A		
146	SHFA 1		
147	TRNN R9, A		
148	SHFA 1		
149	TRNN R6, A		
150	SHFA 1		
151	TRAN 1, A		
152	SHFA 1		
153	TRAN 1, A		
154	SHFA 1		
155	TRAN 1, A		

TABLE 6 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
156	SHFA 1		
157	TRAN 1, A		
158	SHFA 2		
159	ADDU 8		
160	TRNN R7, R1		
161	CONT 11	CONTROL	
162	LAND R1, B, R2		
163	LAND R7, A, R5		
164	LORE R2, R5, B		
165	SHFB 1		
166	SHFA 1		
167	CLRC B_{MS}		
168	DCR	CONTROL	
169	JNZ 160	CONTROL	JUMP TO INSTRUCTION NUMBER 160 ON NON-ZERO COUNTER
170	TRAN M, B_{LS+1}		$B_{LS+1} \leftarrow S_2$
171	TRAN M, B_{LS}		$B_{LS} \leftarrow S_1$
172	CLRA		
173	CLER		
174	TRAN 1, A		
175	SHFA 2		
176	TRAN R7, A		
177	SHFA 1		
178	TRAN R8, A		
179	SHFA 1		
180	TRAN R10, A		

TABLE 6 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
181	SHFA 1		
182	TRAN R9, A		
183	SHFA 1		
184	TRAN R6, A		
185	SHFA 5		
186	THGE 12		
187	SHFA 1		A NOW CONTAINS THE SQUARE ROOT
188	CLRC A_{MS}		
189	SHFA 1		
190	MOVE A		
191	SHFA 11		

TABLE 7

COMPARISON OF THE TWO TSE COMPUTER ORGANIZATIONS

ORGANIZATION	COST	COMPUTATION TIMES PER ITERATION (GATE DELAYS/SECONDS)		MAXIMUM DATA RATE (tses/MINUTE)	PEAK POWER CONSUMPTION (WATTS)
Machine 1 (Figure 3.1)	39A + 23B + 17C	Threshold:	320/1.6	37.5	117
		Add/Subtract:	188/0.94	63.8	
		Square:	3250/16.25 } TOTAL	3.7 words	
		Square Root:	20,950/104.8 } TOTAL	0.6 words	
Machine 2 (Figure 3.13)	88A + 52B + 28C	Threshold:	42/0.21	286	264
		Add/Subtract:	44/0.22	273	
		Square:	2000/10.0 } TOTAL	6 words	
		Square Root:	5780/28.9 } TOTAL	2 words	

NOTE: Power consumption pertains only to the ALU section of the tse computer.

CHAPTER IV

TSE OPERATIONS

Tse operations are subroutined instructions for the tse computer. With the instruction set discussed in Chapter III and listed in the Appendix, one is now able to compute the desired topological information. This instruction set has been designed to provide maximum processing flexibility, and the following tse operations are only a representative set of all the operations possible.

Global maxima. The global maxima operation extracts the area(s) of the input image for which the digitized parameter has the greatest magnitude. Note that the global maxima operation is not an absolute maxima operation in a mathematical sense, since an absolute maxima operation requires comparison of each element with its nearest neighbors. The global maxima operation, as well as all others, may be performed on either the simple machine 1, or machine 2. In either case, the control unit must be capable of program sequencing on intermediate results from the tse processor.

Conditional control is accomplished by the contractor tse device located at the output of the ALU. The contractor output controls the disposition of one of the tse processor status bits which are available to the control unit (see Chapter V on control). The logical disposition of this bit allows the control unit to determine if an intermediate result is logical 0 (black). On this determination, control may sequence the program accordingly.

A routine to compute the global maxima operation is listed in Table 8. Also, an example of this operation is shown in Figure 4.1. The computation time for the global maxima on either of the organizations of Chapter II is approximately the same and is equal to 0.15 seconds for each significant tse of the data word. A global minima operation may also be performed by first complementing the tse data word and then performing the routine of Table 8.

Local maxima/minima. The extraction of local maxima/minima requires the comparison of each element in the image array with each of its eight nearest neighbors, which are shown in Figure 4.2. This comparison is accomplished by means of the thresholding operation. Figure 4.2 shows that proper manipulation of the slide instructions may be used to achieve this comparison. Table 9 contains a routine to perform the local maxima operation. Due to the flexibility of the thresholding operation, either maxima or minima, allowing or disallowing plateau regions, is a programmable option to the user. Figure 4.3 illustrates the generation of the tse neighborhood planes and the computation of the local maxima tse. Note, in Figure 4.3, that positions on the bordering edges of the input image are not eligible to be positions of local maxima/minima since they do not have the required eight neighbors. The local maxima operation requires 11.34 seconds per iteration on machine 1, or 1.53 seconds per iteration on machine 2.

First and second partial derivatives. In many image processing applications, the locations of the boundaries of certain regions are

TABLE 8

GLOBAL MAXIMA ROUTINE

NUMBER	INSTRUCTION	TYPE	COMMENTS
			tse WORD FOR WHICH THE GLOBAL MAXIMA INFORMATION IS DESIRED, IS ASSUMED TO BE IN THE A ACCUMULATOR
1	TRAN A, R1	tse	
2	SHFA 1	tse	
3	CHEK 1	tse	THIS INSTRUCTION SETS UP A CONTROL SUBROUTINE WHICH INPUTS THE CONTRACTOR BIT, AND IF ZERO, JUMPS PROGRAM CONTROL BACK TO INSTRUCTION 1. IF THIS CHEK INSTRUCTION IS EXECUTED A NUMBER OF TIMES EQUAL TO THE NUMBER OF tses OF THE A WORD, THEN THE ROUTINE HALTS.
4	TRAN A, R4	tse	
5	SHFA 1	tse	
6	CHEK 4	tse	
7	LAND R1, R4, R5	tse	
8	CHAK 10	tse	
9	GO TO 17	CONTROL	
10	LAND R4, A, R5	tse	
11	SHFA 1	tse	
12	CHAK 14	tse	
13	GO TO 17	CONTROL	
14	TRAN R1, R6	tse	
15	LAND R6, A, R5	tse	

TABLE 8 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
16	SHFA 1	tse	
17	LAND A, R5, R6	tse	
18	SHFA 1	tse	
19	CHAK 1	tse	
20	TRAN R6, R5	tse	
21	GO TO 17	tse	GLOBAL MAXIMA tse IS IN R5
22	RETURN	CONTROL	

A					A_3				A_2				A_1			
0	1	2	3		0	0	0	0	0	0	1	1	0	1	0	1
2	2	4	3		0	0	1	0	1	1	0	1	0	0	0	1
				=												
4	3	5	4		1	0	1	1	0	1	0	0	0	1	1	0
5	4	6	6		1	1	1	1	0	0	1	1	1	0	0	0

$(A_2)\cdot(A_1)=(R5)_1$				$(R5)_1\cdot(A_3)=(R5)_2$				$(A_3)\cdot(A_2)=(R5)_3$			
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1
$(R5)_1$				$(R5)_2$				$(R5)_3$			

FIGURE 4.1. Example of Global Maxima Operation

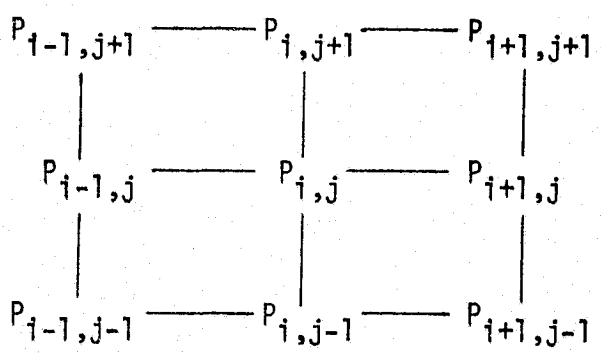


FIGURE 4.2. Neighborhood of a Tse Element

TABLE 9

LOCAL MAXIMA/MINIMA ROUTINE

LMMR (Input digitized image is assumed to be in A accumulator)

NUMBER	INSTRUCTION	TYPE	COMMENTS
1	CONT N	CONTROL	N = NUMBER OF SIGNIFICANT tses
2	SLDR A	tse	ELEMENTS OF A ARE SLID TO THE RIGHT AND STORED IN B ACCUMULATOR
3	SHFA 1	tse	
4	SHFB 1	tse	
5	MOVE B	tse	
6	DCR	CONTROL	
7	JNZ 2	CONTROL	B ACCUMULATOR NOW CONTAINS THE CONTENTS OF A ACCUMULATOR WITH THE i-1,j ELEMENTS OF EACH A tse IN THE i,j POSITION
8	THGE	tse	SUBROUTINE WHICH COMPARES A AND B ACCUMULATORS AND GENERATES THE A > B tse IN THE ALU OUTPUT LATCH
9	MOVE R1	tse	
10	CONT N	CONTROL	
11	SLDD B	tse	ELEMENTS OF B ARE SLID DOWN AND STORED IN B ACCUMULATOR
12	SHFB 1	tse	
13	MOVE B	tse	
14	DCR	CONTROL	
15	JNZ 10	CONTROL	i-1,j+1 ELEMENTS IN THE i,j POSITION

TABLE 9 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
16	THGE	tse	
17	MOVE R4	tse	
18	LAND R1, R4, R1	tse	
19	CONT N	CONTROL	
20	SLDU B	tse	ELEMENTS OF B ARE SLID UP AND STORED IN B ACCUMULATOR
21	SHFB 1	tse	
22	MOVE B		
23	DCR	CONTROL	
24	JNZ 19	CONTROL	
25	CONT N	CONTROL	
26	SLDU B	tse	
27	SHFB 1	tse	
28	MOVE B		
29	DCR	CONTROL	
30	JNZ 23	CONTROL	i-1,j-1 ELEMENTS IN THE i,j POSITION
31	THGE	tse	
32	MOVE R4	tse	
33	LAND R1, R4, R1		
34	CONT N	CONTROL	
35	SLDL B	tse	
36	SHFB 1	tse	
37	MOVE B	tse	
38	DCR	CONTROL	
39	JNZ 35	CONTROL	
40	CONT N	CONTROL	
41	SLDL B	tse	

TABLE 9 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
42	SHFB 1	tse	
43	MOVE B	tse	
44	DCR	CONTROL	
45	JNZ 41	CONTROL	i+1,j-1 ELEMENTS IN THE i,j POSITION
46	THGE	tse	
47	MOVE R4	tse	
48	LAND R1, R4, R1	tse	
49	CONT N	CONTROL	
50	SLDD B	tse	
51	SHFB 1	tse	
52	MOVE B	tse	
53	DCR	CONTROL	
54	JNZ 50	CONTROL	i+1,j ELEMENTS IN THE i,j POSITION FOR THE FIRST PASS, THEN i+1, j+1 FOR THE SECOND PASS
55	THGE	tse	
56	MOVE R4	tse	
57	LAND R1, R4, R1	tse	
58	GO TO 49	CONTROL	FOR PASS TWO
59	CONT N	CONTROL	
60	SLDR B	tse	
61	SHFB 1	tse	
62	MOVE B	tse	
63	DCR	CONTROL	
64	JNZ 60	CONTROL	i,j+1 ELEMENTS IN THE i,j POSITION
65	THGE	tse	

TABLE 9 (continued)

NUMBER	INSTRUCTION	TYPE	COMMENTS
66	MOVE R4	tse	
67	LAND R1, R4, R1	tse	
68	CONT N	CONTROL	
69	SLDU B	tse	
70	SHFB 1	tse	
71	MOVE B	tse	
72	DCR	CONTROL	
73	JNZ 69	CONTROL	
74	CONT N	CONTROL	
75	SLDU B	tse	
76	SHFB 1	tse	
77	MOVE B	tse	
78	DCR	CONTROL	
79	JNZ 75	CONTROL	i,j-1 ELEMENTS IN THE i,j POSITION
80	THGE	tse	
81	MOVE R4	tse	
82	LAND R1, R4, R1	tse	R1 CONTAINS THE LOCAL MAXIMA tse
83	RET	CONTROL	

For: Local Maxima (plateau regions allowed), use CMGE
Local Maxima (plateau regions disallowed), use CMPG
Local Minima (plateau regions allowed), use CMLE
Local Minima (plateau regions disallowed), use CMPL

3	5	4	4		0	1	1	1	1	0	0	0	1	1	0	0
5	7	6	4	=	1	1	1	1	0	1	1	1	1	1	0	0
6	5	7	5		1	1	1	1	1	0	1	0	0	1	1	1
7	6	5	3		1	1	1	0	1	1	0	1	1	0	1	1
X	1	0	1		X	0	1	1	X	1	0	0	X	1	1	0
X	1	0	0		X	1	1	1	X	0	1	1	X	1	1	0
X	0	1	0		X	1	1	1	X	1	0	1	X	0	1	1
X	0	0	0		X	1	1	1	X	1	1	0	X	1	0	1
		(R1)							i-1, j neighbors							
X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X
X	1	0	0		0	1	1	1	1	0	0	0	1	1	0	0
X	0	1	0		1	1	1	1	0	1	1	0	1	1	0	0
X	0	0	0		1	1	1	1	1	0	1	0	0	1	1	1
		(R1)							i-1, j+1 neighbors							
X	X	X	X		X	1	1	1	X	0	1	1	X	1	1	0
X	1	0	0		X	1	1	1	X	1	0	1	X	0	1	1
X	0	1	0		X	1	1	1	X	1	1	0	X	1	0	1
X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X
		(R1)							i-1, j-1 neighbors							

X indicates points which are not eligible to be local maxima, but are eligible to be neighborhood points.

FIGURE 4.3. Example of Local Maxima Operation

X	X	X	X		1	1	1	X		1	1	0	X		1	0	0	X
X	1	0	X		1	1	1	X		0	1	0	X		1	1	1	X
X	0	1	X		1	1	0	X		1	1	1	X		0	1	1	X
X	X	X	X		X	X	X	X		X	X	X	X		X	X	X	X
	(R1)									i+1, j-1 neighbors								
X	X	X	X		1	1	1	X		0	0	0	X		1	0	0	X
X	1	0	X		1	1	1	X		1	1	0	X		1	0	0	X
X	0	1	X		1	1	1	X		0	1	0	X		1	1	1	X
X	X	X	X		1	1	0	X		1	0	1	X		0	1	1	X
	(R1)									i+1, j neighbors								
X	X	X	X		X	X	X	X		X	X	X	X		X	X	X	X
X	1	0	X		1	1	1	X		0	0	0	X		1	0	0	X
X	0	1	X		1	1	1	X		1	1	0	X		1	0	0	X
X	X	X	X		1	1	1	X		0	1	0	X		1	1	1	X
	(R1)									i+1, j+1 neighbors								
X	X	X	X		X	X	X	X		X	X	X	X		X	X	X	X
X	1	0	X		0	1	1	1		1	0	0	0		1	1	0	0
X	0	1	X		1	1	1	1		0	1	1	0		1	1	0	0
X	X	X	X		1	1	1	1		1	0	1	0		0	1	1	1
	(R1)									i, j+1 neighbors								

X - filler points

FIGURE 4.3. (continued)

X	X	X	X		1	1	1	1		0	1	1	0		1	1	0	0
X	1	0	X		1	1	1	1		1	0	1	0		0	1	1	1
X	0	1	X		1	1	1	0		1	1	0	1		1	0	1	1
X	X	X	X		X	X	X	X		X	X	X	X		X	X	X	X
	(R1)									i, j-1 neighbor								

Local Maxima tse

X - filler points

FIGURE 4.3. (continued)

useful for analysis purposes. Partial derivatives may be employed to distinguish these boundaries. For the tse computer, a finite difference approach is used to approximate the partial derivatives. The first partial derivative with respect to the x coordinate may be approximated by either a forward, backward, or central difference. Utilization of the central difference allows less error and does not require significantly more computation effort with respect to the tse computer. The central difference approximation to the first partial derivative with respect to the x coordinate is as follows [8]:

$$\left.\frac{\partial F}{\partial x}\right|_{x_o,y_o} \approx \frac{F(x+h,y) - F(x-h,y)}{2h} .$$

In the tse computer case, h is assumed to be equal to one. The central difference approximation with respect to the y coordinate is analogous to the above expression. The division by 2 is accomplished by a shifting of the accumulator register to the right one position. In the tse computer, at the present stage of development, this shifting amounts to the loss of the least significant tse of the difference. However, when the results of the partial derivative operation are used to compute the gradient of the image, the division by two alleviates any overflow problems that may occur. An example of the partial derivative operation is shown in Figure 4.4.

A routine to compute the first partial derivative with respect to the x coordinate is listed in Table 10. A computation of the partial derivative with respect to the y coordinate may be achieved by

1	2	4	5	4	6	5	4		0	0	1	1	1	1	1	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	1	0
1	2	3	4	4	7	6	5		0	0	0	1	1	1	1	1	0	1	1	0	0	1	1	0	1	0	1	0	0	1	0	1
2	3	4	4	7	7	6	4		0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	1	0	0	1	1	0	0
3	4	5	6	5	5	5	4		0	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	1	0	1	0	1	1	1	0
3	4	7	6	5	5	6	5	=	0	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	1	0	1	0	1	1	0	1
2	3	6	7	6	5	5	3		0	0	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	1	0	1	0	1	1	1
2	2	4	7	6	6	5	4		0	0	1	1	1	1	1	1	1	1	0	1	1	1	0	0	0	0	0	1	0	0	1	0
3	3	4	3	5	6	5	4		0	0	1	0	1	1	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	0	1	0

First partial derivative with respect to x of the image

X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	1	1	0	1	1	-1	X		X	0	0	0	0	0	1	X	X	0	0	0	0	0	0	X	X	1	1	0	1	1	1	X
X	1	0	1	1	0	-1	X		X	0	0	0	0	0	1	X	X	0	0	0	0	0	0	X	X	1	0	1	1	0	1	X
X	1	1	0	0	0	0	X		X	0	0	0	0	0	0	X	X	0	0	0	0	0	0	X	X	1	1	0	0	0	0	X
X	2	1	-1	0	0	0	X	=	X	0	0	1	0	0	0	X	X	1	0	0	0	0	0	X	X	0	1	1	0	0	0	X
X	2	2	0	-1	0	-1	X		X	0	0	0	1	0	1	X	X	1	1	0	0	0	0	X	X	0	0	0	1	0	1	X
X	1	2	1	0	0	-1	X		X	0	0	0	0	0	1	X	X	0	1	0	0	0	0	X	X	1	0	1	0	0	1	X
X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Sign tse

X - filler points

FIGURE 4.4. Example of Partial Derivative Operation

TABLE 10

FIRST PARTIAL DERIVATIVE ROUTINE

PDEX (Assume the tse data word is in the A accumulator)

NUMBER	INSTRUCTION	TYPE	COMMENTS
1	CLER	tse	
2	CONT N+1	CONTROL	
3	SLDR A	tse	
4	SHFB 1	tse	
5	MOVE B	tse	
6	DCR	CONTROL	
7	JNZ 3	CONTROL	F(x-h) IS IN B ACCUMULATOR
8	CONT N+1	CONTROL	
9	SLDL A	tse	
10	SHFA 1	tse	
11	CLRC S_A	tse	
12	MOVE A	tse	
13	DCR	CONTROL	
14	JNZ 9	CONTROL	F(x+h) IS IN A ACCUMULATOR
15	SUBT N+1	tse	SUBTRACT ROUTINE
16	SHFA 1	tse	
17	CLRC A_{MS}	tse	

The above routine computes $\frac{\partial F}{\partial x}\Big|_{x_o,y_o}$. For $\frac{\partial F}{\partial y}\Big|_{x_o,y_o}$, replace instructions 3 and 9 as follows:

3' SLDU A

9' SLDD A

replacing the slides right and left in the routine of Table 10 by slide up and down. The routine of Table 10 requires 2.6 seconds per iteration on machine 1, and 1.025 seconds per iteration on machine 2.

The second partial derivative may be computed by the application of the following approximation method [8]:

$$\left.\frac{\partial^2 F}{\partial x^2}\right|_{x_o,y_o} \approx \frac{F(x+h,y) + F(x-h,y) - 2F(x,y)}{h^2} \quad .$$

This method is shown in the second partial derivative routine of Table 11. The routine of Table 11 requires 5.2 seconds per iteration on machine 1 and 2.5 seconds per iteration on machine 2.

Gradient. The magnitude of the gradient of an image is defined as [9]:

$$|\,\text{grad}\,| = \sqrt{\left(\frac{\partial F}{\partial x}\right)^2 + \left(\frac{\partial F}{\partial y}\right)^2} \quad .$$

As with partial derivatives, the gradient is useful in defining boundaries, and developing topological descriptions of the properties of an image. Figure 4.5 depicts the gradient operation as applied to an image. All operations that are needed to compute the gradient of an image have been developed. These operations are combined in Table 12 as a program to compute the gradient. The complete gradient computation requires 44.2 seconds per iteration on machine 1 and 13.8 seconds per iteration on machine 2.

TABLE 11

SECOND PARTIAL DERIVATIVE ROUTINE

SPDX (Assume the tse data word is in A accumulator and memory)

NUMBER	INSTRUCTION	TYPE	COMMENTS
1	CLER	tse	
2	CONT N+1	CONTROL	
3	SLDR A	tse	
4	SHIA 1	tse	
5	MOVE B	tse	
6	DCR	CONTROL	
7	JNZ 3	CONTROL	
8	CONT N+1	CONTROL	
9	SLDL A	tse	
10	SHIA 1	tse	
11	CLRC S_{AS}	tse	
12	MOVE A	tse	
13	DCR	CONTROL	
14	JNZ 9	CONTROL	
15	ADDU N+1	tse	ADD ROUTINE
16	LODB M	tse	LOAD THE tse DATA WORD INTO B ACCUMULATOR FROM MEMORY, ASSUMING A 12 tse ACCUMULATOR
17	SHFB 11	tse	
18	SUBT N+1	tse	

For $\left.\frac{\partial^2 F}{\partial y^2}\right|_{x_o, y_o}$, substitute for instructions 3 and 9 as follows:

3' SLDU A

9' SLDD A

Original Image

1	2	4	5	4	6	5	4		0	0	1	1	1	1	1	1	0	1	0	0	0	1	0	0	1	0	0	1	0
1	2	3	4	4	7	6	5		0	0	0	1	1	1	1	1	0	1	1	0	0	1	1	0	1	0	1	0	0	1	0	1
2	3	4	4	7	7	6	4		0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	1	0	0	1	1	0	0
3	4	5	6	5	5	5	4		0	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	1	0	1	0	1	1	1	0
3	4	7	6	5	5	6	5	=	0	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	1	0	1	0	1	1	0	1
2	3	6	7	6	5	5	4		0	0	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	1	0	1	0	1	1	1
2	2	4	7	6	6	5	4		0	0	1	1	1	1	1	1	1	1	0	1	1	1	0	0	0	0	0	1	0	0	1	0
3	3	4	3	5	6	5	4		0	0	1	0	1	1	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	0	1	0

Gradient of the Image

X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	1	1	0	2	1	1	X		X	0	0	0	0	0	0	X	X	0	0	0	1	0	0	X	X	1	1	0	0	1	1	X
X	1	1	2	1	1	1	X		X	0	0	0	0	0	0	X	X	0	0	1	0	0	0	X	X	1	1	0	1	1	1	X
X	1	2	1	1	1	0	X		X	0	0	0	0	0	0	X	X	0	1	0	0	0	0	X	X	1	0	1	1	1	0	X
X	2	1	1	0	0	0	X	=	X	0	0	0	0	0	0	X	X	1	0	0	0	0	0	X	X	0	1	1	0	0	0	X
X	2	2	0	1	0	1	X		X	0	0	0	0	0	0	X	X	1	1	0	0	0	0	X	X	0	0	0	1	0	1	X
X	1	2	2	0	0	1	X		X	0	0	0	0	0	0	X	X	0	1	1	0	0	0	X	X	1	0	0	0	0	1	X
X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

(Note Boundary)

X - filler points

FIGURE 4.5. Example of Gradient Operation

TABLE 12

ROUTINE TO COMPUTE THE GRADIENT OF AN IMAGE

GRAD

NUMBER	INSTRUCTION	TYPE	COMMENTS
1	CLER	tse	
2	CALL PDEX	CONTROL	FIRST PARTIAL DERIVATIVE wrt x ROUTINE
3	CALL SQAR	CONTROL	SQUARING ROUTINE
4	STOA Z	tse	STORES THE CONTENTS OF A ACCUMULATOR IN LOCATION Z
5	LODA M	tse	RELOADS THE tse DATA WORD INTO THE A ACCUMULATOR
6	CALL PDEY	tse	FIRST PARTIAL DERIVATIVE wrt y ROUTINE
7	CALL SQAR	tse	
8	LODB Z	tse	
9	ADDU N	tse	ADD ROUTINE
10	CALL SQRT	tse	SQUARE ROOT ROUTINE

The gradient of the image is available in the A accumulator.

The above operations are only a few of the many that are possible due to the flexibility of the tse processor. As the tse computer proceeds in its development, more processing power and improved hardware will enable the tse computer to handle more complex tasks at a greater rate of speed.

CHAPTER V

CONTROL OF THE TSE COMPUTER

The control unit of the tse computer is assigned the task of sequencing the stored instructions in the proper order, selecting the correct information source and destination, and providing the appropriate processing path through the tse processor. One control organization is shown in Figure 5.1. Tse processor control is achieved by the selection of a "control word" which is output from the control unit and interfaced with the tse processor. Each bit of the control word is used to activate or deactivate one or more of the "reformatters" in the tse processor. The reformatters may be termed switched-output devices, in that the electroluminescense may be turned on or off by a conventional logic signal. Thus, proper data paths are chosen by activating the reformatters which lie in the specific processing paths chosen by the instruction. Of course, the control unit must observe the timing constraints dictated by the tse logic device propagation delay.

Conditional control is accomplished by inputting to the control unit the status information provided by the "contractor" outputs in the tse processor. The tse status information is stored in a register which is available for input to the control unit.

Control implementation may be achieved by utilizing any of the small computers now available. Since the instruction cycle time of the tse computer is significantly greater than that of a conventional

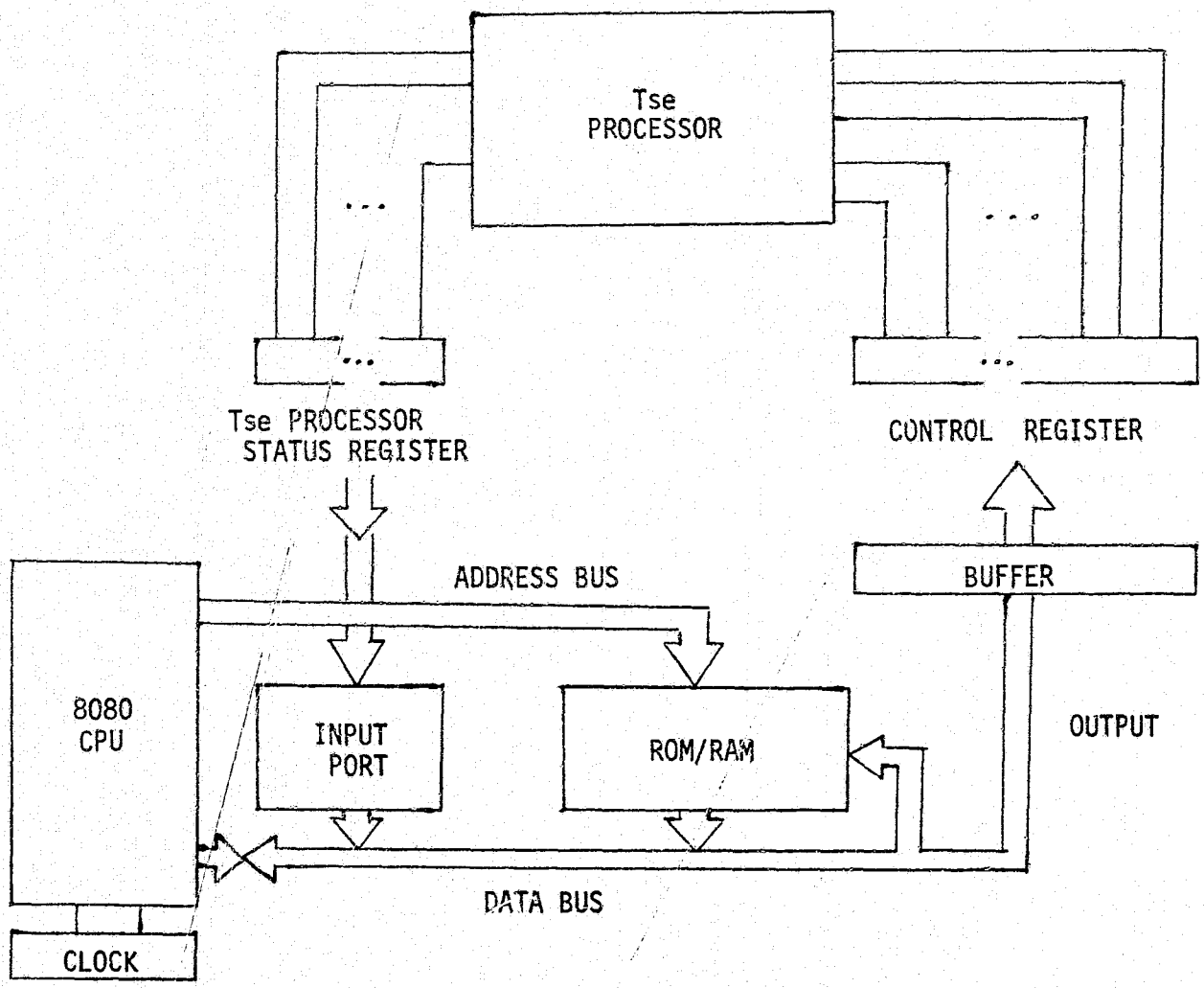


FIGURE 5.1. Tse Processor Control

computer. A control unit organized around a microprocessor is sufficiently fast for the tse processor. The tse computer routines of Chapter III and IV were written using the instruction set of the Intel 8080 microprocessor as the control commands. Thus, the organization of Figure 5.1 shows the control unit based upon the 8080 CPU. However, any of the microprocessors available could be used.

The control unit functions under the management of a "System Monitor Program (SMP)." When the control unit receives an instruction for execution, the SMP determines whether it is a control or tse instruction. If the command received is a control instruction, it is executed without affecting the "control word." However, if the command is a tse instruction, the SMP directs the output of the proper control word to the tse processor, and maintains the control word for the minimum amount of time necessary for the tse data to propagate along the proper paths. Figure 5.2 illustrates the operation of the SMP.

A control unit of this type has the advantages of being small, lightweight, and having low power consumption relative to the tse processor.

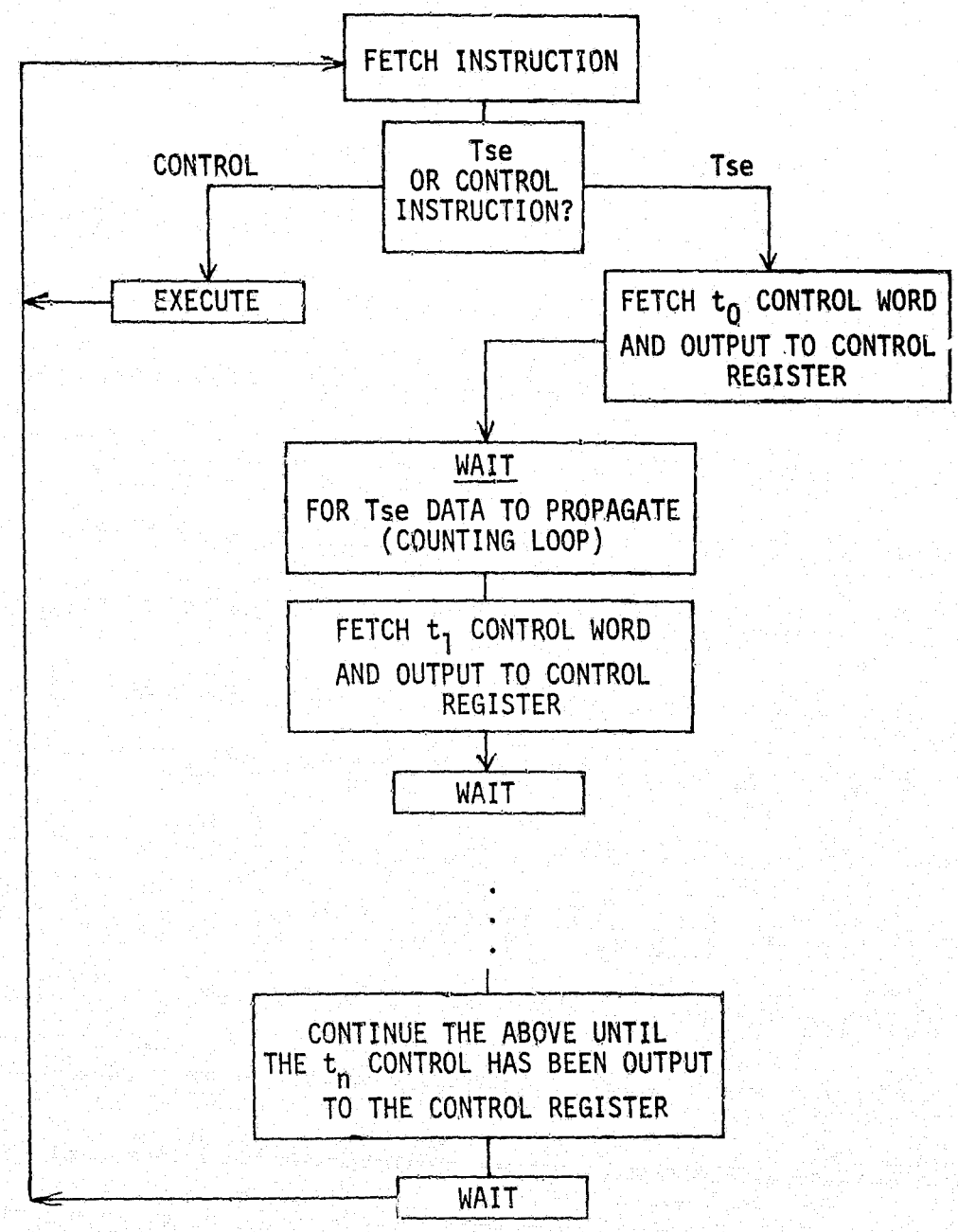


FIGURE 5.2. Function of the Tse Processor Control Unit

CHAPTER VI

CONCLUSION

Tse processing of pictorial information has been shown to be an efficient method relative to conventional processing. By utilizing the ALU organization of Figure 3.13 (page 44) with the CPU organization of Figure 3.3 (page 21), the desired topological information may be extracted speedily and efficiently. With this organization, the processing flexibility to develop other tse or arithmetic operations has been provided. Tse processing capability is limited only by the cleverness of the programmer.

Since the tse hardware is in the early development stage, hardware specifications on speed, power requirement, etc., will improve with time. Consequently, the microprocessor control must also be faster. As these improvements are realized, the tse computer will gain an even greater margin on conventional machines in image processing tasks.

LIST OF REFERENCES

LIST OF REFERENCES

1. Schaefer, D.H., and Strong, J.P., Tse Computers. X-943-75-14, Goddard Space Flight Center, 1975.

2. Kostopoulos, G.K., Digital Engineering. New York: Wiley-Interscience, 1975, pp. 309-315.

3. Unger, S.H., "A Computer Oriented Toward Spatial Problems," Proceedings of IRE (October 1958), Vol. 46, pp. 1744-1750.

4. Kruse, B., "A Parallel Picture Processing Machine," IEEE Transactions on Computers (December 1973), Vol. C-22, No. 12, pp. 1075-1086.

5. Lewin, D., Theory and Design of Digital Computers. London: Thomas Nelson and Sons LTD, 1972, pp. 307-311.

6. Edelstein, L.A., "'Picture Logic' For 'Bacchus' a Fourth-Generation Computer," The Computer Journal (July 1963), Vol. 6, pp. 144-153.

7. Levy, H.H., "Earth-Resources Technology Satellite: NASA Data-Processing Facility," Geoscience Electronics (October 1970), Vol. 8, No. 4, pp. 348-352.

8. Brainerd, W.S., and Landweber, L.H., Theory of Computation. New York: Wiley-Interscience, 1974, p. 216.

9. Kreyszig, E., Advanced Engineering Mathematics. New York: Wiley-Interscience, 1972, pp. 307-311.

APPENDIX

APPENDIX

TSE COMPUTER INSTRUCTION SET

One-byte Instructions

CLER: Clears all ALU latches.

CLRA: Clears the "A" accumulator.

CLRB: Clears the "B" accumulator.

TRNS: Transfers the contents of ALU carry latch 2 to the ALU output latch.

CMPL: Compares the contents of the least significant tse positions of the A and B accumulators. The resulting tse (A < B) is stored in ALU carry latch 2.

CMPG: Compares the contents of the least significant tse positions of the A and B accumulators. The resulting tse (A > B) is stored in ALU carry latch 2.

TCMA: Microprogrammed instruction to replace the contents of the "A" accumulator with its two's complement.

TCMB: Microprogrammed instruction to replace the contents of the "B" accumulator with its two's complement.

SQUR: Microprogrammed instruction to compute the square of a 6-tse data word.

SQRT: Microprogrammed instruction to compute the square root of a 12-tse data word.

Two-byte Instructions

SHFA n: Shift the "A" accumulator right n positions. The magnitude only is affected.

SHFB n: Same as SHFA n, except with "B" accumulator.

SHIA n: Shift the "A" accumulator right n positions. All tses, including the sign tse, are affected.

SHIB n: Same as SHIA n, except with "B" accumulator.

CONT n: Sets up a control instruction subroutine which establishes a counting loop starting at n.

CLRC X: Clears location X. Permissable X are: R1-R10 latches, S_A, S_B, A_{MS}, B_{MS}, B_{LS}, B_{LS+1}, ALU carry latches 1 and 2, and the ALU output latch.

MOVE X: Transfers the contents of the ALU output latch to the designated location X.

MCME X: Transfers the complement of the ALU output latch to the designated location X.

LDCA X: Load the contents of location X into the ALU carry latch 2.

CHEK N: Sets up a subroutine of control instructions which inputs the tse status register, checks that bit position which represents the output of the ALU "contractor", and jumps to the instruction of location N when that bit is logical 0. Otherwise, the next instruction is executed.

CHAK N: Same as CHEK N, except program control jumps on logical 1.

Three-byte Instructions

TRAN X,Y: Transfers the contents of location X to location Y.

TRNN X,Y: Transfers the complement of the contents of location X to location Y.

ADSU X,Y: A single iteration add/subtract instruction. Generates a sum tse in the ALU output latch, and a carry tse in ALU carry latch 2.

Four-byte Instructions

LAND X,Y,Z: Forms the intersection (AND) of the contents of X and Y locations and stores the result in location Z.

LORE X,Y,Z: Forms the union (OR) of the contents of X and Y locations, and stores the result in location Z.

EXOR X,Y,Z: Forms the logical exclusive-or of X and Y locations, and stores the result in location Z.

VITA

James R. Jones was born in Nashville, Tennessee, on January 31, 1947. He attended elementary schools in Old Hickory, Tennessee. In 1965, he graduated with honors from Castle Heights Military Academy in Lebanon, Tennessee. Upon graduation, he entered North Carolina State University in Raleigh, North Carolina, majoring in Architecture.

In 1967, he joined the United States Air Force where he served as a radio intercept analyst in West Germany. While in the Air Force, he worked part-time selling and servicing audio equipment, thus gaining an interest in electronics.

Upon discharge from the Air Force, he entered the University of Tennessee and received a Bachelor of Science degree in Electrical Engineering in 1974. He entered the University of Tennessee Graduate School that same year and earned the Master of Science degree in Electrical Engineering with a major in Digital Systems Engineering and a minor in Control Systems Engineering in June, 1975.

In June, 1975, he joined the geophysics staff of the Region Exploration division of the Shell Oil Company in New Orleans, Louisiana.